Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Monitorovací program sběrnice FOUNDATION Fieldbus

Plzeň, 2013 Martin Šlapa

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin ŠLAPA**
Osobní číslo:	**A09B0561P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Výpočetní technika**
Název tématu:	**Monitorovací program sběrnice FOUNDATION Fieldbus**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování:

1. Prostudujte protokol sběrnice FOUNDATION Fieldbus.
2. Seznamte se s výsledky diplomové práce Pavla Böhma: "Komunikační modul pro průmyslovou sběrnici FOUNDATION Fieldbus".
3. Navrhněte a odlaďte programovací vybavení komunikačního modulu pro zachycení základní komunikace sběrnice FOUNDATION Fieldbus.
4. Navrhněte a odlaďte program umožňující monitorovat provoz na sběrnici FOUNDATION Fieldbus pro PC.
5. Proveďte základní testy vytvořeného programovacího vybavení.

Rozsah grafických prací:	**dle potřeby**
Rozsah pracovní zprávy:	**doporuč. 30 s. původního textu**
Forma zpracování bakalářské práce:	**tištěná**
Seznam odborné literatury:	

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce:	**Doc. Ing. Vlastimil Vavřička, CSc.** Katedra informatiky a výpočetní techniky
Datum zadání bakalářské práce:	**15. října 2012**
Termín odevzdání bakalářské práce:	**10. května 2013**

Doc. Ing. František Vávra, CSc.
děkan

L.S.

Prof. Ing. Jiří Šafařík, CSc.
vedoucí katedry

V Plzni dne 18. října 2012

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 10. 5. 2013 .....

Martin Šlapa

Abstract

FOUNDATION Fieldbus monitoring system

This bachelor thesis describes FOUNDATION fieldbus and options of capturing it's communication.

The first task was to develop control program in C language for used hardware module. The module is equipped with a 32-bit microcontroller based on the ARM architecture. The module also contains circuits for communication on the bus. The main task of the module is forwarding communication in both directions between FOUNDATION Fieldbus and USB.

The second task was to create a program in C # for the PC for capturing, displaying, filtering and exporting data rerouted from a hardware module.

At the end the communication module was tested in an experimental environment, along with the monitoring program.

Obsah

1. Úvod

Průmyslové sběrnice jsou v dnešním světě nedílnou součástí pro sběr, distribuci a vyhodnocení dat nejrůznějších informačních charakterů. Jejich hlavním úkolem je především měření fyzikálních veličin pomocí senzorových systémů, řízení výrobních procesů (například automatizační linky), řízení a distribuce energie, telekomunikaci nebo využití v moderní domácnosti.

Bakalářská práce se zaměřuje na problematiku a monitorování průmyslové sběrnice FOUNDATION Fieldbus. Vysvětluje základní myšlenku komunikace na sběrnici a popisuje jednotlivé vrstvy, které jsou definovány.

Cílem bakalářské práce je vytvořit program umožňující na PC monitorovat tok dat sběrnice FOUNDATION Fieldbus s využitím komunikačního modulu, který navrhnul Pavel Böhm v rámci jeho Diplomové práce. Komunikační modul je zařízení pro mnohé využití při komunikaci na sběrnici FOUNDATION Fieldbus, proto je nutné jej programově přizpůsobit pro monitorování sběrnice a komunikaci s monitorovacím programem na PC.

V dnešní době je časté propojení zařízení pomocí USB a díky podpoře této sběrnice v komunikačním modulu nebyl důvod sběrnici nevyužít při komunikaci mezi modulem a PC.

Pro otestování programového vybavení komunikačního modulu je sestaven experimentální segment sběrnice, na kterém je ověřena jeho správná funkčnost. Testování monitorovacího programu je postaveno na procesorové a paměťové náročnosti a rychlosti odezvy při různých operacích.

2. Teoretická část

2.1. FOUNDATION Fieldbus

FOUNDATION Fieldbus (FF) je průmyslová sběrnice vyvinutá stejnojmennou nezávislou organizací, kladoucí si za cíl stabilizovat oblast průmyslových sběrnic v automatizačním a výrobním procesu. Počátky vývoje sahají až do roku 1970, avšak přijata jako standard významnými standardizačními organizacemi, byla až o třicet let později. Organizace sdružuje významné společnosti zabývající se vývojem pro automatizační a regulační techniku. Tato zařízení organizace ověřuje a registruje jako kompatibilní s FF. Specifikace sběrnice je kompatibilní s projektem ISA SP50 a je součástí standardu IEC 61158. [1] [2]

Hlavní rysy sběrnice FF [3]:

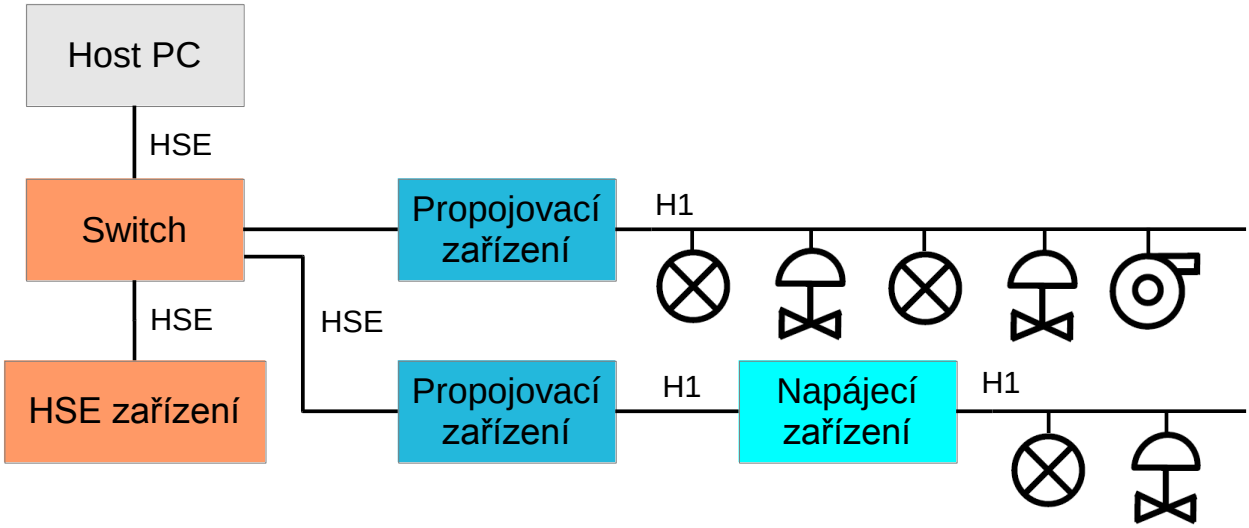
- Digitální sběrnice se sériovým poloduplexním přenosem, která je určená pro komunikaci mezi ovládacími a regulačními prvky a řídícími automaty,
- lze použít původní kabeláž při instalaci bez porušení základních výhod dosavadních komunikačních systémů s proudovou smyčkou 4 - 20 mA,
- standardizované fyzické rozhraní, napájení koncových zařízení po komunikačním vedení, aplikovatelnost ve výbušném výrobním prostředí.

A také dále zavádí nové vlastnosti:

- Obousměrný přenos více parametrů po páru vodičů (proudová smyčka přenáší jednu veličinu),
- úspora rozvodů, na jeden pár vodičů může být připojeno více jednotek,
- schopnost funkční diagnostiky jednotek a jejich propojení,
- rychlá informace o výjimečných stavech,
- implementace principů distribuovaného řízení snižuje nároky na výkonnost a počet řídících terminálů.

2.1.1. Komunikační protokoly

FOUNDATION Fieldbus (FF) je definován dvěma komunikačními protokoly. První z nich se nazývá H1 a je určen pro propojení průmyslových prvků, mezi něž lze zařadit např. senzory, převodníky a akční členy. Druhým protokolem je HSE (High Speed Ethernet), jehož hlavním využitím je vysokorychlostní propojení pracovních stanic, serverů a H1 subsystémů. Příkladné zapojení a provázání obou komunikačních protokolů lze vidět na nadcházejícím obrázku (Obrázek 1). [1] [2] [3]



Obrázek 1: Příklad zapojení sběrnice [1]

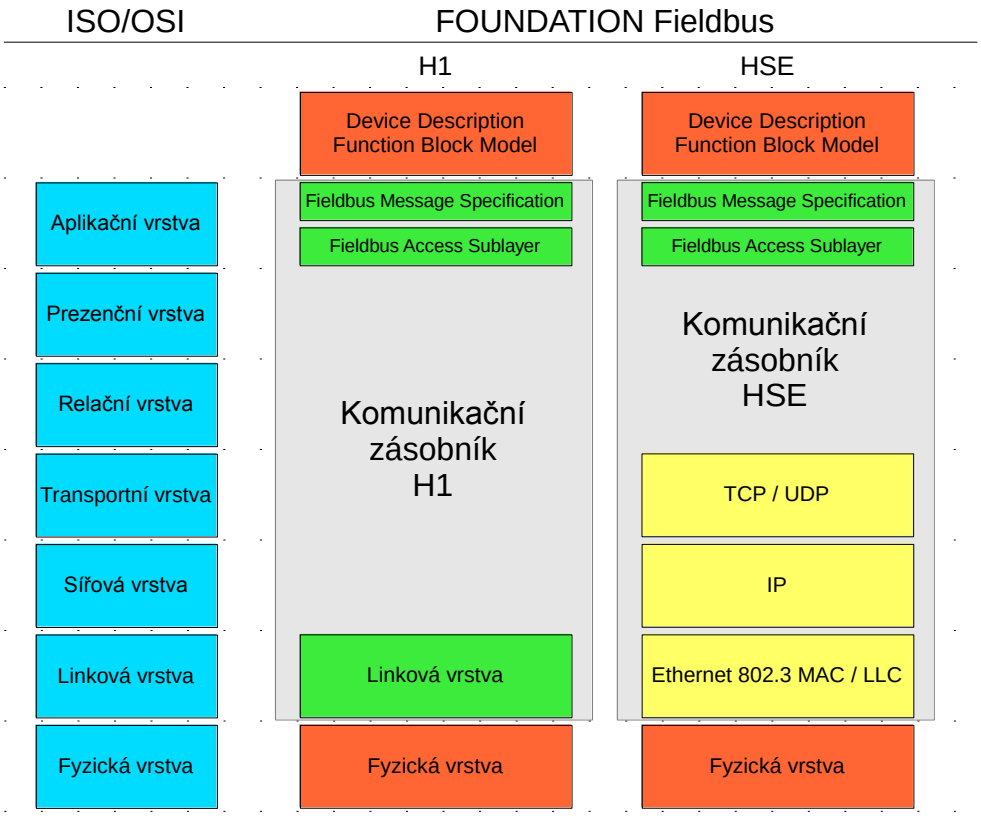
2.1.2. Model

Základní model, sloužící ke komunikaci na sběrnici FOUNDATION Fieldbus (FF), se skládá ze tří základních vrstev:

- **fyzická vrstva** *(Physical Layer)*
- **komunikační zásobník** *(Communication Stack)*
- **uživatelská vrstva** *(User Layer)*

Na rozdíl od referenčního modelu OSI (Open Systems Interconnect model), který má 7 vrstev, se u typu H1 vrstvy 3 - 6 nevyužívají. Je ale přidána nová vrstva v uživatelské části, jejíž součástí jsou i bloky pro řízení průmyslové aplikace.

Komunikační zásobník ještě obsahuje několik zapouzdřených vrstev, které se liší v konkrétním typu sběrnice (H1 nebo HSE). Porovnání jednotlivých modelů OSI a obou typů FF lze vidět na následujícím obrázku (Obrázek 2). [1] [4]



Obrázek 2: Porovnání modelů ISO/OSI, H1 a HSE

2.1.3. Fyzická vrstva

Úkolem fyzické vrstvy *(Physical Layer)* je připravit data z vyšších vrstev na přenos po komunikačním médiu. Aby bylo možné data správně zachytit v dalších uzlech, přidává vrstva několik synchronizačních značek:

- **synchronizační hlavičku** *(Preamble)*, jejímž úkolem je synchronizovat hodiny přijímače s vysílačem
- **začátek rámce** *(Start frame)*, jehož úkolem je ohraničení začátku posílaných dat
- **konec rámce** *(End frame)*, jehož úkolem je ohraničení konce posílaných dat

Dále se ve fyzické vrstvě specifikují parametry pro kódování, napěťové úrovně, napájení atd.

H1 používá synchronní kódování Manchaster, jehož hodinový i datový signál je sloučen do jednoho. Rychlost přenosu je 31,25 kbps a napájení na vodičích je v rozmezí 9 - 32 V. Přesně použité napájení vodičů H1 se odvíjí z bezpečnosti prostředí.

Rychlost přenosu u HSE je 100 / 1000 Mbit/s a vychází z limitace ze specifikací ethernetu. [1] [4] [5]

2.1.4. Linková vrstva

Linková vrstva (Data Link Layer) má za úkol rozdělovat data do rámců a předávat je nižší vrstvě pro přenos po fyzickém médiu dalšímu uzlu. Dále provádí kontrolu přijatých dat a řídí jednotlivé přístupy ke sběrnici.

Přístupy ke sběrnici jsou naplánované aktivním zařízením typu Link Master (LM, Link Active Scheduler - LAS) na každém segmentu. Pro řízení komunikace je využíván *token*, který se cyklicky předává mezi jednotlivé zařízení v daném segmentu sběrnice.

Link Master zvolí LAS zařízení, které se první ohlásí a jehož adresa pak vždy bude *0x04* nebo nižší.

Adresace jednotlivých zařízení se provádí DL adresou *(Data Link address)*, která se dále rozděluje do tří pod adres:

- **Link** (16 bit)
- **Node** (8 bit)
- **Selector** (8 bit)

Link je pro adresaci jednotlivých segmentů, které jsou odděleny mosty. Při komunikaci ve stejném segmentu se adresa vynechává. *Node* se používá pro adresaci daných uzlů v segmentu a adresní rozsah je dán následující tabulkou (Tabulka 1).

