

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

CZECH TECHNICAL UNIVERSITY IN PRAGUE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA KYBERNETIKY

FACULTY OF ELECTRICAL ENGINEERING

DEPARTMENT OF CYBERNETICS



SYNCHRONNÍ DISTRIBUOVANÝ SLEDOVACÍ SYSTÉM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ADAM BAŘTIPÁN



ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ
V PRAZE
CZECH TECHNICAL UNIVERSITY IN PRAGUE

FAKULTA ELEKTROTECHNICKÁ
KATEDRA KYBERNETIKY

FACULTY OF ELECTRICAL ENGINEERING
DEPARTMENT OF CYBERNETICS

SYNCHRONNÍ DISTRIBUOVANÝ SLEDOVACÍ SYSTÉM
SYNCHRONOUS DISTRIBUTED TRACKING SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ADAM BAŘTIPÁN

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. JAN FISCHER, CSc.

PRAHA 2013

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Adam Bařtipán
Studijní program: Kybernetika a robotika (magisterský)
Obor: Robotika
Název tématu: Synchronní distribuovaný sledovací systém

Pokyny pro vypracování:

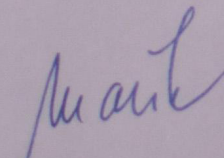
1. Navrhněte a realizujte distribuovaný systém s podporou PTP IEEE - 1588 pro synchronizovaný sběr dat ze senzorů s využitím modulů s procesorem STM32F407.
2. Navrhněte koncepci systému, formát komunikačních příkazů a způsob časování komunikace s jednotlivými moduly s využitím časového multiplexu pro synchronizaci a přenos dat tak, aby i při použití jednoduchých přepínačů rozhraní ETHERNET byl zajištěn spolehlivý přenos synchronizačních značek.
3. Správnost návrhu ověřte při realizaci distribuovaného systému sledování polohy objektu vysílajícího akustické impulsy, které systém napojený na jednotlivé senzory synchronně zachytí a ze zpoždění pak určí polohu objektu.
4. Vytvořte potřebné obvodové komponenty pro připojení senzorů programy pro vestavěný procesor STM32F407 i programové vybavení pro nadřazené PC.

Seznam odborné literatury:

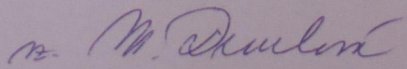
- [1] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4579760>
- [2] Yiu, J.: The definitive Guide to the ARM Cortex- M3. Elsevier, 2007
- [3] STMicroelectronics: Reference Manual, RM0090, doc ID018909 Rev3

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

Platnost zadání: do konce letního semestru 2013/2014


prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry




prof. Ing. Pavel Ripka, CSc.
děkan

ABSTRAKT

Tato práce se zabývá návrhem a realizací distribuovaného měřicího a sledovacího systému složeného z modulů s procesorem ARM, které jsou mezi sebou spojeny a synchronizovány pomocí rozhraní Ethernet a protokolu IEEE-1588. Každý modul je vybaven taktéž rozhraním USB, pomocí kterého může být dále rozšířen. Součástí práce je také návrh testovacího SW a HW, který pomocí synchronně zaznamenaných dat ze třech modulů s mikrofonom určí pozici zdroje zvuku.

KLÍČOVÁ SLOVA

ARM, Cortex-M3, STM32, PTP, IEEE-1588

ABSTRACT

This thesis deals with design and construction of distributed measurement and sound tracking system consisting of small modules with ARM microcontroller and necessary SW equipment. The modules are connected to Ethernet network and synchronized via PTP protocol (IEEE-1588). The software equipment is capable of tracking sound source and visualize its position using data measured by synchronous distributed measurement system.

KEYWORDS

ARM, Cortex-M3, STM32, PTP, IEEE-1588

BAŘTIPÁN, Adam *Synchronní distribuovaný sledovací systém*: diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra kybernetiky, 2013. 67 s. Vedoucí práce byl doc. Ing. Jan Fischer, CSc.

PROHLÁŠENÍ

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

Praha

31.12.2013



(podpis autora)

Rád bych poděkoval vedoucímu práce, doc. Ing. Janu Fischerovi, CSc za vedení práce a odborné konzultace. Dále bych rád poděkoval svým rodičům, Daně Bařtipánové a Jaroslavovi Bařtipánovi, za neocenitelnou morální i materiální podporu v průběhu celého mého studia.

OBSAH

1	Úvod	10
2	Rozbor zadání	11
2.1	Návrh synchronní měřicí ústředny	11
2.1.1	Mikrokontroler	12
2.1.2	Program pro mikrokontroler	12
2.1.3	Operační systém mikrokontroleru	13
2.1.4	TCP/IP a UDP stack	14
2.2	Synchronizace času	15
2.2.1	Způsoby synchronizace času	15
2.2.2	Současné trendy časové synchronizace v síti Ethernet	15
2.2.3	Protokol PTP	18
2.2.4	Způsob synchronizace času pomocí PTP	18
2.3	Možnosti akustické lokalizace	20
2.4	Cíl práce	22
3	Synchronní distribuovaná měřicí ústředna	23
3.1	HW vybavení ústředny	24
3.1.1	Modul ústředny	24
3.1.2	Mikrokontroler	27
3.1.3	Fyzická vrstva rozhraní Ethernet	29
3.1.4	Mikrofonní předzesilovač	30
3.2	Program pro modul	32
3.2.1	Operační systém modulu	32
3.2.2	Distribuovaný systém	34
3.3	Ovládací program pro PC	37
4	Akustický sledovací systém	42
4.1	Uspořádání sledovací soustavy	42
4.2	Předzpracování naměřených dat	42
4.3	Stanovení zpoždění signálů	44
4.4	Výpočet pozice zdroje zvuku	47
4.5	Dosažené výsledky	50
4.5.1	Odchylka určení azimutu θ	50
4.5.2	Odchylka určení vzdálenosti r	52
4.5.3	Odchylky vypočtených souřadnic	53
5	Závěr	55

Literatura	57
Seznam symbolů, veličin a zkratk	58
Seznam příloh	60
A Výrobní podklady	61
A.1 Schéma zapojení	61
A.2 Motiv DPS	63
A.3 Rozmístění součástek na DPS	65
A.4 Seznam součástek	65
B Použité názvosloví	67

SEZNAM OBRÁZKŮ

2.1	Blokové schéma základní myšlenky řešení	11
2.2	Synchronizace času mezi master a slave zařízením	19
2.3	Hyperbola daná zpožděním dvou signálů	20
3.1	Blokové schéma synchronní distribuované měřicí ústředny	23
3.2	Pohled na modul měřicí ústředny shora	24
3.3	Fotografie použitého přepínače s PoE	25
3.4	Pohled na modul měřicí ústředny zespodu	26
3.5	Blokové schéma Connectivity line MCU z rodiny STM32	27
3.6	Blokový diagram propojení mikrokontroleru a fyzické vrstvy	29
3.7	Schéma mikrofonního předzesilovače	31
3.8	Plošný spoj mikrofonního předzesilovače	32
3.9	Znázornění členění systému FreeRTOS na jednotlivé vrstvy	33
3.10	Znázornění časového multiplexu	36
3.11	Ukázka ovládacího programu	38
3.12	Menu umožňující správu modulů	38
3.13	Dialogové okno pro zadání IP adresy	39
3.14	Okno upozorňující na špatně zadanou IP adresu	39
3.15	Volba kanálů pro jednotlivé moduly	40
4.1	Uspořádání sledovací soustavy	42
4.2	Ukázka signálů zaznamenaných mikrofonom	44
4.3	Příklad signálů zaznamenaných mikrofonom	46
4.4	Výstupní vektor algoritmu GCC-PHAT	47
4.5	Hyperbola daná zpožděním dvou signálů	48
4.6	Rozložení mikrofونů v souřadném systému	49
4.7	Série dvaceti měření azimutu	51
4.8	Série dvaceti měření vzdálenosti r	52
4.9	Rozložení mikrofونů v souřadném systému	53
A.1	Vrstva TOP, měřítko 1:1	63
A.2	Vrstva BOT, měřítko 1:1	64
A.3	Rozmístění součástek ve vrstvě BOT, měřítko 1.2:1	65

SEZNAM TABULEK

4.1	Parametry pokusu pro určení odchylky θ a r	51
4.2	Výsledky pokusu pro určení úhlu θ	51
4.3	Výsledky pokusu pro určení vzdálenosti r	52
4.4	Výsledky pokusu pro určení polohy zdroje zvuku	54

1 ÚVOD

Tato práce se zabývá návrhem synchronního distribuovaného sledovacího systému, který za pomoci sběru dat z mikrofونů a jejich následného vyhodnocení určí polohu zdroje zvuku.

Prvním krokem k realizaci sledovacího systému je vytvoření distribuované měřicí ústředny, která bude složena z modulů s mikrokontrolerem Advanced RISC Machine (ARM). Moduly budou taktéž vybaveny rozhraním Ethernet, prostřednictvím kterého bude probíhat komunikace s nadřazeným počítačem.

Aby naměřená data z jednotlivých modulů bylo možno dále vyhodnocovat, je nezbytné znát přesný čas, kdy byla pořízena. Zde mají velkou výhodu centralizované měřicí systémy, které díky bezprostřední blízkosti řídicí jednotky i měřících karet (jsou uvnitř jednoho měřicího boxu) mohou mít rozvedeny centrální zdroj hodin, případně synchronizační pulzy.

Jiná situace nastává v případě, kdy jsou od sebe jednotlivé části měřicího systému vzdáleny desítky metrů (metalické spoje) či kilometrů (optické spoje), jako je tomu u distribuované měřicí ústředny.

V takovém případě je nutné, aby synchronizace času probíhala prostřednictvím komunikace po rozhraní Ethernet a v závislosti na zjištěných odchylkách od referenčního času docházelo k úpravě lokálního času na jednotlivých modulech.

Výše zmíněný požadavek splňuje protokol Precision Time Protocol (PTP) definovaný normou IEEE-1588, který bude v této práci použit k synchronizaci času mezi jednotlivými moduly a referenčním zdrojem času. Tento zdroj bude moci představovat jak nadřazený počítač, tak i kterýkoliv z použitých modulů.

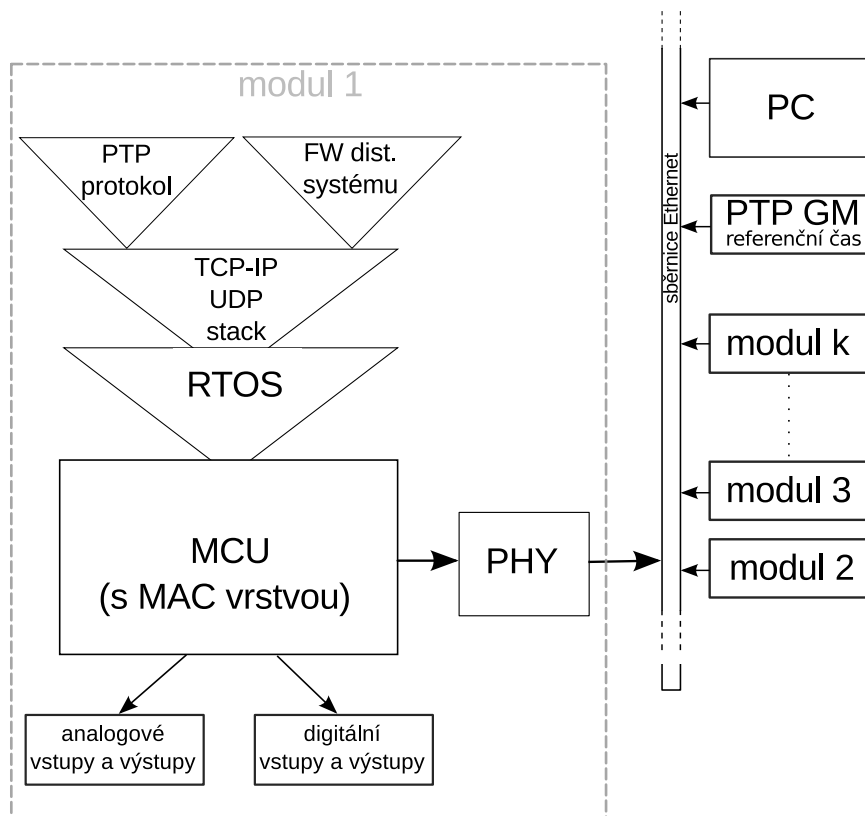
Druhým krokem k realizaci synchronního distribuovaného sledovacího systému je návrh vstupní mikrofونové části, která bude poskytovat vhodně zesílený signál z mikrofonu vhodný k dalšímu zpracování.

Závěrečnou částí mé práce bude realizace sledovacího systému, který pomocí synchronního vzorkování signálu z několika mikrofونů a následného určení zpoždění mezi jednotlivými signály vypočte pozici zdroje zvuku.

2 ROZBOR ZADÁNÍ

2.1 Návrh synchronní měřicí ústředny

Jak již bylo zmíněno v úvodu, základem modulu bude mikrokontroler s ethernetovým rozhraním, na kterém je potřeba implementovat obslužné programy pro komunikaci.



Obr. 2.1: Blokové schéma základní myšlenky řešení

Blokové schéma 2.1 nastiňuje možné řešení celého modulu a distribuovaného systému. Základním stavebním kanemem je modul synchronního distribuovaného systému, který obsahuje mikrokontroler vykonávající naprogramované úkoly (měření, odesílání dat, synchronizace času, atd.).

Jednotlivé moduly komunikují mezi sebou a nadřazeným PC prostřednictvím rozhraní Ethernet, které může mít formu metalických, optických či bezdrátových spojů. Aby mohl mikrokontroler přistupovat k síti, vyžaduje obvod zprostředkovávající fyzický přístup ke komunikačnímu médiu, který je označen "PHY".

Blok nazvaný Real-Time Operating System (RTOS) symbolizuje přítomnost operačního systému v mikrokontroleru, jehož použití může usnadnit realizaci celého synchronního distribuovaného systému.

Potřeba extenzivní komunikace po síti vyžaduje programovací rozhraní, které poskytne poskytně programátorovi pohodlný přístup k síti a síťovým protokolům. Takové rozhraní se v praxi označuje "TCP/IP stack".

Jednotlivé moduly, aby mohli být synchroniovány, musí mít svůj vlastní zdroj hodin se známou odchylkou od referenčního času, který je v obrázku označen jako "PTP GM".

2.1.1 Mikrokontroler

Aby mohl každý modul autonomně vykonávat naplánované úkoly, vyžaduje vlastní řídicí jednotku, kterou v tomto případě bude realizovat mikrokontroler od společnosti ST Microelectronics. Konkrétně bude použit mikrokontroler STM32F407, který je dán zadáním.

Jedná se o mikrokontroler z rodiny ARM Cortex-M4, který disponuje dotatečným výpočetním výkonem pro realizaci zadání. Disponuje také komunikačním rozhraním Ethernet, ale vyžaduje přítomnost fyzické vrstvy rozhraní Ethernet aby byl schopen komunikace s dalšími zařízeními.

Vzhledem k malé paměti RAM bude třeba navrhnout způsob, jakým se budou data dočasně ukládat v procesoru a včas odesílat tak, aby bylo možno kontinuálně měřit a odesílat data, aniž by vlivem nedostatku paměti došlo k jejich ztrátě či pádu operačního systému.

Podrobný popis možností mikrokontroleru i fyzické vrstvy je obsahem sekce 3.1.

2.1.2 Program pro mikrokontroler

Zde popisovaný distribuovaný systém bude vykonávat několik více či méně nezávislých úloh, které bude nutno v rámci práce definovat a vytvořit. Takové programové vybavení se v praxi označuje Firmware (FW), jak je naznačeno na obr 2.1.

Firmware pro modul může být vytvořen v jazyce Assembler, který umožňuje maximální využití veškerých prostředků mikrokontroleru. Nevýhodou tohoto jazyka je ovšem množství zdrojového kódu, který by bylo nutné vytvořit k realizaci takového projektu.

Další nevýhodou je špatná orientace v kódu, která komplikuje jeho následné udržování a vylepšování.

Jinou možností je použití jazyka C či C++, který poskytuje programátorovi výrazně pohodlnější prostředí pro vytváření větších programových celků a výrazně zjednodušuje následnou orientaci v kódu a jeho další rozvíjení.

Cenou za toto pohodlí je zpravidla větší výsledný soubor programu, který je nutné nahrát do mikrokontroleru, a s tím spojené pomalejší vykonávání programu.

Přes výše zmíněný nárůst velikosti programu bude veškeré programové vybavení pro modul psáno v jazyce C/C++, jelikož je vzhledem k velikosti celého projektu výrazně výhodnější.

2.1.3 Operační systém mikrokontroleru

Vzhledem k požadavkům plynoucím ze zadání i ze samotné podstaty zařízení je jasné, že každý modul bude vykonávat několik činností zároveň (měření dat, odesílání dat, synchronizace času, aj.), proto se zde přímo nabízí použití RTOS. Operační systém umožní na jednom procesorovém jádře běh několika nezávislých úloh pomocí rozdělení strojového času na krátké časové úseky, které jsou následně přiřazovány jednotlivým procesům dle jejich priorit.

Po krátké úvaze byl zvolen FreeRTOS¹. Jedná se o operační systém vhodný pro systémy reálného času především díky svým malým nárokům na systémové prostředky. Poskytuje také pokročilé funkce pro řízení běhu jednotlivých procesů včetně mutexů a semaforů, které jsou v této aplikaci klíčové.

K důvodům, proč vybrat právě FreeRTOS přispěla také jeho bezplatnost pro nekomerční i komerční použití, díky které patří mezi nejpoužívanější a nejlépe podporovaný RTOS mezi takřka všemi výrobci mikrokontrolerů.

¹Kompletní informace je možné najít na stránkách výrobce: <http://www.freertos.org/>.

2.1.4 TCP/IP a UDP stack

Transmission Control Protocol (TCP) a User Datagram Protocol (UDP) stack je klíčová část programu mikrokontroleru, jelikož poskytuje uživatelským programům snadno použitelné rozhraní pro navazování spojení, výměnu dat a následné ukončení spojení. Uživatel díky tomu nemusí znát (a především programovat) do detailu celou komunikaci prostřednictvím ethernetového rozhraní, byť pochopení ethernetové komunikace na všech úrovních je bezesporu žádoucí a pro návrh takového modulu dokonce nezbytné.

V mé implementaci distribuovaného systému slouží TCP protokol k přenosu řídicích a informativních zpráv mezi řídicím počítačem a jednotlivými moduly. Tento protokol dále využívá webový server, který je spuštěn na každém modulu a poskytuje některé doplňkové služby jako prohlížení systémových záznamů o činnosti. Naproti tomu UDP protokol je zde využit především pro přenos naměřených dat směrem od modulů do řídicího počítače.

