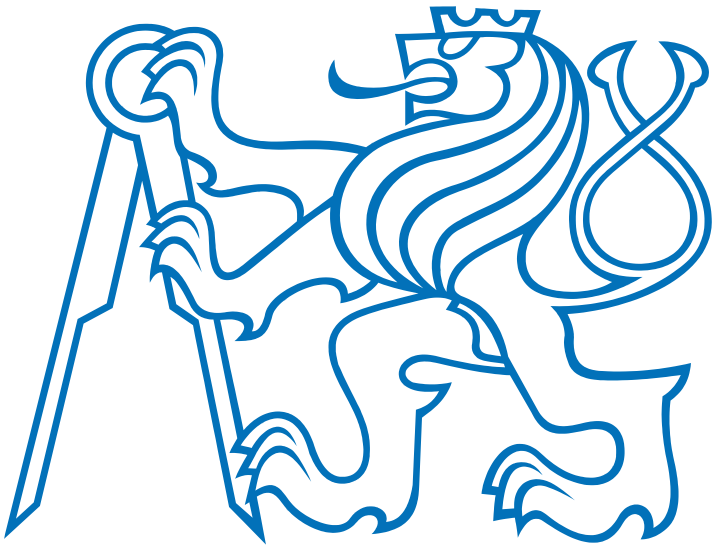
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra měření



DIPLOMOVÁ PRÁCE

Jednotka pro synchronizaci měřicích modulů

Bc. Petr Hájek

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

Nymburk, 2015

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2014-15**

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	**Bc. Petr Hájek**
Studijní program:	**Kybernetika a robotika**
Obor:	**Senzory a přístrojová technika**
Název tématu česky:	**Jednotka pro synchronizaci měřicích přístrojů**
Název tématu anglicky:	**Module for the Synchronization of Measurement Instruments**

Pokyny pro vypracování:

Navrhněte a realizujte jednotku pro synchronizaci měřicích přístrojů a bloků ovládání akčních členů s využitím rozhraní Ethernet a protokolu PTP-1588. Pro realizaci použijte desku s procesorem STM32F407, kterou případně doplňte dalšími potřebnými bloky. Jednotka bude generovat synchronizační a další hodinové impulsy potřebné pro synchronní činnost přístrojů, jejichž ovládání a sběr dat naměřených dat jednotka zajistí prostřednictvím sériového kanálu (UART).
Vytvořte demonstrační sestavu jednoduchého synchronního systému, který bude obsahovat měřicí i akční bloky. Vytvořte potřebné programové vybavení pro vestavné mikrořadiče i program pro nadřazené PC.

Seznam odborné literatury:

[1] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems , http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4579760
[2] RM0090 reference Manual, STMicroelectronics, 2014 www.st.com
[3] Yiu, J.: Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors,

Vedoucí diplomové práce:	doc. Ing. Jan Fischer, CSc.
Datum zadání diplomové práce:	1. prosince 2014
Platnost zadání do[1]:	31. srpna 2016

[redacted]
Doc. Ing. Jan Holub, Ph.D.
vedoucí katedry

L. S.

[redacted]
Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 1. 12. 2014

[1] Platnost zadání je omezena na dobu tří následujících semestrů.

Abstrakt

Tato práce se zabývá návrhem a realizací jednotky TriggerBox Lite pro synchronizaci v sítích Ethernet s využitím protokolu IEEE 1588v2 PTP. Jednotka funguje jako rozhraní, které běžným měřicím a řídicím jednotkám zprostředkovává synchronizaci a komunikaci se zařízeními v síti Ethernet.

TriggerBox Lite je nastavitelný přes Ethernet příkazy dle standardu SCPI a umožňuje generovat synchronní signály, zaznamenávat časy příchodu signálů a přeposílat zprávy z Ethernetové sítě do rozhraní UART, resp. RS 232 a naopak.

Práce ukazuje výsledné parametry zařízení v různých simulacích reálných aplikací a popisuje realizaci 2 demonstračních úloh TriggerBox Lite. Jednu úlohu se synchronními distribuovanými měřeními napětí. Druhou se synchronním otáčením BLDC motory.

Abstract

The aim of this work is to design and create unit TriggerBox Lite for synchronization in Ethernet networks with usage of IEEE 1588v2 PTP protocol. Unit works as interface which provides synchronization and Ethernet communication to ordinary measurement and control units.

TriggerBox Lite is configurable via Ethernet with usage of SCPI standard commands and is able to generate synchronizing signals, store information about time of generation external signals and resend messages from Ethernet network to UART or RS 232 interface and vice versa.

This work presents parameters of this device in several simulations of real applications and shows realization of 2 usage examples of the unit. One example is about synchronized distributed measurements of voltage. Second one is about synchronized control of BLDC motor rotation.

Čestné 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.

Tato práce vznikla v laboratoři měřicích systémů Katedry měření ČVUT FEL v Praze. Využívá výsledky výzkumného týmu ve složení Ing. J. Breuer, doc. Ing. J. Fischer, CSc., doc. Ing. J. Roztočil, CSc. a Ing. V. Vigner.

V Nymburce dne ....

Podpis autora práce

Poděkování

Rád bych tímto poděkoval vedoucímu práce doc. Ing. Janu Fischerovi, CSc. a doktorandovi Ing. Vojtěchu Vignerovi za trpělivost a velice podnětné konzultace, které umožnily vznik této práce. Dále děkuji všem blízkým, zvláště rodičům, sourozencům a přítelkyni Alžbětě za vytvoření podmínek pro studium a neúnavnou životní podporu.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Mnoho moderních průmyslových aplikací se vyznačuje potřebou synchronních měření a akčních zásahů na relativně velké ploše. Při realizaci velkých nákladných projektů je možné použít drahá specializovaná zařízení optimalizovaná pro konkrétní průmyslovou aplikaci. Naopak u menších projektů často zůstává požadavek na přesnou synchronizaci, ale také vyvstává potřeba nízkonákladové realizace celého projektu. Tyto low-cost distribuované systémy často využívají možností standardu sítí Ethernet, který díky své velké rozšířenosti nabízí levné komponenty. Sítě standardu Ethernet umožňují komunikaci pro distribuované systémy. Standard Ethernet však nedefinuje možnosti synchronizace.

Pro Ethernet proto existuje několik komunikačních protokolů, které různými způsoby synchronizaci umožňují. Tyto protokoly definují komunikaci s různými nároky na hardware a různými přesnostmi výsledné synchronizace. Tato diplomová práce se zabývá využitím protokolu IEEE 1588v2 PTP, který poskytuje synchronizaci s přesností v řádu mikrosekund až desítek nanosekund (záleží na konkrétní síti) a umožňuje fungovat i bez speciálního hardware nad rámec běžných sítí standardu Ethernet.

Běžně dostupné levné měřicí a akční moduly obvykle tento standard nepodporují. Je tedy třeba jim synchronizaci a komunikaci přes standard Ethernet zprostředkovat. Předmětem této práce je návrh a realizace jednotky TriggerBox Lite, která bude fungovat jako potřebné rozhraní mezi Ethernetem a zmíněnými moduly. Jednotka TriggerBox Lite se připojí do Ethernetové sítě, kde se synchronizuje s ostatními jednotkami v síti. Tuto synchronizaci následně zprostředkuje podřízeným modulům pomocí synchronních signálů. Dále umožní přeposílání zpráv mezi sítí standardu Ethernet a rozhraním UART (příp. RS 232), které měřicí a řídicí moduly obvykle obsahují. Vznikne tak relativně malá a levná jednotka, která může být základem mnoha distribuovaných systémů.

Vzhledem k tomu, že se bude jednat o finální produkt, je třeba analyzovat požadované funkcionality u konkrétních reálných aplikací tohoto zařízení. Z těchto důvodů budou k jednotce vytvořeny 2 demonstrační úlohy, které budou přímo simulovat konkrétní průmyslové situace. Jednou z demonstračních úloh bude synchronní měřicí systém, který bude ukazovat zejména možnosti jednotky ve sběru dat synchronních odměrů. Druhou úlohou bude systém s akčními členy, konkrétně motory, který bude simulovat využití např. na výrobní lince.

U jednotky bude kladen důraz zejména na její odladění, dokumentaci a finalizaci, aby jednotka byla přímo použitelná v potenciálních distribuovaných systémech. U dokumentace je kladen důraz také na měření parametrů jednotky TriggerBox Lite v různých simulacích použití (v prostředí malých sítí, velkých sítí,…). V rámci práce bude tak připraven plnohodnotný finalizovaný produkt.

2 Teoretický rozbor

Cílem této diplomové práce je navrhnout a realizovat jednotku TriggerBox Lite, která se bude synchronizovat v Ethernetové síti s ostatními jednotkami přes PTP protokol. Jednotka bude umožňovat připojení měřicích nebo řídicích modulů, kterým poskytne synchronizaci a zprostředkuje komunikaci přes Ethernet. Název TriggerBox vychází hlavní funkcionality zařízení – práce se signály (triggery). Přívlastek Lite je k názvu přidáván z toho důvodu, že podobná jednotka byla na fakultě vytvořena a na jejím vývoji jsem se v rámci bakalářské práce [1] podílel.

Jednalo se o jednotku SyncCore, která se také zabývala možnostmi synchronizace a komunikace v sítích Ethernet. Byla však určena jakou součást většího celku – jednotky SyncBox a nebyla tak navrhována jako samostatné zařízení. Pro TriggerBox Lite se jako základ použije hardware i software jednotky SyncCore, rozšíří se o nové funkcionality a budou provedeny optimalizace pro fungování jednotky samostatně. Dále budou připraveny demonstrační úlohy. Jednotka tak bude připravena pro plnohodnotné samostatné využití.

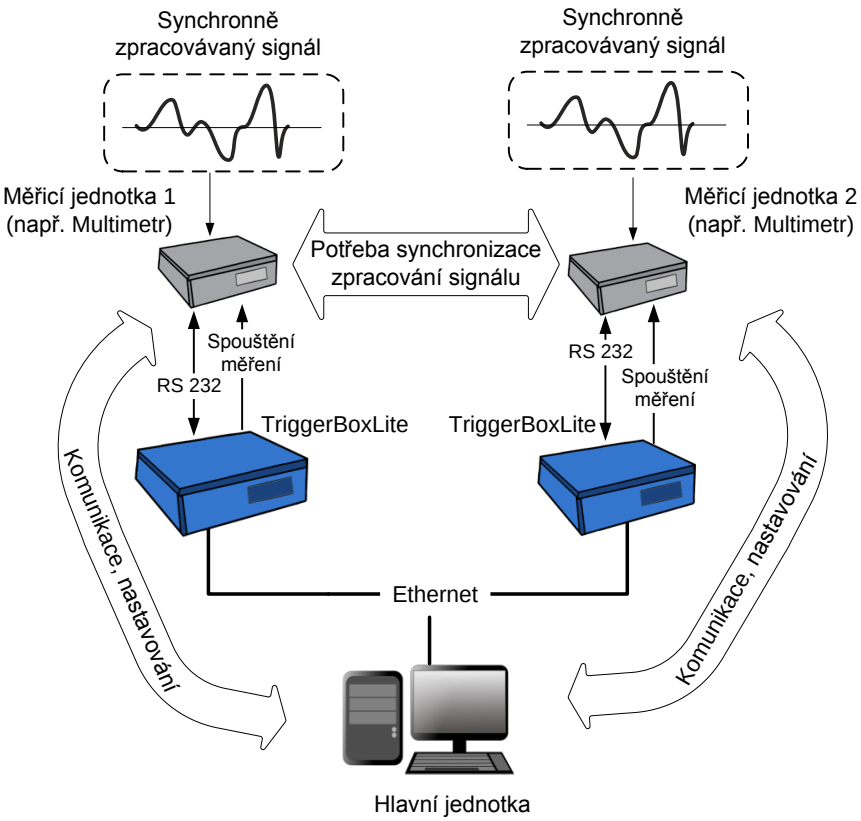
Jak již bylo zmíněno, tato práce vychází z funkčních principů jednotky SyncCore popsaných v rámci bakalářské práci [1]. Na některé analýzy a implementační postupy se proto bude tato diplomová práce odkazovat a nebude je popisovat znovu.

Základní požadavky na jednotku byly již zmíněny. Pro hlubší analýzu a pochopení je však potřeba analyzovat konkrétní příklady využití jednotky. Tyto příklady použití prezentuje následující kapitola.

2.1 Oblast použití TriggerBox Lite

TriggerBox Lite má umožnit synchronní měření nebo synchronní práci s akčními členy v distribuované síti. Požadavkem je využít obyčejné levné měřicí a akční jednotky bez podpory synchronizace. Požadavky jsou pro měřicí nebo řídicí distribuovaný systém z pohledu jednotky TriggerBox Lite relativně podobné. Pro názornost uvádí obr. 1 příklad použití TriggerBox Lite v měřicí distribuované síti.

Příklad znázorňuje měření napětí signálu na více místech ve stejný čas. Běžně může jít např. o měření fázového posuvu napětí v rozvodných elektrických sítích. V těchto sítích je třeba měřit fázový posuv napětí v bodech, které jsou od sebe vzdáleny v řádech kilometrů. Přes vzdálenost je třeba, aby odměry napětí proběhly ve stejný čas a bylo tak možné fázový posuv napětí vypočítat.



Obr. 1 – Synchronizovaný distribuovaný systém

Běžné měřicí jednotky (multimetry) umožňují spouštět měření triggerovacím impulsem a komunikují přes RS-232 rozhraní. Pro umožnění požadovaného synchronního měření s běžnými měřicími jednotkami je potřeba jednotka TriggerBox Lite, která se k měřicím modulům připojí a poskytne jim rozhraní pro synchronizaci a komunikaci, jak je třeba.

V příkladu na obr. 1 jsou TriggerBox Lite připojeny do sítě Ethernet, přes kterou synchronizují svůj čas s ostatními TriggerBox Lite a komunikují s nadřazenou PC jednotkou. PC aplikace ovládá celé distribuované měření. K měřicím modulům jsou TriggerBox Lite připojeny přes komunikační rozhraní RS232 a vodiče pro odesílání trigger impulsů spouštějících měření. V takovéto konfiguraci TriggerBox Lite funguje jako prostředník mezi měřicími moduly a nadřazeným PC. Umožňuje PC komunikovat (nastavovat a vyčítat měření) s měřicími moduly přeposíláním zpráv z Ethernetu na RS 232 a naopak. Zprostředkování komunikace se nazývá funkcí komunikačního Ethernet-RS 232 mostu (nebo také TCP-UART most a Ethernet-UART most).

Dále umožnuje vygenerovat na všech zasynchronizovaných TriggerBox Lite v síti trigger impulsy, kterými se spustí měření ve všech měřicích modulech najednou.

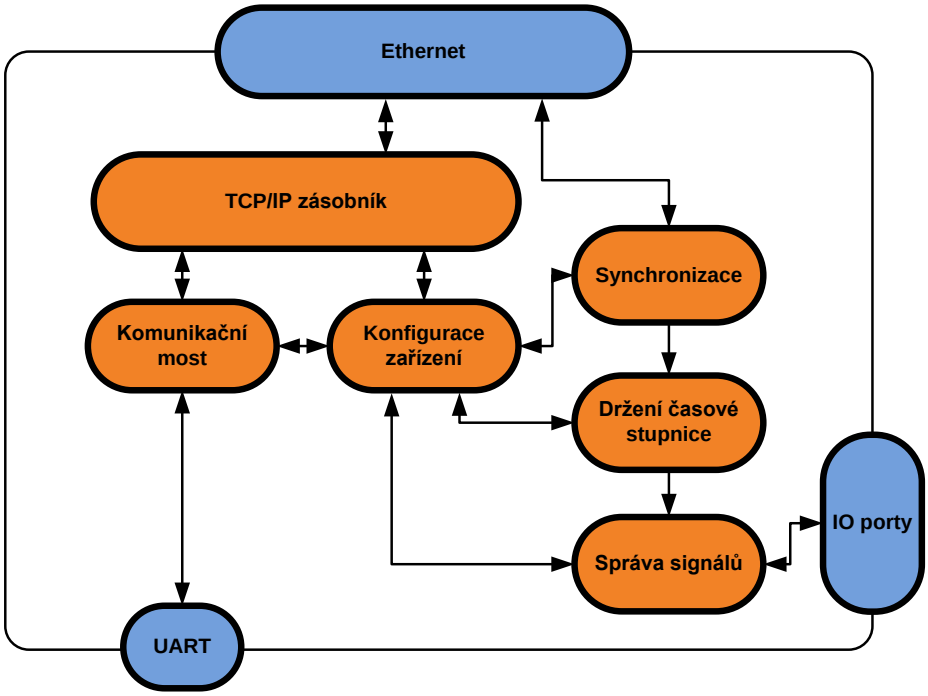
Sekvence měření tedy probíhá v pořadí:

1. PC nastaví měřicí moduly
2. PC nastaví TriggerBox Lite, aby v daném čase všechny vygenerovaly trigger impuls spouštějící měření na modulech
3. TriggerBox Lite v daném čase generují trigger impuls a multimetry provedou odměry
4. PC načte hodnoty z měřicích modulů
5. PC z hodnot vypočte fázový rozdíl signálu na jednotlivých měřených bodech.

2.2 Požadované funkce TriggerBox Lite

Z předchozí kapitoly je patrné, které funkce by měl TriggerBox Lite obsahovat. Jak již bylo zmíněno v úvodu, jednotka vychází ze zařízení SyncBox, konkrétně jeho jádra SyncCore[1], kterou se zabývala má bakalářská práce [1]. Z tohoto důvodu se zde funkce nebudou analyzovat do detailu. Detailnější informace jsou uvedeny v bakalářské práci [1], kapitola 2.2.

Blokové schéma na obr. 2 znázorňuje potřebné funkcionality TriggerBox Lite. Jednotka bude připojena do sítě přes rozhraní Ethernet, na které se napojují softwarové bloky TCP-IP zásobníku a synchronizace přes protokol IEEE 1588 – PTP.



Obr. 2 – Blokové schéma TriggerBox Lite[2]

Jedna z hlavních funkcí – komunikační most Ethernet-UART (resp. Ethernet-RS 232) bude realizována přeposíláním zpráv z UART na Ethernet a naopak (v Ethernetu se pak bude komunikovat přes TCP protokol).

[1] Blíže popisuje kapitola 4.3
[2] Schéma z velké části převzato z [1]

Další klíčovou funkcionalitou bude udržování časové stupnice, která bude synchronní ve všech jednotkách TriggerBox Lite zapojených do sítě. Na základě této časové stupnice bude TriggerBox Lite umožňovat generování signálů v daném čase:

- Generování náběžné a spádové hrany
- Generování PPS (pulse per second)
- Periodické generování hran

Dále také zaznamenávání času příchodu signálové hrany (náběžná nebo spádová).
Pro pohodlnou práci s jednotkou je také nutností konfigurace zařízení, která bude realizováná přes standard SCPI.

2.3 Porovnání s jednotkami na trhu

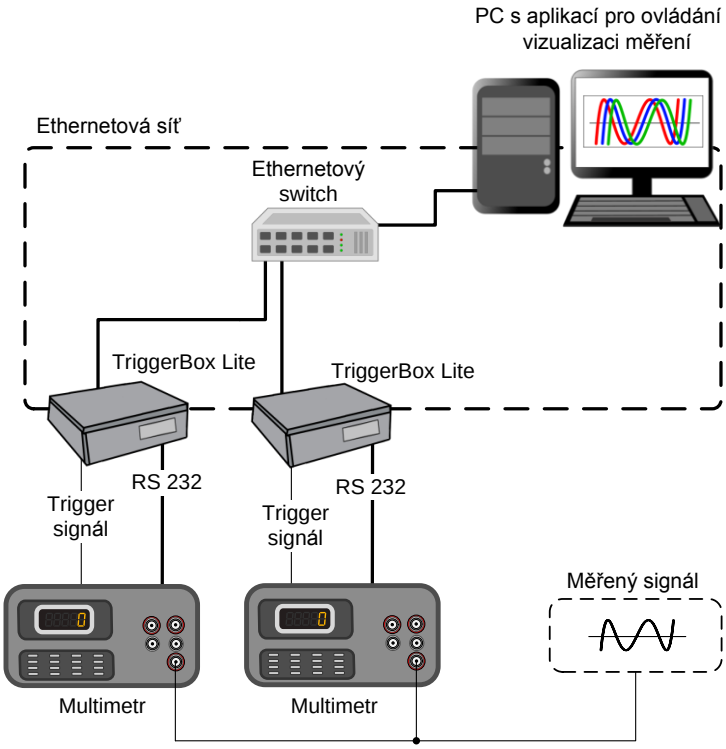
Na trhu existují jednotky, které částečně pokrývají funkcionality TriggerBox Lite. Jedná se o zařízení, která umožňují synchronizaci v Ethernetu přes IEEE 1588 PTP. Poskytují možnosti generování synchronních signálů a zachytávání času příchodu události. Zařízení však již neposkytují funkcionalitu komunikačního TCP-UART mostu, který je pro mnoho aplikací klíčový. Dále se často jedná o velká, komplexní a drahá zařízení.

TriggerBox Lite oproti dostupným zařízením nabízí možnosti malého a levného zařízení. Jednotka neimplementuje nadbytečné funkcionality, které by zvýšily cenu (např. napojení na GPS pro přesný čas,…). Podrobné porovnání s dostupnými jednotkami na trhu je uvedeno v bakalářské práci [1].

2.4 Výběr demonstračních úloh

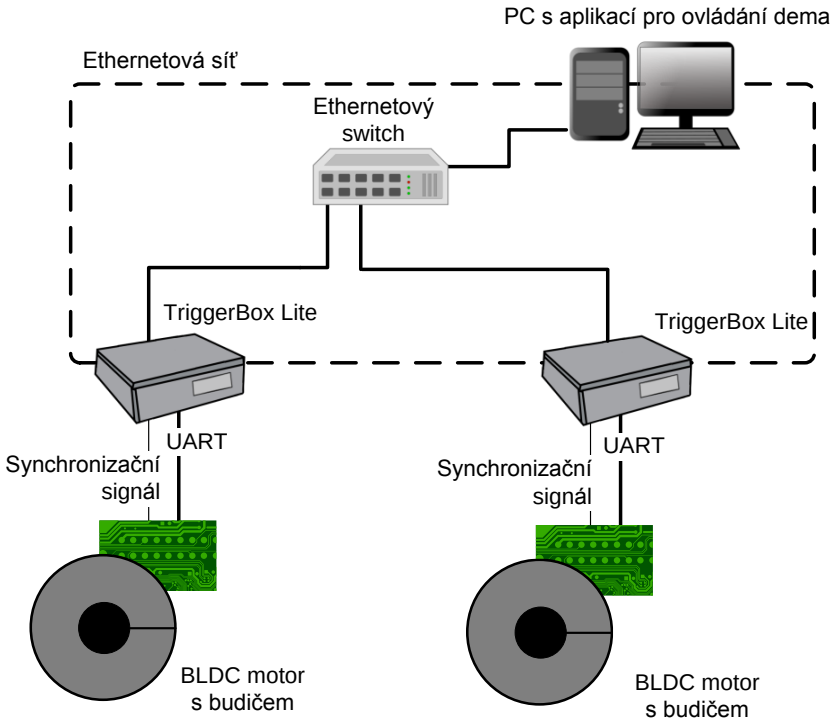
Jednotka TriggerBox Lite je určena pro použití v reálných situacích. Je proto nutné připravit demonstrační úlohy, které mají 2 úkoly. Ukázat funkcionality v praxi a ověřit, zda jednotka implementuje všechny funkcionality potřebné pro reálné aplikace. Při výběru demonstračních úloh byl tedy brán důraz na simulaci konkrétních reálných aplikací. Jedna demonstrační úloha (dále jen „demo“) se bude zabývat měřicím systémem a druhá řídicím.

Měřicí systém je nastíněn již v úvodních kapitolách práce. Jedná se o synchronní měření napětí. V systému budou multimetry měřící napětí na společném signálu. Tyto multimetry budou ovládány PC aplikací prostřednictvím jednotek TriggerBox Lite. Dávkové odměry budou vždy spouštěny ve stejný čas pomocí generování triggerovacího pulzu z TriggerBox Lite. Tato úloha přímo simuluje výše zmiňované měření fázového posuvu v rozvodných elektrických sítích. Blokové schéma ukazuje následující obrázek.



Obr. 3 – Základní schéma dema s voltmetry

Druhou úlohou je synchronní otáčení BLDC motory, které simuluje práci se servomotory na výrobních linkách v průmyslu. K jednotkám TriggerBox Lite budou připojeny BLDC motory, které se budou otáčet se stejnou fází. Schéma úlohy uvádí následující obrázek.



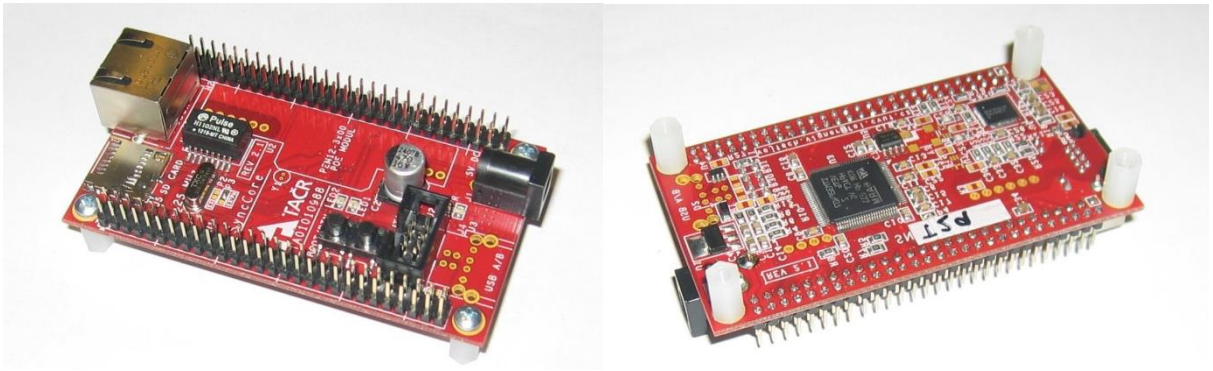
Obr. 4 – Základní schéma dema s BLDC motory

3 Použitý hardware

Pro diplomovou práci je potřeba hardware na 2 základní části práce: pro jednotku TriggerBox Lite a pro demonstrační úlohy. Následující kapitoly se zabývají analýzami a popisy tohoto hardware.

3.1 Hardware použitý pro TriggerBox Lite

Pro TriggerBox Lite bude použit prakticky stejný hardware jako pro jednotku SyncCore popsaný v bakalářské práci [1], liší se pouze v detailech. Základem celé jednotky je mikrokontroler STM32F407, ke kterému je připojena Ethernetová fyzická vrstva Texas Instruments PHYTER® DP83630. Popis použitých funkcí STM32F407 je uveden v [1]. Tato kapitola se zabývá pouze Texas Instruments PHYTER® z důvodu názornosti pro následující kapitoly.

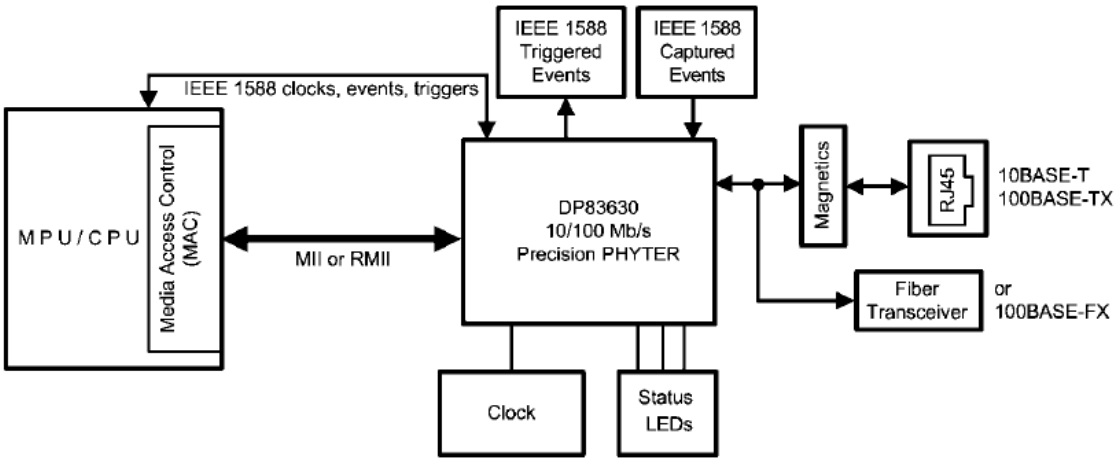


Obr. 5 – Deska pro TriggerBox Lite

3.1.1 Texas Instruments PHYTER®

Ethernetová fyzická vrstva (PHY) Precision PHYTER-IEEE® 1588 Precision Time Protocol Transceiver DP83630. Jedná se o jednotku, která kromě funkce Ethernetové fyzické vrstvy také podporuje pokročilou práci s PTP protokolem.

Tato jednotka je k mikrokontroleru (MCU) připojena přes MII nebo RMII rozhraní, přes které probíhá veškerá komunikace MCU s PHY – tzn. komunikace MCU-Ethernet i nastavování a vyčítání dat z PHY. Toto ukazuje blokové schéma na obr. 6. Jednotka PHY si udržuje interní čas, který je možné synchronizovat s ostatními PTP jednotkami v síti. PHY sama neprovádí synchronizační PTP komunikaci, o tu se primárně stará MCU, ale podporuje zpracování timestamps v PTP zprávách (předpřipravuje je pro MCU). Mikrokontroler přes PHY posílá synchronizační PTP zprávy a z vypočtené synchronizační odchylky reguluje interní čas PHY tak, aby byl synchronní s ostatními PTP jednotkami v síti (konkrétně synchronizuje čas PHY s jednotkou v roli PTP master).



Obr. 6 – Blokové schéma TI PHYTER ®[3]

Změna interního času se v PHY běžně neprovádí skokově (tzn. přenastavením časové informace), ale změnou frekvence interních hodin PHY (toto zprostředkovává NCO[4]). Pokud je např. interní čas jednotky zpožděný za PTP master v síti, zvýší se frekvence interních hodin, a tak dojde k postupnému dorovnání časové odchylky. Ke skokové změně interního času dochází pouze v případě, kdy je synchronizační odchylka příliš velká. Implementace obsluhy PTP regulace však záleží na konkrétním software MCU a ne na PHY.

Velkou předností této PHY je možnost práce se synchronními signály. Jednotka umožňuje generovat synchronní signál o dané frekvenci, kterou je možné následně použít jako vstupní hodinový signál pro MCU. Dále umožňuje generovat signály v daném čase (jednotlivé i periodické) a zaznamenávat čas příchodu signálové hrany. Pro zaznamenávání signálu poskytuje frontu o velikosti 8 záznamů.

PHY vyžaduje vstupní hodinový signál o frekvenci 25 MHz, který interně přenásobí na 125 MHz. Rozlišení jednotky je tedy 8 ns (1/125 MHz).

