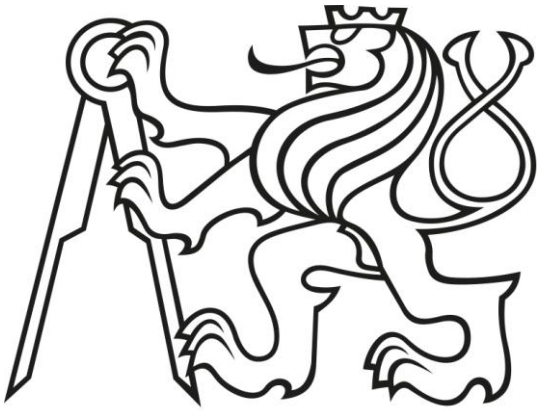
České vysoké učení technické v Praze

Fakulta elektrotechnická

BAKALÁŘSKÁ PRÁCE



Jiří Kalousek

Software pro sběr a zpracování dat z mnohakanálového scintilačního detektoru

Katedra řídicí techniky

Vedoucí bakalářské práce:	Ing. Jiří Zemánek
Studijní program:	Systémy a řízení
Studijní obor:	Kybernetika a robotika

Praha 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jiří Kalousek**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Software pro sběr a zpracování dat z mnohakanálového scintilačního detektoru**

Pokyny pro vypracování:

1. Zdokumentujte stávající realizaci hardware (modul jednokanálového A/D převodníku se vzorkovací rychlostí 100 MSPS doplněný hradlovým polem, agregace více modulů do komplexního systému) pro sběr dat z mnohakanálového scintilačního detektoru pro studium reakcí anti-neutrina.
2. Vytvořte software, který bude ovládat hardwarové zařízení (konfigurace, řízení akvizice), přijímat, vizualizovat a ukládat data z jednotlivých převodníků. Data budou zpracovávána v závislosti na požadavcích uživatele – časové, spektrální analýzy.
3. Diskutujte možnosti použití jiných softwarových nástrojů pro tento účel.

Seznam odborné literatury:

[1] FTDI Ltd. Software Application Development D2XX Programmer´s Guide [online]. Glasgow, 2012. Dostupné z: http://www.ftdichip.com/Support/Documents/ProgramGuides.htm
[2] HEROUT, Pavel. Učebnice jazyka Java. 5., rozš. vyd. České Budějovice: Kopp, 2010, 386 s. ISBN 978-80-7232-398-2.

Vedoucí: Ing. Jiří Zemánek

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 29. 2. 2016

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem č. 1/20.

V Praze dne ....

Název práce: Software pro sběr a zpracování dat z mnohakanálového scintilačního detektoru

Autor: Jiří Kalousek

Katedra: Katedra řídicí techniky

Vedoucí bakalářské práce: Ing. Jiří Zemánek, Katedra řídicí techniky

Abstrakt: Úkolem této bakalářské práce je návrh a realizace softwarového nástroje pro studium anti-neutrina. V úvodní části této práce je zdokumentována stávající realizace hardwaru, kterým je modulární systém s A/D převodníky se vzorkovací rychlostí 100 MSPS. Toto zařízení slouží pro sběr dat z mnohakanálového scintilačního detektoru pro studium reakcí anti-neutrina. V hlavní části práce je popsán návrh a implementace softwaru pro ovládání zdokumentovaného hardwarového zařízení a vizualizaci pořízených dat. Výsledný software byl napsán v jazyce Java s využitím knihovny pro komunikaci s FTDI čipem, převádějícím USB komunikaci na komunikaci po paralelní sběrnici. Uživateli je umožněno pohodlné ovládání celého systému i jeho částí. Data přijatá ze systému mohou být softwarem dále zpracována a uložena do souborů pro pozdější analýzu.

Klíčová slova: A/D převodník, vzorkování, FPGA, mnohakanálové zpracování dat, Java

Title: Software tool for data acquisition and processing for the multichannel scintillation detector

Author: Jiří Kalousek

Department: Department of Control Engineering

Supervisor: Ing. Jiří Zemánek, Department of control engineering

Abstract: The task of this bachelor thesis is to design a new software tool for the anti-neutrino research. Firstly, an existing hardware device is described. The device consists of modules with A/D converters running at sample rate of 100 MSPS. The device performs data taking from the multichannel scintillation detector. Secondly, the design part and the implementation part of the developed software are described. The final software project is programmed in Java with a use of library for the connection with FTDI device that creates bridge between USB and parallel bus. This software is dedicated for controlling the hardware part and visualizing of acquired data. The user of developed tool is allowed to control the whole system and its parts. Acquired data can be processed and saved to a file for later analysis.

Keywords: A/D converter, sampling, FPGA, multichannel data processing, Java

Poděkování

Rád bych poděkoval vedoucímu mé práce Ing. Jiřímu Zemánkovi za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce. Dále bych chtěl poděkovat řediteli Ústavu technické a experimentální fyziky ČVUT doc. Ing. Ivanu Šteklovi, CSc. a taktéž bývalému řediteli Ing. Stanislavu Pospíšilovi, DrSc. za možnost spolupráce na projektu detektoru anti-neutrin. Další poděkování bych rád směřoval kolektivu kolegů z Ústavu technické a experimentální fyziky ČVUT za odbornou pomoc, rady a připomínky. Rovněž bych rád poděkoval své rodině a přátelům za podporu.

Obsah

1 Úvod

Vědci v oboru částicové fyziky se po celém světě v posledních letech čím dál více zaměřují na výzkum neutrinové fyziky. Neutrina jsou částice patřící do skupiny leptonů. Leptony se dělí na nabité leptony (elektron, mion, tauon) a neutrina (elektronové, mionové a tauonové). K těmto šesti elementárním částicím je známo také šest jejich antičástic [1]. Před několika lety byla předpovězena nová hypotetická částice – sterilní neutrino. Tato částice by oproti třem ostatním neměla podléhat slabé interakci, měla by na ni působit pouze gravitace [2]. Z tohoto důvodu nebylo zatím možné tuto částici zachytit dosavadními typy detektorů. Aby mohla být ověřena její existence, bylo potřeba vyvinout nový typ detektoru. Tímto vývojem se zabývá Mgr. Lukáš Fajt ve své diplomové práci Prototyp detektoru reaktorových antineutrin [3].

Kromě ověření existence sterilního neutrina může být nový detektor využit ke sledování výkonu jaderného reaktoru a za určitých okolností i k určení složení jaderného paliva v reaktoru [3].

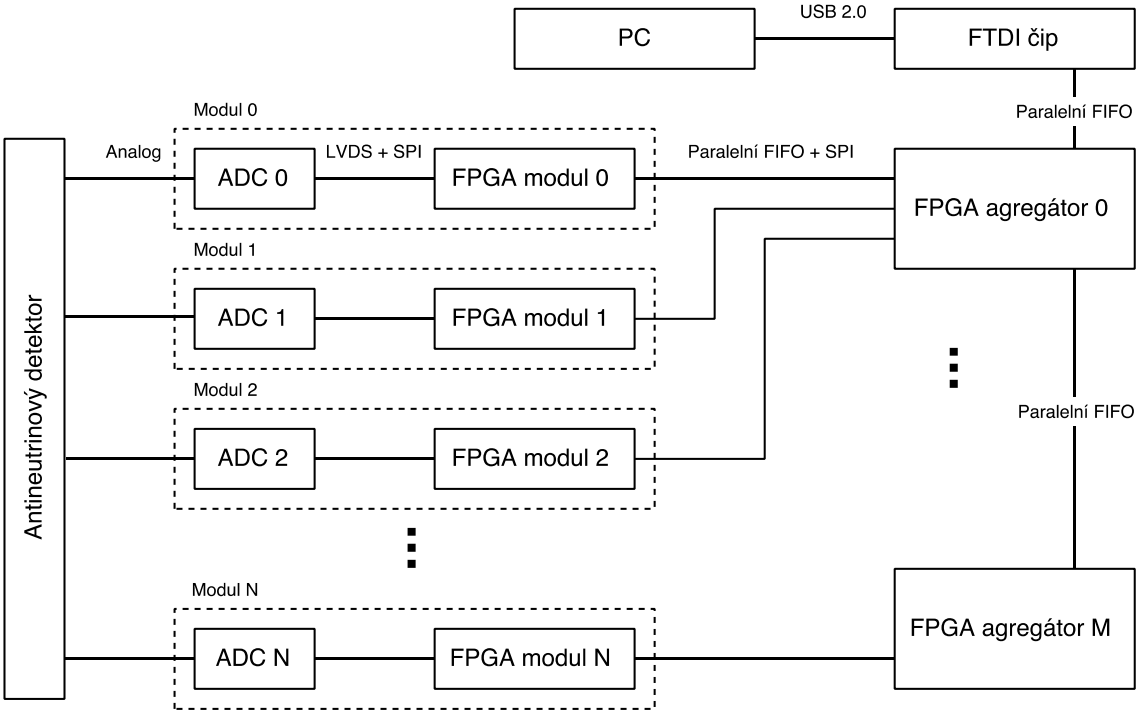
Prototyp detektoru reaktorových antineutrin byl vyvinut v Ústavu technické a experimentální fyziky (ÚTEF) ČVUT ve spolupráci s kolegy z JINR Dubna. Potřebná elektronika zpracovávající analogové elektrické signály z detektoru byla navržena a zrealizována v ÚTEF ČVUT. Signály jsou digitalizovány z několika kanálů zároveň. Získaná data jsou odesílána přes rozhraní USB do počítače, ze kterého je celý experiment řízen.

Cíl práce

Jedním z cílů této práce bylo popsat elektronickou část sestávající z modulů obsahujících A/D převodník a FPGA obvod a z agregátorů těchto modulů vytvářejících jeden komplexní systém pro sběr dat z několika kanálů. Blokové schéma celého systému včetně připojení k PC je znázorněno na obrázku (Obrázek 1). Hlavním úkolem po prostudování funkce elektronické části a způsobu komunikace byl návrh softwaru, který umožní uživateli pohodlné ovládání fyzikálního experimentu a zároveň efektivní sběr dat a jejich vizualizaci. Tato práce se zabývá i popisem zrealizovaného softwaru.

V první řadě je nutné zajistit vyhledání připojeného zařízení, následně navázání a udržování komunikace s hardwarovou částí systému a odesílání správných sekvencí příkazů podle požadavků uživatele na konfiguraci systému. Dále má být softwarem zajištěno ukládání přijatých dat do souborů, jejichž formát je součástí návrhu. Volitelně je také možné provádět předzpracování přijatých dat a až následně ukládat upravená data do souborů. Zároveň je požadováno, aby mohl uživatel sledovat přijímaná data v přímém přenosu zobrazovaná v podobně grafů, tabulek apod. Rozbor konkrétních požadavků je proveden v kapitole 3.

Jedním z požadavků byla také přenositelnost tohoto uživatelského softwaru mezi platformami (OS Windows, Linux). Zároveň bylo žádoucí, aby software fungoval dostatečně efektivně, aby bylo možné rychle ukládat příchozí proud dat. Bylo tedy nutné zvážit všechny požadavky i s ohledem na předpokládané vlastnosti budoucích experimentů a použití zmíněného detektoru v kompletním systému.



Obrázek 1 - Blokové schéma digitalizačního systému

2 Elektronická část

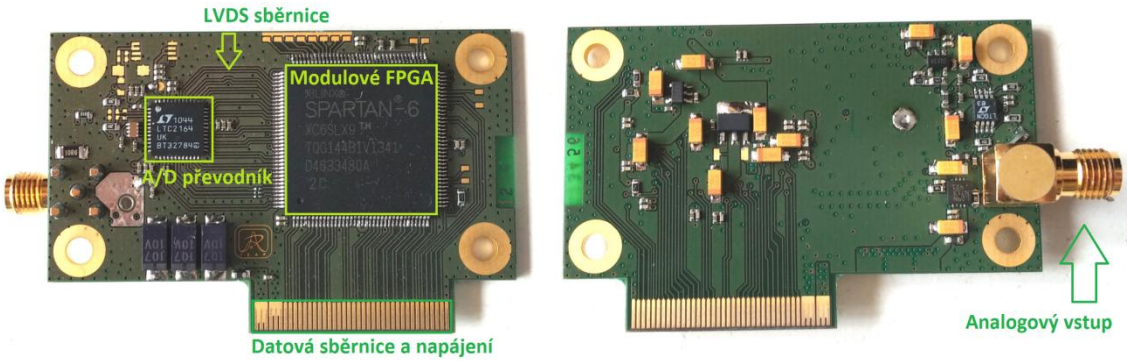
Elektronická část systému je tvořena *digitalizačním modulem*, ve kterém je prováděno vzorkování analogového signálu z detektoru v A/D převodníku. Vzorky jsou odeslány do hradlového pole (FPGA), kde je prováděno předzpracování dat z jednoho kanálu v závislosti na nastaveném režimu. Dále je pomocí řídicích signálů kontrolováno předání dat do dalšího FPGA, které je umístěné na desce sdružující několik modulů. Přenos dat probíhá dále mezi zřetězenými deskami, až jsou data odeslána přes FTDI čip (*komunikační modul*) do počítače.

2.1 Digitalizační modul

Digitalizační modul obsahuje na jedné desce A/D převodník a hradlové pole, které umožňuje variabilně měnit činnost modulu podle zvoleného režimu ve firmware, čímž je modul značně univerzální při dané podobě hardwaru. Modul je navržen jako nesamostatná jednotka určená pro zabudování do složitějších vícekanálových systémů, neobsahuje totiž konfigurační paměť. Firmware je poskytován nadřazeným systémem. Změna činnosti i další vývoj systému je umožněn změnou firmwaru v FPGA.

Signál z detektoru je přiváděn na analogový vstup modulu. Nastavení a příkazy pro modul jsou přijímány po rozhraní SPI[1]. Data z modulu jsou posílána po paralelní sběrnici. Napájení modulu je zajištěno z jednoho vstupního napájecího napětí (typicky 5 V) z nadřazeného systému, ke kterému je modul připojen.

Fyzická podoba digitalizačního modulu je zobrazena na obrázku (Obrázek 2).

