Bakalářská práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Logický analyzátor s mikrořadičem

Vít Vaněček
Kybernetika a robotika

Květen 2020
Vedoucí práce: doc. Ing. Jan Fischer, CSc.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vaněček** Jméno: **Vít** Osobní číslo: **474475**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Logický analyzátor s mikrořadičem

Název bakalářské práce anglicky:

Microcontroller Based Logic Analyser

Pokyny pro vypracování:

Navrhněte a s využitím vybraných mikrořadičů řady STM 32 napojených prostřednictvím rozhraní USB na PC realizujte měřicí přístroj - logický analyzátor pro výukové účely. Cílem je dosáhnout co nejjednoduššího obvodového řešení s využitím desky Nucleo nebo i jen samotného mikrořadiče na nepájivém kontaktním poli. Prověřte možnost využití GUI PulseView pro zobrazení výsledků i pro analýzu komunikačních protokolů. Posuďte možnost realizace doplňkové funkce osciloskopu s využitím GUI PulseView. Srovnejte výsledné parametry hotového přístroje s parametry analyzátorů, případně osciloskopů realizovaných s využitím modulu Arduino.

Seznam doporučené literatury:

[1] Yiu, J.: The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors,
[2] STMicroelectronics: RM0316, STM32F3 Reference manual
[3] STMicroelectronics: DS10362 - STM32F303 Data

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Jan Fischer, CSc., katedra měření FEL

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

Datum zadání bakalářské práce: **30.01.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce:
do konce letního semestru 2020/2021

doc. Ing. Jan Fischer, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____ Datum převzetí zadání

_____ Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací

V Praze dne 14. 8. 2020 .....

Poděkování

Tímto bych rád poděkoval vedoucímu práce doc. Ing. Janu Fischerovi, CSc. za ochotu a pomoc při tvorbě práce, a to i v době distanční výuky. Díky patří také Jiřímu Hladíkovi z firmy STMicroelectronics za konzultace na počátku projektu. Dále děkuji rodině a přátelům za podporu při studiích.

Abstrakt

Tato práce se zaměřuje na návrh logického analyzátoru pro výukové účely za pomocí mikrořadiče. Cílem je dosáhnout co nejjednoduššího obvodového řešení, čehož je dosaženo použitím mikrořadiče STM32F303RE osazeného na desce Nucleo a počítačové aplikace PulseView pro vizualizaci a analýzu dat. V práci je nejdříve navržen firmware pro použitý mikrořadič, využívající hardwarové prostředky mikrořadiče pro dosažení co nejlepších parametrů logického analyzátoru. Následně je navrhnut komunikační protokol pro zajištění komunikace mezi mikrořadičem a počítačovou aplikací. Na konec je modifikována počítačová aplikace PulseView pro přidání podpory navrženého zařízení a komunikačního protokolu.

Klíčová slova: STM32; NUCLEO-F303RE; Logický Analyzátor; Mikrořadič; Sigrok; PulseView

Abstract

The aim of this thesis is to develop a logic analyzer form educational purpose using a microcontroller. The main goal is to create the simplest design. This has been accomplished by using the STM32F303RE microcontroller on the Nucleo board and the computer application PulseView for data visualization and analysis. Thesis begins by developing a firmware for the microcontroller that uses hardware peripheral of the microcontroller to accomplish the best performance of the logic analyzer. Then a communication protocol is developed to enable communication between the microcontroller and the computer application. Lastly PulseView is modified to add support for the developed device and communication protocol.

Keywords: STM32; NUCLEO-F303RE; Logic Analyzer; Microcontroller; Sigrok; PulseView

Title translation: Microcontroller Based Logic Analyser

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Motivací ke vzniku této práce je možnost si jednoduše zobrazit digitální signály ve výuce s mikroprocesorovými systémy a v podobných předmětech. Digitální signály se vyskytují v projektech, na kterých studenti pracují a mohou mít například podobu komunikací jako jsou UART, I2C, SPI apod. Pro zobrazení těchto signálů je potřeba laboratorní osciloskop nebo logický analyzátor. Použití těchto přístrojů ovšem omezuje možnost studenta pracovat na projektech mimo laboratoř, a to je motivací k navrhnutí jednoduchého zařízení, které může student využít jako alternativu za laboratorní logický analyzátor.

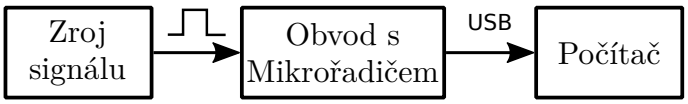
Úkolem je navrhnout co nejjednodušší logický analyzátor s využitím mikrořadiče, který umožní zobrazení digitálních komunikací, jako jsou již zmíněné UART a I2C. Práce bude zaměřena na mikrořadič STM32F303RE na vývojové desce Nucleo a pro vizualizaci dat bude využita počítačová aplikace. Kromě samotného zobrazení digitálních komunikací v počítačové aplikaci je vhodné mít možnost komunikaci analyzovat a dekódovat data, která jsou odesílána. Práce proto prověří možnost použití aplikace PulseView z projektu sigrok, která umožňuje vizualizaci dat z logického analyzátoru a také obsahuje analýzy digitálních komunikací. Hlavní cílem je vytvoření zařízení, pomocí kterého si může student digitální signály zobrazit a analyzovat mimo školní laboratoře a pracovat na projektech i doma, například při distanční výuce.

Kapitola 2
Rozbor a stanovení cílů

Jak již bylo zmíněno v úvodu, cílem této práce je navrhnout logický analyzátor pro výukové účely. Pomocí logického analyzátoru dostává student možnost si naměřit a hlavně zobrazit digitální signály vyskytující se například v projektech, které studenti vytváří v předmětech „Laboratoře průmyslové elektroniky a senzorů“ (`LPE`) a v předmětu „Vestavné systémy“ (`VSY`). V nich studenti programují mikrořadiče a objevují se zde digitální signály v podobě sériové komunikace tipu `UART`, `I2C` anebo `SPI`. Ty jsou využity na komunikaci s různými digitálními senzory nebo i s počítačem. Vizualizace těchto digitálních signálů umožňuje lépe pochopit, co se v projektu děje anebo nalézt chyby.

Student má možnost si digitální i analogové signály vizualizovat ve školních laboratořích pomocí laboratorních osciloskopů. Nevýhodou je omezení, pokud chce na projektu pracovat mimo laboratoř nebo pokud není dostatek osciloskopů pro každého. Toto je motivací k nalezení alternativy, jako například využít samotné mikrořadiče, se kterými studenti pracují. Použití mikrořadiče jako náhradu za laboratorní osciloskop umožní studentovi pracovat na projektu doma, o víkendu, i při distanční výuce.

Zvyšující se rychlost a schopnosti moderních mikrořadičů dává možnost využít je pro zaznamenání signálu, který se poté vizualizuje. Pro vizualizaci je možné připojení externího displeje napojeného na mikrořadič, ale jako jednoduší řešení se nabízí využít počítač nebo notebook, na kterém student pracuje. Poté bude stačit vytvořit nebo zvolit vhodnou aplikaci do počítače, která dokáže zaznamenaný signál z mikrořadiče přijmout a vizualizovat. Na obr. 2.1 je znázorněné jednoduché schéma tohoto procesu.

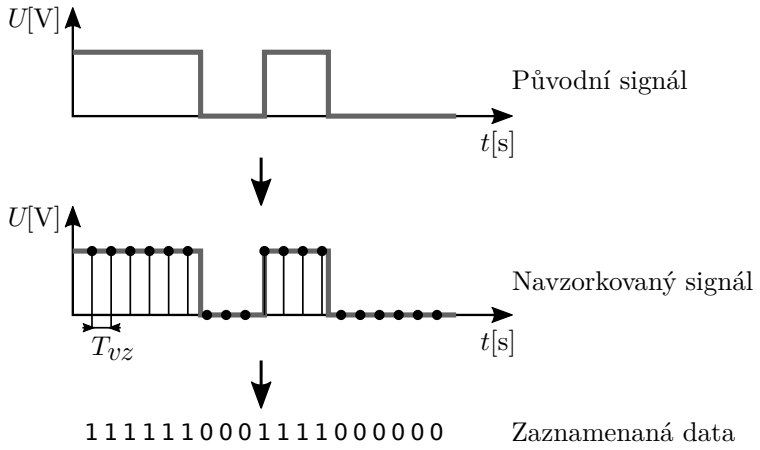


Obrázek 2.1. Schéma logického analyzátoru s mikrořadičem.

Pro studenty již existuje projekt `LEO` pracující s tímto principem. `LEO` zahrnuje firmware pro mikrořadič a software do počítače, které jsou volně ke stažení a udělají z mikrořadiče jednoduchý osciloskop [1]. Osciloskop umožňuje zobrazit primárně analogové, ale i digitální signály za pomocí převodníků ADC obsažených na mikrořadiči. Interní převodníky ADC na čipu mikrořadiče jsou ovšem pomalé a pro digitální signál nemusíme znát jeho analogové hodnoty. Vhodnější je tedy použití digitálních vstupů mikrořadiče, které určují pouze aktuální stav digitálního signálu a umožňují zaznamenat signál rychleji.

2.1 Proces vzorkování signálu

Vzorkování je proces odečítání hodnoty signálu v pravidelných intervalech. Pro logický analyzátor to znamená v pravidelných intervalech určit, zda je signál ve stavu 1 (High) nebo 0 (Low). Více vzorků dohromady tvoří záznam, který může být uložen do digitální paměti a poté vizualizován. Na obr. 2.2 je naznačen tento proces.



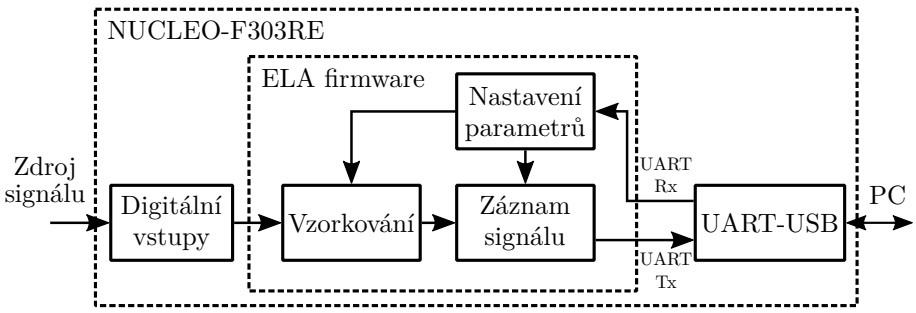
Obrázek 2.2. Vzorkování digitálního signálu.

Perioda vzorkování (interval mezi vzorky) $T_{vz}[s]$ je určen vzorkovacím kmitočtem $f_{vz}[Hz]$ pomocí vzorce (1).

$$T_{vz} = \frac{1}{f_{vz}} \tag{1}$$

2.2 Použitý mikrořadič pro logický analyzátor

Dle zadání práce má logický analyzátor využívat mikrořadiče s řady STM32 a rozhraní USB pro spojení s PC. Podmínkou je také co nejjednodušší obvodové řešení. Byl proto zvolen mikrořadič STM32F303RE ve vývojové desce NUCLEO-F303RE. Výhodou desky NUCLEO je integrovaný port USB a převodník UART-USB, který lze využít pro komunikaci s PC. Díky tomu je možné pro logický analyzátor použít samostatnou desku NUCLEO-F303RE připojenou k počítači a není nutný externí hardware, což zjednodušuje obvodové řešení. Obr. 2.3 znázorňuje schéma logického analyzátoru s použitím desky NUCLEO-F303RE.



Obrázek 2.3. Schéma logického analyzátoru s deskou NUCLEO-F303RE.