Tabulka 1: Adresní rozsahy [1]

0x10 až V(FUN)	rozsah pro Link Master zařízení
V(FUN) + V(NUN) až 0xF7	rozsah pro Basic zařízení
0xF8 až 0xFC	výchozí adresy pro nová zařízení
0xFD až 0xFF	adresy pro dočasné zařízení

V(FUN) a V(NUN) jsou parametry pro adresní mezeru, označující adresy, se kterými se uzel nemůže připojit. *Selector* se používá pro identifikaci koncového spojení v rámci jednoho uzlu, resp. zařízení DLCEP (Data Link Connection End Point). [1] [4] [5]

Na linkové vrstvě rozeznáváme následující typy (viz Tabulka 2) rámců DL PDUs (Data Link Protocol Data Units):

Tabulka 2: Typy rámců linkové vrstvy [6]

Typ rámce	Název	Funkce
EC1, EC2	Establish connection	Připojení DLCEP
DC1, DC2	Disconnect connection	Odpojení
RC1, RC2	Reset connection	Resetování spojení
CA1, CA2	Compel acknowledgement	Výzva k přenosu dat DLS uživatelů
CD1, CD2	Compel data	Výzva publisheru
ED1, ED2	Exchange data	Přenos dat DLS uživatelů
DT1, ..., DT5	Data transfer	Odeslání datové jednotky
SR	Status response	Odpověď na stav LM
CT	Compel time	Vynucení synchronizace času
TD	Time distribution	Synchronizace času
RQ	Round-trip time query	Měření času CT - dotaz
RR	Round-trip time reponce	Měření času CT - odpověď
PN	Probe node	Vyhledávání nových uzlů
PR	Probe response	Odpověď nového uzlu
PT	Pass token	Poskytnutí tokenu
ES	Execute sequence	Předání pravomocí tokenu
RT	Return token	Vrácení tokenu
RI	Request interval	Žádost o více tokenů
CL	Claim LAS	Ohlášení LAS
TL	Transfer LAS	Žádost o LAS
WK	Wakeup	Probuzení komunikace
IDLE	Idle	Neaktivita

2.1.5. Aplikační vrstva

Aplikační vrstva *(Application Layer)* se rozděluje na dvě podvrstvy:

- **Fieldbus Access Sublayer** *(FAS)*
- **Fieldbus Message Specification** *(FMS)*

Podvrstva FAS zajišťuje tři typy virtuálních komunikačních vztahů (Virtual Communication Relationship, VCR):

- **Client / Server** - spojení slouží pro nastavování zařízení operátorem

- **Report Distribution** - slouží k rozesílání událostí a alarmů
- **Publisher / Subscriber** - komunikace pro rozesílání dat

Přehled možností jednotlivých typů VCR uvádím v následující tabulce (Tabulka3).

Tabulka 3: Typy VCR [1]

Client / Server	Report Distribution	Publisher / Subscriber
Změny požadovaných hodnot Změny módů Ladění Upload / Download Nastavení alarmů Nastavení pohledů Vzdálená diagnostika	Zasílání alarmových hlášení Zasílání dat pro vytváření histogramů	Zasílání regulačních údajů jiným zařízení nebo operátorům

Úkolem FMS je vytvoření struktury zprávy pro přenos po sběrnici z dat obdržených od uživatelské vrstvy. Dále FMS poskytuje služby pro síťovou komunikaci a připravuje data ke správné interpretaci pro správné použití v jiném zařízení. [1] [4] [5]

V rámci jednoho fyzického zařízení může být více virtuálních zařízení (Virtual Field Device, VFD), ve kterých jsou obsaženy jejich jedinečné funkce. Umožňují tak oddělit více funkcí jednoho zařízení, které spolu ale jinak nesouvisejí. Běžně se vyskytují dvě VFD. Jedno je pro správu *(Network Management a System Management)* a druhé vykonává některou konkrétní funkci zařízení.

- **Network Management** - správa adres a tagů zařízení, řízení distribuce času a plánování komunikace, řízení systému v nepřirozené situaci (špatná konfigurace, selhání apod.)
- **System Management** - zprostředkování čtení a zápisů objektů komunikačního zásobníku prostřednictvím sběrnice FF

FMS také spravuje objekty, které popisují jednotlivé bloky a parametry ostatních zařízení na sběrnici. Popis všech objektů je zapouzdřen do seznamu objektů (Object Dictionary, OD).

V OD jsou jednotlivé popisky identifikovány indexy. Tyto indexy jsou unikátní v rámci jednoho VFD. Index 0 odkazuje na hlavičku OD. Další indexy do 255 jsou rezervovány pro datové typy a pro popis složitějších datových struktur. Nad 255 jsou indexovány uživatelské služby.

Během komunikace dvou zařízení se VFD správnou interpretaci přenesených hodnot dozví z protějšího OD. [1] [4] [5]

2.1.6. Uživatelská vrstva

Uživatelská vrstva *(User Layer)* zapouzdřuje bloky a objekty obsahující parametry a funkce daných zařízení. Zapouzdřené celky jsou k dispozici pro operátora řídící aplikaci.

V popisu (Device Description, DD) je obsažen seznam všech funkcí, které zařízení podporuje. Uvádí se měřítka, jednotky, jednotlivé parametry, jak interpretovat hodnoty apod. V případě, že v zařízení je obsažena nestandardní funkce, musí zde být bezpodmínečně popsána v plném jejím využití. Pro zápis takové funkce slouží daný popisovací jazyk (Device Description Language, DDL).

V uživatelské vrstvě jsou dále definovány tři základní typy (mimo jiné) bloků:

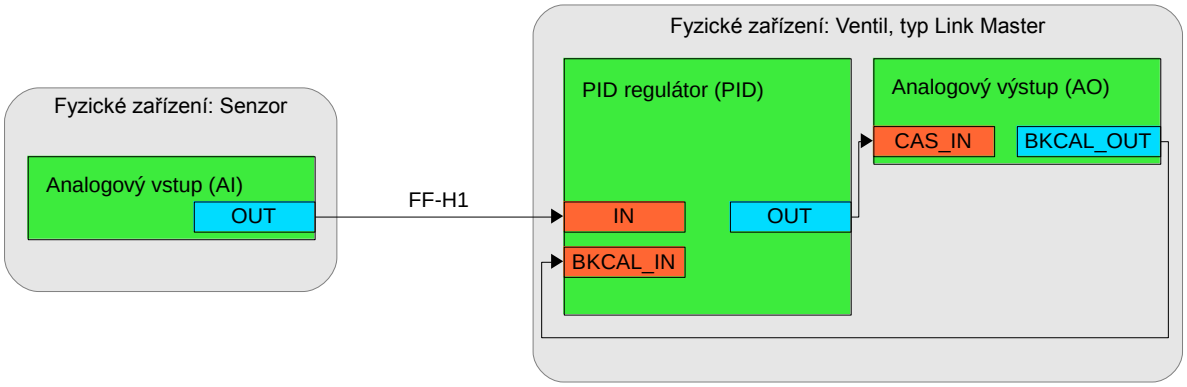
- **zdrojový** *(Resource Block)* - uchovává informace o zařízení, jako je typ, výrobní a sériové číslo, stav ostatních bloků apod.
- **převodní** *(Transducer Block)* - odděluje funkční bloky od fyzického rozhraní akčních členů nebo senzorů
- **funkční** *(Function Block)* - bloky umožňující vytvořit monitorovací nebo řídící aplikaci, kterou zařízení na sběrnici vykonává

Standardní funkční bloky jsou:

AI:	Analog Input	AO:	Analog Output
BG:	Bias / Gain	CS:	Control Selector
DI:	Discrete Input	DO:	Discrete Output
ML:	Manual Loader	PD:	Proportional / Derivate
PID:	Proportional / Integral / Derivate	RA:	Ratio

Propojením těchto bloků a nastavení parametrů v příslušném programu lze jednoduše sestavit řídící aplikaci *(Function Block Application)*. [1]

Příkladná aplikace, která řídí smyčky sestavené ze dvou fyzických zařízení (senzoru a ventilu) a třech funkčních bloků, je vidět na následujícím obrázku (Obrázek 3).



Obrázek 3: Příkladné zapojení řídící smyčky z funkčních bloků [1]

Senzor, je-li dotázán ventilem, navrací hodnotu do PID členu. Tato komunikace probíhá prostřednictvím sběrnice FF. Působení ventilu je zpětně odečítáno do PID členu v rámci fyzického zařízení, tudíž přenos neprobíhá po sběrnici FF.

Identifikace funkčních bloků probíhá pomocí jejich názvů (Tagu) a indexů v seznamu objektů (Object Dictionary).

Další pomocné objekty jsou [1] [4] [5]:

- **odkazovací** *(Link objects)* - definují spojení mezi funkčními bloky
- **zobrazovací** *(View objects)* - definují parametry, které chce uživatel zobrazit
- **poplachové** *(Alert objects)* - definují vytvoření a odeslání alarmové zprávy
- **stavové** *(Trend objects)* - uchovávají hodnoty s časovou značkou pro reprezentaci stavu v grafu

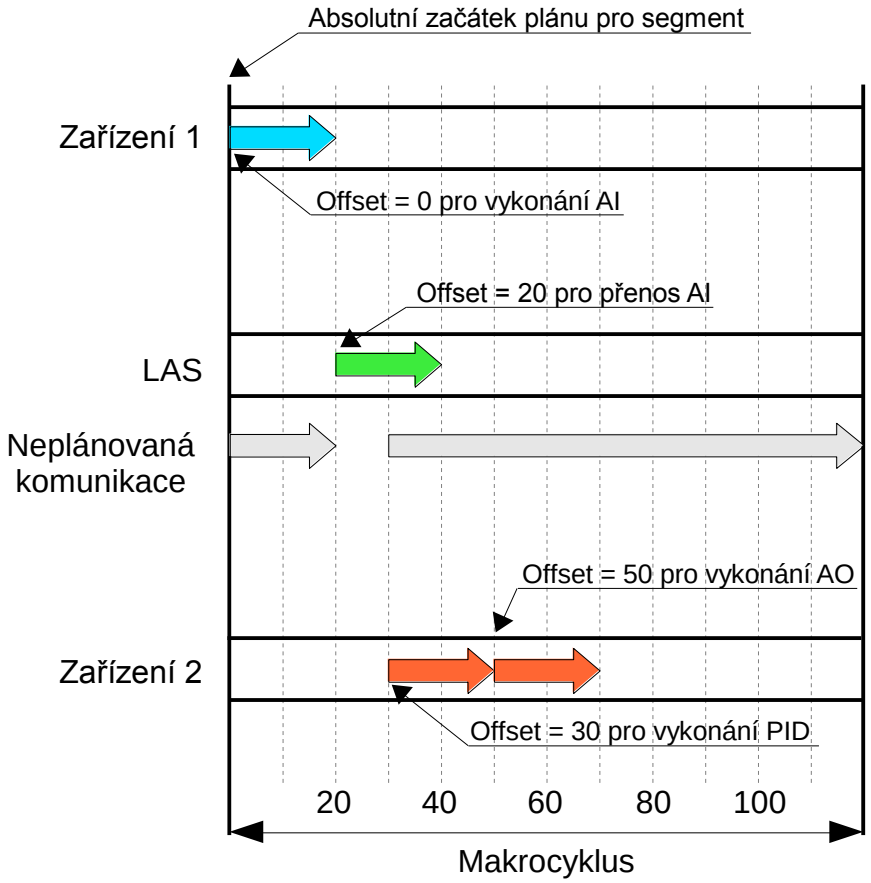
2.1.7. Komunikace na sběrnici

Komunikaci na sběrnici FOUNDATION Fieldbus dělíme na plánovanou *(Schedudled)* a neplánovanou *(Unshedudled)*. Plánované přenosy slouží pro periodicky opakující se úlohy mezi zařízeními na sběrnici. V době, kdy není naplánován žádný přenos, lze použít neplánovaný přenos. Neplánované přenosy jsou využity zejména pro nastavování parametrů a diagnostiku zařízení.

Vykonání funkčních bloků a přenos dat je striktně plánován. Toto načasování provádí operátor při konfiguraci FF systému. O plánování na segmentu se poté stará Link Active Scheduler *(LAS)*, který řídí komunikaci na segmentu. LAS se také stará o synchronizaci času

rozesílání speciálního paketu Time Distribution (TD). Kromě toho také rozesílá Compel Data (CD), Pass Token (PT) a Proble Node (PN). CD slouží jako výzva pro zařízení (publisher), aby neprodleně po doručení odeslalo požadovaná data daným příjemcům (subscriber). Ve volném časovém slotu LAS vyšle jednomu vybranému zařízení v seznamu *(Live List)* PT, který dovoluje zařízení využívat sběrnici pro neplánovanou komunikaci. Zařízení poté může komunikovat jak dlouho potřebuje a dokud nevyprší timeout. Pokud zařízení na PT několikrát za sebou neodpoví, je vyřazeno z Live Listu a slouží pro přidání nového zařízení. Pokud některé zařízení odpoví paketem Probe Response (PR), LAS ho přidá do seznamu. [1]

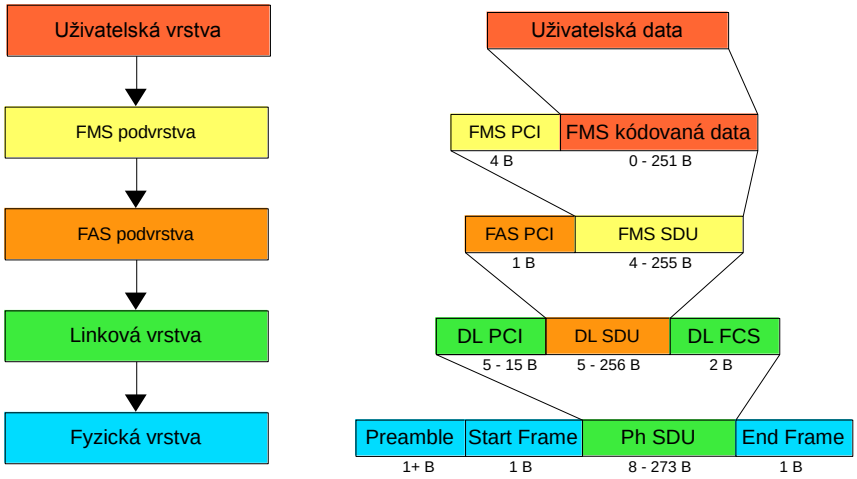
Makrocyklem (macrocycle) je nazván každý plánovaný iterační cyklus. Distribuování dat a vykonávání jednotlivého funkčního bloku je určeno offsetem od začátku makrocyklu. Data musí být s předstihem připravena, aby mohla odpověď odejít ihned po obdržení CD paketu. Na následujícím obrázku (Obrázek 4) lze vidět načasování operací a komunikace sběrnice FF. [2]



Obrázek 4: Příkladné naplánování aplikace FF [1] [2] [7]

Každá z vrstev komunikačního modelu nabaluje na data různé přídavné informace (identifikace dat, typ zařízení, kontrolní součty, délka apod.).