Tento protokol byl zvolen pro přenos dat z důvodu nižších nároků na režiji přenosu, tj. při konstantním objemu odesílaných dat klade nižší nároky na přenosovou kapacitu sběrnice než v případě odesílání stejných dat protokolem TCP.

Jednoduchost UDP protokolu má ale i svá rizika, která ústí v to, že musíme kontrolovat, zda data z modulu dorazila celá (tj. ztrátu packetů) a zda dorazila ve správném pořadí, v jakém byla vyslána (tj. pořadí packetů). Tyto problémy řeší obslužný SW pro počítač.

Jako vhodný TCP/IP stack pro mikrokontroler byl vybrán „lightweight TCP/IP stack“, v praxi častěji označovaný jako lwIP. Samozřejmostí je podpora TCP a UDP protokolů.

Narozdíl od jiných zvažovaných možností (např. uIP TCP/IP stack) poskytuje lwIP plnou implementaci protokolu Internet Group Management Protocol (IGMP), který je nezbytný pro synchronizaci času pomocí PTP (viz 2.2.3), a zdrojové kódy jsou volně dostupné.

Velkou výhodou je v tomto případě fakt, že mnozí výrobci mikrokontrolerů, stejně jako ST Microelectronics, poskytují již hotovou implementaci tohoto stacku pro své procesory. Usnadňují tak vývojářům použití jejich obvodů a zkracují celkový čas vývoje daného zařízení.

2.2 Synchronizace času

2.2.1 Způsoby synchronizace času

V měřicí, řídicí a automatizační technice, kde je v systému použito více nezávislých zařízení, je často vyžadováno přesné časování těchto zařízení tak, aby bylo dosaženo synchronizace jednotlivých události resp. korelace naměřených dat.

Aby bylo možno takovéto synchronizace dosáhnout, jednotlivá zařízení musejí mít buď přístup k centrálnímu zdroji hodinového signálu, nebo si musejí synchronizovat své vlastní zdroje hodinového signálu podle referenčního zdroje.

Zástupcem prvního případu je např. systém PCI Extensions for Instrumentation (PXI), kde je v rámci jednoho chasis² všem modulům poskytnut přístup k referenčnímu zdroji 10MHz hodinového signálu s vysokou přesností. Toto řešení je ovšem únosné pouze v případě, že jsou jednotlivé měřicí moduly poměrně blízko referenčního zdroje taktu, jako v případě PXI (vzdálenost nepřesahuje 1m).

Poměrně často se ale v praxi setkáme s potřebou mít jednotlivé měřicí moduly rozmístěné ve větších vzdálenostech a ojedinělé nejsou ani případy, kdy se jednotlivé moduly v průběhu měření pohybují a mění svou vzdálenost od řídicího modulu.

To jsou typicky distribuované systémy založené na autonomních měřicích a řídicích modulech, kde se zpravidla nerozvádí hodinový signál přímo po komunikační sběrnici ve formě periodického signálu o dané frekvenci.

Důvodem je fakt, že se vzrůstající vzdáleností od zdroje signálu dochází k jeho deformaci např. v důsledku elektromagnetického rušení nebo vzájemné kapacity vodičů. Navíc v případě, že měřicí modul mění vzdálenost od referenčního zdroje, dochází také ke změně zpoždění, s jakým dorazí signál do synchronizovaného zařízení, což se poměrně obtížně kompenzuje.

2.2.2 Současné trendy časové synchronizace v síti Ethernet

Přestože dnes existuje velké množství průmyslových sběrnic s vysokou spolehlivostí přenosu dat, současným trendem na poli distribuovaných systémů je spíše používat sítě typu Ethernet. Hlavním důvodem je nepochybně cena takového řešení, která

²Přístrojová skříň s napájecím zdrojem, sloty pro zasunutí jednotlivých modulů a rozvodem referenčního časového signálu.

je díky masivnímu rozšíření Ethernetu v posledních letech zpravidla nižší než konkurenční možnosti na bázi striktně průmyslových sběrnic (např. RS-485, RS-422, CAN, Profibus a pod.).

Dalším pozitivním faktorem masivního rozšíření Ethernetu je také fakt, že výrobci mikrokontrolerů dnes mají tendenci nabízet ve svých jednočipových řešeních i ethernetové rozhraní, aniž by to zásadním způsobem navyšovalo cenu výsledného produktu.

V neposlední řadě je důležitá univerzálnost a modulárnost ethernetové sítě, díky které je možné síť dále rozvíjet, měnit její strukturu, škálovat a v případě potřeby také zvyšovat propustnost dat (rychlost komunikace).

S výhodou je také možné použít již existujících ethernetových zařízení (přepínače, rozbočovače, opakovače, směrovače a další), které jsou sice primárně určeny například do datových center velkých společností, ale díky své kvalitě a spolehlivosti mohou být nasazeny i v průmyslovém provozu.

Jednou z posledních výhod Ethernetu je také jeho nezávislost na druhu média, prostřednictvím kterého je komunikováno. Lze tedy komunikovat po metalické vedení (nejčastější), v případě prostřední se silným elektromagnetickým rušením není problém průběžně přejít z metalického na optické médium (optické vlákno) a stejně tak může metalické i optické vedení přecházet v případě potřeby v bezdrátovou komunikaci vzduchem, jakou je například WiFi či WiMax [4], [5].

Velkou nevýhodou sítě Ethernet z pohledu časové synchronizace je fakt, že vysílaná data (pakety) nemusí dorazit do cílového bodu ve stejném pořadí, v jakém byla vyslána, což se v rozsáhlejších sítích běžně stává. Dalším problémem může být proměnná (např. v závislosti na vytížení sítě) délka zpoždění mezi okamžiky vyslání a příjmem dat, která se díky nízké předvídatelnosti obtížně koriguje.

Pro účely časové synchronizace (nejen) v prostředí sítě Ethernet vznikla řada protokolů, které se od sebe liší především přesností, s jakou dokáží udržovat stanovený systémový čas.

Jedním z takovýchto protokolů je Reference Broadcast Time Synchronization (RBTS), který pracuje na poměrně jednoduchém principu. Takzvaný broadcast beacon vyšle všem zařízením připojeným ke komunikační sběrnici znamení k synchronizaci, což je zpráva, která neobsahuje informaci o aktuálním čase, ale pouze výzvu

všem účastníkům komunikace k tomu, aby si mezi sebou navzájem synchronizovali čas. Samotný broadcast beacon tedy pouze „odstartuje“ synchronizaci a dále se na ní nepodílí.

Synchronizace probíhá prostřednictvím výměny časových údajů mezi jednotlivými zařízeními a následným dopočítáním časových posunutí vůči každému účastníkovi komunikace.

Tato metoda se nejčastěji používá v bezdrátových sensorových sítích. Její výhodou je jednoduchá implementace takového protokolu a odstranění nejistoty, která u protokolů pracujících na bázi referenčního času vysílaného jedním zařízením (zpráva obsahuje údaj o referenčním čase) vzniká v důsledku předem neznámého zpoždění mezi vysílací stranou a koncovými body sítě, kterým je zpráva určena.

Nevýhodou je paradoxně právě absence referenčního času, která způsobuje, že systémový čas jednotlivých zařízení neodpovídá jednomu konkrétnímu času, který je považován za přesný, ale spíše se blíží aritmetickému průměru systémových časů jednotlivých zařízení.

Vzhledem k předpokládanému použití v bezdrátových sensorových sítích byl při návrhu tohoto protokolu kladen velký důraz na nízkou spotřebu (jednotlivá zařízení bývají často napájena z akumulátorů) na úkor přesnosti časové synchronizace.

Dalším protokolem používaným pro synchronizaci času je tzv. Flooding Time Synchronization Protocol (FTSP), který již pracuje se zdrojem referenčního času, který periodicky vysílá zprávu s časovou značkou odpovídající referenčnímu času. Tato zpráva je adresovaná všem účastníkům komunikace. Každý účastník po příjmu takové zprávy zaznamená čas přijetí a složením obou časových údajů získává informaci o posunutí svého času oproti referenčnímu.

Protokol ovšem nekompensuje proměnné zpoždění mezi vysláním a příjmem zprávy. Díky tomu se průměrná chyba systémového času oproti referenčnímu pohybuje v řádu desítek až stovek mikrosekund. Stejně jako v případě protokolu RBTS i protokol FTSP byla navrhován s ohledem na nízkou spotřebu a to na úkor přesnosti časové synchronizace.

Velice využívaným protokolem pro synchronizaci času užívaným především mezi osobními počítači a síťovými servery je Network Time Protocol (NTP). Tento protokol je poměrně složitý a existuje v několika verzích (aktuální verze má pořadové

číslo 4 a následující verze je ve vývoji), které jsou detailně popsány na stránkách organizace zajišťující jeho vývoj: www.ntp.org. V síti internet dokáže pracovat s odchylkou v řádu desítek milisekund, v lokálních sítích je možné dosáhnout přesnosti na stovky mikrosekund.

Poslední zde zmiňovanou možností je tzv. Precision Time Protocol, jehož implementací na jednočipovém mikropočítači se zabývá tato práce. Více podrobností je možno nalézt v sekci 2.2.3, která je tomuto protokolu věnována.

2.2.3 Protokol PTP

Pro účely synchronizace času v síti Ethernet vznikl protokol definovaný normou Institute of Electrical and Electronics Engineers (IEEE-1588), též označovaný jako PTP, který bude využit i ve zde popisovaném modulu. Jeho cílem je nabídnout alternativu synchronizace hodinového signálu k synchronizaci pomocí centrálně rozváděného hodinového signálu.

Toho je dosaženo periodickým vyměňováním informací mezi master zařízením a jednotlivými slave moduly tak, aby si slave moduly mohly samy korigovat a „sladit“ jejich lokální zdroje hodinového taktu s master jednotkou.

PTP poskytuje standardizovaný protokol pro synchronizaci hodin prostřednictvím sítě umožňující v jeden okamžik vyslat jednu zprávu více příjemcům³, jako například Ethernet. Nabízí možnost synchronizace heterogenních zdrojů hodinového signálu, nezatěžuje síť přílišnými nároky na objem přenesených dat a nevyžaduje mnoho výpočetního výkonu.

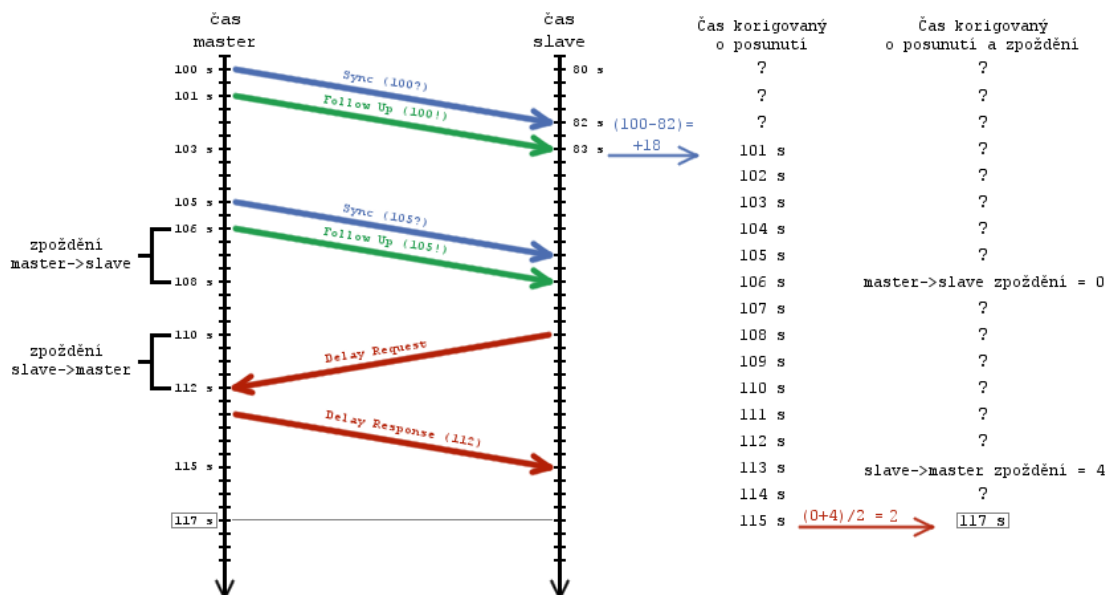
Protokol byl standardizován institutem IEEE poprvé v roce 2002, jeho druhá verze byla vydána v roce 2008. Druhá verze IEEE-1588 protokolu, označovaná jako IEEE-1588-2008 neboli PTP v2, poskytuje vyšší přesnost, robustnost a spolehlivost tohoto protokolu, ale není zpětně kompatibilní s IEEE-1588-2002 neboli PTP v1, tj. s verzí z roku 2002.

2.2.4 Způsob synchronizace času pomocí PTP

Lokální čas slave zařízení se synchronizuje s master jednotkou pomocí obousměrné komunikace, která je naznačena na obr. 2.2. Na dvou svislých osách je vynesena lokální čas nadřazené (master) resp. podřazené (slave) jednotky. Můžeme si všimnout, že

³V zahraniční literatuře se tato vlastnost označuje jako tzv. multicast.

obě osy začínají s jiným počátečním časem (osy jsou orientovány shora dolů). Toto je běžný případ, kdy jsou lokální časy jednotlivých jednotek posunuty vůči sobě, v anglické literatuře je tento jev označován jako *offset*.



Obr. 2.2: Synchronizace času mezi master a slave zařízením

Korekce posunutí se provede po tom, co master vyšle periodicky vysílanou synchronizační zprávu - **Sync(100)**, ke které je připojená časová značka udávající, kdy byla zpráva odeslána mikrokontrolerem.

Po této zprávě může případně následovat ještě jedna zpráva - **Follow Up(100)**, která obsahuje značku s časem, kdy byla zpráva **Sync(100)** skutečně odeslána fyzickou vrstvou Ethernetu. To dovoluje slave zařízení zjistit skutečný čas, kdy byl dán mikrokontrolerem povel k odeslání zprávy a tím pro slave zpřesnit informaci o referenčním čase master jednotky.

Toto má význam v sítích, kde čas vyslání paketu nemůže být znám dopředu. Zástupcem takové sítě je například Ethernet, kde vlivem detekce kolize a následného mechanismu řešení takové kolize obecně nemůže⁴ být znám přesně čas, kdy došlo k odeslání celého paketu. Ve chvíli, kdy se podaří odeslat celý packet, už není zpravidla možné měnit jeho obsah, tedy ani časovou značku.

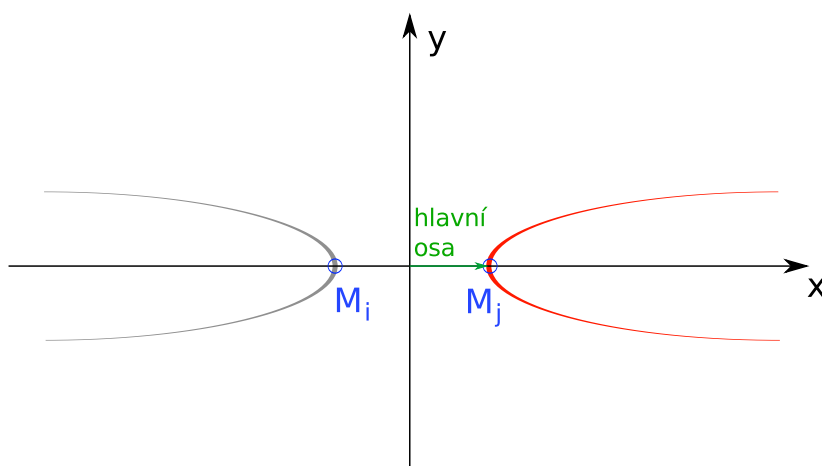
⁴Dnes jsou k dispozici i specializované ethernetové fyzické vrstvy, které již dokáží časovou značku injektovat do odeslaného paketu na HW úrovni, jedná se například o DP83848 od společnosti Texas instruments.

Poté se opakuje vyslání synchronizačního paketu následované dotazem slave zařízení na zpoždění sítě - *Delay Request*. Slave zaznamená čas, kdy odeslal dotaz, a master po přijetí tohoto dotazu obratem odešle zprávu - *Delay Response(112)* s časovou značkou okamžiku, kdy dorazil paket s žádostí. Po této proceduře již slave zná jak časové posunutí obou zdrojů času, tak i obousměrné zpoždění při přenosu zprávy způsobené charakterem sítě⁵. Slave nyní zná referenční čas master jednotky s vysokou přesností⁶ a systémový čas je tedy synchronizován.

V souvislosti s výše zmíněným postupem synchronizace je vhodné poznamenat, že *Sync()* pakety jsou vysílány periodicky (periodu určuje master zařízení). Slave může kdykoliv vyslat požadavek na určení zpoždění sítě (*Delay Request*), master mu ale odpoví zprávou obsahující maximální průměrnou frekvenci, s jakou se smí dotazovat.

2.3 Možnosti akustické lokalizace

Cílem práce je vytvoření distribuovaného systému, který bude schopen synchronně měřit signály z mikrofونů a z jejich zpoždění určí pozici zdroje zvuku v rovině.



Obr. 2.3: Hyperbola daná zpožděním dvou signálů

Aby bylo možno určit pozici zdroje zvuku v rovině, je nutné mít měření z alespoň třech mikrofونů, které leží na odlišných místech[14]. Z těchto třech změřených signálů lze utvořit dva páry mikrofонů (např. 3,1 a 2,1).

⁵Toto zpoždění se v praxi považuje za konstantní, přestože se může například v důsledku proměnlivého vytížení sítě měnit.

⁶Přesnost je závislá na konkrétní implementaci, typicky je nižší než $1\mu\text{s}$.

Zpoždění z každého páru mikrofónů (M_i, M_j) dává informaci o tom, jak vypadá hyperbola na obr. 2.3. Pokud máme dvě zpoždění ze třech mikrofónů, je možné sestavit dvě takovéto hyperboly a zdroj zvuku pak leží na jejich průsečíku.

Hyperboly pochopitelně mohou mít více než jeden průsečík a pak bude třeba stanovit, který průsečík odpovídá skutečné poloze zdroje. Tuto situaci bude ovšem možné podstatně zjednodušit vhodnou volbou souřadného systému, jak uvádí [13].

Klíčovou částí procesu lokalizace zdroje zvuku bude algoritmus pro odhad zpoždění dvou signálů, který bude nutno zvolit a vhodně implementovat.

Pro určení zpoždění dvou signálů slouží například algoritmus Dominant Frequency Selection Algorithm (DFSE), který nejprve určí dominantní frekvenci obsaženou v signálu a následně se snaží hledat zpoždění v signálech pro tuto frekvenční složku.

Jeho nevýhodou je komplikovaná implementace a zanedbání velkého množství informace, kterou signály nesou. V důsledku toho může v určitých situacích docházet k velké chybovosti odhadů zpoždění.