Blok „Nastavení parametrů“ na obrázku 2.3 naznačuje možnost nastavení parametrů vzorkování, jimiž jsou například vzorkovací kmitočet a počet vzorků. Tyto parametry je možné nastavit v počítačové aplikaci a existují hlavně kvůli omezené paměti mikrořadiče. Při použití vysoké vzorkovací frekvence na analýzu pomalého signálu se paměť rychle zaplní a nemusí být zaznamenán celý signál. Proto je poskytnuta možnost parametry vzorkování nastavit na jiné hodnoty.

2.3 Počítačová aplikace pro vizualizaci dat

Hlavními nároky počítačové aplikace pro logický analyzátor jsou možnosti nastavit parametry vzorkování a zobrazení dat přijatých od mikrořadiče. Při zaznamenání nějaké digitální komunikace je ovšem užitečné mít možnost si komunikaci analyzovat a zobrazit data, která jsou v komunikaci odesílána. Pokud by tedy v rámci práce byla napsána vlastní aplikace, bylo by nutné kromě samotné aplikace naprogramovat i knihovny pro analýzu digitálních signálů. Toto by bylo ovšem pracné a pro ulehčení by bylo vhodné nalézt již vytvořenou aplikaci.

Zadání práce doporučuje prověřit možnost využití aplikace PulseView, která je součástí projektu sigrok, jehož cílem je vytvořit uživatelské aplikace pro sbírání a vizualizaci dat ze zařízení jako jsou osciloskopy, multimetry a logické analyzátory [2]. PulseView poskytuje uživatelské rozhraní, které umožňuje nastavení parametrů vzorkování, vizualizaci zaznamenaného signálu a analýzu velké řady digitálních komunikací [3]. Výhodou této aplikace je navíc veřejný zdrojový kód, což umožňuje aplikaci upravit a popřípadě přidat podporu nových zařízení. Cílem této práce je tedy vytvoření logického Analyzátoru, který je kompatibilní s aplikací PulseView.

Kapitola 3
Vývoj ELA firmwaru pro mikrořadič

Základem vytvoření logického analyzátoru je napsat firmware pro zvolený mikrořadič. Ten umožní vzorkování signálu, uložení zaznamenaných vzorků do paměti mikrořadiče a odeslání záznamu do počítače. Firmware musí také využít prostředky mikrořadiče, aby měl logický analyzátor co nejlepší parametry. Firmware je psán v počítači v programovacím jazyce C++ a po kompilaci je nahrán do mikrořadiče pomocí USB konektoru na desce NUCLEO. V této kapitole bude popsán návrh ELA firmwaru a využití hardwarových periferií mikrořadiče pro uskutečnění funkcí logického analyzátoru.

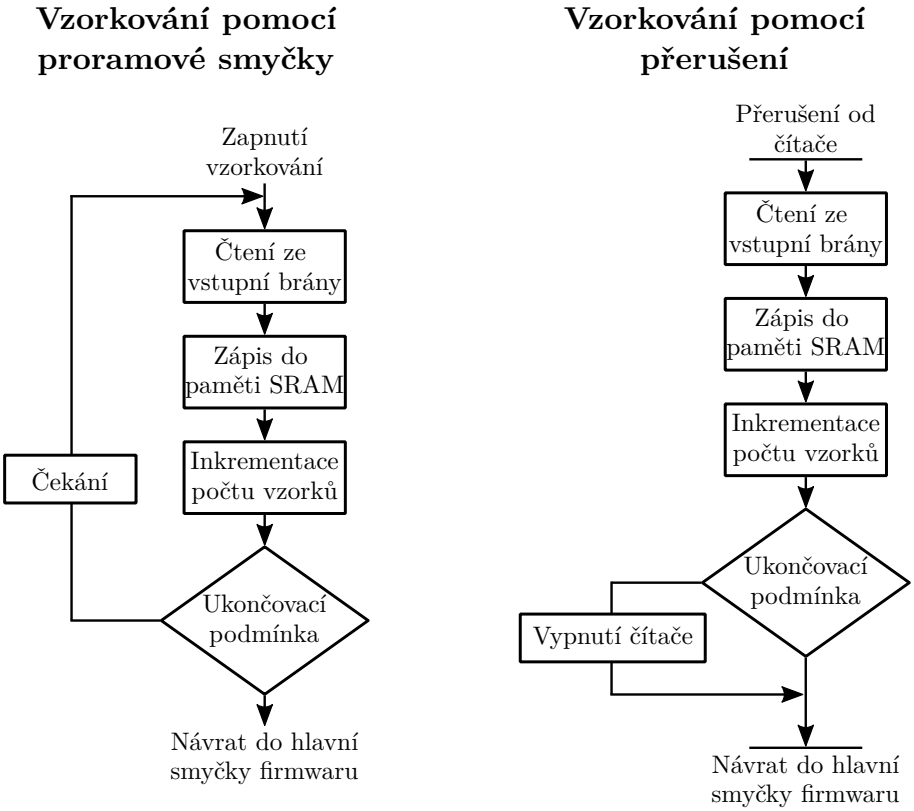
3.1 Čtení vzorků digitálního signálu a ukládání do paměti

Jednou ze základních schopností logického analyzátoru je vzorkování digitálního signálu a uložení vzorků do paměti. V případě mikrořadiče je vzorkování uskutečněno čtením dat ze vstupní brány a uložení do paměti SRAM. Konkrétně je čteno prvních 8 bitů ze vstupního registru brány C neboli z pinů `PC0` až `PC7` a data jsou uložena do osmibitového paměťového pole v SRAM. Bity jsou ze vstupní brány čteny najednou a díky tomu má logický analyzátor 8 kanálů.

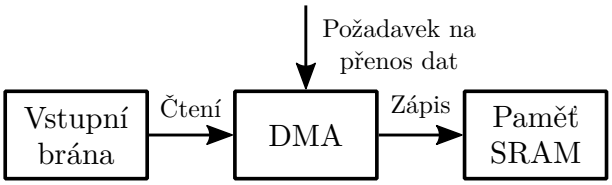
Přenos dat ze vstupní brány musí probíhat pravidelně s frekvencí stejnou jako je vzorkovací frekvence f_{vz}, čehož lze dosáhnout několika způsoby. Nejjednodušším způsobem je programová smyčka, která data přečte, uloží a čeká nastavený čas. Toto řešení ovšem vyžaduje přesně spočítat, jak dlouho smyčka trvá. Smyčka dále zatěžuje procesor mikrořadiče, který nemůže vykonávat jiné funkce. Možná podoba programové smyčky je naznačena na obr. 3.1 vlevo, kdy je po odstartování vzorkování spuštěna a po dosažení nastaveného počtu vzorků se firmware vrací do hlavní smyčky.

Dalším způsobem je pravidelné přerušení vyvolané čítačem. Při přerušení je vyvolána funkce na odečtení vzorku, která probíhá podobně, jako v případě programové smyčky. Načasování je v tomto případě přesnější díky použití čítače a po dokončení odečtení vzorku se firmware může vrátit do hlavní smyčky a vykonávat jiné funkce. Po dosažení nastaveného počtu vzorků je čítač vypnut a odeslání dat je dokončeno v hlavní smyčce firmwaru. Při velkých vzorkovacích frekvencích tento způsob ovšem značně zatěžuje procesor mikrořadiče, podobně jako v případě programové smyčky. Na obr. 3.1 vpravo je možná podoba funkce, která je vykonána při přerušení od čítače.

Posledním způsobem uskutečnění přenosu dat v mikrořadiči je pomocí periferie DMA. Tato periferie umožňuje přenést data v mikrořadiči nezávisle na procesoru, což umožňuje přenos značně urychlit a také uvolnit procesor, který může vykonávat jiné funkce [4]. Vzorkování v ELA firmwaru bylo tedy uskutečněno tímto způsobem, kdy DMA je nastaveno pro čtení dat z registru vstupní brány a uložení do paměťového pole v SRAM. Tento přenos uskutečňuje DMA na základě požadavku, který lze vyvolat například pomocí čítače. Na obr. 3.2 je naznačen proces přenosu dat pomocí DMA.



Obrázek 3.1. Metody vzorkování signálu.



Obrázek 3.2. Přenos dat ze vstupní brány do paměti SRAM pomocí DMA.

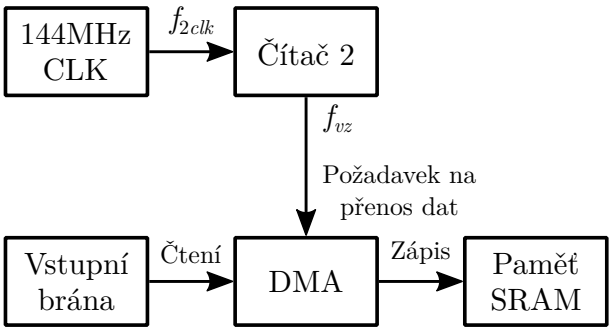
3.2 Časování DMA požadavku pomocí čítače

Čítače v mikrořadiči slouží pro funkce vyžadující přesné načasování nebo periodické opakování. Základní funkce čítačů je počítání pulsů vstupního hodinového signálu (CLK), kdy čítač počítá do předem nastavené maximální hodnoty. Po dosažení této hodnoty se vynuluje a proces se opakuje. Tohoto lze také využít pro periodické vyvolání požadavku na periferii DMA, díky čemuž je možné dosáhnout pravidelného přenosu dat s přesně danou periodou, kterou lze měnit pomocí maximální hodnoty čítače. [5]

Mikrořadič obsahuje několik čítačů s různými funkcemi a pro účely vyvolání DMA přenosu bude využit čítač 2. Tento čítač je 32bitový [6], což je výhodné pro dosažení nízkých frekvencí f_{vz} bez nutnosti použití předděličky hodinového signálu. Frekvence vstupního hodinového signálu čítače f_{2clk} byla nastavena na dvojnásobek hodinového signálu samotného mikrořadiče, čehož je využito pro jemnější nastavení frekvence f_{vz}. Na základě zvoleného hodinového signálu f_{2clk} a žádané vzorkovací frekvence f_{vz} logického analyzátoru je poté možné určit maximální hodnotu čítače T_{2max} pomocí vzorce (1).

$$T_{2max} = \frac{f_{2clk}}{f_{vz}} - 1 \tag{1}$$

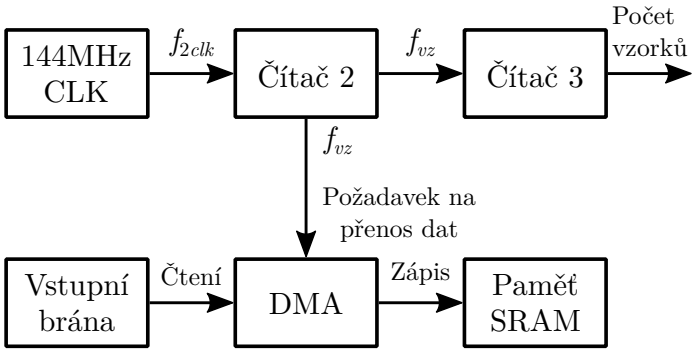
Na obr. 3.3 je proces záznamu s přidaným Čítačem 2.



Obrázek 3.3. Načasovaný požadavek DMA pomocí čítače 2.