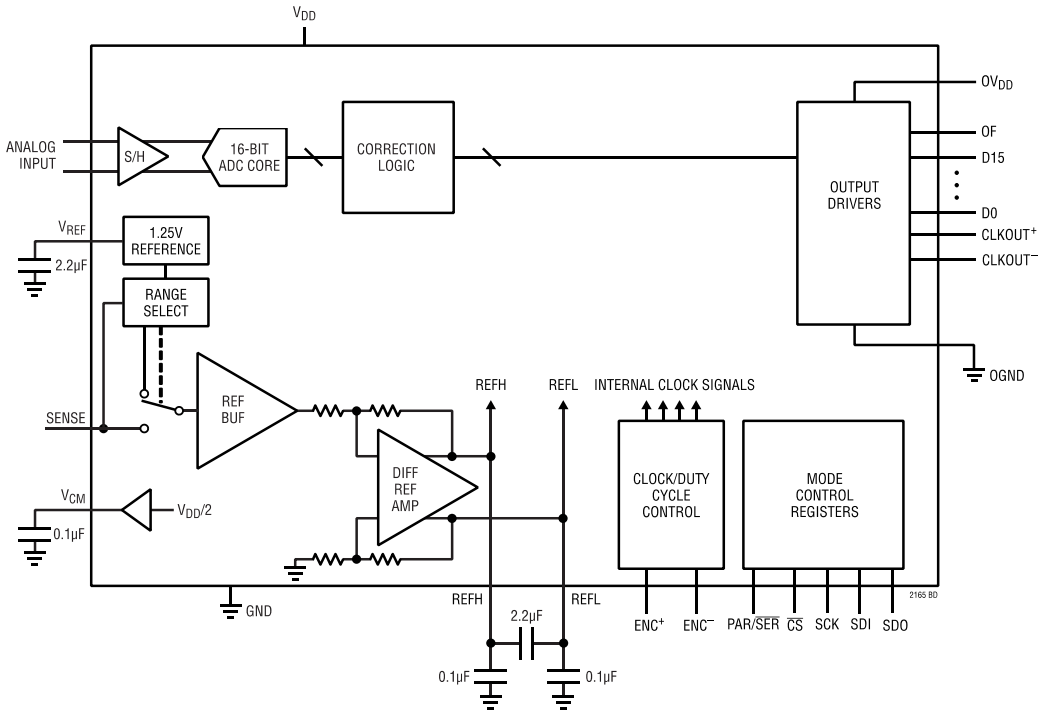


Obrázek 2 – Digitalizační modul

[1] SPI (Serial Peripheral Interface) je sériové komunikační rozhraní určené pro komunikaci mezi mikroprocesory a různými elektronickými integrovanými obvody. Jedno z komunikujících zařízení je tzv. master (řídí komunikaci) a ostatní zařízení jsou označována jako slave (podřízené zařízení). Dva vodiče tohoto rozhraní jsou vyhrazeny pro přenos dat (MISO a MOSI), jeden vodič je určen k šíření hodinového signálu (SCLK) a posledním vodičem je podřízenému zařízení povolována komunikace (SS) [4].

2.1.1 Analogový vstup a vzorkování

K převodu analogového signálu z detektoru na digitální data je použit A/D převodník LTC2164 (výrobce Linear Technology). Tento A/D převodník je 16bitový a je určený pro digitalizaci vysokofrekvenčních signálů. Jeho analogový vstup je diferenciální s výběrem vstupního rozsahu od 1 V do 2 V (peak-to-peak). Parametry převodníku, např. režim datového výstupu a další, se nastavují ve vnitřních registrech přes SPI rozhraní [5]. Vnitřní blokové schéma A/D převodníku je na obrázku (Obrázek 3).



Obrázek 3 - Blokové schéma AD převodníku [5]

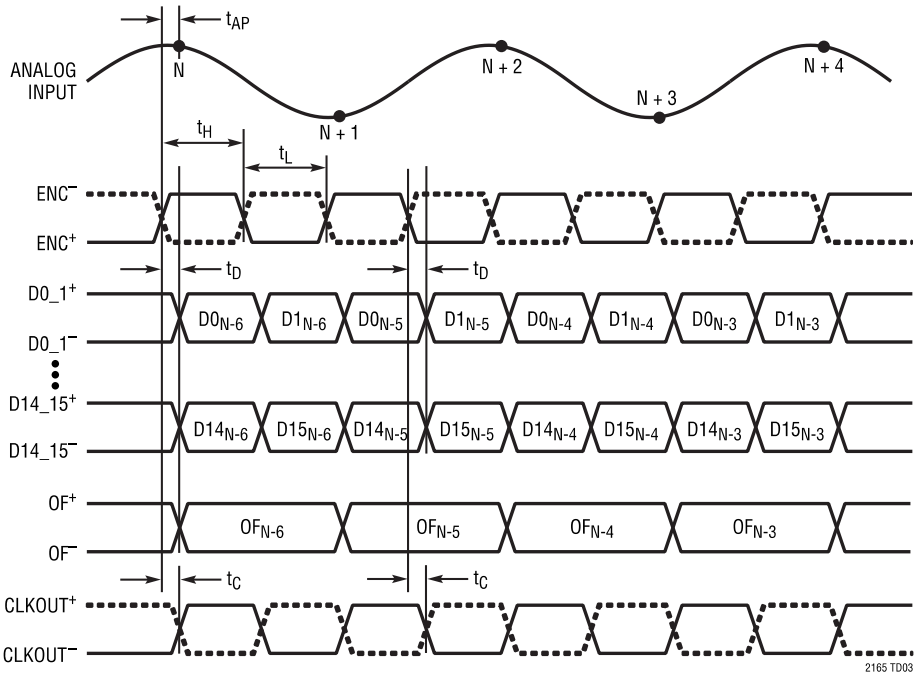
Před samotným A/D převodníkem je umístěn vstupní obvod úpravy signálu (převod do diferenciální formy, antialiasingový filtr), který byl navržen s ohledem na napěťové úrovně, které bude potřeba digitalizovat, aby bylo využito celého vstupního rozsahu převodníku. Situaci lze přirovnat k výběru vhodného měřicího rozsahu při měření např. napětí.

Vzhledem k malému vstupnímu rozsahu A/D převodníku je použit vstupní atenuátor. Jedná se v podstatě o napěťový dělič, kterým je prováděno zeslabení. S tímto obvodem je vstupní rozsah nastaven na ±5 V. Nastavení tohoto rozsahu je pevné, neměnné na úrovni osazení plošného spoje, a není možné ho softwarově měnit. Za vstupním atenuátorem je zapojen operační zesilovač se ziskem 1 jako vstupní oddělovací zesilovač, který převádí analogový jednopólový signál na diferenciální.

Vstupní hodinový signál pro A/D konverzi je 100 MHz. V každém hodinovém taktu je na výstupu příslušný vzorek. Použitý přenos je Double Data Rate LVDS (Low-Voltage Differential Signaling). Tento standard pracuje s nízkými úrovněmi napětí a umožňuje dosažení vysokých rychlostí komunikace. Proto je používán například v informačních systémech v automobilech, v přenosu dat z průmyslových kamer, v komunikaci jednotlivých komponent

osobního počítače a dalších komunikačních systémech a aplikacích, které vyžadují vysokou komunikační rychlost a zároveň odolnost vůči rušení [6].

Přenos dat z použitého A/D převodníku je v tomto režimu proveden tak, že jsou po jednom diferenciálním páru vodičů přenášeny dva bity v jednom hodinovém cyklu při obou hranách hodinového signálu [5]. Průběh signálů je na obrázku (Obrázek 4).



Obrázek 4 - Časový průběh signálů A/D převodníku komunikujícího v režimu Double Data Rate LVDS [7]

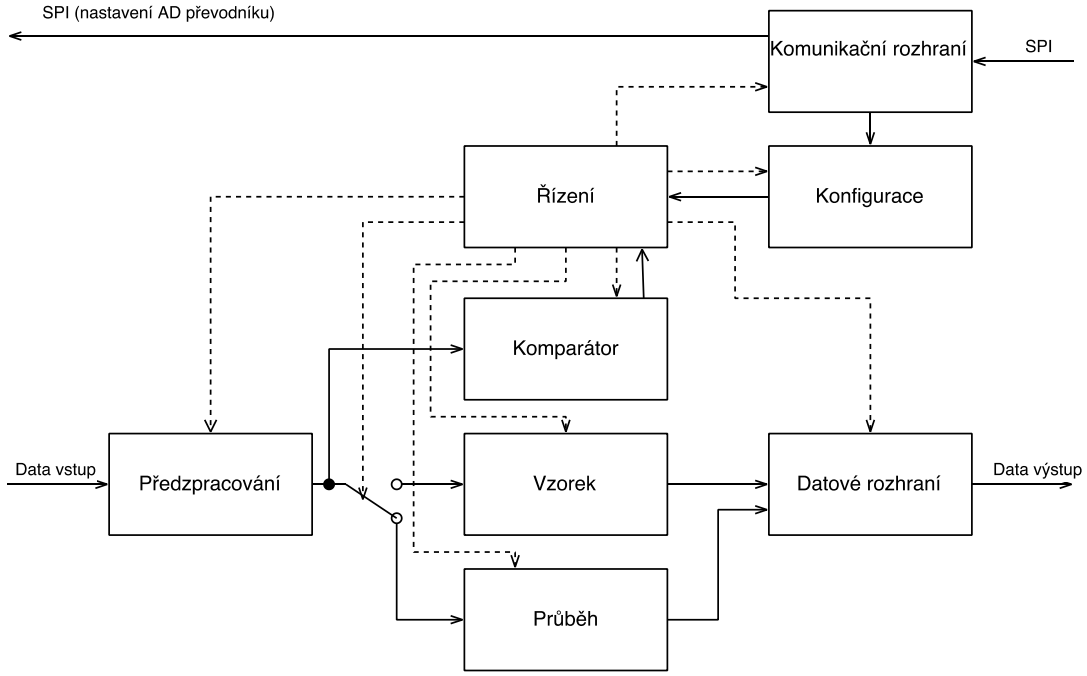
Pravidelné přepínání velkého množství bitů (logických hradel) může způsobovat kolísání napájecího napětí v důsledku proudových rázů a rušení citlivé analogové části obvodů. Tak může dojít k nepřesnosti při konverzi vstupního analogového signálu. Obvod A/D převodníku může být z hlediska formátu odesílaných dat nastaven do režimu, kdy jsou výstupní data upravena tak, aby byly proudové rázy přesunuty do oblastí s větším vstupním signálem a aby přepínání většího množství bitů nastávalo nepravidelně. Za použití takového režimu je pak množství změn rozprostřeno do celého rozsahu. Úprava spočívá v aplikaci logické operace XOR mezi LSB (nejméně významný bit) a všemi ostatními výstupními bity. Při tom bit LSB a ostatní nedatové výstupy nejsou ovlivněny [5]. Ve výsledku se tedy například změna z binárního čísla 01111111_B na 10000000_B transformuje na změnu z čísla 10000001_B na 10000000_B. Tato úprava přenášených dat bude dále označována jako randomizace a inverzní proces, který je potřeba provést na přijímací straně, jako derandomizace.

Další možností, jak může být omezeno případné rušení způsobené přepínáním velkého množství datových bitů, je nastavení režimu střídání polarity bitů. V tomto režimu je ještě před výstupním bufferem realizována inverze lichých bitů. Ostatní bity nejsou ovlivněny [5]. Na přijímací straně musí být opět provedena inverzní operace. Tyto procesy budou dále označovány jako alterizace a dealterizace.

2.1.2 Zpracování dat v FPGA

Data z A/D převodníku jsou odesílána LVDS sběrnicí do FPGA z rodiny Spartan-6, typ XC6SLX9, od firmy Xilinx. Tímto FPGA je nastavován A/D převodník přes rozhraní SPI, jsou přijímána data od A/D převodníku a jsou posílána nadřazenému systém.

Do FPGA musí být nejdříve odeslána konfigurace, která určuje, jak má být FPGA obvodem nastaven A/D převodník, jak mají být zpracovávána příchozí data, kdy a jakým způsobem má být spuštěno ukládání vzorků do bufferu a další. Konfiguraci znázorňuje blok *Konfigurace*. Konfigurací je nastaven režim činnosti. Akvizice začíná v jednom z režimů až poté, co je modul aktivován řídicím příkazem a zároveň je splněna spouštěcí podmínka daná konfigurací pro daný režim. Blokem *Řízení* jsou kontrolovány všechny bloky podle konfigurace. Blokové schéma funkce FPGA je zobrazeno na obrázku (Obrázek 5).



Obrázek 5 - Blokové schéma funkce FPGA

Vstupní data jsou před dalšími operacemi podrobena derandomizaci a dealterizaci (blok *Předzpracování*). Datová slova o velikosti 16 bitů, která odpovídají jednotlivým vzorkům, jsou přiváděna do komparátoru (blok *Komparátor*), ve kterém probíhá srovnávání s nastavenými úrovněmi. Výsledný signál je předáván do dalších bloků, kde např. rozhoduje o začátku akvizice. Vzorky jsou v případě ukládání průběhu (*osciloskopický režim*) ukládány do paměťového bufferu (blok *Průběh*). Tento buffer je velký 20468 bajtů, resp. 10234 slov (vzorků). Vzorky jsou do něj ukládány při vzorkovací frekvenci 100 MHz každých 10 ns. Po skončení ukládání vzorků jsou data teprve odesílána dále. V jiném režimu (*režim vzorku*), kdy není požadováno ukládání celého průběhu, ale je snímán pouze jeden vzorek, je takový vzorek uložen a ihned odesílán (blok *Vzorek*).

Osciloskopický režim

V osciloskopickém režimu (v softwaru nastavován jako Waveform) jsou data ukládána do vnitřního bufferu. Ukládání je prováděno od okamžiku splnění spouštěcí podmínky do okamžiku splnění ukončovací podmínky. Tento režim je realizován komponentou Průběh.

Spouštěcí podmínky:

- hodnota vzorku překročila přednastavený práh
- externí spouštěcí signál o přednastavené hraně
- spouštěcí příkaz (ze softwaru)

Ukončovací podmínky:

- uložení přednastaveného počtu vzorků
- hodnota vzorku klesla pod přednastavený práh

Režim vzorku

V režimu vzorku (v softwaru nastavován jako Sample) je uložen pouze jeden vzorek. V tomto režimu je opět umožněno nastavení spouštěcí podmínky sejmutí vzorku. Ukončovací podmínka zde neexistuje.

Spouštěcí podmínky:

- maximum v signálu, který překonal přednastavený práh
- externí spouštěcí signál o přednastavené hraně
- spouštěcí příkaz (ze softwaru)

Tento režim může být nastaven také tak, aby uložil a odeslal i informaci o délce časového intervalu, kdy byla úroveň signálu nad prahem (v softwaru označován jako Sample+TOT).

2.1.3 Odeslání dat

Po ukončení akvizice, pokud je modulu povoleno odeslat připravená data, jsou tato odeslána po paralelní sběrnici. Sběrnice je navržena jako 16bitová s možností redukce na 8 bitů, což je aktuálně podporovaná varianta. Až když jsou všechna data odeslána, může být teprve spuštěna nová akvizice.

Odesílané vzorky jsou zabaleny do paketu, jehož formát je popsán v kapitole 4.1.3. Tento paket je složen z hlavičky, datové části, časové značky a kontrolního součtu.

