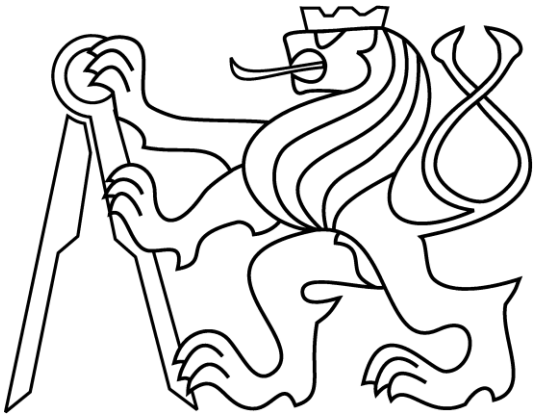
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra elektrických pohonů a trakce



Diplomová práce

Návrh vnitřní komunikace měniče prostřednictvím sériového komunikačního kanálu

Bc. Jiří Petrák

Vedoucí práce: Ing. Miroslav Lev

Studijní program: Energetika, elektronika a management
Obor: Elektrické stroje, přístroje a pohony
2015

Zadání

Poděkování

Rád bych poděkoval panu Ing. Miroslavu Lvovi. za odborné vedení, rady a poskytnuté prostředky pro vytvoření této diplomové práce. Dále bych chtěl poděkovat svým rodičům, přítelkyni a kamarádům za trpělivost a podporu.

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.

V Praze dne 11. 5. 2015

.....

Podpis

Abstrakt

Tato diplomová práce pojednává nejprve o aplikaci standardu Ethernet v průmyslu a přibližuje jeho nejrozšířenější průmyslové standardy a jejich limity. V dalších částech je popsán návrh a realizace vnitřní komunikace měniče IGBT založeném na protokolu Ethernet.

V softwaru Eagle byl vypracován návrh DPS s označením BRD00001, která obsahuje rozhraní pro připojení k měniči IGBT. Na této DPS je umístěn 8bitový mikroprocesor PIC18F97J60 od firmy Microchip zajišťující Ethernetovou konektivitu, řízení měniče, detekci chyb a měření napětí a proudů.

V poslední části jsou vyhodnoceny výkonové vlastnosti a omezení tohoto řešení.

Abstract

This diploma thesis examines firstly the application of the Ethernet standard in the industry and brings its widespread industry standards and their limits. In the next sections the design and implementation of internal communication in IGBT converter based on Ethernet protocol is described.

PCB design marked BRD00001 has been worked out in the software Eagle, which includes an interface for connection to the IGBT converter. On the PCB is located PIC18F97J60 8-bit microcontroller from Microchip providing Ethernet connectivity, inverter control, fault detection and measurement of voltages and currents.

The performance characteristics and limitations of the solution are evaluated in the last part.

Klíčová slova

IGBT měnič, průmyslový Ethernet, Microchip, PIC18F97J60, Ethernetové rozhraní, návrh, realizace, výrobní dokumentace

Key words

IGBT converter, industrial Ethernet, Microchip, PIC18F97J60, Ethernet interface, design, realization, production documentation

Citace

PETRÁK, Jiří. *Návrh vnitřní komunikace měniče prostřednictvím sériového komunikačního kanálu*. Praha, 2015. Diplomová práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra pohonů a trakce

x

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V posledních deseti letech došlo k velkému rozsahu Ethernetu v průmyslu zejména kvůli velké potřebě decentralizace řídicích systémů a zařízení. V současnosti roste podíl průmyslových protokolů dvojnásobným tempem oproti tradičním sběrnicovým systémům. Mnoho firem nabízí ve svém portfoliu zařízení podporující některý z průmyslových Ethernetových standardů a některé z nich dokonce zcela přešla z tradičních protokolů skupiny fieldbus na tyto protokoly. Vzhledem k dynamice tohoto prostředí a mnohdy i nedůvěře k průmyslovému Ethernetu, jsem se rozhodl vybrat toto téma a vytvořit Ethernetové rozhraní pro IGBT měnič.

Na začátku této práce je vysvětlen Ethernet a jeho základní principy, neboť na něm jsou z větší či menší části postaveny všechny verze protokolů průmyslového Ethernetu. Následují části zabývající se topologiemi v běžném i průmyslovém prostředí, kabely, koncovými prvky, bezpečností funkční i informační.

V další části popisuji návrh samotného Ethernetového rozhraní pro řízení měniče, který obsahuje 8bitový mikroprocesor PIC18F97J60 od firmy Microchip. Tento mikroprocesor jsem zvolil z toho důvodu, že obsahuje Ethernetový modul s MAC a PHY což zjednodušuje jeho použití a také kvůli již zkušenostem nabytých s 8bitovými mikroprocesory of firmy Microchip během studia oboru „Elektrické stroje, přístroje a pohony “. DPS dále obsahuje různé obvody pro vysílání spínacích pulzů, detekci chyb, měření, atd.

Pro implementaci TCP/IP byl využit volně použitelný stack od firmy Microchip, který byl upraven pro potřeby této diplomové práce. Byla vytvořena funkce v programovém vybavení MCU (Microcontroller Unit) pro komunikaci s řídícím počítačem skrze protokol UDP sloužící pro otestování výkonu. Řídící počítač byl vybaven aplikací zasílající řídící data a také schopná příjmu dat. Toto programové vybavení na straně MCU i počítače tvoří kompletní řešení Master-Slave pro komunikace skrze Ethernet.

V poslední části jsem se zaměřil na vyhodnocení výkonu PIC18F97J60 a jeho reálného použití pro řízení měničů, automatů PLC, či dalších zařízení.

2 Ethernet

Je to už více jak 40 let co byl vyvinut jeden z nejrozšířenějších komunikačních standardů vůbec - Ethernet. Na svou současnou pozici se dostal nejen díky svým kvalitám, ale také příhodnému načasování na začátku revoluce osobních počítačů a místních sítí (LAN). V současnosti je Ethernet aplikován nejen v místních sítích, ale i v průmyslových aplikacích jako jsou například chytré sítě či internet věcí. Ethernet je však také aplikován např. v medicíně.

2.1 Historie Ethernetu

Zrození Ethernetu je připisováno firmě Xerox (IEEE, [b. r.]), konkrétně Robertu Metcalfemu a Davidu R. Boggsovi. Pan R. Metcalfem působil v Xerox's Palo Alto Research Center (PARC), kde pracoval s první sítí přepojující pakety, a dostala se k němu kniha pojednávající o konferenci z roku 1970 *American Federation of Information Processing Societies* (AFIPS), v které jej zaujala bezdrátová síť známá jako ALOHAnet. Inspirován touto sítí a technologickými aspekty, se kterými se nemohl vůbec ztotožnit, vytvořil v roce 1973 se svým kolegou panem D. R. Boggsem první funkční prototyp Ethernetu, propojující více jak 100 počítačů na vzdálenost 1 km při rychlosti 2,94Mbps.

Posléze pracovalo na dalším vývoji Ethernetu konsorcium firem Digital Equipment Corporation (DEC), Intel a Xerox, které se nazývalo DIX. V roce 1983 byl Ethernet standardizován jako standard IEEE 802.3. Ethernet se však nepřestal vyvíjet a standardizací procházely taktéž další a další verze Ethernetu, jež byly zaměřeny na vyšší přenosové rychlosti, jako je například 10GBASE–T.