Jako vhodnější algoritmus by mohla být použita tzv. Generalized Cross Correlation (GCC), která poskytuje poměrně dobré výsledky pro široké spektrum vstupních signálů, díky čemuž se v praxi často používá.

Nevýhodou GCC je její závislost na informaci o amplitudě signálu, která se ale v mnou řešeném případě bude nepochybně lišit, jelikož vzdálenější mikrofón bude mít zcela jistě nižší amplitudu než ten, co je blíže ke zdroji zvuku.

Z výše zmíněných důvodů se jako nejvhodnější z prostudovaných postupů jeví tzv. GCC-PHAT algoritmus. Jde v podstatě o rozšíření GCC o hodnotící funkci, kterou se násobí jednotlivé vektory signálů vstupující do algoritmu odhadu zpoždění.

Tuto hodnotící funkci, jak ukazuje sekce 4.3, bude možno stanovit tak, aby ze signálů odebrala informaci o amplitudě, ale zanechala informaci o fázi signálů, která přímo určuje jejich zpoždění.

2.4 Cíl práce

Cílem této práce je navrhnout synchronní distribuovaný sledovací systém, jehož základem bude distribuovaná měřicí ústředna, která bude složená z jednotlivých modulů, které si mezi sebou synchronizují čas pomocí protokolu PTP, který je definován normou IEEE-1588. Na každém modulu bude osazen mikroprocesor typu ARM z rodiny STM32, který disponuje několika AD převodníky, z nichž každý má vícero kanálů, na kterých dokáže měřit napětí.

Jelikož bude muset použitý procesor kromě měření také odesílat data po Ethernetu, synchronizovat si svůj čas a také komunikovat s nadřazenou jednotkou (typicky s osobním počítačem), bylo rozhodnuto o použití real-time operačního systému - RTOS. Ten umožňuje současný běh několika procesů zároveň pomocí plánovače, který jim přiděluje omezené časové rámce, ve kterých mohou využívat procesor a ostatní systémové prostředky. RTOS umožňuje taktéž vytváření semaforů a mutexů, které jsou klíčové pro synchronizaci jednotlivých běžících úloh mezi sebou, což je nezbytný předpoklad pro korektní implementaci zadání.

V další části práce bude dále nutné vytvořit SW pro PC a ověřit způsob přenášení naměřených dat mezi jednotlivými moduly a nadřazeným PC. Se vzrůstajícím počtem modulů rostou také nároky na přenosovou kapacitu sítě Ethernet a její využití v čase.

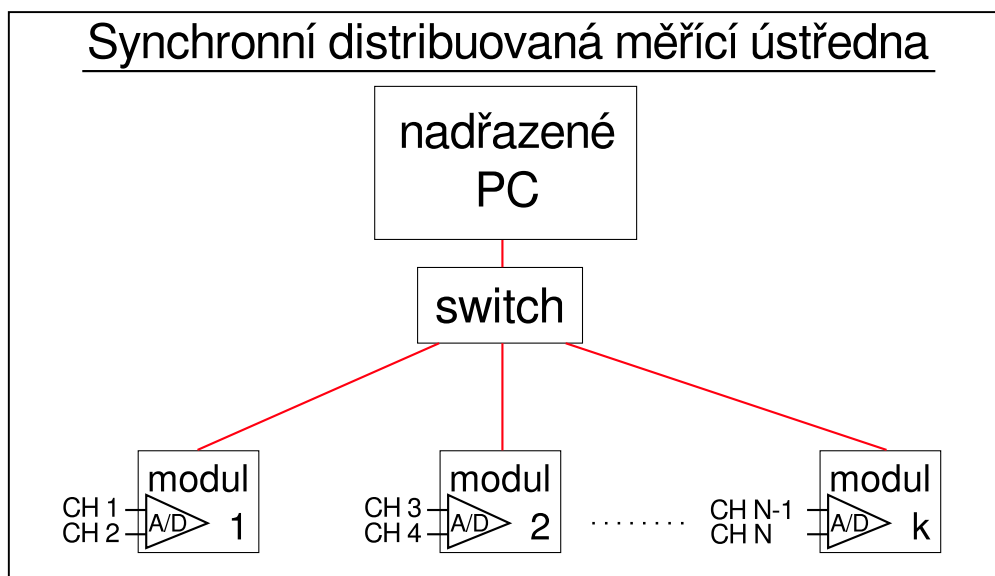
Proto je potřeba navrhnout přenosový protokol tak, aby minimalizoval kolize dat na síti a umožnil průchod taktéž řídicích dat a dat sloužících k časové synchronizaci pro jednotlivé moduly.

Řešením by měl být časově multiplexovaný přístup na sběrnici, kde má každý modul vyhrazeno jedno "časové okno" a pouze v tomto oknu může vysílat objemné soubory naměřených dat. Ovládací SW pro PC bude takto odeslaná data zapisovat do Comma-Separated Values (CSV) souborů, které následně mohou být snadno načteny ve velké většině programů určených pro zpracování dat.

Poslední částí mé práce bude ověření funkčnosti celého systému (především přesnosti časové synchronizace) pomocí lokalizace zdroje zvuku snímaného mikrofony připojenými k jednotlivým deskám. V této části odvodím rovnice pro výpočet souřadnic zdroje zvuku ve 2-D ze zpoždění signálů mezi třemi mikrofony. Výsledná pozice zdroje zvuku se bude vypočítávat a s mírným zpožděním (méně než jedna vteřina) i zobrazovat v prostředí MATLAB®.

3 SYNCHRONNÍ DISTRIBUOVANÁ MĚŘÍCÍ ÚSTŘEDNA

Jak již bylo zmíněno v kapitole 2, klíčovou součástí distribuovaného sledovacího systému je měřicí ústředna, která je tvořena moduly s mikrokontrolery ARM a doprovodným softwarovým vybavením pro PC.



Obr. 3.1: Blokové schéma synchronní distribuované měřicí ústředny

Blokové schéma měřicí ústředny ukazuje obr. 3.1. Jednotlivé moduly distribuované měřicí ústředny (modul 1 až modul k) měří napětí na vstupech A/D převodníků (CH 1 až CH n) a naměřená data posílají dále prostřednictvím přepínače (switch) po rozhraní Ethernet do nadřazeného PC.

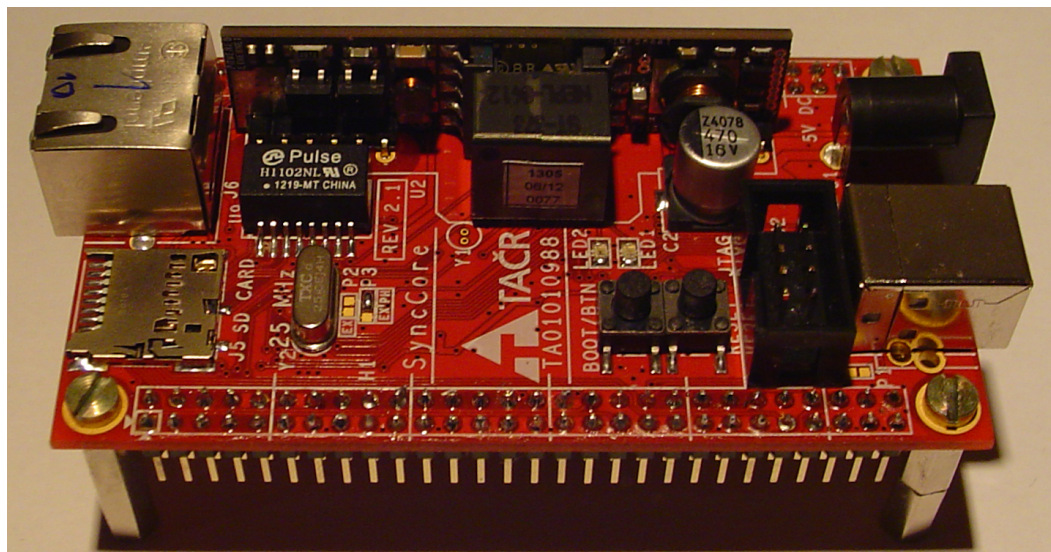
Nadřazené PC řídí veškeré moduly, sbírá z nich naměřená data a ukládá je do přehledně organizovaného CSV souboru, který může být následně dále zpracován.

Rozhraní Ethernet (v obrázku označeno červeně) kromě přenosu řídicích zpráv a naměřených dat slouží také k synchronizaci jednotlivých modulů mezi sebou pomocí protokolu PTP. Spoje rozhraní Ethernet mohou být vedeny klasickým metalickým vedením na vzdálenosti do 100m. Pokud je potřeba delší spoj, je možno využít optické či bezdrátové spoje, které mohou prodloužit vzdálenost mezi moduly a ústřednou až na desítky kilometrů.

3.1 HW vybavení ústředny

3.1.1 Modul ústředny

Měřicí ústředna je složena z modulů o rozměrech 86x49mm, které jsou osazeny mikrokontrolerem ARM z rodiny STM32. Tento modul vznikl na katedře měření v rámci práce na grantu Technologické agentury České republiky¹ a jeho vzhled je vidět na obrázcích 3.2 a 3.4.



Obr. 3.2: Pohled na modul měřicí ústředny shora

Na fotografii 3.2 můžeme na levé straně vidět stříbrný konektor rozhraní Ethernet, pod nímž je umístěn konektor pro SD kartu, která může sloužit například k ukládání konfigurace pro modul a umožňuje tak například změnu parametrů modulu pouhou výměnou paměťové karty, tedy bez zásahu do programu mikrokontroleru.

Dominantním prvkem na horní straně desky plošných spojů je kolmo zapájený napájecí modul Power over Ethernet (PoE). Tento prvek umožňuje napájet celé zařízení z rozhraní Ethernet, což ve výsledku znamená, že stačí k modulu vést pouze jeden kabel, který následně přenáší jak data tak napájení pro modul.

Napájecí PoE modul byl zakoupen jako již hotový polotovár a je osazen čipem AS1135 od společnosti Akros Silicon², který je plně kompatibilní se standardem PoE napájení definovaným normou IEEE 802.3at-2009.

¹ Webové stránky TAČR je možné najít zde: <http://www.tacr.cz/index.php/cz/> .

² Webové stránky výrobce: <http://www.akrossilicon.com> .

Specializovaný integrovaný obvod je pro PoE napájení nezbytný, jelikož výše zmíněný standard definuje sadu zpráv a příkazů, které si nejprve musí vyměnit přepínač (zdroj PoE napájení) a řídicí elektronika napájecího modulu (spotřebiče), než je zapnuto napájení pro dané zařízení. Pokud úspěšně proběhne tato komunikace, může dané zařízení odebírat až 625mA, což je více než dvojnásobek maximální spotřeby zde popisovaných modulů synchronní distribuované ústředny.



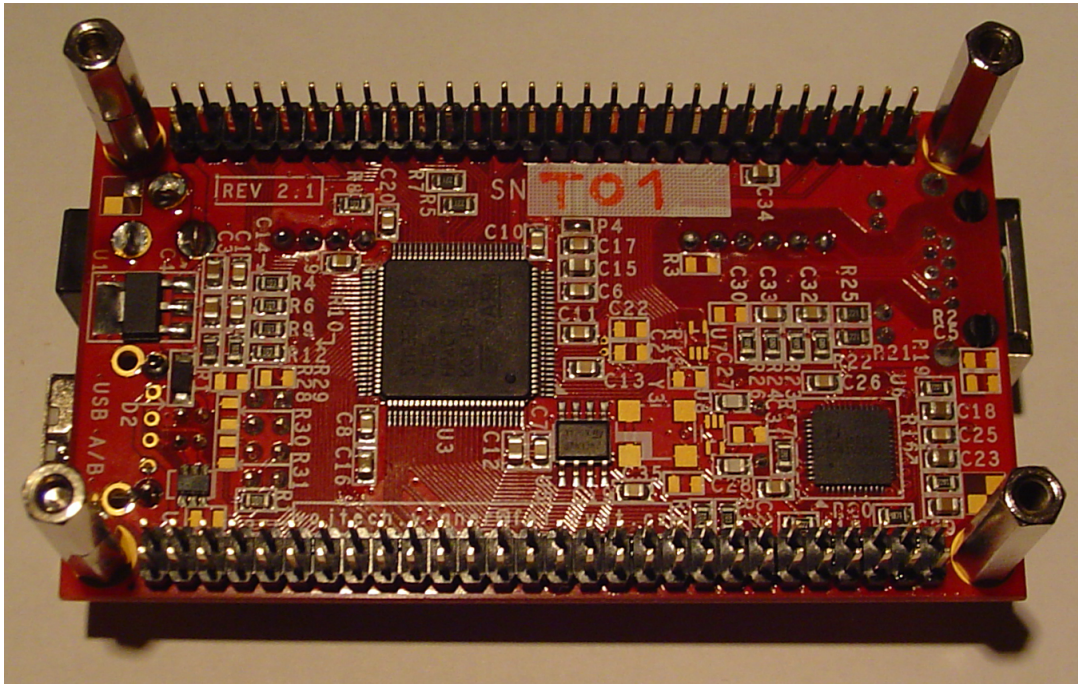
Obr. 3.3: Fotografie použitého přepínače s PoE

Aby bylo možno jednotlivé moduly napájet z rozhraní Ethernet, je třeba použít přepínač (switch), který dokáže poskytovat PoE napájení pro jednotlivá zařízení k němu připojená. Zástupcem takových zařízení je přepínač zobrazený na obr. 3.3, což je běžně dostupný přepínač TL-SF1008P od společnosti TP-LINK, který byl použit při vývoji této diplomové práce.

Obrázek 3.2 také ukazuje, že modul ústředny je možné napájet krom PoE taktéž z běžného černého souosého konektoru o vnitřním průměru 2,5mm, nebo ze stříbrného USB B konektoru, které jsou oba situovány na pravé straně modulu.

V případě napájení z 2,5mm souosého konektoru je potřeba vybrat takový zdroj, který poskytuje stejnosměrné napětí v rozstahu 4 až 20V a dokáže dodat proud alespoň 300mA.

Pokud se rozhodneme napájet modul z USB konektoru, pak musíme taktéž zajistit, že bude možno odebírat proud až 300mA, což znamená, že není doporučeno



Obr. 3.4: Pohled na modul měřící ústředny zespodu

napájet tyto moduly ze stolního počítače či laptopu, jelikož jsou jejich USB porty schopny poskytnout typicky pouze 100mA. Na vyžádání je jeden port USB 2.0 schopen poskytnout 500mA, ale musí tomu předcházet komunikace s připojovaným zařízením, což není pro tuto aplikaci použitelné, jelikož modul vyžaduje vyšší napájecí proud okamžitě po připojení USB kabelu.

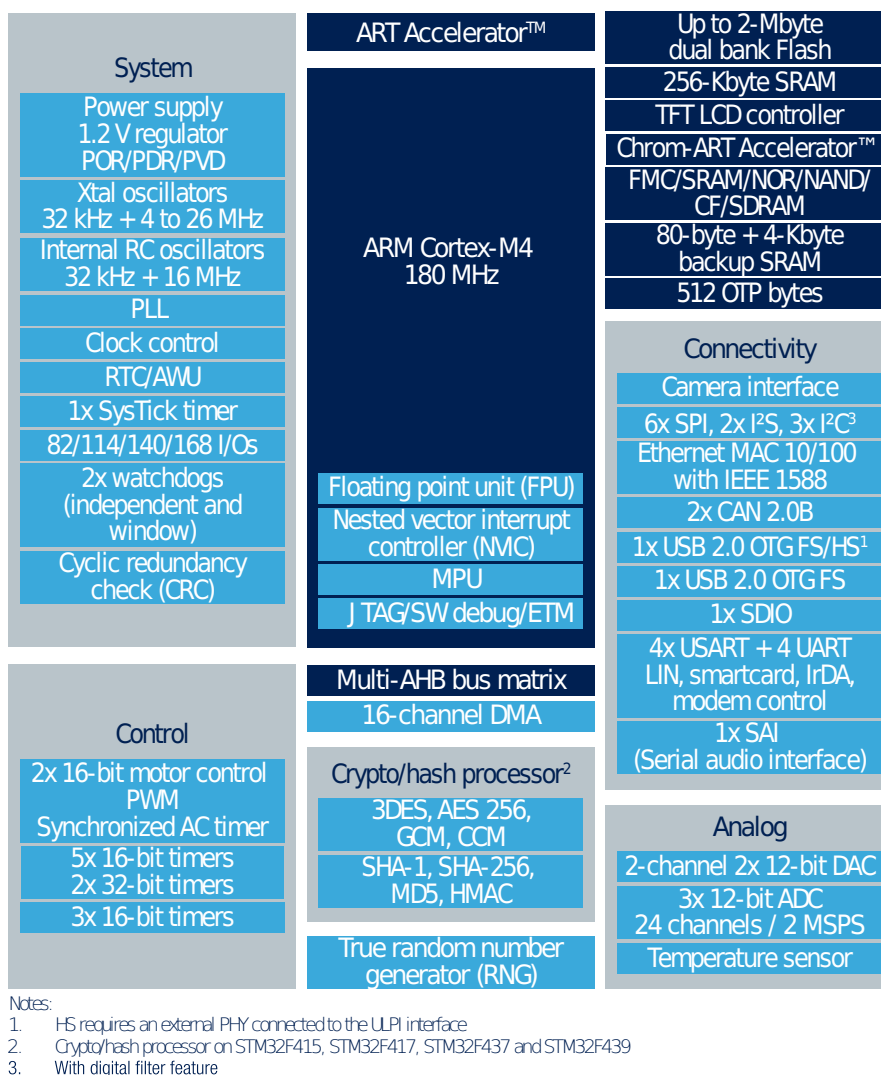
Pro napájení pomocí USB konektoru na modulu jsou vhodné síťové adaptéry s 5V výstupem, které mají USB konektor namísto klasických sousoých. Takovéto adaptéry se běžně dodávají například jako nabíječky k mobilním telefonům či MP3 přehrávačům.

Obr. 3.4 ukazuje pohled na modul zespodu. Zde jsou patrné dva klíčové integrované obvody. Mírně vlevo je vidět pouzdro LQFP100, které patří mikrokontroleru, a v pravém spodním rohu je pak umístěna fyzická vrstva rozhraní Ethernet.

Na obr. 3.4 jsou taktéž patrné dva rozšiřující konektory umístěné na horní a spodní straně, z nichž každý má 56 pinů. Díky tomuto páru konektoru se dá každý modul rozšířit o další desku plošných spojů, která se na konektory nasune a zajistí šrouby. Konektory poskytují jak napájení rozšiřující desky, tak i vstupy A/D převodníků a více jak 60 digitálních vstupně výstupních pinů mikrokontroleru.