Časová značka

V každém modulovém FPGA je implementován čítač pracující s frekvencí 100 MHz. Protože je jeden hodinový signál šířen celým systémem, je možné porovnávat časové značky různých modulů. Délka čítače v FPGA je 2^{48}-1, což při dané frekvenci umožňuje zaznamenat měření po dobu asi 32 dní. V případě přetečení se čítač vynuluje a čítá dále. Hodnotu čítače je možné vynulovat příkazem z řídicího počítače nebo externím signálem s přednastavenou hranou.

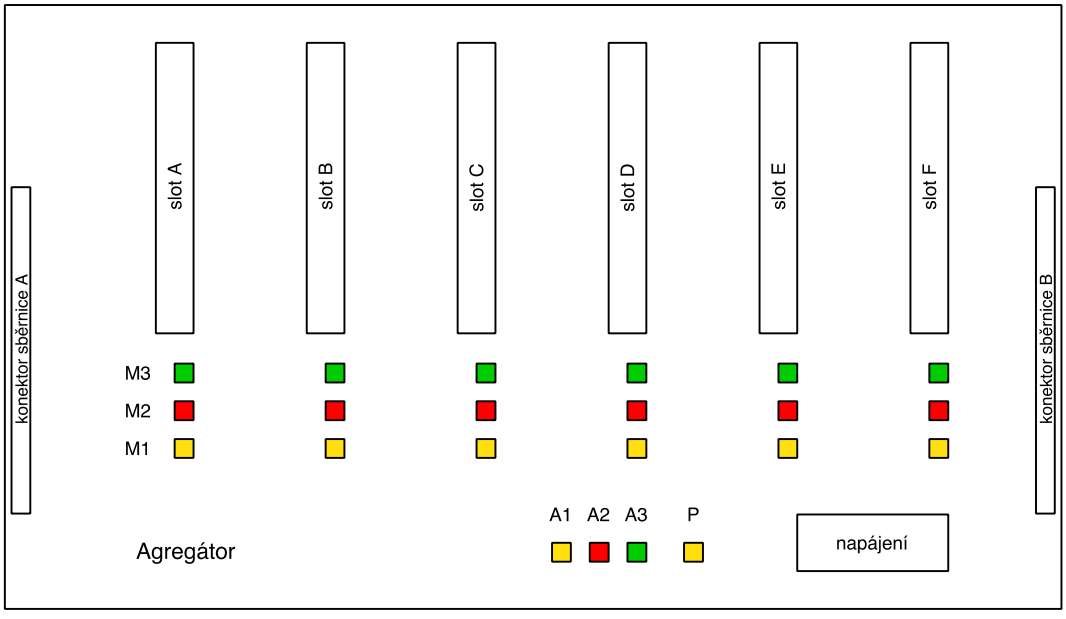
2.2 Agregátor modulů

Nadřazeným systémem pro jednotlivé moduly je tzv. deska *agregátoru* obsahující FPGA z rodiny Spartan-IIE, typ XC2S50E, od firmy Xillinx. Agregátor obsahuje zmíněné FPGA, které skrz 6 slotů komunikuje až se šesti moduly, LED stavové indikátory, komunikační sběrnice a napájecí obvody. Nákres rozložení jednotlivých částí je zobrazen na obrázku (Obrázek 6). Desky agregátoru je možné řetězit. Na začátku řetězce je umístěn komunikační modul s FTDI čipem pro připojení k řídicímu počítači.

Komunikace mezi deskami agregátorů probíhá po paralelní sběrnici s řízením toku dat. Přestože komunikace s FTDI čipem probíhá rozdílně, návrh firmwaru umožňuje nerozlišovat druh připojení. Data jsou vždy přenášena přes FPGA, ve kterém je rozhodnuto o jejich dalším směřování. Šířka datové sběrnice pro komunikaci mezi agregátorem a digitalizačním modulem je 8 bitů doplněných o 4 bity pro řízení komunikace. Konfigurační sběrnice mezi agregátorem a modulem je typu SPI.

2.2.1 Připojení agregátoru a signalizace stavů

Po připojení agregátoru k napájení je z externí paměti nahrán firmware do FPGA na desce agregátoru. Úspěšnost této operace je signalizována svítící LED P. Po propojení komunikačního modulu USB kabelem s řídicím počítačem se začne šířit systémem hodinový signál a je aktivováno nahrávání firmwaru z externí paměti do všech FPGA v digitalizačních modulech. Pokud je úspěšně nahrán firmware do všech připojených digitalizačních modulů, svítí LED M1 a bliká LED A1.



Obrázek 6 - Rozmístění indikačních LED na desce agregátoru

V dalším kroku dochází k adresaci digitalizačních modulů v systému. Bez adresace není možné s modulem komunikovat. Po přijetí adresačního paketu po sběrnici A (viz Obrázek 6) je v něm obsažená adresa přidělena slotu A. Další adresa je přidělena slotu B atd. Pokud jsou všechny sloty již naadresovány, jsou adresační pakety předávány dále po sběrnici. Naadresován je takto i slot, ke kterému není připojen modul, protože v danou dobu není tato informace známa. Že je slot naadresovaný, je signalizováno svítící LED M2. Pokud byly naadresovány všechny sloty dané desky agregátoru, přestane LED A1 blikat a začne svítit. Více o adresaci a formátu adresačního paketu je popsáno v kapitole 4.1.3.

Probíhající odesílání konfigurace do modulu je signalizováno blikáním LED M2. Probíhající přenos dat z modulu do agregátoru je signalizován blikající LED M3. Svítící LED M3 je signalizován stav, kdy je vyžadován přenos dat z modulu do agregátoru, ale agregátor je zahlcen a data nemohou být přijata a poslána dál. K tomuto stavu dojde, když je např. náhle odpojen kabel USB od komunikačního modulu.

2.2.2 Zpracování řídicích příkazů a dat v FPGA agregátoru

Řídicí příkazy prostupují systémem v podobě krátkých paketů. Každý paket obsahuje adresní bajt, na jehož základě je rozhodnuto o dalším zpracování. Pokud se adresa shoduje s některou z adres, která byla přidělena v daném agregátoru některému slotu, paket je poslán digitalizačnímu modulu v tomto slotu. V případě, že není adresa v tomto agregátoru přidělena, je paket odeslán po sběrnici dalšímu agregátoru a proces se analogicky opakuje. Zvláštní typ paketu je označen adresou nula. Tento paket je předán vždy všem slotům i následujícímu agregátoru.

Datové pakety z digitalizačních modulů přijímané hradlovým polem agregátoru jsou přenášeny přes vyrovnávací paměť typu FIFO o kapacitě 1000 vzorků. Ke každému paketu je připojena adresa slotu, ze kterého byl přijat. Zároveň jsou v FPGA přijímány i pakety přicházející od následujícího agregátoru v řetězci.

Komunikace mezi moduly a agregátorem i mezi agregátory je ovládána řídicími signály. Pokud jsou dostupná data pro předání nadřazenému článku, dojde k nastavení signálu přenosu. Detekce tohoto signálu způsobí povolení posílání dat nadřazeným prvkem. Pokud nadřazený prvek není schopen data dále přijímat, např. z důvodu přeplnění vyrovnávací paměti, deaktivuje signál až do doby jejího uvolnění. Přenos končí uvolněním signálu označujícím přítomnost dat k odeslání.

Průchod dat z více cest (modulů a dalšího agregátoru) do jedné datové cesty je řešen přepínáním datového spojení mezi cestami. V FPGA agregátoru je implementován blok *Řízení toku dat*, který využívá zmíněné signály ke kontrole datového toku. Cyklicky je ověřován požadavek na přenos dat ze všech modulů a podřízeného agregátoru. Pokud je takový požadavek nalezen, je realizován přenos podle předchozího popisu. Tato přenosová struktura je navržena tak, aby zabránila zahlcení celého systému jedním kanálem, např. z důvodu zvýšeného šumu, špatného nastavení apod.

Při očekávané četnosti událostí je ztráta dat způsobená nezachycením události z důvodu čekání modulu na dokončení odesílání minimální. Bližší ověření tohoto předpokladu bude možné až při reálném měření.

2.3 Komunikační modul

Přenos dat do řídicího počítače je zajištěn komunikačním modulem napojeným na paralelní sběrnici řetězce agregátorů. Hlavní částí komunikačního modulu je FTDI obvod FT232H. Tímto obvodem může být složitá komunikace protokolu USB 2.0 (Hi-Speed nebo Full Speed) převáděna na jednoduché sériové nebo paralelní sběrnice s různými konfiguracemi – JTAG, I^2C, SPI, UART (RS232), paralelní FIFO 245. Také mohou být nastavovány jednotlivé k tomu určené piny FTDI obvodu – obecně použitelné piny GPIO [7].

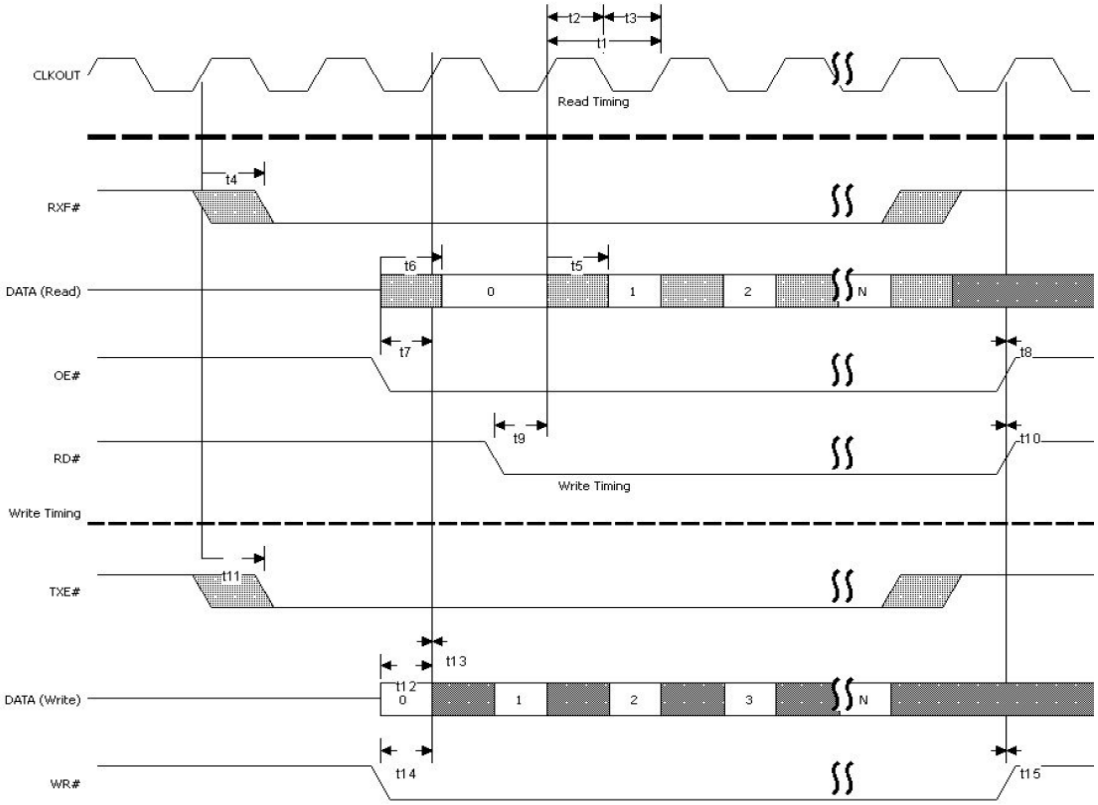
Součástí komunikačního modulu je EEPROM paměť, ve které je nahrána konfigurace FTDI čipu. Tuto paměť je po připojení modulu možno konfigurovat přes USB aplikací FT_Prog [8][9].

FTDI čip je pro přenos dat do počítače i pro posílání konfigurace a příkazů do modulů nastaven do režimu synchronní paralelní FIFO 245 sběrnice (Obrázek 7). V tomto režimu jsou komunikačními piny obvodu FT232H tyto signály [7]:

- D0...D7 – obousměrná datová sběrnice
- RXF# – připravenost dat ke čtení z FT232H
- OE# – datová sběrnice nastavena jako výstup
- RD# – potvrzení přečtení dat z FT232H s každým hodinovým cyklem (dokud jsou RXF# i RD# v 0)
- TXE# – povolení zápisu dat do FT232H
- WR# – zápis dat do FT232H s každým hodinovým cyklem (dokud jsou TXE# i WR# v 0)
- CLK – hodinový signál generovaný v FT232H, kterým je synchronizována komunikace (60 MHz)

Obvod FT232H obsahuje dva buffery – jeden vysílací a jeden přijímací, každý o kapacitě 1 kB. Tyto buffery slouží vnitřní struktuře čipu k vyrovnání toku dat [7].

V dané konfiguraci je možné použít dva další signály, které jsou součástí sběrnice mezi agregátory. Jeden z nich slouží jako hlavní reset celého systému, druhý je v danou chvíli nepoužit.



Obrázek 7 – Časový průběh komunikačních signálů na sběrnici obvodu FT232H [7]

3 Návrh software

Stěžejní částí práce je návrh softwaru pro komunikaci s hardwarovým zařízením, které bylo popsáno v předchozí kapitole. Software by měl obsahovat řídicí a vizualizační časti, které budou sloužit jako uživatelské rozhraní. Předpokládaným uživatelem celého systému by měl být odborník z oblasti částicové fyziky seznámený s režimy, ve kterých může digitalizační systém pracovat, a s tím, jak správně takový systém nakonfigurovat. Potřeby uživatele nejsou v dané chvíli kompletně definované a tak návrh vychází z dílčích požadavků známých v době návrhu s ohledem na možnost dalšího vývoje softwaru.