3.3 Počítání vzorků a ukončení záznamu

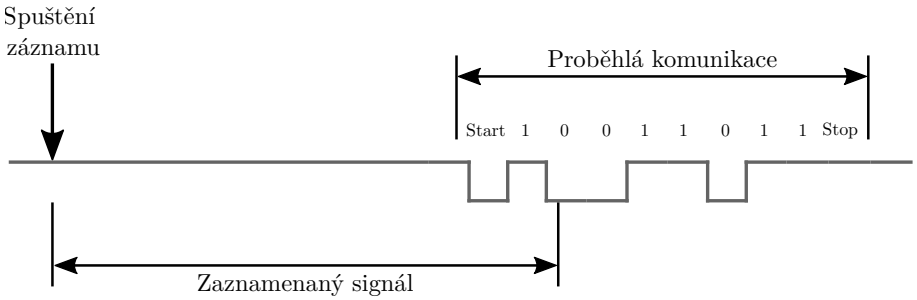
Čítače mohou kromě hodinového signálu také počítat události vyvolané jiným čítačem. Lze nastavit, aby čítač 3 počítal dosažení maximální hodnoty čítače 2, neboli odečtení jednoho vzorku signálu. Tím dosáhneme toho, že hodnota čítače 3 bude odpovídat aktuálnímu počtu vzorků v paměti mikrořadiče. Čítač je dále nastaven do režimu, kdy po dosažení maximální hodnoty vyvolá přerušení, ve kterém se zastaví čítač 2 a tím i vzorkování. Změnou maximální hodnoty čítače 3 je potom měněn počet zaznamenaných vzorků. Maximální počet vzorků nastavený uživatelem je přímo nastaven jako maximální hodnota čítače 3. Na obr. 3.4 je proces záznamu s počítáním vzorků pomocí čítače 3.



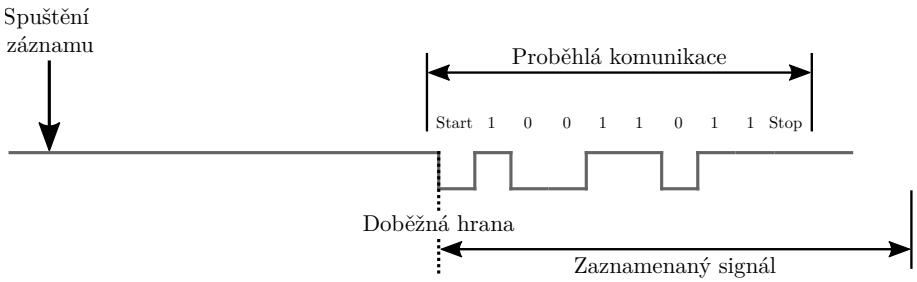
Obrázek 3.4. Počítání vzorků pomocí čítače 3.

3.4 Spuštění vzorkování na základě podmínky

V některých případech, například při pozorování digitální komunikace je vhodné, aby logický analyzátor nezačal vzorkovat okamžitě po spuštění uživatelem, ale na základě nastavené podmínky. Na obr. 3.5 je ukázka případu, kdy logický analyzátor začne vzorkovat okamžitě po spuštění uživatelem, kdy byl zaznamenán převážně neměnící se signál a pouze část komunikace UART. Na obr. 3.6 začal logický analyzátor vzorkovat po doběžné hraně na vstupním signálu, která v případě UART značí počátek odesíláni dat. V tomto případě byla zaznamenána celá komunikace.



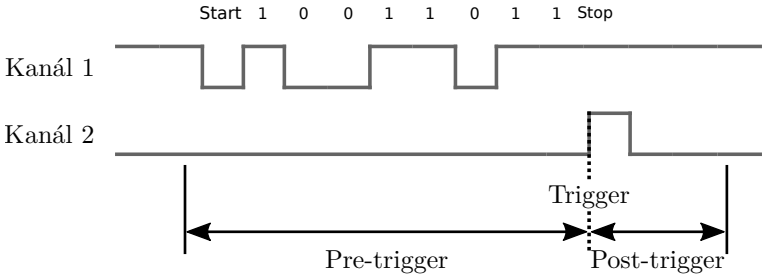
Obrázek 3.5. Záznam UART komunikace bez trigger podmínky.



Obrázek 3.6. Záznam UART komunikace s trigger podmínkou nastavenou doběžnou hranu.

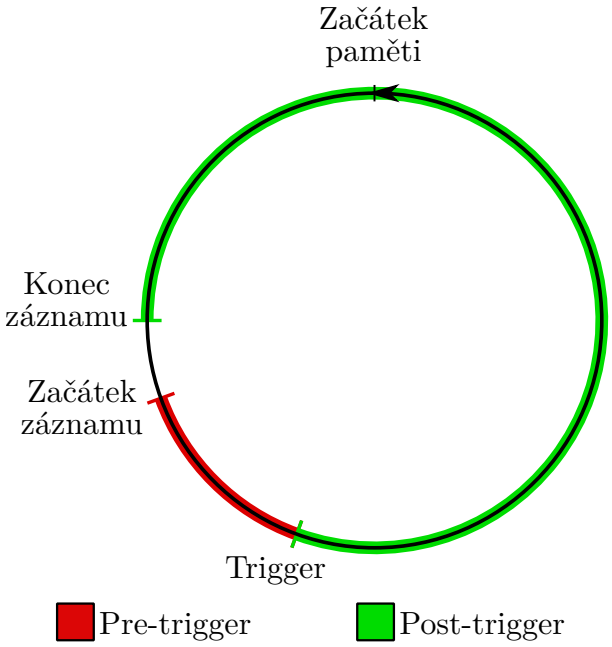
Pro účely spouštění záznamu na základě podmínky byla implementována funkce trigger (spouštěč). Tato funkce umožňuje nastavit, aby se záznam spustil až po tom, co nastane náběžná, doběžná nebo libovolná hrana na jednom z kanálů logického analyzátoru. Pro zjištění, zda nastala hrana vstupního signálu je využito externího přerušení. Externí přerušení lze v mikrořadiči nastavit na libovolný pin a lze také upřesnit, na kterou hranu má přerušení reagovat [6]. Po vyvolání externího přerušení je odstartováno vzorkování dle nastavených parametrů.

Někdy může být užitečné vidět část zaznamenaného signálu předtím, než nastala trigger podmínka. Proto je možné v logickém analyzátoru nastavit nejen maximální počet vzorků, ale i počet vzorků před trigger podmínkou. Záznam je pak rozdělen do části před trigger podmínkou (pre-trigger) a části po podmínce (post-trigger). V příkladu na obrázku 3.7 značí puls na kanálu 2 ukončení komunikace. Trigger podmínka je tedy nastavena na náběžnou hranu na kanálu 2 a samotná komunikace je zaznamenána v pre-trigger části.



Obrázek 3.7. Záznam se vzorky před trigger podmínkou.

Kvůli funkci záznamu před trigger podmínkou je nutné vzorkovat vstupní signál i předtím, než podmínka nastane, čehož je možné dosáhnout nastavením periferie DMA do cyklického režimu. V tomto režimu přenáší DMA data ze vstupní brány do paměťového pole v SRAM a po dosažení konce pole se vrací na začátek a přepisuje stará data [4]. Na obr. 3.8 je naznačeno ukládání záznamu v cyklickém režimu pro funkci trigger s pre-trigger částí.



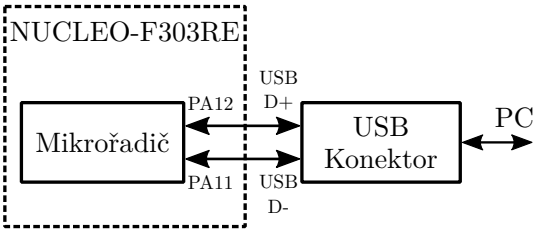
Obrázek 3.8. Uklání v cyklickém režimu.

Nastavená trigger podmínka proces vzorkování mírně pozmění. Po nastavení trigger podmínky a spuštění vzorkování je zapnut pouze čítač 2, čímž započne neustálý přenos dat pomocí DMA v cyklickém režimu. Dále je nastaveno externí přerušení dle trigger podmínky, které spustí čítač 3. Maximální hodnota čítače 3 je nastavena na počet vzorků po trigger podmínce a po dosažení této hodnoty je vzorkování ukončeno. Díky cyklickému režimu jsou po ukončení vzorkování v paměti SRAM uloženy vzorky před i po trigger podmínce.

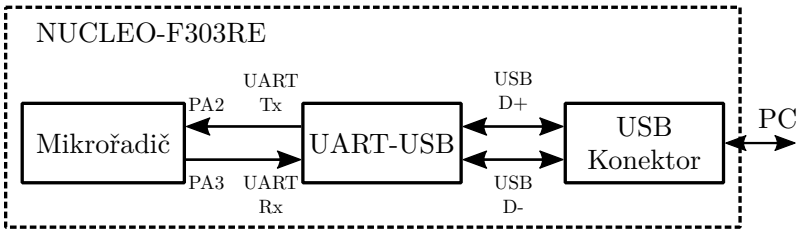
S nastaveným počtem vzorků před trigger podmínkou může dojít k situaci, kdy trigger nastal brzy po spuštění vzorkování a v paměti je zaznamenáno méně vzorků, než je požadováno uživatelem. Pro poznání této situace je využito přerušení periferie DMA, které nastane po prvním zaplnění paměťového pole. Dále je po přerušení čítače 3 uložena poslední adresa paměťového pole, do které DMA zapsalo. Z těchto dvou informací je po ukončení záznamu zjištěno, kolik je v paměti uložených vzorků. Do aplikace v počítači je zaznamenaný signál odeslán i v případě nižšího počtu vzorů.

3.5 Periferie pro komunikaci mikrořadiče s počítačem

Komunikace mezi logickým analyzátorem a počítačem probíhá pomocí USB a pro uskutečnění této komunikace je možné využít periferie USB v mikrořadiči nebo převodník UART-USB na desce Nucelo. Periferie USB umožňuje připojit datové piny externího USB konektoru na piny PA11 a PA12 mikrořadiče, viz obr. 3.9, a uskutečnit komunikaci přímo pomocí mikrořadiče. Tento způsob komunikace má výhodu v rychlosti, ale nastavení a použití periferie USB ve firmwaru je složité a nevýhodná je také nutnost připojení externího USB konektoru. Jednodušší možností je využití převodníku UART-USB, který je spojen s USB konektorem integrovaným na desce Nucleo. Převodník je připojen k periferii USART pomocí pinů PA2 a PA3, viz obr. 3.10, která je oproti USB snadnější na použití. [6]



Obrázek 3.9. Propojení s počítačem pomocí externího USB konektoru.



Obrázek 3.10. Propojení s počítačem pomocí převodníku UART-USB.

Pro logický analyzátor byla zvolena metoda s UART-USB převodníkem a periferií USART. Po prvotním nastavení periferie probíhá odesílání dat pomocí zápisu do odesílacího registru a přijetí čtením z přijímacího registru. V ELA firmwaru je využito přerušení po přijetí dat, které vyvolá přenesení z přijímacího registru do paměťového pole, ze kterého jsou dále zpracována. Při odesílání je nejdříve zkontrolováno, zda byla předchozí data odeslána a následně jsou zapsána do odesílacího registru.