3.2 Hardware použitý pro demonstrační úlohy

K jednotce TriggerBox Lite mají být vytvořeny demonstrační úlohy, které umožní ukázat možnosti jednotky na případných fakultních výstavách. Jako demonstrační úlohy budou vytvořeny 2 konfigurace s TriggerBox Lite:

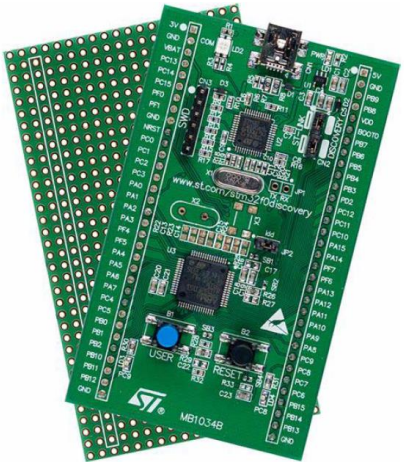
- Synchronní ovládání BLDC motorů
- Synchronní měření signálu voltmetry

Následující kapitoly popisují hardware potřebný k vytvoření těchto úloh.

[3] Převzato z [15]
[4] NCO – numerically controlled oscilator. Oscilátor, který umožňuje měnit svou frekvenci.

3.2.1 STM32F0

Pro obě demonstrační úlohy bude třeba implementovat jednotku s mikrokontrolerem. Jako MCU bude použit modul STM32F0, konkrétně ve vývojovém kitu STM32F0 Discovery.



Obr. 7 – Kit STM32F0 Discovery[5]

Model STMF0 byl vybrán kvůli nízké ceně oproti modelům vyšší kategorie STMF1 a výše, protože nároky na funkcionality nejsou vysoké. Zde je výpis hlavních funkcí modulu STM32F0, které jsou potřeba k implementaci demonstračních úloh:

- Procesorové jádro ARM® 32-bit Cortex®-M0, frekvence až 48 MHz
- Paměti
  - Flash 16 – 256 kB (na použitém kitu je 64 kB)
  - SRAM 4 – 32 kB (na použitém kitu je 8 kB)
- 5-ti kanálová DMA periferie
- 12-bit ADC převodník s dobou převodu 1 µs a 16-ti kanály
- 11 Čítačů (16-bit a 32-bit) umožňující funkcionality PWM, Output compare, Input capture, one-pulse mode output
  - Čítače je zároveň možné napojit na některé periferie. Pro dema je důležité napojení na ADC převodník
- UART periferie s napojením na DMA

Jediným omezením vybraného mikrokontroleru je skutečnost, že nepodporuje spouštění ADC převodu přímo externím pulsem. Tato funkcionalita je potřeba pro implementaci voltmetru v demonstrační úloze se synchronními měřeními napětí. Vliv tohoto omezení je však možné vyřešit. Tato problematika je uvedena v následující kapitole 6.1.

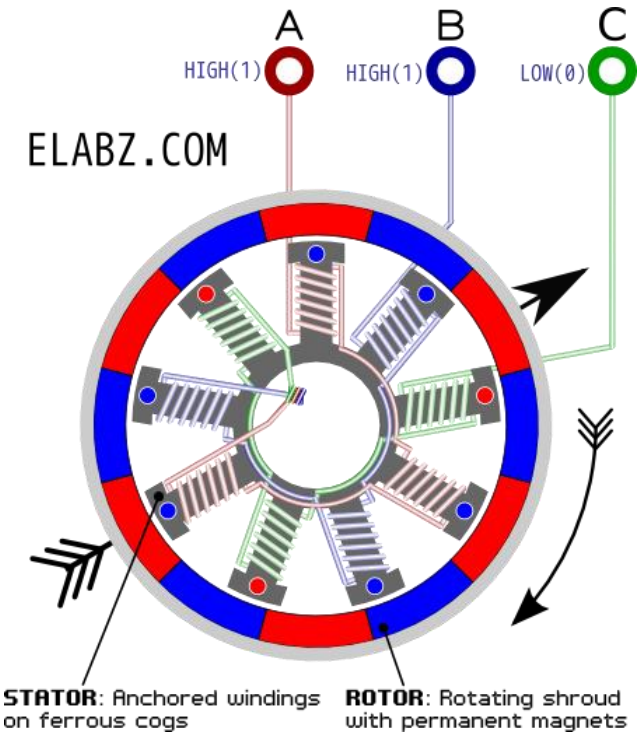
[5] Převzato z [14]

3.2.2 BLDC motory

BLDC motory jsou stejnosměrné elektrické motory, u nichž je komutace (přepínání) řídicích fází motoru prováděna elektronicky. Z tohoto principu také vychází název BLDC – Brushless DC motor, případné také BL motor nebo ECM (Electronically Commutated Motor).

Běžné komutátorové motory provádí komutaci pomocí lamel s kartáči. Kartáče jsou napojeny na řídicí napětí motoru a v každém úhlu natočení se dotýkají jiné lamely, která je napojena na vinutí motoru. Tímto způsobem je docíleno toho, aby elektromagnetické pole v rotorovém vinutí mělo vždy správný směr pro otáčení motoru. BLDC motory žádné kartáče neobsahují a potřebné přepínání vinutí řídí pomocí elektronického řízení. Je tedy potřeba detekovat úhel natočení motoru a z tohoto úhlu odvodit patřičné přepnutí proudu do konkrétního vinutí. U těchto motorů se rozlišují tzv. stavy natočení motoru. Jedná se o úhly natočení, dle kterých se provádí komutace.

Oproti běžným stejnosměrným motorům mají BLDC prohozený stator a rotor. U BLDC motorů je totiž vinutí na statoru a permanentní magnety na rotoru. Schématické znázornění BLDC motoru je na obr. 8.



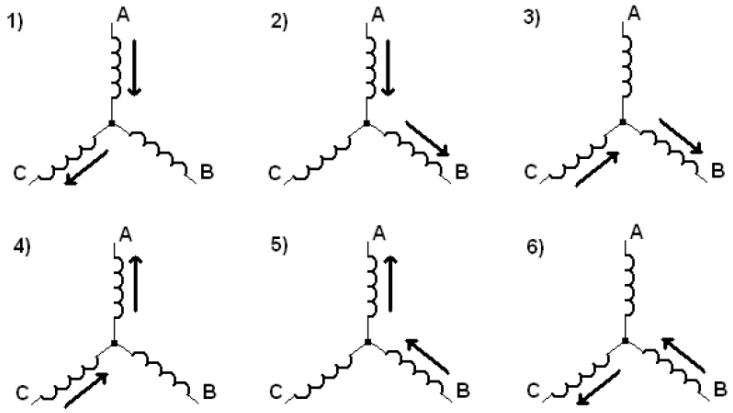
Obr. 8 – Schéma řízení a načítání pozice BLDC motoru[6]

Pro detekci natočení motoru se využívají 2 základní metody: senzorové a bezsenzorové měření. V senzorovém měření se informace o natočení získává z 3 Hallových sond, které z magnetického pole vyvolaného permanentními magnety rotoru, detekují, ve kterém stavu

[6] Převzato z [17]

motor je. U bezsenzorového měření žádné další senzory nejsou. Stav natočení motoru se snímá z velikosti indukovaného napětí nebo průběhu proudu cívkami v čase (podle [2]).

Statorové vinutí v motoru může být zapojeno do „hvězdy“ nebo „trojúhelníku“. V motorech použitých pro tuto práci se používá zapojení do hvězdy. Řízení komutace dle jednotlivých stavů je naznačeno na obr. 9. Vždy jsou aktivní právě 2 vinutí a třetí využito není.



Obr. 9 – Sekvence spínání vinutí motoru[7]

Jednotlivá vinutí bývají obvykle navinuta na více cívkách. Díky tomu je možné znásobit počet stavů na otáčku. Na obr. 8 je stejný typ vinutí jako na motorech používaných pro tuto práci. Jedná se o motory s 3-mi vinutími na 9-ti cívkách (jedno vinutí je vždy na 3 cívkách). Tato konfigurace poskytuje rozlišení 36 stavů na otáčku s tím, že sekvence řídících stavů z obr. 9 má 6 unikátních řídících stavů, které se v jedné otáčce motoru o 360° opakují 6krát.

K ovládání BLDC motorů není kvůli vysokým proudům do vinutí možné použít výstupy mikrokontroleru. Je proto potřeba použít budiče BLDC motorů, které vysoké proudy bez problémů poskytnou. Pro tuto práci byla použita deska budičů implementovaná Ing. Vavroušem v jeho bakalářské práci [2].

3.2.3 Ovládání BLDC motorů z práce Ing. Vavrouše [2]

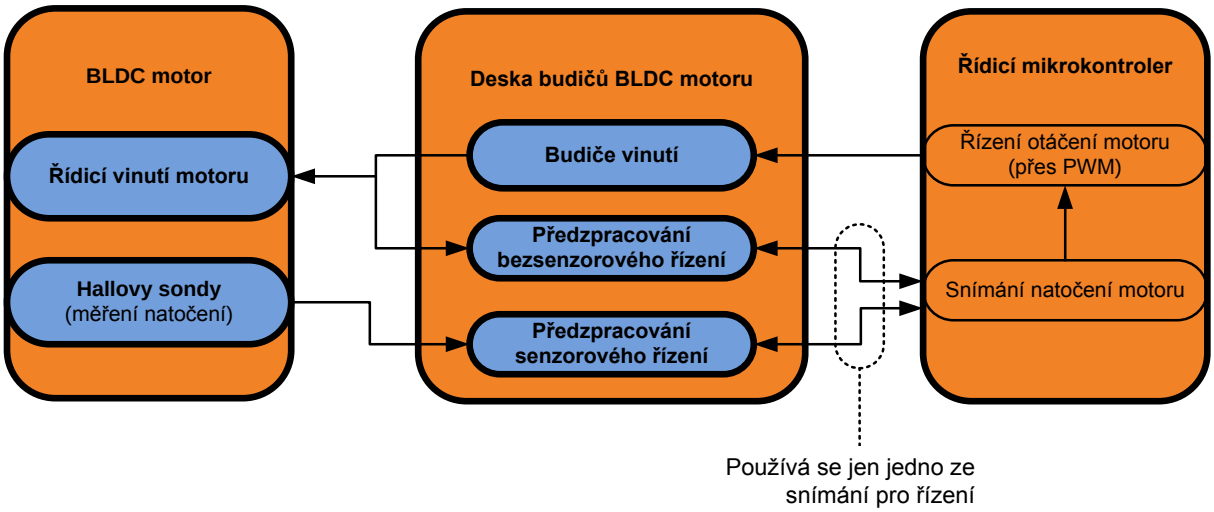
Bakalářská práce Ing. Vavrouše se kromě jiného zabývá implementací řízení BLDC motoru z CD mechaniky. V rámci této práce Ing. Vavrouš vytvořil desku budičů BLDC motorů, připravil jeden vzorový BLDC motor z CD mechaniky a implementoval jednoduchou knihovnu pro základní práci s BLDC motorem.

Deska budičů poskytuje všechno potřebné předzpracování pro ovládání BLDC motorů. Základní částí jsou výkonové budiče pro napájení vinutí motoru. Výkonové budiče umožnují vinutí dostat do všech 3 řídicích situací naznačených na obr. 9, tzn.: proud procházející jedním směrem, druhým směrem a odpojení vinutí. Motor je tak možné provozovat ve 3 základních

[7] Převzato z [2]

režimech: otáčení (řídicí stavy se mění), brždění (zůstává jeden řídicí stav), volnoběh (všechny 3 vinutí jsou odpojeny, tzn. stav vysoké impedance).

Deska dále poskytuje bloky pro předzpracování senzorového řízení. Jedná se o komparátory, které jsou napojeny na Hallovy sondy snímající polohu motoru. Komparátory prakticky digitalizují signál z Hallových sond, aby mikrokontroler bez potřeby A/D převodu detekoval konkrétní natočení BLDC motoru. Deska také poskytuje předzpracování bezsenzorového řízení, které je založeno na snímání proudu vinutími motoru. Programátor mikrokontroleru se tak při použití této desky může rozhodnout pro senzorové nebo bezsenzorové řízení. Blokové schéma desky budičů ukazuje následující obrázek.



Obr. 10 – Blokové schéma řízení BLDC motoru

Řízení motorů je v C knihovně od Ing. Vavrouše implementováno přes PWM výstupy čítače. Řízení BLDC motorů je v principu velice podobné řízení běžných stejnosměrných motorů – otáčky se řídí napětím a moment proudem. PWM výstupy pomocí střídy signálu řídí právě otáčky motoru. Při řízení motoru je třeba dát pozor na možné spálení motoru nebo budiče softwarovou chybou. Pokud by motor stál na místě (neotáčel se) a do jeho vinutí se pouštělo konstantní napětí (ne PWM), vinutí by nekladlo velký odpor a došlo by k poškození součástek. Z uvedených důvodů je nutné motor řídit PWM výstupy čítače, které se při software chybě nezastaví a neumožní motor spálit.

Detekci natočení motoru software MCU rozpoznává pomocí lookup tabulky, kde jsou uvedeny výstupy snímačů natočení (pro tuto diplomovou práci bude použito senzorové řízení, tzn. Hallovy senzory). Další lookup tabulka se používá pro řízení motoru, tzn. jakému stavu otáčení motoru odpovídá která kombinace řízení vinutí motoru.

Software Ing. Vavrouše dále umožňuje řízení motoru na rychlost a pozici (absolutní úhel natočení). Při řízení na pozici ovšem reguluje pouze na celé otáčky. Pro tuto diplomovou práci je nutné regulovat na konkrétní úhel, což bude třeba implementovat jiným způsobem než regulace na celé otáčky. Proto se regulace Ing. Vavrouše nebude moci použít.

4 Použitý software a standardy

Tato kapitola se zabývá synchronizačním protokolem IEEE 1588 PTP použitého v TriggerBox Lite a popisem zdrojových kódů pro SyncCore (základ TriggerBox Lite) převzatých z mé bakalářské práce [1].

4.1 PTP v Ethernetových sítích z pohledu využití TriggerBox Lite

Základem synchronizace jednotek TriggerBox Lite v Ethernetové síti je protokol IEEE 1588 PTP (konkrétně IEEE 1588 PTPv2). Jedná se o protokol, který specifikuje synchronizaci tzv. PTP master jednotky (jednotka jejíž čas se bere jako správný a hlavní v PTP síti) a PTP slave jednotek, které ladí svůj čas podle PTP mastera. Synchronizace probíhá na principu zasílání synchronizačních zpráv mezi jednotkami, ze kterých se dopočítává synchronizační odchylka. Tuto odchylku následně jednotky minimalizují.[8]

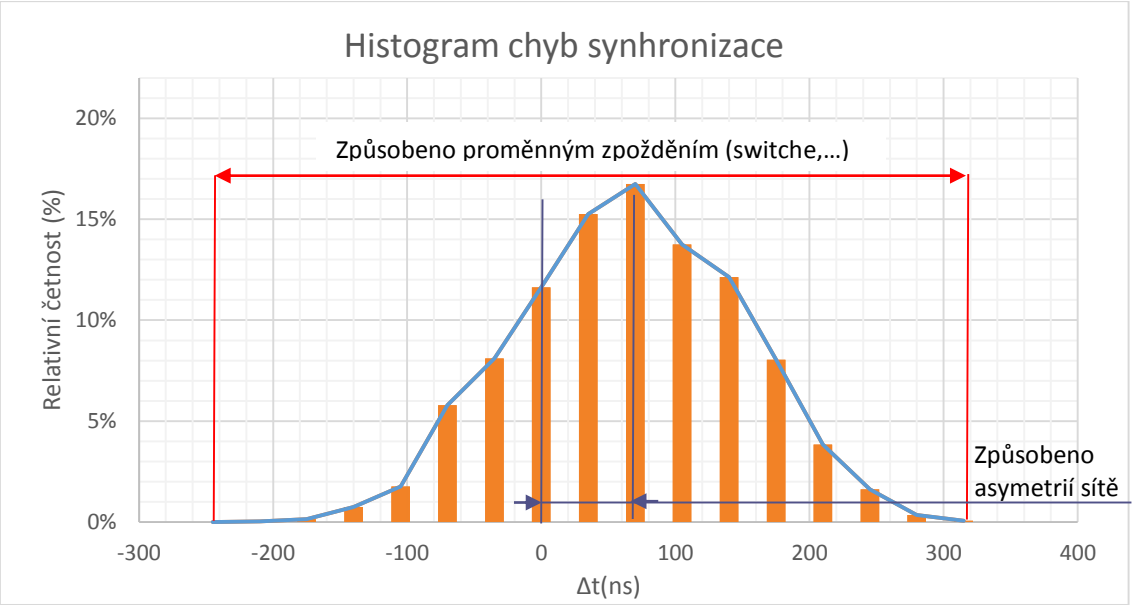
PTP protokol dosahuje přesností v řádech mikrosekund až desítek nanosekund (záleží na konkrétní síti) a velice dokáže dobře synchronizovat v sítích s homogenním a neproměnným zpožděním. Nehomogenitou je označována tzv. asymetrie sítě, která se projevuje tím, že mezi 2 uzly sítě není stejné komunikační zpoždění v jednom směru komunikace jako v druhém. Tato asymetrie způsobuje aditivní chybu synchronizace. Chyba je konstantní[9], ale ne v celé síti. Je konstantní vždy pouze mezi dvěma konkrétními uzly sítě. Dá se tedy říci, že každé dva uzly sítě mají jinou asymetrii než ostatní páry. Asymetrie sítě se tedy dá, ač relativně složitě, kompenzovat měřením sítě. PTP protokol s těmito kompenzacemi nepočítá, a proto se v této práci neřeší.

Druhým problémem PTP sítí je proměnné zpoždění, které je způsobeno zejména mezi-uzly v komunikaci (Ethernovými switchy). U těchto uzlů proměnné zpoždění způsobují interní fronty packetů. Zpoždění vždy záleží na konkrétním vytížení a stavu uzlu. Toto zpoždění zvětšuje směrodatnou odchylku synchronizačních chyb (tím pádem šířku histogramu synchronizačních chyb).

Projevy obou zmíněných problémů naznačuje následující graf.

[8] Blíže v mé bakalářské práci [1]

[9] Chyba se může měnit po restartu sítě nebo po změně konfigurace sítě (přidání nebo odebrání jednotek, změna zátěže sítě,…). Pokud však ke změně nedojde, zůstává chyba prakticky konstantní.

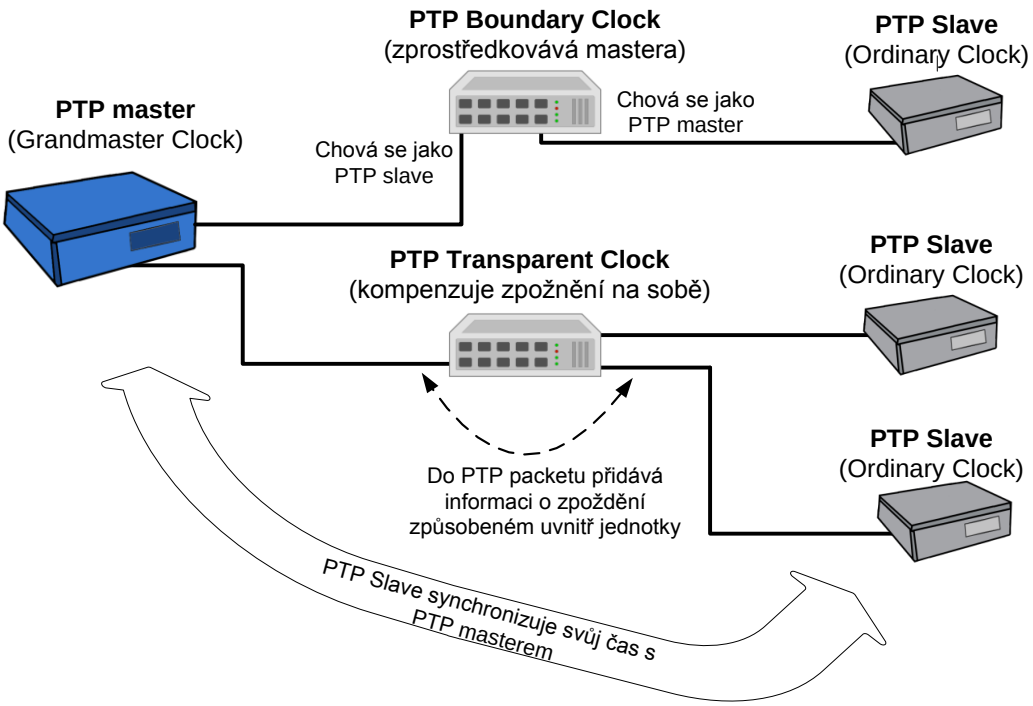


Obr. 11 – Příklad histogramu synchronizačních chyb PTP

Pro odstranění proměnného zpoždění se používá několik mechanizmů. V této práci budou uvedeny jen základní.

První metodou je použití jednotky typu PTP Boundary Clock. Jedná se o jednotku, která je na jedné straně zasynchronizována s PTP masterem, tzn. chová se jako PTP slave, který synchronizuje svůj čas s PTP masterem. Na Druhé straně se chová jako PTP master, na který se synchronizují ostatní připojené jednotky v roli PTP slave. Tímto způsobem ulehčí PTP masterovi komunikaci a sníží počet komunikačních uzlů, přes které by musely synchronizační zprávy mezi PTP master a PTP slave jednotkami putovat. PTP Boundary Clock musí být jednotka s přesnými interními hodinami, aby nezpůsobovala místo zpřesnění spíše větší synchronizační chyby. Proto jsou tyto jednotky relativně drahé. Uvedený princip je znázorněný na obr. 12.

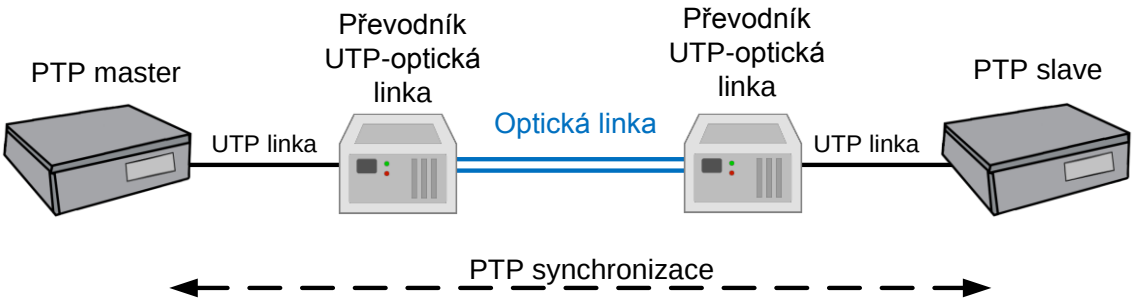
Druhou možností je použití PTP Transparent clock. Zde se jedná o uzel, který měří zpoždění způsobené přímo jeho interní frontou packetů. Tuto informaci následné přidává do těchto PTP packetů a umožňuje tak snížit chybu způsobenou proměnlivým zpožděním. Oproti PTP Boundary Clock se může jednat o jednotku s méně přesnými hodinami.



Obr. 12 – Základní jednotky v Ethernetové síti s PTP protokolem

Pro distribuované systémy, ve kterých je kladen velký důraz na nízkou cenu se dá obejít bez PTP Boundary Clock i PTP Transparent Clock a využijí se obyčejné switche. Jedná se o případ této diplomové práce, která je směřována hlavně na low-cost systémy.

Jednotka TriggerBox Lite je určena i pro využití v synchronních měřeních na větších vzdálenostech. Zde se ovšem naráží na limit možností použití obyčejného UTP[10] kabelu a obyčejných Ethernetových switchů. Maximální běžně uváděný dosah Ethernetové komunikace přes UTP kabely je 100 m. Toto omezení není způsobené tolik ztrátami na vedení jako omezením rychlosti šíření signálu elektrickým vedením. Sto metrů je ovšem pro některé aplikace TriggerBox Lite nedostačující, proto je třeba zohlednit také použití optických kabelů. Optické kabely díky využití světla dosahují vzdálenosti komunikace do 40 km. Kromě vzdálenosti také poskytují např. větší šířku pásma a možnosti odolnosti vůči rušení. Tyto vlastnosti pro TriggerBox Lite nejsou ale tolik důležité.



Obr. 13 – Optická linka v Ethernetové síti s PTP protokolem

[10] UTP – Unshielded Twisted Pair. Kroucená dvojlinka, která funguje jako základní medium pro Ethernetové sítě.

Nevýhodou použití optických kabelů v PTP síti je nutnost použití převodníků optická linka-UTP kabel. Tyto převodníky se totiž chovají jako další uzly s proměnným zpožděním na synchronizační cestě, jak ukazuje obr. 13. Použití optického kabelu proto do synchronizace vnáší chybu proměnlivého zpoždění, a proto by mělo být využíváno jen při potřebě komunikace na velkou vzdálenost.

4.2 Knihovny použité v TriggerBox Lite

Jednotka využívá několik volně dostupných knihoven. Pro komplexní nadhled nad touto diplomovou prací je třeba zmínit jejich seznam:

- FreeRTOS [3] – jednoduchý operační systém určený pro mikrokontrolery.
    - RealTime operační systém, který umožňuje držet kontrolu nad tím, který proces bude kdy spuštěn. *Jedná jeden z nejjednodušších RTOS, který obsahuje základní funkce spojené s plánováním a dále podporu mutexů, semaforů, front a softwarových časovačů.*(převzato z [1])
- lwIP [4] – „lightweight TCP/IP stack“ implementace TCP/IP zásobníku určená zejména pro mikrokontrolery
    - pro TriggerBox Lite pomáhá implementovat TCP server pro TCP-UART most a SCPI rozhraní. Obecně se však jedná o plnohodnotnou implementaci TPC/IP protokolu, takže velká část funkcionalit není využita.
- PTPd [5] – implementace PTP daemona pro IEEE 1588v2 PTP protokol
- Scpi parser [6] – knihovna pro zpracování SCPI příkazů, která byla implementována v rámci katedry měření Ing. Janem Breuerem

Bližší detaily v [1].

4.3 Hotový základ SyncBox, SyncCore

Jednotka TriggerBox Lite vychází ze zařízení, kterým se zabývala moje bakalářská práce [1] – SyncBox. Jednalo se o komplexní zařízení využívající synchronizace přes Ethernetový protokol PTP a poskytující funkcionality spojené se synchronizací. Jádrem tohoto zařízení byla jednotka SyncCore, jejíž firmware byl v rámci bakalářské práce [1] implementován.

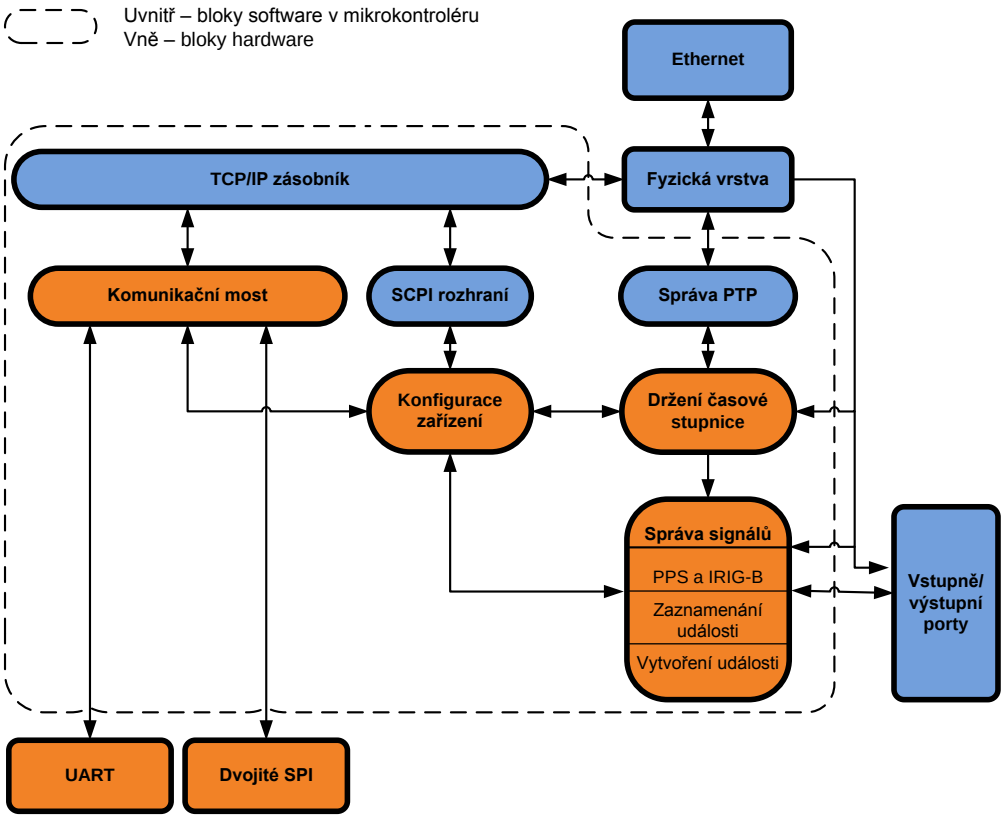
Obr. 14 – SyncBox

Funkcionality SyncBoxu ukazuje následující seznam

- SyncCore (jádro zařízení)
  - Generování signálů (událostí) v daném čase
    - Generování jednorázových nebo periodických událostí
    - Generování synchronizačního signálu IRIG-B[11]
  - Zaznamenání času příchodu signálové hrany
  - Komunikační most
    - Ethernet-UART (přesněji Ethernet-RS 232)
    - Ethernet-SPI
  - Nastavování přes SCPI rozhraní upravené pro Ethernet
- Ostatní jednotky SyncBoxu
  - Možnost synchronizace hodin dle GPS
  - Generování signálů o konkrétních frekvencích pomocí PLL obvodů
  - Rozšířené možnosti komunikace a nastavování přes USB
  - Display

Blokové schéma jednotky SyncCore je uvedeno na následujícím obrázku.

[11] IRIG-B je synchronizační signál, který v sobě nese informaci o aktuálním čase. Používá se v energetice.



Obr. 15 – Blokové schéma SyncCore[12]

Pro TriggerBox Lite se použije firmware a hardware základ SyncCore, kde budou odebrány nepotřebné funkcionality – Ethernet-SPI most a generování IRIG-B. Následně budou doplněny nové funkcionality specifické pro použití TriggerBox Lite.

4.4 Podrobnější popis principů bloků SyncCore

Tato kapitola popisuje základ funkčních principů jednotlivých bloků SyncCore, které byly implementovány v rámci bakalářské práce. Podrobný popis je uveden v [1]. Všechny uvedené bloky jsou identické i v TriggerBox Lite. Dá se tedy korektně mluvit o popisu hotových bloků TriggerBox Lite.

4.4.1 Udržování časové stupnice

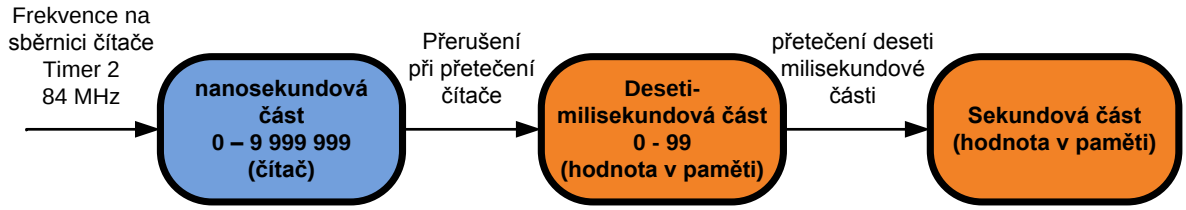
V jednotce SyncCore se informace o systémovém čase udržuje na dvou místech – v Ethernet PHY a čítačích mikrokontroleru. Ethernet PHY funguje jako primární zdroj systémového času pro jednotku, protože přes PTP se zde synchronizuje systémový čas s ostatními PTP jednotkami v síti. Konkrétně SyncCore v roli PTP slave synchronizuje svůj čas s PTP master přes změnu frekvence interních hodin celé jednotky. Pouze při velké synchronizační odchylce provede skokovou změnu hodnoty systémového času (tato situace nastává prakticky jen při zapojení do sítě nebo po chybě sítě).