3.1.2 Mikrokontroler

Jako řídicí prvek na modulu byl zvolen mikrokontroler ARM z rodiny 32-bitový mikrokontroler s jádrem ARM cortex-M3 od firmy ST Microelectronics (STM32), konkrétně model STM32F407[9] z tzv. connectivity line, tj. MCU s rozšířenou nabídkou komunikačních periférií integrovaných přímo na čipu. Blokové schéma použitého MCU z rodiny STM32 můžeme vidět na obr. 3.5³.



Obr. 3.5: Blokové schéma Connectivity line MCU z rodiny STM32

Mikrokontroler obsahuje především ethernetovou 10/100Mb Media Access Control (MAC) vrstvu s podporou IEEE-1588, což je pro danou aplikaci klíčové. Dále

³Obrázek byl převzat ze stránek výrobce: www.st.com.

pak disponuje řadou běžných periférií jako jsou časovače, A/D, D/A převodníky, vstupně-výstupní brány, Universal Serial Bus (USB), Controller Area Network (CAN) či výpočetní jednotka Cyclic Redundancy Check (CRC).

Samotný procesor jako takový disponuje dostatečným výpočetním výkonem, který výrobce udává jako 1,25 DMIPS/MHz⁴, tj. při maximálním taktovací frekvenci jádra MCU 168MHz dokáže vykonat až 210 milionů instrukcí za sekundu[9].

Pro uložení programu mikrokontroleru slouží flash paměť, která má u použitého MCU velikost 1024kB. Paměť pro data je typu Static Random Access Memory (SRAM) a její kapacita je 192kB.

K nahrání programu do paměti procesoru slouží Joint Test Action Group (JTAG) resp. Serial Wire Debug (SWD) rozhraní, která zároveň dovolují řídit běh programu v MCU a prohlížet jednotlivé registry v paměti i přímo v jádru procesoru v průběhu vykonávání programu. Výhodou SWD je nízký počet vodičů a s tím související malý konektor, kdežto JTAG má standardně dvacetipinový konektor, ale na druhou stranu dosahuje vyšších přenosových rychlostí.

V průběhu vývoje jsem využíval téměř výhradně SWD rozhraní, které je možno vidět na obr. 3.2. Těsně za USB konektorem se nachází černý osmipinový konektor J2, který představuje ladící konektor, na němž je krom SWD rozhraní vyvedeno také napájení pro modul.

A/D převodníky mikrokontroleru

Velmi důležitou součástí měřicí ústředny jsou A/D převodníky, které jsou schopny převádět analogová napětí na digitální číslo uvnitř použitého mikrokontroleru. Mezi základní parametry každého A/D převodníku patří rozlišení a rychlost vzorkování.

Rozlišení A/D převodníků se udává v počtu bitů, které reprezentují konvertovanou hodnotu. Mikrokontrolery z rodiny STM32 disponují dvanáctibitovými převodníky a to jim umožňuje rozdělit vstupní napětí na 4096 kvantizačních kroků, což je dostatečná hodnota pro velkou většinu zamýšlených aplikací.

Pokud by ovšem bylo vyžadováno vyšší rozlišení, je možné připojit A/D převodník s vyšším rozlišením prostřednictvím rozšiřujících konektorů.

⁴DMIPS - počet celočíselných operací procesoru za sekundu v milionech

Pokud by naopak bylo požadováno nižší rozlišení, je možno A/D převodník v MCU nastavit korm dvanácti bitů také na šest nebo osm bitů. Výhodou snížení rozlišení je nižší objem dat, které je nutno přenés do nadřazeného PC, přičemž přesnost může být pro konkrétní aplikací stále dostatečná.

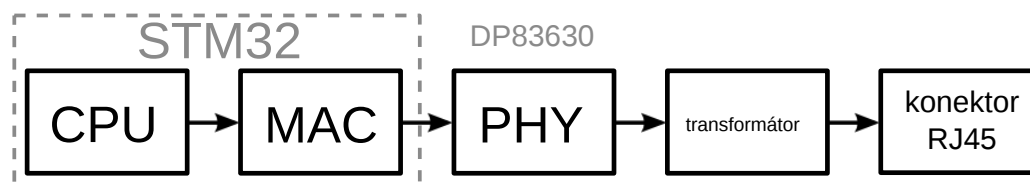
Dalším neméně důležitým parametrem je vzorkovací frekvence, která určuje, jak často je schopen daný A/D převodník odebrat a konvertovat vzorky analogového signálu na vstupu převodníku.

Použitý mikrokontroler z rodiny STM32 disponuje třemi A/D převodníky, které mají dohromady 24 měřících kanálů (vývodů MCU), z nichž některé jsou společné pro všechny tři převodníky.

Každý z vnitřních A/D převodníků může vzorkovat s maximální frekvencí 2.4MSa/s (resp. 2.4MHz), nicméně použitý mikrokontroler umožňuje taktéž zřetězit všechny tři dostupné převodníky a díky společným kanálům mohou měřit všechny tři na jednom pinu MCU (není potřeba jejich vstupy spojovat externě). Toto zřetězení a interní spojení vstupů A/D převodníků umožňuje vzorkovat maximální rychlostí 7.2MSa při maximálním rozlišení 12 bitů.

3.1.3 Fyzická vrstva rozhraní Ethernet

Zde použitý mikrokontroler obsahuje pouze MAC vrstvu, která se stará o správné adresování, řízení přístupu ke komunikačnímu kanálu a případnou detekci kolizí v případě, že v jeden okamžik začne vysílat více stanic. MAC vrstva ovšem není schopna sama vysílat data po síti Ethernet. K tomu potřebuje Ethernet physical transceiver (PHY) - fyzickou vrstvu Ethernetu, což je integrovaný obvod obsahující standardizované komunikační rozhraní Reduced Media Independent Interface (RMII) (případně Media Independent Interface (MII)) pro komunikaci s MAC vrstvou, řídicí a kontrolní obvody a výstupní budiče pro pulsní transformátor, za nímž je zapojen ethernetový konektor. Celou situaci ilustruje obr. 3.6.



Obr. 3.6: Blokový diagram propojení mikrokontroleru a fyzické vrstvy

Central Processing Unit (CPU) spolu s MAC je obsažena na samotném čipu connectivity line MCU z rodiny STM32. K STM32 je pomocí RMII resp. MII rozhraní připojena fyzická vrstva PHY, která je v tomto případě řešena čipem DP83630 od společnosti Texas Instruments[10]. Následuje pulsní transformátor galvanicky oddělující modul od sítě Ethernet a standardní ethernetový konektor.

Výše zmiňovaná fyzická vrstva je vhodná pro použití v systémech, které využívají protokol PTP, jelikož ji výrobce vybavil několika funkcemi, které pracují s PTP protokolem značně zjednodušují a především umožňují dosahovat nižších odchylek od referenčního času. První a nejdůležitější funkcí je přiřazování časových značek k packetům již na HW úrovni samotné fyzické vrstvy, takže časová značka opravdu odpovídá době, kdy byl packet odeslán a nikoli době, kdy byla v mikrokontroleru zavolána funkce pro odeslání dat po Ethernetu.

Další velkou výhodou zde popisované fyzické vrstvy je fakt, že disponuje přesným vlastním oscilátorem s malým jitterem (viz příloha B), jehož odchylka od nominální frekvence je navíc neustále kompenzována dle údajů poskytnutých PTP protokolem.

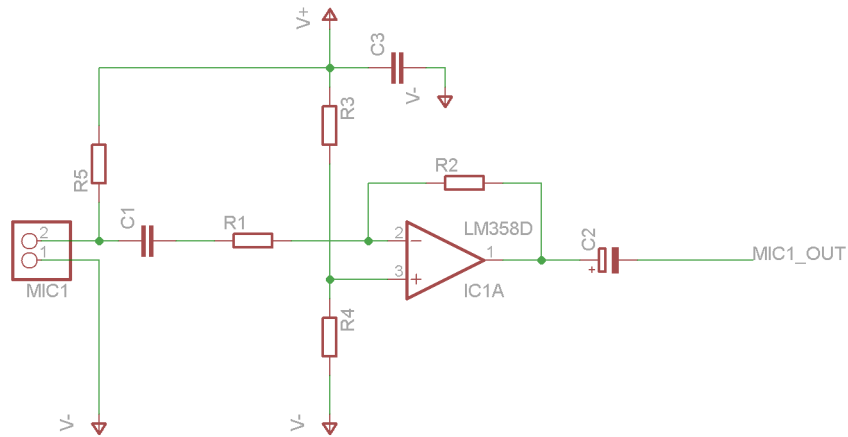
Výstup oscilátoru fyzické vrstvy je dostupný na jednom z jejích pinů, ze kterého je možné přivést hodinový takt do mikrokontroleru. Tímto lze ušetřit jeden krystal či krystalový oscilátor, který by jinak byl nezbytný ke běhu MCU.

Poslední velmi dobře využitelnou vlastností použité fyzické vrstvy je přítomnost dvanácti GPIO pinů, které je možno nakonfigurovat tak, aby v zadaný čas přesly z log. 0 do log. 1 a tím pádem mohou být použity pro vyvolání přerušování v mikrokontroleru.

3.1.4 Mikrofonní předzesilovač

Pro potřebu akustické lokalizace, které se věnuje následující kapitola 4, byla navržena Deska Plošných Spojů (DPS), na níž je realizován předzesilovač pro mikrofonní vozku.

Na obr. 3.7 můžeme vidět schéma mikrofonního předzesilovače. Samotná elektretová mikrofonní vložka (konkrétně typ MCE100) je připojena na vstupní svorky označené MIC1. Rezistor R5 poskytuje napájení pro elektretovou vložku, zatímco kondenzátor C1 slouží k oddělení tohoto stejnosměrného signálu.



Obr. 3.7: Schéma mikrofonního předzesilovače

Zesilování signálu z mikrofonu má na starosti operační zesilovač v invertujícím zapojení. Jelikož nemáme k dispozici souměrné napájení vzhledem k zemi, je potřeba vybrat takový operační zesilovač, který je schopen pracovat pouze s jedním napájecím napětím. Zástupcem této skupiny OZ je zde použitý LM358.

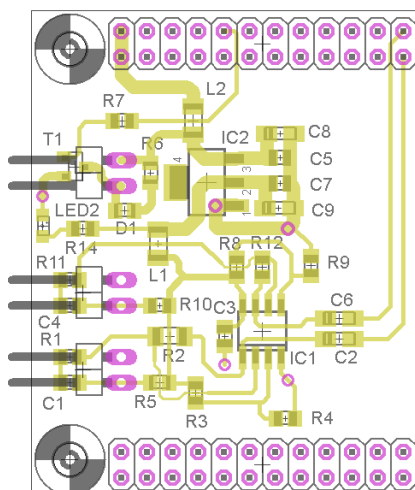
Aby mohl zesilovač správně pracovat, je potřeba přivést na neinvertující vstup OZ pomocí napěťového děliče tvořeného rezistory R3,R4 přibližně polovinu napájecího napětí. Toto zapojení se v literatuře označuje jako "virtuální nula" a umožňuje výstupu OZ rozkmit výstupního napětí na obě strany vzhledem k napětí přivedenému na neinvertující vstup.

$$A_{CL} = -\frac{R2}{R1} \quad (3.1)$$

Zesílení operačního zesilovače v invertujícím zapojení je dáno vztahem 3.1. Pomocí série experimentů jsem dospěl k závěru, že optimální zesílení pro toto zapojení je 10^3 . Rezistor $R1 = 2,2k\Omega$ jsem zvolil tak, aby byl stejný jako výstupní impedance elektretové vložky. Rezistor $R2 = 2,2M\Omega$ lze pak snadno dopočítat z výše zmíněného vztahu.

Vzhledem k tomu, že OZ LM358 obsahuje dvojici operačních zesilovačů, byl by jeden z nich nevyužitý a proto jsem se rozhodl na výsledné DPS realizovat hned dvojici mikrofonních předzesilovačů.

Díky použití malých SMD součástek zbylo na plošném spoji ještě dostatek místa pro budič pro piezo reproduktor a LED diodu, která indikuje přítomnost napájecího



Obr. 3.8: Plošný spoj mikrofonního předzesilovače

napětí.

Veškeré výrobní podklady včetně seznamu součástek a jejich hodnot je možné nalézt v příloze A nebo na přiloženém CD.

3.2 Program pro modul

Obslužný program byl napsán v jazyce C a vyvinut ve vývojovém prostředí Sleepy Cat IDE⁵. Jedná se o robustní prostředí, které používá kompilátor GCC⁶ modifikované pro procesory ARM. Oba výše zmíněné nástroje jsou dostupné zdarma a tím pádem neovlivnily náklady na vývoj tohoto zařízení.

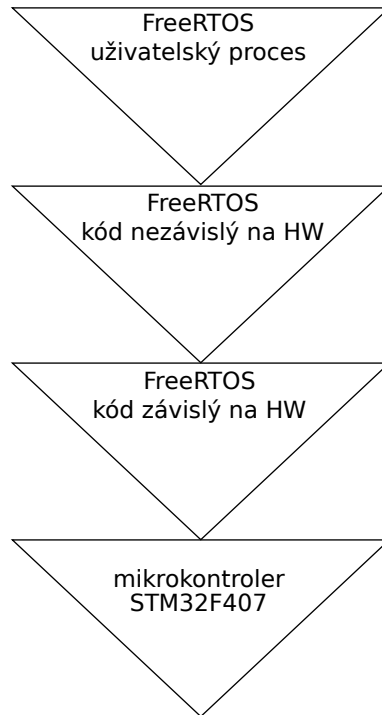
3.2.1 Operační systém modulu

Základním stavebním kamenem, nad kterým je vystavěna architektura celého programu pro MCU, je FreeRTOS. Jedná se o real-time operační systém určený do malých zařízení s omezeným množstvím paměti a výpočetního výkonu. Jedná se o operační systém v plném slova smyslu, jelikož spravuje veškeré systémové prostředky a přerozděluje je právě běžícím procesům dle jejich priorit. Hlavní přednosti FreeRTOS je možné vidět v následujícím seznamu.

- možnost preemptivního plánování

⁵Vývojové prostředí a dokumentaci je možné nalézt na webové stránce: <http://pck338-242.feld.cvut.cz/scide/>. Autorem je Ing. Jan Breuer z kateřy měření.

⁶Aktuální verzi a dokumentaci je možné nalézt na: <https://launchpad.net/gcc-arm-embedded>.



Obr. 3.9: Znázornění členění systému FreeRTOS na jednotlivé vrstvy

- možnost kooperativního plánování více procesů současně
- velikost 6 až 10kB v závislosti na konfiguraci
- možnost obslužení HW přerušeni od MCU
- snadné předávání zpráv mezi jednotlivými procesy
- mutexy s děděnou prioritou
- binární a čítací semaforey

Jak ukazuje obr. 3.9, FreeRTOS se dá rozdělit na kód závislý na použitém HW a na vyšší SW vrstvy, které jsou již na HW nezávislé. Z pohledu vývojáře je kritická především implementace první části, tj. kódu závislého na daném mikrokontroleru. Zde se dá s výhodou využít již hotové implementace od výrobce MCU, společnosti ST Microelectronics. Takto připravený kód je pouze nutné upravit pro hladké zkompilování ve výše popisovaných vývojových prostředcích s ohledem na specifika kompilátoru GCC.

Nejvyšší částí RTOS jsou uživatelské procesy, kterých je v rámci mé práce definováno šest.

- driver Ethernetového rozhraní
- lwIP stack
- PTPd server

- Standard Command for Programmable Instruments (SCPI) server
- proces plnící naměřená data do odesílacích zásobníků
- proces odesílající naměřená data ze zásobníků
- proces plánjící následující časy odesílání dat
- Hypertext Transport Protocol (HTTP) server

Driver Ethernetového rozhraní sestává ze dvou částí, z nichž první se stará o řízení MAC vrstvy mikrokontroleru a druhá řídí fyzickou vrstvu PHY realizovanou obvodem DP83630 popsaném v sekci 3.1.3. Obě tyto části jsou poskytovány výrobcem ST Microelectronics a to jak jako samostatné knihovny, tak i jako procesy pro FreeRTOS.

Jakožto TCP/IP a UDP stack byl použit lwIP stack popsaný v sekci 2.1.4. Stejně jako výše popsaný Ethernetový driver i lwIP stack, resp. jeho implementace pro použitý mikrokontroler STM32F407, je dostupný na stránkách výrobce a připravený pro použití.

Stejně jako v předchozích případech i PTPd server je již připraven od výrobce a jedná se o upravenou verzi PTP daemonu pro prostředí linux, jehož původní zdrojové kódy včetně dokumentace lze naléznout na stránkách autora⁷.

Posledním ze seznamu procesů je HTTP server, který v rámci celého systému nehraje velkou roli a byl využíván především při vývoji tohoto systému pro zaznamenávání důležitých zpráv o stavu systému a jejich následném publikování na webové stránce. Samotný server nabízí široké možnosti využití a to i z důvodu, že je rozšířený o JavaScript Object Notation (JSON), což je nadsavba jazyka JavaScript, která umožňuje generování aktivních webových stránek, které jsou pak schopny například zobrazovat stav jednotlivých pinů mikrokontroleru nebo dokonce měnit jejich stav v závislosti na uživatelské akci.

Zbýlé procesy FreeRTOS jsou popsány v následující sekci 3.2.2, která se věnuje vlastní implementaci distribuovaného měřicího systému.

3.2.2 Distribuovaný systém

Základem programu řídicím modul jsou čtyři následující procesy.

- SCPI server
- proces plnící naměřená data do odesílacích zásobníků

⁷Stránky autora včetně dokumentace jsou na adrese: <http://ptpd.sourceforge.net> .

- proces odesílající naměřená data ze zásobníků
- proces plánjící následující časy odesílání dat

Řízení distribuovaného systému

Řídícím prvkem měřicí ústředny SCPI server, který má za úkol poskytovat jednoduché a unifikované rozhraní pro ovládání distribuovaného systému prostřednictvím rozhraní Ethernet. Tento server vyvinul pan Ing. Breuer⁸. Díky ovládání pomocí SCPI příkazů je práce s modulem distribuovaného systému velice podobná práci např. s digitálním multimetrem připojeným přes General Purpose Interface Bus (GPIB) rozhraní. V následujícím seznamu můžeme nalézt základní SCPI příkazy, které slouží k ovládání distribuovaného systému.

- *DIST:RUN 1* - spuštění měření a odesílání dat
- *DIST:RUN 0* - ukončení měření a odesílání dat
- *DIST:RUN?* - dotaz na stav měření (spuštěno/zastaveno)
- *DIST:TIME?* - dotaz na čas začátku měření
- *DIST:SYNC* - příkaz vykonávající synchronizaci měřících vláken mimo stanovenou synchronizační periodu

Plnění naměřených dat do zásobníků

Vzhledem k potřebě měřit velké objemy dat bylo nutné vytvořit způsob, jakým se budou aktuálně naměřená data ukládat v paměti mikrokontroleru. Jak již bylo zmíněno v sekci 3.1.2, použitý mikrokontroler disponuje pouze 192kB paměti SRAM, která je navíc ještě z velké části obsazena operačním systémem a daty z ostatních běžících procesů. Proto zde bylo s výhodou využito Direct Memory Access (DMA) controlleru, který je součástí mikrokontroleru. Ten umožňuje tzv. double-buffering, což je periodické ukládání vzorků do dvou nezávislých zásobníků (bufferů), mezi kterými je přepínáno vždy po naplnění jednoho z nich.

