

Řízení a monitoring hardwarových modulů

Jaroslav Gargulák

Bakalářská práce
2016



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jaroslav Gargulák**
Osobní číslo: **A13744**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**
Forma studia: **kombinovaná**

Téma práce: **Řízení a monitoring hardwarových modulů**
Téma anglicky: **The Control and Monitoring of Hardware Modules**

Zásady pro vypracování:

1. Vytvořte aplikaci komunikující s hardwarovými moduly po RS485.
2. Umožněte jejich konfiguraci, ovládání a aktualizaci firmwaru.
3. Uchovejte data z připojených modulů a umožněte jejich zobrazení v grafu.
4. K dispozici bude režim naslouchání pro analýzu zpráv na sběrnici.
5. Implementujte podporu více jazyků - čeština, angličtina + příprava na další.
6. Popište použité technologie.

Rozsah bakalářské práce: -
Rozsah příloh: -
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. TROELSEN, Andrew W. C# a .NET 2.0 profesionálně. Vyd. 1. Brno: Zoner Press, 2006, 1197 s. Encyklopedie Zoner Press. ISBN 80-86815-42-0.
2. ALBAHARI, Joseph a Ben ALBAHARI. C# 3.0 in a nutshell. 3rd ed. Sebastopol, Calif.: O'Reilly, 2007, xviii, 838 p. In a nutshell (O'Reilly & Associates). ISBN 0596527578.
3. HRUŠKA, František. Technické prostředky integrované automatizace [elektronická skripta]. Univerzita Tomáše Bati ve Zlíně, 2015 [cit. 2016-01-19]. ISBN 978-80-7454-234-3. Dostupné z: <http://digilib.k.utb.cz//handle/10563/18664>
4. GENERAL: INTEL HEX FILE FORMAT. ARMKEIL: Microcontroller tools [online]. ?2005-2015 [cit. 2016-01-19]. Dostupné z: <http://www.keil.com/support/docs/1584.htm>
5. FT232R USB UART IC Datasheet [online]. [cit. 2016-01-20]. Dostupné z: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

Vedoucí bakalářské práce: **doc. Ing. Martin Sysel, Ph.D.**
Ústav počítačových a komunikačních systémů
Datum zadání bakalářské práce: **19. února 2016**
Termín odevzdání bakalářské práce: **27. května 2016**

Ve Zlíně dne 19. února 2016


doc. Mgr. Milan Adámek, Ph.D.
děkan




prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Jméno, příjmení: Jaroslav Gargulák

Název bakalářské/diplomové práce: Řízení a monitoring hardwarových modulů

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

11. 5. 2016

.....
podpis diplomanta

ABSTRAKT

Bakalářská práce se zabývá vývojem aplikace v jazyce C#, která pomocí sériového rozhraní COM komunikuje s hardwarovými moduly. Aplikace umožňuje zobrazení stavu modulů, jejich řízení a kalibrace. V teoretické části jsou popsány použité technologie a jsou předvedeny některé dostupné utility s podobnou funkčností. V praktické části je popsáno propojení s počítačem, komunikační protokol, hlavní třídy aplikace a možnosti jednotlivých modulů.

Klíčová slova: .NET framework, COM port, hardwarové moduly

ABSTRACT

This bachelor theses focuses on the development of an application in C# language which communicates with the hardware modules by COM interface. The application enables to display the state of modules, their control and calibration. The used technologies and also some accessible utilities are outlined in the theoretical part. Then the practical part describes the connection with pc, the communication protocol, the main classes of application and the possibilities of the individual modules.

Keywords: .NET framework, COM interface, hardware modules

Poděkování

Za mnoho cenných rad při tvorbě této práce děkuji panu doktoru Martinovi Syslovi a také celému kolektivu kantorů Fakulty aplikované informatiky za získané vědomosti. Za podporu při studiu patří díky rodině a přátelům. A v neposlední řadě patří poděkování i mému zaměstnavateli za shovívavost v době mého studia.

Moto

„The last good thing written in C was Franz Schubert's Symphony Number 9.“

Erwin Dieterich, programmer

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 HARDWAROVÉ MODULY	11
1.1 MODUL DIGITÁLNÍCH VSTUPŮ A VÝSTUPŮ.....	11
1.2 MODUL ŘÍZENÍ MOTORŮ.....	12
2 TECHNOLOGIE POUŽITÉ PRO VÝVOJ	14
2.1 SÉRIOVÉ ROZHRAŇÍ COM.....	14
2.1.1 Standard RS-232C.....	14
2.1.2 Standard RS485.....	15
2.1.2.1 Délka vedení.....	15
2.1.2.2 Elektrické zapojení.....	15
2.1.3 Standard RS422.....	16
2.1.4 Srovnání standardů RS232C, RS422 a RS485.....	17
2.1.5 Asynchronní přenos dat.....	17
2.2 PŘEHLED DOSTUPNÝCH UTILIT PRO OBSLUHU COM PORTU.....	18
2.2.1 Putty.....	18
2.2.2 Hercules.....	18
2.2.3 Terminal.....	19
2.3 FRAMEWORK .NET.....	21
2.3.1 Společný runtime jazyků CLR.....	21
2.3.2 Společný systém typů CTS.....	22
2.3.2.1 Typ třída.....	23
2.3.2.2 Typ struktura.....	23
2.3.2.3 Typ rozhraní.....	23
2.3.2.4 Typ výčet.....	23
2.3.2.5 Typ delegát.....	23
2.3.3 Společná specifikace jazyků CLS.....	24
2.3.4 Vývoj a přehled verzí.....	24
2.4 PROGRAMOVACÍ JAZYK C#.....	24
2.4.1 Použité prostředky pro práci s vlákny.....	25
2.4.1.1 Vytvoření vlákna.....	25
2.4.1.2 Synchronizační prostředek AutoResetEvent.....	25
2.5 MICROSOFT VISUAL STUDIO COMMUNITY.....	26
II PRAKTICKÁ ČÁST	27
3 STANOVENÉ CÍLE PRAKTICKÉ ČÁSTI	28
4 PROPOJENÍ MODULŮ S POČÍTAČEM	29
4.1 PŘEVODNÍK PREMIUMCORD USB2.0 NA RS485.....	29
4.2 PROPOJENÍ S POČÍTAČEM.....	30
4.3 KOMUNIKAČNÍ PROTOKOL.....	30
4.3.1 Konfigurace portu.....	30
4.3.2 Struktura zprávy.....	31
5 KOMUNIKAČNÍ JÁDRO	32

5.1	IMPLEMENTACE JÁDRA	32
5.2	IMPLEMENTACE PROTOKOLU	32
5.2.1	Chování přijímacího vlákna v režimu naslouchání	33
6	SPRÁVA KOLEKCE MODULŮ	34
6.1	RODIČOVSKÁ TŘÍDA MODULŮ	34
6.2	RODIČOVSKÁ TŘÍDA MANAŽERU KARET	34
6.3	FUNKCE MANAŽERU MODULŮ	36
6.4	JEDINEČNÝ KLÍČ MODULU	36
7	MODUL DIGITÁLNÍCH VSTUPŮ A VÝSTUPŮ	37
7.1	PODPOROVANÉ PŘÍKAZY	37
7.1.1	Detekce modulu	37
7.1.2	Čtení stavu vstupů	38
7.1.3	Nastavení výstupů	38
7.2	UŽIVATELSKÝ FORMULÁŘ	39
8	MODUL ŘÍZENÍ MOTORŮ	40
8.1	KONFIGURAČNÍ REGISTRY	41
8.1.1	Registry pro konfiguraci krokového motoru	41
8.1.2	Registry pro konfiguraci DC motoru	41
8.1.3	Registry pro konfiguraci modulu	41
8.1.4	Stavové a ovládací registry	42
8.2	PODPOROVANÉ PŘÍKAZY	42
8.2.1	Detekce modulu	42
8.2.2	Zápis a čtení registru	43
8.2.3	Příkaz zastavení	44
8.2.4	Příkaz přepnutí do režimu aktualizace	44
8.2.5	Příkaz aktualizace - zadání počáteční adresy aktualizované stránky	44
8.2.6	Příkaz aktualizace – zápis dat	45
8.2.7	Příkaz aktualizace – čtení dat	45
8.3	AKTUALIZACE FIRMWAREU	46
8.3.1	Formát Intel HEX	46
8.3.2	Implementace	47
8.4	UŽIVATELSKÝ FORMULÁŘ	49
9	GRAFY	50
10	HLAVNÍ FORMULÁŘ APLIKACE	51
10.1	LADÍCÍ KONZOLE	51
	ZÁVĚR	53
	SEZNAM POUŽITÉ LITERATURY	54
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	56
	SEZNAM OBRÁZKŮ	57
	SEZNAM TABULEK	59
	SEZNAM PŘÍLOH	60

ÚVOD

Při vývoji hardwaru a softwaru se musí vývojáři často poprat s různorodými požadavky zákazníků, kteří navíc mnohdy rádi mění své požadavky v době, když už je hardware či software vyvinut. Při vývoji je nezbytné myslet dopředu a připravit vše s jistými rezervami, které by se mohly do budoucna hodit k dalšímu rozšiřování. V případě hardwaru a jeho firmwaru se ideálním řešením nabízí udělat maximum parametrů konfigurovatelných tak, aby nebylo nutné do již funkčního firmwaru či hardwaru zasahovat. Obvyklá cesta, jak parametry hardwarových jednotek, jež jsou určeny k montáži do rozvaděče konfigurovat, je pomocí vestavěného řídicího operátorského počítače, nebo pomocí externího konfiguračního softwaru. Výhoda konfigurace pomocí vestavěného řídicího počítače spočívá v tom, že je vždy na místě zařízení k dispozici. Ovšem v případě nízkého zabezpečení řídicího softwaru, dává jistou možnost neoprávněným osobám změnit nastavení kritických hodnot systému. Konfigurace externím programem, je v jistých směrech lepší, než první uvedená možnost. Externí program má ve většině případů k dispozici pouze vyškolený servisní pracovník a obvykle dovoluje konfigurovat hardwarové moduly i mimo zařízení. To znamená, že po vyjmutí modulu ze zařízení, je možno modul bezpečně otestovat, aniž by byla potřeba cokoli na zařízení spouštět.

Filosofii externího softwaru pro konfiguraci hardwarových modulů bude ctít i program vytvořený v rámci této bakalářské práce. Bude sloužit především k vnitrofiremním účelům a bude se snažit nabídnout efektivní nástroj k testování a konfiguraci modulů ve výrobě. Výstupní kontrola ve výrobě je vždy velmi důležitý krok ke spokojenosti zákazníků, a proto nástroj, který bude spolehlivým pomocníkem a který ušetří čas, bude jistě přínosem.

Program přidá i možnost řízení modulů pro účely vývoje prvotních prototypů zařízení. Při vývoji nového zařízení je obvykle nutné otestovat nejprve fyzické vlastnosti, kde je ovšem zapotřebí alespoň testovací verze řídicího softwaru. Tudíž pro jednoduché testy nebude nutné testovací software připravovat a bude možné použít tento program.

Program poslouží jako nástroj při vývoji, výrobě nových řídicích modulů a diagnostice. Vzhledem k tomu, že má především usnadňovat práci a umožnit i laikovi jeho použití, je pojmenován anglickým spojením EasyCard.

I. TEORETICKÁ ČÁST

1 HARDWAROVÉ MODULY

Moduly, ke kterým bude v rámci této práce vyvíjen obslužný software, jsou vyráběny firmou SVCS Process Innovation. V současné době jsou k dispozici následující moduly:

- Digitální vstupy a výstupy pro snímání a spínání 24V signálů.
- Analogové vstupy a výstupy pro měření a nastavení signálů 0-5V.
- Řízení stejnosměrných i krokových motorů s podporou odečtu polohy pomocí inkrementálních čidel.
- Měření teploty pomocí termočlánků.
- Speciálně zaměřené moduly.

Aplikace bude v první fázi podporovat modul digitálních vstupů/výstupů a modul řízení motorů. Modul řízení motorů disponuje možností aktualizace firmware přes RS485, proto software bude tuto funkčnost také umožňovat.

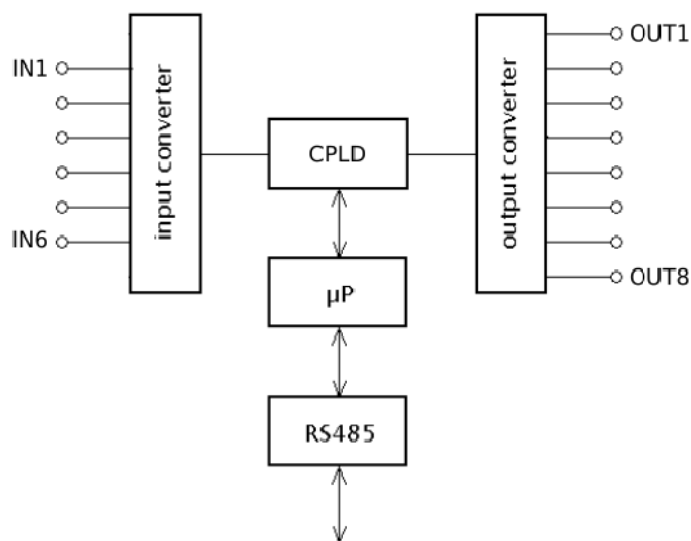
1.1 Modul digitálních vstupů a výstupů

Modul je určen pro generování a snímání 24V signálů. Obsahuje celkem šest vstupů a osm výstupů, jejichž stavy jsou signalizovány pomocí LED. Základem modulu je mikroprocesor PIC z rodiny 16F a je možné jej ovládat pomocí rozhraní RS485. Na jedné sběrnici je možné mít celkem 6 těchto modulů, přičemž omezení je dáno maximální adresou, která se nastavuje pomocí zkratovacích propojek.

Důležité parametry modulu shrnuje následující tabulka.