[12] Převzat o z [1]

Udržování systémového času je implementováno i v čítačích mikrokontroléru z důvodu práce se signály (generování a zachytávání času příchodu události) a potřeby vyčítání systémového času bez režie na komunikaci mikrokontréru s Ethernet PHY přes MII rozhraní.[13] Systémový čas udržovaný v MCU se synchronizuje s Ethernet PHY (princip synchronizace je popsaný níže). Princip udržování systémového času v MCU je zobrazený na obr. 17. Z důvodu rozlišení se čas udržuje ve 3 blocích:

- Sekundová část – počet sekund v rámci systémového času se udržuje v paměti MCU
- Subsekundová část se dělí na 2 části
    - Deseti-milisekundová část - počet deseti-milisekundových časových bloků v sekundě se udržuje v paměti MCU
    - Nanosekundová část – počet nanosekund v deseti-milisekundovém časovém bloku se udržuje v 32bitovém MCU čítači. Hodnota je inkrementována každou periodu frekvence na sběrnici čítače v MCU

Sekundová a deseti-milisekundová část jsou inkrementovány v přerušení MCU čítače vyvolaném při přetečení čítače (čítač přeteče každých 10 ms).

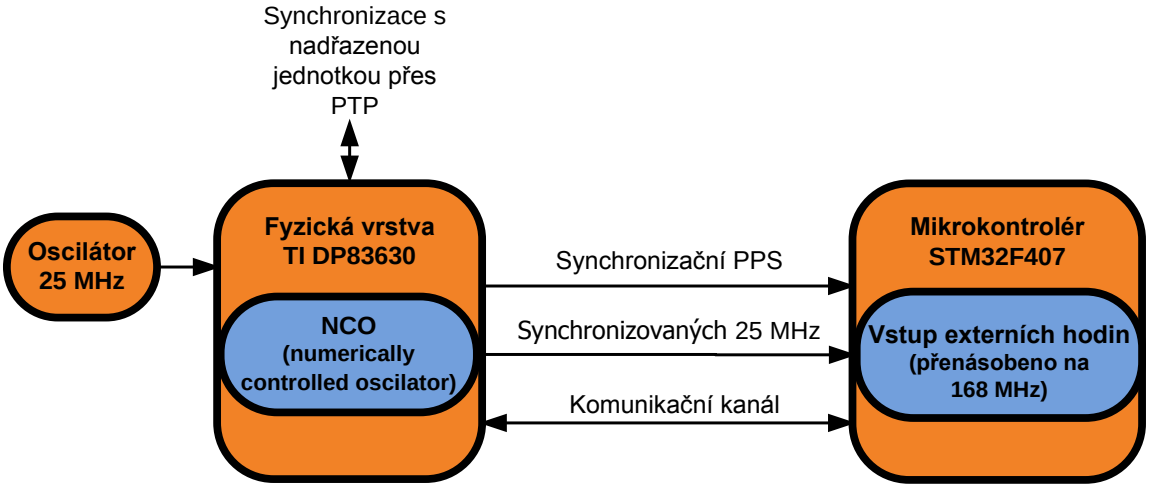


Obr. 16 – Schéma udržování časové stupnice v mikroprocesoru[14]

Synchronizace s Ethernet PHY je založena na dvou principech. Prvním principem je fakt, že MCU bere jako interní hodinový vstup výstupní signál z Ethernet PHY o frekvenci 25 MHz. Díky tomu se frekvence interních hodin MCU a Ethernet PHY nerozcházejí a stačí provádět pouze jednorázové nastavení hodnoty systémového času MCU dle hodnoty v Ethernet PHY.

K jednorázovému nastavení systémového času MCU se používá synchronizační signál PPS[15] vycházející z Ethernet PHY. MCU si pomocí funkce input capture[16] čítače a komunikace přes MII s Ethernet PHY nastaví systémový čas synchronní s Ethernet PHY. Tato synchronizace probíhá pouze při skokové změně systémového času s Ethernet PHY (ta nastává prakticky jen při startu PTP synchronizace nebo po chybě sítě). Schéma synchronizačního procesu je na obr. 17. Detailní princip synchronizace MCU s Ethernet PHY je uveden v [1].

[13] MII (Media Independent Interface) je rozhraní používané pro komunikaci Ethernet MAC a Ethernet PHY
[14] Převzato z [1]
[15] PPS (Pulse per second)
[16] Input capture (IC) je funkce čítače MCU, která zaznamená hodnotu načítanou v čítači při příchodu signálové hrany



Obr. 17 – Schéma synchronizace fyzické vrstvy a mikroprocesoru[17]

4.4.2 Generování a zpracování signálů

Jak již bylo zmíněno, jednotka SyncCore obsahuje 2 místa, která udržují systémový čas – Ethernet PHY a MCU. Obě tyto jednotky obsahují funkcionality generování signálů a zaznamenání času příchozí události (signálové hrany).

Ethernet PHY umožňuje generovat a zachytávat událost přes svoje GPIO[18]. Parametry dané signálové funkce (čas generování události, možnosti zachytávání času signálové hrany,…) se nastavují příkazy zaslanými přes MII rozhraní do Ethernet PHY. Pokud jde o generování signálu, tak stačí pouze zaslat daný příkaz do PHY. V případě, že se jedná o zachytávání času signálové události, tak je třeba aby MCU zaslalo do PHY příkaz k zachytávání události. PHY po zaznamenání události následně zasílá MCU zprávu obsahující informaci o zaznamenané signálové události, kterou MCU zpracuje. Velkou výhodou práce se signály přes Ethernet PHY je fakt, že MCU se nemusí zatěžovat se správou signálů, funkce správy signálů jsou v Ethernet PHY hotové a odladěné. Nevýhodou jsou zpoždění při MII komunikaci a nerozšiřitelnost na specifické signály (např.: PWM modulace, IRIG-B signál…).

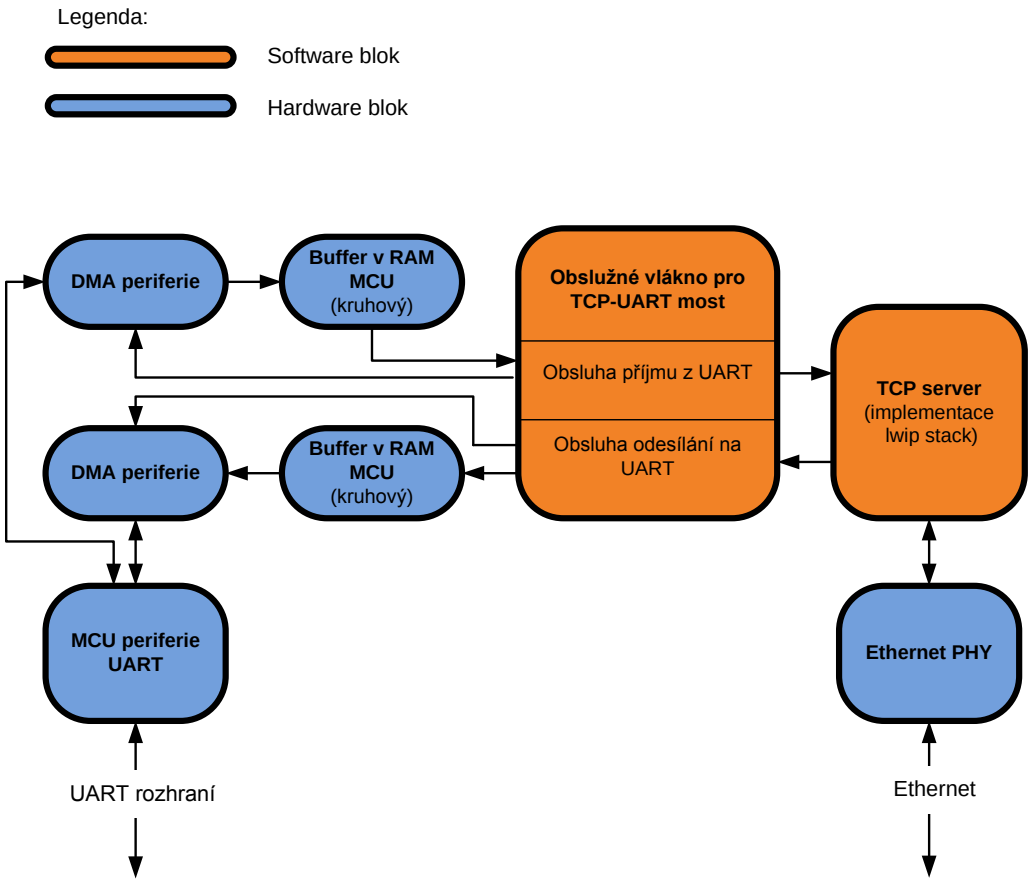
Práce se signály přes MCU funguje přes využití funkcí čítačů Output compare (generování signálů) a Input capture (zachytávání času příchozí hrany).[19] V rámci mé bakalářské práce byly tyto funkce generování událostí a zachycení příchozí události implementovány včetně možnosti generování signálu IRIG-B. Výhodnou přístupu přes MCU je větší volnost pro implementační možnosti (např. právě IRIG-B signál by přes PHY šel implementovat velice obtížně). Naopak nevýhodou tohoto přístupu je složitost implementace rozšiřování počtu vstupů a výstupů.

[17] Převzato z [1]
[18] GPIO – General Purpose input/output
[19] Detailní informace k principům práce se signály přes MCU jsou uvedeny v [1]

Pro TriggerBox Lite byla zvolena možnost práce se signály přes PHY zejména z důvodu přímočařejší implementace a možnosti snadného rozšiřování počtu signálových vstupů a výstupů jednotky. Logika práce se signály přes MCU v kódu zůstane pro potenciální budoucí použití, ale aktuálně se používat nebude.

4.4.3 Komunikační TCP-UART most

Poslední zásadní funkcionalitou SyncCore je komunikační TCP-UART most, který přeposílá data z UART na Ethernet (přes protokol TCP). V principu most funguje tak, že se uživatelský PC program přes Ethernet připojí k TCP serveru, který SyncCore vytvoří a následně se přeposílají data z TCP serveru na UART a naopak. Blokové schéma komunikačního mostu ukazuje obr. 18. Základem komunikačního mostu je FreeRTOS vlákno s vysokou prioritou tak, aby se spouštělo co nejblíže periodě 1 ms. V tomto vlákně probíhá softwarová část přeposílání dat mezi Ethernetem a UART, tzn. obsluha příjmu dat z UART a obsluha odesílání na UART.



Obr. 18 – Blokové schéma komunikačního TCP-UART mostu

Odesílání dat na UART vždy zkusí načíst data zaslaná na TCP server. Pokud jsou dostupná, překopíruje tato data do bufferu v RAM paměti MCU a nastaví DMA pro odesílání dat na UART. Při této operaci obsluha pracuje s bufferem jako s kruhovým[20]. DMA následně data samo postupně kopíruje na periferii UART, která data posílá.

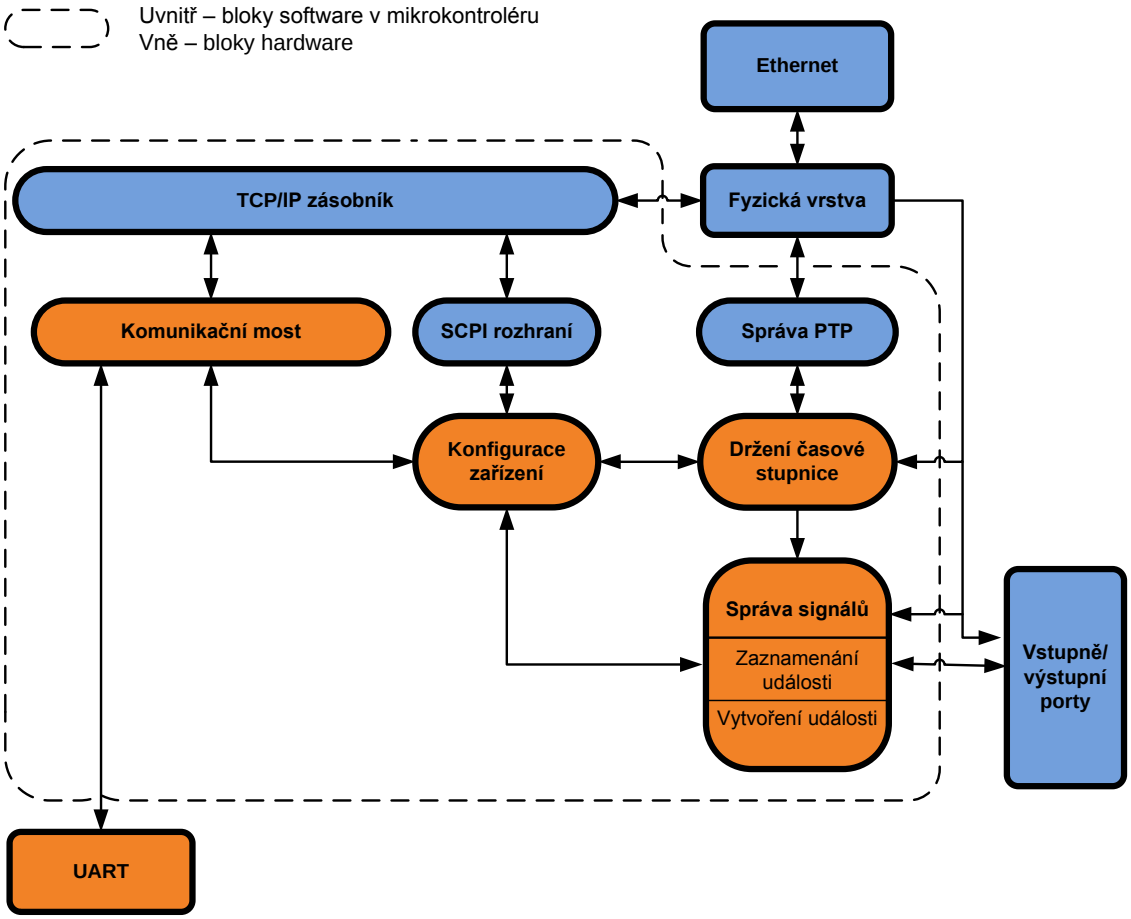
[20] Jedná se o buffer, kde se při dosažení konce bufferu nastaví aktuální bufferový ukazatel na začátek bufferu.

Obsluha příjmu dat z UART funguje podobně, ale opačně. DMA vždy kopíruje přijatá data do bufferu (DMA umí kopírovat do kruhového bufferu). Obslužný kód pak periodicky kontroluje, zda byla přijata data z UART a případně je posílá do TCP serveru.

Jednotka SyncCore také implementuje možnost TCP-SPI mostu. Tato funkcionalita ovšem pro TriggerBox Lite není potřeba, a proto není používaná. V kódu ovšem zůstává pro její potencionální využití.

5 Implementace TriggerBox Lite

Předchozí kapitoly popisují jednotku SyncCore, která je základem výsledné TriggerBox Lite jednotky. Tato kapitola se již zabývá tím, co bylo třeba nově vytvořit pro vznik výsledné TriggerBox Lite. Blokové schéma TriggerBox Lite je na obr. 19. První částí vývoje byla deaktivace nepotřebných bloků TCP-SPI mostu a podpory IRIG-B signálu. Tyto funkce ve zdrojovém kódu TriggerBox Lite zůstávají, ale nejsou využívány. K ponechání nepotřebných funkcí bylo rozhodnuto z důvodu potenciálního využití na další aplikace. Zbylé funkce zůstávají v základu nezměněné a popisují je předchozí kapitoly.



Obr. 19 – Blokové schéma TriggerBox Lite[21]

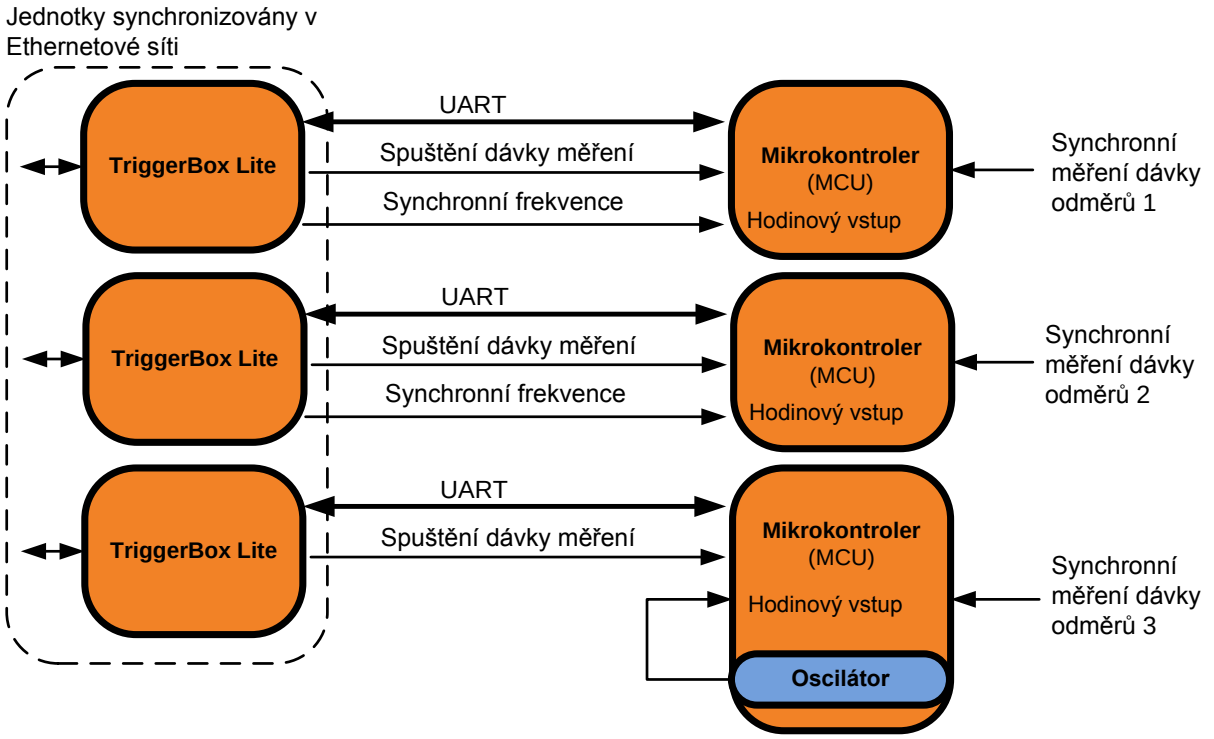
[21] Blokové schéma vychází z [1]

Následující seznam uvádí kroky potřebné k implementaci TriggerBox Lite

- implementace podpory pro více výstupů a podpora pro výstup hodinového signálu, který je využitelný jako hodinový vstup pro další mikrokontroler
- implementace fronty pro generování výstupních událostí, aby bylo možné naplánovat sekvenci událostí
- menší potřebné optimalizace – zejména přidaní druhého UART rozhraní
- ladění chyb a nedokonalostí jednotky

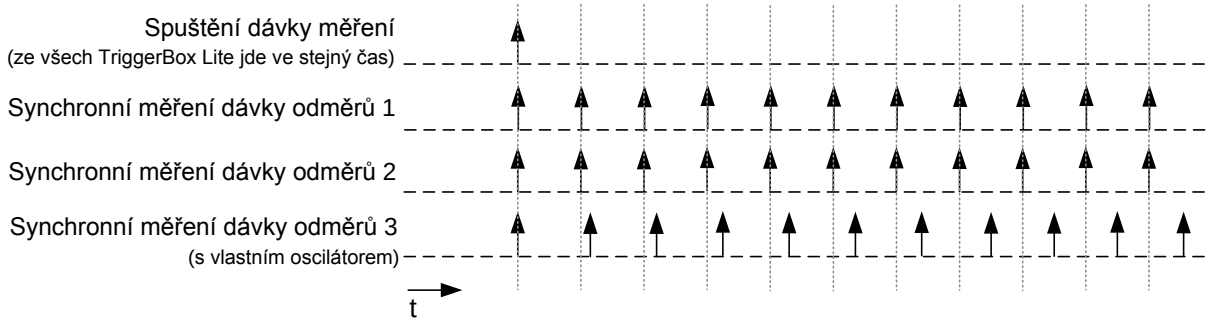
5.1 Rozšíření výstupů jako zdroje hodinového signálu MCU

Základním důvodem potřeby rozšíření možností výstupů generovaných událostí je požadavek na možnost poskytnutí synchronního signálu pro podřízenou jednotku (mikrokontroler). Pokud totiž MCU pracuje interně s časovou informací a běží na vlastím hodinovém zdroji, budou výsledky jejího měření zatížené velkou synchronizační chybou. Příklad takové situace ukazuje obr. 20, kde jsou mikrokontrolery, které po příchodu trigger pulsu pro spuštění dávky měření změří několik odměrů. Perioda odměrů je tedy určena z interních hodin MCU. První 2 MCU používají externí hodinový vstup z TriggerBox Lite, tzn. běží hodinách synchronních s TriggerBox Lite. 3. MCU pro názornost používá vlastní oscilátor.



Obr. 20 – Příklad synchronního měření s dávkovými odměry

Následující obr. 21 pak ukazuje časy jednotlivých odměrů v měřicí dávce. Všechny dávkové odměry začínají ve stejný čas, protože spuštění dávky vychází ze zasynchronizovaných TriggerBox Lite.



Obr. 21 – Ukázka rozsynchronizování měření v příkladu z obr. 20

Časy dávkových odměrů jednotek 1 a 2 zůstávají synchronní, protože používají hodiny odvozené ze zasynchronizovaných TriggerBox Lite. Třetí jednotka začne dávkový odměr ve stejný čas, ale vzhledem k tomu, že používá vlastní oscilátor, který nemá synchronní frekvenci se zasynchronizovanými TriggerBox Lite jednotkami, tak její další odměry s jednotkami 1 a 2 synchronní nejsou. Pro lepší odhad lze chybu v dávkovém odměru vypočítat. Předpokládejme použití krystalu jako oscilátoru s frekvenční tolerancí 30 ppm[22]. Při 500 odměrech v dávce s frekvencí odměrů 100 Hz a maximální chybě frekvence krystalu lze chybu synchronizace posledního odměru ΔT vypočítat:

$$\Delta T = \frac{n}{f_m}(\Delta_k) = \frac{500}{100}(0{,}00003) = 150\ \mu s$$

n (-) – počet odměrů

f_m(Hz) – frekvence odměrů

Δ_k(%) – maximální chyba krystalu (zde 30 ppm)

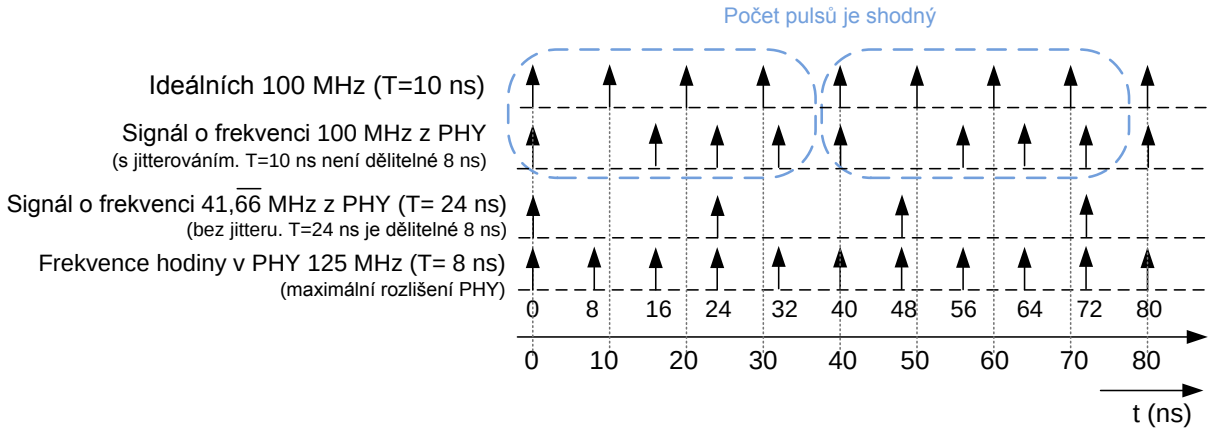
Chyba 150 µs je už pro mnoho aplikací velká, proto je skutečně vhodné pro podobné realizace s dávkovými měřeními poskytnout podřízené jednotce hodinový signál. Pro poskytnutí tohoto hodinového výstupu jsou v TriggerBox Lite 2 možné implementace:

- S využitím výstupů čítačů mikrokontroleru STM32F4
- S využitím výstupů (triggerů) použité PHY

Nevýhodou použití výstupů čítačů je zbytečně rostoucí náročnost implementace a rozšiřitelnost. Hodinový výstup by nebyl použitelný pro žádnou další funkcionalitu. Pouze by generoval signál o dané frekvenci. Při použití výstupů PHY je možné přidat další univerzální výstup, který bude umět generovat události i signály o dané frekvenci (generování signálu o frekvenci je to samé jako generování periodických událostí). Z uvedených důvodů byl přidán další univerzální výstup z PHY.

[22] ppm = Parts per milion, tedy miliontina. Pro příklad 10 ppm=0,000 01%

Každý výstup TriggerBox Lite pro v generování událostí se dá využít jako zdroj synchronní hodinové frekvence i jako zdroj synchronních událostí. Pro hodinový vstup MCU je třeba ještě analyzovat frekvence signálů, které PHY dokáže generovat. Rozlišení PHY je 8 ns (maximální frekvence tedy 125 MHz). To vychází z frekvence signálu, na který je PHY napojená – 25 MHz (perioda 40 ns), která je následně je přenásobená 5x na 125 MHz. PHY ovšem nabízí i možnost nastavení generování signálu frekvenci, jejíž perioda není soudělná 8 ns. Takové frekvence nelze dosáhnout běžným dělením frekvence. V signálu s periodou nesoudělnou 8 ns se ovšem začne objevovat jitter[23], který pro hodinový vstup MCU není vhodný. Zdroj jitteru vysvětluje následující ilustrace.



Obr. 22 – Jitter na výstupním signálu z PHY

Obrázek ukazuje náběžné hrany signálů. Ve spodní části obrázku je 125 MHz, které jsou interně v PHY. Nad tímto průběhem je příklad generovaného signálu o frekvenci, jejíž perioda je dělitelná 8 ns. Zde žádný jitter není. Na druhém průběhu odshora obrázku je vidět průběh generovaného signálu o frekvenci 100 MHz jejíž perioda není dělitelná 8 ns. Díky tomu pulsy nemají konstantní periodu. V porovnání s horním průběhem ideální 100 MHz jsou vidět 2 informace: zaprvé signál z PHY nemá nikde periodu 10 ns jako ideální průběh. Zadruhé počet impulsů je ve větším časovém kvantu stejný jak pro ideální průběh tak generovaný z PHY. Dá se proto říci, že generovaný signál má 100 000 000 pulsů za sekundu, jak se od 100 MHz signálu očekává. Obsahuje ovšem jitter.

Z výše uvedeného lze vyvodit, že PHY dokáže generovat i signály o frekvencích, jejichž perioda není bez zbytku dělitelná 8 ns, ale jako hodinový vstup do MCU nesoudělná frekvence není vhodná kvůli jitteru. Pro MCU je možné bez jitteru generovat signály o frekvencích např.:

- f=5 MHz (T=200 ns)
- f=6,25 MHz (T=160 ns)
- f=25 MHz (T=40 ns)

[23] Jitter v periodickém signálu znamená situaci, kdy se objevují pulsy, které mají jinou periodu než většina. Je to „poskakování“ period pulsů.

5.2 Implementace fronty generovaných událostí

Další důležitou funkcionalitou je fronta pro generování událostí, tedy možnost naplánovat sekvenci událostí a jejich čas. Někdy nestačí vygenerovat jednu událost nebo spustit periodickou sekvenci událostí. Opět je v rámci analýzy nutné zmínit 2 možnosti implementace: přes výstupy čítačů MCU nebo výstupy PHY. Ani jedno z řešení frontu přímo nepodporuje.

První přístup – čítače MCU by fungoval v principu tak, že by se pro nastavení generování každé události z fronty využívalo DMA. Tzn. daný kanál čítače by byl napojen na DMA, aby se při vygenerování události (Output compare - OC funkce) spustil DMA přenos, který do OC registrů kanálu čítače nastavil patřičné hodnoty času pro generování další události. Výhodou řešení je skutečnost, že je možné generovat události rychle za sebou bez součinnosti MCU. Na druhou stranu nevýhodami jsou skutečnosti, že by bylo třeba měnit stávající architekturu generování událostí přes MCU a optimalizovat ji krom fronty událostí i pro periodické události, které nepodporují.

Druhou možností je využít generování událostí přes PHY. Ta ovšem nenabízí lepší možnost implementace fronty událostí než softwarově. Tzn. nastavení generování další události ve frontě bude probíhat softwarově s využitím procesorového času. Zde je tedy z principu velké omezení na frekvenci generovaných událostí za sebou. Na druhou stranu PHY umožňuje generovat periodické události a jsou s její pomocí implementovány všechny výstupní událostí jednotky TriggerBox Lite.

Z důvodu lepší a jednodušší implementace byla zvolena implementace přes výstupy PHY.

Pro implementaci přes PHY se na první pohled jeví jako nejlepší řešení nastavit generování další události ve frontě v okamžiku, kdy přijde od PHY zpráva, že byla vygenerována předchozí událost. Tento přístup má ovšem zásadní nedostatek v rychlosti. PHY je totiž k MCU TriggerBox Lite připojena přes MII rozhraní[24], tzn. informace o vygenerované události se musí přenést přes MII. K informacím z MII rozhraní se pak programové vlákno dostane až ve chvíli, kdy mu task scheduler (plánovač) operačního systému FreeRTOS [3] přidělí procesorový čas. Vzhledem k tomu, že se přepínání procesorového času a plánování děje každou 1 ms, tak se běžně může stát, že by vlákno zpracovalo informaci o vygenerované události až za 20 ms a déle od vygenerování předchozí události. Zároveň není možné přesně garantovat čas, kdy se vlákno dostane k procesorovému času.

Z těchto důvodů je software obsluha fronty událostí implementována přes xTimers FreeRTOS. Tyto timery jsou speciální typ vláken, u kterých se nastavuje perioda spouštění v jednotkách milisekund. Při změně kontextu (aktuálně zpracovávaného vlákna) – tedy na začátku každé

[24] Detailnější popis v kapitole 3.1.1

milisekundy FreeRTOS zjišťuje, zda neuplynula perioda spuštění daného timeru a pokud ano, spouští timer jako první před běžnými vlákny.