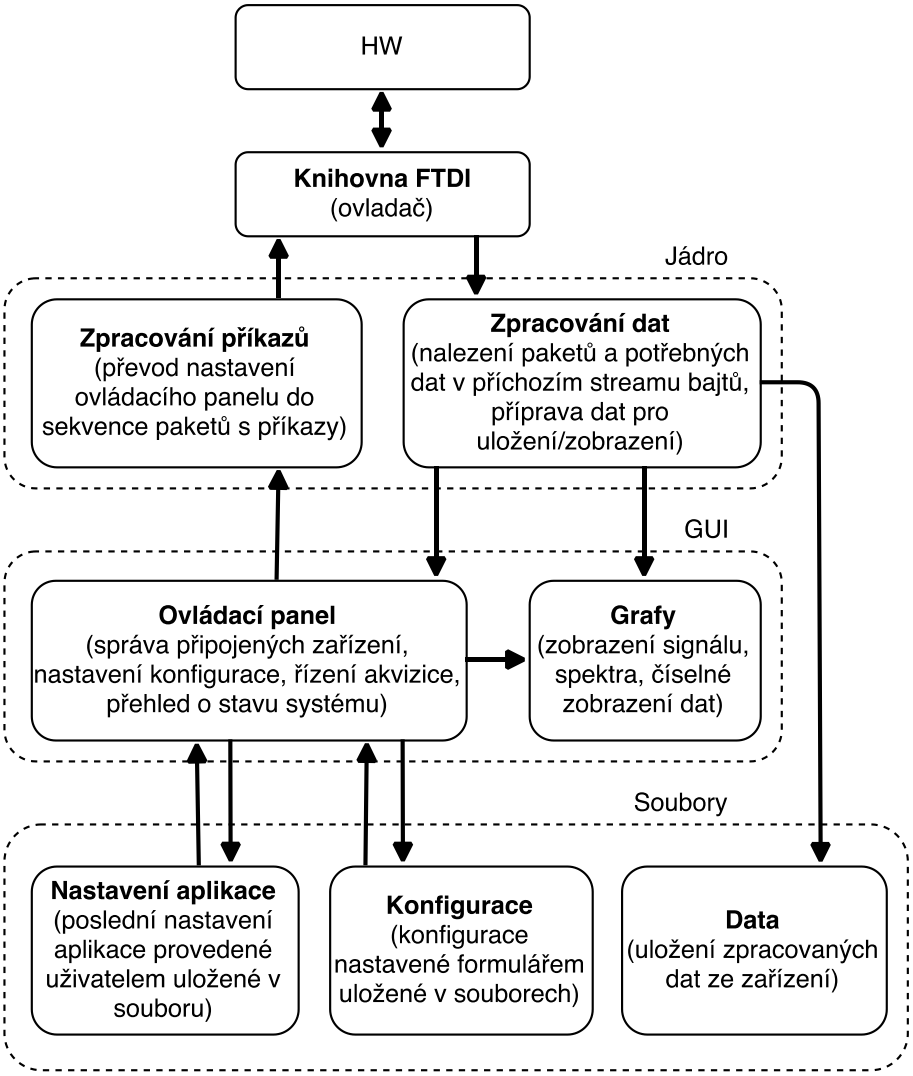
3.1 Rozbor požadavků

Na začátku vývoje softwaru bylo stanoveno několik základních požadavků, které byly v průběhu práce na vývoji softwaru dále rozvíjeny během konzultací s budoucími uživateli softwaru. Byly to hlavně následující:

- Implementace komunikačního protokolu hardwarové části systému pro správnou komunikaci s ním. Možnost budoucí změny v komunikačním protokolu.
- Schopnost rychlého přijetí dat ze systému tak, aby prováděným algoritmem nebyl zpomalován jejich tok, a uložení na disk. Při použití čipu FT232H může být dosahováno reálné přenosové rychlosti až 40 MB/s [7].
- Grafické rozhraní, kterým by měly být pohodlně a intuitivně prováděny úkony jako konfigurace, řízení akvizice a sledování aktivity jednoho modulu, stejně tak jako vybrané skupiny modulů, popřípadě celého digitalizačního systému.
- Rozšíření grafického rozhraní o vizualizaci zpracovávaných dat z uživatelem zvolených modulů. Vizualizací se rozumí vynášení digitalizovaného průběhu do grafu, vytváření a zobrazování spekter (např. amplitudových). Pro další vývoj softwaru se počítá i s budoucím požadavkem dalších vizualizací dat v jiných typech grafů.
- Implementace algoritmu vypočítávajícího numerický integrál z digitalizovaného průběhu, následné ukládání jak celého zachyceného průběhu, tak pouze vypočítaného integrálu pro potřeby energetických spekter.
- Přenositelnost aplikace mezi operačními systémy (Windows, Linux).

3.2 Plán realizace

Podle zadaných požadavků byl vytvořen plán, jaké části softwaru vytvořit. Součástí plánu je blokové schéma (Obrázek 8), ve kterém jsou zobrazeny jednotlivé části softwaru. Tyto části budou dále popsány. Jako programovací jazyk bude zvolen jazyk Java [10], který přináší celou řadu výhod při programování [11]. Tato volba zajišťuje přenositelnost aplikace mezi operačními systémy. Pro vytvoření grafického uživatelského rozhraní bude zvolena softwarová platforma JavaFX jako moderní přístup k návrhu GUI v Javě.



Obrázek 8 - Blokové schéma vytvářené aplikace

První vyvíjenou částí aplikace bude komunikační rozhraní mezi univerzální softwarovou částí a knihovnou zprostředkovávající komunikaci s daným zařízením (FTDI čipem). Připojení této knihovny pro přístup z Javy je možné pomocí knihovny Java Native Access [12]. K tomu je možné využít Java knihovnu, která již toto propojení provádí [13]. Zároveň bude nutné vytvořit součást, pomocí které bude zajištěna správa připojených hardwarových zařízení, poskytnutí funkcionalit podle typu připojeného zařízení a komunikace podle komunikačního protokolu daného zařízení.

S komunikačním rozhraním daného zařízení musí komunikovat části zajišťující zpracování dat a zpracování předávaných příkazů. Část pro zpracování příkazů bude převádět formulář grafického rozhraní do sekvence bajtů určených k odeslání do zařízení. Zpracování bude rozděleno na zachycení a distribuci. Každé připojené zařízení bude obsluhováno samostatným vláknem, které bude z přijímaných dat vybírat pakety a ty ukládat do zásobníku. Distribučním vláknem bude vykonáváno ukládání paketů do souboru na disk, zobrazení podle požadavků uživatele, případně obojí zároveň. Při ukládání zpracovaných dat (např. vypočtený integrál) budou nová data umístěna v datové části paketu.

Grafické rozhraní bude obsahovat ovládací část a vizualizační část. V ovládací části bude prováděno připojování dostupných zařízení a u každého z nich bude umožněno označení modulů pro další práci. Pro konfiguraci, řízení akvizice a sledování aktivity systému budou vytvořeny samostatné karty ovládacího panelu. Karta konfigurace bude obsahovat prvky umožňující výběr z několika možností nastavení, vstupní pole pro zadání číselných hodnot a tlačítka, která budou sloužit k nahrání konfigurace ze souboru, jejímu uložení do souboru a odeslání do zařízení. Na kartě řízení akvizice bude možno spustit a zastavit akvizici včetně nastavení odloženého spuštění a zastavení. Karta sledování aktivity systému bude zajišťovat náhled na stav a chování celého systému. To znamená, že uživatel zde uvidí informace o jednotlivých připojených modulech, přenosové rychlosti, množství přijatých dat a další.

Vizualizaci dat bude vhodné umístit do samostatných oken. Uživatel bude moci zvolit reprezentaci dat. Na výběr bude mít graf typu XY s vyneseným digitalizovaným signálem, barový graf zobrazující amplitudové spektrum nebo tabulku ukazující zachycené číselné hodnoty (např. v režimu vzorku). Společné vlastnosti vizualizačních oken budou umístěny v abstraktní třídě, od které je budou jednotlivá okna dědit. Při dalším vývoji tak bude jednoduché přidávat další možnosti zobrazení.

Aplikace bude po svém ukončení ukládat svoje poslední nastavení a při dalším spuštění bude toto nastavení opět načítat. To je možné zajistit uložením celého objektu do souboru.

Uživateli musí být umožněno uložení konfigurace odesílané do zařízení. Tím bude usnadněna práce při přípravě konfigurace ve formuláři na příslušné kartě. Stav formuláře bude vhodné převést do textového řetězce, který se uloží do souboru ve formě čitelné pro uživatele. Tím se vytvoří možnost nahlédnutí nebo editace v textovém editoru. Tento řetězec pak musí být možné opět načíst ze souboru a nastavit podle něj formulář do původní podoby. K tomu bude využit objekt typu JSON.

3.3 Možnosti použití stávajících softwarových nástrojů

Před započetím vývoje nového softwaru pro mnohakanálový detektor pro detekci reaktorových anti-neutrin byla vzata do úvahy možnost použití či úpravy stávajícího softwarového řešení. Mnohakanálové zpracování dat je úkol, který je řešen nejrůznějšími společnostmi (např. National Instruments, Caen). Existuje několik komerčních produktů, v rámci kterých si může uživatel pořídit kompletní systém včetně hardwaru a k němu na míru dodaného softwaru. Jedná se většinou o osciloskopy nebo digitizéry komunikující s uživatelským počítačem. Přestože existují zařízení založená na univerzálním programovatelném hardwaru a přizpůsobitelné softwarové podpoře, je jejich přenositelnost vzhledem k atypickým a měnícím se požadavkům jednotlivých experimentů značně omezená.

Tato práce navazuje na hardwarový systém, který byl již vytvořen v Ústavu technické a experimentální fyziky ČVUT. Se systémem musí být komunikováno podle jeho komunikačního protokolu. Pro ovládání bylo tedy potřeba vytvořit na míru softwarové rozhraní k hardwarovému zařízení. V tom případě je vhodnější k ovládací části připojit část pro ukládání příchozích dat, což je výhodnější než data zachytávat nějakou jinou aplikací. Zároveň zadavatel stojí o to, aby měl aktuální přehled o datech, stavu systému a jeho nastavení na jednom místě.

Nově vyvíjený software by měl být navržen tak, aby byla možná a snadná případná budoucí změna formátu ukládání dat. Pak může být uvažováno o dalším zpracování dat jiným

softwarovým nástrojem, který zjednoduší zvláště analýzu naměřených dat. Tím může být například Matlab nebo *ROOT*, který je často využívaným nástrojem pro analýzu dat z nejrůznějších experimentů prováděných v oblasti částicové fyziky. Je tak velmi pravděpodobné, že bude v budoucnu požadován export dat právě pro nástroj ROOT.

ROOTem rozumíme nástroj pro zpracování dat ve formě široké sady funkcí, který byl vyvinut v CERN a je využíván vědci, především fyziky, k analýze dat a provádění simulací. Jde vlastně o framework, který může být využit k ukládání dat v komprimovaném binárním formátu, k přístupu k velkému množství dat v rozsáhlých stromových strukturách ROOT souborů, k rychlému provádění matematických a statistických výpočtů a ke snadnému publikování výsledků v nejrůznějších grafech a histogramech. Také je možné využít integrovaný interpreter jazyka C++ pro spouštění vlastních skriptů, zadávat příkazy ke zpracování dat přímo z příkazové řádky tohoto interpreteru nebo napsat vlastní kód v C++, který má být ke zpracování dat spuštěn [14].

4 Realizace softwarové části

V této kapitole bude popsána realizace samotného softwaru včetně souvisejících záležitostí, jako je komunikační protokol s hardwarovou částí systému, způsob zpracování dat apod. Software byl vytvořen na základě již rozebraných požadavků a provedených návrhů.

4.1 Komunikace s elektronickou částí

Komunikace s elektronickou částí probíhá přes USB 2.0 rozhraní. Jak již bylo zmíněno v popisu elektronické části, součástí hardwaru je komunikační modul s čipem FT232H. Čip je při přenosu dat nastaven na komunikaci v módu paralelní synchronní FIFO 245 sběrnice, ve kterém lze dosáhnout přenosové rychlosti až 40 MB/s [7].

Po tomto komunikačním rozhraní jsou směrem do hardwaru odesílány konfigurace pro jednotlivé moduly a řídicí příkazy pro zpracování analogových signálů a jim odpovídajících digitalizovaných dat. Také je přistupováno k několika pinům samotného FTDI čipu kvůli resetu zařízení.

Do připojeného počítače jsou posílána data vzorkovaných analogových signálů. V tomto směru byla snaha o co možná největší propustnost dat, bez jejichž předání není možný další záznam.

4.1.1 Použité knihovny

Pro komunikaci s čipem FTDI dodává společnost Future Technology Devices International Ltd. dynamickou knihovnu funkcí (ftd2xx.dll / libftd2xx.so / libftd2xx.dylib) a příručku pro programátory [15]. Přestože je tato knihovna dynamická, a tudíž není k výslednému spustitelnému souboru přidávána, je potřeba, aby byla distribuována společně s celou aplikací. Systém dynamických sdílených knihoven je dobře propracován pouze na unixových systémech. Na Windows je sdílení knihoven omezeno [16]. Navíc je lepší, pokud záležitost s knihovnami nemusí být řešena uživatelem.

Aby bylo možné přistupovat k funkcím z knihovny a tak komunikovat s čipem z prostředí aplikace napsané v Javě, bylo potřeba využít knihovnu Java Native Access (JNA). Propojení knihovny od FTDI a JNA je vyřešeno knihovnou JavaFTD2XX, která tak umožňuje přístup k FTDI zařízení z Javy [13]. Použitá Java knihovna musela být nepatrně upravena, jelikož se při jejím začlenění do vyvíjeného projektu vyskytly problémy se zjišťováním připojených FTDI čipů a čtením dat nich. Po analýze zdrojového kódu knihovny byl problém odstraněn zavedením kontroly otevřených FTDI zařízení podle jejich sériových čísel. Nalezenou chybou bylo vytváření nových Java objektů pro zařízení, se kterými již bylo otevřeno spojení.

Z knihovny FTDI byly využity tyto funkce:

- FT_CreateDeviceInfoList
- FT_GetDeviceInfoDetail
- FT_OpenEx, FT_Close
- FT_SetLatencyTimer
- FT_SetUSBParameters
- FT_SetFlowControl
- FT_SetTimeouts
- FT_Purge
- FT_SetBitMode
- FT_Write, FT_Read

Podrobný popis těchto funkcí a jejich parametrů je dostupný v FTDI příručce pro programátory [15].

4.1.2 Navázání spojení a inicializace zařízení

Při požadavku na otevření spojení s FTDI čipem je potřeba po zavolání funkce *open()* nejprve nastavit parametry komunikace. K tomu slouží některé FTDI funkce, které již byly zmíněny. Volba parametrů vychází z doporučení výrobce čipu a jsou voleny s ohledem na maximální efektivitu datového přenosu. Nastavení parametrů uvádí Tabulka 1.

Tabulka 1 - Parametry komunikace nastavené v obvodu FTDI

Parametr	Hodnota
Latency timer	3 ms
USB parameters (in, out)	65536, 65536
Flow control	FLOW_RTS_CTS (0x0100)
Timeouts (read, write)	100 ms, 100 ms
Bit mode	BITMODE_SYNC_FIFO (0x0040)

Jelikož je při připojování z důvodu resetu hardwaru nutné změnit stav pinů FTDI čipu, které jsou přímo spojené se zařízením. Nejdříve je nastaven režim BITMODE_CBUS_BITBANG, výstupní pin ACBUS9 je nastaven do log. 1 a pin ACBUS8 je nastaven do log. 0. Poté může být s prodlevou několika milisekund nastaven režim na BITMODE_RESET a následně na BITMODE_SYNC_FIFO. Proces nastavování parametrů komunikace je zakončen vyprázdněním vstupního i výstupního bufferu.