3.6 Generátor obdélníkového signálu

Pro účely testování samotného logického analyzátoru nebo pro funkce vyžadující hodinový signál, byl do logického analyzátoru implementován generátor obdélníkového signálu. Pro tento generátor byla zvolena frekvence $f_{out} = 10\text{kHz}$ a střída $D_4 = 50\%$. Generování samotného signálu zařizuje PWM režim čítače 4, kdy obdélníkový signál generuje čítač na pinu mikrořadiče. Konkrétně byl zvolen kanál 3 čítače 4, který je propojen s pinem PB8 na mikrořadiči. [5–6]

Pro zajištění zvolené výstupní frekvence f_{out} je nutné dopočítat maximální hodnotu čítače T_{4max}. Tato hodnota závisí na frekvenci f_{4clk} hodinového signálu čítače, který byl nastaven na frekvenci 72MHz. Maximální hodnota T_{4max} je pak dopočítána vzorcem (2), který vznikl změnou proměnných ve vzorci (1). Po dosazení vychází hodnota $T_{4max} = 7199$.

$$T_{4max} = \frac{f_{4clk}}{f_{out}} - 1 \tag{2}$$

Střídu výstupního signálu D_4 ovlivňuje hodnota T_{CCR3} zapsaná v porovnávacím registru kanálu 3. Jakmile čítač dosáhne této hodnoty, je výstupní pin PB8 překlopen ze stavu 1 do stavu 0 a po dosažení maximální hodnoty čítače je pin překlopen zpět. Pro vypočítání hodnoty porovnávacího registru byl použit vzorec (3). Po dosazení a zaokrouhlení vychází hodnota $T_{CCR3} = 3600$.

$$T_{CCR3} = T_{4max}\frac{D_4}{100} \tag{3}$$

3.7 Použité hardwarové knihovny pro ovládání periferií

Pro nastavení a ovládání použitých periferií, jako je DMA a čítače, je nutné správně zapisovat a číst data z jejich kontrolních registrů. Jak mají tato data vypadat pro správnou konfiguraci, v jaké sekvenci se mají zapisovat a adresy registrů je nutné zjistit z dokumentace mikrořadiče. Tento proces je možné zjednodušit použitím knihoven, jako Arduino, mbed a STM32F3 HAL, které slouží jako prostředník mezi psaným firmwarem a periferiemi mikrořadiče a mají v sobě již implementovaný postup konfigurace a ovládání periferií. Pro logický analyzátor byla použita knihovna STM32F3 HAL, která je oproti knihovnám jako je mbed složitější na použití, ale výkonově nenáročná a umožňuje jednoduchý přístup přímo k registrům periferií, čehož je v ELA firmwaru také využito. [7]

3.8 Použité programové nástroje pro vývoj ELA firmwaru

Pro základní konfiguraci knihoven STM32F3 HAL byla použita aplikace STM32CubeMX od firmy STMicroelectronics. Tato aplikace poskytuje uživatelské rozhraní pro nastavení periferií mikrořadiče a umožňuje vygenerovat zdrojový kód, ve kterém jsou již volány funkce knihoven STM32F3 HAL, zajišťující zvolené nastavení [8]. Aplikace umožňuje psát uživatelský kód přímo do vygenerovaného, ale pro ELA Firmware byl uživatelský kód psán zvlášť.

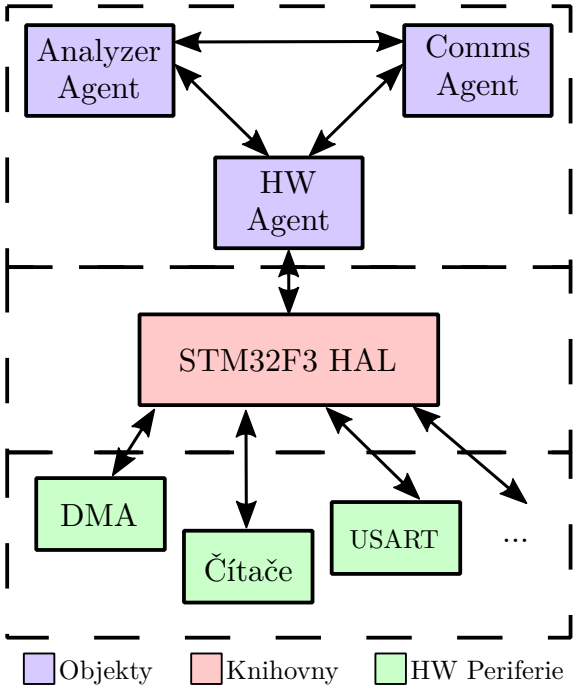
Firmware je možné vyvíjet v různých vývojových prostředích, jako jsou STMCubeIDE, Keil a podobné, ale pro ELA firmware bylo zvoleno prostředí VSCode s rozšířením PlatformIO. VSCode je vývojové prostředí od firmy Microsoft, ve kterém je možné vyvíjet v řadě programovacích jazyků, jako jsou C/C++, Python a podobné. Do prostředí VSCode je možné stahovat rozšíření, která umožní podporu dalších programovacích jazyků i vývoj firmwaru pro mikrořadiče. Do VSCode bylo nainstalováno rozšíření PlatformIO, které umožňuje vytvářet firmware pro velké množství mikrořadičů, včetně použitého STM32F303RE. S rozšířením PlatformIO je možné firmware pro mikrořadič zkompilovat přímo v prostření VSCode a také lze využít nástroje pro debugování při hledání chyb. [9–10]

Pro vývoj pomocí PlatformIO byl stažen balíček podpory mikrořadičů řady STM32 ve kterém je obsažena i vývojová deska NUCLEO-F303RE. PlatformIO umožňuje také použití hardwarových knihoven STM32F3 HAL, které jsou při první kompilaci automaticky staženy. Pro nastavení použitého mikrořadiče a hardwarových knihoven slouží v projektu soubor *platformio.ini*. S konfiguračním souborem *platformio.ini* lze dosáhnout i podpory více mikrořadičů, mezi kterými je možné přepínat nebo zkompilovat firmware pro všechna podporovaná zařízení. Firmware se pro různá zařízení přizpůsobí pomocí maker, která jsou nastavena podle zvoleného zařízení.[10]

3.9 Struktura a rozvržení ELA firmwaru

Psaní firmwaru pro mikrořadič je možné ve dvou programovacích jazycích, kterými jsou C a C++. Jazyk C++ je oproti C komplexnější, ale umožňuje více funkcionalit a možností zpřehlednit firmware. Pro psaní ELA firmwaru byl proto zvolen jazyk C++, ve kterém jsou využity hlavně možnosti vytvoření tříd a objektů, které umožňují firmware lépe rozčlenit a strukturovat. C++ také umožňuje kompatibilitu s kódem psaným v jazyce C, čehož je využito při použití hardwarových knihoven STM32F3 HAL.

Pro jednodušší úpravu a přehlednost je firmware strukturován do 3 hlavních objektů, kterými jsou Analyzer Agent, Comms Agent a HW Agent. Každý objekt je vytvořený za jiným účelem a vzájemně využívají funkce zbylých objektů. Výhodou tohoto přístupu je možnost objekt upravit bez zásahu do zbylých dvou a také lepší orientace v kódu. Kromě těchto objektů obsahuje i speciální funkce určené například pro kompatibilitu s knihovnami STM32F3 HAL a dále inicializační funkce vygenerované aplikací SMT32CubeMX. Na obr. 3.11 je vizualizována struktura firmwaru, včetně hardwarových knihoven a periferií.



Obrázek 3.11. Vizualizace struktury firmwaru.

3.9.1 Popis objektu HW Agent

Objekt HW Agent slouží primárně jako prostředník mezi hardwarovými knihovnami a zbytkem ELA firmwaru. Zbylé objekty firmwaru tedy nepoužívají přímo funkce z hardwarových knihoven, ale funkce z objektu HW Agent. Tento přístup byl zvolen hlavně pro nezávislost zbytku firmwaru na použitých hardwarových knihovnách a pokud jsou změněny knihovny nebo druh mikrořadiče, je nutné ve firmwaru upravit hlavně objekt HW Agent. Toto značně zjednodušuje úpravu ELA firmwaru, nebo i přidání kompatibility s dalším mikrořadičem. HW Agent dále obsahuje funkce pro přípravu mikrořadiče na vzorkování a inicializaci periferií.

3.9.2 Popis objektu Comms Agent

Hlavní funkcí Comms Agent je správná interpretace příkazů odeslaných z počítače. Po přijetí příkazu z PC vyvolá Comms Agent odpovídající funkce ostatních objektů pro vykonání příkazu, jako je započetí vzorkování nebo nastavení vzorkovací frekvence. Dále obsahuje funkce pro překlad dat před odesláním zpět do počítače. Motivací ke vzniku tohoto objektu je nezávislost zbytku ELA firmwaru na aktuálním komunikačním protokolu, který definuje podobu příkazů a odesílaných dat.

3.9.3 Popis objektu Analyzer Agent

Účel objektu Analyzer Agent je nastavení a kontrola parametrů logického analyzátoru jako vzorkovací frekvence, počet vzorků a trigger podmínky. Nastavení odeslaná z počítače jsou přijaty objektem Comms Agent, který je předá objektu Analyzer Agent. Po přijetí nastavení zkontroluje Analyzer Agent, zda jsou v limitech logického analyzátoru. Objekt dále zařizuje zapnutí vzorkování s nastavenými parametry, odeslání navzorkovaných dat zpět do počítače pomocí Comms Agent a inicializaci zbylých objektů.

3.9.4 Pomocné funkce ELA firmwaru

Kromě objektů obsahuje ELA firmware pomocné funkce, které slouží k zařízení kompatibility s hardwarovými knihovnami psanými v jazyce C. Potřeba jsou pouze v případě, kdy funkce z hardwarových knihoven vyvolává funkce z objektu HW Agent. Toto nastává při přerušení, které vyvolá například čítač, nebo trigger. ELA firmware dále obsahuje hlavičkové soubory *ela.h* a *ela_configuration.h*, které obsahují některé základní definice společné pro celý ELA firmware a makra s hodnotami jako je maximální vzorkovací frekvence, velikost datového pole pro vzorky apod.

Kapitola 4
Použité komunikační protokoly

Komunikační protokol definuje způsob komunikace mezi zařízeními, jako jsou použité příkazy a v jaké podobě se odesílají data. Tento protokol musí být podporován v ELA firmwaru i v aplikaci PulseView a v projektu byly použity 2 varianty komunikačního protokolu. Na počátku návrhu logického analyzátoru byl využit komunikační protokol z projektu OLS, který je již v PulseView implementován. Později byl navržen vlastní protokol ELA, který umožnil přidání funkcí s vlastnostmi vhodnějšími pro navržený logický analyzátor. V této kapitole budou popsány poznatky z testování s protokolem OLS a samotný návrh protokolu ELA.

4.1 Testování komunikace pomocí protokolu OLS

Pro prvotní testování logického analyzátoru byl využit komunikační protokol z projektu OLS. Projekt OLS je open-source logický analyzátor založený na FPGA Xilinx Spartan 3E, který využívá jednoduchý komunikační protokol vytvořený pro komunikaci pomocí UART [11]. Výhodou tohoto protokolu je jednoduchost a již existující podpora aplikace PulseView, což v rámci testování urychlilo implementaci, ale projevily se i některé nevýhody.

Hlavní problém, který nastal při testování s OLS protokolem, byla nestabilní komunikace mezi mikrořadičem a aplikací PulseView v operačním systému Windows. Tato nestabilita se projevovala při odesílání zaznamenaných dat z logického analyzátoru do PulseView, kdy nebyla data vždy zobrazena a po několika spuštění vzorkování komunikace přestala fungovat úplně. Tento problém byl později vyřešen modifikací samotného PulseView, což je blíže popsáno v kapitole 5.2. Dále neexistuje příkaz na ukončení vzorkování, což může být problémové při nastavení trigger podmínky, ke které nikdy nedojde. V tomto případě by mikrořadič nikdy nezastavil vzorkování a neustále čekal na trigger podmínku. Nevýhodou je také odesílání informací o zařízení a odesílání zaznamenaného signálu, kdy zařízení, které data přijímá neví, jak velké množství dat bude posláno.

4.2 Návrh komunikačního protokolu ELA

Nedostatky testovaného protokolu OLS bylo motivací k vytvoření komunikačního protokolu ELA, který nedostatky vyřeší. Protokol je navržený pro komunikaci pomocí UART, kde jsou data odesílána po bajtech. Pro samotný návrh byly nejdříve definovány příkazy a jejich podoba. Příkazy byli navrženy tak, aby měli předem danou velikost. V případech kdy toto není možné, nese příkaz v sobě informaci o množství odesílaných dat. Po navržených příkazech byl naprogramován zdrojový kód, určený pro ELA firmware a PulseView, který zjednodušuje implementaci ELA protokolu.