U fronty událostí je tohoto chování využito tak, že je xTimer pro obsluhu fronty událostí spuštěn s periodou 1 ms (tzn. spouští se při každém přeplánování na začátku milisekundy) a kontroluje, zda nebyla v uplynulé milisekundě vygenerována výstupní událost. Kontrolu vygenerování události kontroluje program ze systémového času uloženého v čítači a paměti MCU, protože komunikace s PHY by byla výkonově neúnosná. Pokud program vyhodnotí, že událost vygenerována byla, nastavuje generování další události ve frontě, pokud fronta není prázdná. Pro implementaci fronty jako takové se používá xQueue FreeRTOS. Implementaci periodické obsluhy xTimeru fronty událostí ukazuje následující pseudokód:

```
//funkce spouštěna na začátku každé milisekundy před ostatními vlákny
function obsluha_xTimer_fronty_událostí()

systemový_čas_MCU = načti_systémový_čas_MCU()

if je_naplánována_výstupní_událost()==FALSE
      return//ukonči činnost
čas_poslední_události_fronty=načti_čas_poslední_události_fronty()

if čas_poslední_události_fronty < systemový_čas_MCU
      return//ukonči činnost – událost ještě neproběhla

//v předchozí milisekundě byla vygenerována událost
if je_fronta_prázdná()
      return//ukonči činnost – není další událost k naplánování

naplánuj_generování_další_události_fronty()

end function
```

Takto implementovaná fronta událostí umožňuje naplánovat generování událostí až 10 ms za sebou[25]. Tato naměřená hodnota není překvapující, protože vliv MII komunikace s PHY a plánování vláken FreeRTOS je velký. Pro aplikace TriggerBox Lite je tato frekvence však postačující.

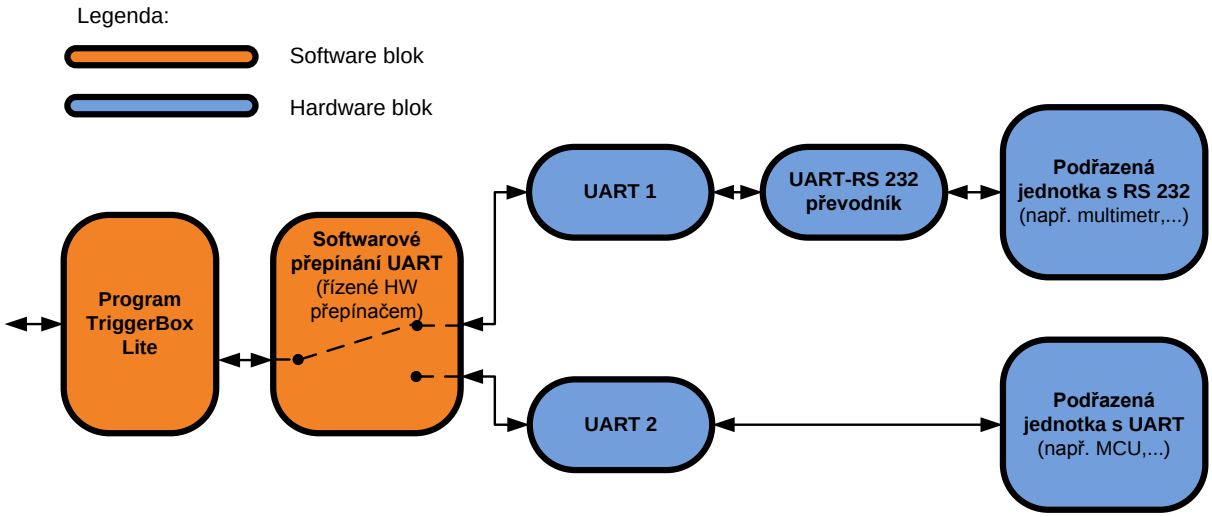
5.3 Menší potřebné optimalizace

Posledním blokem implementace TriggerBox Lite byly menší optimalizace a ladění chyb, které vyplynuly z testování a používání TriggerBox Lite.

Z těchto optimalizací bylo nejpodstatnější zohlednění možností jednotek připojovaných ke komunikačnímu mostu. Měřicí přístroje mají běžně sériové rozhraní RS 232. Ovšem pokud je k TriggerBox Lite potřeba připojit obyčejné MCU, je vhodné zohlednit, že MCU mívají pouze rozhraní UART. Aby tedy TriggerBox Lite vyšel vstříc oběma typům přístrojů, je třeba připravit

[25] Pro 9 ms TriggerBox Lite občas nestihl nastavit generování události. Pokud by se měla garantovat minimální bezpečná prodleva mezi událostmi, je třeba vynásobit prodlevu 10 ms dvěma, tzn. 20 ms.

hardware i firmware tak, aby bylo možné buď vyvést prostý UART nebo přes UART-RS 232 převodník vyvést přímo rozhraní RS 232. Princip ukazuje obr. 23



Obr. 23 – Dvě UART alternativy pro možnost výstupu UART i RS 232

TriggerBox Lite při inicializaci TCP-UART mostu kontroluje logickou úroveň na pinu GPIOD7. Tento pin je připojen na hardwarový přepínač (jumper). Podle této logické úrovně následně nastaví používání UART 1 nebo UART 2. Tento přístup umožnuje vytvořit jednu desku se vstupy a výstupy, kde se až při konkrétní aplikaci pomocí HW přepínače rozhodne, zda se má používat UART nebo RS 232.

5.4 Ovládací aplikace

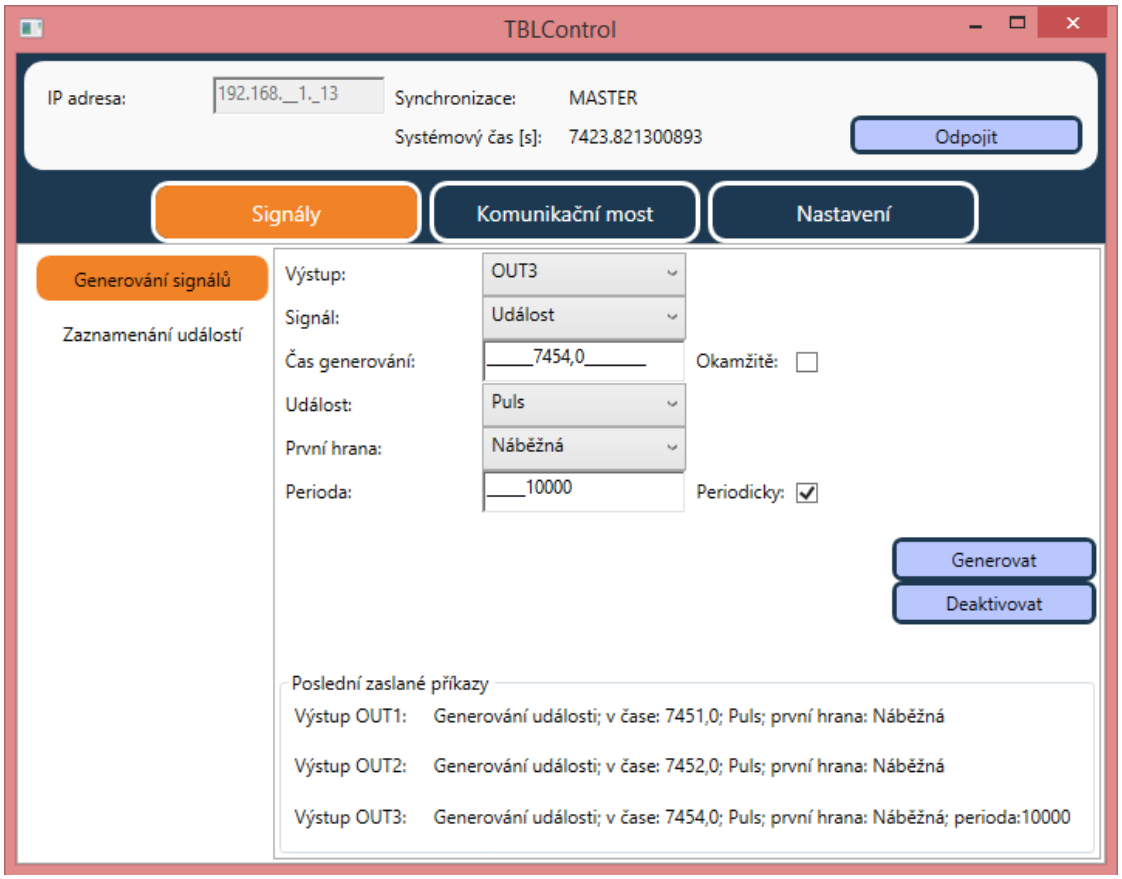
Pro demonstraci všech funkcí TriggerBox Lite bylo třeba poskytnout PC aplikaci, která umožní TriggerBox Lite nastavit a tak ukáže všechny možnosti nastavení jednotky. V rámci bakalářské práce [1] byla vytvořena PC aplikace SyncCoreControl, která potřebné nastavení umožnila. Pro účely této diplomové práce bylo třeba aplikaci jen částečně upravit a opravit problémy s TCP protokolem, které se objevily při přechodu z operačního systému Windows 7 na Windows 8.1. Upravená aplikace byla následně pojmenována TBLControl (TriggerBox Lite Control).

PC aplikace je implementována v jazyce C# s využitím technologie WPF [7]. Umožnuje se připojit k jednotce přes Ethernet a spravovat funkcionality:

- Práci se signály – generování a zaznamenávání signálů přes TriggerBox Lite
- Nastavování a komunikaci přes TCP-UART most
- Nastavení TriggerBox Lite (reset do továrního nastavení, IP adresa, …)

Zároveň zobrazuje čas v jednotce a její synchronizační stav v PTP síti (master, slave, listening).

Náhled aplikace ukazuje obr. 24



Obr. 24 – PC aplikace TBLControl

V rámci aplikace byla vytvořena knihovna pro práci s TriggerBox Lite, kterou je možné použít i v dalších projektech. Pro účely diplomové práce byla použita v obslužných programech pro demonstrační úlohy.

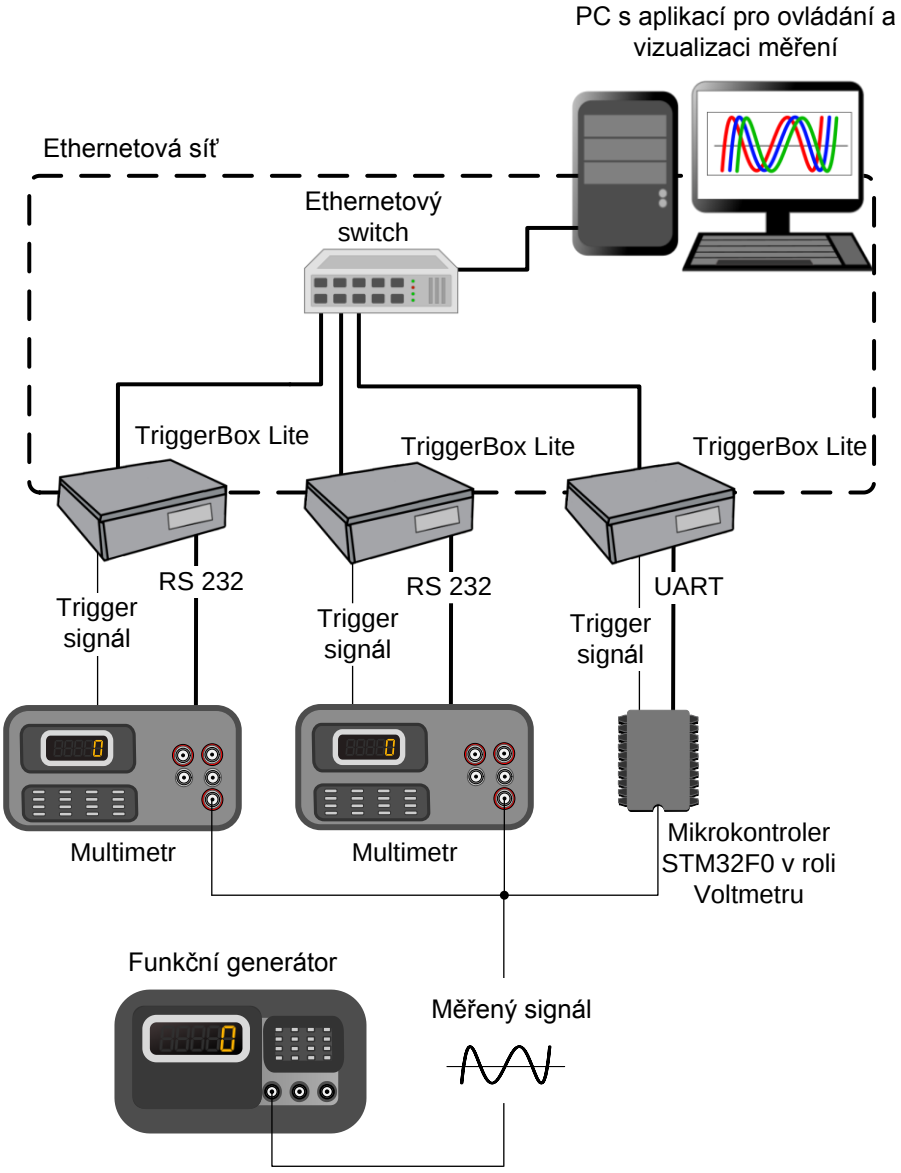
6 Implementace demonstračních úloh

Pro demonstraci funkcionalit TriggerBox Lite bylo potřeba připravit demonstrační úlohy. Z mnoha potenciálních aplikací TriggerBox Lite byly vybrány 2 hlavní. První ukazuje možnosti synchronního měření napětí signálu. Druhou demonstrační úlohou je synchronní otáčení dvěma BLDC motory. Opět se jedná o běžné použití PTP synchronizace v průmyslu, protože synchronizace akčních členů je vedle synchronních měření druhou velkou skupinou využití PTP protokolu.

6.1 Demonstrační úloha s voltmetry

Demonstrační úloha s voltmetry je založena na principu synchronního měření stejného signálu. Díky měření jednoho stejného signálu je na výsledcích měření vidět, zda a jak jsou jednotlivé měřicí jednotky distribuovaného synchronního systému synchronizované. V praxi by se synchronní distribuované měření dalo využít např. k detekci fázového posuvu signálu v elektrické síti.

Schéma dema ukazuje obr. 25. Měřený signál pro úlohu generuje funkční generátor, který může generovat velké spektrum signálu. Pro tuto úlohu je nejnázornější obyčejná funkce sinus. Napětí tohoto signálu následně měří voltmetry.



Obr. 25 – Schéma dema s voltmetry

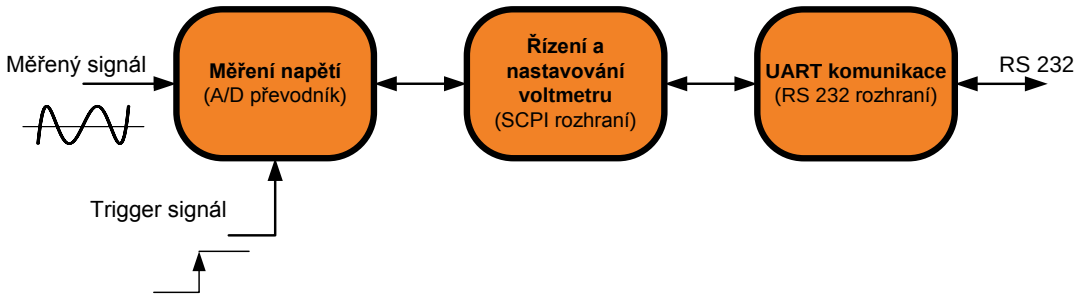
Jako voltmetry jsou použity multimetry s komunikačním rozhraním RS 232 a také moduly STM32F0 (konkrétně STM32F0 Discovery). Důvodem nutnosti použití modulů STM32F0 byl problém s nedostatkem vhodných multimetrů dostupných na katedře. Proto se tedy modul STM32F0 naprogramuje tak, aby implementoval všechny funkce multimetrů, které jsou k demonstraci potřeba.

Jednotlivé odměry voltmetrů se nastavují přes sériový port (u multimetrů přes RS 232 a u modulů STM2F0 přes UART) a následně se spouštějí přes trigger signál. Voltmetry jsou

nastaveny tak, aby byl při příchodu náběžné hrany na trigger vodiči proveden jeden odměr napětí vstupního měřeného signálu. O ovládání měření se stará PC aplikace, která prostřednictvím zasynchronizovaných TriggerBox Lite jednotek nastavuje voltmetry. Konfigurace voltmetrů probíhá přes TCP-UART most TriggerBox Lite. Následné generování trigger signálu nastavuje aplikace také TriggerBox Lite jednotkám na konkrétní čas. PC aplikace po každém měřicím bloku zobrazuje v grafu naměřený průběh signálu, takže je vidět, zda se naměřená data ze všech jednotek překrývají nebo zda se v čase liší.

6.1.1 Použité funkce multimetru

Pro lepší pochopení funkcí potřebných k implementaci voltmetru v STM32F0 je na obr. 26 uvedeno jednoduché blokové schéma funkcí multimetru nutných pro tuto úlohu.



Obr. 26 – Blokové schéma použitých funkcí multimetru

Měřený signál se vzorkuje A/D převodníkem. Tento A/D převod je spouštěný náběžnou hranou trigger signálu. U dostupného multimetru HP34401 je maximální frekvence spouštění měření 1000 triggerů za sekundu a maximální počet uložených hodnot je 512.

Multimetr HP34401 implementuje mimo jiné komunikační rozhraní RS 232 (tzn. Komunikace stejná jako klasický UART jen se specifickými napěťovými úrovněmi). Přes toto rozhraní je možné multimetr nastavit a vyčíst data. Všechna komunikace přes RS 232 je dle standardu příkazů SCPI. To znamená, že z hlediska PC aplikace, která multimetr zprostředkovaně nastavuje, nezáleží na konkrétním typu multimetru. Jediné důležité je, aby implementoval standard SCPI a neměl výrazně jiné specifikace (např. menší paměť na data nebo nižší maximální vzorkovací frekvenci).

Výše popsané funkcionality je třeba v modulu STM32F0 naprogramovat. Hlavní důraz je kladen na to, aby SCPI příkazy a chování jednotky byly stejné jako u HP34401.

6.1.2 Mikrokontroler STM32F0 jako voltmetr

Podle základních specifikací uvedených v předchozí kapitole již lze navrhnout detailní blokové schéma jednotky. Dvě největší části jsou: triggerovaný A/D převod a vhodné odesílání dat.

Pro triggerovaný A/D převod poskytuje mikrokontroler STM32F0 důležité periferie A/D převodníku a DMA napojený na tento převodník. A/D převodník mikrokontroleru poskytuje 12b rozlišení s časem převodu 1 µs, což je dostačující, protože referenční multimetr HP34401

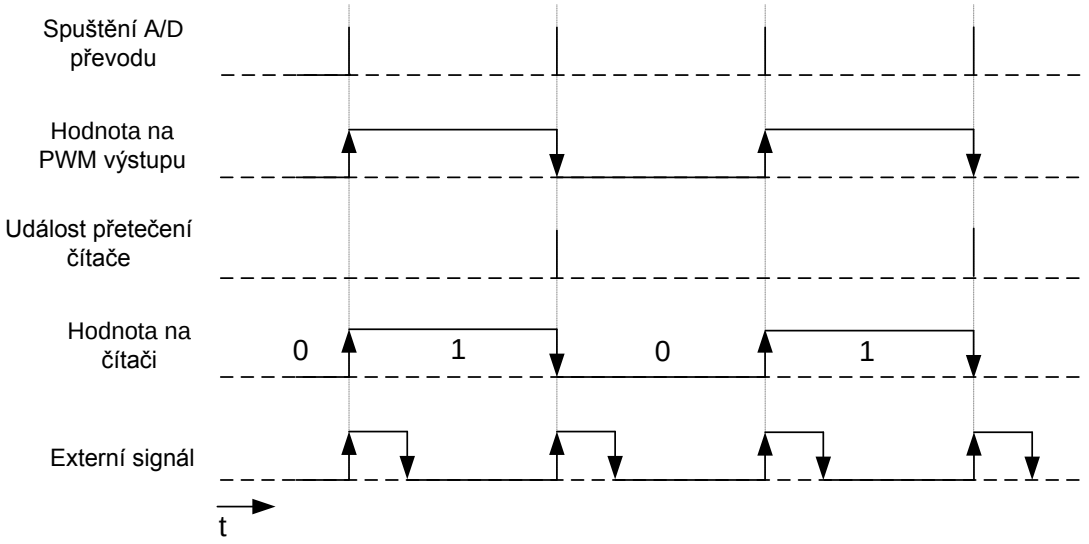
umožňuje provádět triggerovaná měření s periodou 1 ms. Využití DMA přenosu po každém A/D převodu umožní snížit zátěž procesoru, i když pro tuto aplikaci není klíčový – procesor zde není tolik vytížen. Z předchozí kapitoly je patrné, že by jednotka měla být schopna uložit do paměti 512 naměřených vzorků. Převodník umožňuje 12b rozlišení a tím pádem se do paměti ukládají 16b (2B) čísla. V paměti je tedy třeba alokovat:

$$h_b = 2 \cdot 512 = 1024\ B\ paměti$$

Zvolený mikrokontroler STM32F0 Discovery poskytuje 8 kB paměti. V mikrokontroleru tedy zbyde 7 kB paměti na ostatní funkcionality, což je postačující.

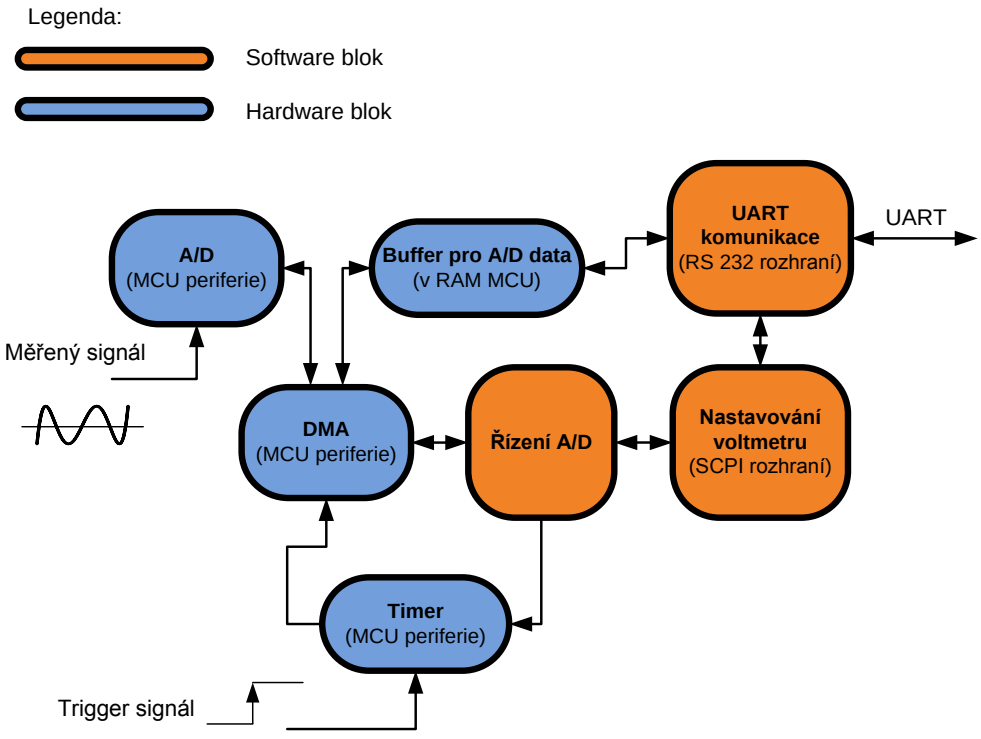
Jako malý problém se u STM32F0 ukázala nemožnost triggerování externím pulsem. Periferie A/D u tohoto mikrokontroleru totiž neposkytuje přímo možnost triggerování externím signálem. Poskytuje pouze triggerování softwarovým spuštěním nebo čítači. Softwarové spouštění je sice nejjednodušší, ale poskytuje relativně velkou nepřesnost způsobenou režií na obsluhu přerušení. Přerušení je totiž třeba obsloužit při příchodu externího pulsu a následně spustit A/D převod.

Druhá a přesnější cesta je přes využití čítačů. Čítače je možné nastavit tak, aby jejich výstup (PWM, událost při přetečení, ...) spouštěly A/D převod. Zároveň čítače poskytují možnost čítání externích pulsů (inkrementují svoji hodnotu při každé náběžné hraně signálu). Tímto principem je možné zpracovat externí signál. Dalším problémem je ovšem fakt, jak provést, aby čítač při každé náběžné hraně signálu spustil převod. Není totiž možné čítač nastavit tak, aby přetekl (a vygeneroval puls/událost při přetečení) při každé náběžné hraně čítaného signálu. Čítač umí přetéct minimálně každou druhou náběžnou hranu (dělí frekvenci signálu minimálně 2). Tento problém se naštěstí také dá vyřešit pomocí PWM výstupu. Čítač se nastaví tak, aby čítal minimální možnou hodnotu – tzn. 0,1 a zároveň generoval PWM s 50% střídou. Znamená to, že při každé náběžné hraně externího signálu změní PWM výstup svou hodnotu. Na A/D převodníku pak stačí nastavit, aby spouštěl převod při změně hodnoty výstupu daného čítače. Díky tomu je A/D převod spouštěn skutečně při každé náběžné hraně externího signálu. Princip je znázorněn na následujícím obrázku.



Obr. 27 – Princip zpracování externího triggeru čítačem

Pro finální realizaci voltmetru byla zvolena možnost externího triggerování přes čítač. Výsledné blokové schéma A/D části voltmetru ukazuje následující obrázek.



Obr. 28 – Blokové schéma měřicí části STM32F0 jako voltmetru

Druhým velkým blokem v rámci implementace voltmetru je UART komunikace a SCPI rozhraní. Pro SCPI rozhraní byla opět zvolena knihovna scpi-parser [8]. Pro tuto knihovnu se nadefinovaly potřebné callbackové funkce a knihovna sama řeší parsování SCPI příkazů a volání patřičné funkce.

Samotná UART komunikace je zprostředkovaná periferiemi UART a DMA napojeným na UART. K přijímání dat z UART se DMA nepoužívá, protože je zde pro SCPI příkazy nutná kontrola, zda

přijatý text neobsahuje znaky ukončující příkaz. Pro SCPI se jedná o znak „\n“[26] (ASCII 10 – znak „new line“) nebo sekvenci „\r\n“ (ASCII 13,10 – znaky „carridge return“, „new line“). Proto se zpracovává každý příchozí znak a kontroluje se, zda se nejedná o znak ukončující SCPI příkaz.[27]

Pro odesílání dat se DMA používá běžným způsobem, tzn. data se připraví do bufferu pro odesílaná data a DMA se nastaví na postupné kopírování dat do periferie UART, která vše odešle. Pro odesílání naměřených hodnot se však implementace částečně zkomplikovala. Odeslání dat totiž ve SCPI probíhá tak, že voltmetr odesílá všechna naměřená data v rámci jedné zprávy, která je odpovědí na SCPI příkaz *fetch*. Buffer pro odesílaná data by tak musel být tak velký, aby se do něho vešlo 512 hodnot měření převedených do textového řetězce. Naměřené hodnoty napětí převedené do textového řetězce mají v implementaci voltmetru HP43301 tvar:

```
+0.00000000E-04,-3.10000000E-03, 4.80000000E-03, 5.30000000E-02
```

Na jednu hodnotu i s oddělovací čárkou tedy vychází 16 znaků. Odesílací buffer by tak musel mít velikost minimálně:

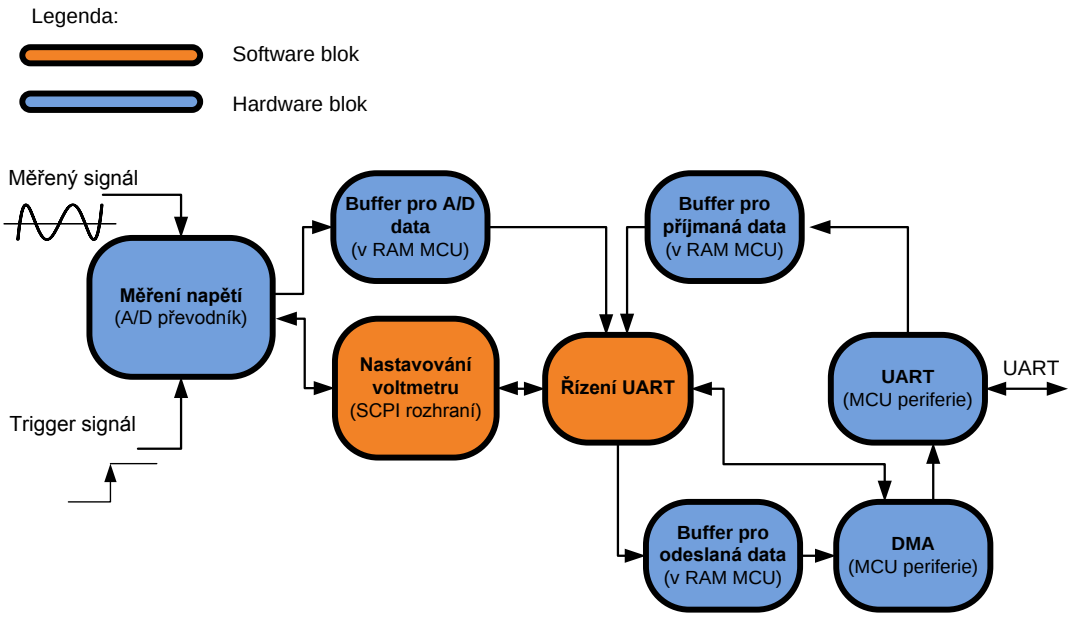
$$h_b = 16 \cdot 512 = 8\ kB\ paměti$$

Což je veškerá RAM paměť mikrokontroleru, a proto není možné ji alokovat.

Pro odesílání dat z měření bylo tedy nutné vytvořit funkcionalitu odesílání dlouhých zpráv. Jedná se o princip, kdy se do odesílacího bufferu postupně kopírují části dat z bufferu pro A/D převod. Díky této optimalizaci je možné zvolit velikost odesílacího bufferu jen 256 B. Výsledné blokové schéma voltmetru ukazující odesílání dat přes UART je uvedeno na obr. 29. Na tomto obrázku jsou pro jednoduchost všechny bloky A/D převodu zahrnuty do jednoho bloku, takže obrázek neukazuje celé detailní blokové schéma voltmetru.

[26] Notace „\r“ je převzata z funkce sprintf() jazyka c, kde zpětné lomítku značí, že následující znak bude placeholder pro speciální znak – např. konec řádku,…

[27] I pro přijímání s kontrolou znaku se dá použít DMA, protože implementuje možnost vyvolat přerušení při detekci znaku. Tato funkcionalita byla však objevena v dokumentaci až po implementaci UART. Jak již však bylo zmíněno, v této implementaci není procesor vytížený, a proto nebyla nutná optimalizace příjmu s DMA.



Obr. 29 – Blokové schéma komunikační části STM32F0 jako voltmetru

Takto implementovaná logika zvládá bez větších problémů UART komunikaci s rychlostí 115 200 Bd/s i při obsluze ostatních funkcí.