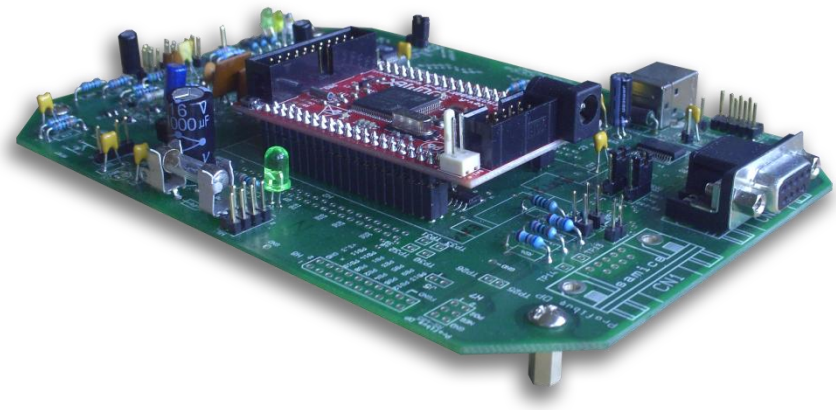
Datová jednotka (Protocol Data Unit - PDU) je označení pro data, která jsou na úrovni stejné vrstvy. V rámci každé vrstvy se k PDU přidává ještě řídící informace vrstvy (Service Data Unit - SDU). U linkové vrstvy se připojuje kontrolní součet celého rámce (Frame Check Sequence - FCS), který slouží pro okamžitou kontrolu přenesených dat. Ve fyzické vrstvě se přidávají synchronizační značky *(Preamble, Start / End frame)*. [7] Viz následující obrázek (Obrázek 5).



Obrázek 5: Nabalování dat při průchodu vrstvami modelu [4]

2.2. Komunikační modul

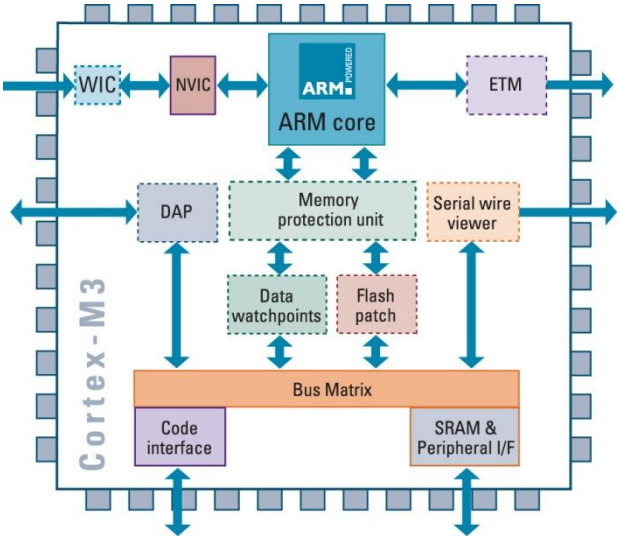
Komunikační modul (Obrázek 6) byl navrhnut a sestaven Pavlem Böhmem v rámci jeho Diplomové práce. Jedná se o desku plošných spojů osazenou konkrétními obvody pro zachytávání komunikace na sběrnici FOUNDATION Fieldbus FF a následné zpracování.



Obrázek 6: Komunikační modul osazen mikrokontrolérem EFM32G880F128

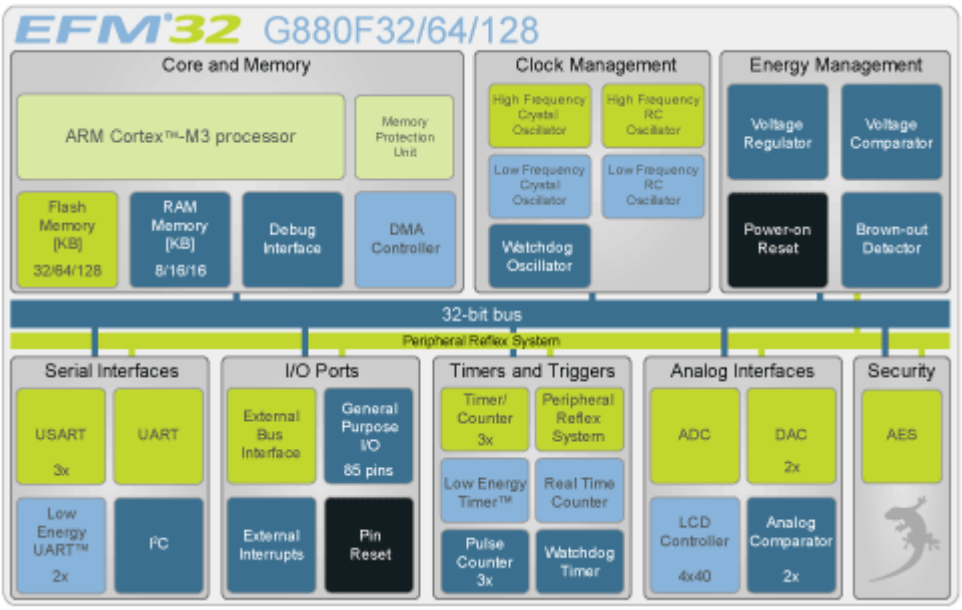
2.2.1. Použité obvody

Mikrokontrolér EFM32G880F128 je centrálním prvkem komunikačního modulu. Jedná se o 32 bitový model s nízkoenergetickou náročností a jádrem založeném na architektuře ARM Cortex-M3. Samotný Cortex-M3 je založen na Harvardské architektuře, která má instrukční sběrnici pro data a pro program oddělenou. Data proto mohou být načítána zároveň, čímž je zvýšen výkon celkového zpracování instrukcí. [1] [7]



Obrázek 7: Blokové schéma Cortex-M3 [7]

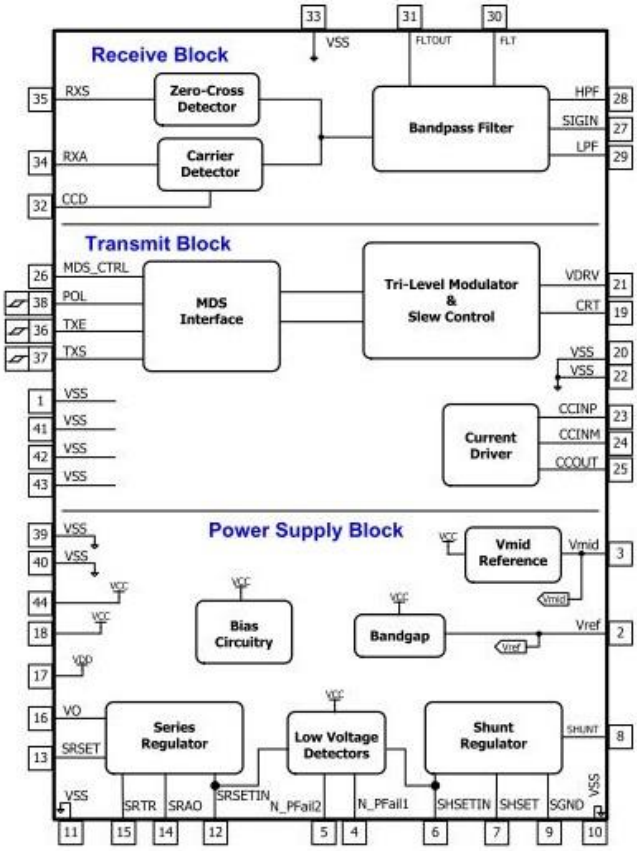
Pro komunikační modul byla použita vývojová deska **EFM32G880F128-H** od společnosti Olimex. Její součástí je MCU, obvody pro napájení, krystaly a vývody pinů MCU pomocí lišt, které se dají vsadit do desky komunikačního modulu. [1] [8]



Obrázek 8: Blokový diagram mikrokontroléru [8]

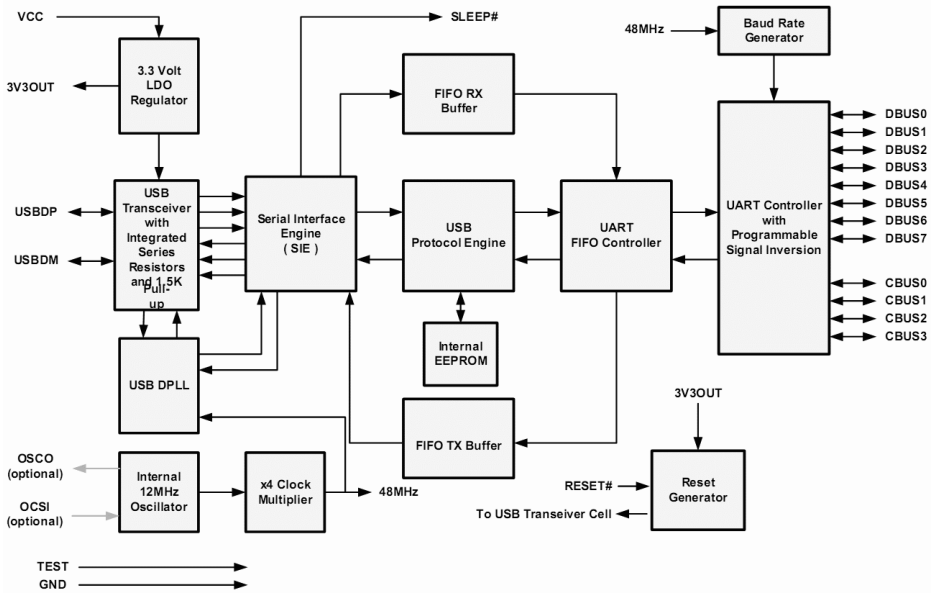
Unified Fieldbus Controller UFC100-F1 implementuje část fyzické a linkové vrstvy FF-H1 a Profibus-PA. Kontrolér se nastavuje a řídí pomocí registrů, které jsou dostupné podle nadefinovaného módu. Dále obsahuje prioritní dekodér přerušení a signalizuje ho do MCU pinem *INTn*. [1]

Media Access AMIS-49200 je jednotka, která implementuje část fyzické vrstvy průmyslových sběrnic Foundation Filedbus-H1 a Profibus-PA. Obvod doplňuje rozhraní přístupu na sběrnici kontroléru UFC100-F1, se kterým je propojen signály RxA (received signal activity), RxS (received signal), TxE (transmission signal enable) a TxS (transmission signal). Obvod umožňuje odebírat napájení ze sběrnice vnitřními regulátory pro analogovou i digitální část. [1] [9]



Obrázek 9: Blokové schéma Media Access AMIS-49200 [9]

FT232RL slouží pro virtualizování sériového portu v počítači přes USB. K dispozici je USB protokol verze 1.1 i 2.0. Řadič UART protokolu je napájen pinem VCCIO. Disponuje vyrovnávací pamětní FIFO a automatickým nastavením rychlosti komunikace. Dále je UART rozšířen o řídící piny protokolů RS232 a RS485. [1]



Obrázek 10: Blokové schéma obvodu FT232R [10]

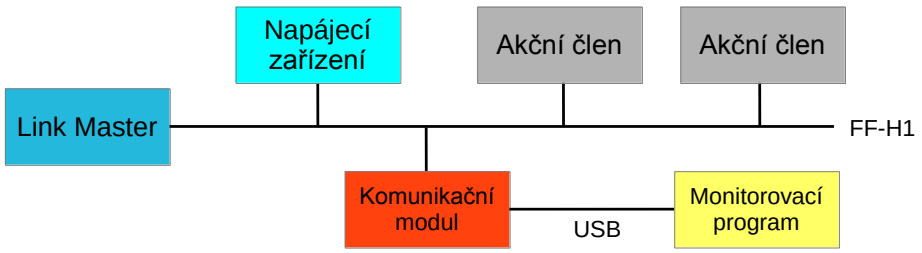
MAX3232 převádí napětí mezi TTL logikou a RS232, který používá záporné a kladné napětí pro vyjádření logických úrovní. [1]

ADM-3078E umožňuje převod z obousměrné komunikace od MCU na poloduplexní podle standardu RS285. Určuje tedy směr pro příjem nebo vysílání na poloduplexní sběrnici RS485. [1]

Zapojení a detailnější informace se dají dohledat v manuálech jednotlivých obvodů a Diplomové práci Pavla Böhma.

3. Praktická část

Úkolem v praktické části je vytvořit program pro komunikační modul, který bude předávat zaznamenanou komunikaci z FF po USB (viz Obrázek 11). Dále vytvořit program pro PC, který bude přeposlaná data z USB zaznamenávat a dále je umět zpracovat formou filtrování.



Obrázek 11: Blokové schéma zapojení během monitorování

3.1. Softwarová realizace komunikační modulu

3.1.1. Požadavky na sw komunikačního modulu

Aby komunikační modul pracoval podle zvolených cílů, musí splňovat následné požadavky:

- Komunikační modul musí umět přijmout datový rámec obdržený ze sběrnice FOUNDATION Fieldbus (FF),
- dále musí přijatý rámec zpracovat a přeposlat po sběrnici USB do PC,
- o svých činnostech jednotlivých sběrnic musí visuálně informovat obsluhu zařízení.

3.1.2. Vývojové prostředí

Pro rychlé a snadné vytvoření programu na řízení komunikačního modulu jsem použil vývojové prostředí IAR Embedded Workbench IDE. Verze používané distribuce je výhradně pro procesory ARM. Jediné omezení distribuce pro bezplatné využití je limitace velikostí kódu na 32 KB.

Důvodem použití právě této distribuce je kompatibilita s programem J-Link. Lze tak přímo ve vývojovém prostředí programovat s pomocí emulátoru JTAG/SWD s UBS rozhraním flash paměť procesoru nebo vkládat breakpointy při vývoji.

Dále v tomto prostředí vyvíjel program pro komunikační modul Pavel Böhm, který jsem použil jako předlohu. Většina konfiguračních funkcí je převzata a nebo částečně upravena pro účely mého programu.