Rok	Událost
1970	první experimentální systém Ethernet (Bob Metcalfe a David Boggs v Xerox Palo Alto Research Center)
1979	začátek standardizace Ethernetu ve spolupráci firem DEC, Intel a Xerox
1980	standard Ethernetu V1.0 (známý jako DIX podle počátečních písmen firem – původců)
1983	standard IEEE 802.3 (metoda přístupu CSMA/CD)
1990	standard IEEE 802.3i 10Base–T: 10Mb/s Ethernet na nestíněném krouceném páru vodičů (UTP), možnost topologie typu hvězda
1995	T standard IEEE 802.3u 100Base: 100Mb/s Ethernet (známý jako Fast Ethernet)
1997	standard IEEE 802.3x Full–Duplex Ethernet
1998	standard IEEE 802.3z 1000Base–X (gigabitový Ethernet)
2002	standard IEEE 802.3ae 10GBase–T (desetigigabitový Ethernet)

Tabulka 1 – Historie standardizace Ethernetu (Zezulka, Hynčica, 2007)

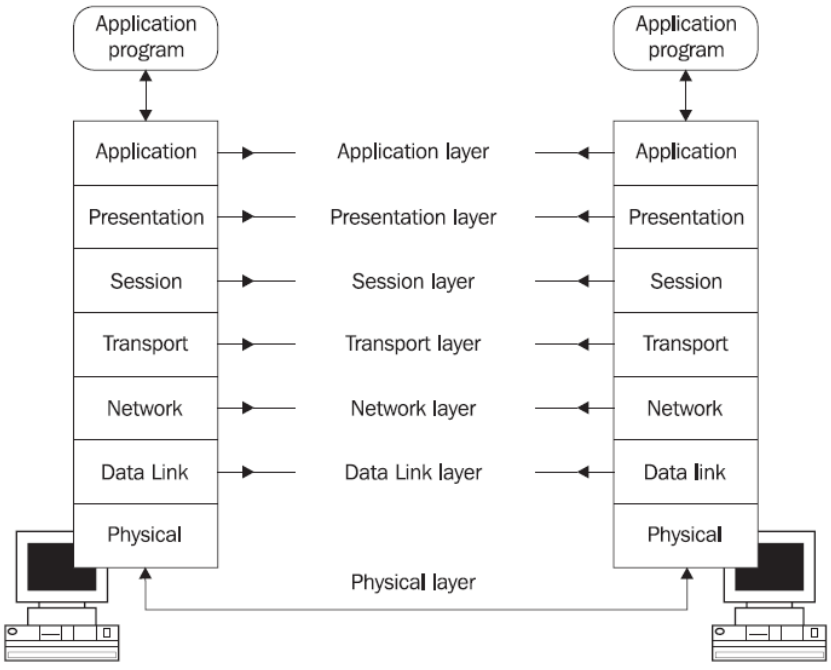
2.2 Ethernet v průmyslu

Pokusy o využití Ethernetu v průmyslu jsou téměř stejně staré jako Ethernet samotný. Na počátku existence Ethernetu bylo fyzické propojení provedeno koaxiálními kabely, které způsobovaly značnou robustnost tohoto provedení, a pro speciální aplikace bylo použito optické vlákno. V této době, v níž se investovaly značné prostředky do rodiny průmyslových sběrnic fieldbus (Zezulka, Hynčica, 2007), kde probíhala doslova patentová válka, nebyl důvod prosazovat komunikační protokol, jenž nebyl původně navržen pro průmyslové aplikace. Mezi výhody sběrnic fieldbus patří skutečnost, že tyto protokoly byly navrženy jako systém sběrnicové topologie. Kdežto Ethernet byl navržen jako hvězdicové zapojení, což umožňovalo používání systému, na nějž byli technici zvyklí, a také to znamenalo nižší náklady na infrastrukturu.

Ke změně situace došlo na začátku 90. let. V této době došlo k masovému rozšíření kancelářské technicky, kde se začal používat 8 žilový kabel dle standardu 10BASE–T, jenž nahradil koaxiální kabel. V pořadí další, standard pro telefonní přípojku konektor RJ45, na sebe nenechal dlouho čekat. Nicméně všechno toto bylo standardizováno pouze pro strukturovanou kabeláž v obytných a kancelářských budovách ISO/IEC 11801 (evropská norma EN50173).

2.3 OSI model

Následující část pojednávající o OSI modelu a jeho částí byla vypracována na základě Osterloh (2005), Pužmanové (2004) a Zezulky a Hynčicy (2007).



Obrázek 1 – ISO model (Rozdíly.cz, [b.r.])

OSI model je referenční model (RM) vytvořený organizací ISO v 80. letech minulého století, který byl přijat normou ISO7498.

Na počátku budování komunikačních sítí, byly používány nejrůznější protokoly a prostředí tudíž bylo silně heterogenní. Následkem tohoto bylo, že zařízení různých firem spolu nekomunikovala, a proto bylo nutné vytvořit otevřený model, který by umožňoval komunikaci napříč firemními řešeními.

Úkolem RM bylo vytvořit základ pro vytvoření norem, které by umožnily funkční propojení otevřených systémů. Není definováno, jakým způsobem, anebo jakou implementací tohoto má být dosaženo. Je pouze stanoveno, že musí být splněna daná norma. Pokud zařízení je v souladu s OSI, tak se jedná o reálný otevřený systém.

RM OSI se skládá ze 7 vrstev, jak je zřejmé z obrázku 1. Každá vrstva má 2 základní funkce. Vrstvy se skládají již z dále nedělitelných celků neboli entit. Každá z vrstev jednak poskytuje své služby sousední vyšší vrstvě a za druhé se řídí souborem pravidel platnými pro její vrstvu, jako je zahájení přenosu, jeho provedení a ukončení. Její dosah je omezen vždy na jednu sousední vrstvu. Mezi stejnými vrstvami dvou systému jsou virtuální spoje, ale k samotnému přenosu dochází pouze ve fyzické vrstvě, jež tvoří spoj mezi dvěma systémy. Toto virtuální spojení znamená, že účastnící spolu komunikují na nevyšší aplikační vrstvě, a nemají zdání o funkcích vrstev 1–6.

Lze si to představit tak, že ředitel firmy A zasílá dopis řediteli firmy B. Ředitel firmy A dopis napíše a předá ho sekretářce, tedy nižší vrstvě, aby jej tato dala do obálky a napsala na obálku adresu. Sekretářka předá dopis k odeslání, kde je na něj nalepena známka a tento je předán poštovnímu doručovateli, jenž provede fyzické doručení. Ve firmě B pak dojde k vyzvednutí zásilky, sekretářka dopis rozbalí a předá jej řediteli firmy B.

Ethernet jako takový pak vychází právě z referenčního modelu OSI a to tak, že první dvě vrstvy RM OSI představují onu metodu Ethernet, 5. a 6. vrstva jsou vynechány a vyjádřeny vrstvou 7 s aplikačními protokoly. Tento celek s vyjádřenými protokoly na 3. vrstvě (IP) a na 4. vrstvě (TCP) tvoří rozšířenou alternativu k RM modelu OSI v místních sítích. Uvedené tři protokoly se souhrnně označují jako *Ethernet Protokol Suite* (Ethernet TCP/IP). Ethernet byl standardizován organizací IEEE jako standard 802.3. Jednotlivé vrstvy Ethernetu jsou popsány níže.