U implementace převodu 16b hodnoty napětí na textový řetězec se vyskytl na první pohled banální problém. Při použití standardní C knihovny stdio.h a její funkce pro generování textových řetězců sprintf() nastala potíž s převodem desetinného čísla na text. Knihovna má samozřejmě plnou podporu pro čísla s desetinnou čárkou, ale je primárně určena pro PC aplikace, kde na rozdíl od MCU není RAM paměť velkou limitací. Knihovna si proto při pokusu o převod čísla s desetinnou čárkou na textový řetězec zkusí dynamicky alokovat okolo 4 kB paměti, což vybraný mikrokontroler nemůže poskytnout.

Jsou pak prakticky 2 řešení problému. Prvním je použití jiné C knihovny, která je určená pro MCU, např. Newlib. Programátor musí ovšem počítat s tím, že naportování knihovny zabere určitý čas. Druhým řešením je naimplementovat si převod sám s použitím stdio.h funkcí pro převod celých čísel. Pro tuto práci byla vybrána druhá jednodušší varianta, protože zde byl problematický převod potřeba pouze na jednom místě.

Při návrhu struktury programu byl kladen důraz na rozšiřitelnost a znovu použitelnost kódu, protože obslužný firmware druhého dema (s BLDC motory) je také implementován na STM32F0. Struktura C knihoven pro voltmetr je následující:[28]

- Main – pouze základní definice a spouštění programu
- Measure – knihovna pro A/D měření, jeho inicializaci, spouštění a zpracování

[28] V tomto seznamu jsou pouze knihovny, které byly implementovány v rámci této práce. Knihovny, které byly převzaty (ST32F0 standard lib, scpi-parser,…) zde popsány nejsou.

- Vmet_scpi – definice SCPI pro knihovnu Scpi-parser [8]. Jedná se o SCPI callback funkce a nastavení Scpi-parseru
- Common_libs/Vmet_usart – knihovna pro USART odesílání s možnostmi popsanými výše. Tato knihovna je záměrně ve složce Common_libs, protože je zároveň importovaná i do firmware STM32F0 pro ovládání BLDC v druhém demu. Kód této knihovny je tedy pro obě dema stejný. V každém demu se jen liší nastavení této knihovny (tzn. C defines a inicializace potřebných callback funkcí)

6.1.3 PC aplikace pro voltmetr demo

Posledním dílem práce na demu s voltmetry je ovládací PC aplikace, která se stará o všechna nutná nastavení. Aplikace je implementována v jazyce C# s využitím .Net technologie WPF.

Aplikace se skládá ze třech základních vláken:

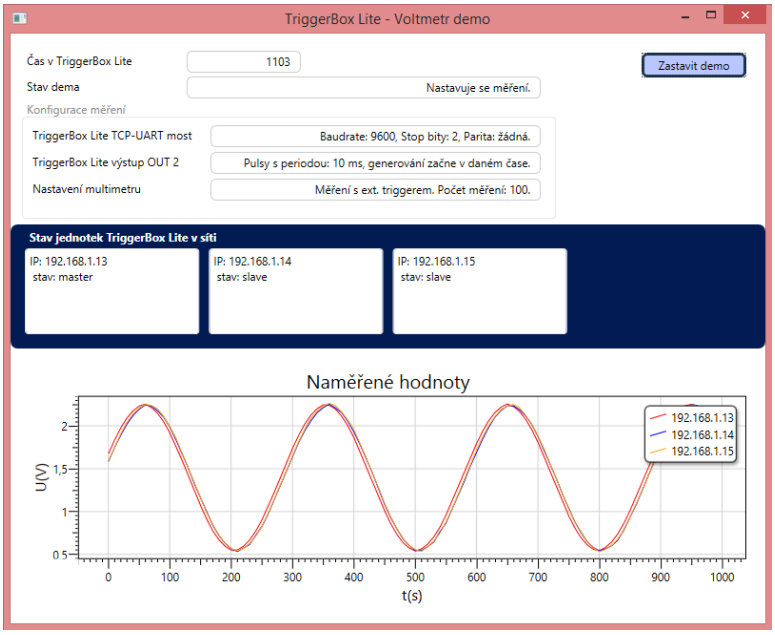
- Vlákno pro uživatelské rozhraní
- Vlákno pro nastavení a vyčítání dat z měření
- Vlákno pro periodické vyčítání času a PTP stavu v TriggerBox Lite jednotkách
- Vlákno pro otevření potřebných připojení k TriggerBox Lite
    - Jedná se o vlákno, které je potřeba při spuštění dema. Po provedení připojení vlákno ukončuje činnost

Nejdůležitějším z pohledu dema je vlákno pro nastavení a vyčítání dat z měření. To periodicky provádí činnost nastíněnou v následujícím pseudokódu

```
Čekej_na_připojení_TBL()//čeká na připojení všech TriggerBox Lite ve vlákně pro otevření všech potřených připojení

//nekonečná smyčka
while true do

   //Blok kódu pro nastavení měření

   čas=získej_aktuální_čas_v_TBL()

   //prochází připojení ke všem  TriggerBox Lite
   for i←0 to délka(seznam_dostupných_TBL)do

      povol_TCP_most_v_TBL(seznam_dostupných_TBL [i])

      //Nastaví voltmetr na odměry napětí při příchodu trigger pulsu. Tzn. jeden trigger puls = jeden odměr. Dále nastaví, aby voltmetr provedl jen určitý počet měření (např. 100) a pak činnost ukončil
      nastav_měření_voltmetru_přes_TCP_most(seznam_dostupných_TBL[i],100)

      //Nastaví generování periodických pulsů (trigger pulsy pro voltmetry) tak, aby se začaly generovat za daný počet sekund od aktuálního času. Tzn. ve všech TriggerBox Lite stejně
      nastav_gener_výstup_události_v_TBL(seznam_dostupných_TBL [i],čas)

   čekej()//čekej na proběhnutí měření
```

```
//Blok kódu pro načtení, zpracování a zobrazení dat z měření
   for i←0 to délka(seznam_dostupných_TBL)do
      vypni_generování_výstupní_události_v_TBL(seznam_dostupných_TBL [i])
      vyčti_data_z_voltmetru_přes_TCP_most(seznam_dostupných_TBL [i])
      vypni_TCP_most_v_TBL(seznam_dostupných_TBL [i])
      zobraz_data_z_voltmetru_na_grafu()
    čekej() //čekej mezi spouštěním měření
```

PC aplikace využívá knihovny pro obsluhu TriggerBox Lite implementované primárně pro obslužnou aplikaci TriggerBox Lite popsanou v kapitole 5.4. Aplikace je zobrazena na obr. 30. V horní části zobrazuje aktuální stav dema. Ve střední části seznam TriggerBox Lite jednotek v síti, jejich IP adresu a PTP stav (PTP master, PTP slave nebo stav hledání dalších PTP jednotek v síti). Ve spodní části je zobrazení graf s hodnotami naměřenými jednotlivými voltmetry v síti.



Obr. 30 – PC aplikace pro voltmetr demo

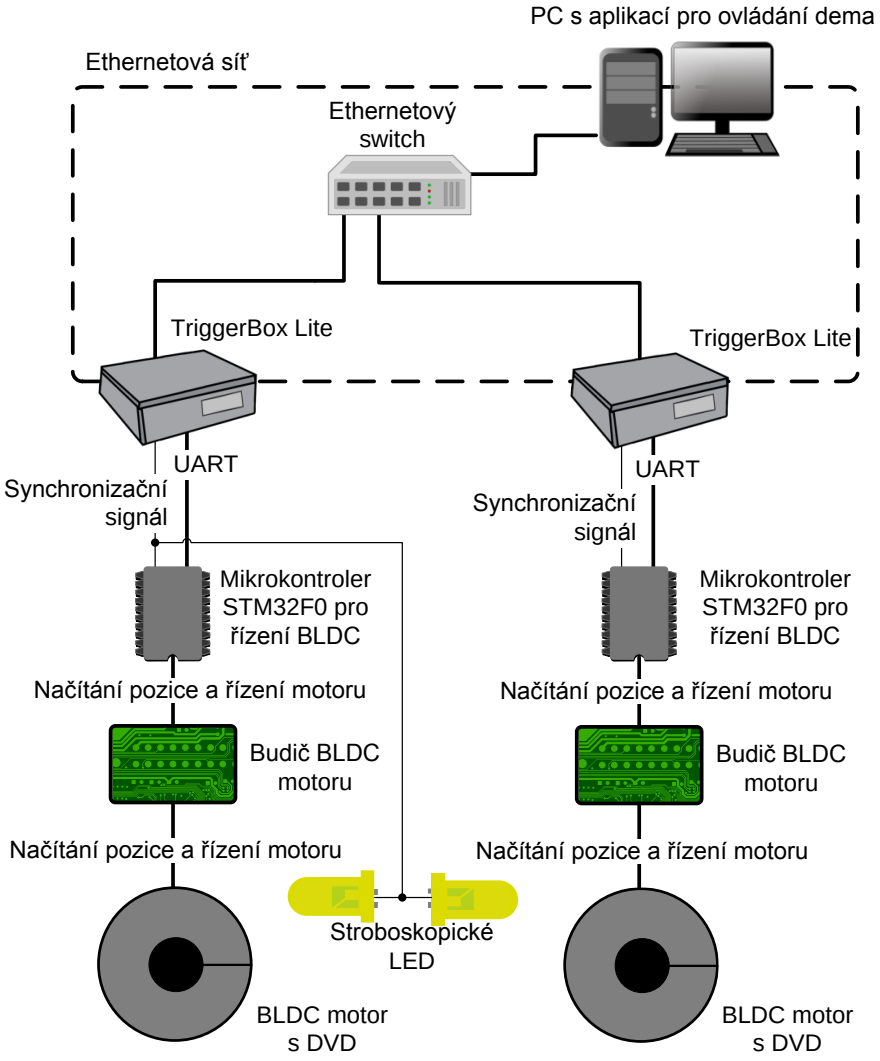
6.2 Demonstrační úloha s BLDC motory

Demo s BLDC motory je druhou demonstrační úlohou pro TriggerBox Lite. V principu jde o ukázku, kde se 2 BLDC motory točí se stejnou fází. Blokové schéma demonstrační úlohy ukazuje obr. 31.

Průběh demonstrační úlohy opět řídí PC aplikace, která nastavuje synchronizované jednotky TriggerBox Lite tak, aby generovaly referenční synchronizační signál pro motory, a nastavuje jednotky pro řízení BLDC motorů přes TCP-UART most v TriggerBox Lite. Tyto jednotky pro řízení BLDC motorů jsou implementovány na modulech STM32F0 Discovery a jsou nastavovány přes UART rozhraní. Synchronizační referenční signál, který přijímají od TriggerBox Lite reprezentují tak, že každá náběžná hrana znamená jednu otáčku motoru. Moduly STM32F0 tedy regulují BLDC motory tak, aby sledovaly referenční otáčky (referenční

signál) a hlavně držely správnou fázi, tzn. při každé náběžné hraně referenčního signálu byly motory natočeny do počáteční polohy.

Moduly STM32F0 spolupracují s BLDC motory prostřednictvím desky s budiči pro BLDC motory, kterou implementoval Ing. Vavrouš v rámci své bakalářské práce [2]. Deska budičů umožňuje pracovat se všemi informacemi, které daný BLDC motor poskytuje – tzn. řídící signály a stavové signály z Hallových sond snímajících natočení motoru.

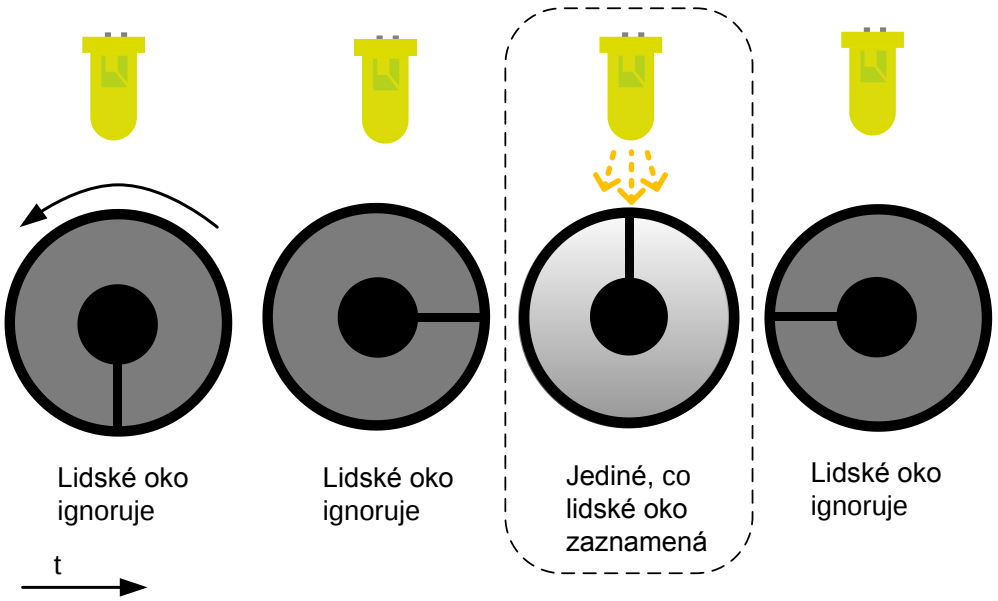


Obr. 31 – Blokové schéma dema s BLDC motory

Použité BLDC motory jsou z CD/DVD mechanik a mají na sobě připevněné DVD, aby bylo lépe vidět, že se motory točí synchronně se stejnou fází. Použití právě BLDC motorů pro demo vyplynulo z potřeb katedry měření, ale nese s sebou několik komplikací. Základním problém je fakt, že tyto motory nejsou primárně stavěny na otáčení s konkrétní fází. Jsou spíše stavěny pro regulaci na danou rychlost, protože jejich rychlost je řízena velikostí napětí resp. proudu protékajícím jejich cívkami. Regulace na konkrétní fázi se musí vyladit relativně přesně přes PI regulátor, přestože rozlišení snímané fáze motoru je 36 „kroků“ na otáčku. Pro snadnější

implementaci tohoto dema by se více hodily krokové motory, které jsou přímo konstruované na řízení fáze.

Další nepříjemností je minimální rychlost otáčení BLDC motorů. Motor totiž potřebuje, aby se plynule otáčel a nezastavoval se na každém kroku. Z této skutečnosti plyne fakt, že není prakticky možné motor udržovat v malých otáčkách tak, aby lidské oko snadno rozpoznalo, že se oba motory v demu točí se stejnou fází. Pro dobré řízení použitých BLDC motorů je vhodné s nimi točit otáčkami minimálně 10 ot/s. Při takové rychlosti však lidské oko nezaznamená fázi motorů. Z těchto důvodů relativně vysokých otáček bylo třeba využít tzv. stroboskopický jev, který je znázorněný na obr. 32.



Obr. 32 – Využitý stroboskopický jev

K točícímu se motoru jsou přidány ještě LED[29], které 1x za otáčku jasně krátce zasvítí na DVD připevněné k motoru. Vzhledem k tomu, že lidské oko zaznamenává světlo integrálně, zaznamená pouze obraz, na který bylo posvíceno. Všechny LED jsou napojeny přímo na referenční synchronizační puls z jednoho TriggerBox Lite.[30] Díky tomu je vidět, zda mají motory stejnou fázi a zda jsou správně regulovány na fázi (správný stav je takový, že oko zaznamená čáry nakreslené na DVD v počáteční pozici).

[29] LED – Light Emitting Diode. Dioda emitující světlo

[30] Informace o tom, že jsou LED napojeny přímo na výstup TriggerBox Lite je zjednodušením pro názornost. Ve skutečnosti jsou LED napojeny na výstup modulu STM32F0 z důvodu ovládání délky impulsu pro svícení LED, ale náběžná hrana pulsu pro LED proběhne prakticky ve stejný čas (s drobným zpožděním) jako náběžná hrana pulsu z TriggerBox Lite.

Demo s BLDC motory je založeno na bakalářské Ing. Vavrouše [2], která se z velké části zabývá tvorbou desky budičů pro BLDC motor z CD mechaniky. Ing. Vavrouš pro demonstrační úlohu poskytl:

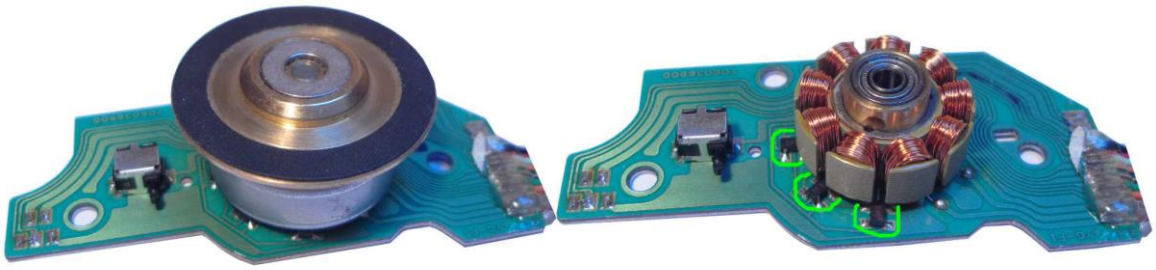
- 2x osazenou desku budičů BLDC motorů
- 1x BLDC motor se vstupy a výstupy připravenými pro budič BLDC motorů
- Kód do STM32F0 poskytující základní driver pro roztočení motoru

Popis výše uvedených položek je uveden v kapitole 3.2.3.

Všechno ostatní – firmware pro STM32F0 ovládající motor včetně regulace, PC aplikace a zajištění druhého BLDC motoru (identifikace a příprava pro napojení na BLDC budič) jsou předmětem této diplomové práce.

6.2.1 Příprava BLDC motoru a identifikace jeho vstupů a výstupů

První důležitou částí implementace byla příprava BLDC motoru pro demo. Vybrán byl motor ze staré DVD mechaniky, který se svými vlastnostmi blížil motoru dodanému Ing. Vavroušem. Pří výběru bylo z důvodu zjednodušení identifikace vstupů a výstupů třeba klást důraz na to, aby měl motor snímací hlavu s permanentními magnety, a aby měl Hallovy sensory. DVD mechaniky totiž obsahují 2 základní typy BLDC motorů: s Hallovými sensory (pro tzv. senzorové řízení) a bez Hallových sensorů (pro tzv. bezsenzorové řízení). Pro tuto práci bylo zvoleno senzorové řízení s Hallovými senzory, protože umožňuje jednodušší detekci polohy motoru.

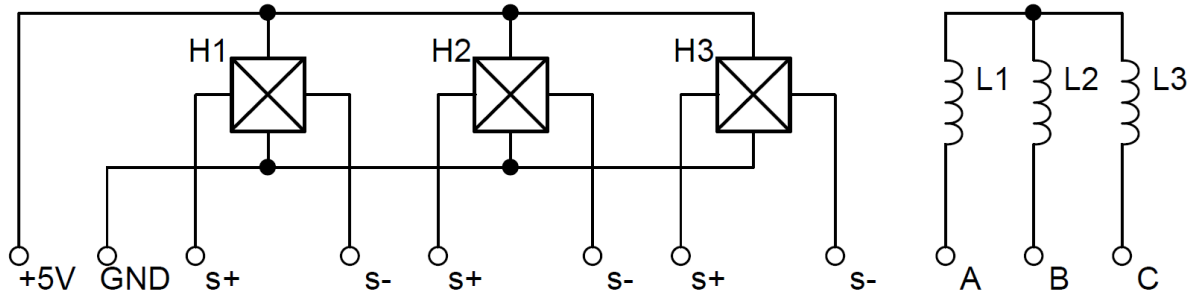


Obr. 33 – BLDC z CD mechaniky a Hallovy sensory[31]

Díky výběru vhodného BLDC motoru proběhla identifikace vstupů a výstupů relativně jednoduše, protože po sejmutí hlavy motoru by na plošném spoji vidět dráhy kontaktů. Nejširší dráhy vodičů na plošném spoji vedou k 3 vinutím motoru. Další 2 vodiče vedou ke spínači, který byl potřeba pro funkci DVD mechaniky, ale pro samotný BLDC motor potřeba není. Všechny zbylé vodiče vedou k Hallovým sensorům. Tyto sensory mají napájení zapojené

[31] Převzato z [2]

paralelně, proto pro přiřazení vodičů konkrétním Hallovým sensorům postačil obyčejný multimetr a pohled na plošný spoj[32]. Schéma zapojení potřebných součástek ukazuje obr. 34.



Obr. 34 – Schéma zapojení použitého motoru a Hallových sensorů[33]

Pro takto identifikovaný BLDC motor bylo potřeba napájet konektory na desku budičů BLDC motoru a přiřadit správnému výstupu budiče správný vstup motoru. Šlo zde zejména o správné přiřazení vinutí A,B,C a jednotlivých Hallových sensorů tak, aby pořadí souhlasilo s tím, jak má. Tato identifikace byla provedena tak, že se motorem otáčelo a pozorovaly se stavy výstupů Hallových sensorů. K tomuto pozorování posloužila aplikace Ing. Vavrouše pro autodetekci stavů BLDC motoru. Těmito kroky byly přípravy druhé BLDC motoru ukončeny.

6.2.2 Implementace firmware pro STM32F0 jako řídící jednotku BLDC motoru

Základní specifikací pro firmware modulu STM32F0 jsou požadavky na to, aby

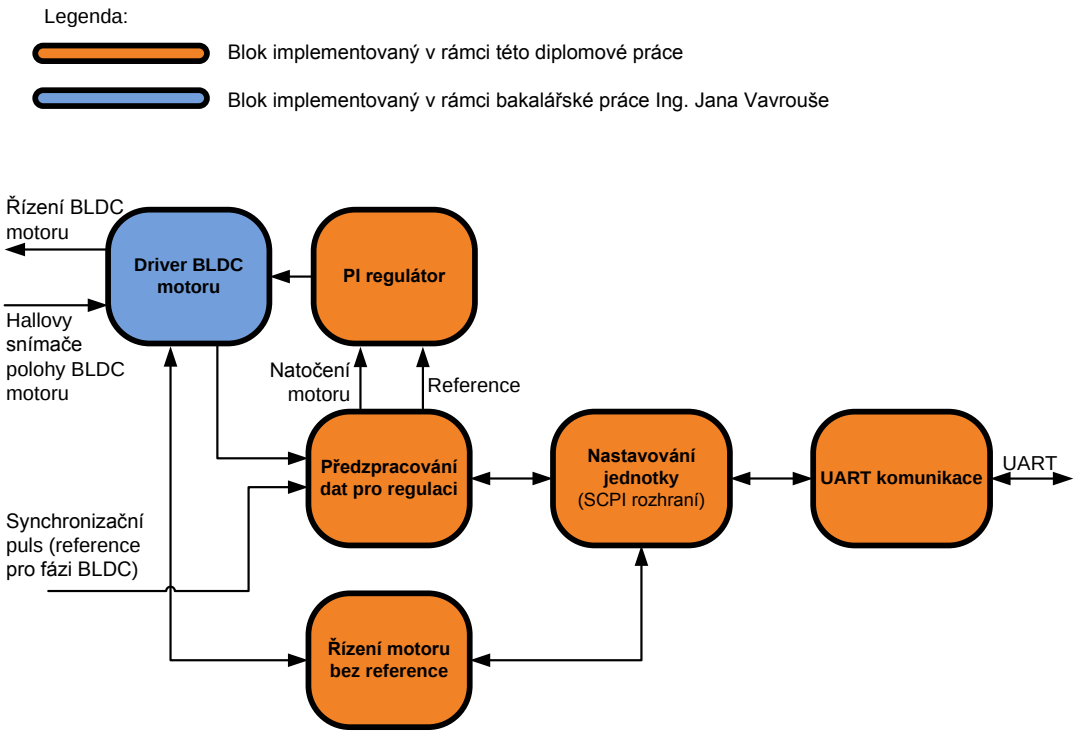
- jednotka otáčela s BLDC motorem
- přijímala referenční signál, který bude určovat počet otáček a jejich fázi
- umožňovala nastavování přes UART, konkrétně SCPI rozhraní

Po analýze těchto požadavků lze nakreslit základní strukturu programu uvedenou na obr. 35.

UART komunikace se SCPI rozhraním je implementované stejnými principy a C knihovnou jako pro voltmetr. Popis této C knihovny je v kapitole 6.1.2. Využívá se zde jen část možností této knihovny a to přijímání dat z UART a následné zpracování SCPI příkazů.

Zbylé bloky v levé části obr. 35 se věnují pouze práci s motorem. Veškerá práce s motorem prochází přes blok Driver BLDC motoru, který základně předzpracovává potřebná data pro motor a ze senzorů. V rámci této diplomové práce byla snaha tento driver upravovat minimálně – spíše opravit drobné chyby nebo provést nutné optimalizace, jinak s ním bylo pracováno jako s knihovnou třetí strany. Vzhledem k tomu, že obsahuje skutečně velice základní API pro práci s motorem, tak to nebyl problém.

[32] V některých motorech se Hallovy sensory zapojují do série. To znepříjemňuje měření multimetrem, protože nelze snadno identifikovat +5V a GND
[33] Převzato z [2]



Obr. 35 – Blokové schéma jednotky pro řízení BLDC motoru

Dalším jednodušším blokem je Řízení motoru bez reference. Jedná se spíš o funkcionality používané k základnímu ladění, protože se stará o:

- Točení motorem na jednu nebo druhou stranu s pevným řídícím napětím motoru (přesněji pevnou řídicí střídou signálu pro ovládání motoru. Více v kapitole 3.2.3)
- Točení motorem na daný počet otáček jedním nebo druhým směrem
- Funkce zastavení motoru
- Funkce „volnoběh“. Tzn. do vedení motoru není generován řídicí proud

Výše uvedené funkcionality by se daly případně použít pro jinou demonstrační úlohu, kde by BLDC motory otáčely šroubovicí, který by hýbala s pojezdem. Stačilo by tam rozlišení na celé otáčky. Pojezdy by se pak synchronně pohybovaly vpřed a vzad.

Posledními bloky jsou PI regulátor a Předzpracování dat pro regulaci. Tyto bloky se starají o hlavní funkci jednotky – řízení motoru tak, aby sledoval otáčky a fázi dle referenčního signálu. Přímo o regulaci řídícího signálu motoru se stará PI regulátor, který porovnává absolutní fázi[34] motoru s požadovanou fází motoru dle referenčního signálu. Tento regulátor je vyladěný na referenční signál s konstantní periodou, tzn. regulátor není připraven na signál, který rychle mění periodu, protože v zadání dema je kladen požadavek hlavně na fázi otáčení motoru, kterou jako jedinou dokáže pozorovatel dema vyhodnotit.

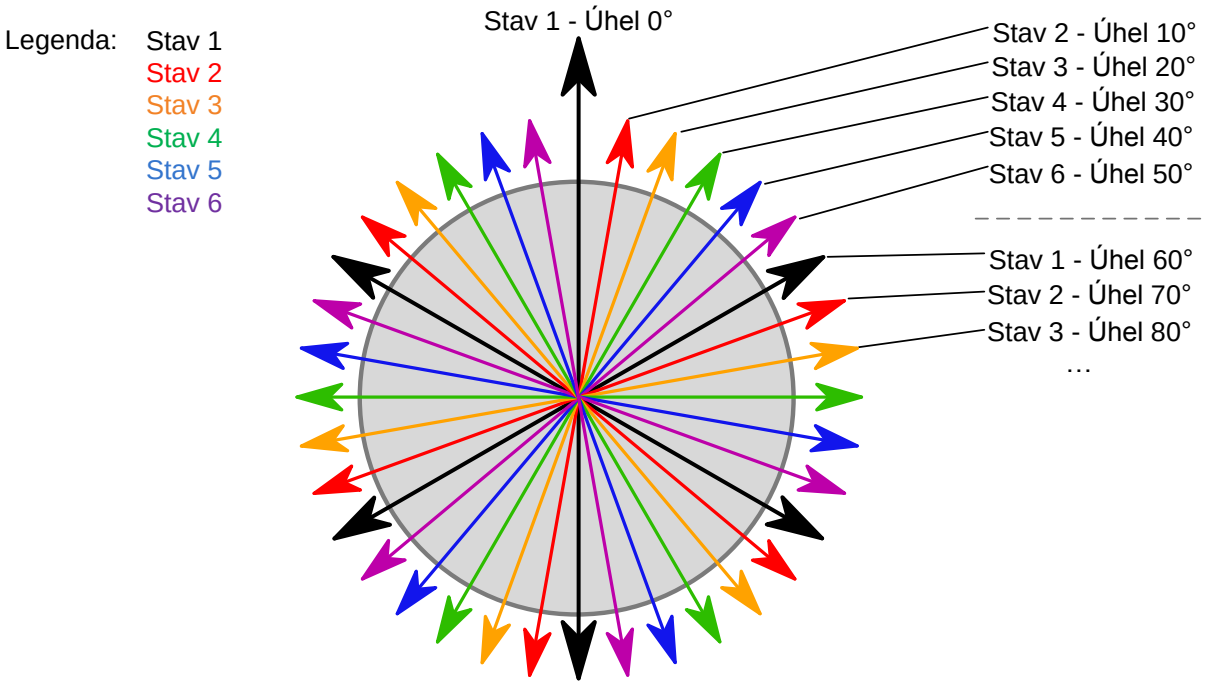
[34] Absolutní fází motoru se zde myslí absolutní úhel, o který se motor otočil od počáteční hodnoty.

Regulátor pracuje kromě absolutní saturace řídícího signálu (maximální a minimální řídící signál) i s vypínáním integrační složky pro situace, kdy se absolutní fáze motoru liší od požadované fáze o více než 5 otáček. Integrační složka je klíčová pro přesnou regulaci na fázi motoru, ale při velkém rozdílu od požadované fáze způsobuje kmitání systému. Díky vypínání integrační složky se systém relativně rychle ureguluje na požadované otáčky a následně přesně ureguluje na požadovanou fázi.

Blok předzpracování dat pro regulaci je v jednotce nutný kvůli rozlišení veličin měřených pro regulaci – konkrétně kvůli rozlišení referenčního signálu (ten generuje jednu náběžnou hranu pro každou požadovanou otáčku) a rozlišení snímaného natočení motoru (zde je rozlišení 36 poloh na otáčku). Blok předzpracování dat pro regulaci přepočítává oba tyto signály na rozlišení jednoho stupně (tzn. 360° na otáčku) pomocí časových odhadů. Toto rozlišení výrazně zlepšuje možnosti regulace. U referenčního signálu se předpokládá, že je jeho perioda konstantní. Díky tomu je možné tuto periodu snadno změřit a vydělit 360° a tím získat požadovanou délku trvání otočení motoru o 1°.

Pro odhad natočení motoru se opět měří délka trvání otočení motoru o jeden stav (motor má 36 stavů na otáčku). Tato délka trvání stavu se vydělí 10. Jelikož motor nemění svou rychlost (a tedy délku trvání stavu) skokově ani se zde nevyskytuje šum nebo překmity, stačí délku trvání stavu odměřit vždy z posledního předešlého stavu. Zbytečné průměrování by zhoršilo odezvu regulace na změny.

Nevýhodou detekce natočení pouze z Hallových senzorů je fakt, že senzory dokážou detekovat pouze stav, ve kterém z 6-ti řídicích stavů se BLDC motor nachází. Těchto 6 stavů se ovšem v otočení o 360° opakuje 6 krát. Tuto problematiku znázorňuje následující obrázek.