4.2.1 Popis navržených příkazů

Pro ovládání mikrořadiče pomocí počítačové aplikace je nutné definovat příkazy, které spouští různé funkce nebo nastavují parametry. Příkazy byly navrženy pro komunikaci

UART, která odesílá data po bajtech a data větší než 1 bajt jsou odesílána po částech. Příkazy jsou dále pro přehlednost rozděleny do dvou skupin, kterými jsou jednoduché a komplexní. V tabulce 4.1 je seznam všech typů příkazů včetně jejich velikostí v bajtech

Příkaz	Podpříkaz	Množství dat [B]
Reset		1
Handshake		1
Start		1
Stop		1
Set	Samplerate	6
	Sample-Count	6
	Pin-Mode	6
	Pretrig-Count	6
Get	Samplerate	2
	Sample-Count	2
	Pin-Mode	4
	Pretrig-Count	2
	Metadata	2
	Sampled-Data	2
Report	Samplerate	6
	Sample-Count	6
	Pin Mode	6
	Pretrig-Count	6
	Metadata	≥ 13
	Sampled-Data	≥ 10

Tabulka 4.1. Seznam příkazu v komunikačním protokolu s velikostí v bajtech

4.2.2 Jednoduché příkazy

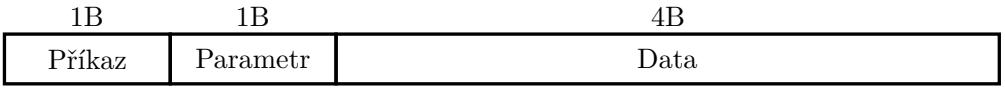
Tato skupina zahrnuje příkazy, které nevyžadují odesílání žádných dalších dat, postačuje pouze typ příkazu. Základními příkazy jsou Start a Stop, které slouží k zapnutí a vypnutí vzorkování logického analyzátoru. Dalším příkazem je Reset, který slouží pro resetování logického analyzátoru do výchozích hodnot. Po přijetí tohoto příkazu resetuje logický analyzátor parametry jako nastavená vzorkovací frekvence nebo nastavení trigger podmínek, které jsou uloženy ve firmware. Příkaz je odesílán aplikací PulseView před prvním připojení k logickému analyzátoru.

Posledním jednoduchým příkazem je Handshake, jehož účel je ověření kompatibility. Při prvním připojení posílá PulseView příkaz Handshake, na který logický analyzátor odpoví odesláním znaků `ELAPV1`. Po tom, co PulseView tyto znaky přijme, je připojení úspěšné a komunikace může pokračovat. To slouží k jednoznačné identifikaci zařízení a zabraňuje situaci, kdy se PulseView pokouší komunikovat se zařízením, které nepodporuje ELA protokol.

4.2.3 Komplexní příkazy

Tyto příkazy se neodesílají samostatně, ale i s dalšími daty. Po odeslání typu příkazu vždy následuje podpříkaz, který upřesňuje jaká data budou odeslána nebo jaká data jsou od zařízení požadována. Příkladem je příkaz Set, kde podpříkaz Samplerate označuje odesílání vzorkovací frekvence f_{vz} a podpříkaz Sample-Count označuje odeslání počtu

vzorků. Dalším příklad je příkaz Get, kde Samplerate značí žádost odeslání informace o aktuálně nastavené vzorkovací frekvenci f_{vz}. Na obrázku 4.1 je naznačena obecná podoba komplexního příkazu.

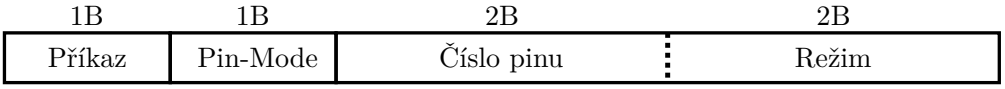


Obrázek 4.1. Základní podoba příkazů Set, Get a Report.

Příkaz Set slouží k nastavení logického analyzátoru. Podpříkazy zahrnují Samplerate Sample-Count, Pin-Mode a Pretrig-Count. Samplerate nastavuje požadovanou vzorkovací frekvenci f_{vz} a Sample-Count nastavuje počet vzorků, který logický analyzátor zaznamená. Pretrig-Count nastaví kolik z celkového počtu vzorků má být zaznamenáno před trigger podmínkou. Po všech 3 těchto podpříkazech následují 4 bajty dat s hodnotou na kterou se má parametr nastavit. Dalším podpříkazem je Pin-Mode sloužící k zapnutí nebo vypnutí vstupního kanálu mikrořadiče a také pro nastavení trigger podmínky. Po tomto podpříkazu následují 2 bajty dat s číslem značící nastavovaný vstup a další 2 bajty dat s číslem značící režim, do kterého je vstup nastavován. Těmito režimy jsou:

- Zapnutý vstup.
- Vypnutý vstup.
- Trigger na náběžnou hranu.
- Trigger na doběžnou hranu.
- Trigger na obě hranu.

Na obrázku 4.2 je podoba příkazu Set Pin-Mode s posílanými daty.



Obrázek 4.2. Podoba příkazů s parametrem Pin-Mode.

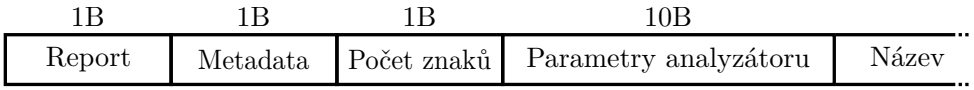
Příkaz Get slouží pro získání informace o aktuálních nastavení logického analyzátoru. Příkaz sdílí podpříkazy s příkazem Set a obsahuje navíc podpříkazy Metadata a Sampled-Data. Po podpříkazu jsou odeslána další data pouze u podpříkazu Pin-Mode, po kterém následuje číslo vstupu. Podpříkaz Metadata znační požadavek o základní informace o schopnostech logického analyzátoru, jako je počet vstupů a maximální vzorkovací frekvence. Podpříkaz Sampled-Data slouží pro opětovné odeslání zaznamenaných dat v logickém analyzátoru.

Příkaz Report značí odpověď logického analyzátoru na příkaz Get a sdílí s ním i podpříkazy. V případě podpříkazů Samplerate, Sample-Count, Pin-Mode a Pretrig-Count mají data po podpříkazu stejnou podobu, jako v případě příkazu Set. Podpříkazy Metadata a Sampled-Data mohou mít různé množství odesílaných dat. Data po příkazu Report Metadata obsahují informace o parametrech logického analyzátoru, kterými jsou:

- Maximální vzorkovací frekvence.
- Maximální počet vzorků.
- Počet kanálů.
- Název zařízení.

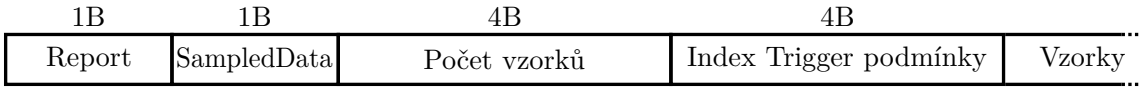
Název zařízení může mít různou délku a je proto kromě parametrů logického analyzátoru nejdříve odeslán počet znaků v názvu. Aby zařízení mělo informaci o počtu znaků co

nejdříve, je tato informace odeslána jako první, poté následují parametry logického analyzátoru a název je odeslán jako poslední. Podoba příkazu Report Metadata je znázorněna na obr. 4.3.



Obrázek 4.3. Odeslání parametrů logického analyzátoru příkazem Report Metadata.

Příkaz Report Sampled-Data je odeslán vždy před odesláním zaznamenaného signálu z logického analyzátoru. Po příkazu Report Sampled-Data následuje číslo, značící množství vzorků, které budou odeslány. Dále je odesláno také druhé číslo značící při kolikátém vzorku nastala trigger podmínka. Pozice trigger podmínky slouží pouze pro PulseView, aby toto místo mohlo být označeno při vizualizaci zaznamenaného signálu. Podoba podpříkazu Sampled-Data je na obr. 4.4.



Obrázek 4.4. Odeslání zaznamenaného signálu příkazem Report Sampled Data.

4.3 Příklad komunikace mezi logickým analyzátorem a PulseView

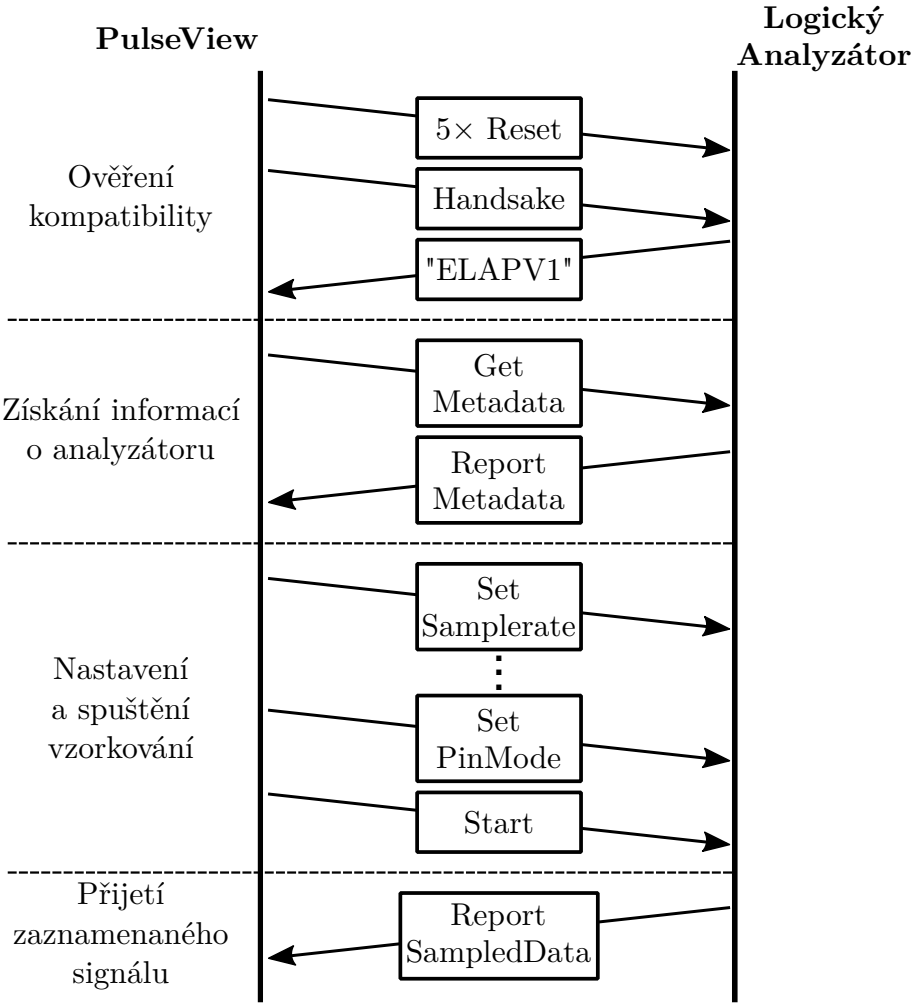
Před prvním spojením mezi analyzátorem a PulseView vybere uživatel virtuální sériový port, na kterém je zařízení připojeno. PulseView na tento port nejdříve 5× odešle příkaz Reset. Toto opakované odeslání příkazu Reset je kvůli možné situaci, kdy je logický analyzátor zaseknut ve stavu přijímání komplexního příkazu, například Set Samplerate, a očekává zbylá data. To může nastat při náhlým odpojením USB kabelu při odesílání komplexního příkazu z PulseView do logického analyzátoru. Pokud uživatel znovu připojí USB kabel a pokusí se připojit PulseView s logickým analyzátorem, mohou nastat 2 situace:

- V první situaci přijal logický analyzátor pouze typ příkazu, například Set, a očekává podpříkaz. Komunikační protokol je navržen tak, aby příkaz Reset nemohl být interpretován jako podpříkaz, logický analyzátor tedy tuto chybu pozná a při dalším přijetí příkazu Reset je řádně resetován.
- V druhé situaci přijal logický analyzátor typ příkazu i podpříkazu, například Set Samplerate, a očekává několik bajtů dat. V tomto případě nahradí opakovaný příkaz Reset očekávané bajty a poslední odeslaný Reset již analyzátor resetuje.