2.3.1 Fyzická vrstva

Je to nejnižší vrstva umožňující fyzickou komunikaci mezi systémy. Mezi její funkce patří aktivování, udržování a deaktivování fyzického spojení, vytváření datových jednotek, přenos bitů, jejich řazení, identifikování fyzického spojení (jednobodové/mnohobodové, half duplex/full duplex, mechanické a elektrické vlastnosti), oznamování poruchových stavů a reportování kvality spojení vyšší vrstvě.

2.3.2 Spojová vrstva

Tato vrstva zajišťuje dynamické spojení s vlastním přenosovým médiem. Její základní funkcí je zahajování, udržování a závěr datových spojení, identifikace koncových bodů, řazení přijatých rámců, synchronizace, oznamování chyb a jejich oprava. Dále počítá kontrolní součet CRC (Cyclic redundancy Cheb) nebo FCS (Frame check sequence) a pokud se hodnota shoduje s hodnotou uvedenou v rámci, tak je předána další vrstvě. Pokud ne, tak je rámec zahozen bez informování vysílače o tom, že nebyl doručen. Příjemce nikdy nepředává poškozený rámec vyšší vrstvě.

Definice pojmů

- Paket – datový blok předávaný na transportní vrstvě; skládá se z hlavičky a uživatelských dat
- Rámec – datový blok posílaný na úrovni spojové vrstvy
- Datagram – do jisté míry se jedná o synonymum k paketu, ale na rozdíl od paketu, což je obecný název pro datový blok, jsou datagramy vždy zasílány nezabezpečeným spojením – tedy službami nezajišťující navazování komunikace a potvrzování příjmu (např. UDP)
- Zpráva – datový blok přenášený na úrovni aplikační vrstvy

Spojovou vrstvu můžeme rozdělit na podvrstvu řízení logického spoje (LLC) a na podvrstvu řízení přístupu k přenosovému prostředku (MAC). První zmíněná vrstva sousedí se síťovou vrstvou a poskytuje jí rozhraní mezi konkrétním přenosovým prostředkem a vyššími vrstvami. Podvrstva MAC sousedí přímo s fyzickou vrstvou a obsahuje adresu MAC (fyzická adresa), která je unikátní a ve standardu IPv4 je tvořena 48 bity uváděnými v hexadecimálním tvaru. Prvních 24 bitů je tvořeno jednoznačným identifikátorem výrobce, jenž přiděluje IEEE a zbývajících 24 bitů je označení samotného fyzického rozhraní.

Konkrétně podvrstva MAC dále definuje metodu přístupu k médiu. Metoda přístupu k médiu je v případě Ethernetu založena na principu CSMA/CD, který říká, že každý připojený účastník má stejné právo na použití přenosového média, pokud v tom momentě na něm nevysílá jiný účastník – „best effort“. To tedy znamená, že jednotliví účastníci spolu soupeří o právo přenášet data. Když se pokusí 2 účastníci vysílat současně, dojde ke kolizi a rámce budou poškozeny. Jednotliví účastníci poslouchají nosnou frekvenci a v případě ticha, tzn. nulového signálu, mohou začít vysílat. CSMA/CD je metoda, která počítá s kolizemi, avšak aby účastník zjistil, že vůbec ke kolizi došlo, musí být maximální doba přenosu signálu od vysílací stanice až k té nejvzdálenější stanici kratší, než je doba vysílání nejkratšího možného rámce. Tato doba je definována jako RTD (Round trip delay).

Nejkratší rámec ve standardu Ethernetu s přenosovou rychlostí 10Mbps je 64 bajtů, tj. 512 bitů. Musí tedy být zajištěno, že první bit prvního bajtu nedorazí na nejvzdálenější stanici později, než bude odvysílán první bit 32. bajtu. Pokud to přepočítáme na čas, tak RTD je 52,1 µs a polovina tohoto času je 25,6 µs. Pokud vysílací stanice nedostane informaci

o tom, že nastala kolize, má za to, že rámec nebyl porušen. Na základě předchozího výpočtu se vypočítává maximální přípustná délka spoje tzv. kolizní doména. V případě, že je kolizní doména delší, nebude kolize rozpoznána vysílací stanicí, což bude mít za následek pokles výkonu sítě, neboť ztrátovost rámců budou muset řešit protokoly vyšších vrstev, a to bude vyžadovat další režii, což bude mít za následek snížení přenosové rychlosti. Jakmile vysílač ukončí vysílání, čeká tento minimálně 9,6 µs, než se pokusí o další vysílání. V tomto čase mají ostatní účastníci možnost přistoupit k médiu a odeslat rámce čekající na odeslání.

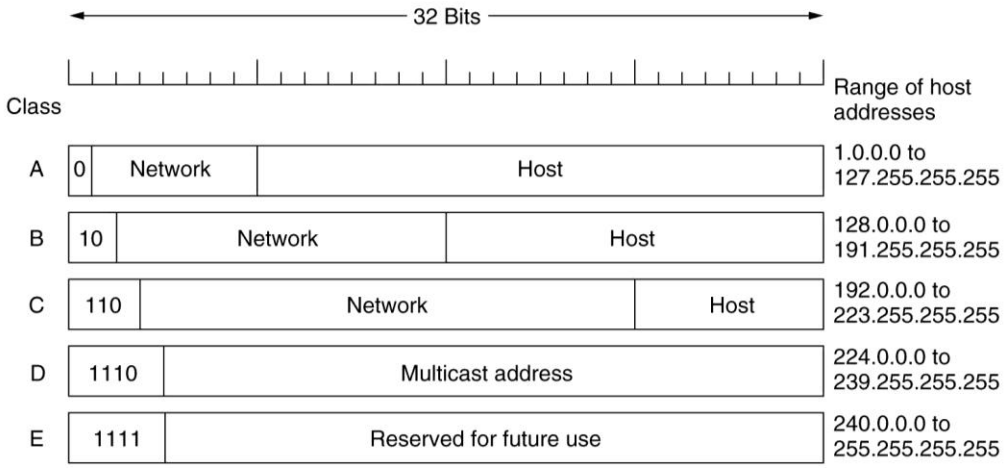
Během vysílání poslouchá i samotný vysílač, jenž detekuje případnou kolizi. Pokud k této dojde, vysílač začne vysílat jam signál, který informuje všechny ostatní zařízení o kolizi. Po skončení vysílání signálu jam, každý z vysílajících čeká náhodné množství času, na další odeslání rámce. Pozdní kolize, anebo jejich velké množství je nežádoucí. Pokud však bude na jeden segment sítě připojeno velké množství zařízení, budou velmi časté.

Kolizím se nelze vyhnout, pokud účastníci používají pouze základní přenosové frekvence. Avšak díky pokrokům v elektronice a používání více kroucených párů vodičů (oproti koaxiálnímu kabelu) došlo k výraznému nárůstu rychlostí. V současnosti nejvíce používaný standard 100BASE–T, využívá dva páry kroucených vodičů (jeden pár pro vysílání Tx a druhý pro příjem Rx) pro plný duplex. S příchodem přepínačů (switch) namísto rozbočovačů (hub) docílíme dalšího snížení pravděpodobnosti kolizí, neboť přepínače posílají rámce pouze do segmentu sítě, pro nějž je rámec určen.