Parametr	Hodnota
Napájecí napětí	24 V
Logická 0 digitálních vstupů	< 8V
Logická 1 digitálních vstupů	> 16V
Digitální výstupy	0/24V max. 100mA

Tab. 1: Parametry modulu digitálních vstupů a výstupů



Obr. 1: Blokové schéma modulu digitálních vstupů a výstupů

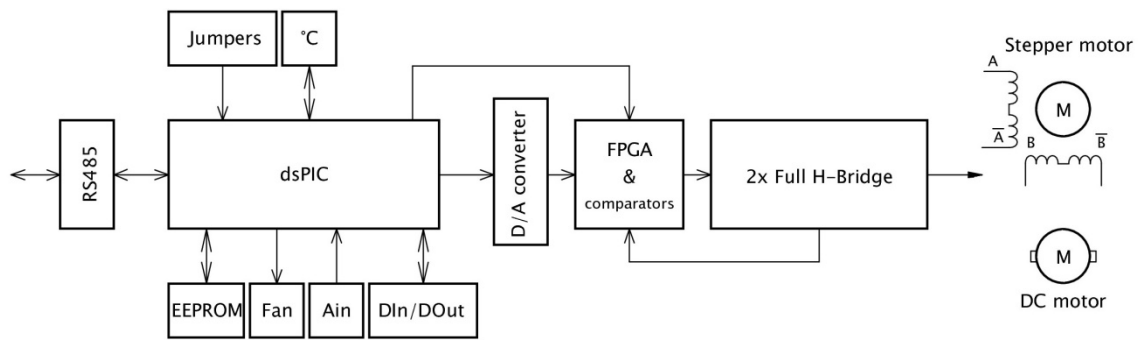
1.2 Modul řízení motorů

Modul umožňuje řízení dvoufázových krokových motorů a stejnosměrných motorů. Je vybaven šesti 24V digitálními vstupy, které poslouží například pro zjištění stavů koncových spínačů. Dále modul disponuje analogovým vstupem pro připojení tachogenerátoru a vstupem pro připojení snímače otáček motoru. Pomocí rozhraní RS485 je možné modul nakonfigurovat pro konkrétní použití a řídit jej. Modul je postaven na signálovém mikroprocesoru dsPIC a dává nadřazenému systému k dispozici informace o poloze a rychlosti motoru, teplotě výkonového stupně, napájecím napětí a informace o interních chybách. Celkem je možné mít připojeno až 16 modulů, jejichž adresy od 1 do 16 je možné nastavit pomocí otočného přepínače.

Důležité parametry modulu shrnuje následující tabulka.

Parametr	Hodnota
Napájecí napětí	24 V
Proud krokovým motorem	0,19 až 3A (volitelné)
Logická 0 digitálních vstupů	< 8V
Logická 1 digitálních vstupů	> 16V
Možnost zpětné vazby motoru	Rotační enkodér, tachogenerátor

Tab. 2: Parametry modulu řízení motorů



Obr. 2: Blokové schéma modulu řízení motorů

2 TECHNOLOGIE POUŽITÉ PRO VÝVOJ

Při výběru prostředků pro vývoj, byla zohledňována především dostupnost a náročnost na jejich používání.

2.1 Sériové rozhraní COM

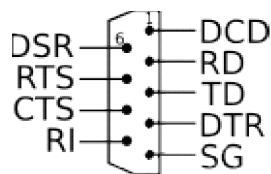
Rozhraní COM port je zkratkou Communications port. Je založen na integrovaných obvodech UART (Universal Asynchronous Receiver Transmitter) a byl součástí již prvních počítačů. Využíván byl především ke komunikaci s periferiemi jako myš, tiskárna nebo modem. U osobních počítačů je již vytlačen modernějším USB, nicméně v průmyslu se stále používá například ke konfiguraci různých regulátorů nebo síťových prvků.[7][15]

2.1.1 Standard RS-232C

Standard definuje řídicí signály, typy konektorů a popisuje synchronní a asynchronní režim přenosu. Data jsou přenášena relativně k uzemněnému vodiči, logické jedničky odpovídá napětí od -3V do -12V a logické nule odpovídá napětí od 3V do 12V. Pásmo od -3V do 3V zajišťuje hysterizi, kdy k překlopení do druhé stavu dojde po překonání těchto úrovní. Maximální délka kabelu je standardem stanovena na 15 metrů. Předepsané konektory jsou DB9P a DB25P. DB25P obsahuje signály nutné pro synchronní přenos a vzhledem k tomu, že port COM nepodporuje synchronní režim přenosu, je nejběžněji využívaným konektorem DB9P, který tyto signály nemá. Signály konektoru DB9P jsou definovány v následující tabulce.[5][15]

Signál	Funkce
PG	Ochranné uzemnění připojené ke krytu zařízení
SG	Signálová zem
TD	Sériová data – výstup vysílače
RD	Sériová data – vstup přijímače
DTR	Oznámení požadavku na použití zařízení
DSR	Potvrzení signálu DTR - zařízení navázalo spojení
RTS	Žádost o povolení k přenosu
CTS	Potvrzení signálu RTS – přenos povolen
DCD	Příznak detekce nosného signálu
RI	Indikátor vyzvánění

Tab. 3: Signály RS-232C konektoru DB9P[15]



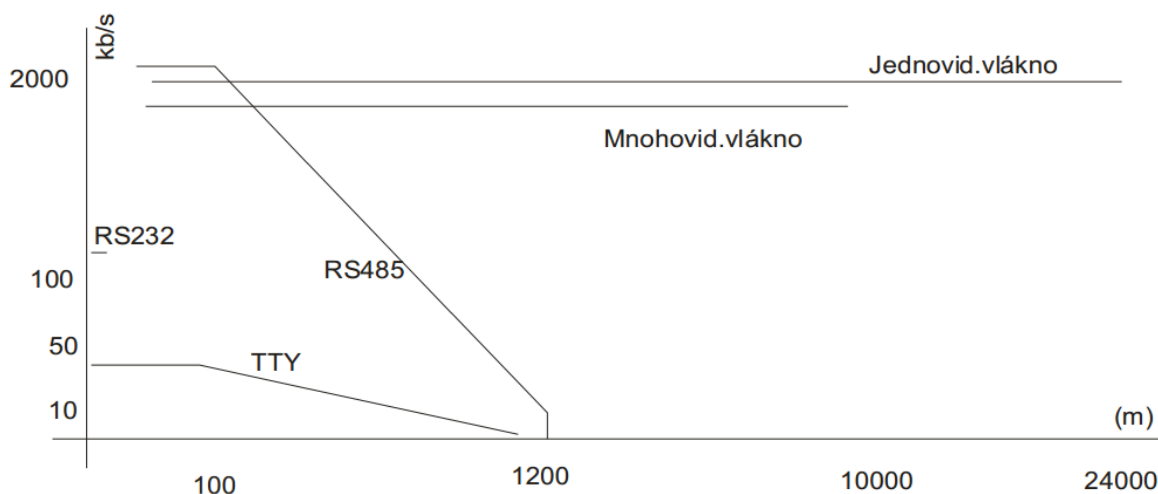
Obr. 3: Zapojení konektoru DB9P

2.1.2 Standard RS485

Komunikace pomocí RS485 je v průmyslovém prostředí často užívaným komunikačním rozhraním, jelikož vyniká dobrou odolností a spolehlivostí. Dovoluje propojení typu point-to-point, nebo typu sběrnice, kdy na sběrnici může být připojeno až 32 jednotek. Většího počtu jednotek lze docílit zvětšením jejich vstupního odporu, ovšem za cenu zmenšení maximální přenosové rychlosti a délky vedení. [5][15]

2.1.2.1 Délka vedení

Délka metalického vedení může být až 1200m a v závislosti na délce vedení je možné komunikovat rychlostí až 2Mb/s. V případě využití optického vedení je možné jednotky propojovat až vzdálenosti několika kilometrů. [5]



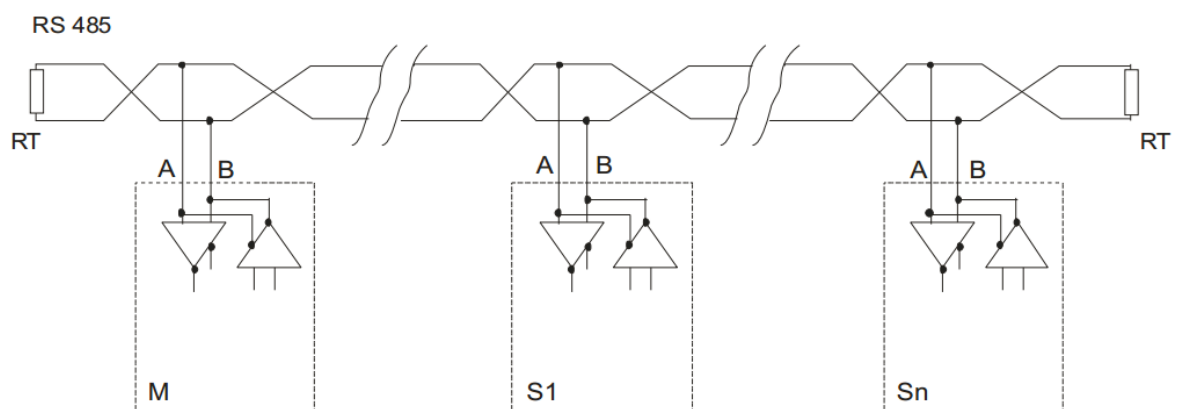
Obr. 4: Poměr rychlosti přenosu a délky propojení [5]

2.1.2.2 Elektrické zapojení

Elektrické zapojení je definováno standardem, který ustanovilo sdružení EIA. K propojení jedné signálové cesty se používají dva vodiče, nejlépe v provedení kroucené dvojlinky (twist), označené jako „A“ a „B“. Bohužel ze standardu není naprosto zřejmé, který

z vodičů je „A“ a který „B“. Proto se lze v praxi setkat se situací, kdy je nutné vodiče zaměnit. Nicméně, normou je definováno proudové omezení, a proto při prohození vodičů nehrozí poničení zařízení. Rozhraní se může vyskytovat ve dvou podobách zapojení a to se dvěma vodiči, nebo se čtyřmi vodiči. Zapojení se čtyřmi vodiči využívá jeden pár pro vysílání a druhý pár pro příjem. Dvou vodičové zapojení komunikuje po jednom páru oběma směry, kdy směr vysílání je u každého zařízení programově přepínán. V obou případech je zvolena řídicí master jednotka, která se dotazuje podřízených jednotek. Tím je zajištěno, že v jeden moment nebude vysílat více zařízení. Vysílače, které momentálně nevysílají, jsou přepnuty do stavu vysoké impedance. [6][15]

Přenesená informace je přijímačem vyhodnocována jako napěťový rozdíl $U_A - U_B$ mezi vodiči A a B. Rozdílové napětí, které je menší než $-0,2V$, odpovídá logické jedničce a rozdílové napětí větší než $0,2V$ odpovídá logické nule. Případné rušení, které by se indukovalo na obou vodičích stejné, se díky odečtení vyruší. Proto je rozhraní odolné i v průmyslovém prostředí. Aby nedocházelo k odrazům signálů, je nutné koncová zařízení vybavit terminačními odpory obvykle 120Ω . [15] [5]



Obr. 5: Schéma dvou vodičového propojení RS485[5]

2.1.3 Standard RS422

Standard RS422 je elektricky kompatibilní s RS485 a pracuje rovněž pomocí symetrických komunikačních linek. Hlavní rozdíl se nachází ve vysílačích. Vysílače RS422 nejsou schopné se přepnout do stavu vysoké impedance, a proto je i maximální počet připojených jednotek omezen na 1 vysílač a 10 přijímačů.

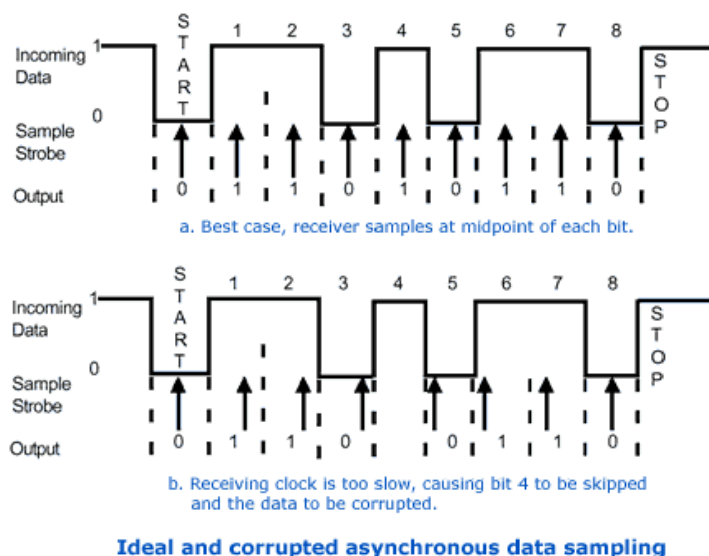
2.1.4 Srovnání standardů RS232C, RS422 a RS485

	RS232	RS422	RS485
Max. počet zařízení	1 vysílač 1 přijímač	5 vysílačů 10 přijímačů	32 vysílačů 32 přijímačů
Max. délka vedení	15 m	1200 m	1200 m
Max. přenosová rychlost na 15m	19.2 kbps	10 Mbps	10 Mbps
Logická 1	-3V až -12V	2V až 6V (B>A)	1,5V až 5V (B>A)
Logická 0	3V až 12V	2V až 6V (A>B)	1,5V až 5V (A>B)
Min. vstupní úroveň	+3V	Rozdíl 0,2V	Rozdíl 0,2V
Výstupní proud	500mA	150mA	250mA

Tab. 4: Srovnání RS232C, RS422 a RS485[15][16]

2.1.5 Asynchronní přenos dat

Asynchronní přenos nevyžaduje ke své funkci další podpůrné signály a na jeho principu pracují i uvedené standardy RS232C, RS485 a RS422. Nejmenší objem dat přenesený v rámci jednoho přenosu je jeden znak, který může mít 5,6,7 nebo 8 bitů. Každý přenos začíná vysláním startovacího bitu, který má hodnotu logické nuly a oznamuje přijímači začátek přenosu. Za ním následují data a celý přenos je ukončen stop bitem, který má hodnotu logické jedničky. Tímto způsobem je pomocí datových vodičů zajištěna synchronizace přijímače a vysílače. Ta je pro sériovou komunikaci nutným prvkem, pokud by nebyla zajištěna, přijímač by vysílaná data sice přečetl, ale došlo by k posunu či vynechání některých bitů jako na obrázku 6. [7][15]



Ideal and corrupted asynchronous data sampling

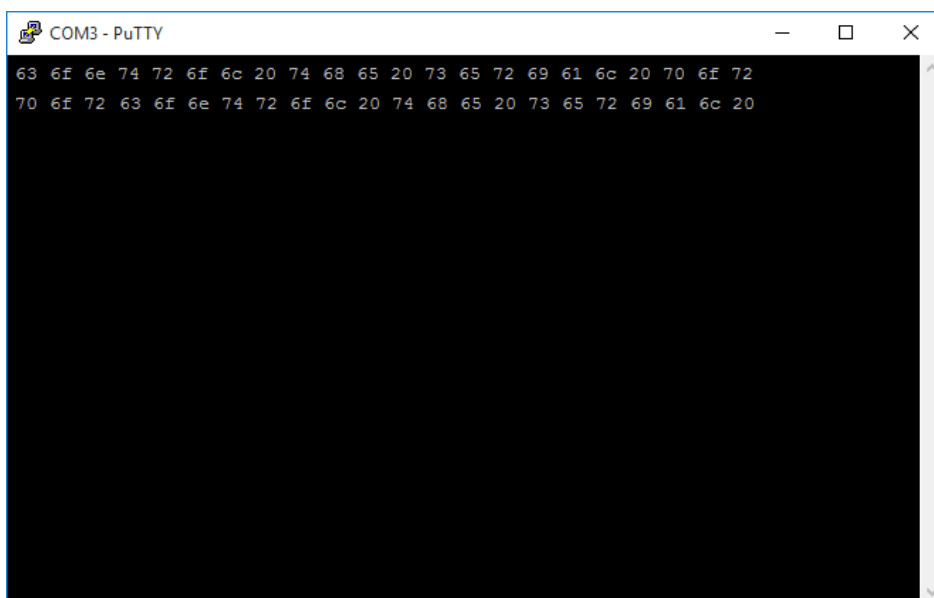
Obr. 6: Synchronizace vysílače a přijímače [7]

2.2 Přehled dostupných utilit pro obsluhu COM portu

Možnost, jak navázat sériovou komunikaci mezi běžným osobním počítačem a externím zařízením, je pomocí portu COM. Ten je možné obsluhovat pomocí volně dostupných utilit, které ovšem nenabízí komfortní ovládání pro běžného uživatele a jsou vhodné spíše pro vývoj hardwaru, nebo občasné potřeby čtení dat.