Po úspěšném resetování logického analyzátoru je odeslán příkaz Handshake, na který očekává odpověď znaky `ELAPV1`. Pokud zařízení odpoví správně, odesílá PulseView příkaz Get Metadata, na který odpoví analyzátor příkazem Report Metadata s informacemi o zařízení. Uživatel má po úspěšném připojení analyzátoru možnost nastavit parametry vzorkování a vzorkování spustit. Po spuštění odesílá PulseView všechny parametry pomocí příkazů Set a následně spustí vzorkování příkazem Start. Pokud uživatel zastaví v PulseView vzorkování, je odeslán příkaz Stop a analyzátor vzorkování ukončí.

V případě úspěšného ukončení vzorkování odesílá mikrořadič příkaz Report Sampled-Data, za kterým následuje počet vzorků a číslo s indexem vzorku, při kterém nastala trigger podmínka. Pokud trigger podmínka nebyla nastavena, je odeslán počet vzorků

a index vzorku, kdy nastala trigger podmínka, je nulový. Poté odešle mikrořadič navzorkovaná data do PulseView. Pro uskutečnění dalších vzorkování jsou již odesílány pouze příkazy Set pro nastavení nově zvolených parametrů vzorkování a Start pro odstartování nového vzorkování. Proces připojení Logického Analyzátoru k PulseView a zapnutí vzorkování je naznačen na obr. 4.5



Obrázek 4.5. Komunikace mezi logickým analyzátorem a PulseView.

4.4 Naprogramování komunikačního protokolu

Komunikační protokol je nutné integrovat do ELA firmwaru a také do aplikace PulseView. Integraci by bylo možné udělat pro každý program zvlášť, ale lze využít kompatibility obou programů s kódem psaným v programovacím jazyce C. Integrace je poté ulehčena vytvořením kódu společného pro oba programy, který zjednodušuje práci s komunikačním protokolem.

Hlavním prvkem, který byl využit pro práci s komunikačním protokolem, je struktura `elap_cmd_t`, do které lze uložit informace o typu příkazu, podpříkazu a data která jsou příkazem odesílána. Kód dále obsahuje funkce, které přeloží příkaz a data do správné formy pro odeslání pomocí komunikace UART. V opačném případě, pro přijetí příkazů, slouží funkce pro překlad přijatých bajtů do formy struktury `elap_cmd_t`, kterou lze již lépe zpracovat.

Na obr. 4.6 je ukázka kódu pro odeslání příkazu. Na začátku kódu je deklarace proměnných, kde `rx_buffer` je paměťové pole pro odesílané bajty, `command` je struktura pro odesílaný příkaz a `command_size` je jednoduchá číselná proměnná. Následně je struktura `command` nastavena na příkaz Set Pin-Mode s trigger podmínku na náběžnou hranu na kanálu 1. Poté je struktura `command` a paměťové pole `rx_buffer` předáno funkci `elap_cmd_to_packet()`, která příkaz přeloží do bajtů a ty uloží do paměťového pole. Funkce `elap_cmd_to_packet()` vrací do proměnné `command_size` počet bajtů, který byl do pole zapsán. Proměnná `command_size` a paměťové pole `rx_buffer` jsou předány funkci `send_buffer()`, která data odešle. V případě přijetí příkazu je proces opačný, kdy je do funkce `elap_packet_to_cmd()` předáno paměťové pole s přijatými bajty a funkce vrací strukturu typu `elap_cmd_t` s přeloženým příkazem.

```
// Deklarace proměnných
uint8_t rx_buffer[20];
elap_cmd_t command;
int command_size;

// Nastavení odesílaného příkazu
command.type = CMD_SET;
command.subtype = SUB_PIN_MODE;
command.data.pin_mode.number = 1;
command.data.pin_mode.mode = PM_TRIGGER_RISING;

// Překlad příkazu do bajtů
command_size = elap_cmd_to_packet(&command, rx_buffer, 0);

// Odeslání bajtů
send_buffer(rx_buffer, command_size);
```

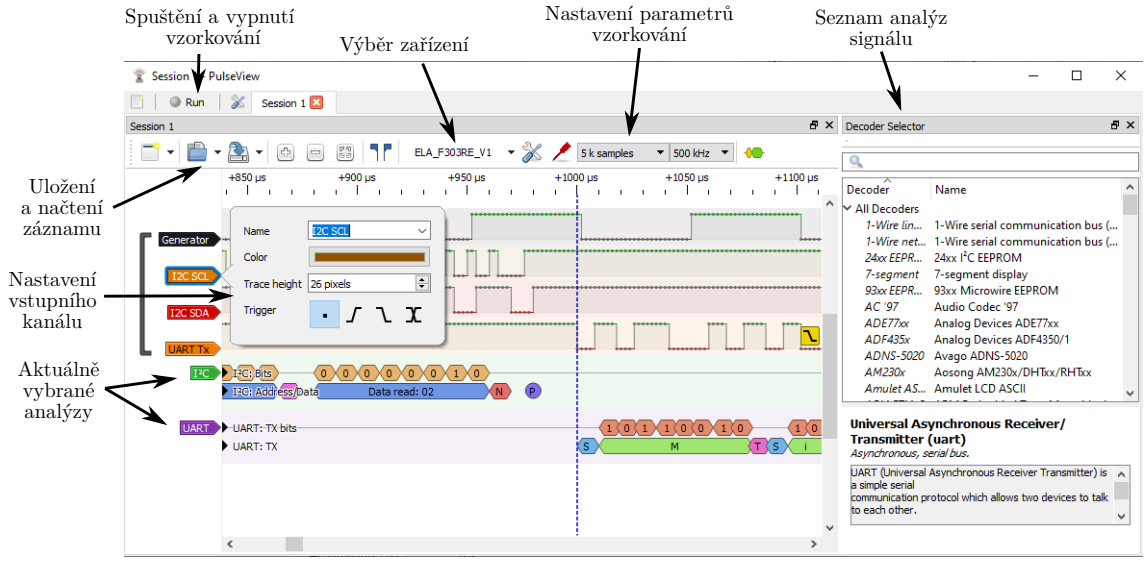
Obrázek 4.6. Příklad kódu pro odeslání příkazu.

Kapitola 5
Modifikace aplikace PulseView

Jak již bylo zmíněno v kapitole 2.3, pro přijetí a vizualizaci dat v počítači byla zvolena aplikace PulseView. Ta je součástí otevřeného projektu sigrok, jehož cílem je vytvořit sadu aplikací, které umožňují připojení k zařízením, jako jsou logické analyzátory a osciloskopy. Všechny součásti projektu, včetně zdrojových kódů jsou otevřené, veřejně dostupné a kdokoliv může aplikace upravit a případně přispět do projektu svým kódem [2]. Tato kapitola bude zaměřena na popis samotné aplikace PulseView a na modifikaci aplikace pro přidání podpory komunikačního protokolu ELA.

5.1 Podrobnější pohled na aplikaci PulseView

PulseView je multiplatformní počítačová aplikace, která poskytuje uživatelské rozhraní pro připojení k zařízením podporovaným projektem sigrok. Aplikace umožňuje nastavení parametrů zařízení, vizualizaci dat zaznamenaných zařízením a také analýzy digitálních komunikací. PulseView je založeno na knihovnách Qt a díky tomu je kompatibilní s několika operačními systémy včetně Linux, Windows apod. Stejně jako ostatní součásti projektu sigrok je i PulseView vyvíjeno jako otevřená aplikace a zdrojový kód je volně ke stažení [3]. Na obrázku 5.1 je ukázka podoby aplikace PulseView s popisem některých prvků.

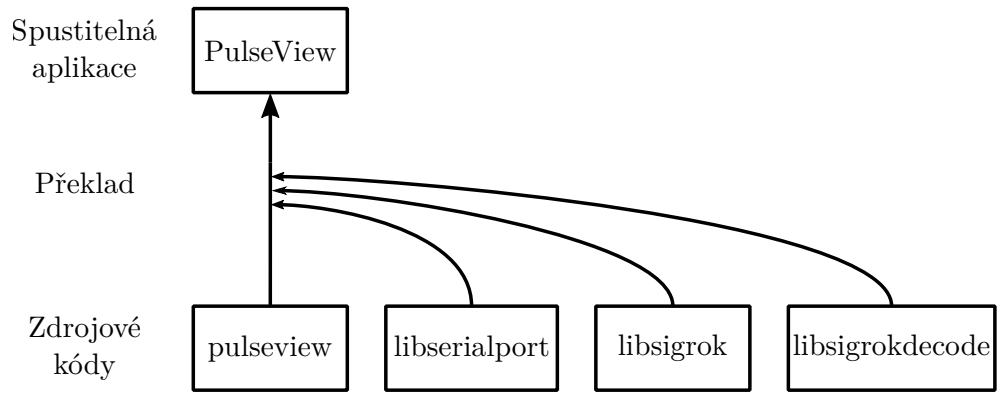


Obrázek 5.1. Aplikace PulseView s popisem některých prvků.

5.1.1 Součásti aplikace PulseView

Samotné PulseView je pouze grafické rozhraní, které využívá knihoven z projektu sigrok. Tyto knihovny jsou do aplikace přidány při kompilaci (přeložení) zdrojového kódu a dohromady tvoří jednu spustitelnou aplikaci. Kompilace zdrojového kódu s potřebnými

knihovnami je znázorněn na obr. 5.2. Všechny knihovny jsou psané v jazyce C a samotné PulseView je psáno v jazyce C++.



Obrázek 5.2. Znázornění překladu zdrojového kódu PulseView s potřebnými knihovnami.

Knihovna libserialport slouží pro komunikaci s virtuálními sériovými porty v počítači [12]. Knihovna libsigrokdecode umožňuje funkci dekódování a analýzu zaznamenaných digitálních komunikací. Samotné dekodéry pro konkrétní komunikaci jsou napsané v jazyce Python [13]. Knihovna libsigrok zařizuje kompatibilitu s podporovanými logickými analyzátory, osciloskopy apod. Pro každé zařízení umožňuje komunikaci, nastavení parametrů vzorkování, správné přijetí dat a další funkce [14].

5.2 Poznatky z testování s protokolem OLS

Na počátku projektu, byla komunikace mezi logickým analyzátorem a aplikací PulseView uskutečněna pomocí komunikačního protokolu OLS. Při tomto testování se objevil problém v komunikaci v operačním systému Windows, kdy PulseView nepřijalo vždy záznam z analyzátoru kompletně a data se nezobrazila. Při dalších pokusech o zapnutí vzorkování již komunikace nefungovala vůbec nebo pouze vyjímečně. Řešení problému nebylo nalezeno na internetu, ani po konzultaci s vývojáři projektu sigrok a bylo proto nutné najít příčinu a vytvořit originální řešení.

5.2.1 Hledání příčiny a řešení problému v komunikaci

První možnou příčinou, byla komunikace UART a převodník UART-USB, které logický analyzátor využívá pro komunikaci s počítačem. Byl proto vyzkoušen externí UART-USB převodník, založený na čipu FTDI. Pro připojení toho převodníku byla přepnuta periferie USART v mikrořadiči na piny nepropojené s převodníkem integrovaným na desce Nucleo. Na tyto piny byl připojen externí převodník, pomocí kterého probíhala všechna komunikace s počítačem. S externím převodníkem ovšem nastal stejný problém jako s převodníkem desky Nucleo a neprojevila se žádná změna.