Z principu metody CSMA/CD je zřejmé, že se tato metoda nehodí pro průmyslové aplikace s ohledem na skutečnost, že nedokážeme přesně definovat, kdy bude rámec odeslán. Výše uvedené je také často uváděno jako hlavní důvod kritiků pro použití Ethernetu v průmyslovém prostředí. Pro uplatnění Ethernetu v průmyslu je tedy nezbytné obejít metodu CSMA/CD pro časově kritické zprávy.

2.3.3 Síťová vrstva

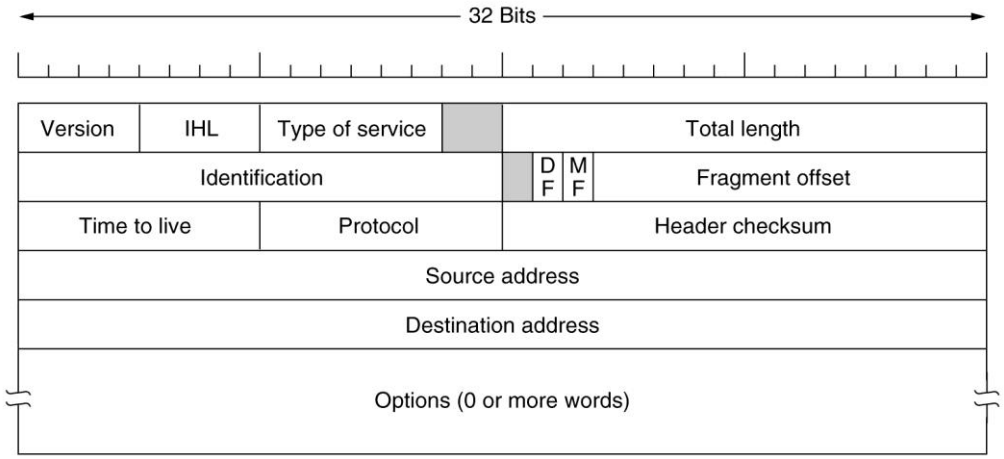
Tato vrstva je reprezentována protokolem IP, jenž je nejzatíženějším protokolem z celé skupiny protokolů TCP/IP. Tento protokol je zodpovědný za logické adresování v síti a přenášení datagramů v rámci jedné i více sítí. Tento protokol však provádí i další operace, mezi něž patří fragmentování datagramů, jejich zpětné skládání atd. Jedná se o nespojový protokol, tudíž nezaručuje doručení datagramu, potvrzování příjmu ani datový tok. Pokud dojde při doručování k jakémukoliv problému, pak se protokol IP spoléhá na protokol ICMP, jenž je považován za integrální součást protokolu IP. Protokol ICMP v případě výskytu chyby předá přesný popis chyby protokolům na vyšších vrstvách (TCP), které musejí zajistit spolehlivost komunikace.



Obrázek 2 – Třídy adres (Grygárek, [b.r.])

V současnosti se nejvíce používá protokol IPv4, který má velikost hlavičky o velikosti nejméně 20 bajtů, pokud neobsahuje navíc přídavné volby a adresovací prostor 4 bajty. IP Adresy jsou rozdělené do tříd, jež označujeme A - E, které charakterizují jejich použití.

Pro dosažení co nejlepší efektivity je adresovací prostor použit jak pro adresování sítě, tak pro adresování koncového zařízení.



Obrázek 3 – Hlavička IP (Grygárek, [b.r.])

Vzhledem k tomu, že v každé třídě je možný jiný počet koncových zařízení, je možné z toho odvodit, že třída A bude použita pro velmi rozsáhlé sítě, Třída B pak větší firmy či univerzity a třída C pro ostatní. Dále existují třídy D a E, které jsou vyhrazeny pro skupinové multicast vysílání. Rozsah adres třídy D je 224–239 a rozsah třídy E je 239–247, které zatím čekají na své uplatnění.

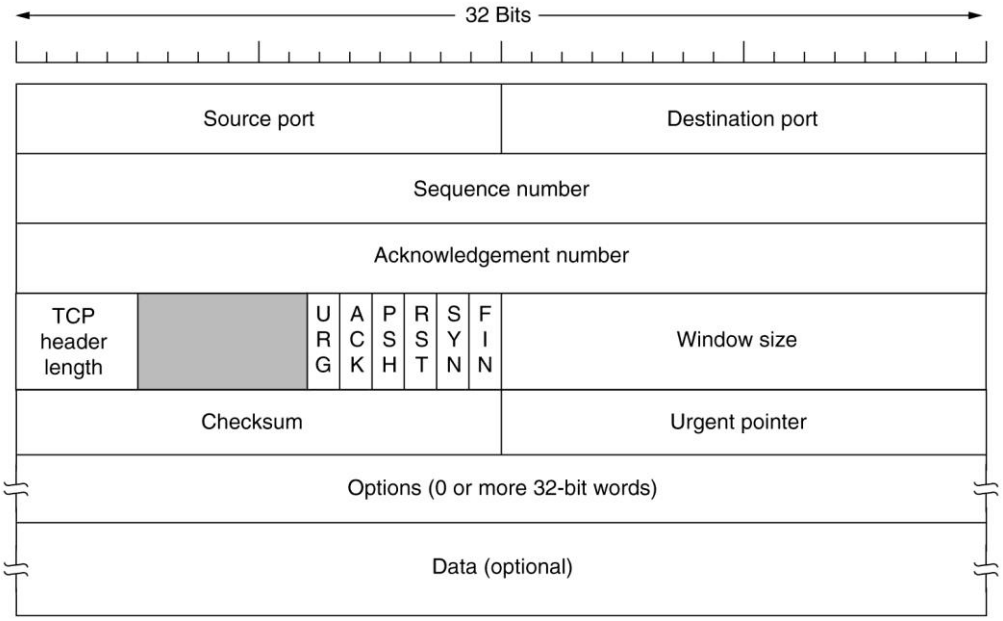
Dále existují některé vyhrazené adresy pro speciální úlohy například:

- 255.255.255.255 slouží pro vysílání do všech uzlů a sítí – broadcast;
- 0.0.0.0 – tedy adresa složená ze samých nulových bitů představuje odkaz na neznámy počítač v síti. To může být důležité například v situacích, kdy počítač

nezná svou adresu, anebo nezná adresu vzdáleného počítače, na niž má datagram doručit;

- 127.0.0.1 - tato adresa slouží pro testování zpětné smyčky. Při jejím použití se jedná o volání místního uzlu, při kterém není generován žádný síťový provoz;

Pro směrování paketů v síti je také nezbytná maska podsítě, jež určuje, které byty tvoří adresu sítě či podsítě. Na základě porovnání IP adresy cílového počítače s maskou podsítě vysílacího počítače může vysílací počítač určit, jestli může vyslat datagram přímo na IP adresu cílového počítače, anebo musí předat datagram na směrovači plnícím úlohu brány, protože se cílový počítač nachází v jiné síti nebo podsíti.



Obrázek 4 – Hlavička protokolu TCP (Grygárek, [b.r.])

IP adresa musí být v celé síti jedinečná, avšak v dnešní době 32 bitové adresování nestačí pokrýt potřeby milionů uživatelů připojených k internetu. Výše uvedené vedlo k zavedení protokolu IPv6, jenž má 128butové adresné pole a který pokryje adresovací potřeby na velice dlouho dobu. V protokolu IPv6 došlo také ke změně hlavičky, díky které mohou směrovače rychleji směřovat pakety. Nedílnou součástí protokolu IPv6 je také zabezpečovací mechanismus zajišťující ochranu proti nežádoucímu čtení paketu.