Dále je provedena inicializace hardwaru. Ta spočívá v přidělení adres jednotlivým modulům. Pro adresaci modulů je vyhrazen speciální paket (Tabulka 2). Programem by měly

být odeslány adresační pakety pro celý adresní prostor. Z důvodu variability systému (možnost přidání dalších kanálů) není počet modulů nastaven pevnou hodnotou. Následně se provede ověření odezvy softwarovým příkazem, který zajistí poslání datového paketu do počítače. Úloha, kterou je prováděno připojení zařízení, čeká na příchozí pakety. Pokud nějaký modul neodpoví, je adresa označena jako neplatná a další komunikace na této adrese neprobíhá. Důvodů pro absenci odpovědi může být celá řada, zvláště pak hardwarového charakteru.

Tabulka 2 - Adresační paket (3 bajty)

0	170	adresa

Pokud bylo spojení s FTDI čipem otevřeno, nastaveno a zařízení je inicializováno, jsou spouštěna další vlákna aplikace, která souvisí se čtením, ukládáním a vizualizací dat.

4.1.3 Formát konfiguračních paketů

Odesílanými konfiguračními pakety jsou nastavovány digitalizační moduly. Každý odeslaný paket má 3 bajty (Tabulka 3). Je odeslána adresa modulu, ke kterému paket směřuje, kód příkazu a data příkazu (Tabulka 4).

Tabulka 3 - Paket s příkazem (3 bajty)

adresa	příkaz	data

Tabulka 4 - Popis jednotlivých příkazů odesílaných do zařízení

Příkaz	Data bity	Označení	Popis
140	[6:4]	mode	Nastavení módu digitalizačního modulu.
	[3:2]	start_mode	Nastavení spouštěcí podmínky akvizice.
	[1:0]	end_mode	Nastavení ukončovací podmínky akvizice.
127	-	passive_byte	Po odeslání tohoto příkazu 2x za sebou je v FPGA očekáván jako další bajt příkaz.
142	[7]	gothrough	Nastavení dat z A/D převodníku v přímé fomě.
	[6]	derandom	Změna dat formou derandomizace.
	[5]	dealter	Změna dat formou dealterizace.
	[4]	polarity	Nastavení očekávané polarity pulsu.
	[3:2]	extrigcfg	Nastavení externího spouštění akvizice.
	[1:0]	gatecfg	Nastavení blokace externího spouštění.
143	[7]	addheader	Požadavek hlavičky v příchozím paketu.
	[6]	addtime	Požadavek časové značky v příchozím paketu.
	[5]	addchecksum	Požadavek kontrolního součtu v příchozím paketu.
	[4:3]	strescfg	Nastavení resetování časové značky.
	[2:1]	discardcfg	Nastavení externího resetu přenosu.
	[0]	bus8	Požadavek na použití 8bitové sběrnice.
128	[7:0]	LLD	Spodní komparační úroveň (LLD [15:8]).
129	[7:0]	LLD	Spodní komparační úroveň (LLD [7:0]).
130	[7:0]	ULD	Horní komparační úroveň (ULD [15:8]).
131	[7:0]	ULD	Horní komparační úroveň (ULD [7:0]).
132	[7:0]	hyst	Nastavení hystereze komparátoru.
133	[7:0]	clkmux	Nastavení dělení výchozího hodinového signálu.
134	[7:0]	pretrig	Počet vzorků před příchodem spouštěcího signálu (pretrig [15:8]).

135	[7:0]	pretrig	Počet vzorků před příchodem spouštěcího signálu (pretrig [7:0]).
136	[7:0]	count	Počet uložených vzorků od spouštěcího signálu (count [15:8]).
137	[7:0]	count	Počet uložených vzorků od spouštěcího signálu (count [7:0]).
138	[7:0]	posttrig	Počet uložených vzorků po hlavní fázi (posttrig [15:8]).
139	[7:0]	posttrig	Počet uložených vzorků po hlavní fázi (posttrig [7:0]).
148	[1:0]	maskdata	Nastavení požadavku na příjem dat o konfiguraci HW.
141	-	trigger_command	Softwarové spuštění akvizice.
144	-	inhibit_clear	Povolení odesílání dat z HW.
191	-	inhibit_set	Zákaz odesílání dat z HW.
145	-	reset_command	Reset zařízení.
146	-	timestamp_reset	Reset časové značky.
147	-	discard_command	Příkaz pro zahození dat v bufferu.

4.1.4 Formát přijímaných paketů

Od zařízení jsou přijímány pakety obsahující data a k nim potřebné informace. Tabulkou 5 je znázorněno, z jakých částí je paket složen. Tabulka 6 pak dále popisuje jednotlivé části tohoto paketu.

Tabulka 5 - Přijímaný paket (jednotlivé bajty)

ADDRESS [7:0]	HEADER [31:24]	HEADER [23:16]	HEADER [15:8]	HEADER [7:0]
D_LENGTH [15:8]	D_LENGTH [7:0]	DATA_0 [15:8]	DATA_0 [7:0]	…
DATA_N [15:8]	DATA_N [7:0]	TIME [47:40]	TIME [39:32]	TIME [31:24]
TIME [23:16]	TIME [15:8]	TIME [7:0]	CHECKSUM [15:8]	CHECKSUM [7:0]

Tabulka 6 - Popis jednotlivých částí paketu

Část paketu	Počet bitů	Význam, hodnoty
ADDRESS	8	Adresa modulu
HEADER	32	1111 1111 1111 0000 000*a bbbb* 0000 1111
DATA_LENGTH	16	Počet 16 bitových slov (N)
DATA_0	16	-
…	16	-
DATA_N	16	-
TIME	48	Časová značka (start ukládání vzorků)
CHECKSUM	16	Kontrolní součet (HEADER až TIME)

V části HEADER jsou bity označené *a* a *b*. Pokud je v režimu vzorku nastaven bit *a*, nastal tzv. pileup. To znamená, že došlo k více událostem ve stejný čas a získaná data jsou výsledkem složeného signálu. Výsledná hodnota je tak špatně, protože neodráží správně ani jednu, ani druhou událost. Tato funkcionalita zatím není hardwarem podporována a tento bit tedy zatím vždy zůstává nastaven na hodnotu 0. V osciloskopickém režimu je tímto nastaveným bitem signalizováno, že došlo k přetečení paměti v FPGA pro ukládání vzorků. Uživatel by pak měl být při zpracování dat informován, že data mohou být nekompletní a nemusejí vypovídat o skutečnosti.

Číslo složené z bitů označených *b* má pro všechny módy stejný význam. Jde o číslování paketů, jak jsou odesílány za sebou. Po resetu zařízení je prvnímu paketu přiděleno číslo 0. Číslování jde vždy do hodnoty 15 a pak znovu od 0. Tímto číslováním může být ověřeno, že žádný paket nebyl ztracen při přenosu.

Část TIME je 48bitové číslo, které je časovou značkou. Hodnota je stavem čítače, který je vynulován zapnutím zařízení nebo odesláním k tomu určenému příkazu pro reset čítače (timestamp reset). V režimu vzorku je tímto číslem udáván časový okamžik zaregistrování události (podle nastavené spouštěcí podmínky). V osciloskopickém režimu jde o okamžik spuštění akvizice (bez pretriggeru), tedy registrace externího triggeru nebo překonání prahové hodnoty aktuálním vzorkem.

Každá část DATA_x odpovídá jednomu 16bitovému číslu, které je jedním vzorkem. Zařízení také umožňuje vynechání jednotlivých částí paketu a posílání například jen částí ADDRESS, DATA_LENGTH a DATA v nejvíce zredukované podobě. Vytvořeným softwarem není podporován příjem paketů v této redukované formě vzhledem k nemožnosti detekce chyb.

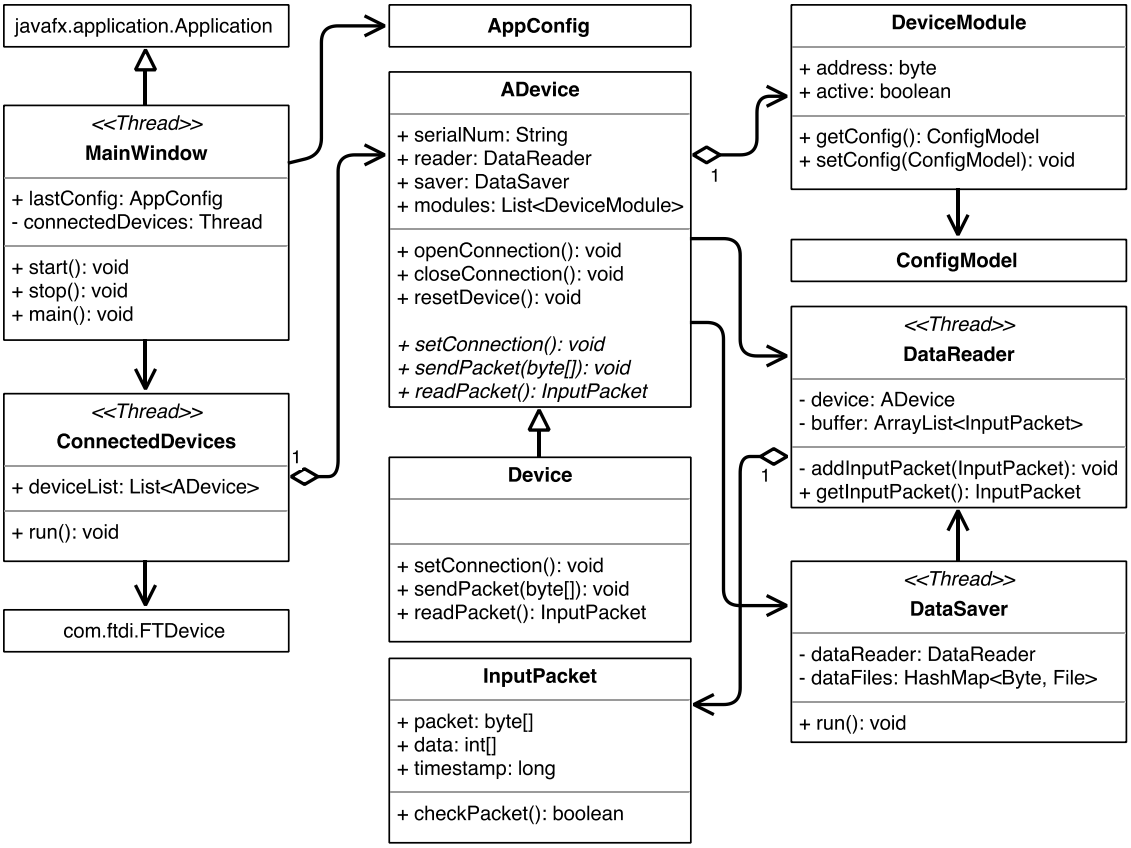
Na konci paketu je umístěn kontrolní součet, který odpovídá součtu 16bitových slov paketu s přetečením v maximální hodnotě dané 16 bity.

4.2 Struktura aplikace

Základní struktura aplikace je zakreslena diagramem tříd (Obrázek 9), které jsou popisovány v této kapitole. Po spuštění aplikace zavoláním metody *main()* ve třídě *MainWindow* je touto metodou nejdříve ověřena dostupnost uloženého objektu (*AppConfig*) s nastavením aplikace v souboru ve stejném adresáři, v kterém byla aplikace spuštěna. Nastavení aplikace je načteno, nebo je vytvořeno nové výchozí nastavení. Poté je zavolána metoda, která již spouští grafické rozhraní JavaFX. Jelikož třída *MainWindow* rozšiřuje *javafx.application.Application*, musí implementovat metodu *start()*, která je volána při startu grafiky. Touto metodou je provedeno načtení fxml dokumentu hlavního okna, jeho kontroléru (*FXMLDocumentController*) a také je na tomto místě spuštěno vlákno třídy *ConnectedDevices*.

Úlohou třídy ConnectedDevices je dotazovat se knihovny JavaFTD2XX na seznam připojených FTDI zařízení, nová zařízení držet v seznamu a tento seznam objektů typu *Device* poskytovat grafickému rozhraní. Rovněž je zajišťováno odstraňování objektů z paměti v případě, že je FTDI zařízení odpojeno. Pokud je zavřeno okno aplikace, jsou řádně ukončena otevřená spojení a zařízení jsou uvolněna.

Z grafického rozhraní mohou být spouštěny akce nad připojenými zařízeními. Takovou akcí je v první řadě otevření spojení se zařízením, nastavení komunikace a přečtení informací o otevřeném zařízení, jeho modulech a hardwarových parametrech všech dostupných modulů. V každém objektu Device je držen seznam jeho dostupných modulů jako objektů typu *DeviceModule*. Dále je odtud dostupné vlákno třídy *DataReader*, které je spouštěno s otevřením spojení se zařízením, a stejně tak i vlákno třídy *Saver*. Třída Device dědí od třídy *ADevice* vlastnosti, které mohou být s velkou pravděpodobností společné s případným jiným zařízením, které by komunikovalo podle odlišného komunikačního protokolu. Tímto je struktura připravena na rozšíření o další typy zařízení.



Obrázek 9 - Struktura části aplikace popsaná diagramem tříd

Ve třídě DataReader je zajišťováno čtení příchozích dat od zařízení, ve kterých je hledán paket podle definice komunikačního protokolu. Při jeho nalezení je jako objekt typu *InputPacket* zařazen do FIFO zásobníku paketů. Zároveň jsou v této třídě drženy informace o příchozí komunikaci, jako přenosová rychlost, doba od poslední odezvy, množství přijatých paketů atd. Tyto informace jsou jiným vláknem aktualizovány v grafickém rozhraní.

Ve třídě Saver je znám odkaz na třídu DataReader, takže mohou být vyzvedávány pakety z FIFO zásobníku a data mohou být ukládána do souboru. Zároveň je hlídáno množství dat uložených v souboru a podle nastavení aplikace uživatelem jsou vytvářeny nové soubory, aby se předešlo vzniku souborů nadrozměrných velikostí.

Hlavní okno, které lze také nazvat ovládacím panelem, je řízeno kontrolérem, kterým je zároveň držen seznam dalších otevřených oken, pro které je hlavní okno rodičem. V případě zavření hlavního okna jsou okna ze seznamu řádně ukončena. Kontrolér hlavního okna obsahuje metody k obnovování jeho prvků, jako jsou tabulky a indikátory, a metody, které jsou propojeny s ovládacími prvky okna.