Obr. 36 – Stavy natočení motoru, které měří Hallovy senzory

Je tedy patrné, že z Hallových senzorů není možné zjistit, zda je motor natočen o 0° nebo 60°, protože v obou případech naměří Hallovy senzory stav 1. Program proto měří pouze úhel natočení od spuštění dema – tedy relativní úhel a ne absolutní. Proto je vhodné před spuštěním dema nastavit motory do stejné pozice, aby měly stejný absolutní úhel.

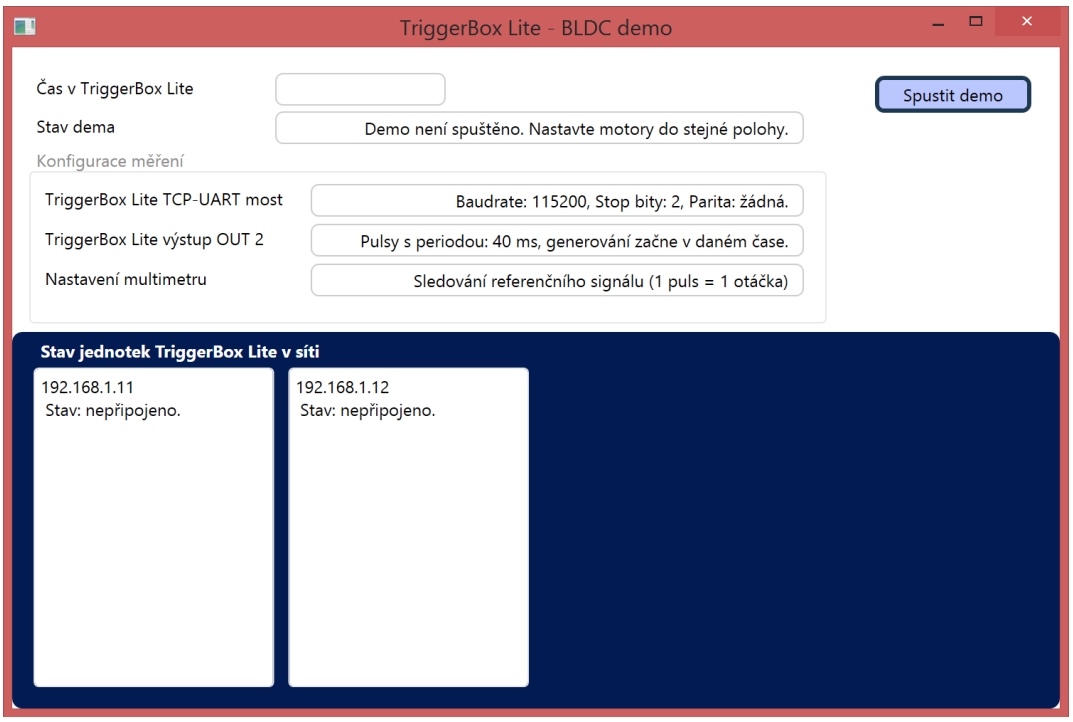
Struktura popsaného firmware je následující:

- Main – pouze základní definice a spouštění programu
- Bldc_regulace – hlavní knihovna aplikace, která se stará o řízení a regulaci motoru
- BLDC_scpi – definice SCPI pro knihovnu Scpi-parser [8]. Jedná se o SCPI callback funkce a nastavení Scpi-parseru
- Test – knihovna obsahující testy jednotky
- Common_libs/Vmet_usart – knihovna pro USART komunikaci
- BLDC_driver/bldc – knihovna BLDC driveru od Ing. Vavrouše

6.2.3 PC aplikace pro BLDC demo

Pro demonstrační úlohu bylo opět potřeba implementovat PC aplikaci, která se stará řízení jednotek v distribuovaném systému. Základ aplikace byl převzat z dema pro voltmetry. Oproti aplikaci dema s voltmetry tato aplikace nevyžaduje periodické nastavování a vyčítání dat.

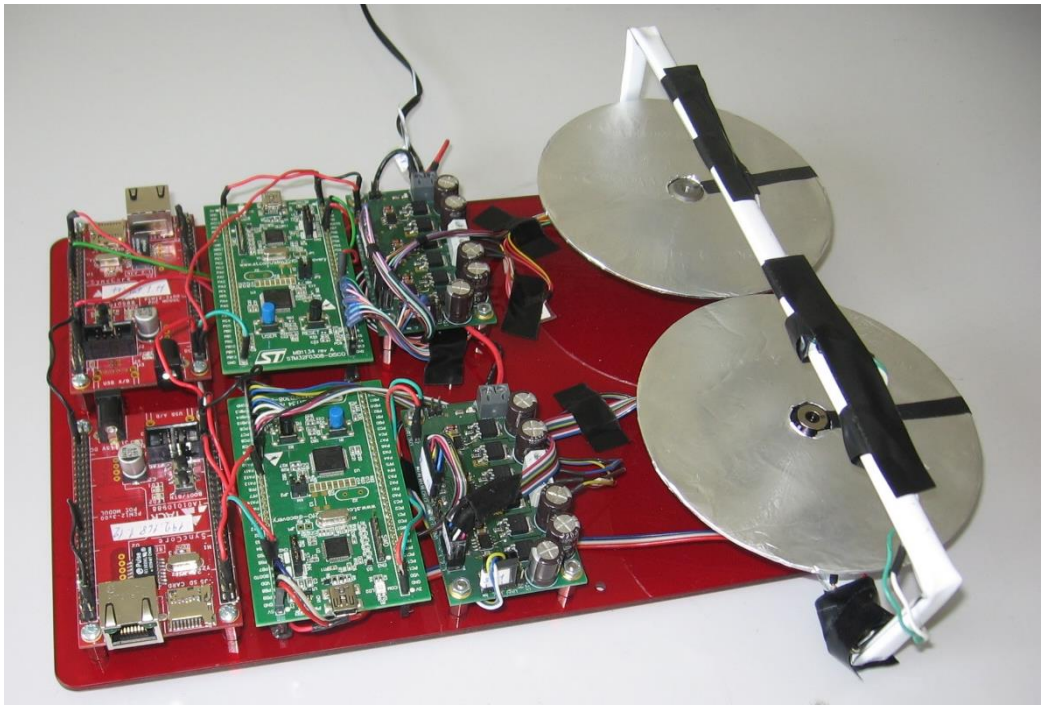
Aplikace se po spuštění připojí k jednotkám v síti. Dále nastaví přes komunikační most řídicí jednotky pro BLDC motory a nakonec spustí na TriggerBox Lite jednotkách generování referenčního synchronizačního signálu. Tím nastavovací role aplikace končí a pouze detekuje čas v jednotkách a umožnuje demo zastavit.



Obr. 37 – PC aplikace pro BLDC demo

6.2.4 Výsledné demo

Po přípravě částí dema popsaných v předchozích kapitolách bylo třeba tuto demonstrační úlohu připravit pro možné prezentování na výstavách. Jednotlivé jednotky demonstrační úlohy byly proto připevněny na plexisklo a propojeny. Výsledná podoba dema je uvedena na následujícím obrázku.



Obr. 38 – Výsledná realizace BLDC dema

Před spuštěním dema je potřeba nastavit motory do stejné pozice, jak je vysvětleno v předchozí kapitole. Po spuštění dema je vidět, jak se motory postupně ustalují vůči referenčnímu signálu (dorovnávají fázi) až se dorovnají. Stejnou fázi jsou pak motory schopny dlouhodobě udržovat. Demo nemá problém zajistit dorovnání fáze i po působení vnějších vlivů jako je třeba přibrzdění motoru prstem.

7 Měření a testování

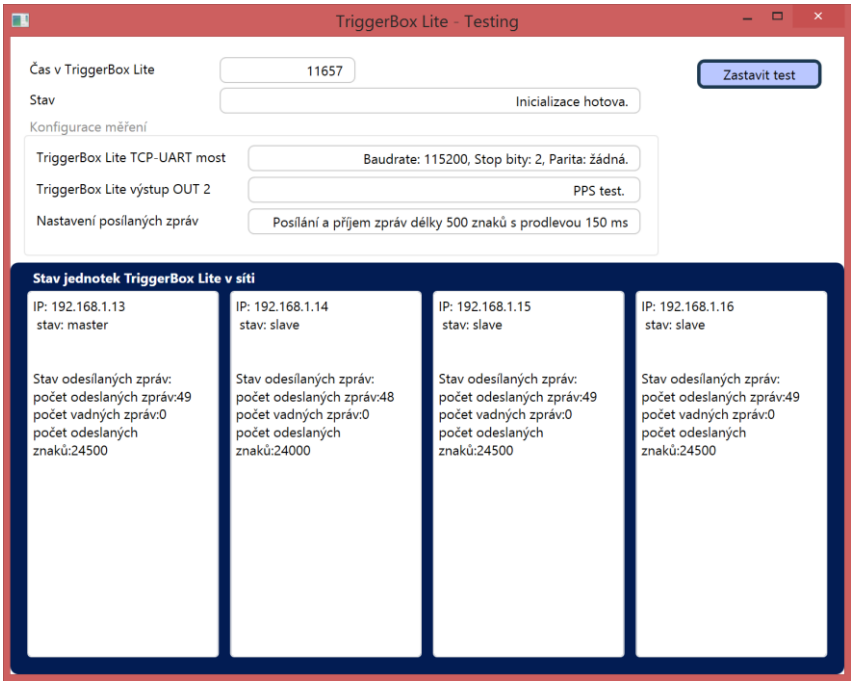
Tato kapitola se zabývá měřením důležitých parametrů TriggerBox Lite potřebných pro využívání jednotky. Jedná se zejména o úroveň synchronizace v různých konfiguracích, které simulují reálné potenciální využití TriggerBox Lite.

7.1 Testovací aplikace

Pro reálnější simulaci různých testovacích a měřicích scénářů byla vytvořena testovací C#/WPF aplikace. Jako základ knihoven a uživatelského rozhraní byly použity již implementované C# aplikace (ovladací aplikace TriggerBox Lite a aplikace pro demonstrační úlohy).

Aplikace umožňuje pro libovolný počet nastavených jednotek v síti následující testovací a měřicí scénáře:

- Nastavení signálů na výstupu OUT 2. Konkrétně
  - PPS
  - Naplnění fronty generovaných událostí tak, aby se vygenerovala sekvence náběžných hran s daným odstupem
- Simulace zatížení komunikačního mostu při komunikaci s podřízenou jednotkou.
  - Je implementováno pro echování, tzn. na TriggerBox Lite je výstup UART mostu napojený na vstup. Tím pádem to, co je odesláno přes most, je také přijato. Aplikace následně vyhodnocuje shodnost odeslaných a přijatých zpráv. Simulace reálného zatížení je implementována tak, že se data odesílají na jednotlivé jednotky postupně, ne paralelně. Jedná se tedy o situaci, kdy se provádí komunikace typu dotazování a čekání na odpověď, která je pro využití TriggerBox Lite typická.



Obr. 39 – Testovací aplikace TriggerBox Lite – Testing

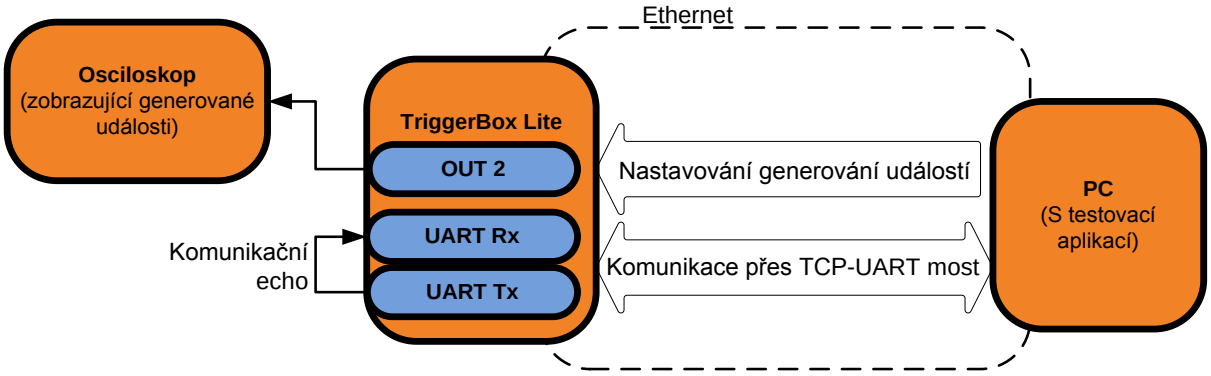
Aplikace je snadno konfigurovatelná přes C# konstanty a rozšiřitelná pro další typy testů v případě potřeby.

7.2 Měření generování událostí přes frontu

Generování událostí bylo implementováno přes software frontu a FreeRTOS vlákna. Je proto potřeba klást zvláštní důraz na otestování minimální prodlevy mezi generovanými událostmi.

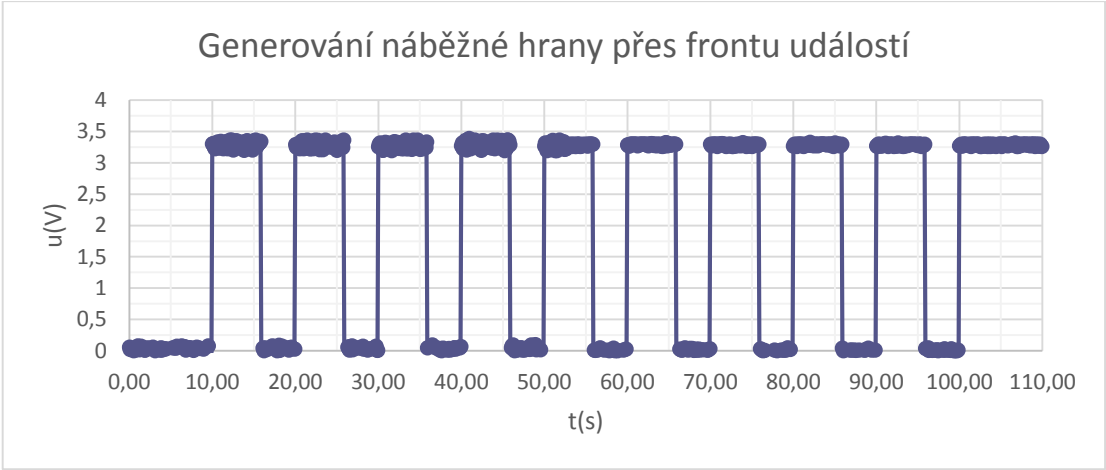
Pro změření této prodlevy byla s pomocí testovací PC aplikace vytvořena situace, kdy je TriggerBox Lite pod zátěží komunikačního mostu (zprávy o délce 500 znaků s periodou

odesílání cca 200 ms) a testovalo se, jakou prodlevu mezi událostmi umožní. Schéma měření ukazuje následující obrázek.



Obr. 40 – Schéma měření generování událostí přes frontu

TriggerBox Lite začal spolehlivě generovat všechny naplánované události od prodlevy 10 ms. Pro menší prodlevy nastávala situace, kdy jednotka některou událost vynechala. Tato minimální prodleva je způsobená plánováním FreeRTOS jak je popsáno v kapitole 5.2. Obr. 41 ukazuje záznam z osciloskopu pro 10 náběžných hran (10 je velikost fronty událostí, tzn. maximální počet) s prodlevou 10 ms naplánovaných přes frontu.



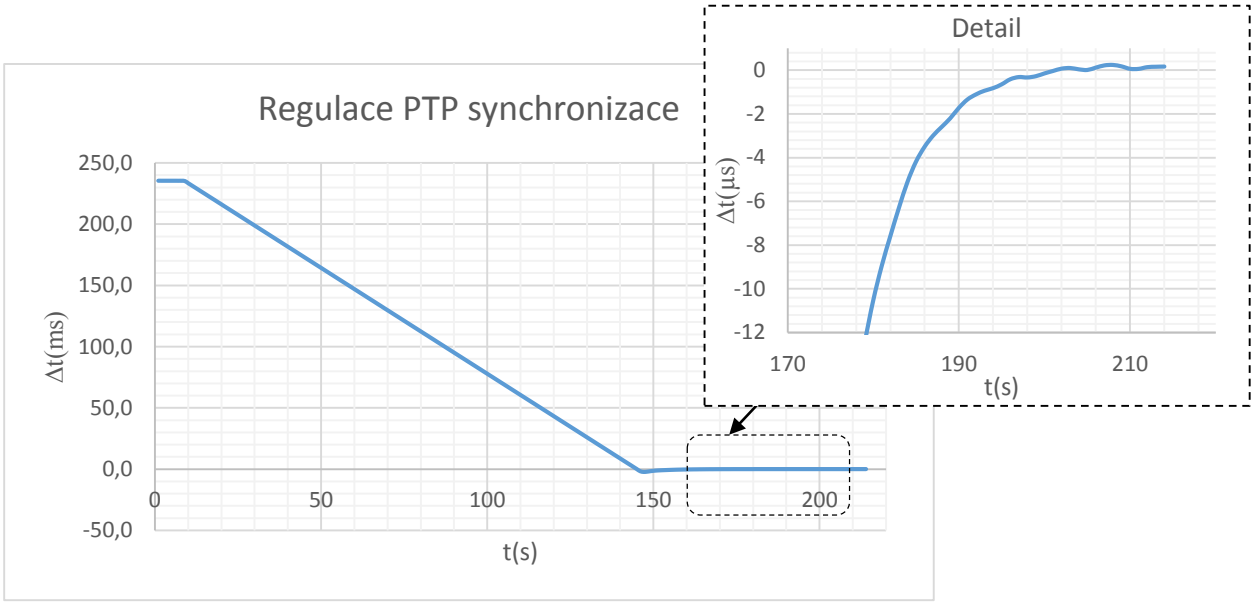
Obr. 41 – Záznam z osciloskopu pro události generované přes frontu

Prodleva 10 ms byla naměřena jako nejmenší fungující. Proto je třeba do specifikace uvést hodnotu včetně rezervy, tzn. 2x větší. Minimální spolehlivá prodleva mezi naplánovanými událostmi je tedy 20 ms.

7.3 Měření regulace synchronizační odchylky s využitím PTP

Synchronizace s využitím PTP protokolu používá ke srovnání času na PTP slave jednotce s PTP master jednotkou změnu frekvence interních hodin jednotky. Jen při velkém rozdílu hodin dojde ke skokové změně času. Pokud dochází k postupné synchronizaci přes změnu frekvence interních hodin (tedy bez skokové změny času), může tato operace trvat. Pro potenciální aplikace TriggerBox Lite je důležité, jak rychlá je regulace synchronizační odchylky, tzn. jak dlouho se musí čekat na ustálení po zapnutí jednotek.

Pro toto měření byly do sítě zapojeny 2 jednotky TriggerBox Lite, které byly spuštěny v podobný čas (tzn. synchronizační odchylka mezi nimi byla cca do 0,5 s a nedošlo tak ke skokové změně času). Jednotkám bylo zapnuto generování PPS pulsů. Na čítači pak šlo pozorovat, jak se synchronizační odchylka zmenšovala.



Obr. 42 – Regulace PTP synchronizace

Předchozí graf dobře znázorňuje průběh synchronizace. Je zde patrné, že synchronizace časového rozdílu 238 s trvá 145 s na přesnost v řádu milisekund. Následně je pak třeba dalších 60 s pro synchronizaci na řád desetin mikrosekund, kde se synchronizační odchylka ustálí.

V jednotce je nastaven minimální synchronizační offset pro časový skok 1 s. Tzn. jednotky s odchylkou synchronizace do 1 s se postupně dorovnávají změnou interních hodin. Doba regulace se dá spočítat následovně.

Doba zmenšení synchronizační odchylky o 1 ms:

$$T_{\Delta 1ms} = \frac{T_{ustal\_ms}}{\Delta T_{synchr}} = \frac{145}{238} = 0{,}61\ s$$

Maximální synchronizační odchylka bez skoku je ΔT_{max} = 1000 ms. Doba regulace synchronizační odchylky z hodnoty 1 ms na 1 µs je $T_{\Delta 1\mu s}$ 50s.

Maximální doba ustálení synchronizace jednotek T_{max} je tedy:

$$T_{max} = \Delta T_{max} \cdot T_{\Delta 1ms} + T_{\Delta 1\mu s} = 1000 \cdot 0{,}61 + 50 = 660\ s$$

Znamená to tedy, že je vhodné jednotky nechat po zapojení do sítě ustálit po dobu 11 s, aby bylo jisté, že jsou již zasynchronizovány. Čas by se dal zmenšit nastavením jednotky na častější synchronizační skoky, ale ty jsou pro aplikace TriggerBox Lite nežádoucí. Při časovém skoku

totiž mohou být přeskočeny některé naplánované výstupní události nebo porušena jedna perioda periodických výstupních událostí.

7.4 Měření úrovně synchronizace

Měření úrovně synchronizace jednotek TriggerBox Lite se zaměřuje na jednotlivé případy, ve kterých může být jednotka nasazena. Následující kapitoly proto obsahují výstupy z měření synchronizace pro situace:

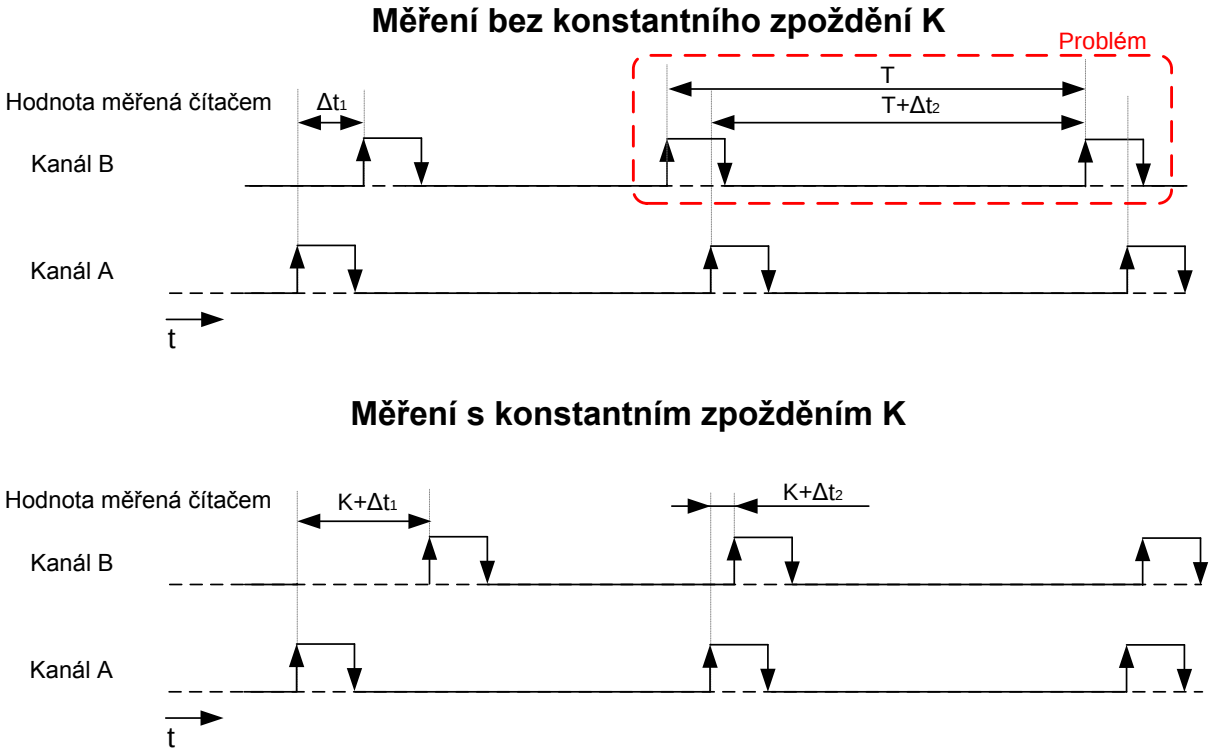
- Měření 2 jednotek v síti – pro základní pojem o tom, jak na synchronizaci působí spojovací uzle sítě
    - bez switche nebo jiného spojovacího uzlu
    - s běžným Ethernetovým switchem
    - s využitím optické linky – případ sítě na velkém prostoru
- Měření 3 jednotek v síti – ukázka, jak se projeví další jednotka v síti. V této situaci je také porovnána synchronizace při a bez vytížení TriggerBox Lite jednotek zprávami
    - s běžným switchem
    - se switchem podporujícím PTP Transparent Clock funkcionalitu
- Měření 6 jednotek v síti – ukázka větší sítě. Opět zde je porovnávána situace s a bez zátěže zprávami
    - pouze s běžným switchem. Dostupný PTP switch má jen 4 Ethernetové sloty, proto nemohl být použit.

V sítích, kde je víc jak 2 jednotky, je vždy měření prováděno PTP master vůči PTP slave jednotce. Toto omezení je zde z toho důvodu, aby se dalo odhadnout maximální zpoždění PTP slave – PTP slave. Zpoždění PTP slave – PTP slave může být totiž až dvojnásobkem oproti PTP master – PTP slave.

Ve výsledcích jsou kromě histogramů rozdílů hodin jednotek také uvedeny 3 hodnoty: průměrná hodnota rozdílů hodin $\bar{x}$, směrodatná odchylka rozdílů hodin *s* a maximální odchylka rozdílů hodin od střední hodnoty rozdílů $x_{max-\bar{x}}$. (dále jen „maximální hodnota“). Nejdůležitějším parametrem je směrodatná odchylka rozdílů hodin *s*, protože porovnáváním její hodnoty lze porovnávat úrovně synchronizace v různých konfiguracích. Střední hodnota $\bar{x}$ je daná asymetrií sítě a ta se v dané konfiguraci sítě bude lišit u každého porovnávaného páru TriggerBox Lite v síti. Z uvedených důvodů není střední hodnota rozdílů hodin brána v potaz při porovnávání jednotlivých situací. Posledním parametrem je maximální odchylka rozdílů hodin od střední hodnoty rozdílů $x_{max-\bar{x}}$ (maximální hodnota), která ukazuje maximální chybu synchronizace bez projevů asymetrie sítě. Tento parametr je uveden v absolutní hodnotě.

Všechna měření jsou prováděna čítačem Agilent 53131A přes časové odchylky PPS signálů jednotek. Každé měření bylo zpracováno z 2840 hodnot (tzn. každé měření běželo 47 minut). Na čítači byla využita měřicí funkce Time Interval A-B. Tzn. měří se časový interval od příchodu

náběžné hrany na kanálu čítače A do příchodu náběžné hrany na kanálu B. Při použití tohoto principu se objevuje problém, který je naznačen na obr. 43. Pokud náběžná hrana na kanálu A proběhne až po hraně na kanálu B, nezměří čítač záporný časový rozdíl signálu. Změřená hodnota pak odpovídá periodě signálu + časovému rozdílu signálů. Aby se tomuto problému předešlo, bylo k signálu na kanálu B přidáno konstantní zpoždění K (generování PPS se na TriggerBox Lite připojenému ke kanálu B nastavilo tak, aby začalo o čas K později než na TriggerBox Lite připojenému ke kanálu A). Po odečtení konstantního zpoždění K pak hodnota odpovídá časovému rozdílu hodin jednotek. Časový rozdíl K byl stanoven na 50 µs (musí být co nejmenší, aby se mnoho neprojevily nejistoty čítače způsobené nepřesností interního oscilátoru).



Obr. 43 – Měření časového intervalu A-B na čítači

Rozlišení čítače Agilent 53131A je pro měření časového intervalu podle manuálu 0,75 ns. Podle datasheetu tohoto čítače [9] lze odhadnout nejistotu typu B u měření časového intervalu vypočte dle vzorce:

$$u = T_{BaseErr} \cdot T_i + T_{triErr} + 1{,}5$$

Kde:

$T_{BaseErr}$ je chyba časové základny čítače.

T_i je změřený časový interval. U uvedených měření je to 50 µs.

T_{triErr} je chyba triggerování, která je závislá na charakteru náběžných hran vstupního signálu. U výstupů TriggerBox Lite byl použit budič výstupů, který umožňuje generovat náběžnou hranu signálu kratší než 0,8 ns. Pro odhad nejistoty je pro hodnotu T_{triErr} brána maximální hodnota – tedy 0,8 ns.

Dále dle datasheetu [9]

$$T_{BaseErr} = T_{age} + T_{temp} + T_{voltage}$$

Kde:

T_{age} je faktor „stárnutí“ oscilátoru čítače. Pro daný čítač je to hodnota $3 \cdot 10^{-7}$.

T_{temp} je faktor teploty okolí. Pro daný čítač v rozmezí 0 – 50 °C je hodnota $5 \cdot 10^{-6}$.

$T_{voltage}$ je faktor síťového napětí. Podle datasheetu [9] se v případě tohoto měření neprojeví, proto je zanedbán.

Díky výše uvedenému je patrné, že

$$T_{BaseErr} = T_{age} + T_{temp} + T_{voltage} = 3 \cdot 10^{-7} + 5 \cdot 10^{-6} + 0 = 5{,}3 \cdot 10^{-6}$$

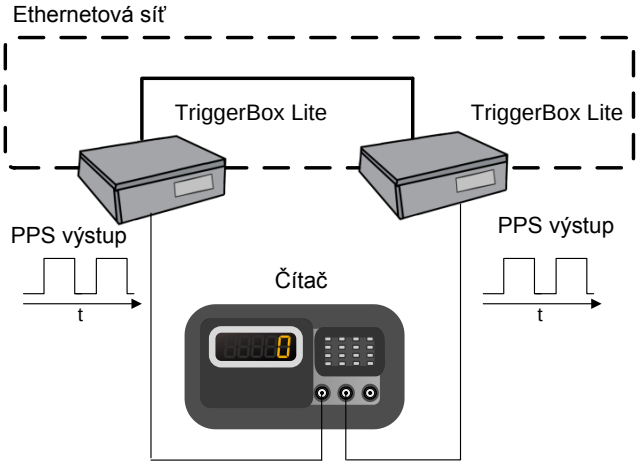
Výsledný odhad nejistoty měření časového intervalu je:

$$u = T_{BaseErr} \cdot T_i + T_{triErr} + 1{,}5 = 5{,}3 \cdot 10^{-6} \cdot 5 \cdot 10^{-5} + 0{,}8 \cdot 10^{-9} + 1{,}5 \cdot 10^{-9}$$
$$= 2{,}6 \cdot 10^{-9} s = 2{,}6\ ns$$

Pro porovnání měřených úrovní synchronizace by však postačovala i menší přesnost – na desítky nanosekund. Uvedená nejistota u jednotlivých naměřených a vypočtených hodnot proto znovu uváděna nebude.