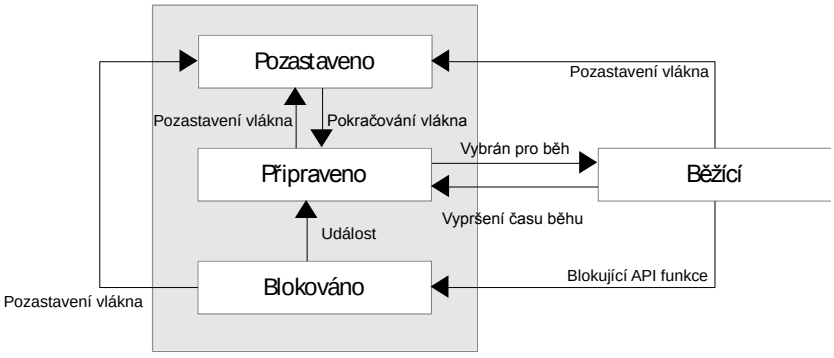
3.1.3. FreeRTOS

FreeRTOS je operační systém reálného času volně dostupný, šiřitelný pod licencí GNU GPL (General Public License). Většina operačního systému je napsána pomocí programovacího jazyka C, pouze několik funkcí je vytvořeno v jazyce assembleru.

Podporuje plně preemptivní i kooperativní zpracování vláken. Pro komunikaci mezi vlákny, popřípadě přerušeními a vlákny, poskytuje nástroje pro synchronizaci a komunikaci.

FreeRTOS obsahuje plánovač s prioritním plánováním, přičemž každé vlákno je ohodnoceno prioritou jeho chodu. Nejnižší číslo je nula, které znamená nejnižší možnou prioritu. Maximální číslo pro nejvyšší prioritu lze nastavit v souboru s nastavením FreeRTOS. Při stejné prioritě nastavených vláken je použita tzv. metoda "kruhu" (Round robin), kdy se vlákna pravidelně střídají v kruhu.

Následující obrázek (Obrázek 12) znázorňuje stavy, do kterých se vlákno může dostat. V jedné chvíli může být obsluhováno pouze jedno vlákno, ostatní vlákna mohou být pozastavena, připravena a nebo blokována.



Obrázek 12: Stavy vláken

Synchronizační nástroje, které FreeRTOS poskytuje, zajišťují správnou posloupnost obslužných instrukcí.

Mezi nejzákladnější patří mutex, který řeší úlohu výlučného přístupu do dané oblasti programu. Mutex může nabývat pouze dvou stavů, odemčený a zamčený. Při vstupu ve vlákně do kritické sekce se mutex zamkne, a dokud ho stejné vlákno opět neodemkne, ostatní vlákna se do kritické sekce nedostanou a budou ve stavu blokovaných vláken.

Semafor je další nástroj pro synchronizaci vláken. Může být použit pro počítání událostí, nebo pro úlohu typu producent – konzument. Jeho hodnota může nabývat kladných čísel včetně nuly. Hodnota se mění pomocí funkcí *xSemaphoreGive()* a *xSemaphoreTake()*. Při zavolání funkce *xSemaphoreTake()* na semafor, který má hodnotu větší než 0, nebude vykonávání programu zablokováno. Pokud se ale zavolá funkce na semafor, který má nulovou hodnotu, vykonávání programu bude blokováno do doby, dokud jiné vlákno nenavýší semafor do kladných hodnot pomocí funkce *xSemaphoreGive()*.

Když potřebujeme synchronizovat nejenom dvě nebo více vláken, ale také si mezi nimi předávat data, použijeme frontu zpráv. Jedná se o paměť typu FIFO (First In First Out), která zároveň při pokusu o čtení prázdné paměti zařizuje blokaci vlákna. To zajišťuje, že vlákno, které se o data přihlásilo, se zablokuje do doby, než data budou k dispozici. Funkce pro používání fronty jsou *xQueueSend()* a *xQueueReceive()*. [12]

FreeRTOS lze jednoduše nakonfigurovat pomocí definic v souboru *FreeRTOSConfig.h*, nejdůležitější nastavení v implementaci jsou [1]:

- **#define configCPU_CLOCK_HZ 32000000UL**
  slouží pro informaci operačního systému o konfiguraci rychlosti taktu
- **#define configTICK_RATE_HZ ((portTickType) 1000)**
  spolu s *configCPU_CLOCK_HZ* určují periodu pro aktivování plánovače
- **#define configUSE_PREEMPTION 1**
  nastavuje, že se bude využívat preemtivní plánovač

3.1.4. Inicializace mikrokontroléru a jednotlivých obvodů

V prvních krocích běhu se provádí funkce *CHIP_Init()*. Ta je definována výrobcem a je obsažena v balíčku knihoven. Funkce zajišťuje odstranění programových odlišností spojených s výrobou procesoru.

Pro sériovou asynchronní linku je použit UART0 v lokalizaci 0, tzn. na pinech 6 a 7 portu F. Každá periferní jednotka vyžaduje přivedení hodinového signálu pro jejich aktivaci. Pro nastavení UART slouží funkce *CMU_ClockEnable()*, která má dva parametry. První parametr je jednotka, která se má nastavit a druhý parametr je zda-li má být jednotka aktivována nebo deaktivována.

Funkcí *GPIO_PinModeSet()* aktivujeme vstupní pin Rx a výstupní pin Tx. První parametr funkce je označení portu, dále je číslo pinu, mód a inicializační logická hodnota pinu.

Pro nastavení správné rychlosti komunikace slouží registr CLKDIV. Vzoreček pro výpočet z dokumentace mikrokontrolélu je

256 * ((UART_PERCLK_FREQUENCY/(16 * UART_BAUDRATE)) - 1),

kde UART_PERCLK_FREQUENCY je rychlost hodin periferních jednotek (v našem případě rychlost CPU) a UART_BAUDRATE je požadovaná rychlost přesnou. Posledním nastavením sériové linky je povolení příjmu a vysílání v řídícím registru CMD.

Konfigurace řadiče UFC100 probíhá pomocí registrů. Ten jich obsahuje několik desítek a jsou vždy 8-bitové. Dělí se na řídící, informační a datové. Pro samotné nastavení slouží řídící, informativní sdělují stavy některé z jednotek radiče a jsou nastavovány samotnou jednotkou.

Následuje nastavení dalšího rozhraní s obvodem MAU. V registru *MauCntl* je plně duplexní režim, který umožní zpětně přijímat vlastní odeslané rámce. To je důležité zejména pro monitorování komunikace. V dalším registru *PhlPara* lze nastavit délku hlavičky (preamble) při vysílání a mezeru mezi jednotlivými rámci. Samotné nastavení se odvíjí od konfigurace sběrnice, ke které je zařízení připojeno. [1]

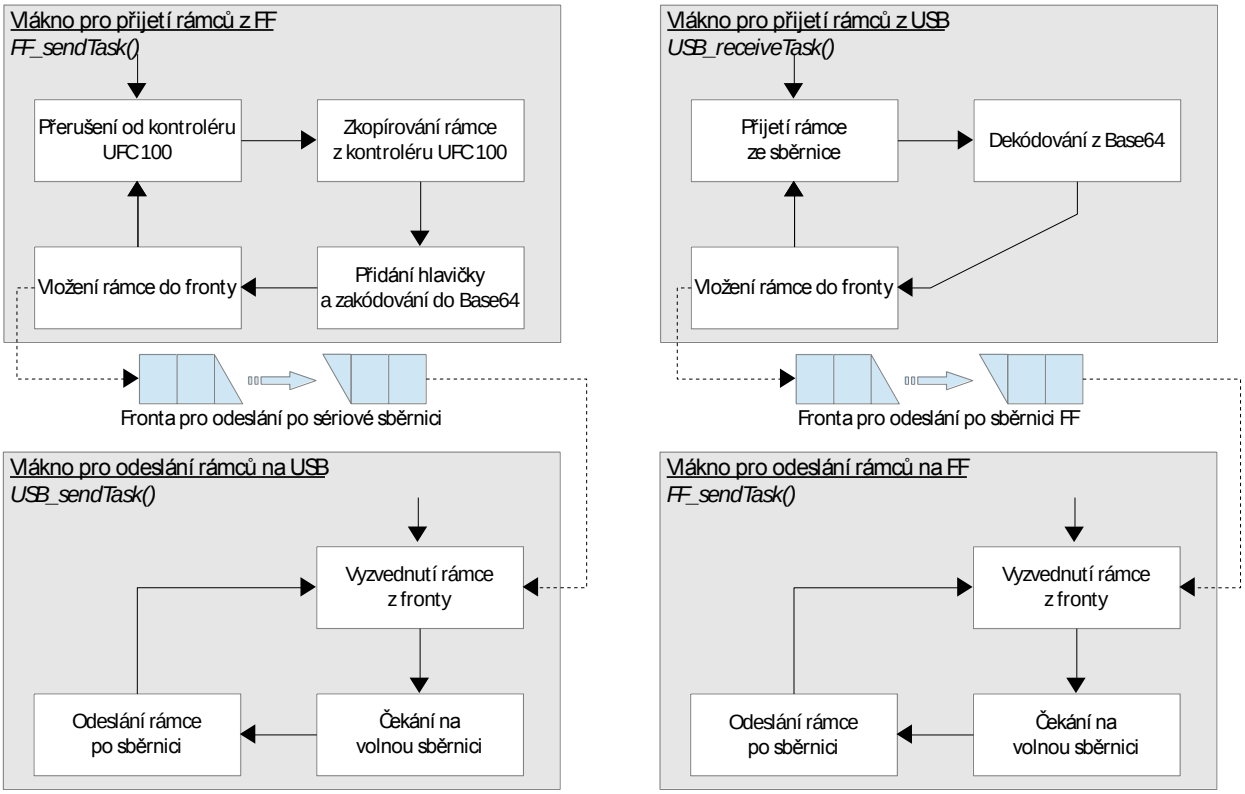
Detailnější informace, ohledně konfigurace celého komunikačního modulu, poskytne Diplomová práce Pavla Böhma, ze které program vychází.

3.1.5. Funkce a rozvržení vláknových úloh

Řízení přeposílání dat mezi FF a USB zajišťují čtyři hlavní vlákna. Dvě zajišťují příjem a odchod dat po sběrnici FF a další dvě zase příjem a odchod dat po USB. Komunikace vláken je řízena pomocí dvou front, které implementují návrhový vzor producent-konzument (viz Obrázek 13).

Funkce jednotlivých vláken jsou:

- **FF_sendTask()** - pro odeslání dat po sběrnici FF; obsahuje nekonečnou smyčku, ve které se čeká na data vložená do fronty pro odeslání (princip producent - konzument) a následně je použita funkce *UFC_sendFrame()*, která provede samotné odeslání na sběrnici
- **FF_receiveTask()** - pro příjem dat po sběrnici FF; před nekončenou smyčkou, která čeká na příchozí přerušení a příjem dat, je inicializační procedura pro přípravu na první příchozí data, v samotné smyčce je vybrán první volný slot ve frontě, do kterého se zapíšou přijaté data
- **USB_sendTask()** - pro odeslání dat po sběrnici USB; obsahuje opět nekončenou smyčku, ve které se čeká na data z fronty obdržené ze sběrnice FF, data se posléze zakódují pomocí kódování Base64 a jsou odeslána po sériové sběrnici USB do PC
- **USB_receiveTask()** - pro příjem dat po sběrnici USB; v nekonečné smyčce se čeká na příchozí data, ta jsou dále dekódována a vyhodnocena; pokud se jedná o informativní paket, data jsou uložena do fronty pro odeslání po USB, jedná-li se ale o paket datový, data jsou uložena do fronty pro odeslání po FF



Obrázek 13: Rozvržení úloh jednotlivých vláken a komunikace mezi nimi

3.1.6. Využití Base64

Jak bylo zmíněno v předchozí kapitole, na USB sběrnici je použito kódování dat algoritmem zvaným Base64. Důvodem jeho využití bylo, že dokáže libovolná data reprezentovat pouze jako ASCII znaky. Nevýhodou kódování je nárůst objemu dat přibližně od 30 %.

Jelikož na sběrnici USB používám vyšší přenosovou rychlost, než je na FOUNDATION Fieldbus (FF), nárůst nenarušuje propustnost a navíc usnadňuje dělení jednotlivých přeposlaných rámců FF.

Princip kódování dat do Base64 lze snadno vysledovat v následující tabulce (Tabulka 4). Na data se nahlíží jako na proud bitů, které se po šesti bitech prezentují jako ASCII znak.

Tabulka 4: Příkladné zakódování textu kódováním Base64 [13]

Výchozí text	**M**								**A**								**n**							
ASCII	77								97								110							
Bity znaků	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
Indexy	19				22				5				46											
Zakódování Base64	**T**				**W**				**F**				**u**											

Funkce pro kódování *base64_encode()* a dekódování *base64_decode()* dat jsou definovany v hlavičkovém souboru *base64.h*. Samotné funkce pro převod jsou v souboru *base64.c*. [14]

3.1.7. Signalizace komunikace

Abychom mohli nějakým způsobem pozorovat aktivitu na jednotlivých sběrnicích, využil jsem LED diody, které jsou na desce mikrokontroléru EM-32G880F128-H. Signalizace obou diod neovlivňuje běh programu jednotlivých vláken, protože je pro každou diodu vytvořena smyčka ve vlákně s nízkou prioritou.

Jedna z LED diod je využita pro odchozí a příchozí data na straně sběrnice FF. Druhá dioda zase pro odchozí a příchozí data na straně USB.

Délka jednotlivých bliknutí je 50 ms. Pokud je zavolána funkce pro bliknutí během již probíhající akce, je tento požadavek ignorován.

3.2. Softwarová realizace monitorovacího programu

3.2.1. Požadavky na sw monitorovacího programu

Aby monitorovací program pracoval podle zvolených cílů a splňoval zadání, musí splňovat následné požadavky:

- Musí umět navázat spojení s komunikačním modulem,
- zaznamenávat přijaté pakety,
- filtrovat rámce dle základních parametrů linkové vrstvy FOUNDATION Fieldbus,
- visuálně prezentovat zaznamenané rámce,

- umožnit obsluze programu export vyfiltrovaných rámců do souboru.