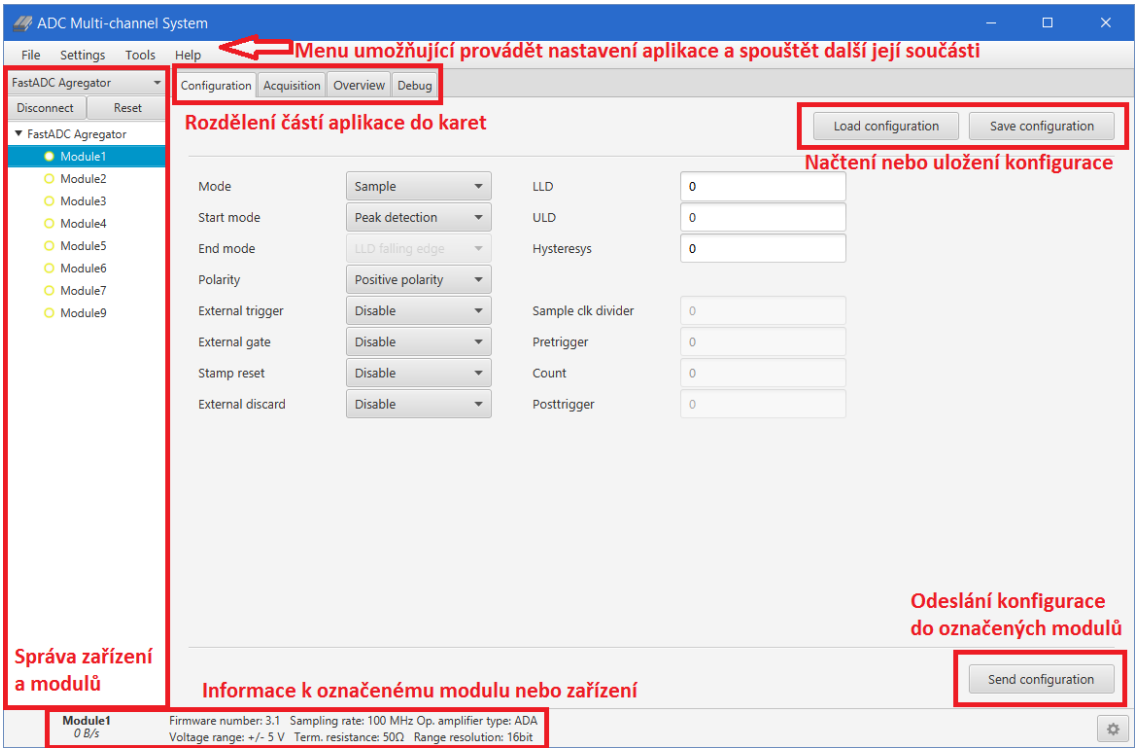
Dalšími grafickými prvky jsou okna s grafy a okna s nastavením. Kontroléry oken s grafy mají společného předka – třídu, kterou je zajištěno, že pro všechna okna bude dostupný datový zdroj pro jejich grafy. O aktualizaci každého grafu je postaráno vláknem, které pouze provádí vytažení dat z naposledy přijatého paketu a předání těchto dat kontroléru okna grafu. Toto se provádí v intervalu, jehož výchozí nastavení je 1 s, aby měl uživatel přehled o činnosti systému. Pomocí oken s nastavením je možné měnit parametry celé aplikace (např. adresování modulů, ukládání dat).

4.3 Popis grafického rozhraní

Grafické rozhraní v JavaFX bylo navrhováno s pomocí programu SceneBuilder. Návrh byl rozdělen do dvou etap. Nejprve byl vytvořen ovládací panel umožňující ovládání aplikace i samotného hardwaru, včetně oken nastavení. V druhé etapě byla vytvářena okna obsahující grafy a tabulky pro vizualizaci dat.

4.3.1 Ovládací panel

Po spuštění aplikace je otevřen ovládací panel (Obrázek 10). Ten je rozdělen na část správy připojených zařízení a modulů, konfigurační část, akviziční část a část poskytující přehled o stavu celého systému. U horního okraje okna je menu, které umožňuje nastavování aplikace a spouštění dalších součástí aplikace. U spodního okraje okna je umístěna lišta s informacemi o označeném zařízení nebo modulu.



Obrázek 10 - Ovládací panel po otevření aplikace a připojení zařízení

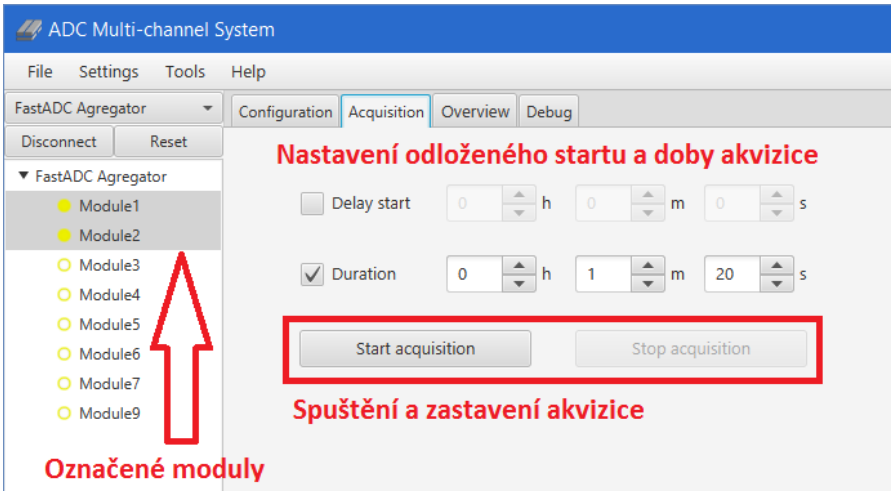
V části správy připojených zařízení je vybráno zařízení, jehož připojení je požadováno. Tlačítkem pro připojení je provedeno otevření spojení se zařízením a také jeho inicializace. Po připojení a inicializaci je pod každým zařízením vidět seznam jeho modulů. V této části aplikace spravující připojená zařízení je také umístěno tlačítko pro odpojení zařízení a pro reset. V obsluze tlačítka reset je proveden hardwarový reset zařízení a nová adresace modulů. U každého modulu je signalizován stav barevným indikátorem. Žlutý nevyplněný terčík signalizuje, že modul zatím nebyl od otevření aplikace nakonfigurován. Pokud je terčík vyplněný žlutě, modul již nakonfigurován byl. V průběhu akvizice je pak terčík zelený, pokud dochází k příjmu dat od zařízení, nebo červený, pokud žádný příjem neprobíhá.

Uživateli je umožněno přepínání mezi kartami *Configuration*, *Acquisition*, *Overview* a *Debug*. Poslední zmíněná karta je pro účely ladění a dalšího vývoje aplikace. Na této kartě je umístěno textové pole obsahující výpisy z aplikace.

Karta Configuration je určena pro vytvoření konfigurace jednotlivých modulů pomocí grafického formuláře. Každý modul může být konfigurován odlišně dle aktuálních požadavků experimentu. Při označení více modulů je nastavována konfigurace pro všechny tyto moduly. Stav formuláře je aktualizován tak, aby po kliknutí na jiný modul v seznamu formulář zobrazoval konfiguraci použitou pro daný modul. V případě, že jsou označeny moduly, z nichž každý má v některých položkách jinou konfiguraci, je tato skutečnost uživateli signalizována popiskem „(different)“.

Jednotlivými částmi formuláře je nastavován režim, ve kterém má modul pracovat, podmínky začátku a konce zaznamenávání události, polarity řídicích signálů apod. Dále jsou nastavovány číselné hodnoty pro spodní a horní komparační úroveň, jejich hysterezi, dělení výchozího hodinového signálu a údaje související s délkou zaznamenaného průběhu (počet vzorků, počet vzorků před spuštěním akvizice a po uložení hlavního nastaveného počtu vzorků). Ve spodní části formuláře je umístěno tlačítko pro odeslání konfigurace do označených modulů.

Na kartě Acquisition (Obrázek 11) jsou všechny ovládací prvky neaktivní, dokud nebyla do zvolených modulů odeslána konfigurace. Po aktivování ovládacích prvků je zde možno nastavit odložený start a dobu trvání akvizice. Po stisku tlačítka *Start acquisition* je vybraným modulům poslán příkaz *Inhibit clear*, kterým je povoleno zaznamenávání událostí hardwarem a jejich posílání do počítače. Obdobně po stisku tlačítka *Stop acquisition* je zaznamenávání zakázáno příkazem *Inhibit set* (popis v kapitole 4.1.3). Pokud byl nastaven odložený start nebo doba trvání akvizice, jsou příkazy pro *Inhibit clear* a *Inhibit set* odesílány automaticky po vypršení vnitřního časovače.



Obrázek 11 - Ovládací panel před spuštěním akvizice

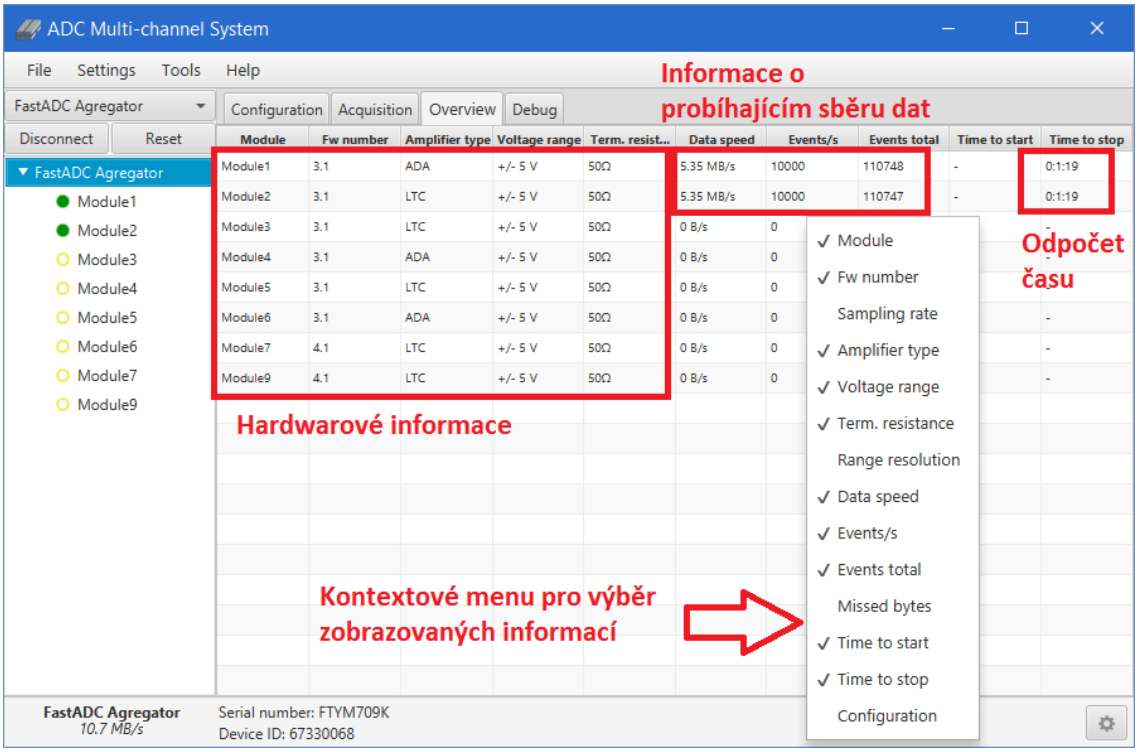
Na kartě Overview (Obrázek 12) je možno sledovat informace o jednotlivých modulech. Příkladem jsou následující:

- Aktuálně nahraná verze firmwaru
- Frekvence vzorkování
- Typ použitého operačního zesilovače
- Napěťový vstupní rozsah
- Zakončovací impedance
- Rozlišení v bitech

Další informace zobrazující se v tabulce na kartě Overview se týkají probíhající akvizice a sběru dat. Jedná se např. o tyto:

- Rychlost datového přenosu z modulu
- Počet detekovaných událostí za sekundu
- Počet událostí od spuštění akvizice
- Počet ztracených bajtů při komunikaci (např. vadné pakety)
- Čas do začátku akvizice
- Čas do konce akvizice
- Doba probíhající akvizice
- Konfigurace odeslaná do modulu

O tom, jaké informace mají být zobrazovány, může uživatel rozhodnout výběrem příslušných položek v kontextovém menu, které je otevřeno stisku pravý tlačítkem myši na tabulku.



Obrázek 12 - Ovládací panel a karta Overview

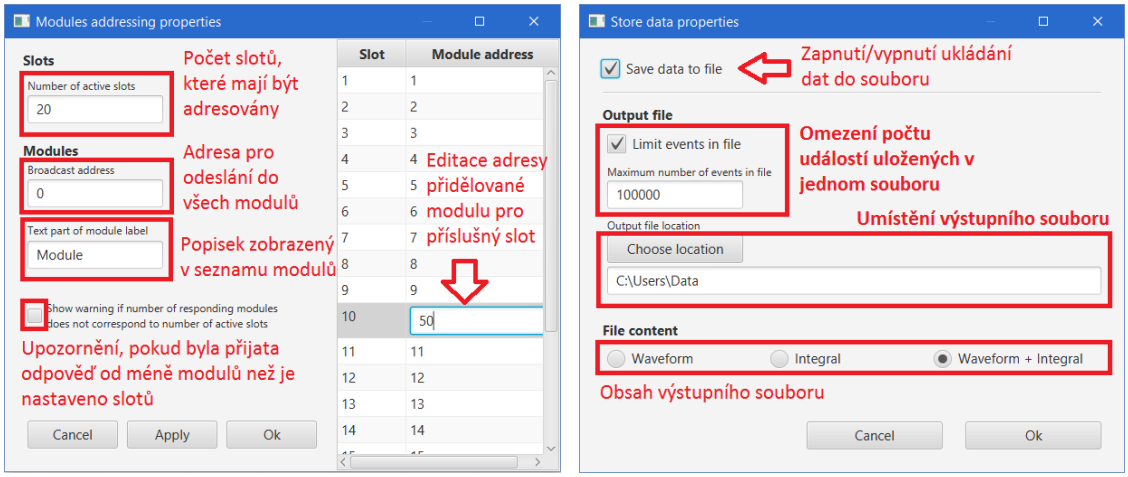
4.3.2 Nastavení aplikace

Nastavení aplikace je po jejím ukončení ukládáno do souboru, aby mohlo být opět načteno při příštím spuštění. Ukládají se volby provedené v ovládacím panelu i dalších oknech určených k nastavování aplikace. U ovládacího panelu jde např. o zapamatování zvolených sloupců určených k zobrazení v tabulce na kartě Overview.

Další okna je možné otevřít z menu ovládacího panelu. V dosavadní verzi aplikace jde o okno pro nastavení adresace modulů a okno pro nastavení exportu dat.