2.2.1 Putty

Tento jednoduchý program, který se vzhledem i chováním podobá linuxovému terminálu, je prezentován především jako SSH a TELNET klient. Má ovšem možnost otevřít i COM port a číst z něj data. Bohužel práce s COM portem je velmi omezená a neumožňuje žádné formátování čtených dat. Proto se hodí spíše pro diagnostické účely.



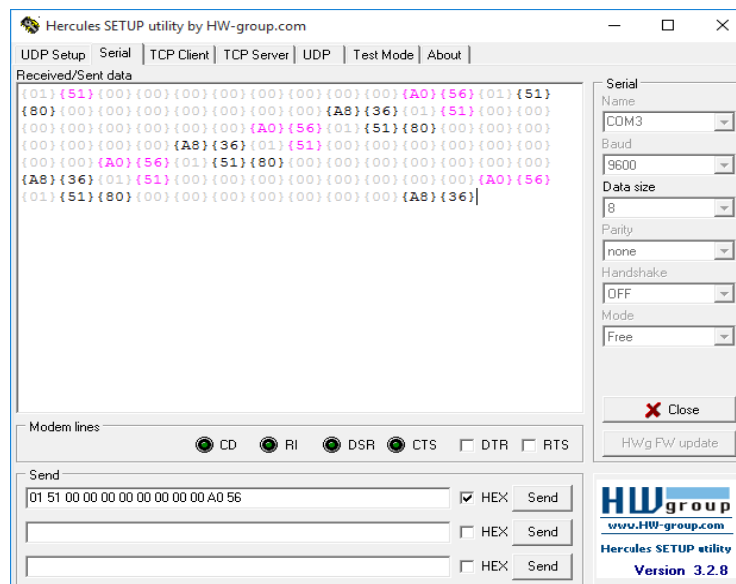
Obr. 7: Ukázka utility Putty

2.2.2 Hercules

Hercules je volně šiřitelný program od firmy HW Group, původně vyvinutý pro potřeby konfigurace hardwarových modulů vyráběných tutéž firmou. Nakonec se rozrostl i o užitečné terminály pro TCP/IP klienta, TCP/IP server, UDP a pro sériový COM port.[11]

Sériový terminál Hercules je podstatně propracovanější na rozdíl od terminálu Putty. Umí zapisovat data na port, a kromě základního nastavení jako jméno portu, rychlost, parita apod. umožňuje i ovládání řídicích signálů DTR a RTS. Data pro odeslání je možné zapsat textově nebo v HEX formátu a lze si takto připravit tři různé zprávy. Na port je možné za-

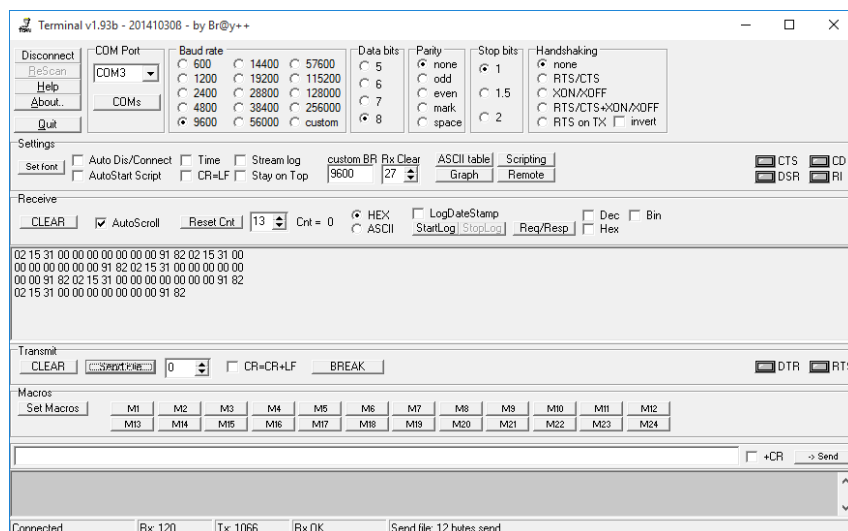
psat i soubor. Zobrazená přijatá a odeslaná data lze formátovat do hexadecimální, nebo textové podoby a lze je uložit do souboru.[12]



Obr. 8: Ukázka utility Hercules

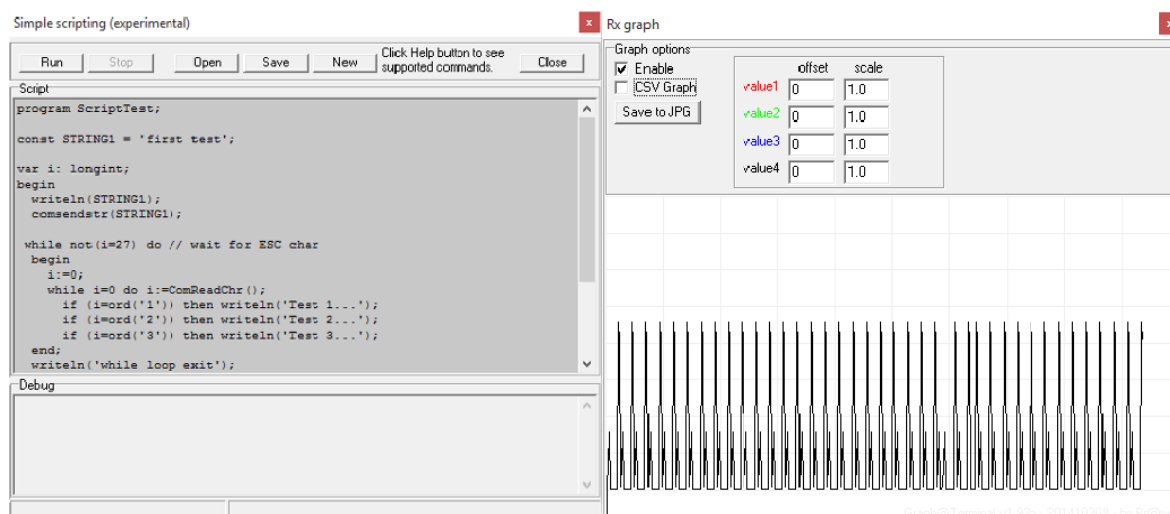
2.2.3 Terminal

Ze všech uvedených je Terminal nejpropracovanější utilita pro obsluhování COM portu. Umí nastavit veškeré parametry komunikace i ovládat řídicí signály. Přijímaná data umí ukládat do souboru a ze souboru umí data i odesílat. Umožňuje vytvářet makra a spouštět je pomocí klávesových zkratk. [11]



Obr. 9: Ukázka utility Terminal

Poslední novinkou programu Terminal je podpora jednoduchého skriptovacího jazyka pro vytváření různých sekvencí a dále možnost sledování datového toku v grafu. [11]



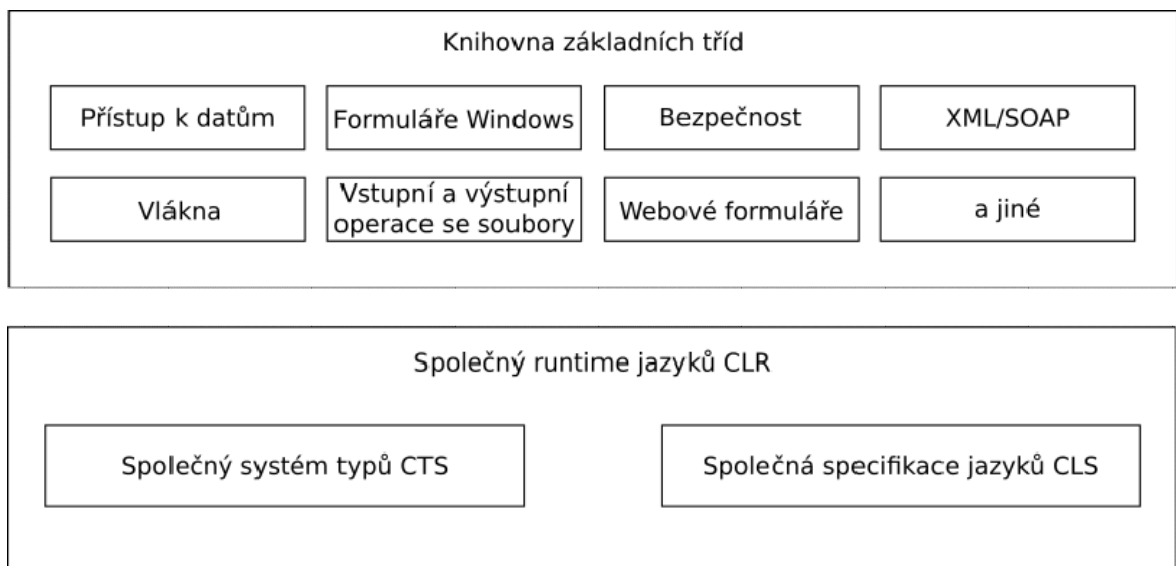
Obr. 10: Ukázka utility Terminal – graf

Utilitě terminal lze vytknout příliš malá tlačítka, na kterých se při fokusu stane text nečitelným. Další nevýhodou může být nemožnost zadávání dat pro odeslání v hexadecimálním formátu, nicméně se dá nahradit odesláním dat přímo ze souboru vytvořeného například hex editorem.

2.3 Framework .NET

Jedná se o poslední model společnosti Microsoft, pro vytváření aplikací na systémech Windows, ale také na různých distribucích Linuxu, nebo Mac OS X. Framework .NET je radikálním odklonem od předcházejících komplikovaných řešení a přináší řadu výhod. Mezi ně patří například možnost kombinovat kód přeložený z různých jazyků, rozsáhlost a propracovanost knihoven a s tím související rychlost a efektivita vývoje. [1]

Strukturu frameworku .NET si lze představit podle následujícího obrázku.



Obr. 11: Struktura .NET [1]

Knihovna základních tříd je podporou pro programátora cílové aplikace a poskytuje základní prostředky pro práci se soubory, vlákny, externími zařízeními, formuláři Windows a spoustu dalších. [1]

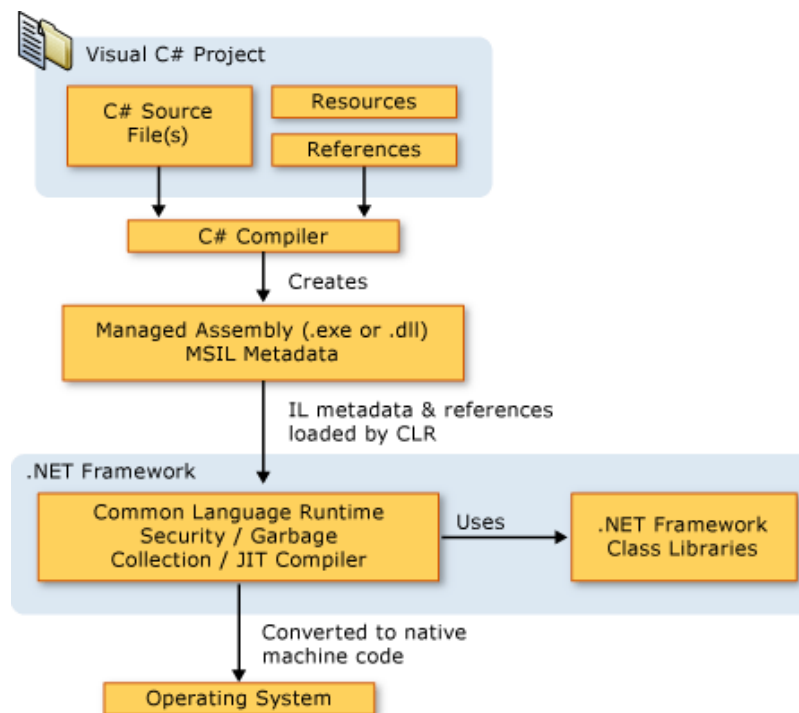
Moduly CLR, CTS a CLS jsou základními součástmi .NET frameworku, díky kterým je možné aplikace vytvářet v různých jazycích pro různé platformy. [1]

2.3.1 Společný runtime jazyků CLR

CLR vrstva je základem běhového prostředí frameworku. Při kompilaci programu napsaného jakýmkoliv jazykem podporující .NET, je vždy výstupem společný intermediární jazyk – Intermediate Language (IL). Na rozdíl od kompilace programů napsaných například v C/C++, kdy je výstupem strojový kód pro konkrétní platformu, je jazyk IL na platformě nezávislý. Výsledný soubor, zvaný assembly, se ovšem neskládá jen z jazyka IL. K tomuto zkompilovanému programu se ještě přidávají metadata a takzvaný manifest. Me-

tadata detailně popisují všechny typy (myšleno třídy, výčet, delegáti apod.) jejich implementace rozhraní nebo z čeho jsou odvozeny apod. Manifest obsahuje popis samotné assembly, její verzi, požadované externí assembly, nebo informace o jazykové kultuře. Výsledný soubor skládající se z uvedených tří částí je uložen s příponou *.exe nebo *.dll. [2][1]

Vrstva CLR prezentovaná systémovou knihovnou mscorere.dll se stará o načtení této assembly do paměti a načtení požadovaných typů popsanych v metadatech. Následně s využitím just-in-time kompilátoru provede překlad jazyka CIL do strojového kódu cílové platformy. Vždy se překládá jen ta část programu, která je potřeba. Přeložené části se uchovávají v paměti pro případné další použití. [1]



Obr. 12: Kompilace v .NET frameworku [3]

2.3.2 Společný systém typů CTS

CTS specifikuje jak musí typ vypadat, aby byl kompatibilní s .NET. Typem se zde rozumí třída, struktura, rozhraní, výčet nebo delegát. Celkem je tedy specifikováno 5 základních typů.