Díky tomuto můžeme v určitém okamžiku přečíst jeden ze zásobníků (ten do kterého se právě nezapisuje) a máme jistotu, že data v něm obsažená nebudou v průběhu čtení prepisována. Toto umožňuje kontinuální měření a odesílání dat, o které se stará následující proces.

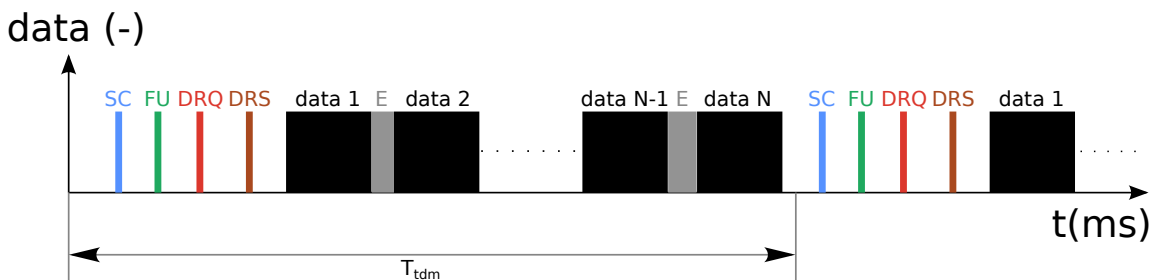
Odesílání naměřených dat

Pokud máme v zásobníku naměřená data, je potřeba je odeslat do PC. Vzhledem k tomu, že začátek odměru je pro všechny moduly stejný, je pro ně stejný

⁸Dokumentace i zdrojové kódy jsou zde: <http://jaybee.cz/software/open-source-scpi-ieee-488.2-parser-library/>.

také okamžik, kdy mají naměřené zásobníky a potřebovaly by data odeslat po síti Ethernet. Pokud by toto však dělaly všechny moduly současně, docházelo by k mnohačetným kolizím na síti, které by vyústily ve ztrátu dat.

Časový multiplex Aby k takovýmto jevům nedocházelo, je nutné rozdělit kapacitu přenosového media (sítě Ethernet) mezi jednotlivé moduly předem definovaným způsobem, který respektuje časové nároky každého modulu na odeslání daného objemu naměřených dat.



Obr. 3.10: Znázornění časového multiplexu

Tento problém jsem se rozhodl řešit pomocí časového multiplexu, tj. rozdělení přístupu k rozhraní Ethernet na časové rámce, z nichž každý je přiřazen jednomu modulu. Způsob řešení ilustruje Obr. 3.10. Zde je vidět, že kromě naměřených dat z jednotlivých modulů (*data 1* až *data N*) je potřeba přenášet také synchronizační pakety protokolu PTP (barevné sloupce *SC*, *FU*, *DRQ*, *DRS*).

Vzhledem k tomu, že celý distribuovaný měřicí systém je postaven na synchronizaci času pomocí PTP je potřeba umožnit volný průchod těchto synchronizačních zpráv tak, aby nekolidovaly s odesílanými daty z jiných modulů. Na toto navržený časový multiplex bere zřetel a žádný z modulů nezačne odesílat data dokud neproběhla výměna synchronizačních zpráv v aktuální synchronizační periodě PTP protokolu, která má v mém případě standardní trvání jednu sekundu.

Perioda časového multiplexu je v obrázku označena T_{tdm} a určuje po jaké době začne znovu ten samý modul odesílat nově naměřená data. Hodnotu periody je potřeba zvolit tak, aby v jejím rámci bylo možné přenést veškerá naměřená data do PC a vyměnit si synchronizační zprávy protokolu PTP.

Abych dále omezil možnost kolizí dat na síti způsobených například mírně zpožděným začátkem odesílání dat jednoho z modulů (toto zpoždění může být

způsobeno např. čekáním na mutex, který v danou chvíli vlastní jiný proces), zavedl jsem tzv. chybové okno, což je v Obr. 3.10 naznačeno šedým obdélníčkem s popisem *E*. Toto okno vymezuje časovou oblast, ve které by žádný z modulů neměl vysílat data, pokud začal vysílat data včas. Pokud by však došlo ke zpožděnému odesílání, tak data odeslaná po konci časového rámce modulu (tj. již odeslaná v chybovém okně), dorazí v pořádku do PC, jelikož v chybovém okně vysílá pouze onen "zpožděný" modul a jím odeslaná data nemohou kolidovat s daty od ostatních modulů.

Planování běhu jednotlivých procesů

Funkce posledního procesu distribuované měřicí ústředny je poměrně jednoduchá. Jedním z úkolů je zjišťovat aktuální stav měřicího a odesílacího procesu (popsáno výše) a blokovat/odblokovávat jejich běh pomocí mutexů v závislosti na aktuálních potřebách systému.

Druhým úkolem tohoto procesu je plánovat následující časy spouštění měřicího a odesílacího procesu v rámci vymezeného časového okna pro daný modul. Tento proces, narozdíl od předchozích dvou, spotřebuje pouze zanedbatelné množství paměti a výpočetního výkonu, jelikož krom výpočtu následujícího časového okna pouze dohlíží na správný běh měření a odesílání dat.

3.3 Ovládací program pro PC

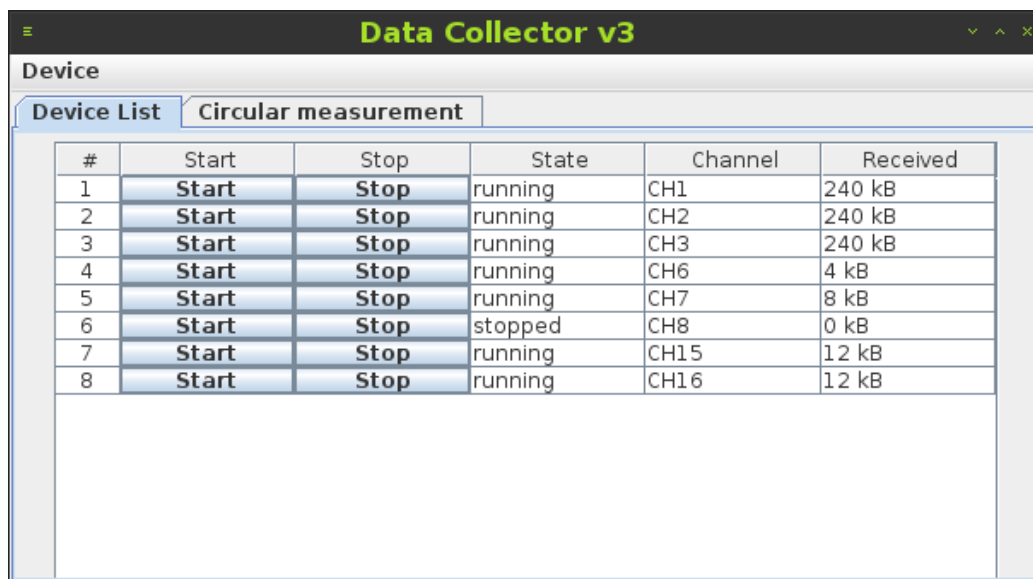
Veškeré moduly synchronní distribuované měřicí ústředny jsou řízeny z nadřazeného PC, které taktéž sbírá naměřené soubory dat z jednotlivých modulů. Za tímto účelem vznikla ovládací aplikace pro PC, jejíž ukázka je na obr. 3.11.

Aplikace byla napsána v jazyce JAVA a k jejímu vytvoření bylo použito vývojové prostředí NetBeans, které je volně dostupné ze stránek výrobce⁹.

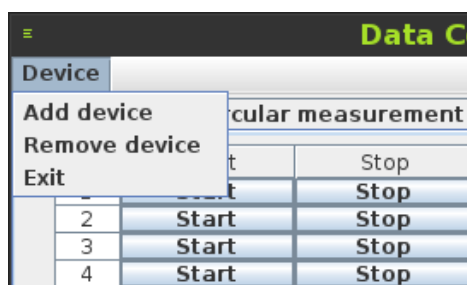
Program je multiplatformní a ke svému běhu vyžaduje běžné PC s 32-bitovým nebo 64-bitovým operačním systémem GNU/Linux, WindowsTM, Mac OS X nebo SolarisTM. Ať již použijeme jakýkoliv operační systém, je nutné, aby v něm bylo nainstalováno Java Runtime Environment (JRE) ve verzi 6 či vyšší¹⁰.

⁹Dokumentace je dostupná zde: www.netbeans.org .

¹⁰JRE je možné stáhnout zde: <http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html> .



Obr. 3.11: Ukázka ovládacího programu



Obr. 3.12: Menu umožňující správu modulů

Program je možné spustit dvěma způsoby. Buď použijeme připravený spouštěcí skript 'runDataCollector' (resp. 'runDataCollector.bat' pro WindowsTM), nebo můžeme spustit samostatný archiv s aplikací v adresáři, kde se nachází, pomocí příkazu:

```
java -jar DataCollector_v3.jar
```

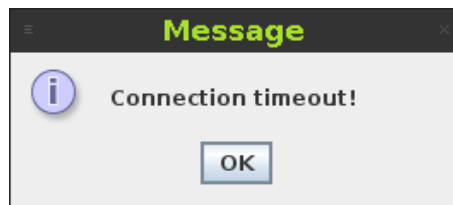
Po spuštění programu je nejprve nutné přidat alespoň jeden měřící modul, aby mohl obslužný program sbírat data. Přidání se provede pomocí menu *Device -> Add device*, které je možné vidět na obr. 3.12.

Pokud klikneme na *Add device*, objeví se dialogové okno (viz obr. 3.13), které se dotáže na konkrétní IP adresu přidávaného modulu.



Obr. 3.13: Dialogové okno pro zadání IP adresy

Pokud zadáme správnou IP adresu a PC se dokáže s modulem spojit (je připojeno ke stejné síti), tak se přidávaný modul objeví jako poslední řádka v tabulce tak, jak ukazuje obr. 3.11.



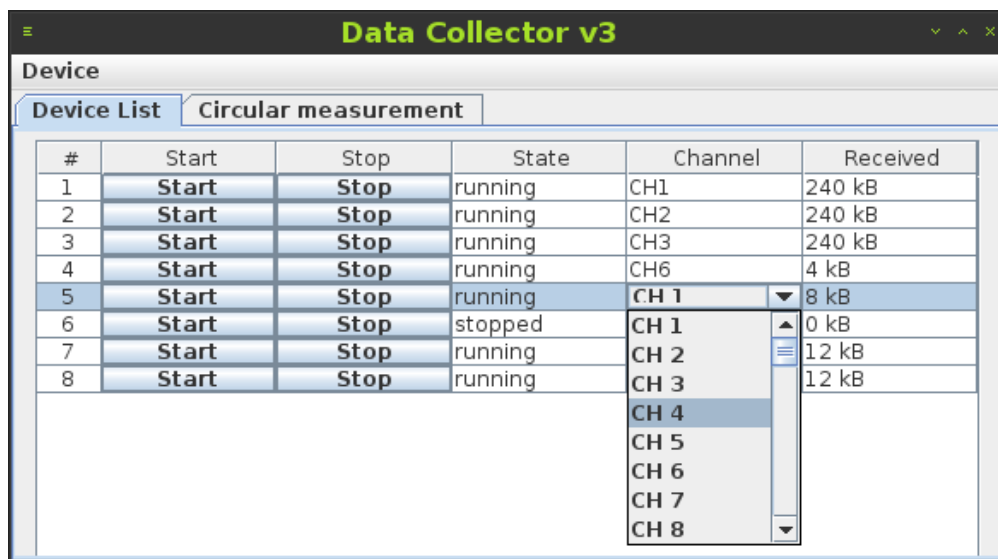
Obr. 3.14: Okno upozorňující na špatně zadanou IP adresu

V případě zadání špatné IP adresy se po dobu pěti vteřin snaží aplikace spojit se zařízením na zadané adrese, pokud dané zařízení neodpovídá, zobrazí se okno (viz obr. 3.14) informující o nedostupnosti zadaného zařízení.

V praxi bude často potřeba od sebe odlišit měřící kanály pomocí jednoznačných jmen zadaných před začátkem měření. K tomu slouží sloupec s názvem *Channel*, který po kliknutí zobrazí seznam možných názvů měřících kanálů, které nesou název *CH N*, kde *N* leží v intervalu 1 až 253 (maximální počet připojených modulů je 253). Celou situaci ilustruje obr. 3.15.

Uživatel si tedy může definovat, jak se bude jmenovat sloupec naměřených dat ve výstupním CSV souboru, bez ohledu na IP adresu modulu či pořadí, v jakém byl přidán do aplikace.

Pokud máme přidáný alespoň jeden modul v aplikaci a nastaveno jméno měřícího kanálu, můžeme odstartovat synchronní měření pomocí tlačítka *Start*.



Obr. 3.15: Volba kanálů pro jednotlivé moduly

Toto tlačítko pošle modulu zprávu o tom, že může začít měřit a odesílat data, ale samotný start měření nastává až s příchodem synchronizačního pulzu, který je veden z Pulse Per Second (PPS) vývodu fyzické vrstvy.

Díky tomu, že PPS pulzy jsou generovány na základě času korigovaného PTP protokolem, je maximální časový rozdíl mezi příchodem náběžné hrany PPS pulzu (která startuje měření) na jednotlivých modulech maximálně v řádu stovek nanosekund, což poskytuje synchronní soubor dat z modulů, kde maximální odchylka času odměru se od referenčního času liší nanejvýš o $1\mu s$.

Jak již bylo zmíněno, naměřená data se zapisují do CSV souboru. Pro každý modul se vytváří separátní soubor (jeho název je stejný jako IP adresa, pouze jsou tečky nahrazeny pomlčkami), kam jsou uložena naměřená data.

Na prvním místě v CSV souboru je vždy název kanálu tak, jak byl zvolen v ovládací aplikaci, následuje zalomení řádku a dále jsou zapisována již naměřená data. Na jeden řádek připadá vždy jedna hodnota, po které následuje čárka (znak ',') a časová značka, která udává okamžik, kdy byla hodnota změřena. Dále následuje zalomení řádky pomocí znaků Carriage Return (0x13) a Line Feed (0x10). Na následující řádce pokračuje zápis další hodnoty.

Pokud modul využívá více než jeden kanál A/D převodníku, jsou jeho podkanály označeny malými písmeny *CH Na*, *CH Nb*, *CH Nc*, atd. Ke každému podkanálu jsou

dopočítány časové značky a výsledný CSV soubor je tvořen vždy dvojicí sloupců pro každý podkanál zapsaných vedle sebe (CH 1, čas 1, CH 2, čas 2, CH 3, čas3, ...).

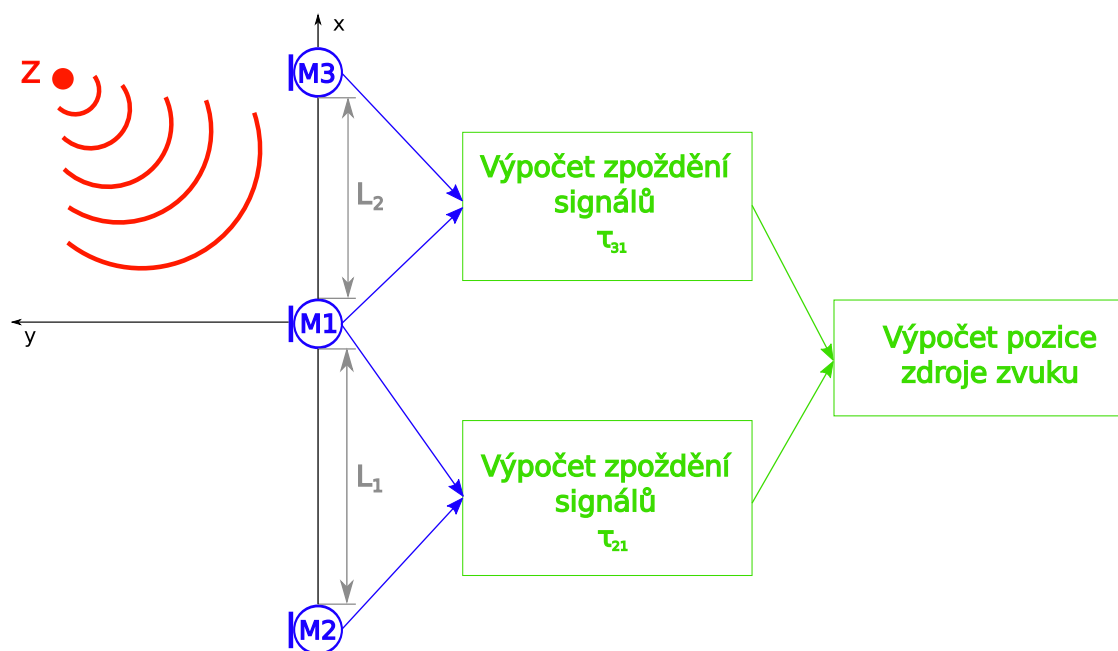
Speciální vlastností zde popisované aplikace je schopnost kruhového měření na více kanálech, které je využito k implementaci akustického sledovacího systému popisovaného v následující kapitole. Kruhové měření se spouští a vypíná pomocí dvou tlačítek na záložce *Circular measurement*, která je vidět na obr. 3.11.

4 AKUSTICKÝ SLEDOVACÍ SYSTÉM

Akustický sledovací systém slouží k určení pozice zdroje zvuku za pomoci vyhodnocování signálů zaznamenaných třemi mikrofony. Odhad pozice je následně vykreslen na obrazovce počítače.

4.1 Uspořádání sledovací soustavy

Složení celého synchronního distribuovaného sledovacího systému je vidět na obr. 4.1. Zdroj zvuku je označen Z. Modře jsou vyznačeny mikrofony snímající zvuk, který je zaznamenán příslušným modulem a poslán do PC. Na nadřazeném počítači je spuštěn program pro měřicí ústřednu a také prostředí MATLAB® (v obrázku zeleně), které provádí následné zpracování signálu a vykreslení pozice zdroje zvuku.



Obr. 4.1: Uspořádání sledovací soustavy

4.2 Předzpracování naměřených dat

Data získaná pomocí kruhového měření z měřicí ústředny jsou uložena do CSV souboru, který je tvořen šesti sloupci hodnot. Data jsou uložena tak, že liché sloupce obsahují naměřené hodnoty z jednotlivých mikrofonů a sudé sloupce obsahují časové značky odpovídající předchozímu sloupci.