Rozmach protokolu IPv6 je brzděn nákladnou výměnou směrovačů a také tím, že lze omezení počtu IP adres obejít pomocí překladu síťových adres NAT. Směrovače s podporou NAT převádí vnitřní privátní adresy na registrované vnější adresy IP při pokusu počítače, umístěném ve vnitřní síti, komunikovat s počítačem umístěném v internetu. Toto přidělování je dynamické, a tak i pro obhospodaření většího počtu interních stanic stačí jen malého počtu fyzických připojení do vnější sítě.

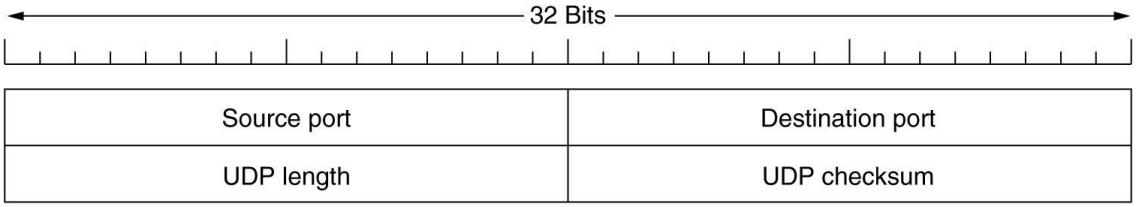
2.3.4 Transportní vrstva

Jedná se o vrstvu, která je chápána jako vrstva zajišťující spolehlivou komunikaci a využívá služeb síťové vrstvy. Zajištění spolehlivé komunikace platí v případě, že použijeme protokol TCP, který je zaveden v transportní vrstvě. Tento protokol je spojově orientovaný, zajišťuje bezchybný příjem dat díky potvrzování aktivního spojení. V případě, že vysílací stanice neobdrží potvrzení o doručení paketu, je paket odeslán znovu.

Každé navázání komunikace je navázáno trojnásobným potvrzením „handshake“, po kterém může vysílací stanice začít posílat data. Doručení každého paketu je potvrzeno zprávou odesílateli. V případě ztráty spojení je informována vyšší vrstva.

Na transportní vrstvě může být aktivních více spojení, jež jsou rozlišeny portem. Kombinace IP adresy a portu tvoří unikátní koncový bod. Tento se nazývá *socket*.

Protokol TCP není jediný, který můžeme použit na transportní vrstvě. Pro rychlejší doručování informací může být použit též protokol UDP. Protokol UDP nemá tak náročnou režii, neboť v jeho případě nedochází k navazování komunikace a pakety nejsou při přijetí potvrzovány. UDP je vhodné pro cyklický a rychlý přenos dat.



Obrázek 5 – Hlavička protokolu UDP (Grygárek, [b.r.])

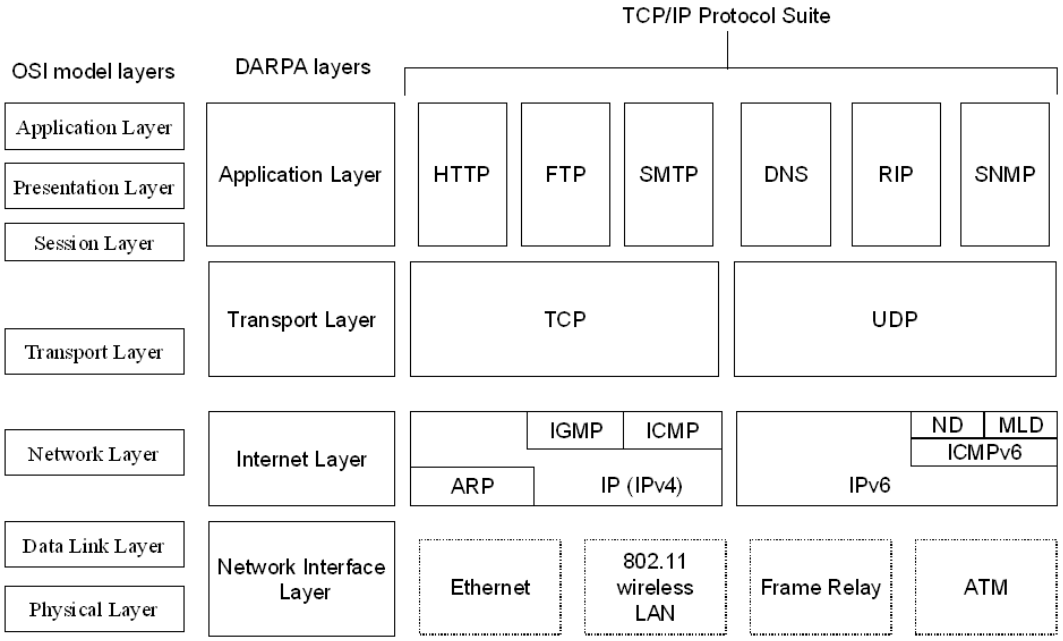
Definice pojmů

- Unicast – zpráva určená jedinému příjemci
- Multicast – zprávy určené většímu množství příjemců
- Broadcast – zprávy určené všem stanicím v síti

2.3.5 Aplikační vrstva

Tato vrstva přestavuje rozhraní mezi daty, odesílanými daty a daty připravených k odeslání. Protokoly skupiny TCP/IP jsou zodpovědné za veškerou agendu, co se týče navázání, udržení a ukončení komunikace mezi jednotlivými účastníky a protokoly aplikační vrstvy předávají data uživateli tak, aby jim rozuměl.

Na obrázku 6 je uveden úplný referenční model Ethernet TCP/IP včetně protokolů používaných v místních sítích i pro směrování paketů mimo ně a také porovnání s dalšími referenčními modely.



Obrázek 6 – Architektura TCP/IP Protocol Suite (Microsoft.com, ©2015)

Nejběžněji používané protokoly v aplikační vrstvě a jejich stručná charakteristika.

- Telnet (*Telecommunication Network*) – připojení se ke vzdálenému počítači
- Ping – ověření funkčnosti komunikace
- RPC (*Remote procedure call*) – vzdálené volání procedur
- FTP (*File Transfer Protocol*) – přenos souborů mezi počítači
- TFTP (*Trivial File Transfer Protocol*) – zjednodušený FTP
- SMTP (*Simple Mail Transfer Protocol*) – přenos zpráv elektronické pošty
- TIME – synchronizace vnitřních hodin
- DHCP (*Dynamic Host Configuration Protocol*) – automatická konfigurace síťových rozhraní
- BootP – nahrazen protokolem DHCP
- POP3 (*Post Office Protocol*) – stahování elektronické pošty
- DNS (*Domain Name System*) – převod doménových jmen na IP adresy
- NTP (*Network Time Protocol*) – synchronizace vnitřních hodin
- SNTP (*Simple Network Time Protocol*) – zjednodušený NTP

2.4 Fyzické provedení

Průmyslový Ethernet se odlišuje od Ethernetu používaném v kancelářském prostředí, nejenom v použitých protokolech avšak též v jejich implementaci a také HW komponentami a topologií sítě.

Rozdíly mezi průmyslovým prostředím a prostředím v kancelářích je více než zřejmé. V kancelářích není velké rozmezí teplot (max. 0–50 °C), vibrace či elektromagnetické záření. Pro automatizační techniku bylo nezbytné vyvinout aktivní síťové prvky, kabely a konektory,

které budou splňovat požadavky pro instalaci v široké škále prostředí. Například ve strojovnách, ale i na vrcholcích vysílacích věží, kde nezbytný stupeň krytí může být od IP20 až do IP67.