2.3.2.1 Typ třída

Typ třída je základním prvkem objektově orientovaného programování. Obvykle je složena z vlastností, metod a událostí a je charakterizována následujícími vlastnostmi.

- Použitelná jako základní třída jiné třídy.
- Implementuje nějaké rozhraní.
- Je abstraktní – nelze vytvořit přímo.
- Viditelná i pro externí assembly.

V C# se vytváří pomocí klíčového slova `class`. [1]

2.3.2.2 Typ struktura

Typ struktura slouží k definici uživatelských typů a je velmi podobná třídě. Je založená na hodnotové sémantice a hodí se k vyjádření matematických dat. V C# se vytváří pomocí klíčového slova `struct`. [1]

2.3.2.3 Typ rozhraní

Typ rozhraní se používá v kombinaci s třídou nebo strukturou. Rozhraní definuje pouze seznam členů, jejichž chování naimplementuje konkrétní třída. Implementuje-li třída rozhraní, znamená to, že implementuje chování všech prvků, které rozhraní definuje. Jakýkoliv počet tříd může implementovat rozhraní svým jedinečným způsobem. V C# se vytváří pomocí klíčového slova `interface`. [1]

2.3.2.4 Typ výčet

Typ výčet slouží k vytvoření seznamu dvojic jméno – hodnota v rámci jedné logické skupiny. S výhodou ho lze použít například pro výčet vlastností objektu. V C# se vytváří pomocí klíčového slova `enum`. [1]

2.3.2.5 Typ delegát

Typ delegát je typově bezpečný ukazatel na jakoukoliv metodu. Využití nachází především u volání událostí. V C# se definuje pomocí klíčového slova `delegate`. [1]

2.3.3 Společná specifikace jazyků CLS

CLS je specifikací, která je důležitá zejména pro tvůrce kompilátorů, pro jazyky podporující .NET. Popisuje veškeré detaily nutné k tomu, aby byl kompilátor schopen vygenerovat kód srozumitelný a spustitelný pomocí CLR. [1]

2.3.4 Vývoj a přehled verzí

Následující tabulka zobrazuje přehled vydaných verzí .NET frameworku.

Verze .NET	Rok vydání	Funkce
1.0	2002	První verze. Součástí C# 1.0, VB.NET
1.1	2003	Technologie ASP.NET, podpora IPv6.
2.0	2006	Generické kolekce, podpora 64bit aplikací
3.0	2006	WPF – pokročilý framework pro tvorbu formulářů. WCF – framework pro servisně orientované aplikace
3.5	2007	Podpora LINQ, weby s podporou AJAX
4.0	2010	Rozšíření základních knihoven
4.5	2012	Podpora HTML 5 v ASP.NET, aktualizace WPF a WCF
4.5.1	2013	Vylepšení výkonu a podpory multiplatformních aplikací
4.5.2	-	Nové rozhraní pro ASP.NET
4.6	2015	Nové jádro ASP.NET, podpora protokolu HTTP/2

Tab. 5: Verze .NET frameworku [8][9]

2.4 Programovací jazyk C#

Jazyk C# byl vyvinut společností Microsoft speciálně pro potřeby programování v prostředí .NET. Jedná se o čistě objektově orientovaný jazyk se syntaxí vycházející z C++ a jeho hlavní přednosti jsou: [3][1]

- Automatická správa paměti – garbage collection.
- Nepotřebnost manipulace s ukazateli.
- Podpora typově bezpečných delegátů pro vyvolání událostí.
- Podpora generických typů a lambda výrazů.
- Podpora properties pro zjednodušený přístup k privátním členům třídy – není nutné vytvářet zvlášť getter a setter metody.

Ukázka programu „Hello World“ s výpisem na konzoli:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp
{
    class Program
    {
        static void Main( string[] args )
        {
            string Hello = "Hello World!";
            Console.WriteLine( Hello );
        }
    }
}
```

Obr. 13: Hello World v C#

2.4.1 Použité prostředky pro práci s vlákny

Stěžejním bodem pro vytvoření GUI aplikace komunikující po sériovém portu, je využití vláken. Aby byl zajištěn plynulý běh aplikace, je nutné vytvořit vlákno, které bude komunikaci obsluhovat. V aplikaci bude vytvořeno několik vláken většinou v modelu producent/konzument a pro tvorbu a synchronizaci vláken bude využito základních prostředků jazyka C#.

2.4.1.1 Vytvoření vlákna

Vlákno je prezentováno třídou *Thread*, které se do konstruktoru předává delegát na metodu, která bude vláknem vykonána. Metodou *Start* se vlákno spustí a ukončí se ve chvíli, jakmile vykoná celou metodu, která mu byla předána. Po ukončení je vlákno uklizeno garbage collectorem a nelze jej podruhé odstartovat. Je nutné vytvořit nové vlákno.[13]

2.4.1.2 Synchronizační prostředek *AutoResetEvent*

AutoResetEvent představuje základní synchronizační prostředek. Obsahuje metodu *WaitOne*, která je blokující, a na které se vlákna řadí do pomyslné fronty. Zablokovaná vlákna je možné odblokovat zavoláním metody *Set* z jiného vlákna. Jedním zavoláním dojde k propuštění jednoho vlákna. Vlákno je možné zablokovat pouze na určitý čas, který lze zadat jako nepovinný parametr metodě *WaitOne*. Zavoláním metody *Set* v době, kdy na *WaitOne* není žádné čekající vlákno, bude automaticky propuštěno první vlákno, které na *WaitOne* narazí. Ostatní vlákna budou blokována a budou čekat na další volání *Set*. [13]

```
class BasicWaitHandle
{
    static EventWaitHandle wh = new AutoResetEvent(false);

    static void Main()
    {
        new Thread(Waiter).Start();
        Thread.Sleep(5000); // Chvilku počkáme
        wh.Set();           // Pustíme vlákno dál
    }
    static void Waiter()
    {
        Console.WriteLine("Čekám...");
        wh.WaitOne();           // Čekat na propustku
        Console.WriteLine("Propuštěn!");
    }
}
```

Obr. 14: Použití synchronizace pomocí *AutoResetEvent* [13]

2.5 Microsoft Visual Studio Community

Visual Studio Community je nástrojem od společnosti Microsoft pro vývoj desktopových i webových aplikací pro Windows, Android a iOS. Podporuje kódování v jazycích C#, Visual Basic, C++, HTML, Javascript a další. Verze Community je zdarma pro malé týmy vývojářů a školní a open-source projekty. Je nutná pouze registrace. [4]

Obsahuje veškeré standardní funkce nutné k vývoji, jako ladění či návrhář formulářů. Disponuje užitečnou funkcí IntelliSense, která při psaní kódu nabízí automatické doplnění nebo automatické opravení kódu. [4]

II. PRAKTICKÁ ČÁST

3 STANOVENÉ CÍLE PRAKTICKÉ ČÁSTI

Jedna ze součástí výroby hardwarových modulů je i jejich konfigurace pro konkrétní zařízení. Tato zařízení bývají vybavena vestavěným řídicím počítačem se specifickým softwarem, který moduly obsluhuje a je schopen je i konfigurovat. Ovšem se vzrůstajícím počtem typů modulů a různých zařízení, vzniká ve výrobě problém moduly efektivně testovat, jelikož je vždy zapotřebí řídicí počítač s odpovídajícím softwarem, který umí vyrobené moduly obsluhovat. To způsobuje, že je výroba nucena učit se moduly testovat v různých softwarech a různým způsobem. Tento velice nepraktický způsob dal vzniknout myšlence vyvinout jednotný software, který bude schopen s jakýmkoliv modulem pracovat.

V praktické části bakalářské práce je s využitím uvedených technologií takový software vyvinut. Je určen především pro běžné uživatele a servisní techniky, kde je možné, pomocí jednoduchého grafického rozhraní, hardwarové moduly ovládat a konfigurovat, což by nebylo v uvedených utilitách snadno realizovatelné. Rovněž umožňuje manuálně vysílat a přijímat zprávy pomocí přehledné konzole, kterou jistě uvítají vývojáři hardwaru.

Pro jednoduchost použití je software spustitelný na standardním PC a slouží i jako diagnostický nástroj pro zařízení, které už jsou v provozu. Je použitelný i pro vývojové prototypy zařízení, kdy pro hlavní řídicí počítač obvykle není vyvinut software. Pro tyto případy software umožňuje zobrazení vybraných veličin v grafu a podporuje jazykové mutace.

4 PROPOJENÍ MODULŮ S POČÍTAČEM

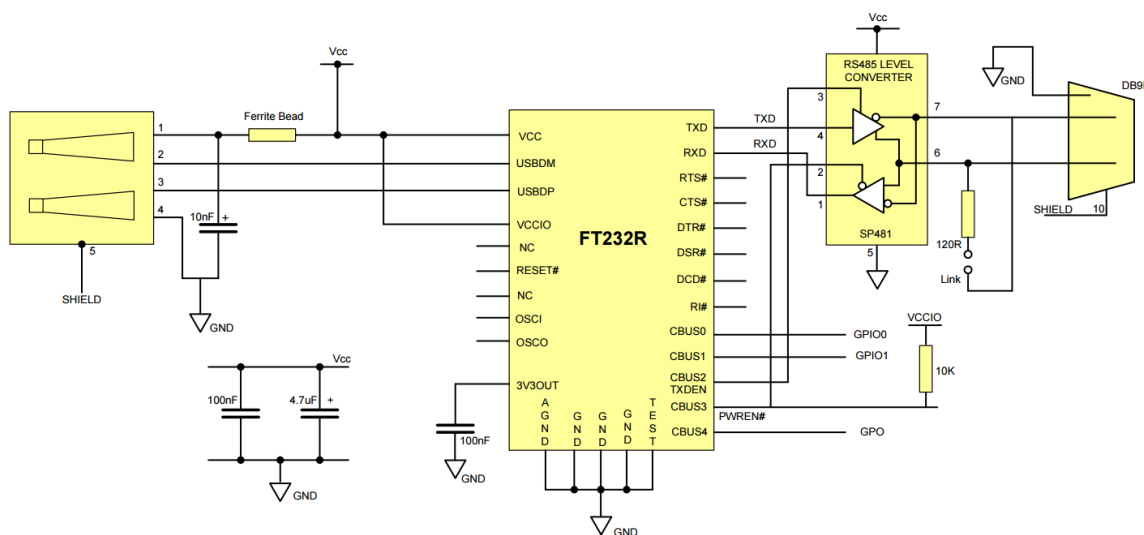
Základním stavebním kamenem praktické části je propojení hardwarových modulů s počítačem.

Moduly využívají ke komunikaci rozhraní RS485. Standardní stolní počítače ovšem tímto rozhraním vybavené nejsou, a proto je ideálním řešením použít cenově dostupný převodník RS485-USB. Dalším řešením by bylo, použít převodník z RS485 na RS232 a využít sériový port počítače. Zde je ovšem problém se stále menším výskytem těchto portů u stolních počítačů.

4.1 Převodník PremiumCord USB2.0 na RS485

PremiumCord je převodník, který nevyžaduje externí napájení, po připojení a instalaci ovladačů, je na počítači vytvořen nový virtuální COM port, pomocí kterého lze komunikovat jako se standardním sériovým portem.

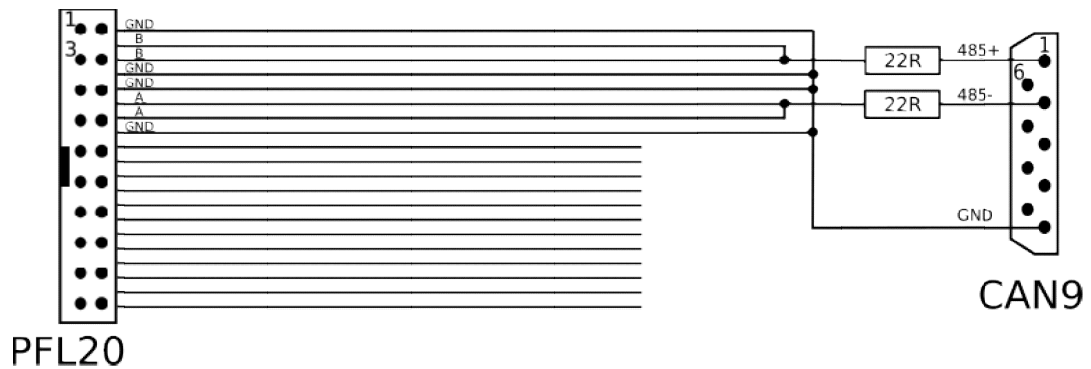
Základem převodníku je čip FT232RL od firmy FTDI, který může být použit i pro jiné převodníky jako USB na RS422, nebo USB na RS232. Výstup z čipu je vždy upraven na požadované logické úrovni. [10]



Obr. 15: Příklad zapojení převodníku USB na RS485[10]

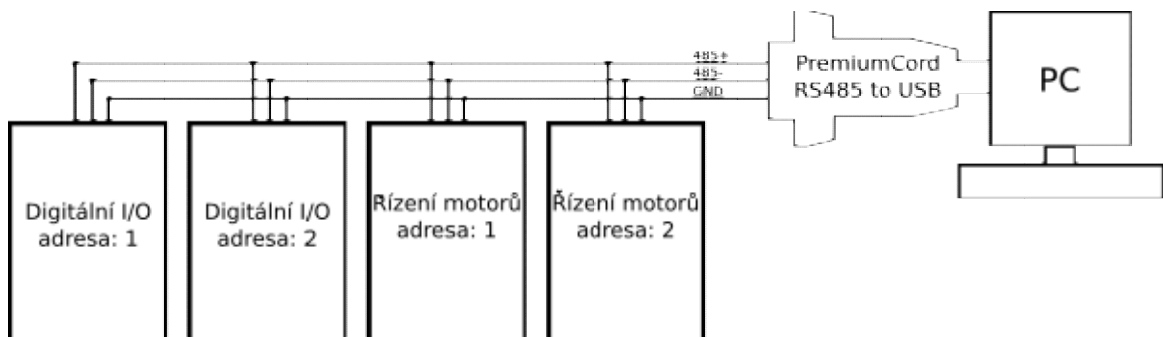
4.2 Propojení s počítačem

Při propojení modulů s počítačem je nutné dbát zejména na správnou adresaci modulů a zapojení vodičů. Moduly se navzájem propojují 20 žilovým kabelem s konektorem PFL20 a s převodníkem jsou propojeny následujícím způsobem:



Obr. 16: Zapojení kabelu PFL20 – CAN9

Typické propojení a adresace například dvou digitálních karet a dvou karet řízení motorů, by vypadala následujícím způsobem.



Obr. 17: Propojení a adresace modulů

4.3 Komunikační protokol

4.3.1 Konfigurace portu

Sériový port je nakonfigurován s následujícími parametry:

- Baud rate: 9600 bd
- Počet datových bitů: 8
- Počet stop bitů: 1

4.3.2 Struktura zprávy

Ke komunikaci jsou využívány krátké zprávy o pevné délce 12 bajtů s následující strukturou.

Adresa	Příkaz	data byte 1	data byte 2	data byte 3	data byte 4	data byte 5	data byte 6	data byte 7	data byte 8	CRC16 low	CRC16 high
--------	--------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	--------------	---------------

Obr. 18: Struktura zprávy

Adresa a příkaz tvoří společně jedinečnou identifikaci modulu na sběrnici, kde každý typ modulu podporuje pouze svou určitou skupinu příkazů. Adresa slouží k rozlišení pouze v rámci skupiny modulů stejných typů. Každý modul má adresu nastavitelnou pomocí zkratovací propojky. Na konci zprávy je dvoubajtový kontrolní součet, který je počítán následujícím algoritmem.