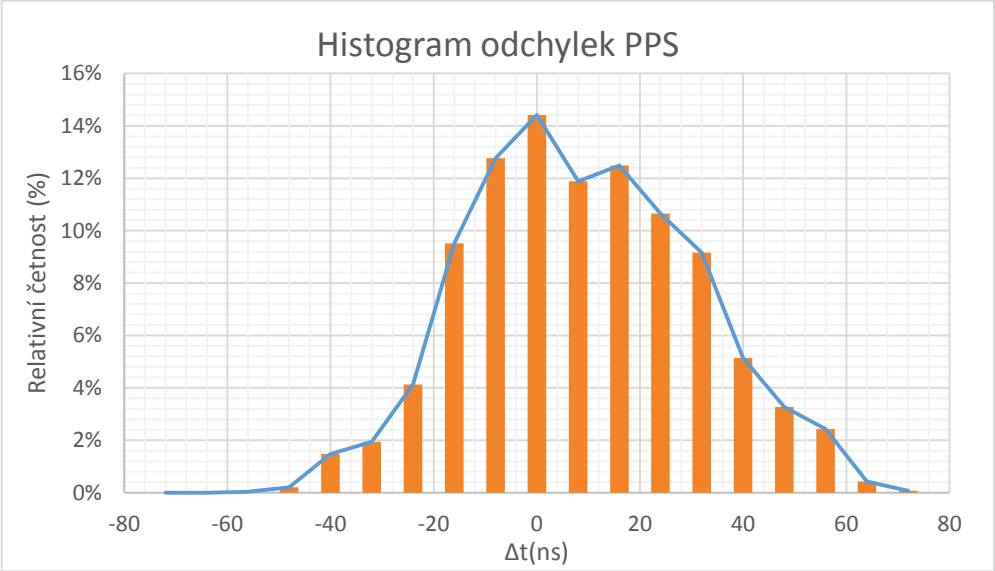
7.4.1 Měření v síti 2 jednotek bez aktivních spojovacích prvků

První situací je měření synchronizace bez aktivních spojovacích prvků (switchů,...). Tzn. v nejjednodušší síti, která není zatížena ničím jiným než PTP synchronizačními zprávami. Jednotky byly nejprve nastaveny pro generování PPS výstupu a následně napojeny napřímo, aby se synchronizovaly. Schéma zapojení měření je na následujícím obrázku.



Obr. 44 – Schéma měření 2 jednotek bez spojovacích prvků

Výsledný histogram rozdílů hodin jednotek TriggerBox Lite na obr. 44 ukazuje prakticky nejlepší možnou úroveň synchronizace, která je minimálně zatížena vlivy chybových faktorů. Rozlišení Δt uvedeného histogramu je 8 ns (je zobrazen s hranicemi tříd 8 ns).



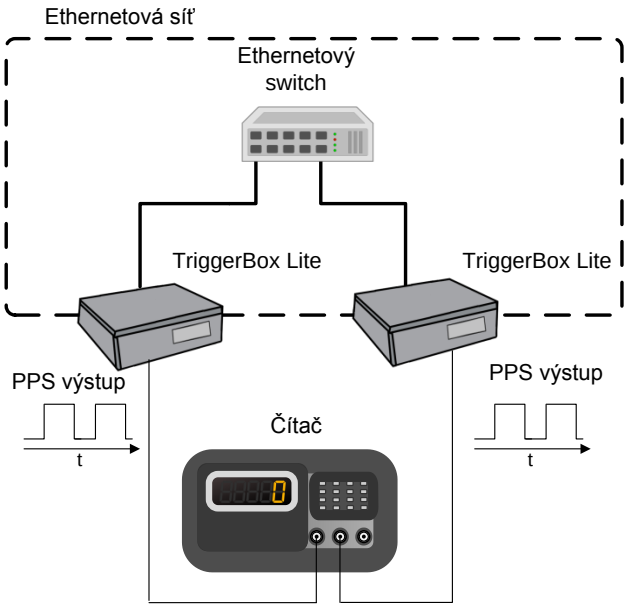
Obr. 45 – Histogram odchylek synchronizace u 2 jednotek bez spojovacího prvku

Z měření vyšly hodnoty synchronizačních rozdílů hodin jednotek TriggerBox Lite v síti.

- Směrodatná odchylka rozdílu hodin jednotek s $= 22$ ns
- Průměrná hodnota rozdílu hodin jednotek $\bar{x} = 5\ ns$
- Maximální hodnota rozdílu hodin jednotek $x_{max-\bar{x}} = 80\ ns$

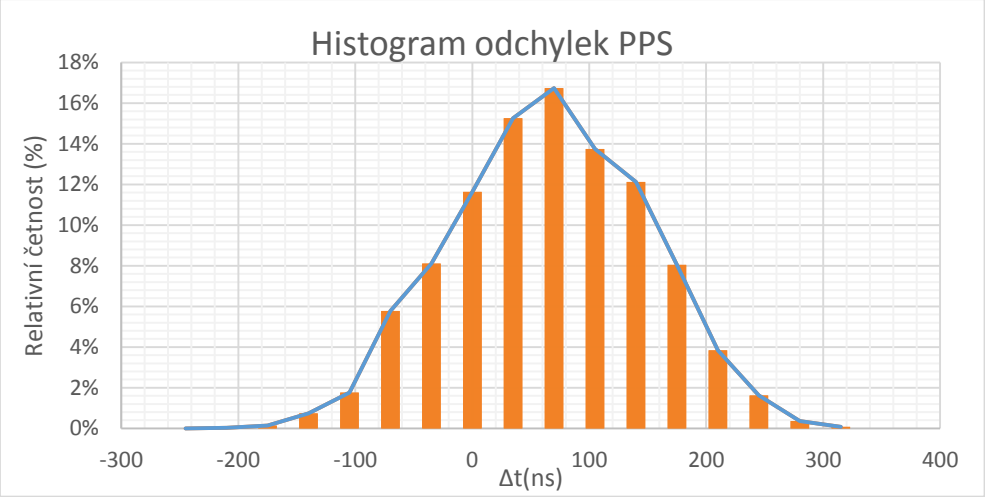
7.4.2 Měření v síti 2 jednotek s běžným Ethernetovým switchem

V tomto případě se jednalo o nejjednodušší síť se spojovacím uzlem, ve které byly 2 jednotky TriggerBox Lite. Ty byly společně v rámci Ethernetu připojeny obyčejným Ethernetovým switchem TP LINK TL F1008P. Jednalo se tedy o switch, který nemá podporu pro PTP zprávy. Podobné zapojení bude použito v případě, kdy je třeba levné síťové řešení. Schéma ukazuje následující obrázek.



Obr. 46 – Schéma měření s Ethernetovým switchem

Histogram rozdílů hodin jednotek je uveden na následujícím obrázku. Je zde vidět velký nárůst směrodatné odchylky, který je způsobený proměnnými zpožděními zpráv na switchy. Dále histogram ukazuje, jak na PTP síť působí asymetrie sítě, která posune graf histogramu na ose x tak, že střed není v 0. Asymetrie se však bude lišit na každém páru koncových uzlů v síti, proto ji není vhodné brát v potaz jako parametr, který by se měl v jednotlivých měřených situacích porovnávat. Důležitou hodnotou je směrodatná odchylka, tedy jak moc se synchronizační chyba, tzn. rozdíl hodin jednotek, mění.



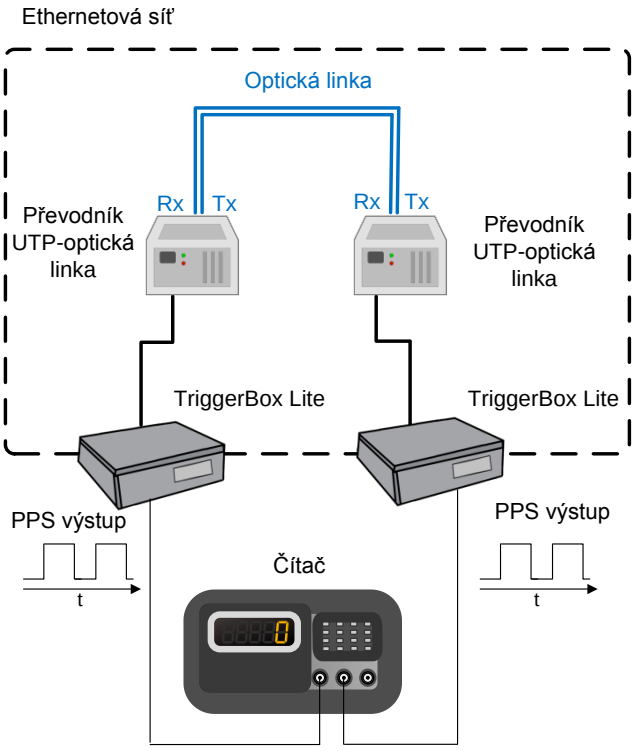
Obr. 47 – Histogram odchylek hodin (PPS) – 2 jednotky, běžný switch

Rozlišení Δt uvedeného histogramu je 35 ns (je zobrazen s hranicemi tříd 35 ns). Z měření vyšly hodnoty rozdílu hodin jednotek.

- Směrodatná odchylka rozdílu hodin jednotek s $= 82$ ns
- Průměrná hodnota rozdílu hodin jednotek $\bar{x} = 48\ ns$
- Maximální hodnota rozdílu hodin jednotek $x_{max-\bar{x}} = 289\ ns$

7.4.3 Měření v síti 2 jednotek s optickými převodníky

Tento scénář měření ukazuje možnosti použití TriggerBox Lite na síti většího rozsahu, kde je třeba použít optické kabely. Zapojení dvou TriggerBox Lite do Ethernetové sítě proběhlo přes 2 převodníky UTP-optický kabel (tzn. signál byl převeden z UTP na optický kabel a následně zpět na UTP). Schéma měření je uvedeno na obr. 48. Pro toto měření bylo použito 2 krátkých optických jednovidových kabelů (Rx, Tx) s délkou 3,5 m a použité optické převodníky byly TP Link MC110CS.



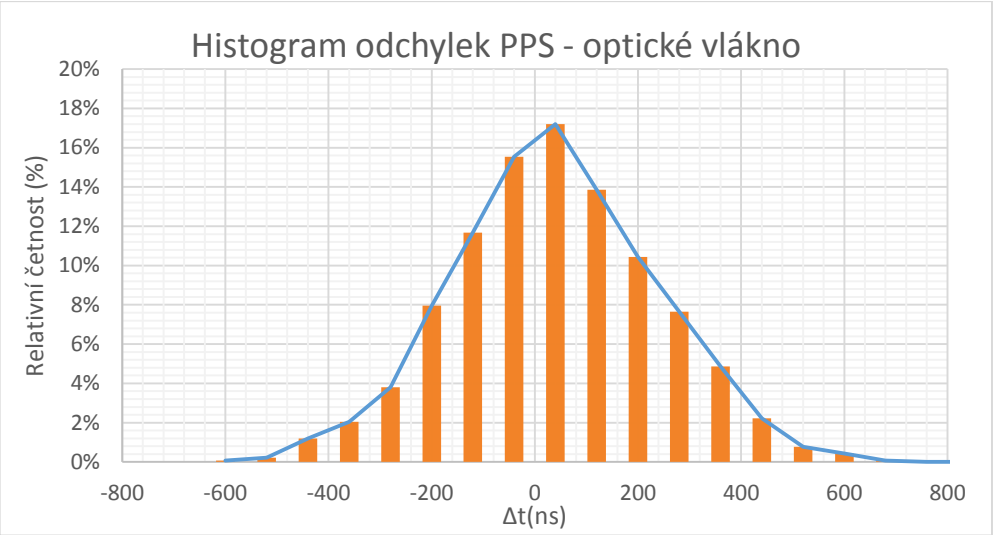
Obr. 48 – Schéma měření s optickými převodníky

Naměřené hodnoty

- Směrodatná odchylka rozdílu hodin jednotek s = 197 ns
- Průměrná hodnota rozdílu hodin jednotek $\bar{x} = 0\ ns$
- Maximální hodnota rozdílu hodin jednotek $x_{max-\bar{x}} = 771\ ns$

Rozlišení Δt uvedeného histogramu na obr. 49 je 80 ns (je zobrazen s hranicemi tříd 80 ns).

Směrodatná odchylka rozdílů hodin 2,5 krát větší než u použití běžné switche. Řádově to odpovídá tedy použití svou switchů v síti. Tento výsledek byl pochopitelně očekávaný, protože v síti s optickou linkou jsou 2 aktivní spojovací uzly (2 převodníky UTP-optická linka).



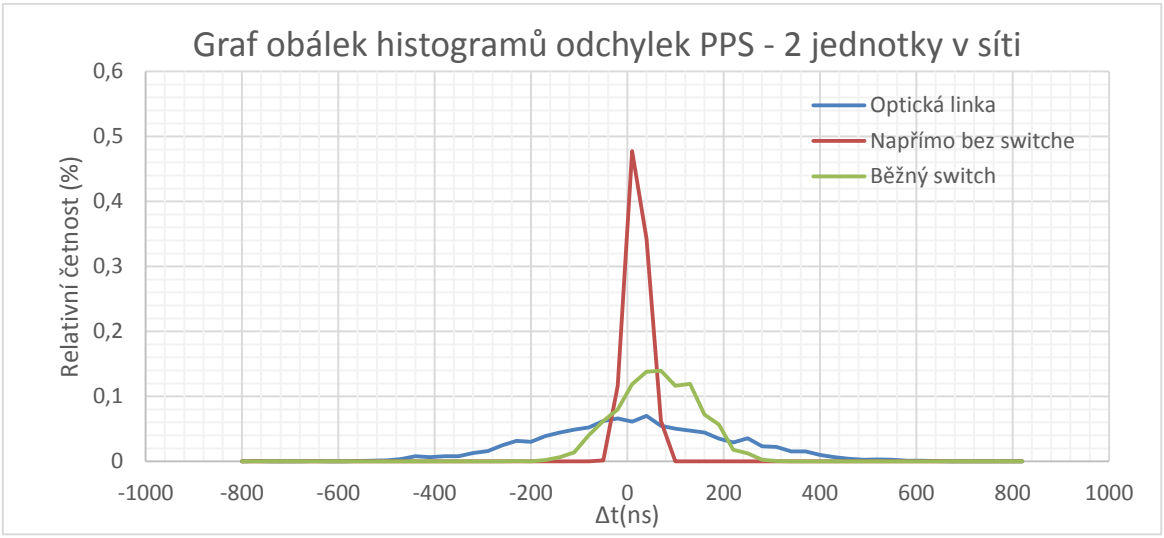
Obr. 49 – Histogram odchylek hodin (PPS) - 2 jednotky, optické převodníky

7.4.4 Porovnání výsledků měření 2 jednotek

Při spojení výsledků z předchozích měření do společných tabulek a grafů dostaneme následující data.[35] Rozlišení Δt uvedeného histogramu na obr. 49 je 30 ns (je zobrazen s hranicemi tříd 30 ns).

Veličina / Měření	Průměrná hodnota $\overline{x}$ (ns)	Směrodaná odchylka s(ns)	Maximální hodnota $x_{max-\bar{x}}$ (ns)
bez spojovacích prvků	5	22	80
s obyčejným switchem	48	82	289
s optickými převodníky	0	197	771

Tab. 1 – Porovnání odchylek hodin (PPS) 2 jednotek v síti



Obr. 50 – Graf obálek histogramů odchylek hodin (PPS) 2 jednotek – porovnání

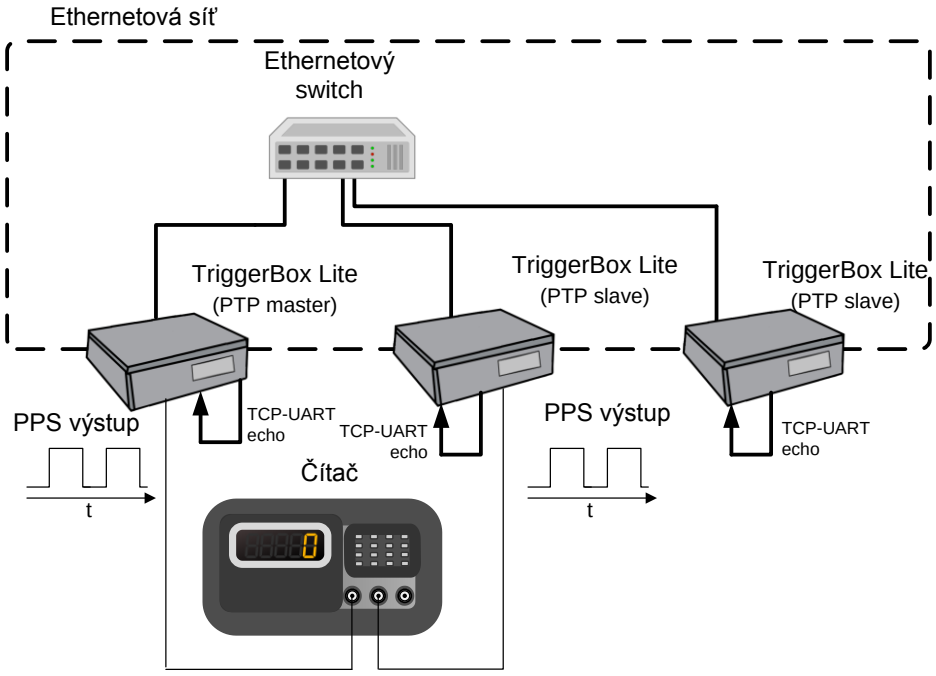
[35] Na grafu jsou uvedeny obálky histogramů, které jsou zde použity pro lepší přehlednost grafu. Hlavním důležitým parametrem je totiž šířka histogramu, která je z jeho obálky dobře patrná.

Jak již bylo řečeno, nejdůležitějším parametrem je směrodatná odchylka, která bude při použití podobné konfigurace sítě prakticky neměnná. Z výsledků je patrné, jak každý aktivní spojovací prvek přidává do PTP sítě proměnné zpoždění (zvyšuje směrodatnou odchylku), které degraduje úroveň synchronizace.

7.4.5 Měření v síti 3 jednotek s různými switchy a zátěžemi

Dalším blokem měření je použití 3 jednotek v síti. Ve všech uvedených měřených scénářích se vyskytuje pouze jeden spojovací prvek – switch. Porovnávají se vlastnosti použití běžného levného switche (TP LINK TL F1008P) a dražšího switche s podporou funkcionality PTP Transparent clock (Hirchmann MS23), který dokáže kompenzovat proměnné zpoždění.

Dalším porovnávaným parametrem je zatížení sítě a jednotek TriggerBox Lite komunikací přes TCP-UART most. Opět se jedná o „echování“ signálu, jak bylo popsáno v kapitole 7.1. Zasílány byly zprávy o délce 500 znaků s periodou odesílání cca 200 ms (jednalo se o sériové odesílání zpráv, tzn. na každou jednotku šla 1 zpráva z 600 ms). Takto nastavené odesílání simuluje běžné měření, které může TriggerBox Lite přes TCP-UART most poskytovat. Schéma měření na následujícím obrázku.



Obr. 51 – Schéma měření s 3 jednotkami

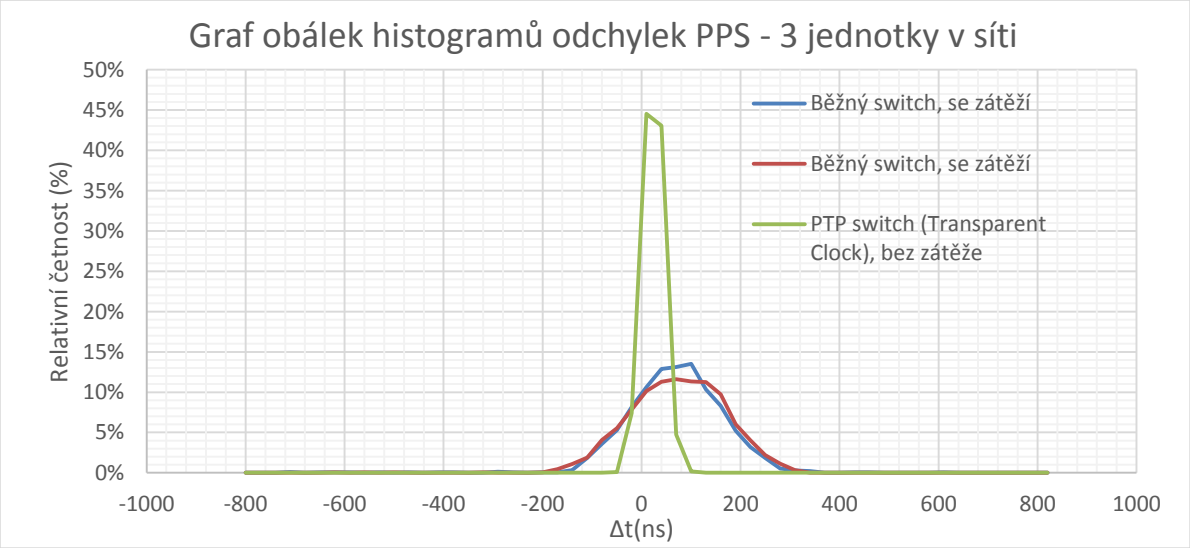
Výsledky měření ukazuje následující tabulka.

Veličina / Měření	Průměrná hodnota $\bar{x}$ **(ns)**	Směrodaná odchylka **s (ns)**	Maximální hodnota $x_{max-\bar{x}}$ **(ns)**
obyčejný switch bez zátěže	57	95	678
obyčejný switch se zátěží	53	98	792
PTP switch bez zátěže	8	20	69

Tab. 2 – Porovnání odchylek hodin (PPS) 3 jednotek v síti

Z hodnot je patrné, že mezi obyčejným switchem a PTP switchem je velký rozdíl. PTP switch vykazuje podobné výsledky jako zapojení jednotek TriggerBox Lite napřímo bez spojovacích prvků v kapitole 7.4.1. Obyčejný switch pro 3 jednotky vykazuje lehce větší směrodatnou odchylku než pro 2.

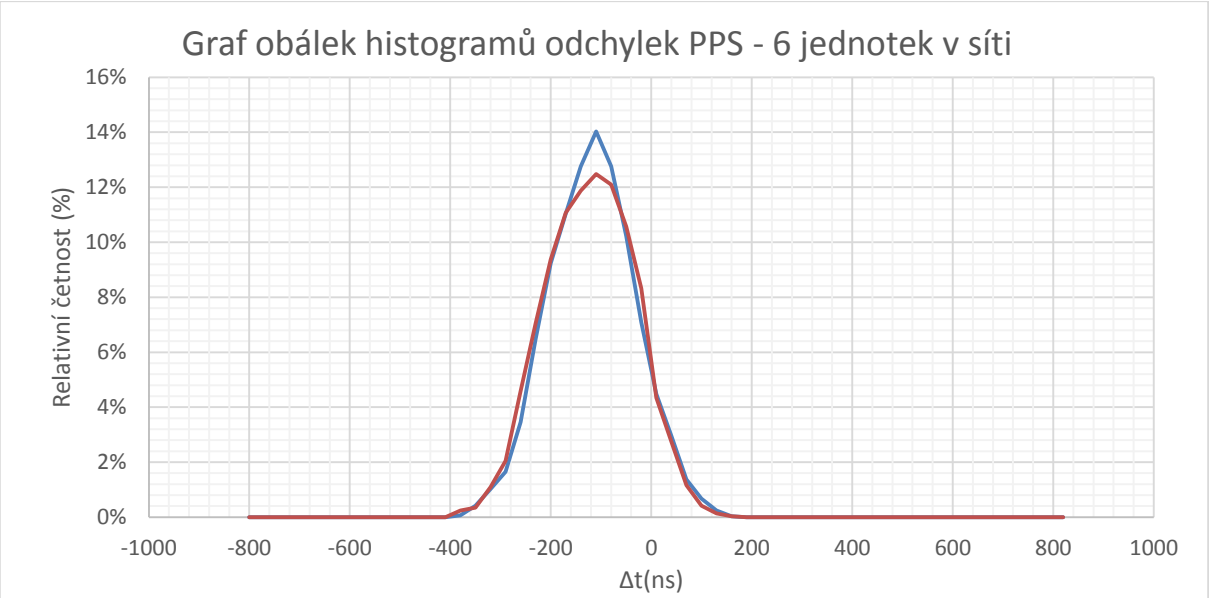
Co se týče porovnání synchronizace se zátěží a bez (zátěží myšleno „echování“ přes TCP - UART most), tak je patrné, že testovaná zátěž nemá na synchronizaci vliv. Hodnoty směrodatných odchylek jsou prakticky stejné. Následující obrázek ukazuje obálky histogramů synchronizačních odchylek. Rozlišení Δt uvedeného histogramu je 30 ns (je zobrazen s hranicemi tříd 30 ns).



Obr. 52 – Graf obálek histogramů odchylek PPS - 3 jednotky v síti

7.4.6 Měření v síti 6 jednotek

Posledním měřením synchronizace je situace větší sítě s 6 jednotkami TriggerBox Lite v síti. Jedná se tedy o simulaci např. v úvodu zmiňovaného synchronního měření fázového posuvu v elektrické síti. PTP switch v tomto případě není testován, protože dostupný PTP switch měl pouze 4 dostupné Ethernetové sloty. Proto je pro měření použit jen běžný Ethernetový switch (TP LINK TL F1008P). Konfigurace měření je stejná jako v přechozí kapitole.



Obr. 53 – Graf obálek histogramů odchylek PPS – 6 jednotek v síti

Rozlišení Δt uvedeného histogramu na je 30 ns (je zobrazen s hranicemi tříd 30 ns). Z histogramu je patrné, že úroveň synchronizace stále zůstává podobná jako pro 3 jednotky v síti s běžným Ethernetovým switchem.

Veličina / Měření	Průměrná hodnota $\bar{x}$ **(ns)**	Směrodaná odchylka **s (ns)**	Maximální hodnota $x_{max-\bar{x}}$ **(ns)**
obyčejný switch bez zátěže	-137	88	275
obyčejný switch se zátěží	-133	86	277

Tab. 3 – Porovnání odchylek hodin (PPS) 6 jednotek v síti

7.4.7 Porovnání výsledků úrovně synchronizace

Kombinací všech výše uvedených měření vychází porovnání úrovně synchronizace v různých scénářích, do kterých je TriggerBox Lite určen. Následující tabulka uvádí všechny 3 parametry jednotlivých situací (průměrná hodnota, směrodatná odchylka a maximální hodnota).

Měření	Jednotek v síti	Průměrná hodnota $\overline{x}$ **(ns)**	Směrodaná odchylka **s (ns)**	Maximální hodnota $x_{max-\overline{x}}$ **(ns)**
bez spojovacích prvků	2	5	22	80
s obyčejným switchem	2	48	82	289
s optickými převodníky	2	0	197	771
obyčejný switch bez zátěže	3	57	95	678
obyčejný switch se zátěží	3	53	98	792
PTP switch bez zátěže	3	8	20	69
obyčejný switch bez zátěže	6	-137	88	275
obyčejný switch se zátěží	6	-133	86	277

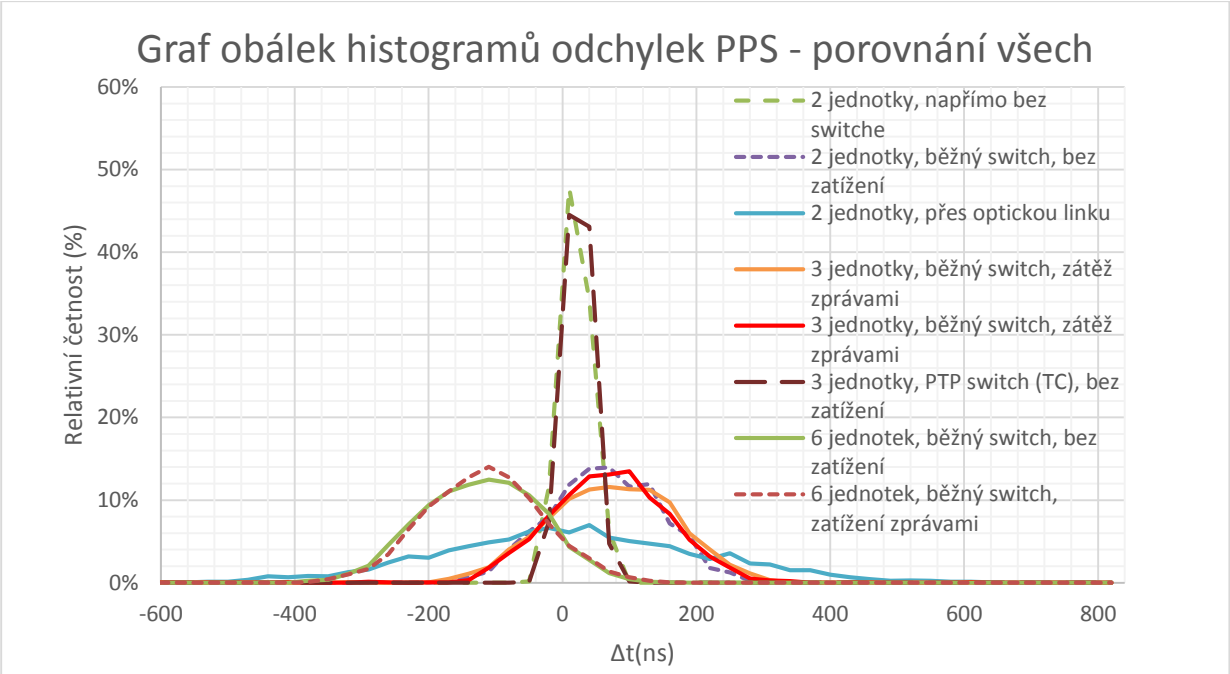
Tab. 4 – Porovnání odchylek hodin (PPS) všech měření

Z uvedených výsledků je patrné, že počet jednotek v síti nemá prakticky vliv na úroveň synchronizace. Zatížení jednotek TriggerBox Lite zprávami také nemá v uvedeném rozmezí vliv na synchronizaci. Pokud by bylo zatížení zprávami TCP-UART mostu vyšší (např. při přenášení dat z kamery), problém by pravděpodobně nastal jak v TriggerBox Lite, tak v PTP síti a došlo by k degradaci přesnosti synchronizace. Pro tyto situace však jednotka není navrhována.

Položkou, na které velice záleží, je volba aktivních spojovacích prvků v síti. Při použití switche s podporou PTP Transparent Clock zůstává úroveň synchronizace prakticky stejná jako při zapojení jednotek napřímo bez switche. Jedná se tedy o směrodatnou odchylku synchronizační chyby 20 ns a maximální chyba synchronizace do 100 ns. Při použití běžného Ethernetového switche směrodatná odchylka chyby stoupne na 100 ns.

Použití optické linky, která vyžaduje použití optických převodníků, se synchronizační chyba zvýší 2x oproti použití switche, tzn. směrodatná odchylka chyby na 200 ns. Optická linka by se tedy měla používat pouze v situacích, kde vzdálenost neumožňuje použití UTP kabelů.

Porovnání obálek histogramů všech měření ukazuje následující obrázek. Rozlišení Δt uvedeného histogramu je 30 ns (je zobrazen s hranicemi tříd 30 ns).



Obr. 54 – Graf obálek histogramů odchylek PPS všech měření

8 Dokumentace výsledného zařízení TriggerBox Lite

Popis výsledného zařízení kombinuje výstupy bakalářské práce [1] s výstupy z této diplomové práce. Kapitola tedy dokumentuje všechny implementované funkcionality a jejich chování včetně těch, jejichž chování je popsáno pouze v bakalářské práci [1].