V okně pro nastavení adresace modulů (Obrázek 13, vlevo) je možno nastavit počet slotů, které budou adresovány, společnou adresu pro odeslání do všech modulů a popisek zobrazovaný v seznamu modulů v ovládacím panelu. Adresy modulů pod jednotlivými sloty mohou být změněny v tabulce, jejíž pravý sloupec je možno editovat. Také je možno povolit výpis upozornění, že nebyla přijata odpověď od takového počtu modulů, od jakého byla očekávána.

V okně pro nastavení exportu dat (Obrázek 13, vpravo) je možné vypnout ukládání dat do souboru, omezit počet událostí uložených v jednom souboru, vybrat umístění souboru a nastavit typ ukládání dat.

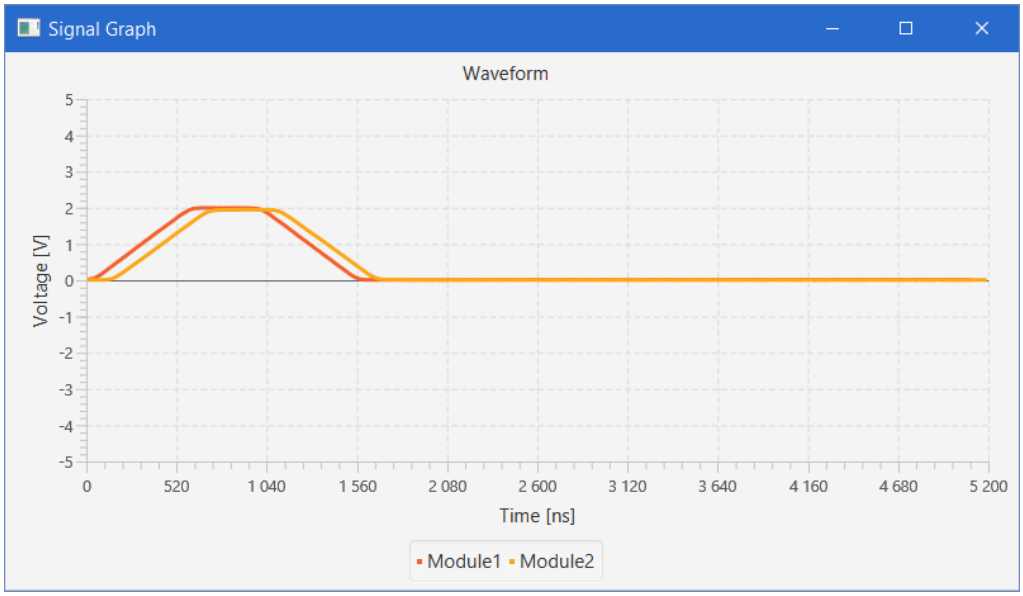


Obrázek 13 – Okno umožňující nastavení adresace modulů (vlevo) a okno pro nastavení možností exportu dat (vpravo)

4.3.3 Vizualizace dat

Vizualizace dat je rozdělena do tří typů grafů podle toho, jaký režim modulu byl nastaven (osciloskopický nebo režimu vzorku). Spuštění okna s grafem pro vybrané moduly je možné z kontextové nabídky po kliknutí pravým tlačítkem myši na seznam modulů v ovládacím panelu. Pokud je označeno více modulů, jsou zobrazována data ze všech těchto modulů v jednom grafu. Grafy jsou určeny pro průběžné a orientační zobrazování dat přicházejících v reálném čase z digitalizačního systému do počítače. Z tohoto důvodu jsou v určitém časovém intervalu vybírány naposledy přijaté pakety z fronty paketů a data z nich jsou zobrazena v grafu. Výchozí interval je nastaven na 1 s.

Pro osciloskopický režim je připraven signálový graf (Signal Graph) zobrazující průběh navzorkovaného signálu (Obrázek 14). Tento graf je přizpůsobován konfiguraci odeslané do modulů (délka záznamu) a také hardwarovým parametrům vyčteným z modulů (osy grafu).



Obrázek 14 – Graf s navzorkovaným signálem

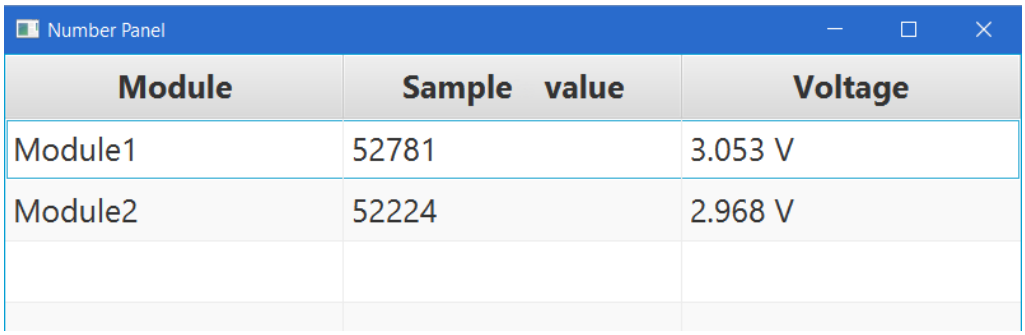
V případě vynášení průběhu do grafu je potřeba provádět redukci v počtu vzorků záznamu. Pro zobrazení na běžném typu monitorů je zbytečné, aby bylo grafu předáváno větší množství dat, než může být reálně zobrazeno při daném rozlišení. Navíc nižší rozlišení snižuje výpočetní nároky na zobrazení. Například 10000 vzorků je možné převést na počet, který se bude blížit rozlišení grafu, aniž by zároveň došlo ke ztrátě zobrazované informace. K tomuto zjednodušování byl použit *Ramer–Douglas–Peuckerův algoritmus*, jehož úkolem je nalézt pro zadanou křivku podobný průběh s menším počtem bodů. První a poslední bod původní řady je zafixován a oba body jsou propojeny přímkou. Je nalezen bod od této přímky nejvzdálenější, jehož vzdálenost od přímky je zároveň delší než zadaná vzdálenost ε (ta je vstupním parametrem algoritmu). Pak je tento nalezený bod propojen přímkou s počátečním bodem. Stejně tak později i s koncovým bodem. Mezi nimi je opět hledán nejvzdálenější bod, který přesahuje vzdálenost ε. Pokud žádný takový neexistuje, všechny body, které jsou blíže k přímce, než je vzdálenost ε, jsou vyřazeny. Algoritmus takto rekurzivně rozděluje a zjednodušuje křivku, dokud nejsou vyřazeny všechny body ve vzdálenosti menší, než je ε. Tento algoritmus může být popsán i následujícím pseudokódem [17]:

```
function DouglasPeucker(PointList[], epsilon)
  // Find the point with the maximum distance
  dmax = 0
  index = 0
  end = length(PointList)
  for i = 2 to ( end - 1) {
      d = perpendicularDist(PointList[i], Line(PointList[1], PointList[end]))
      if ( d > dmax ) {
          index = i
          dmax = d
      }
  }
  // If max distance is greater than epsilon, recursively simplify
  if ( dmax > epsilon ) {
      // Recursive call
      res1[] = DouglasPeucker(PointList[1...index], epsilon)
      res2[] = DouglasPeucker(PointList[index...end], epsilon)

      // Build the result list
      ResultList[] = {res1[1...length(res1)-1], res2[1...length(res2)]}
  } else {
      ResultList[] = {PointList[1], PointList[end]}
  }
  // Return the result
  return ResultList[]
end
```

K využití tohoto algoritmu v Javě byla použita knihovna JTS a její funkce *simplify()* [18].

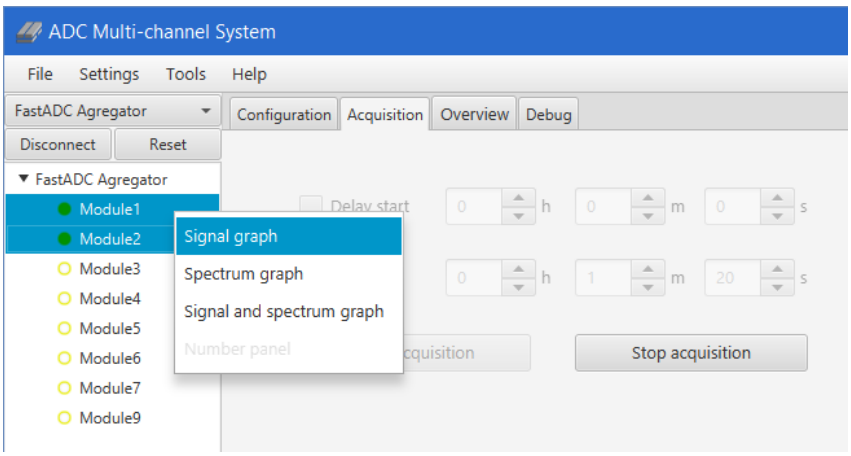
Pro režim vzorku je jako náhrada grafu určena tabulka v okně nazvaném *Number Panel* (Obrázek 15). V této tabulce jsou zobrazovány nalezené vzorky pro všechny moduly, které byly vybrány.



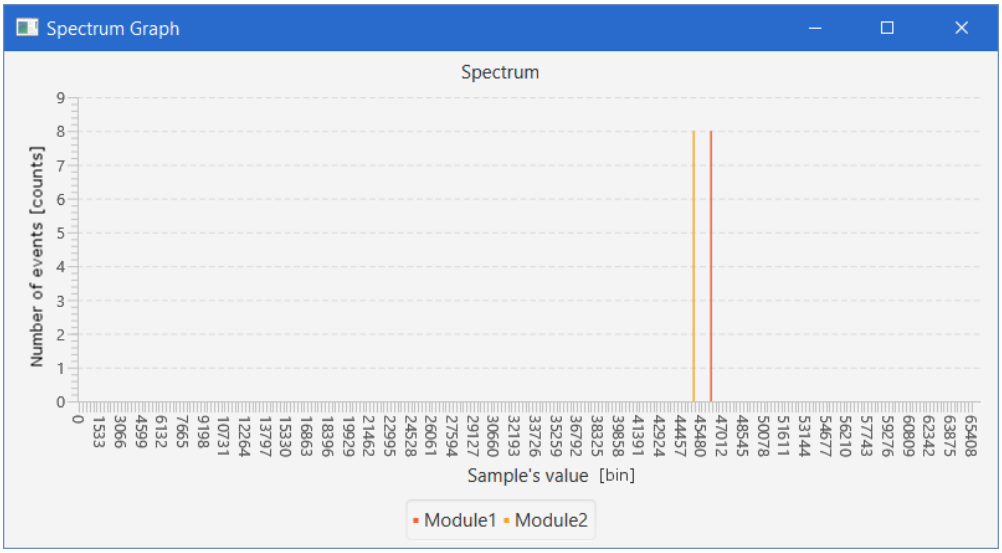
Obrázek 15 – Tabulka zobrazující vzorky

Pro všechny režimy je možno zobrazovat graf amplitudového spektra. V osciloskopickém režimu je v každém průběhu nalezen vzorek odpovídající maximální napěťové úrovni. V režimu vzorku je přijatá hodnota tou jedinou, která je zanesena do amplitudového spektra. Vytváření spektra v paměti a jeho zobrazování jsou oddělené procesy. Z důvodu velkého rozlišení spektra, které je 65536 bodů (odpovídá 16 bitům rozlišení A/D převodníku), nejsou do grafu vynášena všechna data, ale jsou nejprve zjednodušována podobně jako v případě vynášení signálu. Rozdílem je použitý algoritmus. Pro tento účel byl navržen vlastní algoritmus, který v zadaném rozsahovém okně hledá minimální a maximální hodnotu. Tím by mělo být zaručeno, že se pro uživatele významná informace neztratí pouhým špatným zobrazením. Podle rozlišení grafu je tedy rozsah 65536 rozdělen na 2^n oblastí. Konstanta n je vhodně zvolená podle velikosti okna. V každé oblasti je nalezeno maximum a minimum a tyto dvě hodnoty jsou vyneseny do sloupcového grafu (Obrázek 17). Při přiblížení je uživatelem vybraná oblast opět rozdělena na 2^n oblastí a zobrazení je přepočítáno, aby uživatel viděl více detailů (Obrázek 18 a Obrázek 19).

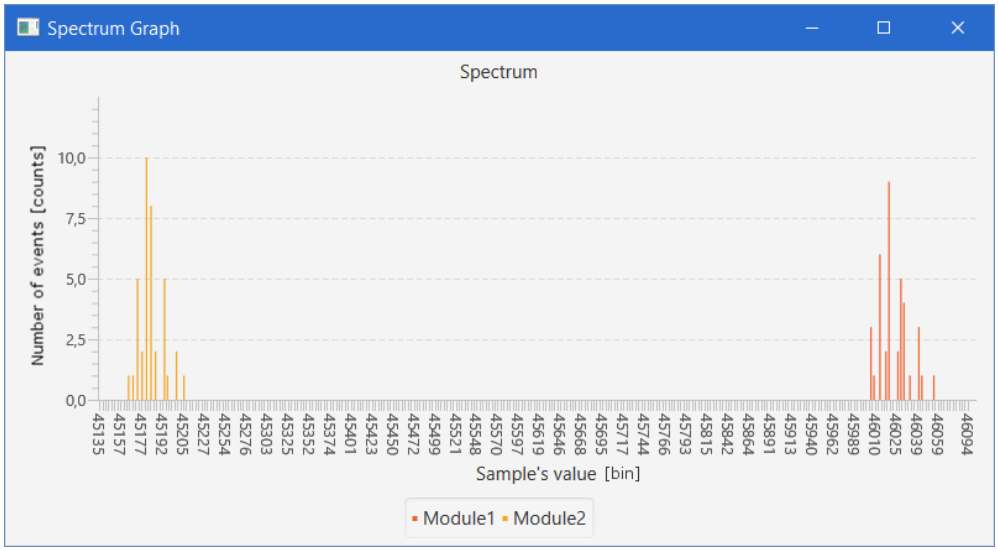
Při otevírání grafu je z kontextové nabídce umožněno také otevřít okno s kombinací signálového grafu a sloupcového grafu spektra (Obrázek 16).