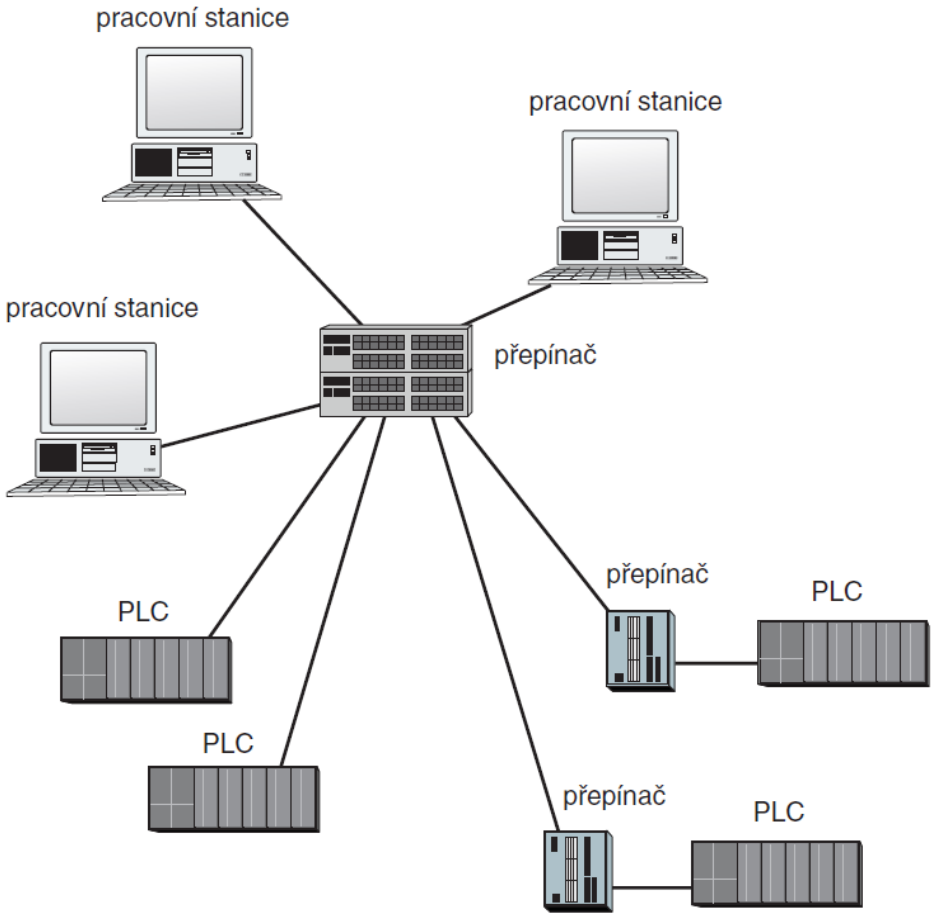
V průmyslovém prostředí je také kladen mnohem větší důraz na větší spolehlivost a co nejnižší přenosové časy a nejistoty (jitter), což vede ke změnám architektury sítí a kvalitnějším síťovým prvkům.

2.5 Topologie

V současnosti se můžeme setkat s nejrůznějšími topologiemi sítí. Těmi základními jsou hvězda, strom, sběrnice, kruh a síť se smyčkami. Jednotlivé topologie jsou blíže popsány níže.

2.5.1 Hvězda

Hvězda je nejrozšířenější topologií v kancelářských sítích, nicméně v případě poruchy centrálního prvku (switch) dojde k výpadku celé sítě. Další nevýhoda je větší množství



Obrázek 7 – Architektura hvězda (Zezulka a Hynčica, 2007)

kabeláže, které se podepíše na její ceně. Na druhou stranu je tato síť méně náchylná na poruchy kabeláže a související výpadky sítě. Mezi výhody patří také použití jednoduchých

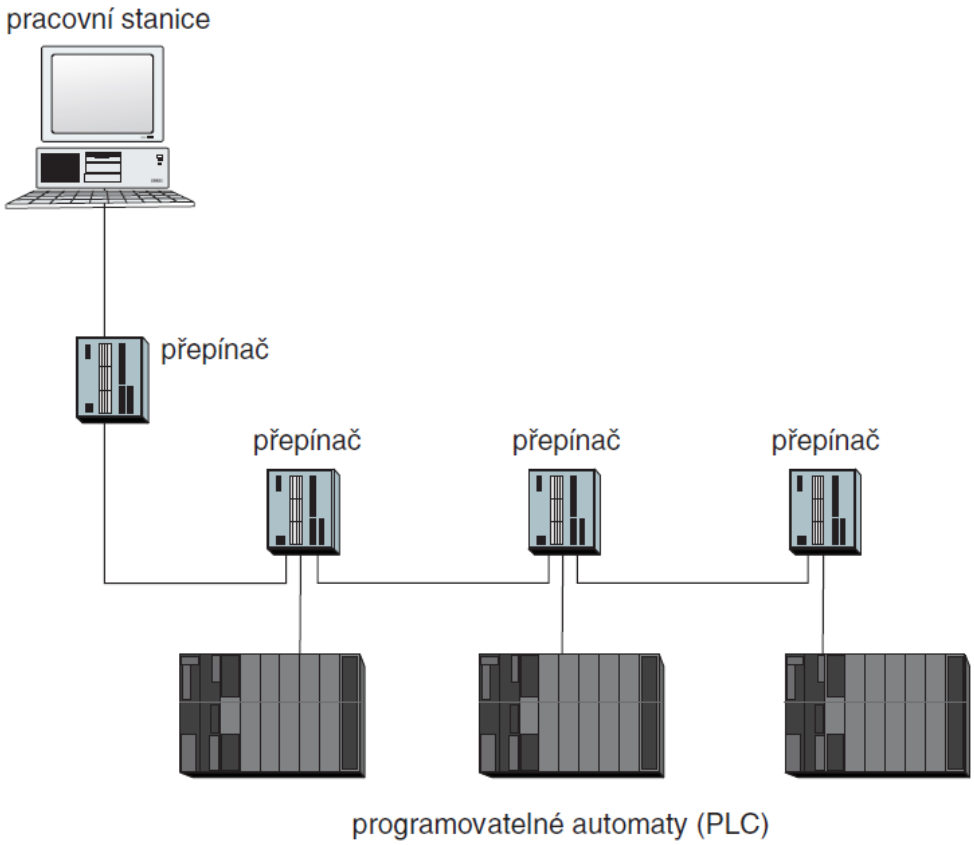
směrovacích protokolů. Důraz je v této typologii kladen na velký výkon a spolehlivost centrálního uzlu

2.5.2 Strom

Jedná se o propojení několika zapojení hvězda a z toho důvodu má tato topologie podobné vlastnosti.

2.5.3 Sběrnice

Tento model se příliš nepoužívá, neboť jakákoliv porucha na vedení má za následek nefunkčnost celé sítě. Pro tento model je typické jednodušší přidávání uzlů, ale také složitější řízení a větší riziko vzniku kolizí v případě připojení velkého množství stanic.