Po vyloučení komunikace UART jako příčiny problému, byla komunikace otestována s aplikací sigrok-cli. Tato aplikace využívá stejně jako PulseView knihovny libsikgrok, ale neposkytuje uživatelské rozhraní a ovládá se pomocí příkazového řádku [15]. Spuštění a nastavení parametrů vzorkování proběhlo následujícím příkazem:

```
sigrok-cli --driver ols:conn=COM3 --config samplerate=10k \
           --output-format bits --channels 0-8 --time 100
```

Záznam je po přijetí z logického analyzátoru uložen do souboru, který lze načíst a vizualizovat v PulseView. S aplikací sigrok-cli byla komunikace spolehlivá a zaznamenaný signál byl vždy úspěšně přijat.

Využití stejných knihoven libsigrok pro připojení k zařízení aplikací PulseView i sigrok-cli ukazuje, že problém je nejspíše v samotné aplikaci PulseView nebo ve způsobu jakým knihovny používá. Hlavním rozdílem v použití knihoven libsigrok mezi aplikacemi je po dokončení záznamu. U aplikace sigrok-cli je komunikace s logickým analyzátorem uzavřena a aplikace je ukončena. V případě aplikace PulseView se komunikace po přijetí záznamu neukončuje a zůstává otevřena. Pro vyzkoušení možného vyřešení problému bylo tedy implementováno uzavření a opětovné otevření komunikace mezi PulseView a logickým analyzátorem před každým spuštění vzorkování.

Po prozkoumání zdrojového kódu PulseView a potřebných knihoven bylo určeno, že implementaci uzavření a opětovného otevření je nutné udělat přímo v knihovnách libsigrok v kódu pro samotné zařízení. V rámci testování byl upraven kód pro analyzátor OLS, kdy před každým spuštěním záznamu jsou vyvolány funkce, které uzavřou a opětovně otevřou komunikaci se zařízením. Tato oprava problém s komunikací ve Windows vyřešila a byla použita i při implementaci vlastního zařízení a komunikačního protokolu ELA do aplikace PulseView.

5.3 Přidání zařízení ELA do PulseView

Navržený komunikační protokol ELA a opravy zjištěné při testování s protokolem OLS je nutné implementovat do aplikace PulseView úpravou zdrojového kódu aplikace nebo knihoven, které aplikace využívá. Jak již bylo zmíněno v kapitole 5.1.1, podporu hardwarových zařízení zařizují v aplikaci PulseView knihovny libsigrok a pro implementaci nového zařízení je tedy nutné upravit pouze tyto knihovny.

5.3.1 Úprava zdrojového kódu knihoven libsigrok

Všechna zařízení, která knihovna libsikrok podporuje, mají vlastní zdrojový kód ve složce:

```
libsigrok/src/hardware
```

V této složce má každé zařízení svoji podsložku, která obsahuje zdrojový kód pro podporu zařízení. Ten je rozdělen do 3 základních souborů, kterými jsou

- *api.c*,
- *protocol.c*,
- *protocol.h*.

Soubor *api.c* obsahuje základní informace o schopnostech zařízení a také funkce pro první připojení, změnu parametrů vzorkování a zapnutí vzorkování. Soubory *protocol.c* a *protocol.h* obsahují pomocné funkce pro komunikaci a další funkce specifické pro konkrétní zařízení.

Pro přidání vlastního zařízení byla tedy vytvořena složka:

```
libsigrok/src/hardware/embedded-logic-analyzer
```

a v ní vytvořeny základní soubory *api.c*, *protocol.c* a *protocol.h*. Kromě těchto souborů byl dále navíc přidán kód ELA protokolu neboli *ela_protocol.c* a *ela_protocol.h*. Následně byl naprogramován zdrojový kód, ale aby bylo nové zařízení přidáno při kompilaci knihoven, je nutné upravit ještě soubory obsahující nastavení kompilace. Těmito soubory jsou *configure.ac* a *Makefile.am*, které jsou v kořenové složce knihoven. V souboru *configure.ac* byl přidán následující řádek:

```
SR_DRIVER([Embedded Logic Analyzer],
          [embedded-logic-analyzer],
          [serial_comm])
```

Do souboru *Makefile.am* byli přidány cesty k novým souborům se zdrojovým kódem:

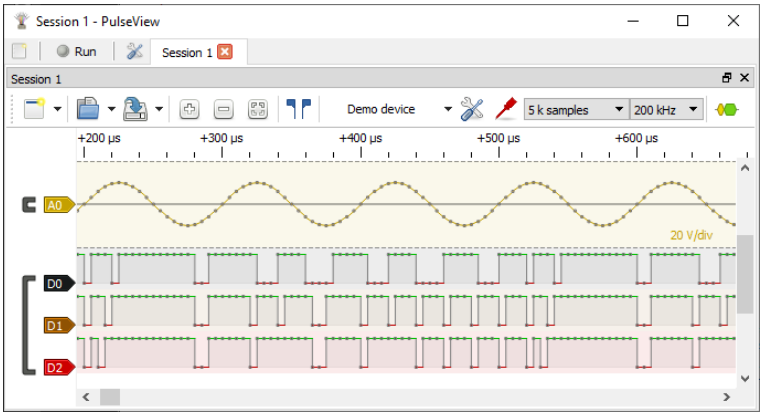
```
if HW_EMBEDDED_LOGIC_ANALYZER
src_libdrivers_la_SOURCES += \
   src/hardware/embedded-logic-analyzer/ela_protocol/ela_protocol.h \
   src/hardware/embedded-logic-analyzer/ela_protocol/ela_protocol.c \
   src/hardware/embedded-logic-analyzer/protocol.h \
   src/hardware/embedded-logic-analyzer/protocol.c \
   src/hardware/embedded-logic-analyzer/api.c
endif
```

5.3.2 Kompilace PulseView a potřebných knihoven

Projekt sigrok poskytuje balík nástrojů sigrok-util, který mimo jiné obsahuje skripty pro kompilaci PulseView pro různé operační systémy. Ve složkách se skripty jsou navíc textové soubory README, poskytující informace o potřebných nástrojích pro kompilaci a postup samotné kompilace [16]. V rámci projektu bylo zkompilováno PulseView pro operační systém Windows, ale samotná kompilace probíhala v operačním systému Linux. Kompilační skript stahuje automaticky z internetu nejnovější zdrojové kódy knihoven a PulseView a bylo ho proto nutné upravit, aby pro knihovny libsigrok byl použit upravený zdrojový kód s novým zařízením.

5.4 Posudek doplňkové funkce osciloskopu

Aplikace PulseView a knihovny libsigrok podporují kromě logických analyzátorů také osciloskopy. V knihovnách libsigrok v souboru *api.c* je možné nastavit také kombinované zařízení, které má schopnosti logického analyzátoru i osciloskopu. V PulseView jsou potom vizualizovány nejen digitální a ale i analogové kanály, což je ukázáno na obr. 5.3. Schopnosti osciloskopu by bylo nutné implementovat do ELA Firmwaru, ELA protocolu i do knihoven libsigrok. Libsigrok dále podporuje přidání nastavení pro přepínání režimu zařízení, čehož by mohlo být například využito pro přepnutí mezi režimy osciloskopu a logického analyzátoru, ale tato možnost nebyla vyzkoušena



Obrázek 5.3. Ukázka analogových a digitálních vstupů v PulseView.

Kapitola 6
Porovnání hotového logického analyzátoru s podobnými projekty

Dle zadání mají být parametry hotového logického analyzátoru ELA porovnány s logickými analyzátory založenými na modulu Arduino. Pro toto porovnání byly zvoleny 2 projekty nalezené na stránce Github. Oba tyto projekty byly otestovány pomocí modulu Arduino Mini Pro s mikrořadičem ATmega328P s napájecím napětí 5V a hodinovým signálem o frekvenci 16MHz. V této kapitole budou porovnány schopnosti zvolených logických analyzátorů co se týče vzorkovací frekvence, počtu vzorků a funkce trigger.

6.1 Logický analyzátor vytvořený uživatelem aster94

Projekt, dostupný z [17], zahrnuje firmware založený na knihovnách Arduino a také GUI aplikaci do počítače. Firmware podporuje moduly Arduino s mikrořadičem ATmega328P, modul Arduino Mega a také některé mikrořadiče STM32 a ESP8266. Firmware nevyužívá pravidelného vzorkování vstupního signálu, ale pouze pozoruje změnu na vstupních pinech zařízení a pokud změna nastane, je uložen čas. To znamená, že logický analyzátor nemá nastavitelnou vzorkovací frekvenci a maximální počet vzorků nahrazuje maximální počet změn, který se nastavuje ve zdrojovém kódu. Aplikace v počítači nepodporuje nastavení žádných parametrů, pouze odstartování záznamu. Dále aplikace neobsahuje analýzu signálu. Logický analyzátor má 6 vstupních kanálů.

6.2 Logický analyzátor vytvořený uživatelem gillham

Projekt, dostupný z [18], zahrnuje firmware založený na knihovnách Arduino a neposkytuje GUI aplikaci. Firmware využívá komunikačního protokolu OLS, což znamená, že lze logický analyzátor spojit i s aplikací PulseView. Firmware podporuje moduly Arduino s mikrořadiči ATmega168, ATmega328P, ATmega2560 a ATmega32U4. Maximální vzorkovací frekvence analyzátoru je 4MHz, má 6 vstupních kanálů a maximální počet vzorků pro mikrořadič ATmega328P je 1024. Zařízení také podporuje trigger podmínky, ty ale při testování nefungovaly příliš dobře, protože se náhodně měnil počet vzorků před trigger podmínkou.

6.3 Porovnání parametrů hotového analyzátoru ELA

Finální verze ELA logického analyzátoru a má následující paremetry:

- Vzorkovací frekvence nastavitelná až do 12 MHz.
- Nastavitelný počet vzorků nastavitelný až 60000.
- 8 digitálních vstupů.
- Jednokanálová trigger podmínka, nastavitelná na náběžnou, doběžnou nebo obě hrany.
- Nastavitelný počet vzorků před trigger podmínkou.
- Generátor obdélníkového signálu s frekvencí 10kHz.
- Možnost propojení s aplikací PulseView s analýzou komunikací.

Projekty založené na základních modulech Arduino jsou omezené hlavně pamětí mikrořadiče, což omezuje maximální počet vzorků. Použitý modul s mikrořadičem ATmega328P má paměť SRAM o velikosti 2kB [19], což je 32× méně než 64kB v mikrořadiči STM32F303RE [20]. Analyzátory obou projektů mají o 2 vstupní kanály méně a částečně fungující funkci trigger umožňuje pouze analyzátor, vytvořený uživatelem gillham.

Mikrořadič ATmega328P neobsahuje periferie pro rychlý přenos dat, jako je DMA v mikrořadiči STM32F303RE a v obou porovnávaných projektech je vzorkování řešeno programovou smyčkou. Toto může způsobit nepřesné vzorkování a omezuje rychlost vzorkování. Maximální dosažitelná frekvence vzorkování analyzátorem ELA je 3× větší, než analyzátorem z projektu od uživatele gillham. Nevýhodou analyzátoru od uživatele gillham je také použití protokolu OLS a projevují se problémy v komunikaci s aplikací PulseView ve Windows. Použití moderního mikrořadiče STM32F303RE tedy umožnilo vytvořit logický analyzátor s lepšími parametry oproti analyzátorům využívajícím Arduino Moduly.

Kapitola 7
Závěr

V rámci této bakalářské práce byl vytvořen logický analyzátor za použití mikrořadiče řady STM32. Tento logický analyzátor slouží pro výukové účely a byl navrhnut s cílem co nejjednoduššího obvodového řešení. Toho bylo dosaženo použitím mikrořadiče SMT32F303RE na desce Nucleo a použití počítačové aplikace pro vizualizaci dat z logického analyzátoru. Analyzátor je propojen s počítačem pomocí USB kabelu a pro vizualizaci byla využita existující aplikace PulseView. Tato aplikace umožňuje nastavení parametrů logického analyzátoru a také obsahuje analýzu digitálních komunikací.