Parametry signálů z mikrofonů Signál z mikrofonů je vzorkován s periodou $20\mu\text{s}$ a A/D převodník má nastaveno osmibitové rozlišení, které vstupní analogové napětí mění na hodnotu mezi 0 a 255. Počet řádků v CSV souboru odpovídá počtu změřených vzorků + 1 (první řádek obsahuje názvy kanálů). Množství odebraných vzorků je možno měnit v programu popisovaném v sekci 3.3. Pokud bychom například chtěli zaznamenat signály v trvání jedné vteřiny, nastavíme odběr na 50 000 vzorků.

Do prostředí MATLAB® jsou data nahrána pomocí funkce `csvread()`, což uloží načtený soubor ve formě matice do tzv. "workspace" prostředí MATLAB®, kde jsou jednotlivé signály (liché sloupce matice) uloženy do slupcových vektorů pojmenovaných `signal1` až `signal3`.

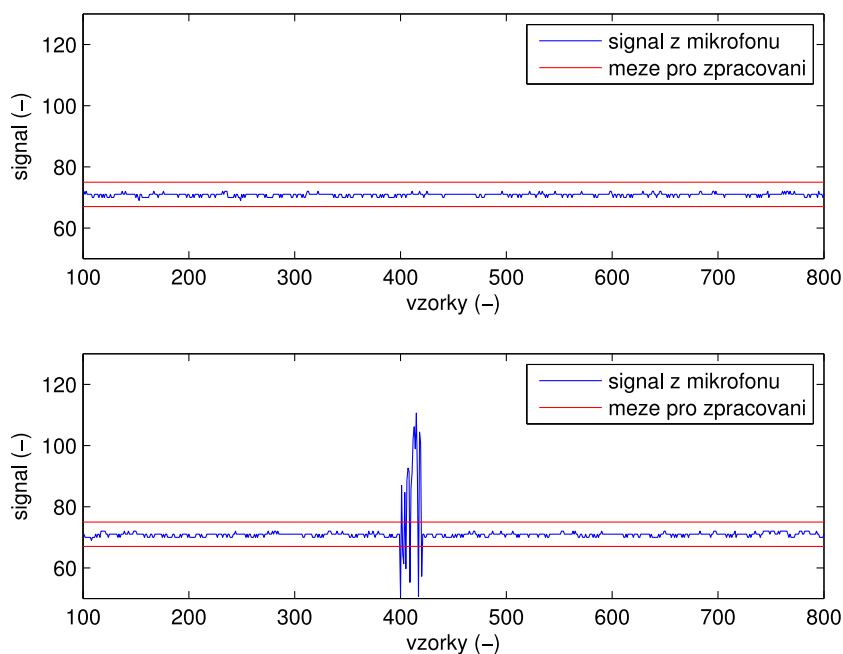
Vzhledem k tomu, že použité kruhové měření startuje veškeré odměry současně s maximální odchylkou v řádu stovek nanosekund, která je pro akustickou lokalizaci vzhledem k rychlosti zvuku zanedbatelná, nemusíme v tomto případě brát v úvahu časové značky k jednotlivým signálům, neboť nás zajímá pouze vzájemný časový posun signálů. Ten je možné spočítat jako posun v počtu prvků vektoru vynásobený vzorkovací periodou.

Díky současnému startování odměrů tedy stačí načítat z CSV souboru pouze vektory signálů bez jejich časových značek. Tento přístup zjednodušuje následnou přípravu na zpracování. Pokud by nebylo využito současného startování, bylo by nutné naměřené signály nejprve zarovnat dle stanovené časové značky a následně vybrat vhodný časový interval signálů ke zpracování.

Dalším krokem v předzpracování signálu je zjištění, zda naměřené signály obsahují dostatek informace na to, aby bylo možno určit jejich skutečné zpoždění.

Pokud například není v okruhu několika málo metrů kolem mikrofonů přítomen žádný zdroj zvuku, je zaznamenaný signál tvořen převážně vlastním šumem mikrofonů a elektromagnetickým rušením, jak ukazuje obr. 4.2 nahoře. Takový signál pochopitelně nemá smysl dále zpracovávat, neboť neobsahuje žádnou informaci o pozici zdroje zvuku.

Proto jsou u jednotlivých signálů nejprve zjištěna maxima a minima, která představují jednoduché vodítko pro určení, zda dané signály mohou být použity pro výpočet pozice.



Obr. 4.2: Ukázka signálů zaznamenaných mikrofonem

Pokud signál obsahuje pouze šum, nezmění se jeho amplituda o více než 4 (empiricky stanovená konstanta) od střední hodnoty signálu. Meze pro určení, zda jde pouze o šum, nebo již signál vhodný ke zpracování, jsou v obr. 4.2 naznačeny červenými čarami.

V případě, že signál překročí stanovené meze, jak ukazuje obr. 4.2 dole, je vyhodnocen jako vhodný ke zpracování. Pokud nepřekročí ani jednu z vyznačených mezí, nebo překročí pouze jednu, je vyhodnocen jako šum, dále se nezpracovává a následuje načtení nové sady signálů.

4.3 Stanovení zpoždění signálů

K odhadu podobnosti resp. vzájemného zpoždění dvou signálů slouží nejčastěji korelace. Mějme dány dva diskétní signály $s_1(k)$, $s_2(k)$, pak je jejich vzájemná korelace $R_{s_1s_2}(\tau)$ v časové oblasti dána vztahem 4.1. Vzájemné zpoždění je pak možno určit ze vztahu 4.2

$$R_{s_1s_2}(\tau) = \sum_{t=0}^{N-1} s_1(t)s_2(t-\tau) \quad \tau = -N + 1 \dots N - 1 \quad (4.1)$$

$$\tau_x = \operatorname{argmax}(R_{s_1 s_2}(\tau)) \quad (4.2)$$

Tento obecný přístup ovšem není příliš vhodný, jelikož dává spolehlivé výsledky pouze pokud signály obsahují minimum šumu. Rovnice 4.1 totiž ukazuje, že jsou brány v úvahu pouze amplitudy vstupních signálů.

Signály $s_1(k)$, $s_2(k)$ ovšem nesou informaci také o frekvenci a fázi signálu, kterou je možno získat přechodem do frekvenční oblasti pomocí Fourierovy transformace rovnice 4.1. Pro tyto případy je nejprve nutné přepsat rovnici tak, že korelaci upravíme na konvoluci (značeno $*$), jak ukazuje 4.3.

$$R_{s_1 s_2}(\tau) = \sum_{t=0}^{N-1} s_1(t) s_2(-(\tau - t)) = s_1(t) * s_2(t) \quad (4.3)$$

Nyní můžeme provést Fourierovu transformaci s využitím znalosti, že konvoluce dvou vektorů v časové oblasti představuje ve frekvenční oblasti prosté násobení jednoho vektoru \hat{S}_1 s vektorem komplexně sdružených čísel k vektoru \hat{S}_2 , který bude dále značen jako \hat{S}_2^* . Výsledek této operace ukazuje rovnice 4.4, přičemž odhad spoždění je stále možné nalézt pomocí vztahu 4.2.

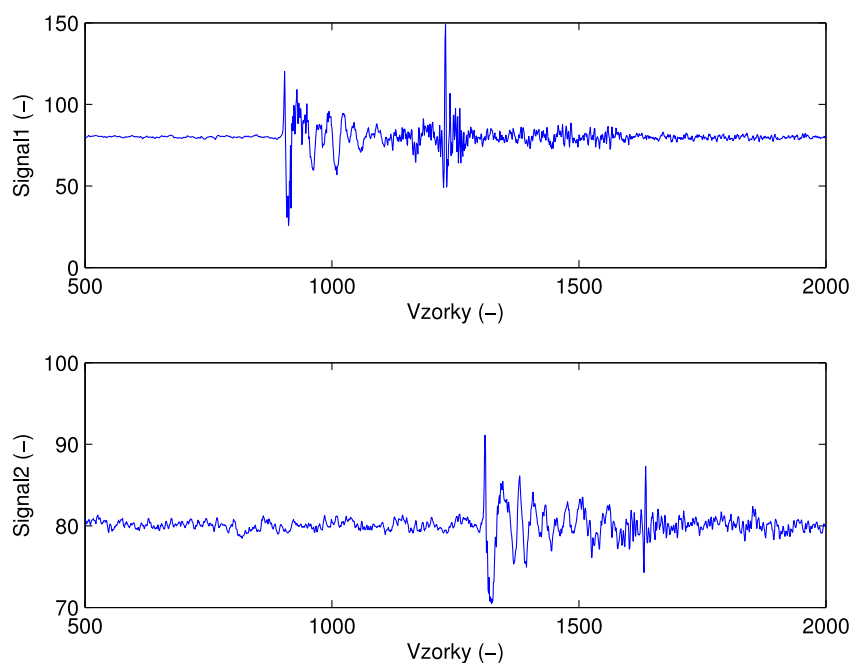
$$R_{s_1 s_2}(\tau) = \sum_{k=-N/2}^{N/2-1} \hat{S}_1(e^{j\omega}) \hat{S}_2^*(e^{j\omega}) e^{j\omega\tau} \quad \omega = \frac{2\pi k}{N} \quad (4.4)$$

Nyní se nabízí možnost použití GCC [12], což je korelace ve Fourierově spektru, která je navíc ještě násobena váhovou funkcí $\hat{\Psi}(e^{j\omega})$.

$$R_{s_1 s_2}(\tau) = \sum_{\omega} \hat{\Psi}(e^{j\omega}) \hat{S}_1(e^{j\omega}) \hat{S}_2^*(e^{j\omega}) e^{j\omega\tau} \quad (4.5)$$

Tvar váhové funkce $\hat{\Psi}(e^{j\omega})$ je možno definovat tak, potlačil vliv amplitud signálů a korelace se počítala pouze na základě informace o fázi, kterou oba signály obsahují. Tato informace je obzvláště důležitá, neboť jejich vzájemný fázový posuv přímo udává časové zpoždění mezi nimi. Proto jsem se rozhodl použít váhovou funkci, kterou ukazuje rovnice 4.6.

$$\hat{\Psi}_{PHAT}(e^{j\omega}) = \frac{1}{|\hat{S}_1(e^{j\omega})| |\hat{S}_2(e^{j\omega})|} \quad (4.6)$$

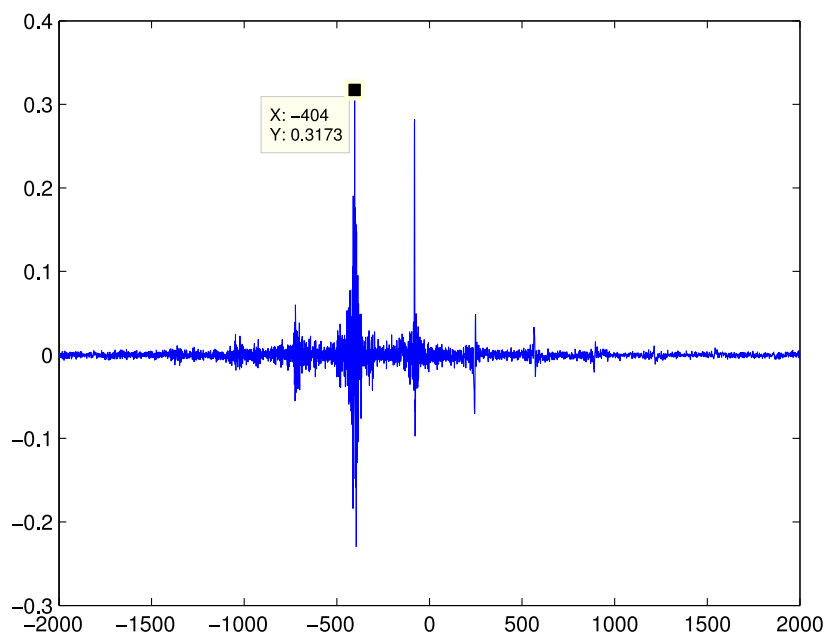


Obr. 4.3: Příklad signálů zaznamenaných mikrofonom

Použití GCC s výše popsanou funkcí se často v literatuře nazývá fázová transformace a označuje se GCC-PHAT. Takto definovaná korelace je velmi vhodná pro zpracování reálných signálů, které se v případě mikrofónů mohou vlivem vzdálenosti od zdroje zvuku výrazně lišit v amplitudě, nicméně tvar signálu zůstává zachován.

Praktickou ukázkou zpracování reálných signálů z obr. 4.3 pomocí zde popsaného postupu je možné vidět na obr. 4.4. Jak ukazuje kurzor v obrázku, odhad zpoždění, který představuje maximum ve výstupním vektoru algoritmu GCC-PHAT, vychází -404 vzorků (signál 1 předbíhá signál 2 o 404 vzorků, proto má záporné znaménko). Jejich časové zpoždění v sekundách je možno spočítat vynásobením počtu vzorků vzorkovací periodou. V tomto případě je tedy signál 2 zpožděn za signálem 1 o $404 * 0.00002 = 8,08 * 10^{-3}$ s.

Pro zvýšení spolehlivosti odhadu zpoždění je v rámci sledovacího systému implementován výpočet histogramu z deseti opakovaných měření a vybrán je ten odhad, který byl ze všech pozorování nejfrekventovanější.



Obr. 4.4: Výstupní vektor algoritmu GCC-PHAT

4.4 Výpočet pozice zdroje zvuku

Mějme dány dva mikrofony M1, M2 se známými souřadnicemi (x_1, y_1) , (x_2, y_2) . Zpoždění, s jakým dorazí signál od zdroje Z s poicí (x, y) k jednomu z mikrofonů, pokud je známa rychlost zvuku v_z , lze vypočítat z rovnice 4.7.

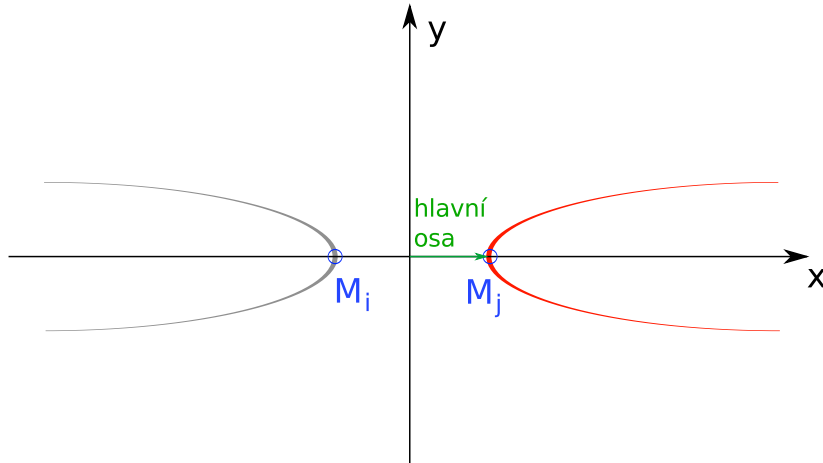
$$\tau_i = \frac{1}{v_z} \sqrt{(x - x_i)^2 + (y - y_i)^2} \quad (4.7)$$

Zpoždění, s jakým dorazí akustický signál ke vzdálenějšímu z mikrofonů oproti tomu bližšímu, označíme τ_{ij} a lze ho vypočítat následovně.

$$\tau_{ij} = \tau_i - \tau_j = \frac{1}{v_z} \left(\sqrt{(x - x_i)^2 + (y - y_i)^2} - \sqrt{(x - x_j)^2 + (y - y_j)^2} \right) \quad (4.8)$$

Rovnice 4.8 koresponduje s hyperbolou na obr. 4.5, v jejíž ohniscích jsou mikrofony M_i, M_j a její hlavní osa je dána $v_z \tau_{ij} / 2$, kde v_z je rychlost zvuku.

Pokud je zpoždění mezi signály kladné, leží zdroj zvuku na té části hyperboly, která se nachází v pravé polorovině kartézské souřadné soustavy (v obrázku červeně). Je-li zpoždění záporné, znamená to, že zdroj zvuku leží v levé polorovině.



Obr. 4.5: Hyperbola daná zpožděním dvou signálů

Abychom byli schopni určit jednoznačně určit polohu zdroje v rovině, je potřeba znát zpoždění z alespoň dvou párů mikrofonů, které leží na rozdílných souřadnicích. Díky znalosti druhého zpoždění můžeme z rovnice 4.8 sestavit soustavu dvou hyperbolických rovnic, jejichž řešení je možno hledat například iterativně.

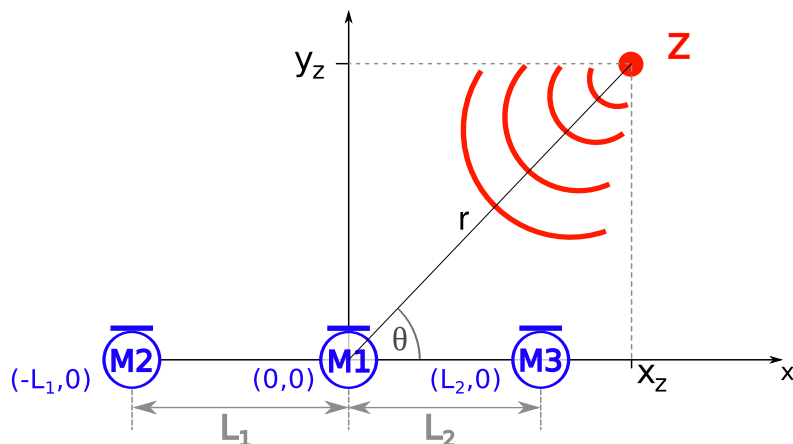
Jiným přístupem k řešení rovnic je postup, který popsal Dr. Y. T. Chan [14]. Ten vytvořil algebraické řešení soustavy hyperbolických rovnic pro speciální případ třech mikrofonů 4.9, které je snadno řešitelné a poskytuje unikátní reálné řešení.

$$\begin{bmatrix} x_z \\ y_z \end{bmatrix} = - \begin{bmatrix} x_2 & y_2 \\ x_3 & y_3 \end{bmatrix}^{-1} \times \left(\begin{bmatrix} r_{21} \\ r_{31} \end{bmatrix} r + \frac{1}{2} \begin{bmatrix} r_{21}^2 - K_2 + K_1 \\ r_{31}^2 - K_3 + K_1 \end{bmatrix} \right) \quad (4.9)$$

Přičemž konstanty r_{21} , r_{31} a K_1 až K_3 jsou následující.

$$\begin{aligned} r_{21} &= \tau_{21} v_z \\ r_{31} &= \tau_{31} v_z \\ K_1 &= x_1^2 + y_1^2 \\ K_2 &= x_2^2 + y_2^2 \\ K_3 &= x_3^2 + y_3^2 \end{aligned}$$

Pokud máme všechny tři mikrofony v jedné linii, jak ukazuje obr. 4.6, můžeme definovat souřadnou soustavu tak, že osa x je dána spojnicí mikrofonů. Osa y směřuje kolmo na membrány mikrofonů a prochází středem mikrofonu M1.