```
private byte[] GetCRC16( byte[] Data )
{
    uint Result = 0;
    byte[] Retval = new byte[2];

    if( Data != null )
    {
        if( Data.Length == MessageLength )
        {
            for( int i = 0; i < (MessageLength - 2); i++ )
            {
                uint TmpByte = (uint)Data[i];
                for( int ii = 0; ii <= 7; ii++ )
                {
                    if( ((TmpByte ^ Result) & 1) == 1 )
                    {
                        Result = Result >> 1;
                        Result = ((Result ^ 0x2001) | 0x8000);
                    }
                    else
                    {
                        Result = Result >> 1;
                    }
                    TmpByte = TmpByte >> 1;
                }
            }
        }
        Retval[1] = (byte)(Result & 0xFF);
        Retval[0] = (byte)((Result >> 8) & 0xFF);
    }
    return Retval;
}
```

Obr. 19: Algoritmus pro výpočet kontrolního součtu

5 KOMUNIKAČNÍ JÁDRO

Komunikační jádro je prezentováno třídou *Port485Core* a zajišťuje styk se spodní komunikační vrstvou. Umožňuje otevření portu, registraci událostí pro zpracování zpráv a obsahuje rutiny pro permanentní vyčítání stavu z modulů zprávami, které jsou jádru předány. Dále obsluhuje detekci modulů, logování přijatých a odeslaných zpráv a v neposlední řadě umožňuje odeslání libovolné zprávy na modul.

5.1 Implementace jádra

Třída *Port485Core* obsahuje vlastní vlákno, které neustále kontroluje výstupní frontu zpráv. Je-li fronta prázdná, jsou dokola vysílány takzvané periodické zprávy. Jedná se o zprávy předané jádru, které by měly zajišťovat vyčtení nejdůležitějšího stavu z modulů. Aby byla zajištěna rychlá odezva při větším počtu modulů, mělo by být periodických zpráv co nejméně. Ovšem ne méně jako jedna pro každý modul, tím by nebyla zajištěna kontrola spojení s modulem. Na základě informací z periodických zpráv, lze poté reagovat jednorázovým vyčtením dalších zpráv pro získání podrobnějších informací.

Ve stejném vlákne jako vysílání periodických zpráv probíhá i detekce modulů připojených na sběrnici. Základem je opět předání seznamu zpráv, které budou během detekce vysílány. Detekce se spouští pouze na požadavek uživatele a o zpracování odpovědi, jako zápis do seznamu detekovaných modulů, se stará nadřazený objekt.

5.2 Implementace protokolu

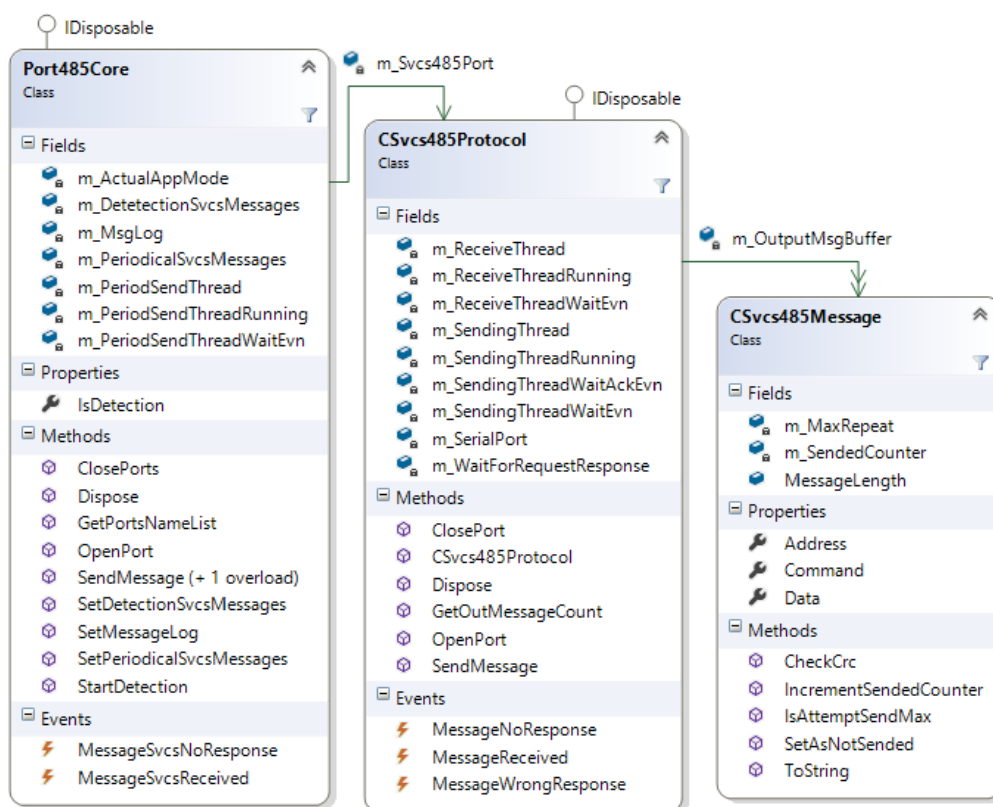
Jádro ke své činnosti využívá třídu, která představuje nejnižší komunikační vrstvu a obstarává samotnou komunikaci. Třída se jmenuje *Svcs485Protocol..* Jejím základem je přijímací a vysílací vlákno s frontou zpráv, které čekají na odeslání. Dále umožňuje registraci událostí, které jsou vyvolány v případě příjmu nové zprávy, nebo nedoručení požadované zprávy.

Hlavní úkolem třídy je odeslat požadovanou zprávu a vrátit odpověď. Přejde-li požadavek na odeslání zprávy, je zpráva zařazena do fronty a probuzeno vysílací vlákno, které se postará o zápis zprávy na port. Odpoví-li modul na zprávu, je probuzeno přijímací vlákno, které vyčte data z portu a ověří kontrolní součet. Při souhlasném kontrolním součtu přijímací vlákno odstraní požadavek z výstupní fronty a vyvolá událost doručené odpovědi.

V případě, kdy modul neodpovídá, nebo nesouhlasí kontrolní součet odpovědi, je zpráva znovu vysílána. Pokud ani po vícenásobném vyslání modul neodpoví, je vyvolána příslušná událost a požadovaná zpráva je odstraněna z výstupní fronty.

5.2.1 Chování přijímacího vlákna v režimu naslouchání

V případě výskytu dat na portu, je přijímací vlákno probuzeno a začne vyčítat data z portu po jednotlivých bajtech. Jakmile vyčítá 12 bajtů, což je stanovená pevná délka zprávy, přejde vlákno ke zpracování zprávy. V režimu naslouchání, kdy je komunikace řízena jiným programem, není s jistotou znám začátek zprávy, protože naslouchání může začít právě ve chvíli, kdy je jiný program v polovině vysílání. Mohlo by se tedy stát, že vyčtených celkový 12 bajtů bude složeno ze dvou různých zpráv. Pro tyto případy je přijímací vlákno vybaveno FIFO frontou o délce 12 bajtů, do které jsou přijímané bajty řazeny. Je-li fronta plná, je s každým přijatým bajtem počítán kontrolní součet. Pokud součet souhlasí, fronta je nulována a zpráva předána ke zpracování. Pokud nesouhlasí, dojde s dalším přijatým bajtem k posunutí prvků ve frontě a opětovné kontrole kontrolního součtu. Takto je fronta posouvána, dokud kontrolní součet nezačne souhlasit, čímž je docíleno nalezení začátku zprávy.



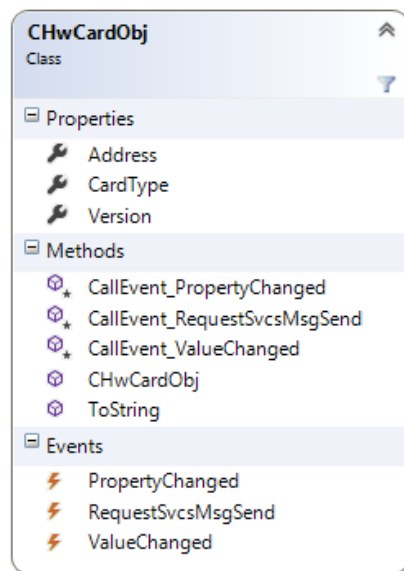
Obr. 20: Zjednodušený diagram tříd komunikačního jádra

6 SPRÁVA KOLEKCE MODULŮ

Pro usnadnění práce a zpřehlednění kódu byly definovány některé základní rodičovské třídy.

6.1 Rodičovská třída modulů

Každá konkrétní implementace modulu je potomkem třídy *CHwCardObj*. Rodičovská třída implementuje property typ modulu, adresu a verzi modulu. Dále generuje události indikující požadavek na vyslání zprávy, změnu hodnoty a změnu vlastnosti. Protože události definované v rodičovské třídě nelze přímo volat z odvozené třídy, implementuje ještě *protected* metody pro volání těchto událostí.



Obr. 21: Diagram rodičovské třídy modulů

6.2 Rodičovská třída manažeru karet

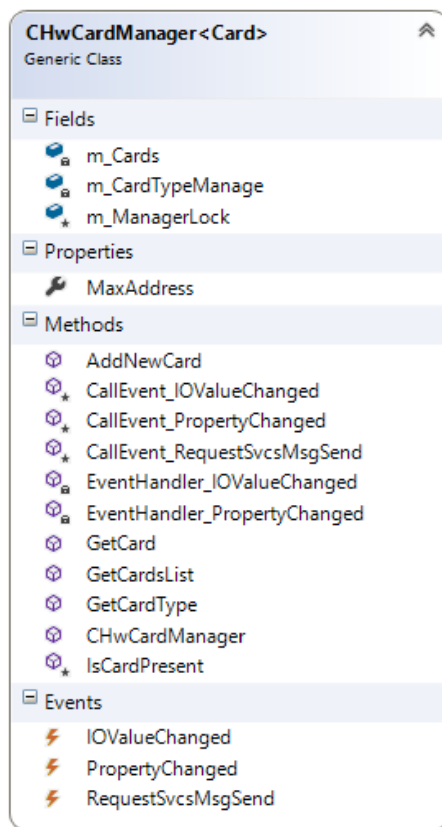
Třídy, které spravují kolekci s detekovanými moduly, a na kterou se obrací GUI s požadavky na data, se nazývají manažery. Pro každý typ modulu existuje specifický manažer, který je potomkem třídy *CHwCardManager*. Kolekce s detekovanými moduly je společná pro všechny manažery a reference na ni je předávána v konstruktoru. Rodičovská třída je implementována jako generická a typem může být pouze objekt, který je potomkem *CHwCardObj*, tedy objekt nějakého modulu. Třída implementuje události stejné, jako jsou implementovány u třídy modulu, tedy události indikující změnu hodnoty modulu, změnu vlastnosti modulu a požadavek na vysílání zprávy. Manažer událostem modulů naslouchá.

Pokud dojde k vyvolání události u modulu, dojde k vyvolání stejné události i v manažeru. Jako parametr se předá reference na modul, od kterého byla událost vyvolána. Dále třída implementuje metody pro vyhledávání karet ve společné kolekci. Díky genericitě, nemusí tyto metody vracet pouze obecný typ *CHwCardObj*, ale vrací již přetypovaný typ modulu. Příkladem může být metoda využívající jazyka LINQ pro vyhledání všech karet daného typu v kolekci.

```
/// <summary>
/// Returns list with all detected cards
/// </summary>
/// <returns>List with all detected cards</returns>
public List<Card> GetCardsList()
{
    lock ( m_Cards )
    {
        return m_Cards.Values.Where( val => val.CardType == m_CardTypeManage ).ToList().Cast<Card>().ToList();
    }
}
```

Obr. 22: Metoda pro vyhledání modulů v kolekci

Třída implementuje jedinou property maximální možná adresa modulu a metody pro vyvolání událostí z potomka.



Obr. 23: Diagram rodičovské třídy manažeru modulů

6.3 Funkce manažeru modulů

Instance manažeru modulů je vytvářena hlavní třídou aplikace. Ta manažeru předává přijaté zprávy z komunikačního jádra k dekodování. Po předání zprávy, manažer vyhledá modul s požadovanou adresou a následně modulu zprávu předá. Modul po zpracování vyhodnotí, zda došlo ke změně jeho hodnot a případně vyvolá příslušnou událost. Ta je zaregistrovaná v okně vlastností modulu a při zavolání události reaguje obnovením svých hodnot.

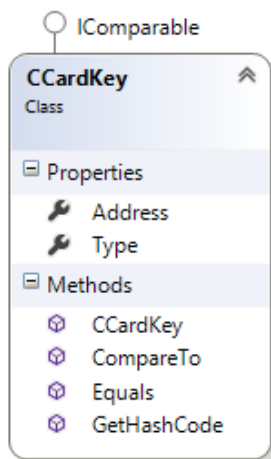
Během detekce modulů v režimu master i v režimu odposlouchávání, zajistí manažer při příchozí zprávě vytvoření instance nově zjištěných modulů. Ty přidá do kolekce detekovaných modulů, která je typu „tříděný slovník“. Klíčem je objekt třídy *CCardKey* a hodnotu představuje objekt modulu.