3.2.2. Vývojové prostředí

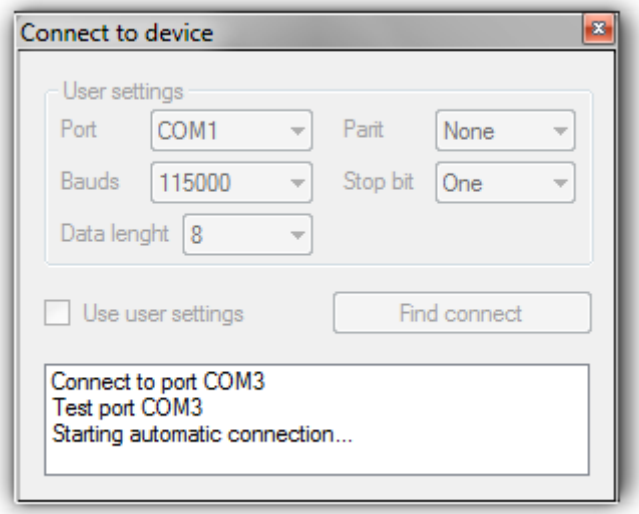
K vytvoření monitorovacího programu jsem využil Microsoft Visual Studio 2010. Jako programovací jazyk jsem zvolil C#, protože je snadno použitelný ve formulářových aplikacích ve Windows a díky platformě .NET Framework lze jednoduše přistupovat k dílčím prvkům jako je například port USB. [14]

3.2.3. Spojení s komunikační modulem

Spojení mezi komunikačním modulem a PC je pomocí USB. Jedná se o virtualizaci sériové komunikace.

Pro připojení z PC ke komunikačnímu modulu jsem vytvořil vlastní třídu *ConnectUSB*. Třída obsahuje metody pro testování validního připojení s modulem a pro spojení využívá komponentu *SerialPort*, která zajišťuje obsluhu sériové komunikace.

Před samotným připojením ke komunikačnímu modulu se musí nastavit korektní parametry spojení. Nastavení lze nakonfigurovat ve formulářovém okně *FormConnect.cs*, které obsahuje dvě možnosti připojení (viz Obrázek 14).



Obrázek 14: Průběh připojování ke komunikačnímu modulu

První možností je nechat program automaticky vyhledat komunikační modul. Druhá možnost je ručně zvolit parametry spojení, které se aktivují zatržením *CheckBoxu* s popiskem *Use user settings*.

Po stisknutí tlačítka *Find connect* začne probíhat připojovací procedura, která se skládá z několika kroků. Prvním krokem je deaktivace tlačítka kvůli zamezení vícenásobného provedení a aktivace části formulářového okna, ve kterém se zobrazí komponenta *ListBox*. Do *ListBox* zapisují jednotlivé události během připojování. Dalším krokem je aktivace vlákna pro samotný cyklus připojování.

Nejprve se rozhodne, zda-li se jedná o automatické připojování a nebo manuální. Při manuálním připojování se nastaví komponenta *SerialPort* přesně podle vyplněného formuláře, naopak při automatickém připojování se použije základní nastavení, definované pro komunikační modul a výběr komunikačního portu (COM) se postupně mění, dokud není nalezeno správné spojení.

Samotný postup testování spojení je u automatického i manuálního nastavení totožný. Nejprve se provede otevření portu příkazem *serialPort.Open(),* následně se zavolá metoda *isValidConnect()* instance třídy *ConnectUSB*, která provede detekci zařízení. Detekce probíhá formou odeslání informativního paketu a následné čekání na odpověď. V případě, že komunikační modul neodpoví regulérním paketem do 500 ms, je považován za nevyhovující.

Pakety, které se přenášení po sériové sběrnici (USB), lze rozdělit z hlediska monitorovacího programu na dva typy. Prvním typem je paket informativní, na který má komunikační modul povinnost odpovědět. Druhým typem je datový paket, který obsahuje strukturu rámce FF, a může být odeslán v obou směrech. V monitorovacím programu se ale využívá pouze směr *komunikační modul → monitorovací program*. Pro určení typu paketu je použit první byte. [14]

3.2.4. Zachytávání komunikace

K zachytávání komunikace se používá, jak jsem již napsal v předchozí kapitole, komponenta *SerialPort*, která podporuje synchronní i asynchronní čtení a zápis. Při tvorbě spojení a detekcí komunikačního modulu se využívá synchronní spojení, protože vyžadujeme přesnou posloupnost. Během monitorování sběrnice se naopak používá asynchronní čtení.

Při přijetí dat ze sériové sběrnice vyvolá komponenta *SerialPort* událost, při které se zavolá metoda *serialPort1_DataReceived()*. V metodě se otestuje, jestli je platné připojení a zda-li paket není určen pro synchronní čtení. Následně jsou data dekódována z Base64 a vložena do

nové instance třídy *PacketUSB*. Tato třída reprezentuje data jako rámec FOUNDATION Fieldbus (FF) a obsahuje informace o času přijetí, délce, typu rámce apod.

Třída *PacketUSB* obsahuje tyto proměnné a metody:

- *private static int id* - privátní proměnná je využita při automatickém číslování jednotlivých paketů
- *public string PID* - veřejná proměnná obsahuje id paketu ve formě řetězce
- *public string PType* - veřejná proměnná obsahuje typ FF rámce ve formě řetězce
- *public string PLenght* - veřejná proměnná obsahuje délku FF rámce ve formě řetězce
- *public string PTime* - veřejná proměnná obsahuje čas přijetí paketu ve formě řetězce
- *public string PLink_to, PNode_to, PSelector_to* - veřejné proměnné obsahující adresu příjemce
- *public string PLink_from, PNode_from, PSelector_from* - veřejné proměnné obsahující adresu odesílatele
- *private string text* - privátní proměnná obsahující shrnující informace o paketu, které se používají při ladění programu
- *public byte type* - veřejná proměnná pro číselnou reprezentaci typu paketu
- *public byte[] data* - veřejná proměnná obsahující data FF rámce
- *private DateTime time* - privátní proměnná obsahující časovou značku přijetí
- *public static string[] name* - obsahuje pole řetězců, ve kterých jsou jednotlivé typy FF rámců
- *PacketUSB(byte[] dataSet)* - první kontruktor třídy, který vytvoří strukturu paketu ze zvolených (obvykle příchozích) dat
- *PacketUSB(byte typeSet, byte[] dataSet)* - druhý kontruktor třídy, který vytvoří strukturu paketu ze zvolených (obvykle vytvořených) dat a nastaví jeho typ

- *public static string Name(int code)* - statická metoda pro převedení typu FF rámce na textovou reprezentaci
- *private void setAddress()* - metoda pro nastavení prezenčních adres FF rámce
- *public static int getID(string s)* - metoda pro převod textového typu FF rámce na číselný
- *public static string byte2HexString(byte[] data, int from, int count)* - statická metoda pro převedení pole bytů na řetězec s možností definování délky převáděných dat a startovní pozicí
- *private static string byte2HexString(byte[] data)* - statická metoda pro převedení pole bytů na řetězec
- *public string byte2HexString()* - metoda pro vytvoření řetězce reprezentující byte data jako hexadecimální znaky
- *public String genPacket()* - složení paketu pro odeslání po sériové sběrnici

Proměnné začínající znakem P jsou využity v zobrazování v komponentě *DataGridView*, která prezentuje přijaté FF rámce formou přehledné tabulky. [14]

Instance třídy *PacketUSB* se dále ukládají do dvou bindovacích seznamů *SortableBindingList*, které podporují zobrazování a řazení v komponentě *DataGridView*. První seznam, s názvem proměnné *packets*, je určen výhradně pro záznam jednotlivých dat, druhý seznam, s názvem proměnné *packetsGridView*, je určen pro vstupní data do zobrazovací komponenty. Mezi seznamy je ještě použita filtrovací procedura, kterou detailněji popisuji v kapitole o filtrování. [16]

3.2.5. Zobrazení

Zobrazování dat je rozděleno na tři části:

- První částí je zobrazení samotné tabulky zaznamenaných a vyfiltrovaných dat pomocí komponenty *DataGridView*,
- druhou částí je detailnější zobrazení vybraného paketu pomocí komponenty *TreeView*,

- třetí částí je hexadecimální prezentace vybraného paketu pomocí komponenty *RichTextBox*.

Aby bylo možné propojit jednotlivá data ze třídy *PacketUSB* se sloupci *DataGridView*, musel jsem nadefinovat ve správě sloupců *Edit Columns* jednotlivé názvy sloupců, jejich vlastnosti a hlavně provázanost s proměnnými ve třídě *PacketUSB* pomocí *DataPropertyName*.

Jelikož není z hlediska výkonu vhodné aktualizovat data vždy po každé změně (přibližně 20x za vteřinu), použil jsem vlákno třídy *Thread* na metodu *updateDataGrid()*, která aktualizuje data v tabulce vždy po půl vteřině. Metoda běží v nekonečné smyčce a jelikož se nemůže přímo přistupovat ke komponentám z jiného vlákna, musel jsem použít pro přístup k *DataGridView* tzv. delegáty. A tak vlákno na aktualizaci tabulky čeká na výzvu od vlákna, které spravuje *DataGridView* a poté provede aktualizační proceduru.

Aby byly jednotlivé řádky v tabulce přehledné, použil jsem stylování řádků. Výhodou je, že samotné stylování řádků je již součástí *DataGridView*, takže pouze stačilo nastavit na komponentě hodnoty *RowsDefaultCellStyle* pro liché řádky a *AlternatingRowsDefaultCellStyle* pro řádky sudé.

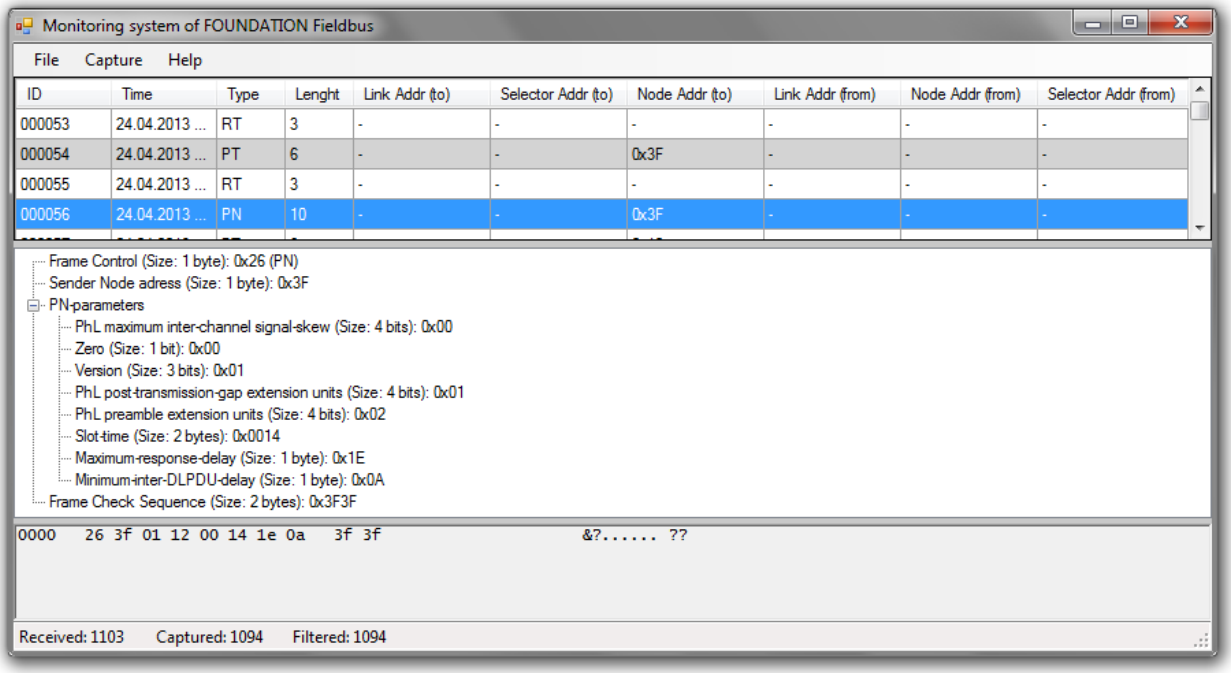
Další uživatelskou vlastností zobrazovaných dat je, že je lze řadit pouhým kliknutím na název sloupce *DataGridView*. Tuto vlastnost poskytuje třída *SortableBindingList*, která se dědí z klasické třídy *BindingList* a rozvíjí vlastnosti právě o řazení např. v komponentě *DataGridView*.

Při výběru (kliknutím myší na řádek v tabulce) daného rámce se zavolají dvě metody, které analyzují rámec a zobrazí výsledky v komponentách *TreeView* a *RichTextBox*.

První metoda je *showHexPacket()*, která načítá jednotlivá data jako hexa znaky a prezentuje je do třech sloupců. V levém sloupci jsou adresy jednotlivých znaků rámce, v prostředním sloupci jsou zobrazena hexa data ve dvou sloupcích po osmi bytech. V pravém sloupci jsou data zobrazována jako ascii znaky. Znaky, které nelze zobrazit tisknutelným znakem, jsou zastoupeny tečkou. Všechen text je zobrazován komponentou *RichTextBox*, která podporuje formátování textu. Hlavním důvodem použití byla podpora neproporcionálního písma, která udržuje formát sloupců. Další výhodou komponenty *RichTextBox* je přizpůsobení vzhledu vůči celému programu.

Druhou metodou je *showTreePacket()*, která podle typu FOUNDATION Fieldbus (FF) rámce provede rozdělení dat do jednotlivých skupin. Jelikož se ale na testovací sběrnici FF-H1 nepodařilo zachytit všechny typy možných rámců a není tato prezentace rámce součástí zadání práce, není metoda kompletně otestovaná. Jedná se spíše o demonstraci některého ze směrů, kterým lze monitorovací program dále rozvíjet. Pro struktury jednotlivých typů rámců jsem použil informace z normy ČSN EN 61158-4-1.

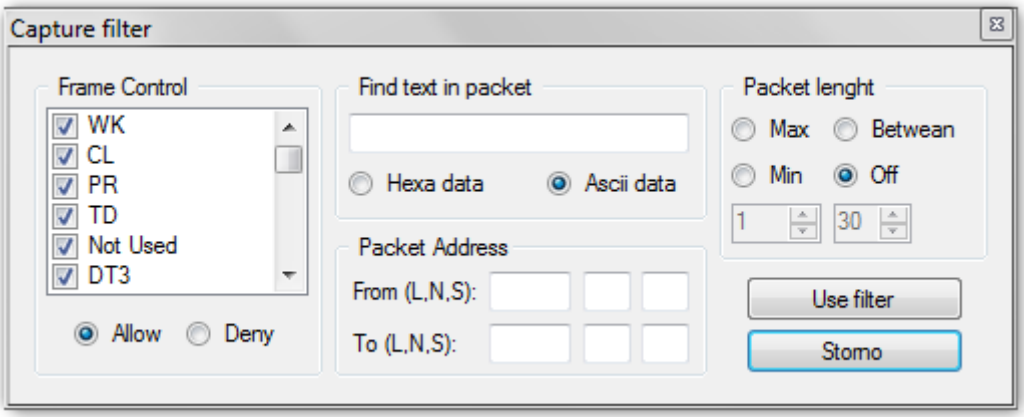
Pro názorný příklad, zobrazování rámců v tabulce a pro zobrazení detailů jednotlivých informací o rámci, uvádím následující obrázek (Obrázek 15).