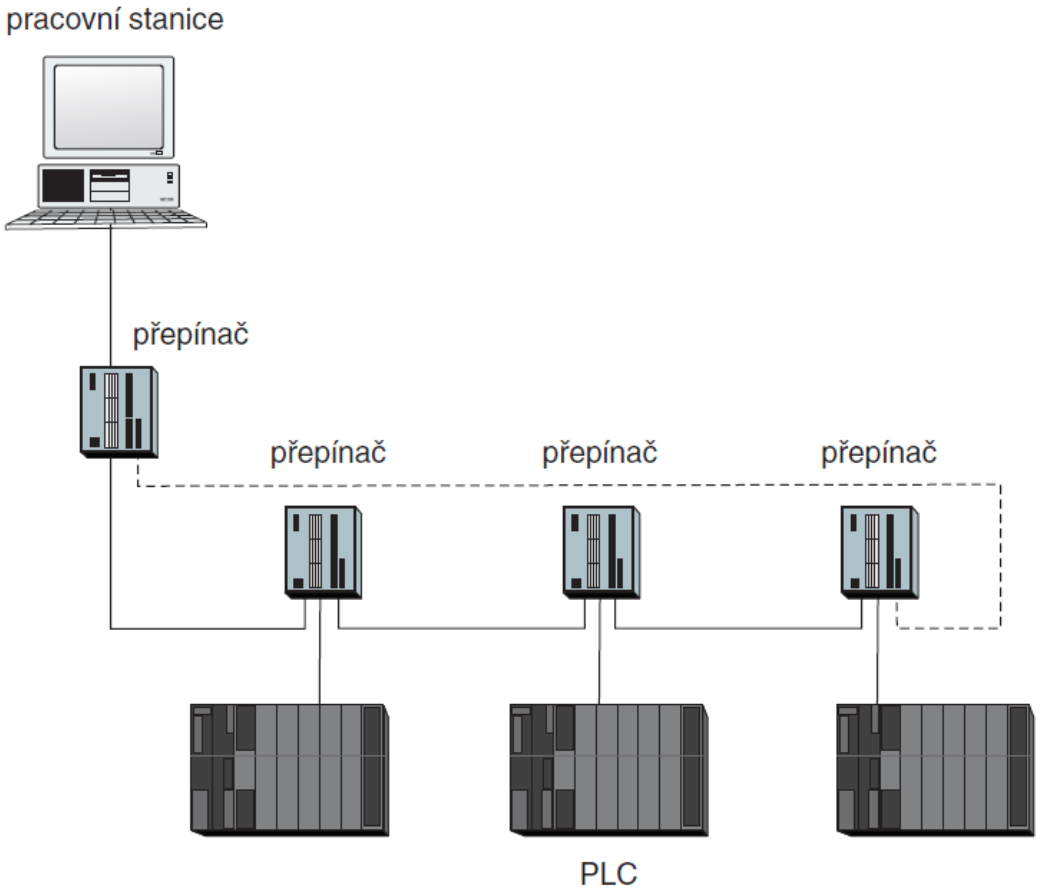


Obrázek 8 – Architektura sběrnice (Zezulka a Hynčica, 2007)

2.5.4 Kruh

Příklad topologie je uveden na obrázku 9, kde je čárkovaně vyznačena redundance spoje. Tento spoj je neaktivní do té doby, dokud nenastane závada na libovolném spojení mezi přepínači.

Tuto funkci zajišťuje protokol STP *Spanning tree* dle IEEE 802.1d (Bouška, ©2005–2015), (Cisco.com, 2006), který si nejdříve pomocí rozesílání paketu prohledá celou síť a zjistí všechny trasy mezi přepínači. Ty nejkratší nechá aktivní, redundantní spoje zablokuje tak, že přijímají pouze zprávy BPDU (*bridge protokol data units*), které oznamují změnu v topologii sítě. Mezi základní vlastnosti STP patří to, že maximální doba konvergence (čas, než port projde ze stavu blocking do forwarding) je 50s. V praxi se tato doba pohybuje kolem 30s, ale i přes to je tato doba příliš dlouhá pro průmyslové sítě.



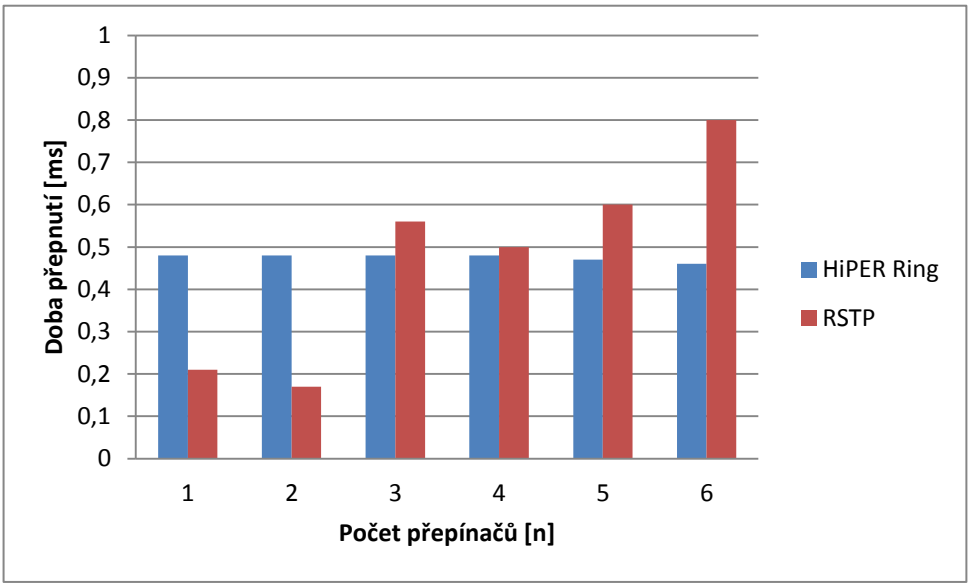
Obrázek 9 – Architektura kruh (Zezulka a Hynčica, 2007)

Pro urychlení rekonfigurace lze využít RSTP Rapid *Spanning tree* dle IEEE 802.1w (Bouška, ©2005–2015), který disponuje dobou konvergence max 1s, ale je omezen co do počtu připojených přepínačů a také se zde vyskytuje větší riziko vytvoření dočasných smyček, zdvojení paketů, atd.

Dalším zajímavým protokolem vycházejícím z RSTP je MSTP, který se aktuálně nachází v normě IEEE 802.1q. Tento protokol podporuje více sítí (VLAN – Virtual Local Area Network) a díky tomu můžeme seskupit VLANy do Spanning Tree instancí, čímž dojde k využití více cest a rozdělování zátěže. Vzhledem k tomu, že na jednom přepínači pak běží více Spanning tree instancí, jsou na tento kladeny vyšší HW nároky.

U firmy Cisco, která je přední výrobce síťové infrastruktury, se můžeme setkat se zajímavou funkcí nazývající se Flex Link. Tato funkce nefunguje na základě STP, ale pouze definuje určitou linku jako záložní. V případě výpadku dojde k přepnutí na záložní linku do 50ms.

Někteří výrobci vytvářejí vlastní proprietární řešení. Příkladem může být firma Hirschmann, přední výrobce komponent pro průmyslový Ethernet. Na obrázku 10 je graficky vyjádřeno porovnání rychlostí mezi jejich redundantní sítí HiPER Ring a RTSP.



Obrázek 10 – Porovnání řešení HiPER Ring (Hirschmann) a RTSP (Belden.com, [b.r.])