```
// Dictionary with all detected card  
private SortedDictionary<CCardKey, CHwCardObj> m_Cards;
```

Obr. 24: Kolekce detekovaných modulů

6.4 Jedinečný klíč modulu

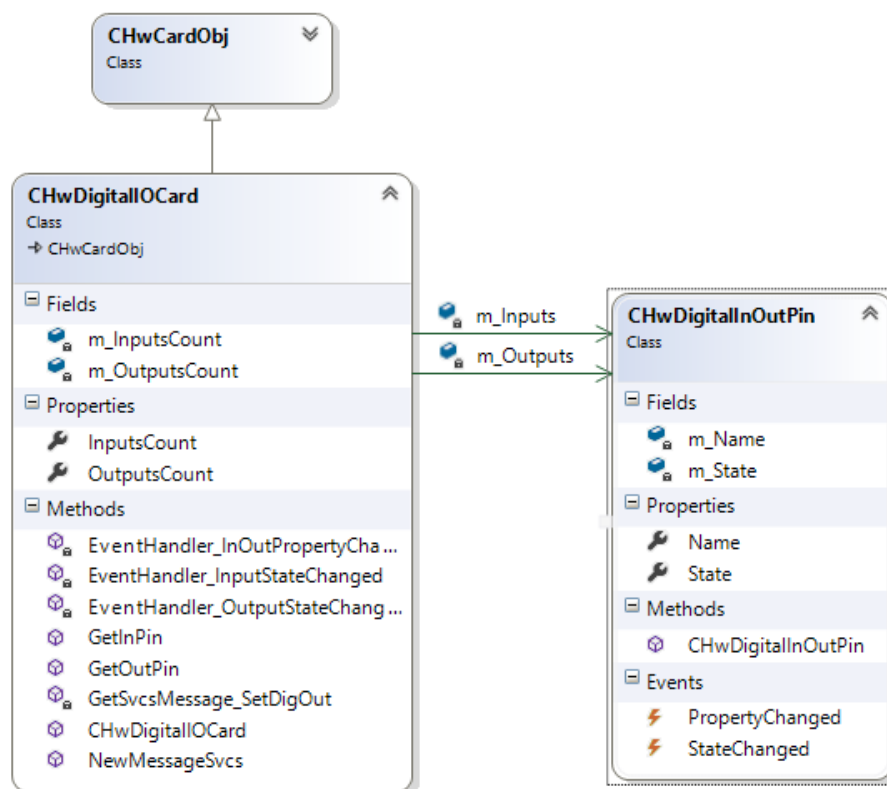
Pro jedinečnou identifikaci modulů je využívána třída *CCardKey*, která implementuje rozhraní *IComparable*. To zajišťuje možnost identifikace modulu ve slovníkové kolekci i seřazení modulů na hlavním formuláři. Třída rovněž přepisuje metody *Equals* a *GetHashCode*, které jsou využívány k vytváření kolekcí dat pro grafy.



Obr. 25: Diagram třídy klíče modulu

7 MODUL DIGITÁLNÍCH VSTUPŮ A VÝSTUPŮ

Modul je reprezentován třídou *CHwDigitalIOCard*, která přijímá v konstruktoru parametry definující počet vstupů a výstupů. V současné době jsou moduly vytvářeny s pevným počtem 6 vstupů a 8 výstupů. Třída poskytuje metody pro získání vstupního nebo výstupního pinu, pomocí nichž je možné piny číst a nastavovat. Při změně výstupního pinu dojde k vyvolání události, kterou přijme objekt modulu. Ten vytvoří zprávu a pomocí události ji předá manažeru, který zajistí její vyslání. Z hlediska funkce umožňuje modul pouze dvě základní věci - čtení stavu vstupů a nastavení výstupů. Celkem je pro komunikaci s modulem využíváno 5 příkazů.



Obr. 26: Diagram třídy modulu digitálních vstupů a výstupů

7.1 Podporované příkazy

7.1.1 Detekce modulu

Příkaz je používán v režimu master pro inicializaci komunikace s modulem a pro ověření jeho přítomnosti na sběrnici. V odpovědi na detekci se nachází číslo verze připojeného modulu.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x12 (Detekce)

Tab. 6: Struktura požadavku detekce modulu digitálních vstupů a výstupů

Odpověď:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x13 (Detekce OK)
Data	1 byte	0 – 0xFF, verze modulu

Tab. 7: Struktura odpovědi detekce modulu digitálních vstupů a výstupů

7.1.2 Čtení stavu vstupů

V režimu master je příkaz pro čtení vstupů předáván manažerem jako periodický požadavek, který je komunikačním jádrem neustále vysílán. Odpověď obsahuje jeden bajt, kde stavy vstupů odpovídají jednotlivým bitům. Nejnižší bit představuje první vstup.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x15 (Čtení vstupů)

Tab. 8: Struktura požadavku čtení digitálních vstupů

Odpověď:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x15 (Čtení vstupů)
Data	1 byte	0 – 0x3F, 1. vstup na nejnižším bitu

Tab. 9: Struktura odpovědi čtení digitálních vstupů

7.1.3 Nastavení výstupů

Příkaz pro nastavení výstupů je vysílán v režimu master pouze při požadavku na změnu výstupu. Požadovaný stav výstupů je prezentován jedním bajtem, kde nejnižší bit představuje výstup číslo 1. Po zpracování požadavku modul odpoví potvrzením.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x16 (Zápis výstupu)
Data	1 byte	0 – 0xFF, 1. výstup na nejnižším bitu

Tab. 10: Struktura požadavku nastavení digitálních výstupů

Odpověď:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x14 (Potvrzení)

Tab. 11: Struktura odpovědi nastavení digitálních výstupů

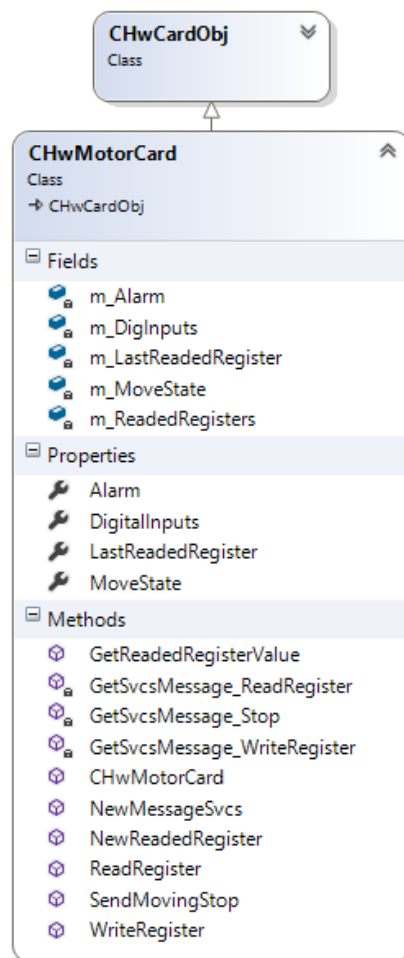
7.2 Uživatelský formulář

Modul nepodporuje žádné speciální nastavení či kalibrace. Uživatelský formulář pro ovládání je proto velmi jednoduchý. Obsahuje dva sloupce zaškrťovacích políček, kde v levém sloupci jsou v reálném čase zobrazovány stavy digitálních vstupů a v pravém je možno ovládat digitální výstupy. Obsah okna je vytvářen dynamicky a do budoucna je počítáno s různým počtem vstupů i výstupů. Samotný formulář je vytvářen až po stisku tlačítka v hlavním formuláři a je možné mít otevřený libovolný počet oken.

Obr. 27: Formulář modulu digitálních vstupů a výstupů

8 MODUL ŘÍZENÍ MOTORŮ

Modul je široce využitelný pro řízení krokových i stejnosměrných motorů. S tím souvisí i množství registrů (parametrů), které lze po sériové lince nastavit. Každý registr má své unikátní číslo a jejich změnou lze modul konfigurovat pro dané použití i jej řídit. Seznam všech podporovaných registrů prezentuje enum *EMotorCardSvcsRegisters*. Implementace modulu je pomocí třídy *CHwMotorCard*. Třída poskytuje informace o aktuálním pohybu, alarmovém hlášení a stavech vstupů. Dále poskytuje metody pro čtení a zápis registrů.



Obr. 28: Diagram třídy modulu řízení motorů

8.1 Konfigurační registry

8.1.1 Registry pro konfiguraci krokového motoru

Registr	Identifikátor
Počet kroků na otáčku	0x01
Maximální dovolená rychlost [RPM]	0x02
Minimální dovolená rychlost [RPM]	0x03
Akcelerace [RPM/sec]	0x04

Tab. 12: Registry pro konfiguraci krokového motoru

8.1.2 Registry pro konfiguraci DC motoru

Registr	Identifikátor
Počet pulzů na otáčku	0x05
Maximální dovolená rychlost v ot/min	0x06
Minimální dovolená rychlost v ot/min	0x07
Otáčky motoru při plném napětí v ot/min	0x08
Napětí tachogenerátoru při plných otáčkách motoru	0x09
Napětí tachogenerátoru při minimálních otáčkách motoru	0x0A
Napětí tachogenerátoru při nulových otáčkách motoru	0x0B
Akcelerace v ot/min/sec	0x0C
P konstanta regulátoru řízení otáček	0x0D
I konstanta regulátoru řízení otáček	0x0E
D konstanta regulátoru řízení otáček	0x0F

Tab. 13: Registry pro konfiguraci DC motoru

8.1.3 Registry pro konfiguraci modulu

Registr	Identifikátor
Údaj AD převodníku měřící výkonové napájení při 24V	0x10
Údaj AD převodníku měřící výkonové napájení při 0V	0x11
Teplota sepnutí ventilátoru	0x12
Teplota odstavení výkonového stupně	0x13

Tab. 14: Registry pro konfigurace modulu

8.1.4 Stavové a ovládací registry

Registr	Identifikátor
Teplota výkonového stupně	0x21
Údaj AD převodníku měřící výkonové napájení	0x22
Napájecí napětí ve voltech	0x23
Údaj AD převodníku měřící napětí z tachogenerátoru	0x24
Čtení: Aktuální rychlost v ot/min Zápis: Rychlost budoucího pohybu	0x25
Čtení: Aktuální pozice Zápis: Cílová pozice – motor se začne otáčet	0x26
Akcelerace budoucího pohybu	0x27
Spojka – vypne/zapne buzení motoru	0x29
Ovládání digitálních výstupů	0x2C
Stav modulu - pohyb, vstupy a signalizace alarmu	0x2E

Tab. 15: Stavové a ovládací registry

8.2 Podporované příkazy

8.2.1 Detekce modulu

Standardně je příkaz používán v režimu master pro inicializaci komunikace. Dále je používán po dokončení aktualizace firmwaru, kdy je nutné ověřit, zda program v modulu běží a komunikuje. V odpovědi se nachází číslo verze připojeného modulu a mód, pro který je modul nastaven. Mód se nastavuje otočným přepínačem na modulu a určuje zařízení, ve kterém se bude modul používat.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x4B (Detekce)

Tab. 16: Struktura požadavku detekce řízení motorů

Odpověď:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x4C (Detekce OK)
Data	1 byte	0 – 0xFF, major verze firmware
Data	1 byte	0 – 0xFF, minor verze firmware

Data	1 byte	0 – 0xFF verze hardware
Data	1 byte	0 – 0xFF mód, 1 – „VTR“, 2 – „SFL“

Tab. 17: Struktura odpovědi detekce řízení motorů

8.2.2 Zápis a čtení registru

Příkaz je používán jak pro zápis, tak i pro čtení registrů. Čtení registru o stavu modulu (0x2E) je manažeru předáváno jako periodický požadavek.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x52 (Registr RW)
Data	4 bytes	Nová hodnota registru (Little-endian)
ID registru	2 bytes	15 bitů – ID Nejvyšší bit 1 - signalizuje zápis, 0 - čtení

Tab. 18: Struktura požadavku zápisu/čtení registru modulu řízení motorů

Odpověď:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x52 (Registr RW)
Data	4 bytes	Čtená/nastavená hodnota registru (Little-endian)
ID registru	2 bytes	15 bitů – ID Nejvyšší bit 1 - signalizuje zápis, 0 - čtení

Tab. 19: Struktura odpovědi zápisu/čtení registru modulu řízení motorů

8.2.3 Příkaz zastavení

Příkaz se používá k okamžitému přerušení právě vykonávaného pohybu.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x50 (Stop)
Data	1 byte	1

Tab. 20: Struktura požadavku zastavení pohybu modulu řízení motorů

Odpověď:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x4D (Potvrzení)

Tab. 21: Struktura odpovědi zastavení pohybu modulu řízení motorů

8.2.4 Příkaz přepnutí do režimu aktualizace

Vysláním příkazu modul ukončí prováděný aplikační program, přepne se do režimu aktualizace a vyčkává na příkazy pro přepis aplikačního programu.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x6F (Reset)

Tab. 22: Struktura požadavku pro přepnutí do režimu aktualizace modulu řízení motorů

Bez odpovědi.

8.2.5 Příkaz aktualizace - zadání počáteční adresy aktualizované stránky

Příkazem modul nastaví počáteční adresu zapisovaných, nebo čtených data aplikačního programu.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x70 (Actualization-Control)
Data	1 byte	3 – zápis od adresy

		2 – čtení od adresy
Data	3 bytes	Adresa (Big-endian)

Tab. 23: Struktura požadavku zadání počáteční adresy aktualizované stránky modulu řízení motorů

Odpověď:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x4D (Potvrzení)

Tab. 24: Struktura odpovědi zadání počáteční adresy aktualizované stránky řízení motorů

8.2.6 Příkaz aktualizace – zápis dat

Příkaz provede zápis osmi bajtů dat aplikačního programu do paměti modulu. Po zápisu modul automaticky posune interní ukazatel zapisovaného místa o 8 bajtů. Proto pro přepis celé stránky paměti souvislými daty stačí zadat pouze počáteční adresu stránky.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x71 (Actualization-Data)
Data	8 bytes	Data aplikačního programu

Tab. 25: Struktura požadavku zápisu dat aplikačního programu modulu řízení motorů

Odpověď:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x4D (Potvrzení)

Tab. 26: Struktura odpovědi zápisu dat aplikačního programu modulu řízení motorů

8.2.7 Příkaz aktualizace – čtení dat

Příkaz je používán pro ověření zapsaného aplikačního programu. V požadavku je definována počáteční adresa stránky paměti a posun od této adresy. V odpovědi modul zašle 8 bajtů dat od definovaného místa.

Požadavek:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x71 (Actualization-Data)
Data	3 bytes	Počáteční adresa čteného stránky paměti (Big-endian)
Data	2 bytes	Posun od počáteční adresy

Tab. 27: Struktura požadavku čtení dat aplikačního programu modulu řízení motorů

Odpověď:

Adresa	1 byte	0 – 0xFF
Příkaz	1 byte	0x4D (Potvrzení)
Data	8 bytes	Data aplikačního programu

Tab. 28: Struktura odpovědi čtení dat aplikačního programu modulu řízení motorů

8.3 Aktualizace firmwaru

Modul řízení motorů disponuje užitečnou funkcí aktualizace firmwaru, bez použití speciálního programátoru a to pomocí sběrnice RS485. Aby byl modul schopen standardními zprávami přepsat svou paměť programu, musí být mimo paměť hlavního aplikačního programu umístěn malý program, do kterého se modul přepne v režimu aktualizace. Základem modulu je mikroprocesor od firmy Microchip, jehož překladač generuje soubory s výsledným firmwarem ve formátu Intel Hex 32.

8.3.1 Formát Intel HEX

Formát Intel Hex je často využíván pro programy, které jsou uloženy v pamětech typu ROM. Jedná se o textový soubor, kde jeden řádek představuje jeden Intel HEX záznam a strojový kód je společně s řídicími znaky zapsán textově jako šestnáctkové číslo. [14]

Každý záznam (řádek) má následující strukturu:

```

:11aaaatt[dd...]cc
:10246200464C5549442050524F46494C4500464C33
|||||CC->Checksum
|||||DD->Data
|||||TT->Record Type
|||AAA->Address
|LL->Record Length
:-->Colon

```

Obr. 29: Struktura záznamu Intel HEX [14]

Délka dat LL udává počet bajtů v záznamu. V poli AAAA je vždy jen spodních 16 bitů adresy záznamu. Vrchních 16 bitů adresy je z počátku procházení souboru rovno 0, dokud se neobjeví záznam, který má v poli TT hodnotu 02 nebo 04. [14]

Záznamy typu 02 rozšiřuje adresu o 4 bity.