Obrázek 15: Zobrazení zachycených rámců a podrobné informace o vybraném rámci

3.2.6. Filtrace

Aby bylo možné zachycená data třídit, uspořádat apod., musel jsem vytvořit filtrační proceduru. Jak jsem již zmínil v kapitole o zachytávání komunikace, v programu jsem použil dva seznamy pro ukládání dat. V prvním jsou uloženy všechny zachycené rámce přeposlané z komunikačního modulu, do druhého se filtrují rámce skrz metodu *capturePacket_Filter()*. Metoda využívá další třídu *PacketFilter*, ve které jsou uloženy jednotlivé parametry filtrování. Tyto parametry si obsluha programu může nastavit v dialogovém okně, které lze vyvolat skrze menu *Capture* → *Filter*. V následujícím obrázku (Obrázek 16) lze vidět možnosti pro filtrování rámců.



Obrázek 16: Dialogové okno pro nastavení parametrů filtru

Ve filtrování lze navolit výčet dovolený *(Allow)* nebo naopak zakázaných *(Deny)* typů rámců. Dále lze vyfiltrovat rámce, které obsahují zadaný ascii nebo hexa řetězec. Také lze omezit výběr rámce podle jeho délky, kde jsou k dispozici tři možnosti omezení. Prvním omezení je *Max*, které omezuje horní hranici délky, druhým je *Min*, které naopak omezuje spodní hranici. Poslední možností je *Betwean*, které umožňuje zadat obsluze rozsah délky rámce. Poslední filtr se vztahuje na adresy v rámci. Jak víme z teoretické části, rámec může obsahovat až tři typy adresace pro adresáta i odesílatele. Pro každý typ (Link, Node, Selektor) jsem vyčlenil kolonky zvlášť.

Po uložení nastavení filtrace (stisknutím tlačítka *Use filter*) se provede výmaz všech rámců obsažených v seznamu *packetsGridView* pro zobrazení v *DataGridView*. Následuje přehodnocení / přefiltrování všech doposud zaznamenaných rámců ze seznamu *packets*, skrze metodu *capturePacket_Filter()*, která již má k dispozici nové nastavení. Všechny následně zachycené rámce se filtrují jednotlivě.

Metodat *capturePacket_Filter()* při testování postupuje následovně:

- Jako první metoda zjišťuje z nastavení, zda-li seznam rámců je pro povolené nebo zakázané typy a prochází jednotlivé údaje ze seznamu *PacketFilter.FC_AllowItem*, podmínku lze zapsat:

```
if (PacketFilter.FC_AllowItem[PacketUSB.getID(pck.PType)]
    ^ !PacketFilter.FC_Allow) { ... }
```

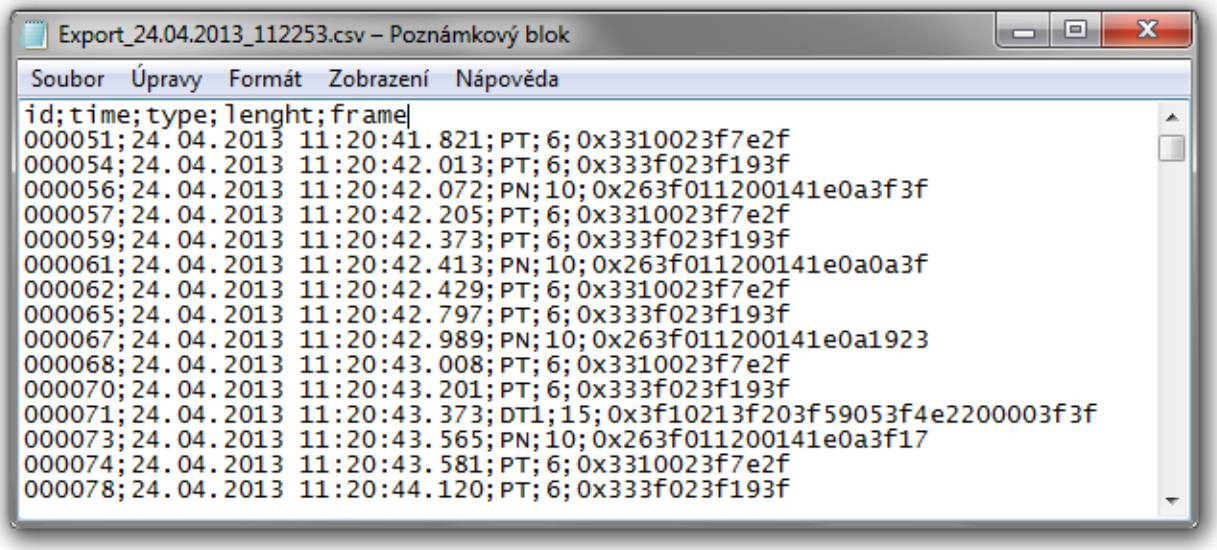
- následuje podmínka ohledně délky rámce, podle zvoleného typu omezení se otestuje rámce a vyhodnotí zda-li požadavky splňuje,

- dále jsou testovány adresy (jsou-li vyplněné obsluhou),
- nakonec se vyhledává v obsahu rámce zvolený řetězec (opět je-li vyplněn obsluhou) a porovnává se buď jako ascii řetězec, a nebo přímo jako hexa data.

Pokud rámec splňuje všechna filtrační omezení, je přidán do seznamu *packetsGridView* a následně vypsán v tabulce *DataGridView*.

3.2.7. Export

Zachycené rámce lze dále exportovat do CSV souboru (viz Obrázek 17). Exportování dat se provádí v menu *File → Export*, kdy je obsluze zobrazena dialogová nabídka pro umístění a název exportovaného souboru. Po potvrzení nabídky se provede metoda *exportCSV()*, která uloží všechna aktuálně obsažená data ze seznamu *packetsGridView* ve formátu CSV.



Obrázek 17: Obsah exportovaného CSV souboru

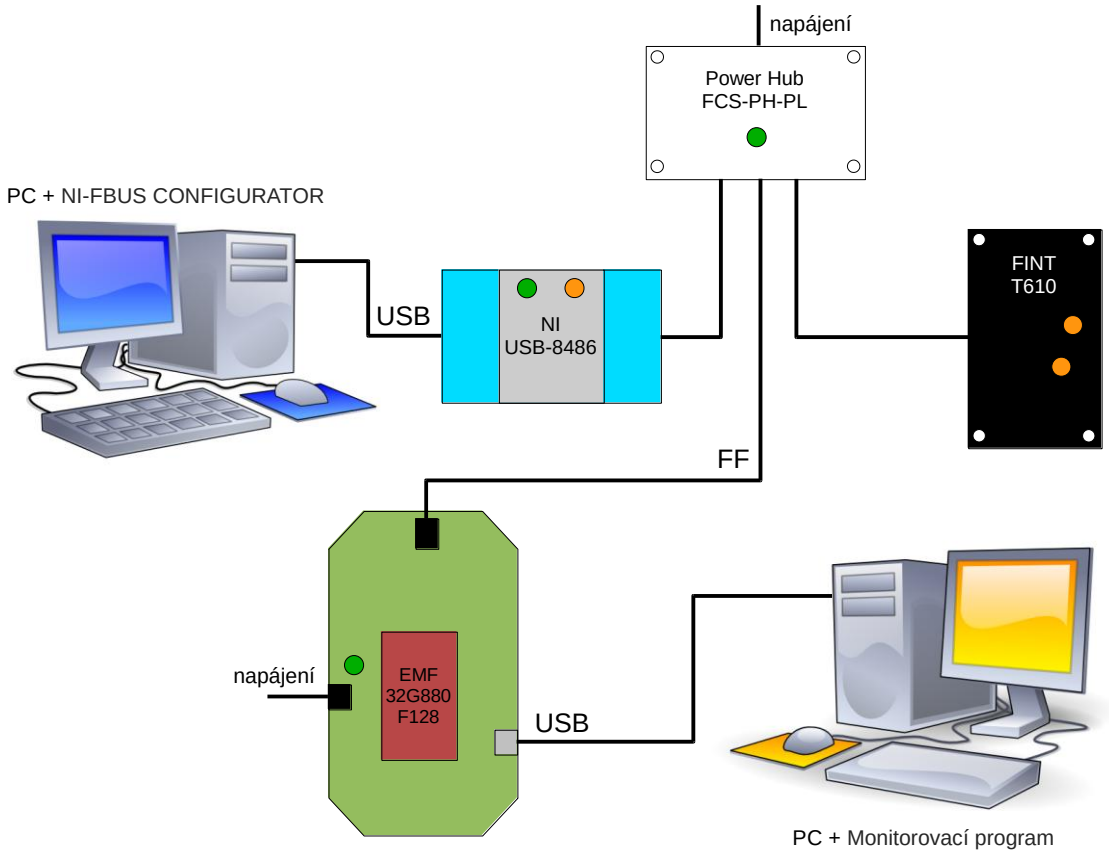
Důvodem použití formátu CSV byla jeho jednoduchost a následná použitelnost k dalšímu zpracování. Do programu lze jednoduše přidat metody pro export do dalších formátů, jako je například XLS nebo PDF. Pro tyto typy je ale nutné použití dalších knihoven. [15]

3.3. Otestování programového vybavení

Testování probíhalo se skutečnými zařízeními sběrnice FF-H1, které byly k dispozici:

- **National Instrument (NI) USB-8486** - konfigurátor segmentu sběrnice FF-H1 s využitím rozhraní USB v programu NI-FBUS Communication Manager
- **Fieldbus International (FINT) T610 Modbus to Foundation** - převodním FF-H1 na modbus se čtyřmi AI bloky
- **Realcon Inc. Power Hub FCS-PH-PL** - čtyř konektorová výhybka s napájením sběrnice

Zapojení jednotlivých zařízení ilustruje následující obrázek (Obrázek 18), jedná se pouze o schéma zapojení, reálné zapojení lze vidět v příloze.



Obrázek 18: Zapojení pro testování komunikačního modulu a monitorovacího programu

Komunikační modul je zapojen mezi USB-8486 a FINT T610 [17]. To umožňuje zachytávat veškerou komunikaci na sběrnici. Zařízení USB-8486 má, v základním nastavení, na

sběrnici funkci LAS, takže jedním z jeho úkolů je rozesílání tokenů jednotlivým zařízením připojených na segmentu. Jeho ovládání řídí program NI-FBUS [18] [19]. Více informací o NI-FBUS lze dohledat v příloze a stránkách výrobce.

Během vývoje programu pro komunikační modul jsem narazil na problém při zachytávání paketů. Při připojení osciloskopu na sběrnici bylo patrné, že signál se po připojení modulu do segmentu úplně zaruší. Po dlouhém sledování rušícího elementu jsem došel přes obvody komunikačního modulu až k notebooku, který sloužil jako zařízení pro monitorovací program. Při odpojení notebooku od napájecí sítě rušení přestalo. Původně jsem si myslel, že může být problém ve zdroji, ale při otestování zdroje na elementy rušení se ukázalo, že i zdroj je v pořádku. Další hledání problému vysvětlila až Diplomová práce Pavla Böhma, která poukazuje na fakt, že musí být zemnící vodič propojen se záporným pólem sběrnice. Rušení bylo tedy pravděpodobně způsobováno rozdílnými potenciály mezi zařízeními.

Abych otestoval stabilitu a spolehlivost komunikačního modulu s naprogramovaným řídicím programem, nechal jsem ho spuštěn pět hodin v kuse. Při návratu komunikační modul stále signalizoval LED diodami aktivitu a při připojení monitorovacím programem byly rámce stále přeposílány.

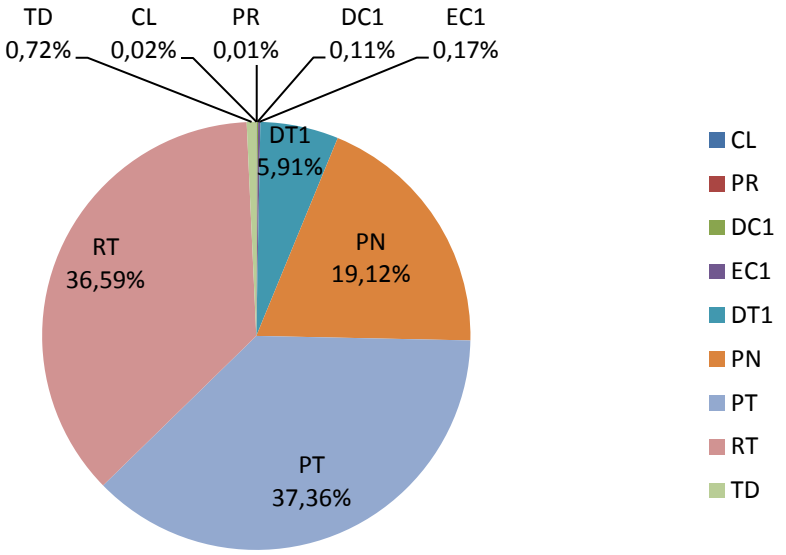
Poslední testování bylo provedeno pro samotný monitorovací program. Test byl zaměřen na spolehlivost, časovou a paměťovou náročnost a rychlost odezvy při filtrování přijatých dat.

Při spuštění samotného monitorovacího programu se alokuje přibližně 6 MB operační paměti (při spuštění z Visual Studia je hodnota vyšší přibližně o 1 MB, kvůli debugovacím značkám při ladění). Důvodem tak vysoké hodnoty je použití formulářových oken.

Pro zjištění paměťové náročnosti, během zachytávání rámců, jsem nechal spuštěný program přibližně dvacet minut. Během této doby se zachytávala komunikace mezi USB-8486 a FINT T610, která byla přeposílána po USB z komunikačního modulu. Po uplynulé době bylo zachyceno přes 10 000 rámců. Paměť programu se zvýšila na 21 MB. Procesor byl během zachytávání dat využit v průměru na 5 %. Při nastavení parametrů filtru před samotným zachytáváním dat se následně neprojeví na úbytku výkonu. Pokud se parametry filtru změní již se zaznamenanými daty, a nebo během samotného zachytávání, program trvá pouze zlomek vteřiny přefiltrování rámců. Během této operace je procesor vytížen maximálně na 25 %. Lze

tedy konstatovat, že program je v rámci možností jazyka C# velmi svižný a nenáročný na výkon PC.