Obr. 4.6: Rozložení mikrofonů v souřadném systému

$$\begin{bmatrix} x_z \\ r \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} r_{21}^2 - L_1^2 \\ r_{31}^2 - L_2^2 \end{bmatrix} \times \begin{bmatrix} -L_1 & r_{21} \\ L_2 & r_{31} \end{bmatrix}^{-1} \quad (4.10)$$

Umístění všech mikrofonů do jedné přímky a mikrofonu M1 do počátku souřadné soustavy má za následek výrazné zjednodušení rovnice 4.9. Ta se pak dá upravit do tvaru 4.10, ze kterého můžeme následně vyjádřit jednu ze souřadnic zdroje a také jeho vzdálenost r . V tomto případě je ovšem snazší vyjádřit azimut θ pomocí poměru $\frac{x_z}{r}$ a z něho dopočítat obě souřadnice pomocí trigonometrických vzorců.

$$r = \frac{L_1 \left(1 - \left(\frac{r_{21}}{L_1} \right)^2 \right) + L_2 \left(1 - \left(\frac{r_{31}}{L_2} \right)^2 \right)}{2 \left(\frac{r_{31}}{L_2} + \frac{r_{21}}{L_1} \right)} \quad (4.11)$$

$$\theta = \cos^{-1} \left(\frac{x_z}{r} \right) = \cos^{-1} \left(\frac{L_2^2 - 2rr_{31} - r_{31}^2}{2rL_2} \right) \quad (4.12)$$

$$x = r \cos(\theta) \quad (4.13)$$

$$y = r \sin(\theta) \quad (4.14)$$

Výše uvedené rovnice umožňují ze znalosti zpoždění třech signálů zaznamenaných mikrofony spočítat pozici zdroje zvuku. Podrobné odvození veškerých vztahů uvádí například [13], [14].

4.5 Dosažené výsledky

Pro ověření funkčnosti výše popsaného postupu jsem vytvořil sadu algoritmů, které jsou pomocí časovače periodicky spouštěny a s 500ms periodou vykreslují vypočtenou pozici zdroje zvuku (sledovaného objektu).

- `[s1, s2, s3] = loadDataFromCsv2('jmenoSouboru.csv')`
 - načtení signálů ze souboru do workspace
- `[s1n, s2n, s3n] = processData(s1, s2, s3)`
 - předzpracování signálů dle 4.2
- `[\tau_{21}, \tau_{31}] = getTDOA(s1n, s2n, s3n)`
 - určení zpoždění signálů
- `[\tau_{21}^*, \tau_{31}^*] = getTimeDifferencesHist(\tau_{21}, \tau_{31})`
 - histogramový filtr odhadů zpoždění dle 4.3
- `plotPosition(\tau_{21}^*, \tau_{31}^*)`
 - výpočet pozice a její zobrazení dle rovnic 4.11 až 4.14

Zde popisované pokusy probíhaly v běžné místnosti (bez protiodrazového obložení stěn) o půdorysu 5,5 x 7m a výšce 2,6m. Vzdálenost mezi jednotlivými mikrofony byla $L_1 = L_2 = 0,15m$ a rychlost zvuku byla stanovena na $v_z = 342.21m \cdot s^{-1}$. Jednotlivé moduly měřící ústředny včetně mikrofونů jsem položil na stůl o výšce 0,8m a rozmístil je přesně dle obr. 4.6.

Vzhledem k tomu, že v rovnicích pro výpočet pozice zdroje zvuku figurují dvě proměnné θ a r , je možné pomocí opakovaných měření při známé (stacionární) pozici zdroje zvuku určit střední hodnotu a směrodatnou odchylku ze souboru naměřených dat. Takto vypočtené hodnoty následně mohou porovnat s reálnými.

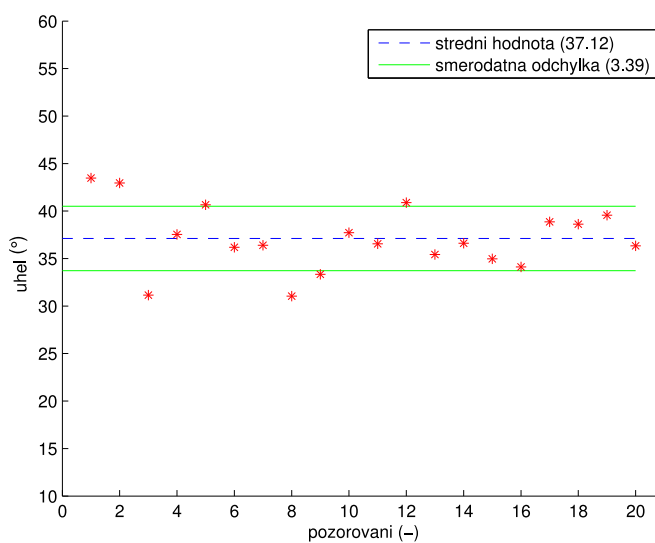
4.5.1 Odchylka určení azimutu θ

Pro určení statistických parametrů měření sloužil pokus, kde zdroj zvuku byl umístěn do prosturu a byly pečlivě změřeny jeho souřadnice, azimut a vzdálenost od počátku souřadného souřadného systému (radius). Veškeré údaje jsou uvedeny v tab. 4.1.

Následující graf ukazuje jednotlivé změřené hodnoty pro dvacet pozorování s časovými rozestupy 500ms. Vzorkovací frekvence pro signály z mikrofونů byla nastavena na 50kHz.

parametr	symbol	hodnota
azimut	θ	35°
radius	r	$1m$
délka (x)	x_z	$0,82m$
výška (y)	y_z	$0,58m$

Tab. 4.1: Parametry pokusu pro určení odchytky θ a r



Obr. 4.7: Série dvaceti měření azimutu

Zelené čáry vyznačují pásmo jedné směrodatné odchytky a modrá přerušovaná čára udává střední hodnotu souboru naměřených dat.

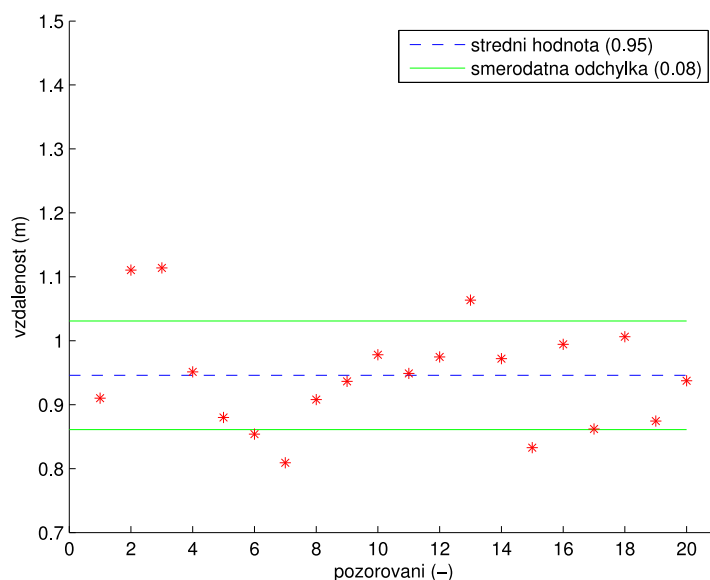
Z grafu a tab. 4.2 je patrné, že střední hodnota vypočteného azimutu (z dvaceti statisticky nezávislých pozorování) je $37,72^\circ$. Od skutečné se tedy v průměru liší o $2,72^\circ$. Směrodatná odchytka tohoto měření byla $3,39^\circ$. Relativní chyba měření činí 6.05% .

symbol	skutečná hodnota	střední hodnota	max	min	směr. odchytka
θ	$35,0^\circ$	$37,12^\circ$	$42,37^\circ$	$32,72^\circ$	$3,39^\circ$

Tab. 4.2: Výsledky pokusu pro určení úhlu θ

4.5.2 Odchylna určení vzdálenosti r

Určení odchylny radiusu (vzdálenosti) r bylo provedeno se stejnými parametry (viz tab. 4.1) jako v předchozím případě. Bylo provedeno dvacet pozorování statického zdroje zvuku s 500ms rozestupy při vzorkovací frekvenci 50 kHz. Na obr. 4.8 můžeme vidět výsledek jednotlivých měření.



Obr. 4.8: Série dvaceti měření vzdálenosti r

Z grafu a tab. 4.3 je patrné, že střední hodnota vypočteného radiusu je 0,95m. Od skutečné se tedy v průměru liší o 5 cm. Směrodatná odchylna tohoto měření byla 8cm.

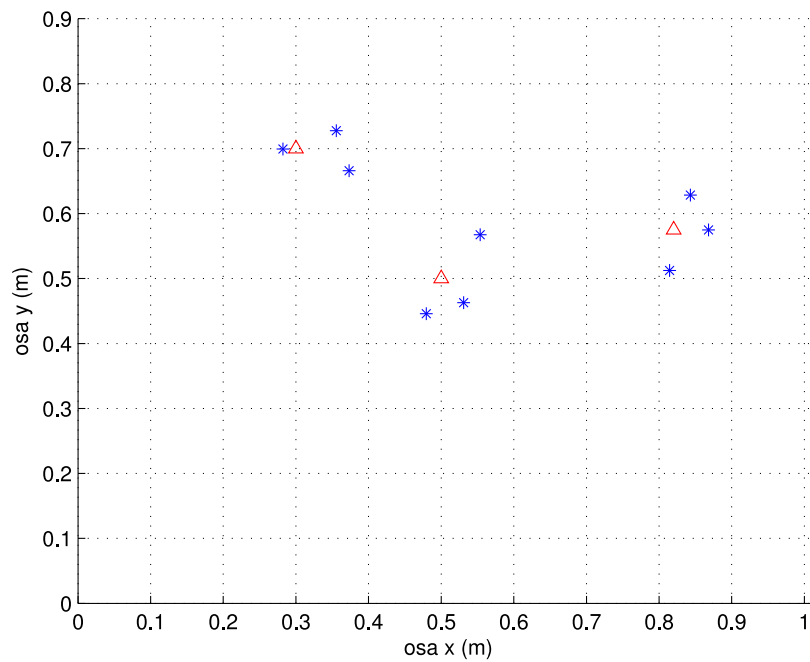
symbol	skutečná hodnota	střední hodnota	max	min	směr. odchylna
r	1,0m	0,95m	1,12m	0,82m	0,08m

Tab. 4.3: Výsledky pokusu pro určení vzdálenosti r

V tomto případě bylo dosaženo mírně přesnějších výsledků než při určování azimutu vzhledem k tomu, že relativní chyba měření činí pouze 5.0%.

4.5.3 Odchytky vypočtených souřadnic

V tomto pokusu bylo provedeno devět měření, vždy tři pro každou pozici zdroje zvuku. Signál z mikrofonů byl vzorkován frekvencí 50 kHz. Na obr. 4.9 jsou červenými trojúhelníky vyznačeny skutečné pozice zdroje zvuku a body * ukazují změřenou pozici.



Obr. 4.9: Rozložení mikrofonů v souřadném systému

Tab. 4.4 porovnává skutečné a změřené pozice zdroje zvuku pro všech devět měření. Zde lze vidět, že maximální odchylka pro souřadnici x_z byla 8cm a pro y_z 7cm.

skutečná pozice (m, m)	změřená pozice (m, m)	odchylka ($\Delta x(m)$, $\Delta y(m)$)
(0,82 , 0,58)	(0,81, 0,51)	(0,01, 0,07)
	(0,87, 0,58)	(-0,05, 0,00)
	(0,84, 0,62)	(-0,02, -0,04)
(0,50 , 0,50)	(0,49, 0,44)	(0,01, 0,06)
	(0,53, 0,46)	(-0,03, 0,04)
	(0,55, 0,57)	(-0,05, -0,07)
(0,30 , 0,70)	(0,29, 0,70)	(0,01, 0,00)
	(0,38, 0,68)	(-0,08, 0,02)
	(0,35, 0,72)	(-0,05, -0,02)

Tab. 4.4: Výsledky pokusu pro určení polohy zdroje zvuku

5 ZÁVĚR

Cílem práce bylo navrhnout a realizovat synchronní distribuovaný sledovací systém, jehož základem bude distribuovaná měřicí ústředna, ovládací SW pro stolní počítač a programové vybavení pro výpočet pozice zdroje zvuku ze zpoždění signálů změřených mikrofony.

V rámci práce bylo vytvořen program pro mikrokontroler, který realizuje základní měřicí funkce. Každý modul je možné ovládat pomocí komunikačních příkazů definovaných v kapitole 3.

Mikrokontroler je základním stavebním kamenem modulu distribuovaného systému. Tento obvod řídí veškeré procesy, které modul vykonává a spolu s fyzickou vrstvou rozhraní Ethernet umožňuje komunikaci s ostatními moduly a PC.

Na modulech byl implementován synchronizační protokol PTP definovaný normou IEEE-1888 z roku 2008. Tento protokol umožňuje synchronizovat lokální zdroje času, které jsou součástí každého modulu, s referenčním zdrojem času, který může být tvořen kterýmkoliv z modulů, nadřazeným PC či speciálním zdrojem přesného času odvozeného od Global Positioning System (GPS).

Bylo ověřeno, že díky využití protokolu PTP je odchylka lokálních zdrojů času od referenčního menší než jedna mikrosekunda (řádově stovky nanosekund). Při využití speciálních síťových přepínačů s podporou PTP protokolu bylo dosaženo odchylky menší než sto nanosekund.

Aby nekolidovaly velké objemy naměřených dat z modulů (které data průběžně odesílají do PC) se synchronizačními značkami PTP protokolu, který je vzhledem k distribuované povaze systém klíčový, byl navržen časový multiplex, který dělí přístup ke sběrnici Ethernet do časových rámců, které zašťují vyhrazené pásmo pro synchronizační zprávy.

Časový multiplex také zajišťuje separátní časové rámce pro odesílání dat z jednotlivých modulů, takže jednotlivé moduly neodesílají data současně. Tím je předcházeno periodickým naporům na přenosovou kapacitu rozhraní Ethernet.

Dále bylo vytvořeno SW vybavení pro nadřazený počítač, prostřednictvím kterého je možné ovládat jednotlivé moduly a ukládat data, která jsou prostřednictvím

modulů měřena. Data jsou uložena ve formě CSV souboru, kde každý sloupec dat reprezentuje jeden kanál měřící ústředny.

Po vytvoření funkčního prototypu distribuované měřící ústředny bylo možno začít realizovat akustický sledovací systém, jehož vstupní část je tvořena mikrofony s předzesilovačem a byla v rámci práce navržena a realizována na desce plošných spojů.

K periodické lokalizaci objektu byl rozšířeno programové vybavení měřící ústředny o funkci kruhového měření, které periodicky plní naměřená data do souboru o konstantí velikosti (tedy jednotlivá data jsou posouvána).

V rámci práce byl navržen a implementován algoritmus GCC-PHAT pro odhad zpoždění signálů a vytvořeny metody předzpracování dat a filtrování výstupních hodnot s ohledem na fyzikální zákonitosti spojené s akustickou lokalizací v uzavřené místnosti.

Finální fází diplomové práce bylo ověření správnosti realizace celé zde popisované práce na kontinuální lokalizaci (sledování) zdroje zvuku v rovině pomocí třech mikrofonů rozmístěných v místnosti.

Výsledky všech pokusů jsou zdokumentovány v sekci ?? a to jak pro určení vzdálenosti zdroje od počátku souřadné soustavy či azimutu, tak pro výpočet samotných souřadnic zdroje.

Dosažené výsledky jsou vzhledem k chybě určení radiusu a azimutu poměrně dobré a souhlasí také s přesností popisovanou v literatuře [11], [13] nebo [14].

LITERATURA

- [1] Yiu J. *The definitive guide to the ARM Cortex-M3* Elsevier, 2007, ISBN 978-0-7506-8534-4
- [2] Sloss A., Symes D., Wright C. *ARM System developers guide* Elsevier, 2004, ISBN 1-55860-874-5
- [3] Zurawski R. *Networked embedded systems* CRC Press, 2009, ISBN 978-1-4398-0761-3
- [4] David Tse, Pramod Viswanath *Fundamentals of Wireless Communication* Cambridge University Press, 2005, ISBN 978-0-5218-4527-4
- [5] P. Marshall, J. Rinaldi *Industrial Ethernet ISA*, 2009, ISBN 978-1-5561-7892-4
- [6] Ereth McKnight-MacNeil, B.Sc. *CS-MNS: Analysis and Implementation* Carleton University, 2010
- [7] ARM Limited *Cortex-M4TM - Technical Reference Manual* Publ. ARM DDI 0337B, 2008
- [8] ST Microelectronics *RM0008 - Reference manual* Doc. ID 13902 Rev 17
- [9] ST Microelectronics *STM32F407 datasheet* Doc ID 15274 Rev 5
- [10] Texas Instruments *DP83630 Precision PHYTER - IEEE® 1588 Precision Time Protocol Transceiver (Rev. B)* SNLS335B Rev 2
- [11] Dutoit T., Marqués F. *Applied signal processing* Springer Science+Business Media, 2009, ISBN 978-0-387-74534-3
- [12] C. H. Knapp, G. C. Carter. IEEE Transactions on Acoustics, Speech and Signal Processing *The generalized correlation method for estimation of time delay* VOL. ASSP-24, NO. 4, AUGUST 1976, s. 320 - 327
- [13] Carter G.C. IEEE Transactions on Acoustics, Speech and Signal Processing *Time Delay Estimation for Passive Sonar Signal Processing* VOL. ASSP-29, NO. 3, JUNE 1981, s. 463 - 470
- [14] Y. T. Chan, K. C. Ho. IEEE Transactions on Signal Processing *An simple and efficient estimator for hyperbolic position location* VOL. 42, NO. 8, AUGUST 1994, s. 1905 - 1915

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ARM	Advanced RISC Machine
CAN	Controller Area Network
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSV	Comma-Separated Values
DFSE	Dominant Frequency Selection Algorithm
DMA	Direct Memory Access
DPS	Deska plošných spojů
FTSP	Flooding Time Synchronization Protocol
GCC	Generalized Cross Correlation
GPIO	General Purpose Input/Output
GPS	Global Positioning System
HTTP	Hypertext Transport Protocol
IEEE-1588	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protocol
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group
LED	Light-Emitting Diode
MAC	Media Access Control
MCU	Mikrokontroler
MII	Media Independent Interface
NTP	Network Time Protocol
PC	Personal Computer