```

Address from the data record's address field      2462
Extended segment address record data field      1200
-----
Absolute memory address                          00014462

```

Obr. 30: Záznam Intel HEX typu 02 [14]

Záznam typu 04 rozšiřuje adresu o vrchních 16 bitů.

```

Address from the data record's address field      2462
Extended linear address record data field        FFFF
-----
Absolute-memory address                          FFFF2462

```

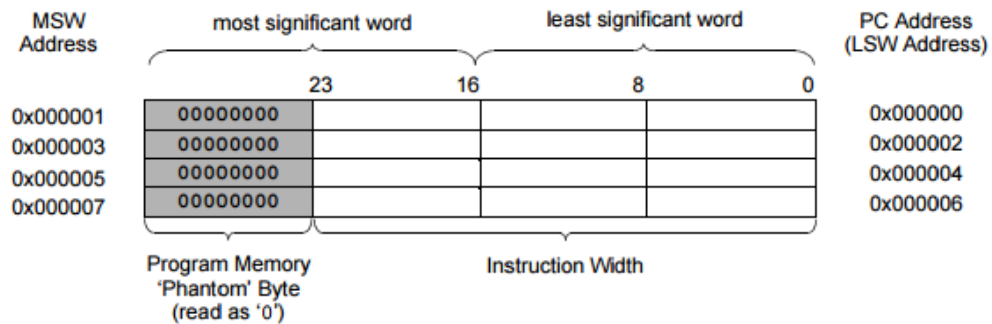
Obr. 31: Záznam Intel HEX typu 04 [14]

Celý záznam je ověřován kontrolním součtem. U bezchybného záznamu musí být modulo 256 ze součtu všech bajtů záznamu rovno 0. [14]

8.3.2 Implementace

Hlavní obslužné rutiny aktualizace jsou implementovány ve třídě *Svcs485Updater*. Třída je navržena tak, aby se dala použít i pro aktualizaci dalších typů modulů. Její instance je vytvořena manažerem modulu, který v konstruktoru předá komunikační příkazy a velikost jedné stránky paměti modulu. Třída poskytuje tři veřejné metody pro zahájení a přerušování aktualizace a metodu, která předává třídě příchozí zprávy. Požadavky na odchozí zprávy jsou předávány nadřazenému objektu pomocí událostí, stejně jako informace o průběhu a dokončení aktualizace.

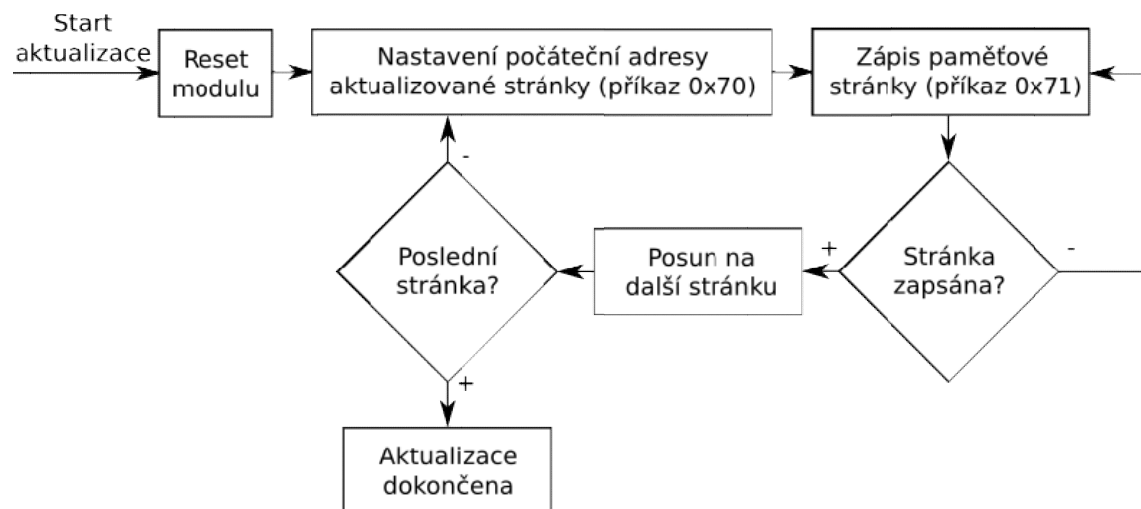
Zahájit aktualizaci je možné voláním metody *StartUpdate* s parametrem cesty k souboru a adresy modulu. Metoda pomocí třídy *IntelHexLoader* vytvoří kolekci záznamů převedených z textové podoby do datové. Poté provede ověření všech záznamů a při úspěšném načtení pokračuje v dalším zpracování. To spočívá v odstranění takzvaných fantom bajtů. Jelikož mikroprocesory PIC používají tři bajtové instrukce a formát Intel Hex počítá se 4 bajtovými instrukcemi, je čtvrtý bajt doplněn nulovým fantom bajtem.



Obr. 32: Organizace paměti programu u PIC [17]

Poslední krok spočívá v úpravě adres jednotlivých záznamů. Mikroprocesory PIC ukládají jednu instrukci na dvě paměťová místa, tedy po zapsání jedné tří bajtové instrukce, je nutné adresný ukazatel posunout pouze o dvě pozice. Formát Intel Hex zvyšuje adresu s každým bajtem, tedy s každou instrukcí posouvá ukazatel 4 krát. Výsledná úprava tedy spočívá ve vydělení adresy dvěma.

Upravené záznamy jsou poté zorganizovány po stránkách o velikosti 1536 bajtů, což se rovná 512 ti instrukcím. Tyto záznamy jsou poté za pomoci background workeru odesílány do modulu.



Obr. 33: Průběh aktualizace firmwaru

8.4 Uživatelský formulář

Formulář umožňuje využít veškeré možnosti modulu a je rozdělen do tří částí. Ve vrchní části jsou zobrazeny stavové informace o modulu, které jsou při zobrazeném formuláři periodicky vyčítány. V druhé části je možné řídit pohyb motoru. Lze zadat konkrétní pozici, nebo zvolit nekonečný pohyb až do zastavení na koncových spínačích. Ve třetí části je možné zapsat kalibrační data a aktualizovat firmware.

Okno je možné zobrazit nezávisle pro více modulů, ovšem aktualizace firmwaru bude dovolena pouze pro jeden modul. Při aktualizaci jsou všechny ovládací prvky na formuláři aktualizovaného modulu deaktivovány.

Řízení motorů

Řízení motorů, Adresa: 5 Adresa: 5

Status

Napětí: 23,5 V, AD: 2716 Vstup tachy: 0 Teplota: 30,4 °C

Pozice: 0 Rychlost: 0,00 RPM

Vstupy 1 2 3 4 5 6

Pohyb: Home pozice Alarm: Žádný

Ovládání

Rychlost [RPM]: 0000,00 Akcelerace [RPM/s]: 0000,00 Pozice: 000000000

Go - home Stop

Go - pozice Go - nekonečno

Spojka Výstup 1 Výstup 2

Kalibrace Aktualizace firmwaru

D:\MicrostepController1.5_v9.0.10.production.hex Firmware

14%

Zápis dat na stránku 0x3800
Zápis dat na stránku 0x3400
Zápis dat na stránku 0x3000
Zápis dat na stránku 0x2C00
Zápis dat na stránku 0x2800
Zápis dat na stránku 0x2400
Zápis dat na stránku 0x2000
Zápis dat na stránku 0x00
Kontrola bootloderu
Přepínání do bootloderu.

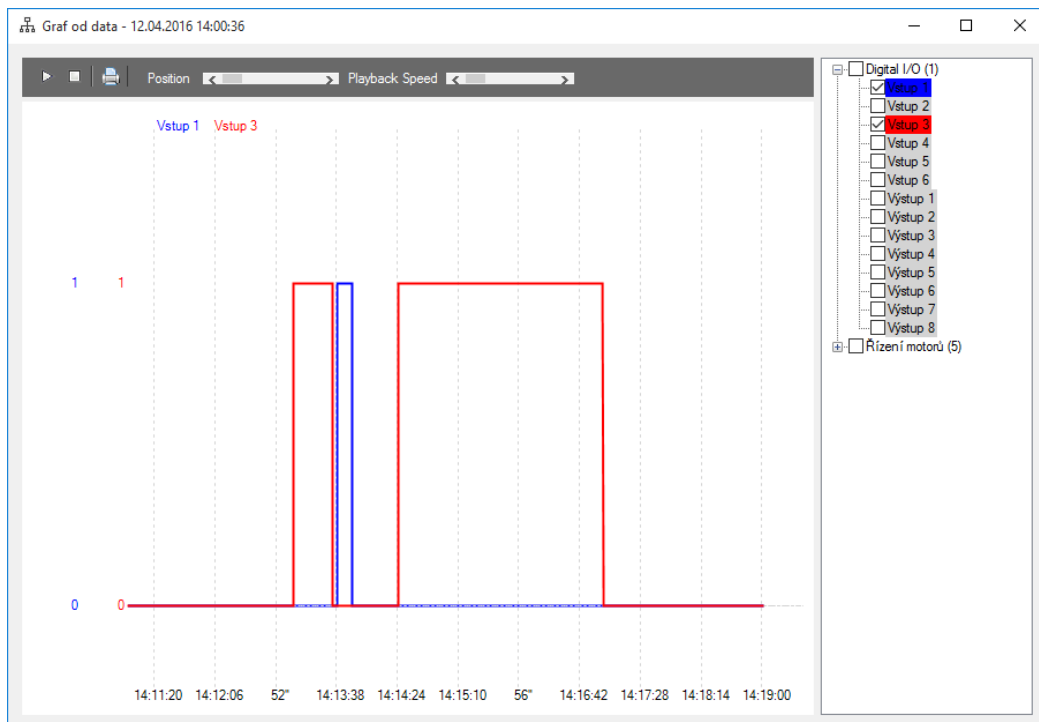
Aktualizuj Stop

Obr. 34: Formulář modulu řízení motorů

9 GRAFY

Pro vyhodnocení měřených dat z modulů, byla do programu implementována možnost zobrazení dat v grafu. Základ tvoří volně dostupná knihovna ze serveru www.codeproject.com nazvaná jako „*A simple C# library for graph plotting*“. [18]

Při výběru knihovny byl základní požadavek na zobrazení více křivek různou barvou v jednom plátně, což uvedená knihovna splňuje. U knihovny není problém určit rozsahy os nebo změnit rozteč mřížky. Toho bylo využito k vylepšení ovládání grafů, protože knihovna implicitně nepodporuje pohyb po ose Y nebo zoom pomocí kolečka myši. K realizaci bylo zapotřebí zasáhnout i do zdrojového kódu knihovny, protože události myši byly zachytávány knihovnou a nebyly předávány dále.



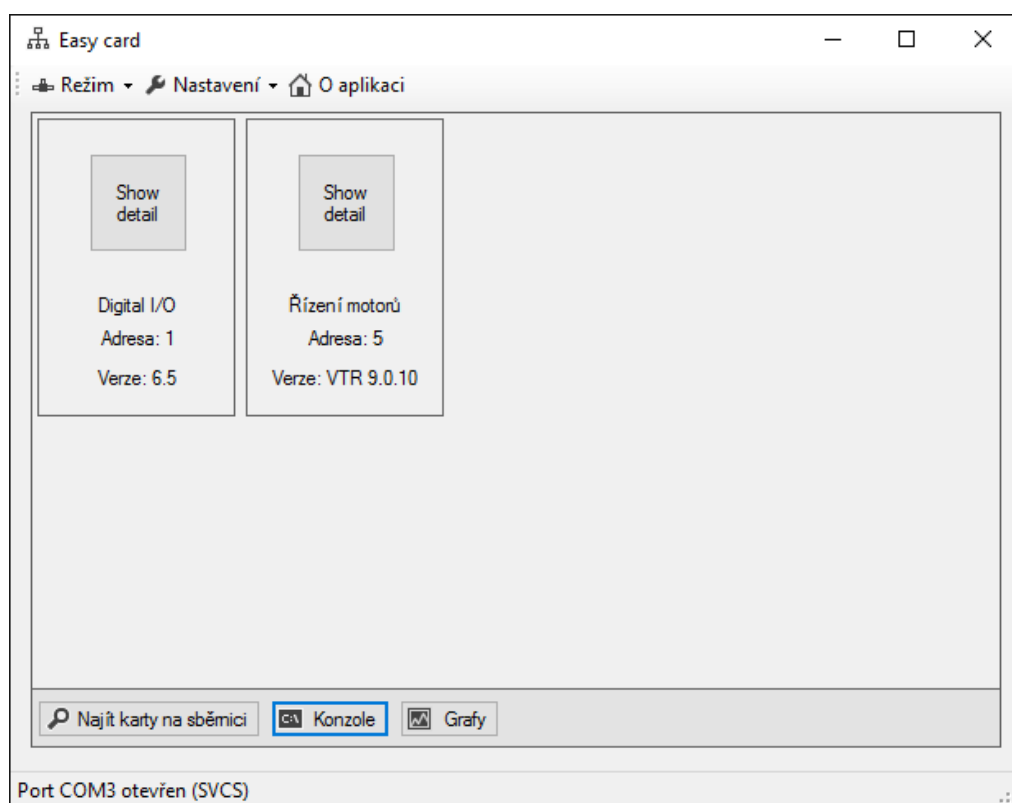
Obr. 35: Formulář grafů

O zásobování grafů daty se stará třída *CGraphDataCollector*. Ta obsahuje vlákno, které každých 100 milisekund snímá vzorky ze všech registrovaných zdrojů. Zdrojem může být objekt implementující rozhraní *ICardDataCollecting*, v tomto případě je to manažer modulů. Rozhraní definuje metody, které zajistí vrácení základního a rozšířeného názvu hodnoty a metodu, která vrací list s jedním vzorkem všech hodnot ze všech modulů daného typu. Tyto vzorky jsou vláknem ukládány do slovníkové kolekce. V současné době nejsou data ukládána na disk a po ukončení programu jsou smazána.

10 HLAVNÍ FORMULÁŘ APLIKACE

Na hlavním formuláři jsou zobrazovány moduly, které byly detekovány při poslední detekci. Každý modul je prezentován panelem s tlačítkem pro otevření okna detailu. Všechny panely jsou umístěny do plovoucího layoutu, aby při různé šířce okna docházelo k přeskupení.

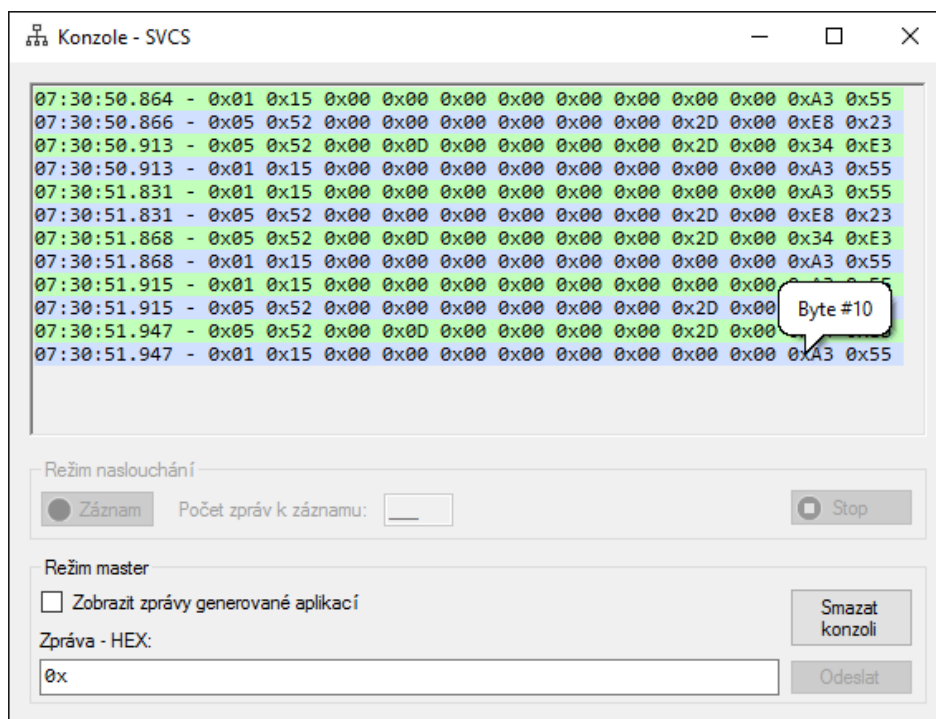
Pomocí horního menu je možné aplikaci přepnout do režimu naslouchání, vybrat port nebo změnit jazyk aplikace. Tlačítka ve spodní části slouží k detekci modulů, zobrazení grafů a zobrazení ladící konzole.



Obr. 36: Hlavní formulář

10.1 Ladící konzole

Pro potřeby naslouchacího režimu a pro vývojové ladění byla do aplikace přidána konzole zaznamenávající veškeré přijaté a odeslané zprávy. V režimu naslouchání konzole slouží k nahrávání komunikace, která je řízena jinou master jednotkou. V režimu master je možné pomocí konzolového řádku vytvořit jakoukoliv zprávu, která bude doplněna kontrolním součtem a vyslána na port.



Obr. 37: Ladicí konzole

ZÁVĚR

V praktické části bakalářské práce byla vytvořena aplikace, která poskytuje prostředky pro řízení a monitorování hardwarových modulů. Hlavní motivací byl problém, kdy při výrobě těchto modulů neexistovala možnost jednoduchého a efektivního testování. Vzhledem k většímu počtu typů modulů, bylo rozhodnuto o prvotním zpracování modulu digitálních vstupů a výstupů, kterých je vyráběno nejvíce a modulu pro řízení motorů, které jsou nejnáročnější na testování.

Během práce byl nejprve vytvořen základ komunikačního jádra a ověřena stabilita propojení mezi počítačem a moduly za pomoci USB převodníku. Při testování byly zaznamenány občasné výpadky zpráv, což bylo způsobeno nesprávným propojením zemnicích vodičů na straně redukce. Po odhalení chyby, se připojení jevílo stabilní a bez výpadku zpráv i na delším vedení. Dále bylo nutné se hlouběji seznámit s frameworkem .NET a jazykem C#. Postupně bylo navrženo GUI aplikace a po vyřešení drobných problémů s vlákny a předáváním informací mezi nimi, bylo úspěšně dokončeno i komunikační jádro.

Výsledkem je aplikace pro Windows spustitelná na běžných počítačích s frameworkem .NET verze 4 a vyšší, která dokáže využít veškerých možností připojených modulů. Jejím prostřednictvím je možné provést konfigurace nově vyrobených modulů i poslouchat sběrnici v existujících zařízeních. Aplikace přidává možnost zobrazení veličin v grafech a umožňuje přepnutí do českého nebo anglického jazyka.