Na testovacím segmentu sběrnice byly zaznamenány při testech jen některé typy rámců. Procentuální zastoupení lze vidět na následujícím grafu (Obrázek 19).



Obrázek 19: Zastoupení jednotlivých typů rámců odeslaných na sběrnici

Číselné podklady ke grafu jsou k dispozici v následující tabulce (Tabulka 5). Jedná se o údaje zaznamenané od spuštění USB-8486.

Tabulka 5: Zaznamenané počty jednotlivých typů rámců

Typ rámce	Počet
CL	2
PR	1
DC1	11
EC1	17
DT1	601
PN	1944
PT	3799
RT	3720
TD	73
Celkem	10168

4. Závěr

V rámci práce jsem shrnul informace o způsobu komunikace a problematice monitorování na sběrnici FOUNDATION Fieldbus. Cílem práce bylo navrhnutí programu, který by byl schopen monitorovat a zaznamenávat komunikaci na PC probíhající po sběrnici za využití komunikačního modulu.

Realizace se skládala ze dvou částí. V první části jsem vytvořil programové vybavení pro samotný komunikační modul, abych ho přizpůsobil pro účely monitorování a přeposílání zaznamenaných dat do PC ze sběrnice FOUNDATION Fieldbus. Program jsem vyvíjel v prostředí IAR Embedded Workbench IDE.

Dále jsem vytvořil komunikační pravidla na USB sběrnici pro automatické vyhledání spojení s komunikačním modulem. Ten musel obsahovat tato pravidla pro správnou komunikaci. V druhé části jsem se soustředil na vytvoření samotného programu pro zachytávání a další zpracování dat přeposlaných komunikačním modulem. Pro vývoj programu jsem použil programovací jazyk C#, který se mi zdál pro tyto účely dostačující. Celý program byl tvořen ve vývojovém prostřední Microsoft Visual Studio 2010.

Samotné testování probíhalo na experimentálním segmentu sběrnice FOUNDATION Fieldbus. V rámci testování komunikačního modulu, respektive jeho programového vybavení, byl kladen důraz na stabilitu a spolehlivost řídícího programu. U testování monitorovacího programu na PC byl kladen důraz na paměťovou a procesorovou náročnost. Mimo jiné se testovala i schopnost reakce programu při různých složitějších operacích.

Počítač, na kterém byl testován monitorovací program, byl notebook ASUS N50Vn, který běžel pod operačním systémem Microsoft Windows 7 x64. Obsahuje procesor Intel(R) Core(TM)2 Duo CPU P8700 2.53 GHz a disponuje 4 GB operační paměti DDR2. Vzhledem ale k jeho minimálnímu využití lze předpokládat, že monitorovací program lze bez problému využít i na slabších sestavách.

Přehled zkratek

ARM	Advanced RISC Machine
CPU	Central Processing Unit
CSV	Comma Separated Values
DD	Device Description
DDL	Device Description Language
DL	Data Link
DLCEP	Data Link Connection End Point
FAS	Fieldbus Access Sublayer
FCS	Frame Check Sequesnce
FF	Foundation Fieldbus
FIFO	First In First Out
FMS	Fieldbus Message Speci cation
GNU	GNU is Not UNIX
GPL	General Public License
HSE	High Speed Ethernet
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
ISA	The International Society of Automation
JTAG/SWD	Joint Test Action Group/Serial Wire Debug
LAS	Link Active Scheduler
LED	Light Emitting Diode
LM	Link Master
MAU	Media Access Unit
MCU	Machine Control Unit
OD	Object Dictionary
OSI	Open Systems Interconnect model
PA	Process Automation
PC	Personal Computer
PDU	Protocol Data Unit
PID	Proportional-Integral-Derivative
SDU	Service Data Unit
UART	Universal Asynchronous Receiver / Transmitter
USB	Universal Serial Bus
VCR	Virtual Communications Relationship
VFD	Virtual Fieldbus Device

Literatura

1. **Böhm, Pavel.** Komunikační modul pro průmyslovou sběrnici FOUNDATION Fieldbus. [Diplomová práce]. 2012.

2. **Glanzer, David A.** *FOUNDATION Fieldbus Technical Overview.* 2003. FD-043 Revision 3.0.

3. **Šponar, Radek.** Vlastnosti a užití průmyslových sběrnic. *Elektrorevue.* [Online] 5. 4 2004. [Citace: 10. 4 2013.] http://www.elektrorevue.cz/clanky/04019/index.html.

4. Yokogawa Electric Corporation. *FOUNDATION Fieldbus Book – A Tutorial.* [Online] 4 2012. [Citace: 9. 4 2013.] http://www.yokogawa.com/pdf/provide/E/GW/TI/0000001497/0/TI38K02A01-01E.pdf.

5. **Verhappen, Ian a Pereira, Augustino.** *Foundation Fieldbus, 3rd Edition.* Summit, NJ : ISA, 2008. 978-1934394762.

6. *Unified Fieldbus Controller, UFC100 User's Manual.* [Dokument PDF]

7. **Mahalik, N. P.** *Fieldbus Technology: Industrial Network Standards for Real-Time Distributed Control.* s.l. : Springer, 2003. 3-540-4183-0.

8. **Sadasivan, Shyam.** *An Introduction to the ARM Cortex-M3 Processor.* [Dokument PDF] 2006.

9. *EFM32G Reference Manual.* [Dokument PDF] Oslo : autor neznámý, 2011.

10. *Datasheet AMIS-492x0 Fielbus MAU.* [Dokument PDF] Denver, Colorado : autor neznámý, 2008.

11. *FT232R USB UART IC Datasheet.* [Dokument PDF] Glasgow : Future Technology Devices International Limited, 2010.

12. Představujeme FreeRTOS. *MCUhobby.cz.* [Online] 10. 1 2010. [Citace: 9. 4 2013.] http://www.mcuhobby.cz/2010/01/predstavujeme-freertos/.

13. Base64. *Wikipedia, the free encyclopedia.* [Online] 26. 4 2013. [Citace: 27. 4 2013.] http://en.wikipedia.org/wiki/Base64.

14. How do I base64 encode (decode) in C? *Stack Overflow.* [Online] 4. 12 2008. [Citace: 10. 4 2013.] http://stackoverflow.com/questions/342409/how-do-i-base64-encode-decode-in-c.

15. **Bayer, Jürgen.** *C# 2005 Velká kniha řešení.* Brno : Computer Press, a.s., 2007. 978-80-251-1620-3.

16. **Wassenhove, Tim Van.** Presenting the SortableBindingList<T>. *Tim Van Wassenhove.* [Online] 22. 2 2007. [Citace: 10. 4 2013.] http://www.timvw.be/2007/02/22/presenting-the-sortablebindinglistt.

17. **Norendal, J.** *The MODbus to FF build-in, the T610.* [Dokument PDF]

18. *NI-FBUS Hardware and Software User Manual.* [Dokument PDF]

19. *NI-FBUS Configurator User Manual.* [Dokument PDF]

Seznam obrázků

Seznam tabulek

Přílohy

A. Uživatelský manuál k monitorovacímu programu

Program je určený pro zobrazení, filtraci a export dat zachycených komunikačním modulem na sběrnici FF. Je naprogramován v jazyce C# s použitím knihovny .NET Framework 4.0, která je bezpodmínečně nutná pro spuštění programu.

Knihovnu .NET Framework 4.0 lze běžně stáhnout pro danou verzi Windows. Seznam podporovaných verzí lze vidět v následující tabulce (Tabulka 6).

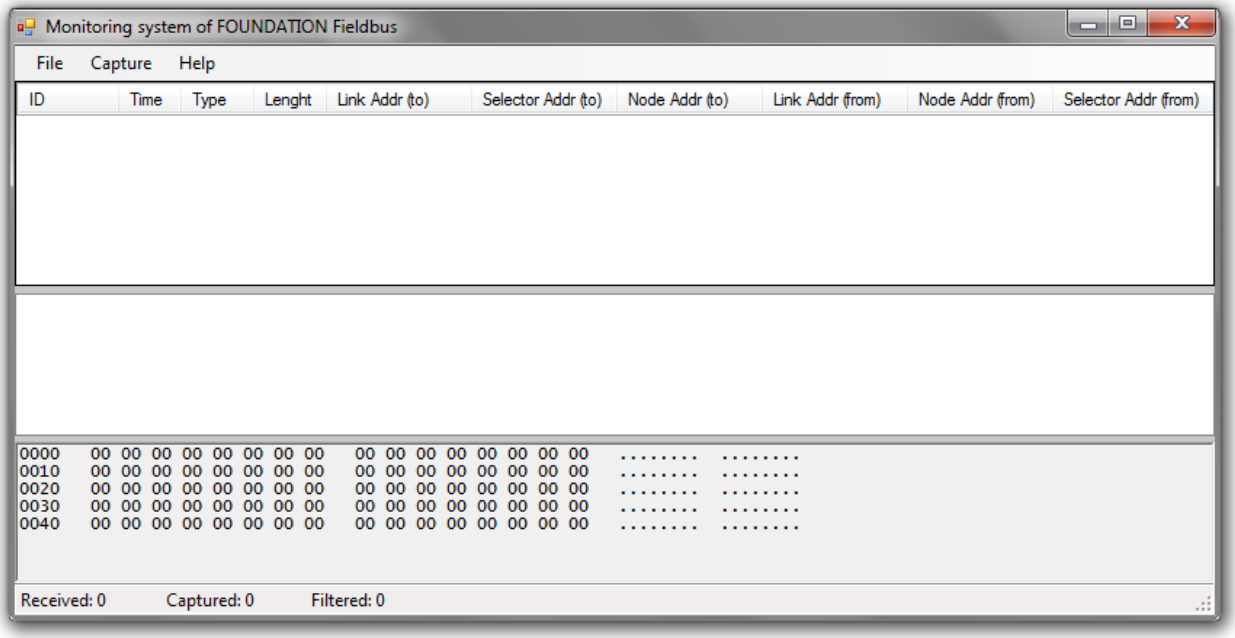
Tabulka 6: Seznam podporovaných OS

Windows XP SP3
Windows Server 2003 SP2
Windows Vista SP1 nebo novější
Windows Server 2008 (není podporován v roli jádra serveru)
Windows 7
Windows Server 2008 R2 (není podporován v roli jádra serveru)
Windows 7 SP1
Windows Server 2008 R2 SP1

V rámci každého operačního systému jsou podporovány architektury x86 i x64. A minimální požadavky na hardware jsou:

- minimální procesor: Pentium III - 1 GHz
- paměť RAM: 512 MB
- požadované místo na disku pro x86: 600 MB
- požadované místo na disku pro x64: 1.5 GB

Uživatelské rozhraní programu je jednoduše uspořádáno (viz Obrázek 20).



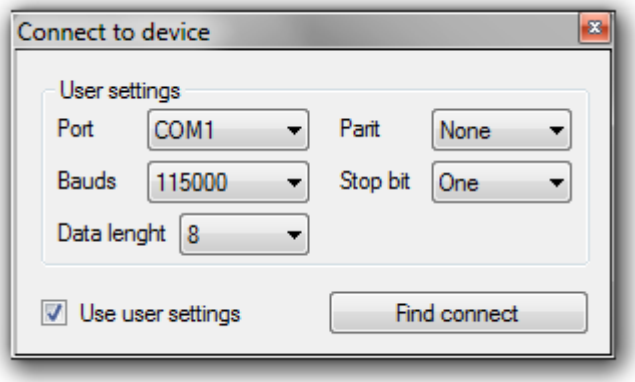
Obrázek 20: Okno programu při spuštění

Okno programu je rozděleno do tří podsekcí. V horní podsekci je tabulka se seznamem zachycených rámců. Sloupce tabulky jsou dynamické, takže po kliknutí lze obsah tabulky seřadit. V tabulce se zobrazují pouze rámce, které projdou filtrační procedurou.

Další podsekce je v prostřední části okna. Slouží pro detailní zobrazení informací o zachyceném rámci (viz Obrázek 15), který lze vybrat kliknutím na řádek tabulky z první podsekce.

V poslední podsekci se zobrazuje vybraný rámec jako hexadecimální data uspořádaná do dvou sloupců vždy po osmi bytech. Vedle hlavních sloupců jsou ještě dva další sloupce, které data reprezentují jako ASCII znaky, je-li to možné. V případě, že se jedná o netisknutý znak, je znak nahrazen tečkou.

Pro připojení ke komunikačnímu modulu stiskneme menu *File* → *Connect*. Otevře se dialogové okno (viz Obrázek 21), ve kterém je možno nastavit parametry připojení. V případě automatického vyhledání komunikačního modulu stačí nezatrhnout políčko *Use user settings*.



Obrázek 21: Dialogové okno pro nastavení připojení ke komunikačnímu modulu

Po úspěšném připojení je okno automaticky uzavřeno. Není-li komunikační modul připojen ke sběrnici, nebo není-li na sběrnici žádný provoz, monitorovací program testuje každé dvě vteřiny, zda-li je spojení s modulem aktivní.

Pro zahájení zaznamenávání obdržených rámců od komunikačního modulu stiskneme menu *Capture → Start*. Průběh zaznamenávání jednotlivých rámců lze vidět na Obrázku 15. Pro ukončení zaznamenávání obdobně stiskneme menu *Capture → Stop*. Pokud chceme všechny doposud zaznamenané rámce smazat, stiskneme menu *Capture → Clear*.

Pro nastavení filtrování zaznamenaných rámců stiskneme menu *Capture → Filter*, čímž se otevře dialogové okno s nastavením filtračních pravidel (viz Obrázek 16). Filtrovat lze podle typu, obsahu nebo délky rámce. Dále lze rámce rozlišit podle jednotlivých adres. Při potvrzení nastavení filtru je ihned aktivní pro nově příchozí rámce. Již zaznamenané rámce se znovu přehodnotí podle nových filtračních pravidel a zobrazí se v tabulce.