Následující seznam udává podporované funkce TriggerBox Lite:

1. **Synchronizace přes Ethernetový protokol IEEE 1588v2 PTP**
   - 1.1. Jednotka se automaticky synchronizuje s libovolnou PTP jednotkou v Ethernetové síti. PTP jednotky v síti automaticky rozhodnou, která jednotka bude PTP master a které PTP slave.
   - 1.2. V Ethernetových sítích s jedním běžným switchem je dosahováno přesnosti synchronizace do 2 µs i při použití 6-ti jednotek v síti. Zmíněná přesnost je brána jako PTP Slave-PTP Slave. Přesnost synchronizace PTP Slave-PTP Master je dvojnásobná, tzn. do 1 µs.
   - 1.3. Po zapojení jednotek do sítě je třeba počkat 11 minut na ustálení synchronizační odchylky způsobené konfigurací, která minimalizuje synchronizační skoky
2. **Komunikační TCP-UART (Ethernet-UART) most**
   - 2.1. Jednotka umožňuje přeposílat zprávy z Ethernetu do UART rozhraní a naopak. Případně do RS232 při použití převodníku UART-RS 232. V Ethernetové síti jednotka posílá zprávy přes TCP protokol a vystupuje jako TCP server.

2.2. Zařízení připojené k TriggerBox Lite přes Ethernet musí pro komunikaci přes TCP-UART most otevřít TCP připojí s TriggerBox Lite na dané IP adrese a portu pro komunikační most. Jednotka zprávy následně posílá v rámci tohoto TCP připojení.

2.3. Maximální bezpečný datový tok TCP-UART mostem je 120 kb/s v obou směrech paralelně. Jednotka však povolí komunikovat rychlostí až 4 Mbit/s, ale při souvislé komunikaci může dojít ke ztrátě dat.

3. Možnosti připojení přes TCP protokol

3.1. Komunikace s jednotkou přes Ethernet probíhá prostřednictvím protokolu TCP.

3.2. TriggerBox Lite vystupuje v síti jako TCP server se statickou IP adresou, která se dá nastavit. Jednotlivé funkcionality jsou pak dostupné na 3 portech:

3.2.1. Komunikační most na portu 5027 (dá se přenastavit)

3.2.2. Rozhraní pro SCPI příkazy na portu 5025

3.2.3. Rozhraní pro Service request na portu 5026 (více k Service request dále)

4. Nastavovací SCPI rozhraní

4.1. Všechny funkce TriggerBox Lite jsou nastavitelné přes SCPI rozhraní. Seznam všech příkazů je uveden v příloze B

4.2. SCPI rozhraní implementuje také mechanismus Service Request (známý např. ze standardu IEEE 488). Tento mechanismus umožňuje nadřazené aplikaci zaslat informaci s požadavkem na obsluhu. Pro obdržení Service Request je nutné se připojit k TCP serveru jednotky na port 5026 a čekat na tyto Service Request zprávy z TriggerBox Lite. (Pro zapnutí funkcionality Service Request je třeba na port pro SCPI příkazy zaslat sekvenci příkazů *SRE 32; *ESE 2)

4.3. Nastavení zůstávají uložena v EEPROM paměti jednotky, takže zůstávají uložena i po vypnutí zařízení.

5. Udržování časové stupnice

5.1. Jednotka si udržuje interní časovou stupnici s rozlišením 8 ns. Tato časová stupnice je synchronizována s ostatními PTP jednotkami v síti. Od této časové stupnice se odvozují všechny signálové funkce jednotky (generování signálů a zaznamenávání času příchozí události)

6. Generování synchronních signálů

6.1. Jednotka umožňuje generovat události v daném čase na 3 výstupech OUT1, OUT2 a OUT3. Všechny 3 výstupy jsou rovnocenné a poskytují stejné funkcionality.

6.2. Podporované jsou jak jednotlivé události, tak periodické události se začátkem generování v daném čase.

6.3. Zařízení implementuje frontu výstupních událostí, tzn. je možné nastavit sekvenci událostí, které se v daném čase provedou. Fronta umožnuje uložit 10 generovaných událostí.

6.3.1. Minimální bezpečný odstup mezi událostmi naplánovanými ve frontě musí být 20 ms. Jednotka umožní generovat události i s menším odstupem, ale může dojít k přeskočení některé naplánované události.

6.4. Výstupní události mohou být následujících typů:

6.4.1. Náběžná nebo spádová hrana.

6.4.2. Puls s první hranou náběžnou nebo spádovou.

6.5. Rozlišení času generovaných událostí je 8 ns.

6.6. Perioda u periodických událostí může být libovolná větší 8 ns. Pokud však není násobkem 8 ns, dochází k jitterování signálu.[36]

6.7. Výstupy se dají použít pro poskytnutí hodinového signálu mikrokontroleru třetí strany. Perioda poskytnutých hodin však musí být dělitelná 8 ns kvůli eliminaci jitteru.

7. Zaznamenání času příchozí události

7.1. Zařízení umožňuje zaznamenat čas příchodu signálové hrany. Konkrétně je možné nastavit zaznamenávání náběžné, spádové nebo obou hran.

7.2. Rozlišení času zaznamenaných událostí je 8 ns.

7.3. Při zaznamenání více událostí za sebou se události ukládají do fronty o velikosti 10 hodnot.

7.4. Maximální perioda zaznamenaných událostí je 20 ms.

7.5. Při zaznamenání události jednotka umožňuje generování signálu Service Request.

[36] Více v kapitole 5.1

9 Závěr

V rámci této diplomové práce byla vytvořena jednotka TriggerBox Lite. Zařízení funguje jako rozhraní mezi jednotkami připojenými k Ethernetové síti (většinou PC nebo PTP jednotky) a měřicími nebo řídicími moduly připojenými přes rozhraní UART, resp. RS 232. Podřízeným měřicím nebo řídicím modulům zprostředkovává synchronizaci a komunikaci s nadřazenou jednotkou připojenou přes Ethernet. TriggerBox Lite tak funguje jako jednotka synchronizace přístrojů pro distribuované systémy, jak bylo vytyčeno v zadání.

TriggerBox Lite se po připojení do Ethernetové sítě automaticky synchronizuje s ostatními jednotkami podporujícími IEEE 1588v2 PTP protokol. Synchronizace u malých sítí dosahuje přesnosti 2 µs i při vytížení komunikací. Takové přesnosti jsou pro oblast použití TriggerBox Lite postačující. Jednotka umožňuje přeposílání zpráv z Ethernetu na UART, resp. RS 232 přes tzv. TCP-UART most s maximálním spolehlivým datovým tokem 120 kb/s. Jednorázově umožnuje i datové toky do 4 Mbit/s, ale při souvislé komunikaci může docházet ke ztrátě dat. Bezpečných 120 kb/s je postačující, protože oblast použití jednotky neobsahuje aplikace s velkým datovým tokem, jako jsou např. kamery. U kamer by bylo potřeba kromě jiného také použít další metody synchronizace přes PTP protokol, které zohledňují velké datové toky.

TriggerBox Lite poskytuje podřízeným jednotkám možnosti generování a zaznamenání synchronizačních signálů, a to včetně zpracování přes fronty. Rozlišení jednotky pro práci se signály je 8 ns. Zařízení TriggerBox Lite obsahuje nastavovací rozhraní se SCPI příkazy. Nastavení zůstávají uložená v EEPROM paměti jednotky i po jejím vypnutí. Všechna nastavení a důležité parametry jednotky jsou zdokumentovány a přiloženy k této práci.

Pro demonstraci funkcí jednotky byly naprogramovány a sestaveny 2 demonstrační úlohy. První úloha ukazuje synchronní měření sinusového signálu na několika synchronních bodech s běžnými multimetry připojenými k TriggerBox Lite. Pro účely katedry byl také implementován voltmetr s mikrokontrolerem STM32F0, který v demonstrační úloze funguje vedle běžných průmyslových multimetrů jako rovnocenné zařízení. Druhou úlohou je synchronní otáčení BLDC motory. Zde byly připraveny 2 BLDC motory z CD/DVD mechanik včetně řídicí jednotky implementované na modulu STM32F0. Řídicí jednotky BLDC motorů jsou napojeny na TriggerBox Lite jednotky a regulují motory na otáčení se synchronní fází. Tato úloha byla vytvořena jako hotová sestava na demonstračním panelu.

Vytvořená jednotka TriggerBox Lite je otestována a připravena do provozu v aplikacích synchronních distribuovaných sítí. Splňuje požadavky jak na nízkou cenu realizace synchronních distribuovaných systémů, tak na přesnost v řádu jednotek µs i požadavek spolehlivosti.

Reference

[1] P. Hájek, „Bakalářská práce: Modul pro synchronizaci sběru dat,“ 2013.

[2] J. Vavrouš, „Jednotka pro sběr dat a řízení s podporou PTP,“ *Bakalářská práce, ČVUT FEL,* 2012.

[3] „FreeRTOS,“ Real Time Engineers Ltd., [Online]. Available: http://www.freertos.org/.

[4] „lwIP - A Lightweight TCP/IP stack - Summary,“ [Online]. Available: http://savannah.nongnu.org/projects/lwip/.

[5] „PTPd - Precission Time Protocol daemon,“ [Online]. Available: http://ptpd.sourceforge.net/.

[6] J. Breuer, „Scpi parser,“ [Online]. Available: https://github.com/j123b567/scpi-parser.

[7] „Windows Presentation Foundation,“ Microsoft, [Online]. Available: https://msdn.microsoft.com/cs-cz/library/ms754130(v=vs.110).aspx.

[8] J. Breuer, „Scpi-parser library documentation,“ [Online]. Available: https://github.com/j123b567/scpi-parser.

[9] Agilent technologies, „Agilent 53131A/132A 225 MHz Universal Counter,Manual Part Number 53131-90055“.

[10] F. Zezulka, „Synchronizace v distribuovaných řídicích systémech: Precision Time Protocol (PTP) IEEE 1588,“ Automa, 2010.

[11] IEEE.org, „IEEE Standard for a Precision Clock Synchronization Protocol for NetWordek and Control Systems,,“ 2008. [Online]. Available: http://ieee.org/stamp/stamp.jsp?tp=&arnumber=4579760.

[12] „Openclipart,“ [Online]. Available: https://openclipart.org.

[13] ST Microelectronic, „RM0091 Reference Manual STM32F0x1/STM32F0x2/STM32F0x8“.

[14] ST Microelectronic, „UM1525 User Manual STM32F0DISCOVERY“.

[15] Texas Instruments, „DP83630 Precision PHYTER - IEEE 1588 Precision Time Protocol Transceiver datesheet“.

[16] I. J. Šimon, „BLDC aneb DC motor s nulovými náklady,“ *ELEKTRO,* sv. 10, pp. 13,14, 2011.

[17] „eLABS.com,“ [Online]. Available: http://elabz.com/brushless-dc-motor-with-arduino/.

[18] D. Arnold, „Meinberg blog,“ 2013. [Online]. Available: http://blog.meinbergglobal.com/2013/10/21/ieee-1588-clock-types/.

[19] „National Instruments - Cable Lengths and Transmission Speeds,“ 2012. [Online]. Available: http://www.ni.com/product-documentation/13724/en/.

[20] J. M. Cimbala, „Histograms,“ 2014. [Online]. Available: https://www.mne.psu.edu/me345/Lectures/Histograms.pdf.

[21] J. Yiu, The definite Guide to the ARM Cortex – M3, Elseviever, 2007.

[22] Hewlett-Packard, „HP 34401A Multimeter - Users Guide,Part Number 34401-90004,“ 1996.

Přílohy

A Seznam použitých zkratek a názvosloví

TriggerBox Lite	Jednotka, který je vyvíjená v rámci této diplomové práce
TBL	TriggerBox Lite
SyncBox	Jednotka fungující jako prostředník pro synchronizaci. Na vývoji se podílela práce [1]
SyncCore	Hlavní součást SyncBox. Vývojem se zabývala práce [1]
PHY	Ethernetová fyzická vrstva Texas Instruments DP83630
PTP	Precision time protocol (protokol synchronizace v Ethernetu IEEE 1588)
A/D	Analogově digitální
TCP/IP	Transmission Control Protocol/Internet Protocol (rodina protokolů pro Ethernet)
IEEE	Institute of Electrical and Electronics Engineers
UART	Universal asynchronous reciever transmitter
SPI	Serial peripheral interface
SCPI	Standart commands for programmable instruments
PWM	Pulse width modulation
CCR	Capture Compare Register
API	Aplication programming interface
PPS	Pulse per second
LED	Light-emitting diode
demo	Demonstrační úloha
GPIO	General Purpose input/output

B Dokumentace SCPI příkazů jednotky TriggerBox Lite[37]

V TriggerBox Lite jsou kromě uvedených příkazů implementovány i standardní příkazy dle standardu SCPI. Ty není nutné uvádět v tomto výčtu.

B.1 Generování signálů

Generování signálů umožňují 3 výstupy s ekvivalentními výstupy.

`SIGnal:OUT1|2|3:DISable`

- Deaktivace aktuálního signálu vysílaného na zadaném výstupu.

`SIGnal:OUT1|2|3:EVENt <čas_sekundy>, <čas_nanosekundy>, <typ_události>, <typ_hrany>, <periodicky>, <délka_periody>`

- Nastavení generování události.
    - Čas_sekundy: unix timeStamp – počet sekund od 1.1.1970
        - Pokud je zadáno 0, je událost vygenerována sekundu pro zpracování příkazu

[37] Seznam příkazů je převzat z bakalářské práce [1] a aktualizován pro TriggerBox Lite

- Čas_nanosekundy: počet nanosekund od začátku sekundy.
- Typ_události:
  - PULSE – dvě hrany (např.: náběžná a sestupná)
  - EDGE – jedna hrana
- Typ_hrany:
  - POSitive - náběžná hrana (u pulsu – první hrana náběžná)
  - NEGative - sestupná hrana (u pulsu – první hrana sestupná)
- Periodiky: Zda se bude událost periodicky opakovat
  - 0 – pouze jednou
  - 1 – periodicky
- Délka_periody: počet nanosekund. Maximálně 3 999 999 999

- Příklad nastavení náběžné hrany na čas 123.1s, která se bude opakovat s periodou 1s:
  - `SIG:OUT1:EVEN 123, 100000000,EDGE,POS,1, 1000000000`

B.2 Zaznamenání času událostí

Pro zaznamenání času událostí jsou připraveny dva vstupy, které jsou svými funkcemi ekvivalentní.

`SIGnal:IN1|2:DISable`

- Vypnutí zachytávání událostí na zadaném vstupu.

`SIGnal:IN1|2:EVENt <typ_hrany>, <jedna_hrana>`

- Nastavení zachycení události.
  - Typ_hrany: hrana, při níž se uloží čas jejího příchodu
    - POSitive – náběžná hrana
    - NEGative – sestupná hrana
    - BOTH – libovolná hrana
  - Jedna_hrana:
    - 1 – zaznamená se jen první hrana
    - 0 – hrany se zaznamenávají, dokud se vstup nevypne

- Při zaznamenání události se uloží čas jejího příchodu do bufferu, který se vyčítá po jednom záznamu příkazem SIGnal:IN:DATA?
- Pokud je jednotka nastavena na generování Service request (je třeba nastavit sekvencí příkazů *SRE 32; *ESE 2), jednotka po zaznamenání události generuje zprávu Service request na portu 5026
- Příklad nastavení vstupu IN1, aby zaznamenával všechny příchozí události:
  - `SIG:IN1:EVEN BOTH,0`

SIGnal:IN:DATA?

- Vrátí jeden záznam o příchozí události.
- Pokud nebyly zaznamenány události, vrátí hodnotu NONE.
- Pokud byla zaznamenána událost, vrátí zprávu ve tvaru:
- <vstup>, <hrana>, <čas_události>
    - Vstup: vstup na kterém byla událost zaznamenána. Buď 1 nebo 2
    - Hrana: zaznamenaná hrana. Možnosti:
        - POS – náběžná hrana
        - NEG – sestupná hrana
    - Čas_události: čas, ve kterém událost přišla. Hodnota je v Unix timestamp s desetinnou částí
    - Příklad přijaté zprávy ukazující příchod náběžné hrany na IN1 v čase 123.15:
    - „1,POS,123.15"

B.3 Komunikační most

Komunikační most TCP-UART umožňuje přeposílat data z Ethernetu (TCP protokol) na UART a naopak.

BRIDge:CONF:PORT <port>

- Port: port, na kterém bude spuštěn USART nebo SPI most. V základním nastavení je to 5027

BRIDge:USARt:CONF <spustit>,<přenosová_rychlost>, <stop_bity>, <parita>

- Nastavení USART-TCP a spuštění mostu.
    - Spustit: 1 – spustit, 0 - vypnout
    - Přenosová_rychlost: přenosová rychlost USART v jednotkách Bd
    - Stop_bity: 0.5/1/1.5/2
    - Parita:
        - ODD – sudá
        - EVEN – lichá
        - NONE – žádná

- Příklad spuštění UART mostu s rychlostí 115200Bd se stop bitem 0.5 a bez parity:
    - BRID:UART:CONF 1,115200,0.5,NONE

BRIDge:USARt:CONF?

- Vrátí nastavení UART a informaci, za je most spuštěn.
- Výsledek je ve formátu:

    - „Enabled:1, Baudrate: 115200, Stop bits: 1, Parity: NONE"

B.4 Nastavení LAN připojení

LAN:IPaddress <IP_část1>, <IP_část2>, <IP_část3>, <IP_část4>

- Nastavení IP adresy TriggerBox Lite.
- V továrním nastavení je hodnota 192.168.1.10
- Příklad nastavení adresy 192.168.2.10:
    - `LAN:IP 192,168,2,10`

LAN:GATEway <gateway_část1>, <gateway_část2>, <gateway_část3>, <gateway_část4>

- Nastavení výchozí brány.
- V továrním nastavení je hodnota 192.168.1.1

LAN:GATEway?

- Vrátí výchozí bránu ve formátu
    - `192.168.1.1`

LAN:NETMask <netmask_část1>, <netmask_část2>, <netmask_část3>, <netmask_část4>

- Nastavení masky podsítě.
- V továrním nastavení je hodnota 255.255.255.0

LAN:NETMask?

- Vrátí masku podsítě ve formátu
    - `255.255.255.0`

B.5 Hodiny

`TIME:SYNChronized?`

- Vrací PTP stav jednotky.
    - MASTER – pokud je PTP master
    - SLAVE – pokud je PTP slave
    - LISTENING – pokud hledá PTP zařízení

`TIME:VALue <čas>`

- Nastavení hodnoty systémového času.
    - Čas: hodnota unix timestamp
- Příkaz je ignorován, pokud je TriggerBox Lite jako PTP slave

`TIME:VALue?`

- Vrátí aktuální hodnotu systémového času.
- Navrácená hodnota je ve formátu Unix timestamp s desetinnou částí. Například:
    - `123.45678912`

B.6 Ostatní příkazy

`*RST`

- Nastaví TriggerBox Lite do továrního nastavení.
- Tovární nastavení:
    - IP adresa: 192.168.1.10
    - Maska podsítě: 255.255.255.0
    - Výchozí brána: 192.168.1.1
    - Port komunikačního mostu: 5027
    - Komunikační most: vypnutý

B.7 Chybové kódy

-300 událost (IN) nebyla zaznamenána
-301 fronta zachycených událostí plná
-302 fronta generovaných událostí je plná
-303 chyba generování výstupních událostí (chyba plánování událostí do fronty)

C Seznam použitých vstupů a výstupů TriggerBox Lite

Následující seznam udává seznam všech použitých výstupů TriggerBox Lite desky a potřebná propojení.

- UART. [38]
  - UART 1
    - Tx – PB6
    - Rx – PA10
  - UART 2
    - Tx – PD8
    - Rx – PD9
- I/O
  - OUT1 – PHY_8
  - OUT2 – PHY_11
  - OUT3 – PHY_9
  - IN1 – PHY_1
  - IN2 – PHY_10
- Nutné propojit PHY_4 a PA0
  - je potřeba kvůli synchronizaci PHY a MCU hodin
- Přepínače, tlačítka
  - PD7 – přepínač UART1 nebo UART2.
    - Tento pin je nastaven jako pull-up. Pokud je připojen na zem, použije se pro TCP-UART most UART2. Pokud není na zem připojen, použije se UART1.
    - Úroveň PD7 se vyčítá při inicializaci TCP-UART mostu. Je potřeba, aby byla požadovaná logická úroveň nastavena při spuštění celé jednotky. Pokud by došlo ke změně logické úrovně pinu po spuštění zařízení, může dojít k nedefinovanému chování zařízení.
  - PD1 – reset do továrního natavení.
  - Tento pin je opět nastaven jako pull-up. Pokud je připojen na zem (např. tlačítkem), dojde k uvedení jednotky do továrního nastavení a následnému restartu zařízení. Tato funkce je potřeba zejména k nastavení tovární IP adresy. Popis továrního nastavení je v kapitole B.6

[38] K dispozici jsou 2 UART výstupy. O tom, který se má využít, rozhoduje logická úroveň na pinu PD7. Více níže v seznamu nebo v kapitole 5.3

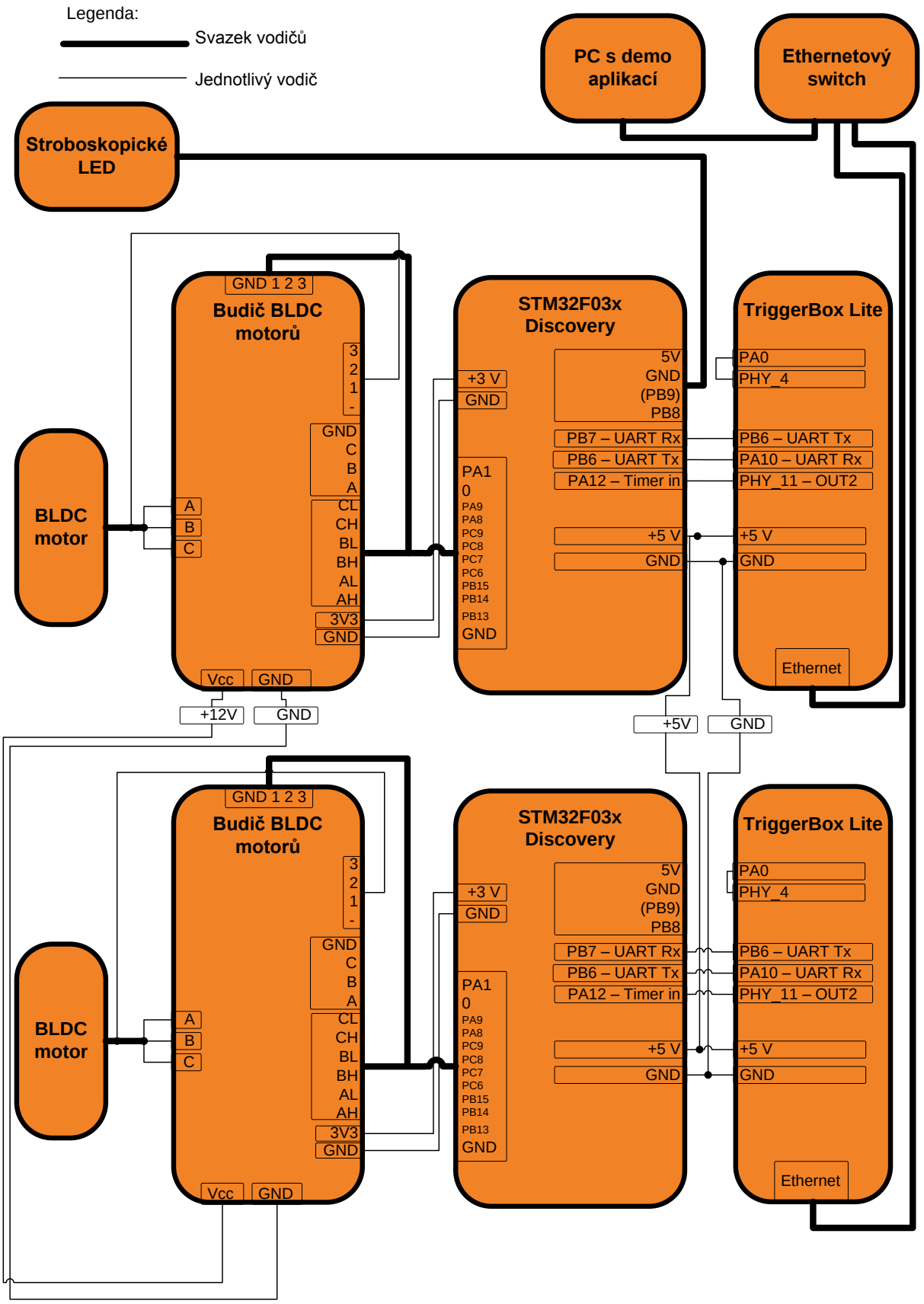
D Zapojení BLDC dema

Tato kapitola poskytuje jednoduchý návod jak zapojit BLDC demo.

Obr. 55 ukazuje zapojení jednotlivých součástí BLDC dema. Nákres počítá s využitím svazků kabelů připravených z rámci této diplomové práce. U konektorů na konci svazků kabelů je vždy označení, na které piny má konektor přijít.

U STM32F0x Discovery a TriggerBoxLite nesouhlasí pozice pinů se skutečnou realizací. U desky budiče BLDC motoru pozice naznačených pinů souhlasí se skutečnou realizací, aby byl nákres názornější pro zapojení.

Na PC s aplikací je třeba nastavit statickou IP adresu (např. 192.168.1.50)



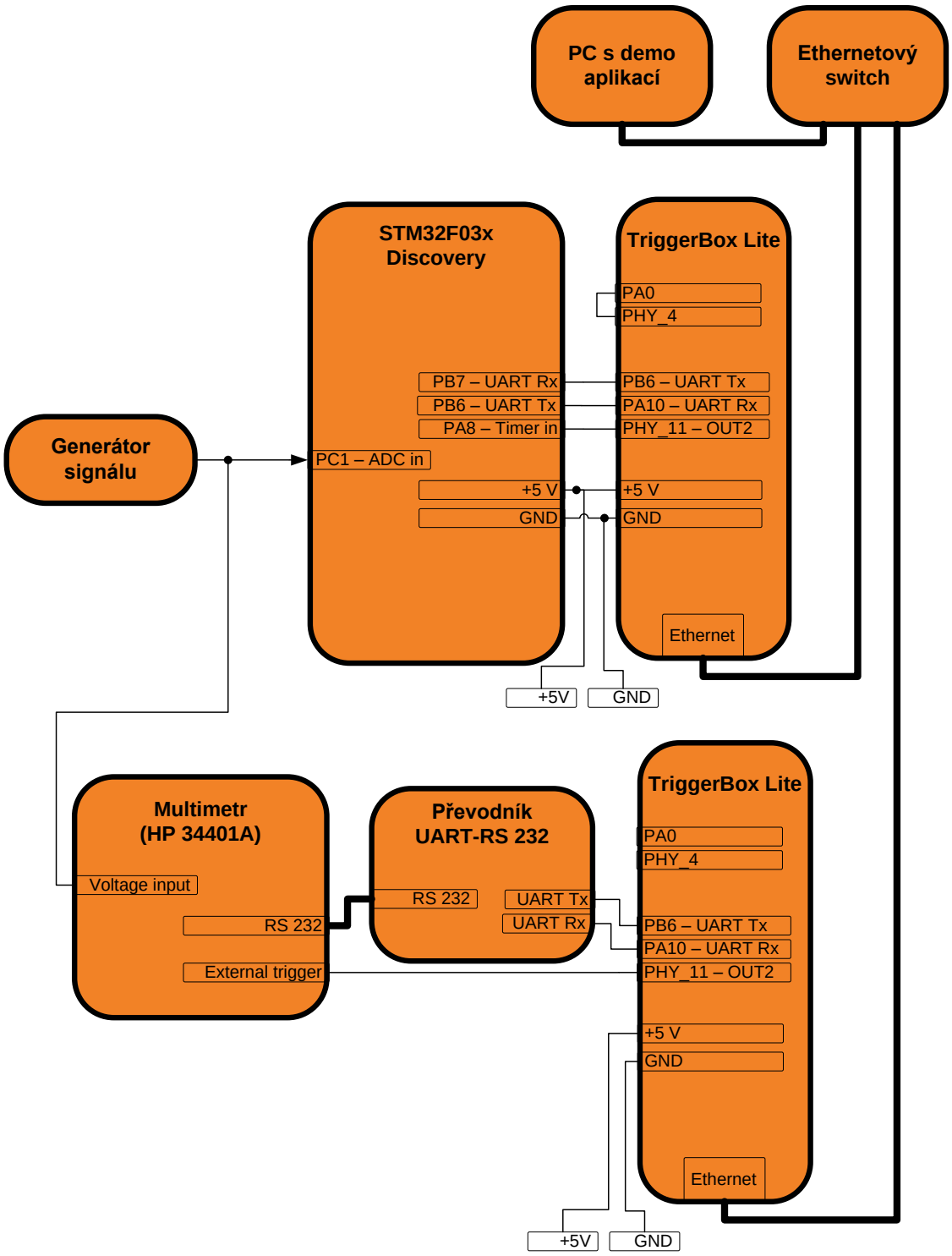
Obr. 55 – Zapojení BLDC dema

E Zapojení dema s voltmetry

Obr. 56 ukazuje zapojení dema s voltmetry.

U generování signálu přes generátor je třeba dát pozor na to, aby napětí signálu bylo v rozmezí 0-3,3V. Jinak může dojít k poškození modulu STM32F03x Discovery.

Na PC s aplikací je třeba nastavit statickou IP adresu (např. 192.168.1.50)



Obr. 56 – Zapojení voltmetr dema

F Obsah přiloženého DVD

Přiložené DVD obsahuje následující položky:

- Složka *PC aplikace* – obsahuje zkompilované PC aplikace
  - TBL-BLDCDemo
  - TBL-VmetrDemo
  - TBLControl
- Složka *Zdrojové kódy* – obsahuje zdrojové kódy k PC aplikacím i firmware MCU
  - Složka *STMF0* – obsahuje zdrojové kódy k firmware MCU STM32F0, který byl potřeba pro BLDC demo a Vmetr. Obojí je připraveno pro vývojové prostředí SC-IDE. Kódy vytvořené v rámci této diplomové práce jsou zde:
    - *Project**Standalone**BLDC* – pro řídicí jednotku BLDC motorů
    - *Project**Standalone**Vmeter* – pro voltmetr na STM32F0
    - *Project**Standalone**Common_libs* – knihovny společné pro oba FW
  - Složka *TBL-BldcDemo* obsahuje kódy k PC aplikaci pro BLDC demo
  - Složka *TBLControl* obsahuje kódy k PC aplikaci pro ovládání TriggerBox Lite
  - Složka *TBL-Testing* obsahuje kódy k PC aplikaci pro testování TriggerBox Lite
  - Složka *TBL-VmetrDemo* obsahuje kódy k PC aplikaci pro voltmetr demo
  - Složka *TriggerBoxLite* obsahuje kódy k FW TriggerBox Lite. Kódy vytvářené v rámci této práce jsou pak k nalezení ve složce:
    - *Project**FreeRTOS**TriggerBoxLite*. Konkrétně se jedná o soubory:
      - *conf_scpi.c, conf_scpi.h, eth_bridge.c, eth_bridge.h, main.c, pps_irig.c, pps_irig.h, scpi-def.c, syncPhy.c, syncPhy.h*

Vytvořené zdrojové kódy obsahují 2 notace komentářů. Část souborů obsahuje komentáře v češtině, protože tyto české komentáře byly použity v bakalářské práci. Nebyl důvod je překládat do angličtiny, proto celý projekt zůstal s komentáři v češtině. Druhá část souborů (Ve složce *STM32F0*) obsahuje komentáře v angličtině, protože je to programátorsky standardnější. Vždy tedy celý projekt obsahuje české nebo anglické komentáře.