Pro vytvoření logického analyzátoru byl nejdříve navrhnut firmware pro zvolený mikrořadič. Při vývoji tohoto firmwaru byly prozkoumány různé možnosti implementace funkcí logického analyzátoru a nejlepších výsledků bylo dosaženo použitím hardwarových periferií mikrořadiče. Kromě základních funkcí byla dále implementována funkce spouštěcí podmínky, včetně možnosti nastavit velikost záznamu před a po tom, co podmínka nastane. Parametry finální verze logického analyzátoru jsou následující:

- 8 vstupních kanálů.
- Nastavitelný počet vzorků až na 60000.
- Vzorkovací frekvence nastavitelná až na 12 MHz.
- Jednokanálová spouštěcí podmínka, nastavitelná na náběžnou, doběžnou nebo na obě hrany.
- Nastavitelná velikost záznamu před spouštěcí podmínkou.
- Generátor obdélníkového signálu s frekvencí 10kHz.

Propojení logického analyzátoru a počítačové aplikace PulseView bylo na počátku projektu uskutečněno pomocí komunikačního protokolu OLS, který je již aplikací podporován. Při testovaní s tímto protokolem se ovšem projevily problémy v komunikaci mezi logickým analyzátorem a aplikací PulseView v operačním systému Windows. Ukázalo se, že tento problém je zaviněn implementací použitého komunikačního protokolu OLS, což vedlo k navržení vlastního protokolu ELA. Po navržení protokolu ELA byl napsán zdrojový kód, který ulehčil implementaci protokolu do firmwaru mikrořadiče i do aplikace PulseView. Následně byl modifikován zdrojový kód aplikace PulseView, kde bylo přidáno nové zařízení, využívající navržený protokol ELA. Při modifikování aplikace PulseView byly využity poznatky z testování s protokolem OLS a bylo implementováno originální řešení pro vylepšení spolehlivosti komunikace mezi logickým analyzátorem a PulseView.

V práci byly splněny všechny body zadání a navržený logický analyzátor je připraveno pro použití ve výuce.

Literatura

[1] Jiří Hladík. *LEO: Little Embedded Oscilloscope (online)* . https://embedded.fel.cvut.cz/platformy/leo. (cit. 2020-08-13).

[2] sigrok. *sigrok wiki (online)*. 2020. https://sigrok.org/wiki/Main_Page. (cit. 2020-08-13).

[3] sigrok. *PulseView (online)*. 2020. https://sigrok.org/wiki/PulseView. (cit. 2020-08-13).

[4] STMicroelectronics. *AN2548: Using the STM32F0/F1/F3/Gx/Lx Series DMA controller (online)*. 2020. https://www.st.com/resource/en/application_note/cd00160362-using-the-stm32f0f1f3gxlx-series-dma-controller-stmicroelectronics.pdf. (cit. 2020-08-13).

[5] STMicroelectronics. *AN4776: General-purpose timer cookbook for STM32 microcontrollers (online)*. 2019. https://www.st.com/resource/en/application_note/dm00236305-generalpurpose-timer-cookbook-for-stm32-microcontrollers-stmicroelectronics.pdf. (cit. 2020-08-13).

[6] STMicroelectronics. *RM0316: STM32F303xB/C/D/E, STM32F303x6/8, STM32F328x8, STM32F358xC, STM32F398xE advanced ARM®-based MCUs (online)*. 2017. https://www.st.com/resource/en/reference_manual/dm00043574-stm32f303xbcde-stm32f303x68-stm32f328x8-stm32f358xc-stm32f398xe-advanced-armbased-mcus-stmicroelectronics.pdf. (cit. 2020-08-13).

[7] STMicroelectronics. *UM1786: Description of STM32F3 HAL and low-layer drivers*. 2020. https://www.st.com/resource/en/user_manual/dm00122016-description-of-stm32f3-hal-and-low-layer-drivers-stmicroelectronics.pdf. (cit. 2020-08-13).

[8] STMicroelectronics. *STM32CubeMx (online)*. https://www.st.com/en/development-tools/stm32cubemx.html. (cit. 2020-08-13).

[9] Microsoft. *Docs: Visual Studio Code (online)*. https://code.visualstudio.com/docs. (cit. 2020-08-13).

[10] PlatformIO. *Docs: PlatformIO (online)*. 2020. https://docs.platformio.org/en/latest/. (cit. 2020-08-13).

[11] Dangerous Prototypes. *Open Bench Logic Sniffer (online)*. 2015. http://dangerousprototypes.com/docs/Open_Bench_Logic_Sniffer. (cit. 2020-08-13).

[12] sigrok. *Libserialport (online)*. 2018. https://sigrok.org/wiki/Libserialport. (cit. 2020-08-13).

[13] sigrok. *Libsigrokdecode (online)*. 2018. https://sigrok.org/wiki/Libsigrokdecode. (cit. 2020-08-13).

[14] sigrok. *Libsigrok (online)*. 2018. https://sigrok.org/wiki/Libsigrok. (cit. 2020-08-13).

[15] sigrok. *Sigrok-cli (online)*. 2019.
https://sigrok.org/wiki/Sigrok-cli. (cit. 2020-08-13).

[16] sigrok. *sigrok-util (online)*.
https://sigrok.org/gitweb/?p=sigrok-util.git;a=summar. (cit. 2020-08-13).

[17] aster94. *Logic analyzer (online)*.
https://github.com/aster94/logic-analyzer. (cit. 2020-08-13).

[18] gillham. *Logic analyzer (online)*.
https://github.com/gillham/logic_analyzer. (cit. 2020-08-13).

[19] Atmel. *DATASHEET: ATmega328P (online)*. 2015.
http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. (cit. 2020-08-13).

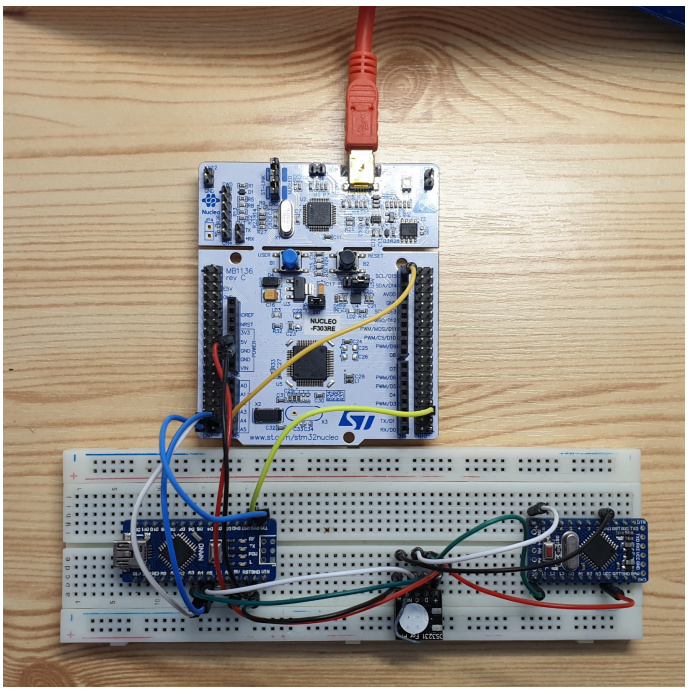
[20] *DS10362: STM32F303xD STM32F303xE (online)*. 2016.
https://www.st.com/resource/en/datasheet/stm32f303re.pdf. (cit. 2020-08-13).

Příloha A
Zkratky

LEO	Little Embedded Oscilloscope - Název projektu, ve kterém je navržen osciloskop založený na NUCLEO-F303RE.
ADC	Analog to Digital Converter - Analogově digitální převodník.
OLS	Openbench Logic Sniffer - Název projektu, ve kterém je navržen logický analyzátor založený na Spartan 3E FPGA.
ELA	Embedded Logic Analyzer - Název firmwaru a komunikačního protokolu navrženého v této práci.
DMA	Direct Memory Access - Periferie mikrořadiče pro rychlý přenos dat.
HW	Hardware.
Comms	Communications - Komunikace.
HAL	Hardrware Absraction Layer - Hardwarová abstrakční vrstva.
CLK	Clock - Hodinový signál.
UART	Universal Asynchronous Receiver-Transmitter - Druh asynchronní sériové komunikace.
I2C	Inter-Integrated Circuit - Druh synchronní sériové komunikace.
SPI	Serial Peripheral Interface - Druh synchronní sériové komunikace.
PC	Personal Computer - Osobní počítač.
PWM	Pulse Width Modulation - Pulsně šířková modulace.
USB	Universal Serial Bus - Universální sériová sběrnice, která se používá převážně pro propojení počítače s periferiemi.
RTC	Real Time Clock - Hodiny reálného času

Příloha B
Ukázka zapojení pro testování

Na obrázku B.1 je ukázka zapojení, kterým byli testovány schopnosti logického analyzátoru a aplikace PulseView. Deska NUCLEO-F303RE je spojeno s modulem Arduino (dole vlevo) pomocí nepájivého pole. Modul Arduino Nano je naprogramován aby každé 2 vteřiny zažádal pomocí I2C o data vteřiny a minutu z RTC modulu (dole uprostřed) a tyto data následně odeslal pomocí UART. Modul Arduino Mini Pro (dole vpravo) byl použit pro porovnání logických analyzátorů v kapitole 6. Ukázka analyzované komunikace v aplikaci PulseView je v příloze C.

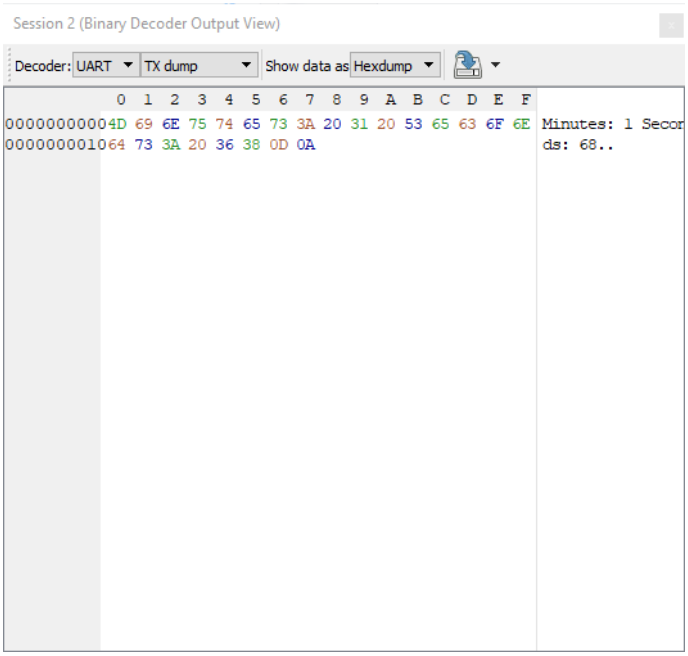


Obrázek B.1. Fotka zapojení pro testování logického analyzátoru.

Příloha C
Ukázka analyzované komunikace v aplikaci PulseView

Signál vizualizovaný v PulseView na obrázcích C.2 a C.3 byl zachycen pomocí zapojení v příloze B.

Obrázek C.2. Ukázka analýzy komunikace I2C.



Obrázek C.3. Ukázka dat zachycených z komunikace UART.

Příloha D
Obsah přiloženého CD

- `/ELA_Firmware` - Zdrojový kód ELA firmwaru a zkompilovaný binární soubor.
- `/ELA_Protocol` - Zdrojový kód ELA protokolu.
- `/PulseView` - Zkompilovaná aplikace PulseView, zdrojový kód upravených knihoven libsigrok a upravený skript pro kompilaci.
- `/Thesis` - Bakalářská práce ve formátu pdf.