Vyvinutá aplikace je již testována oddělením vývoje hardwaru, a pokud se nevyskytnou závažnější problémy, dojde k rozšíření o další moduly. Výhledově je plánován vývoj nové generace modulů s novým komunikačním protokolem. Bude-li aplikace u současné generace přínosem, bude u té příští hlavním a jediným prostředkem pro konfigurace a kalibrace.

SEZNAM POUŽITÉ LITERATURY

1. TROELSEN, Andrew W. *C# a .NET 2.0 profesionálně*. Brno : Zoner Press, 2006. 80-86815-42-0.
2. HERCEG, Tomáš. dotNETportal.cz. *Úvod do .net frameworku*. [Online] 3. 4 2009. [Citace: 23. 1 2016.] <http://www.dotnetportal.cz/clanek/125/Uvod-do-NET-Frameworku>.
3. MSDN Microsoft . *Introduction to the C# Language and the .NET Framework*. [Online] [Citace: 24. 1 2016.] <https://msdn.microsoft.com/cs-cz/library/z1zx9t92.aspx>.
4. Visual Studio. *Visual Studio Community*. [Online] [Citace: 24. 1 2016.] <https://www.visualstudio.com/products/visual-studio-community-vs>.
5. HRUŠKA, František. *Technické prostředky integrované automatizace*. Zlín : Univerzita Tomaše Bati, 2012. 978-80-7454-234-3.
6. POUCHA, Pavel. hw.cz. *Přenos dat po linkách RS485 a RS422*. [Online] 25. 8 1999. [Citace: 26. 1 2016.] <http://vyvoj.hw.cz/teorie-a-praxe/dokumentace/prenos-dat-po-linkach-rs485-a-rs422.html>.
7. TIŠNOVSKÝ, Pavel. ROOT.CZ. *Sběrnice RS-422, RS-423 a RS-485*. [Online] 18. 12 2008. [Citace: 26. 1 2016.] <http://www.root.cz/clanky/sbernice-rs-422-rs-423-a-rs-485/>.
8. MSDN Microsoft. *Verze a závislosti rozhraní .NET Framework*. [Online] [Citace: 29. 1 2016.] [https://msdn.microsoft.com/cs-cz/library/bb822049\(v=vs.110\).aspx](https://msdn.microsoft.com/cs-cz/library/bb822049(v=vs.110).aspx).
9. ŠTORC, Ondřej. ITnetwork.cz. *Historie .NETu*. [Online] [Citace: 29. 1 2016.] <http://www.itnetwork.cz/csharp/historie-dotnetu>.
10. FTDI chip. *FT232R USB UART IC Datasheet*. [Online] [Citace: 1. 2 2016.] http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.
11. OLMR, Vít. hw.cz. *Terminal – sériový terminál pro Windows*. [Online] [Citace: 2. 2 2016.] <http://vyvoj.hw.cz/teorie-a-praxe/terminal-seriovy-terminal-pro-windows.html>.
12. MRÁZEK, Oldřich. hw.cz. *Hercules – TCP, UDP a sériový terminál pro Windows*. [Online] 24. 9 2004. [Citace: 2. 2 2016.] <http://vyvoj.hw.cz/produkty/hercules-tcp-udp-a-seriovy-terminal-pro-windows.html>.
13. ALBAHARI, Joseph a Ben. *C# 3.0 in a nutshell. 3rd ed.* Sebastopol, Calif. : O'Reilly, 2007. 0596527578.

14. ARMKEIL: Microcontroller tools. *GENERAL: INTEL HEX FILE FORMAT*. [Online] [Citace: 10. 3 2016.] <http://www.keil.com/support/docs/1584.htm>.
15. Gook, Michael. *Hardwarová rozhraní*. Brno : Computer Press, 2006. 80-251-1019-2.
16. Integrity Instruments. *What is the "RS" in RS232/RS485/RS422?* [Online] [Citace: 9. 4 2016.] <http://www.rs-485.com/comspec.html>.
17. Microchip. *dsPIC33E/PIC24E Program Memory*. [Online] [Citace: 11. 4 2016.] <http://ww1.microchip.com/downloads/en/DeviceDoc/70000613d.pdf>.
18. Zimmermann, Stephan. CODE PROJECT. *A simple C# library for graph plotting*. [Online] 4. 9 2014. [Citace: 25. 4 2016.] <http://www.codeproject.com/Articles/32836/A-simple-C-library-for-graph-plotting>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AIn	Analog Input
CLR	Common Language Runtime
CLS	Common Language System
COM	Communication Port
CPLD	Complex Programmable Logic Device
CTS	Common Type System
DOut	Digital Output (DO)
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIFO	First In First Out
FPGA	Field Programmable Gate Array
HTML	HyperText Markup Language
IL	Intermediate Language
LINQ	Language Integrated Query
SSH	Secure Shel
TCP/IP	Transmission Control Protocol/Internet Protocol
TELNET	Telecommunication Network
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
uP	Mikroprocesor

SEZNAM OBRÁZKŮ

Obr. 1: Blokové schéma modulu digitálních vstupů a výstupů	12
Obr. 2: Blokové schéma modulu řízení motorů.....	13
Obr. 3: Zapojení konektoru DB9P	15
Obr. 4: Poměr rychlosti přenosu a délky propojení [5]	15
Obr. 5: Schéma dvou vodičového propojení RS485[5].....	16
Obr. 6: Synchronizace vysílače a přijímače [7].....	17
Obr. 7: Ukázka utility Putty	18
Obr. 8: Ukázka utility Hercules	19
Obr. 9: Ukázka utility Terminal.....	19
Obr. 10: Ukázka utility Terminal – graf	20
Obr. 11: Struktura .NET [1].....	21
Obr. 12: Kompilace v .NET frameworku [3].....	22
Obr. 13: Hello World v C#	25
Obr. 14: Použití synchronizace pomocí AutoResetEvent [13].....	26
Obr. 15: Příklad zapojení převodníku USB na RS485[10].....	29
Obr. 16: Zapojení kabelu PFL20 – CAN9	30
Obr. 17: Propojení a adresace modulů.....	30
Obr. 18: Struktura zprávy	31
Obr. 19: Algoritmus pro výpočet kontrolního součtu.....	31
Obr. 20: Zjednodušený diagram tříd komunikačního jádra	33
Obr. 21: Diagram rodičovské třídy modulů.....	34
Obr. 22: Metoda pro vyhledání modulů v kolekci.....	35
Obr. 23: Diagram rodičovské třídy manažeru modulů	35
Obr. 24: Kolekce detekovaných modulů	36
Obr. 25: Diagram třídy klíče modulu.....	36
Obr. 26: Diagram třídy modulu digitálních vstupů a výstupů	37
Obr. 27: Formulář modulu digitálních vstupů a výstupů.....	39
Obr. 28: Diagram třídy modulu řízení motorů.....	40
Obr. 29: Struktura záznamu Intel HEX [14].....	47
Obr. 30: Záznam Intel HEX typu 02 [14].....	47
Obr. 31: Záznam Intel HEX typu 04 [14].....	47
Obr. 32: Organizace paměti programu u PIC [17].....	48

Obr. 33: Průběh aktualizace firmwaru	48
Obr. 34: Formulář modulu řízení motorů	49
Obr. 35: Formulář grafů.....	50
Obr. 36: Hlavní formulář	51
Obr. 37: Ladící konzole	52

SEZNAM TABULEK

Tab. 1: Parametry modulu digitálních vstupů a výstupů	11
Tab. 2: Parametry modulu řízení motorů.....	12
Tab. 3: Signály RS-232C konektoru DB9P[15]	14
Tab. 4: Srovnání RS232C, RS422 a RS485[15][16]	17
Tab. 5: Verze .NET frameworku [8][9].....	24
Tab. 6: Struktura požadavku detekce modulu digitálních vstupů a výstupů	38
Tab. 7: Struktura odpovědi detekce modulu digitálních vstupů.....	38
Tab. 8: Struktura požadavku čtení digitálních vstupů	38
Tab. 9: Struktura odpovědi čtení digitálních vstupů.....	38
Tab. 10: Struktura požadavku nastavení digitálních výstupů	39
Tab. 11: Struktura odpovědi nastavení digitálních výstupů	39
Tab. 12: Registry pro konfiguraci krokového motoru	41
Tab. 13: Registry pro konfiguraci DC motoru.....	41
Tab. 14: Registry pro konfigurace modulu.....	41
Tab. 15: Stavové a ovládací registry.....	42
Tab. 16: Struktura požadavku detekce řízení motorů.....	42
Tab. 17: Struktura odpovědi detekce řízení motorů	43
Tab. 18: Struktura požadavku zápisu/čtení registru modulu řízení motorů.....	43
Tab. 19: Struktura odpovědi zápisu/čtení registru modulu řízení motorů	43
Tab. 20: Struktura požadavku zastavení pohybu modulu řízení motorů	44
Tab. 21: Struktura odpovědi zastavení pohybu modulu řízení motorů.....	44
Tab. 22: Struktura požadavku pro přepnutí do režimu aktualizace modulu řízení motorů	44
Tab. 23: Struktura požadavku zadání počáteční adresy aktualizované stránky modulu řízení motorů	45
Tab. 24: Struktura odpovědi zadání počáteční adresy aktualizované stránky řízení motorů	45
Tab. 25: Struktura požadavku zápisu dat aplikačního programu modulu řízení motorů.....	45
Tab. 26: Struktura odpovědi zápisu dat aplikačního programu modulu řízení motorů	45
Tab. 27: Struktura požadavku čtení dat aplikačního programu modulu řízení motorů	46
Tab. 28: Struktura odpovědi čtení dat aplikačního programu modulu řízení motorů.....	46

SEZNAM PŘÍLOH

PŘÍLOHA PI: CD SE ZDROJOVÝMI KÓDY APLIKACE

PŘÍLOHA P I: OBSAH PŘILOŽENÉHO CD

Přiložené CD obsahuje:

BpEasyCard	Zdrojové kódy aplikace
Dokumentace	Dokumentace zdrojových kódů
fulltext.pdf	Bakalářská práce