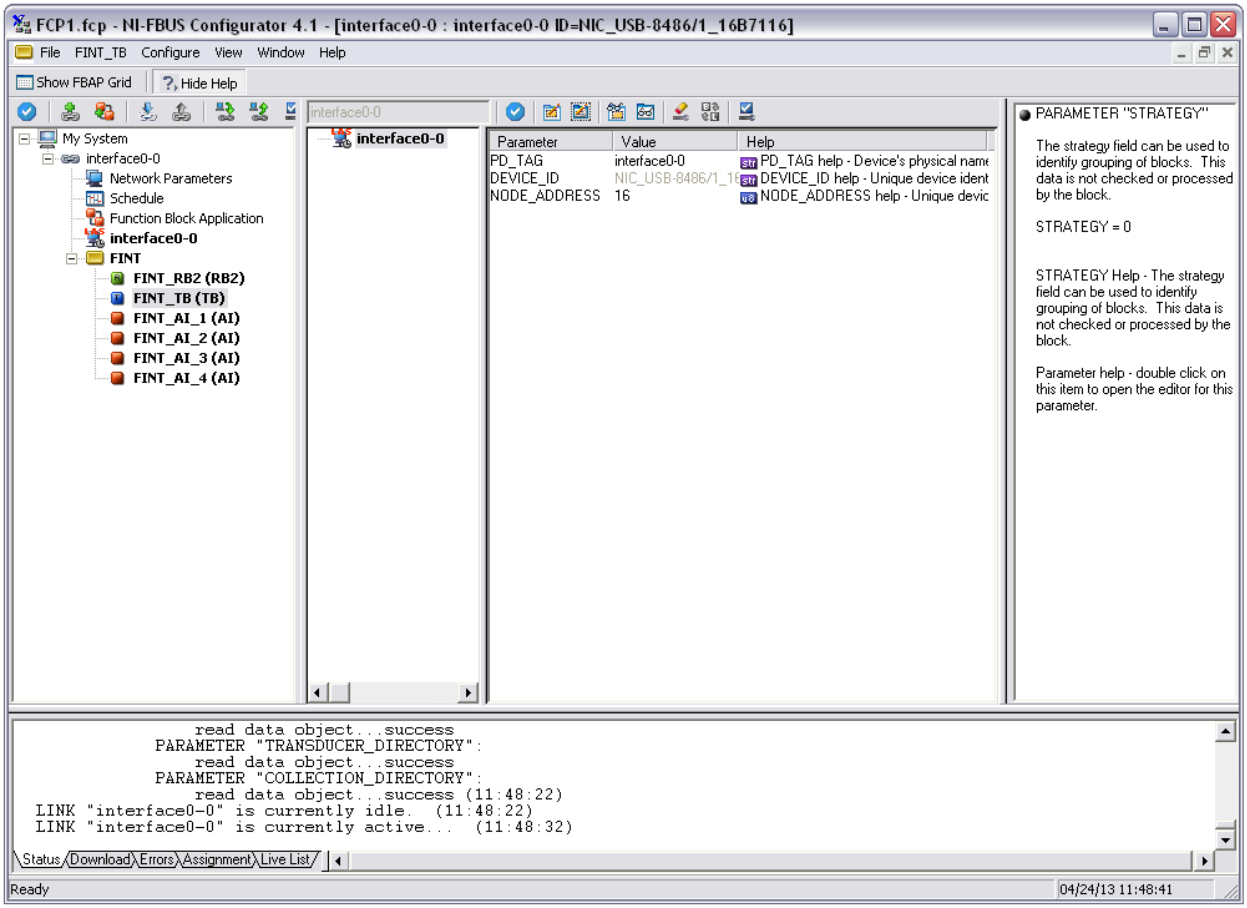
Informace o počtu přeposlaných, zaznamenaných a vyfiltrovaných rámců během připojení jsou ve stavovém řádku okna.

Export zaznamenaných dat lze provést stisknutím menu *File → Export data*, čímž se otevře dialogové okno pro vybrání názvu souboru a cesty uložení. Formát uloženého souboru je typu CSV. Příkladný obsah exportovaného souboru lze vidět na Obrázku 17.

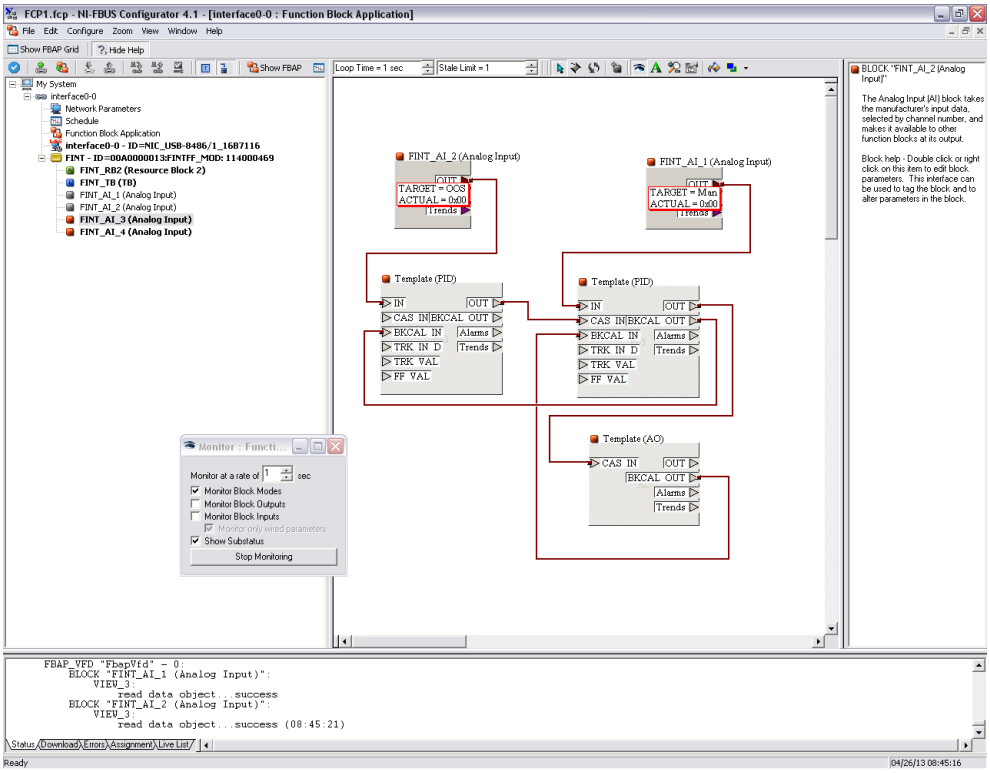
Pro ukončení programu stiskneme menu *File → Exit*. Program automaticky provede odpojení do komunikačního modulu.

B. Ukázka programu NI-FBUS

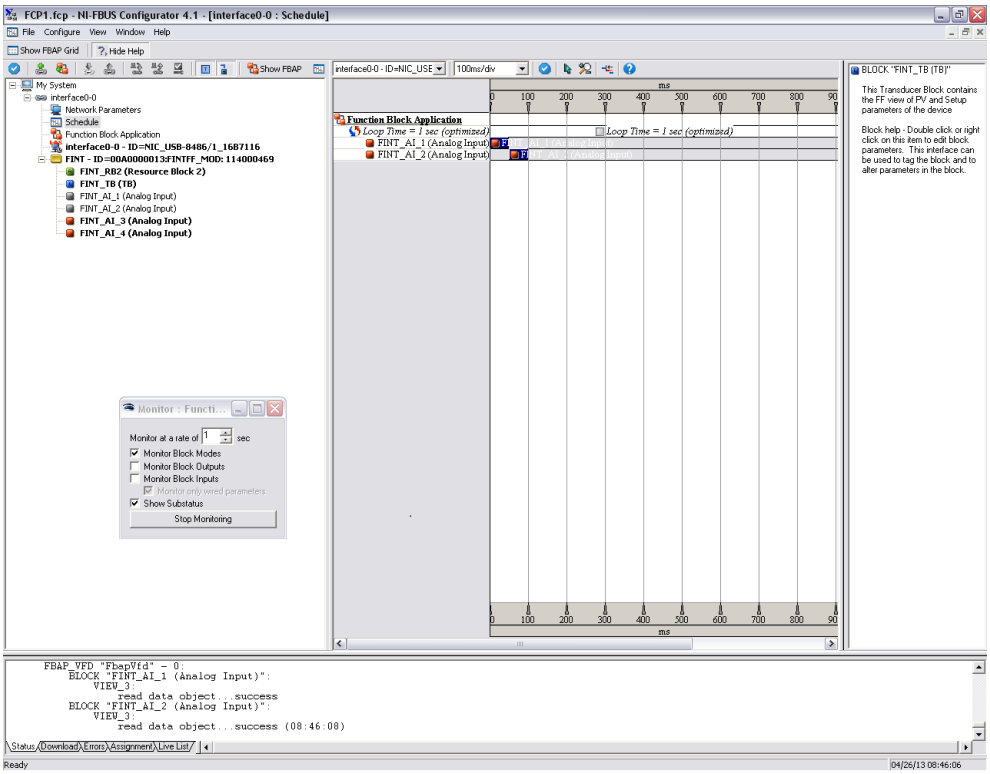
NI-FBUS Configurator je samostatná aplikace, která poskytuje konfigurační nástroje pro nastavování FF segmentů. Konfigurátor automaticky umí detekovat zařízení (viz Obrázek 22) na sběrnici, rozeznat jednotlivé funkční bloky zařízení a následně je umožní používat v grafickém návrhovém prostředí (viz Obrázek 23). Další výhodou je automatická optimalizace a celkový návrh plánu pro obsluhu jednotlivých funkčních bloků (viz Obrázek 24).



Obrázek 22: Ukázka programu NI-FBUS Configurator s detekovaným zařízením FINT T610

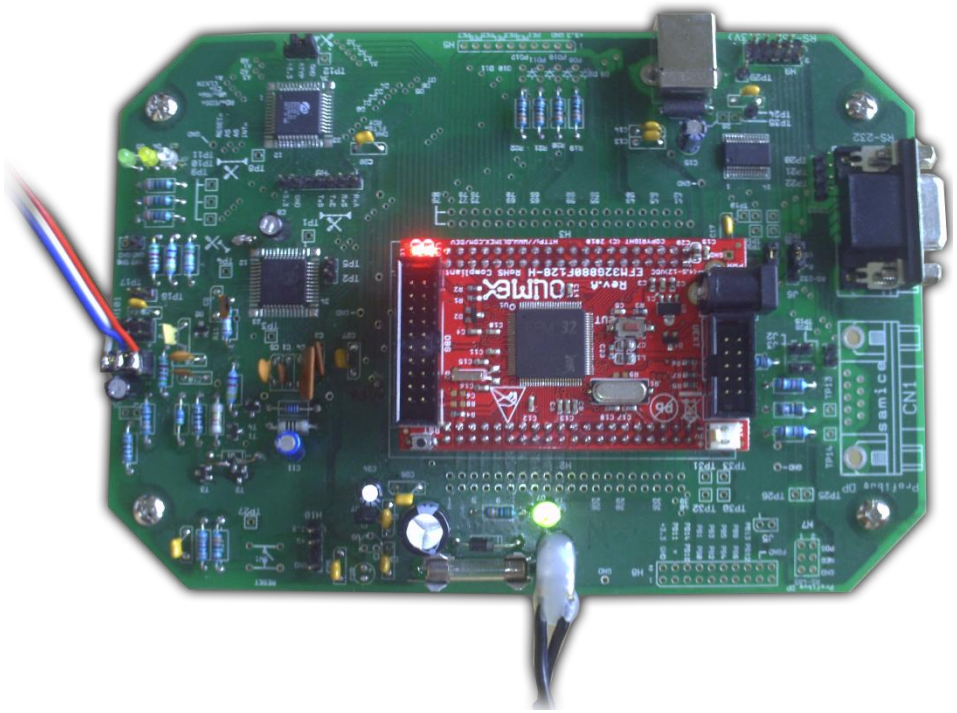


Obrázek 23: Příkladné zapojení funkčních bloků v NI-FBUS

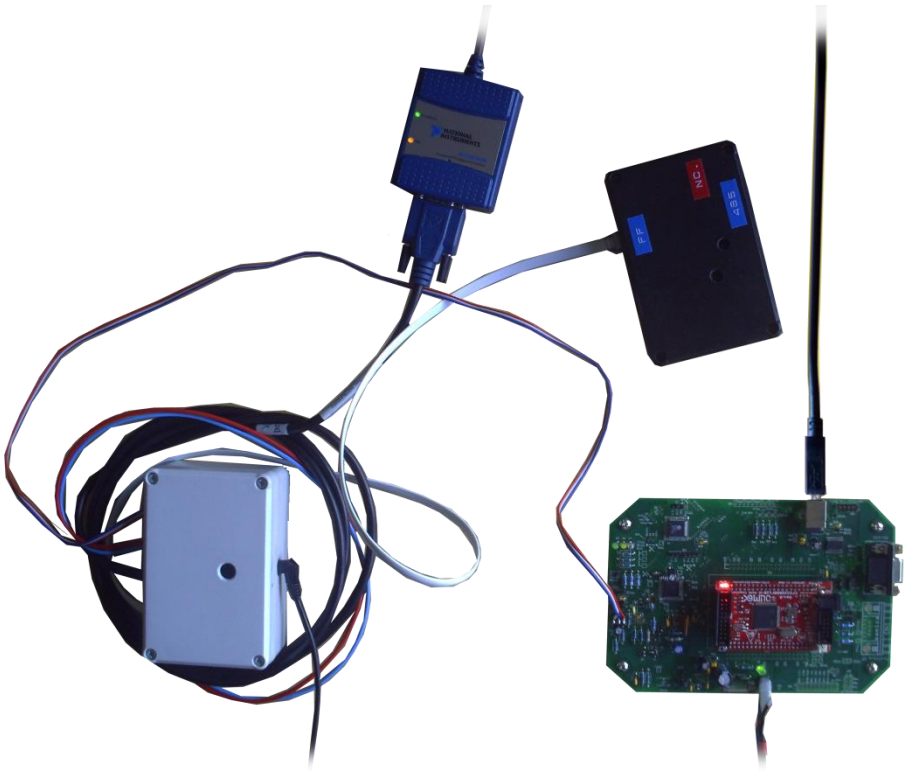


Obrázek 24: Navržený plán obsluhy funkčních bloků v NI-FBUS

C. Fotodokumentace



Obrázek 25: Komunikační modul během provozu



Obrázek 26: Zařízení použité pro experimentální segment sběrnice FF