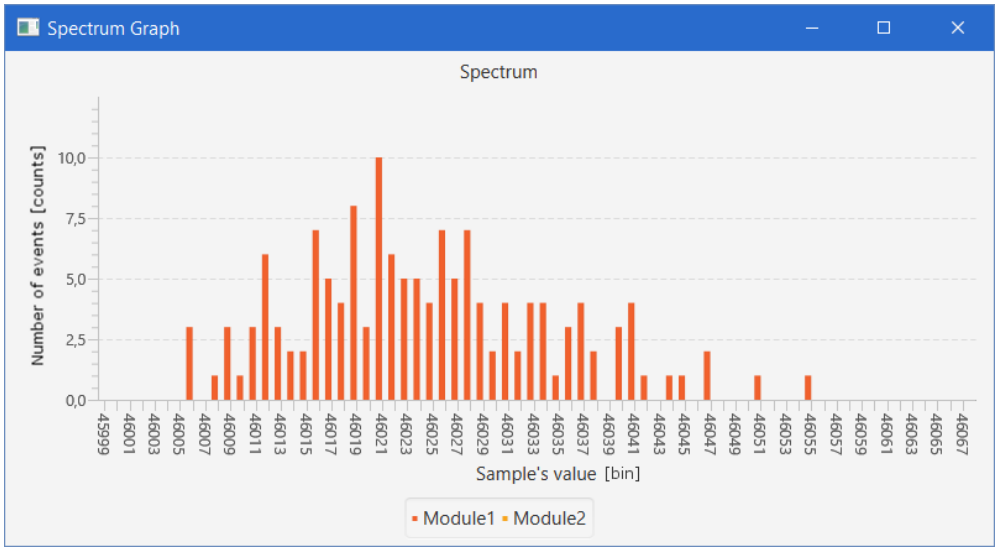
Obrázek 16 – Kontextové menu k otevření grafu



Obrázek 17 – Graf se spektrem (celý rozsah, nepřiblíženo)



Obrázek 18 – Graf se spektrem (po přiblížení je vidět více detailů)



Obrázek 19 – Graf se spektrem (přiblíženo na hodnoty jednotlivých vzorků)

4.4 Uložení a zpracování dat

Pakety přicházející ze zařízení jsou vkládány do zásobníku, ze kterého jsou jiným vláknem vybírány k uložení. K tomuto zásobníku je přistupováno synchronizovaně ze tří vláken. První vkládá pakety do zásobníku, druhé je ukládá na disk a třetí bere naposledy přijatý paket a provádí vizualizaci. Je možné provádět ukládání celého průběhu nebo vypočteného integrálu dle nastavení uživatelem. Volitelně může být uživatelem uložena do souboru připravená konfigurace z formuláře, jež byl popsán v kapitole 4.3.1.

4.4.1 Uložení průběhu

Jestliže je požadováno ukládat celý průběh signálu, jsou ukládacím vláknem vybírány pakety ze sdíleného zásobníku a všechny bajty tohoto paketu jsou zapisovány do otevřeného proudu (BufferedOutputStream), který je navázán na vytvořený binární soubor s příponou .mwav.

Aby bylo zajištěno sledování počtu uložených událostí (paketů) v souboru, jsou na začátku každého souboru vyhrazeny 4 bajty pro uložení 32 bitového čísla (integer). Dosáhne-li počet událostí v souboru nastaveného limitu, je vytvořen nový soubor stejného názvu s vyšším pořadovým indexem (Module1_000, Module1_001).

Tento mechanismus funguje stejně i v případě ukládání vypočteného integrálu s tím, že soubory mají jinou příponu.

4.4.2 Výpočet a uložení integrálu

Je-li požadováno ukládání integrálu průběhu, je ukládacím vláknem také vypočítána plocha pod grafem. Výpočet je prováděn obdélníkovou metodou [19]. Jde o nejjednodušší metodu výpočtu numerického integrálu, která sice není nejpřesnější, což ale v danou chvíli pro účely experimentu nevadí. Interval <a,b>, který odpovídá délce navzorkovaného signálu, je již rozdělen na n stejných dílků.

$$a = x_0 < x_1 < \cdots < x_n = b$$

$$k = 0, \dots, n-1$$

Na každém intervalu mezi dvěma po sobě jdoucími vzorky je vypočítána hodnota y_k, která odpovídá nahrazení části průběhu konstantní funkcí.

$$y_k = \frac{f(x_k) + f(x_{k+1})}{2}$$

Pro výsledný integrál potom platí:

$$\int_a^b f(x)dx \approx \frac{b-a}{n} \sum_{k=0}^{n-1} y_k$$

Vypočtený integrál je uložen do binárního souboru v podobném formátu, jako je ukládán celý průběh, ve formě 32bitového čísla. Všechny části původního paketu jsou tedy zachovány kromě kontrolního součtu a datové části, která je nahrazena tímto 32bitovým číslem odpovídajícím vypočtenému integrálu. Soubor má příponu .mint. Indexace vytvářených souborů při dosažení maximálního počtu událostí funguje podle popisu v kapitole 4.4.1.

4.4.3 Uložení konfigurace

Pro uložení konfigurace bylo navrženo převedení konfiguračního modelu získaného z formuláře do objektu typu JSON. Tento objekt lze uložit do souboru a opět načíst ze souboru zpět do objektu. Pro práci s JSON objekty byla použita knihovna JSON-java [20]. Řetězec zapsaný do souboru je zároveň čitelný uživateli při otevření v textovém editoru, jak uvádí následující příklad:

```
{
    "device_serial_number": "FTYM709K",
    "module_address": 1,
    "input": {
        "posttrig": "0",
        "hyst": "10",
        "lld": "33000",
        "pretrig": "10",
        "count": "500",
        "uld": "0",
        "clkmux": "0"
    },
    "combobox": {
        "mode": "Waveform",
        "endmode": "Number of samples",
        "gatecfg": "Disable",
        "strescfg": "Disable",
        "startmode": "LLD rising edge",
        "extrigcfg": "Disable",
        "discardcfg": "Disable",
        "polarity": "Positive polarity"
    }
}
```

4.5 Distribuce aplikace

Pro připojení zařízení je nutné nainstalovat ovladač D2XX, který může být nalezen na webu společnosti FTDI [21] a do paměti pomocí aplikace FT_Prog zapsat patřičnou konfiguraci [9].

Aplikace je distribuována jako jeden samostatný spustitelný .jar soubor, ve kterém jsou zabaleny všechny použité knihovny. Pro spuštění aplikace je vyžadováno mít nainstalovánu Javu verze 8 a vyšší. Podporovanými systémy jsou Windows a Linux. Aplikace by měla být připravena i pro spuštění na Mac OS, což ale nebylo dosud testováno.

4.5.1 Windows

Pod operačním systémem Windows Vista a vyšším by měla být instalace ovladače provedena automaticky z webu Windows Update po připojení FTDI čipu. V případě výskytu problému nebo použití starší verze OS je možné postupovat podle příručky pro instalaci ovladače na Windows (např. [22]).

4.5.2 Linux

Pod operačním systémem Linux je potřeba stažené soubory včetně ovladače zkopírovat do umístění /usr/local/lib. Dále je nutné provést vytvoření symbolického odkazu na soubor libftd2xx.so.1.1.12. Odkaz musí být pojmenován libftd2xx.so. Navíc je třeba provést nastavení práv souboru libftd2xx.so.1.1.12 na 0x0755.

Z příkazového řádku se provede odstranění ovladače virtuální sériové linky (VCP driver) příkazem „sudo rmmod ftdi_sio“ a „sudo rmmod usbserial“, což je vyžadováno při každém připojení zařízení (v budoucnu bude vytvořen skriptu, kterým budou uvedené příkazy provedeny).

Po provedení výše zmíněných úkonů musí být aplikace spuštěna s administrátorským oprávněním. Veškerý postup uvedený v této podkapitole je podrobně vysvětlen v příručce pro instalaci FTDI ovladače na Linuxu [17].

5 Závěr

Před navržením softwaru pro ovládání elektroniky napojené na detektor reaktorových antineutrin byla prostudována a zdokumentována hardwarová část systému. Bylo nutné pochopit fungování celého systému a způsob, jakým je možné s ním komunikovat přes USB. Byly popsány jednotlivé hardwarové součásti a jejich funkce. Také byl zdokumentován komunikační protokol, který je rozdělen na formát paketů s příkazy odesílanými do zařízení k jeho nastavování a ovládání a na formát paketů s daty odesílanými jednotlivými moduly provádějícími záznam analogového signálu.

Ve druhé fázi práce byly rozebrány požadavky na software a na základě těchto požadavků byl proveden návrh jednotlivých částí softwaru. Byly řešeny jednotlivé dílčí úkoly, např. způsob rozšiřitelnosti komunikační části aplikace o jiný komunikační protokol, provádění konfigurace vybraných skupin modulů, zpracování dat z celého zařízení a jeho modulů a způsob vizualizace dat přicházející z digitalizačního hardwaru. V této fázi byla také zvolena Java jako programovací jazyk pro tvorbu softwaru a použití platformy JavaFX pro realizaci grafického uživatelského prostředí.

Nakonec byla vytvořena aplikace v Javě, která umožňuje připojení digitalizačního systému přes USB k počítači, jeho ovládání a ukládání potřebných souborů. V průběhu akvizice mohou být data zobrazována v grafu, který je přizpůsobován uživatelem jak z hlediska typu grafu a zdrojových dat pro něj, tak podle hardwarové i softwarové konfigurace zařízení.

Software byl navržen s ohledem na další vývoj a nové potřeby experimentu a vyhodnocování dat. V blízkém budoucnu bude dozajista potřeba přidání nových typů zpracování a grafů dle požadavků, které zatím nebyly definovány a objeví se až během realizovaných měření.

Seznam použité literatury

[1] „Lepton", *Wikipedie*, 07-čer-2014. [Online]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Lepton&oldid=11560119. [Viděno: 30-dub-2016].

[2] „Neutrino", *Wikipedie*, 15-úno-2016. [Online]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Neutrino&oldid=13353775. [Viděno: 30-dub-2016].

[3] Mgr. Lukáš Fajt, *Prototyp detektoru reaktorových neutrin*, roč. 2015. Praha: MFF UK.

[4] „Serial Peripheral Interface", *Wikipedie*, 02-led-2014. [Online]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Serial_Peripheral_Interface&oldid=11067126. [Viděno: 20-kvě-2016].

[5] Linear Technology Corporation, „LTC2165/LTC2164/LTC2163 - 16-Bit, 125/105/80Msps Low Power ADCs", 06-zář-2011. [Online]. Dostupné z: http://cds.linear.com/docs/en/datasheet/216543f.pdf.

[6] „Low-voltage differential signaling", *Wikipedia, the free encyclopedia*, 18-bře-2016. [Online]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Low-voltage_differential_signaling&oldid=710714190. [Viděno: 30-dub-2016].

[7] Future Technology Devices International Ltd, „FT232H Single Channel Hi-Speed USB to Multipurpose UART/FIFO IC", 05-zář-2011. [Online]. Dostupné z: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232H.pdf.

[8] „FTDI Utilities". [Online]. Dostupné z: http://www.ftdichip.com/Support/Utilities.htm. [Viděno: 22-kvě-2016].

[9] Future Technology Devices International Ltd, „User Guide for FTDI FT_PROG Utility", 04-čer-2016. [Online]. Dostupné z: http://www.ftdichip.com/Support/Documents/AppNotes/AN_124_User_Guide_For_FT_PROG.pdf.

[10] P. Herout, *Učebnice jazyka Java*, 5. České Budějovice: Kopp, 2010.

[11] „Proč právě Java?" [Online]. Dostupné z: http://www.abclinuxu.cz/clanky/programovani/proc-prave-java. [Viděno: 02-kvě-2016].

[12] „Java Native Access", *Wikipedia, the free encyclopedia*, 05-dub-2016. [Online]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Java_Native_Access&oldid=713659038. [Viděno: 03-kvě-2016].

[13] „JavaFTD2XX: Access FTDI's D2XX via JNA: Wiki: Home — Project Kenai". [Online]. Dostupné z: https://kenai.com/projects/javaftd2xx/pages/Home. [Viděno: 16-kvě-2016].

[14] „About ROOT | ROOT a Data analysis Framework". [Online]. Dostupné z: https://root.cern.ch/about-root. [Viděno: 16-kvě-2016].

[15] Future Technology Devices International Ltd, „Software Application Development D2XX Programmer's Guide", 23-úno-2012. [Online]. Dostupné z: http://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer's_Guide(FT_000071).pdf.

[16] „Knihovna (programování)", *Wikipedie*, 12-led-2016. [Online]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Knihovna_(programov%C3%A1n%C3%AD)&oldid=13222610. [Viděno: 26-kvě-2016].

[17] „Ramer–Douglas–Peucker algorithm", *Wikipedia, the free encyclopedia*, 31-bře-2016. [Online]. Dostupné z:

https://en.wikipedia.org/w/index.php?title=Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm&oldid=712802607. [Viděno: 19-kvě-2016].

[18] „JTS Topology Suite". [Online]. Dostupné z: http://www.vividsolutions.com/jts/JTSHome.htm. [Viděno: 19-kvě-2016].

[19] „Java - algoritmy numerického integrovania - kiwiki". [Online]. Dostupné z: http://www.kiwiki.info/index.php/Java_-_algoritmy_numerick%C3%A9ho_integrovania. [Viděno: 11-kvě-2016].

[20] „stleary/JSON-java", *GitHub*. [Online]. Dostupné z: https://github.com/stleary/JSON-java. [Viděno: 20-kvě-2016].

[21] „D2XX Direct Drivers". [Online]. Dostupné z: http://www.ftdichip.com/Drivers/D2XX.htm. [Viděno: 19-kvě-2016].

[22] Future Technology Devices International Ltd, „FTDI Drivers Installation guide for Windows 7", 15-čer-2015. [Online]. Dostupné z: http://www.ftdichip.com/Support/Documents/AppNotes/AN_119_FTDI_Drivers_Installation_Guide_for_Windows7.pdf.

[23] Future Technology Devices International Ltd, „FTDI Drivers Installation Guide for Linux", 04-kvě-2016. [Online]. Dostupné z: http://www.ftdichip.com/Support/Documents/AppNotes/AN_220_FTDI_Drivers_Installation_Guide_for_Linux%20.pdf.

Seznam obrázků

Seznam tabulek

Seznam zkratek

A/D	Analog/digital (converter)
CERN	Conseil Européen pour la recherche nucléaire
CAEN	Costruzioni Apparecchiature Elettroniche Nucleari
ČVUT	České vysoké učení technické
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIFO	First In First Out
FTDI	Future Technology Devices International
GUI	Graphic User Interface
JNA	Java Native Access
JSON	JavaScript Object Notation
JTAG	Join Test Action Group
LED	Light-Emitting Diode
LLD	Lower Level of Discrimination
LSB	Least Significant Bit
LVDS	Low-Voltage Differential Signaling
MISO	Master In Slave Out
MOSI	Master Out Slave In
MSB	Most Significant Bit
MSPS	Mega Samples Per Second
OS	Operační systém
SCLK	Synchronous Clock
SPI	Serial Peripheral Interface
SS	Slave Select
TOT	Time Over Threshold
UART	Universal Synchronous / Asynchronous Receiver and Transmitter
ULD	Upper Level of Discrimination
USB	Universal Serial Bus
VCP	Virtual COM Port

Obsah přiloženého CD

Bakalarska_prace.pdf

ADCMultichannel.jar

knihovny.rar

dokumentace.rar