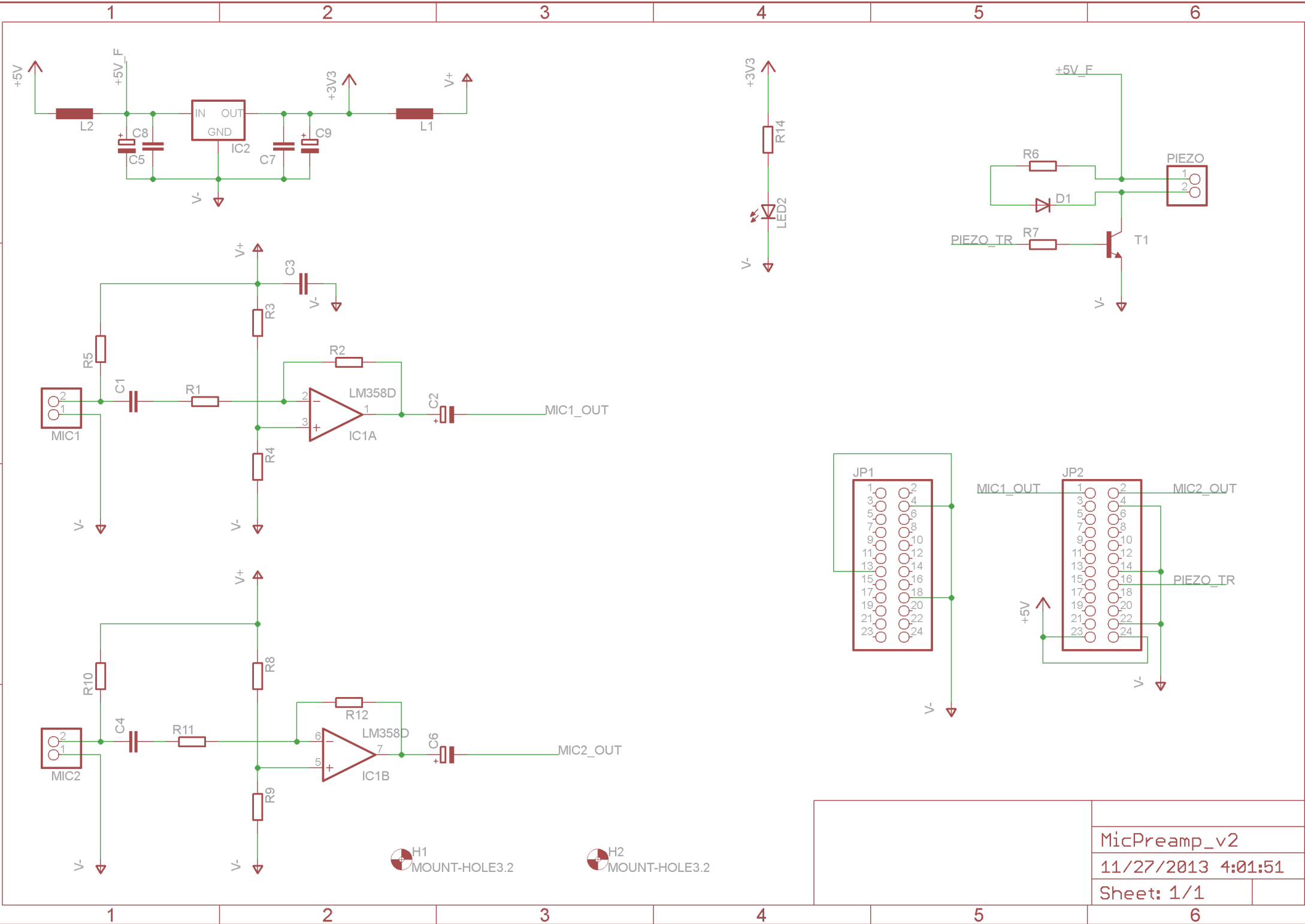
PCI	Peripheral Component Interconnect
PHY	Ethernet physical transceiver
PoE	Power over Ethernet
PTP	Precision Time Protocol
PXI	PCI Extensions for Instrumentation
RBTS	Reference Broadcast Time Synchronization
RISC	Reduced Instruction Set Computer
RMII	Reduced Media Independent Interface
ROM	Read Only Memory
RTC	Real-Time Clock
RTOS	Real-Time Operating System
SCPI	Standard Command for Programmable Instruments
SRAM	Static Random Access Memory
STM32	32-bitový mikrokontroler s jádrem ARM cortex-M3 od firmy ST Microelectronics
SWD	Serial Wire Debug
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USART	Universal synchronous/asynchronous receiver/transmitter
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair

SEZNAM PŘÍLOH

A	Výrobní podklady	61
A.1	Schéma zapojení	61
A.2	Motiv DPS	63
A.3	Rozmístění součástek na DPS	65
A.4	Seznam součástek	65
B	Použitá názvosloví	67

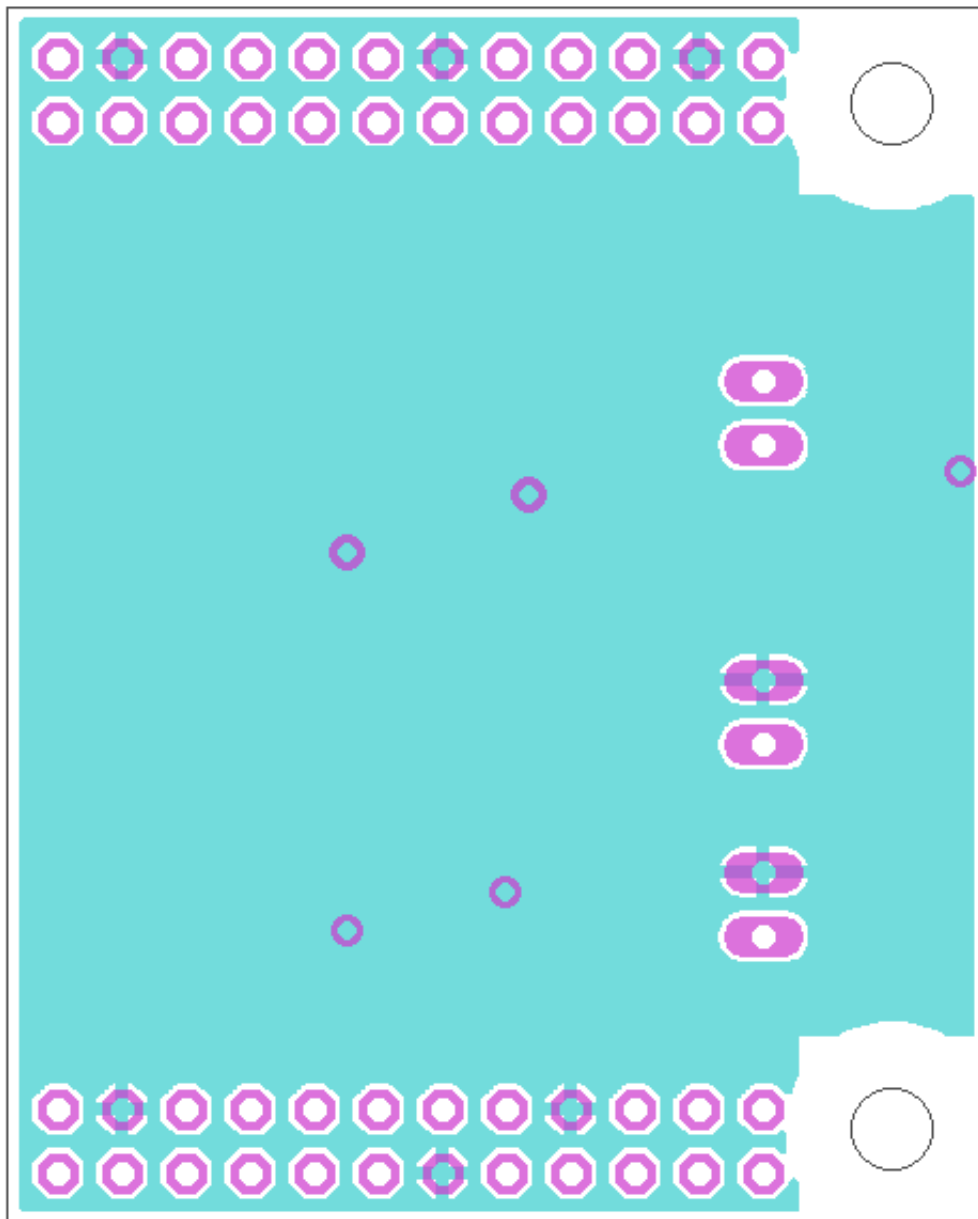
A VÝROBNÍ PODKLADY

A.1 Schéma zapojení

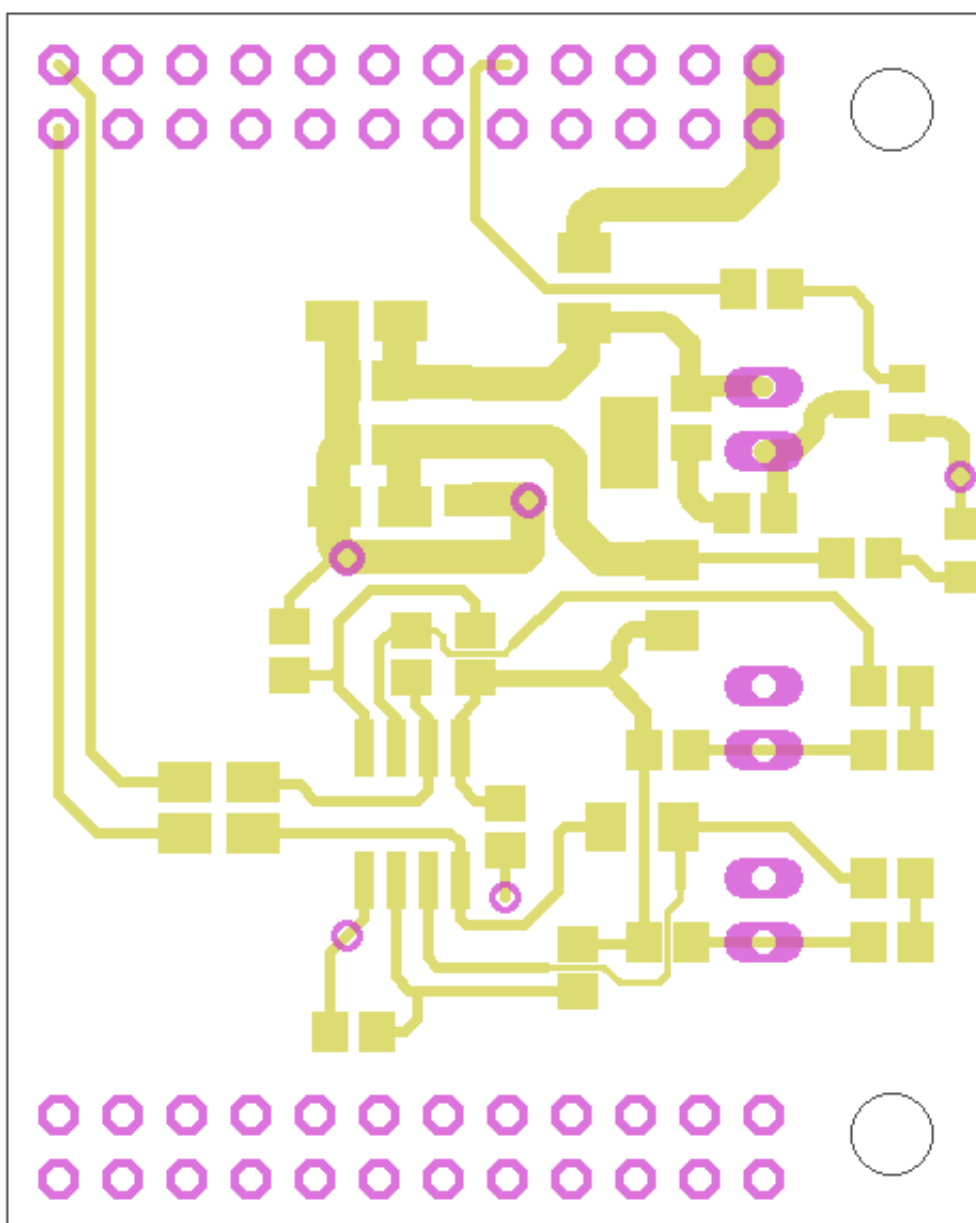


MicPreamp_v2	
11/27/2013 4:01:51	
Sheet: 1/1	

A.2 Motiv DPS

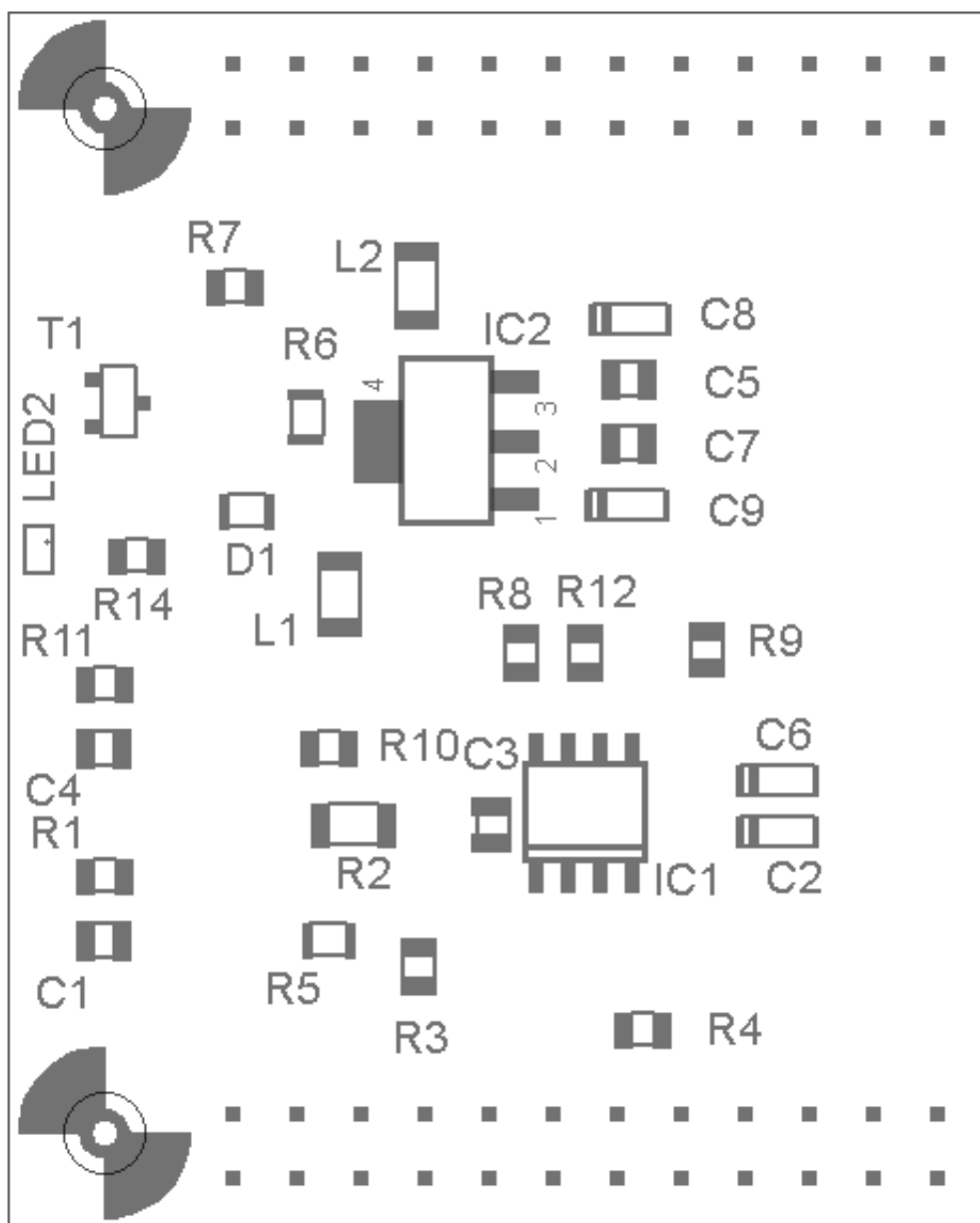


Obr. A.1: Vrstva TOP, měřítko 1:1



Obr. A.2: Vrstva BOT, měřítko 1:1

A.3 Rozmístění součástek na DPS



Obr. A.3: Rozmístění součástek ve vrsně BOT, měřítko 1.2:1

A.4 Seznam součástek

název	hodnota	pouzdro
R1	2,2k Ω	0805
R2	2,2M Ω	0805
R3	10k Ω	0805
R4	10k Ω	0805
R5	2,2k Ω	0805
R6	100 Ω	0805
R7	10k Ω	0805
R8	10k Ω	0805
R9	10k Ω	0805
R10	2,2k Ω	0805
R11	2,2k Ω	0805
R12	2,2M Ω	0805
R13	330 Ω	0805
R14	330 Ω	0805
C1	100nF/25V	0805
C2	4,7 μ F/25V	0805
C3	100nF/25V	0805
C4	100nF/25V	0805
C5	10 μ F/25V	0805
C6	4,7 μ F/25V	0805
C7	100nF/25V	0805
C8	100nF/25V	0805
C9	10 μ F/25V	0805
L1	68nH/Imax =500mA (R=0,38 Ω)	1206
L2	68nH/Imax =500mA (R=0,38 Ω)	1206
D1	1N4148	DO-213AA
LED1	zelená	0805
T1	BC847	SOT-23
IC1	LM358D	SO-08
IC2	SPX1117-3.3	SOT-223
JP1	pinová lišta 24 pinů, dvouřadá	RM = 2,54mm
JP2	pinová lišta 24 pinů, dvouřadá	RM = 2,54mm
MIC1	dutinková lišta 2 piny, dvouřadá	RM = 2,54mm
MIC2	dutinková lišta 2 piny, dvouřadá	RM = 2,54mm

B POUŽITÉ NÁZVOSLOVÍ

Teorie časové synchronizace s sebou nese značné množství technických pojmů a termínů. Tato příloha poskytuje definici několika základních pojmů, čtenáři by to mělo usnadnit porozumění a pochopení textu této práce.

Pojmem *master* budu dále v textu označovat měřící modul, Personal Computer (PC) nebo jakékoliv jiné zařízení, které v dané aplikaci funguje jako časová reference, podle které ostatní zařízení v síti synchronizují svůj vlastní lokální čas.

Označení *slave* bude použito pro měřící modul, PC nebo jakékoliv jiné zařízení, které synchronizuje svůj vlastní lokální čas podle referenčního *master* zařízení.

Pojem *jitter* se v elektronice používá pro označení nežádoucí odchylky jedné nebo více charakteristik zdroje hodinového taktu od výrobcem udávané hodnoty.

V elektronice se termínem (frekvenční) *drift* označuje nežádoucí jev, při kterém dochází k posunu frekvence zdroje periodického signálu od jeho nominální hodnoty.