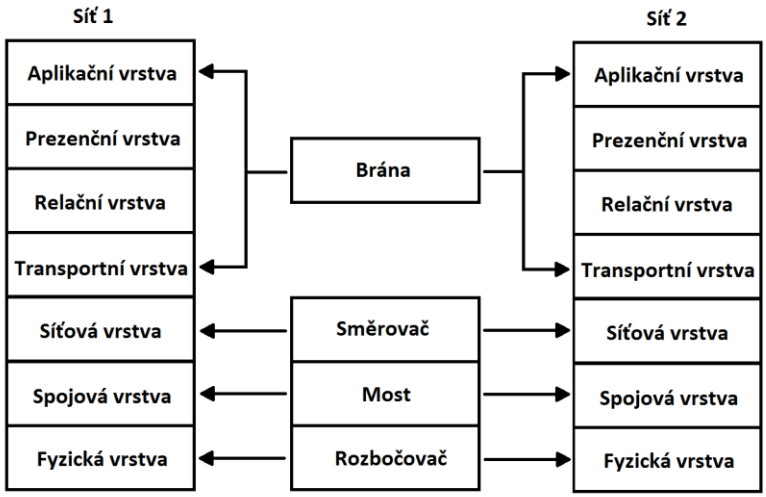
2.5.5 Síť se smyčkami

Jedná se o síť nabízející více možných spojů mezi uzly. V případě plného propojení tzn. každý switch s každým, se tak jedná o síť *full mesh. P*okud je však propojení pouze částečné, nazýváme ji jako *partial mesh.*

V této síti může být velké množství redundantních spojů, a v takovém případě to bude mít za následek zvýšení času pro změnu topologie v případě, že dojde k výpadku spojení na některém z aktivních spojů.

2.6 Aktivní síťové prvky

Pro propojení jednotlivých koncových zařízení výše uvedenými topologiemi je zapotřebí aktivních prvků. Těchto existuje více typů a liší se vlastnostmi a funkcemi v síti.



Obrázek 11 –Síťové prvky a jím odpovídající vrstvy

2.6.1 Rozbočovač (Hub)

U lokálních sítí se můžeme také setkat s názvem „opakovače“. V sítích Token Ring jsou to koncentrátory a u Gigabitového internetu se nazývají jako distributory. Funkce rozbočovače tkví v okamžitém odeslání jakéhokoliv příchozího signálu na všechny výstupy. Dnes už se už většinou setkáme s aktivními rozbočovači zesilující odesílaný signál, ale k jakýmkoliv jiným úpravám, jako je například kontrola příchozích rámců, jejich třídění, odstraňování poškozených rámců aj., nedochází.

To sebou přináší výhody, a to v podobě minimálního zpoždění přenosu zprávy přes rozbočovač, ale také nevýhody v podobě navýšení koncových zařízení v jedné kolizní doméně, tudíž vždy může vysílat pouze jeden účastník

Rozbočovač je prvek pracující na fyzické vrstvě a dnes se primárně používá k diagnostice sítě, kdy na určité spojení připojíme rozbočovač a na dalším portu můžeme zachytávat veškerou síťovou komunikaci.

2.6.2 Most

Most je prvek pracující na spojové vrstvě směrující rámce na základě tabulky MAC adres, která je tvořena záznamy posledního úspěšného doručení. Most dokáže pracovat pouze v režimu „store and forward“, to znamená, že nejprve musí celý rámec nejprve přijmout a teprve na základě cílové adresy MAC dojde k přepnutí portů, což také umožňuje přepočítání kontrolní součtu a filtrování poškozených rámců. Záznamy v tabulce pro

doručování jsou vytvářeny dynamicky, tzn., že přicházející rámce s cílovou MAC adresou, kterou most nezná, most odešle na všechny porty. Jakmile dostane odpověď od cílové stanice, že zpráva byla úspěšně doručena, je vytvořen záznam do tabulky s omezenou životností a pokud není zaslán další rámec pro tuto stanici v určeném čase, je záznam smazán. Výhody tohoto směrování tkví v tom, že rámce jsou vždy zaslány pouze do toho segmentu sítě, kde se nachází cílová stanice.

Výhodou mostu pak je schopnost rozdělit síť do menších kolizních domén a také dobře funguje jako přechod mezi sítěmi různých rychlostí. Naopak nevýhodou mostu je, že tento podporuje pouze jednu doménu pro rámce na všeobecnou adresu.

2.6.3 Přepínač

V dnešní době přepínače prakticky vytlačili mosty, které se dnes používají spíše ve speciálních aplikacích. Avšak s funkcí „most“ se můžeme běžně setkat na směrovačích.

Přepínače mají mnoho společných vlastností s mosty, jako je propojování lokálních sítí, učení adres MAC a také to, že nemění rámce. Navíc ale mohou segmentovat lokální sítě a podporují virtuální lokální sítě, což umožňuje vytvoření více domén všeobecného vysílání. Rychlost přeposlání rámců je u přepínačů také vyšší, neboť zpracování rámců probíhá v HW části a ne v SW, a také můžeme zvolit průběžné zpracování rámců (cut–throught). Průběžné zpracování rámců funguje metodou, kdy ihned po analyzování cílové MAC se začne rámec odesílat na příslušný port a tím můžeme zkrátit zpoždění, nicméně nemůžeme filtrovat poškozené rámce, neboť nedochází ke kontrole.

2.6.4 Brány

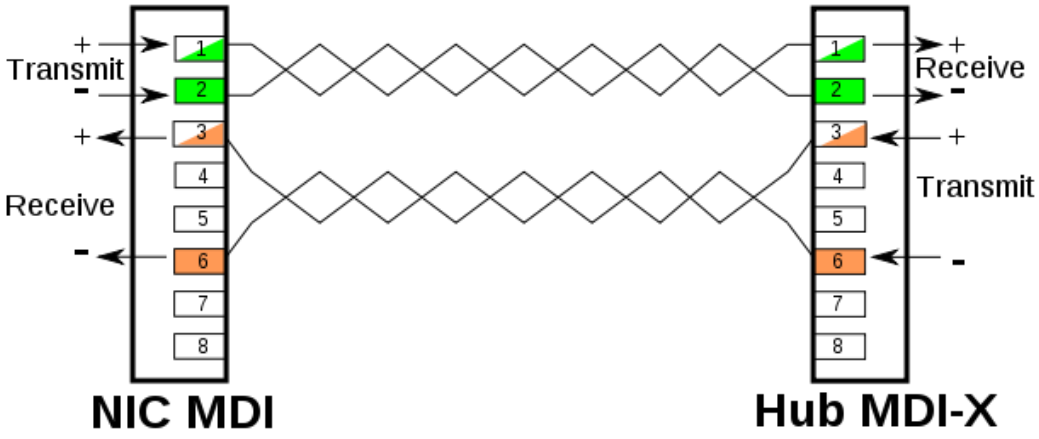
Brány jsou zařízeními propojující sítě kombinací softwaru nebo hardwaru, které komunikují různými protokoly. Toto vyžaduje analyzování dat i na vyšších vrstvách OSI modelu.

2.6.5 Kompatibilita

Aby bylo možné propojovat různé síťové zařízení fungující v různých režimech, musí být porty schopny automaticky měnit nastavení. Jedna ze základních funkcí je *autonegotiation*, což je funkce nastavující rychlost připojení 10/100Mbit a jeho režim duplex/half duplex. Tato funkce nám umožňuje ještě před započetím komunikace nastavit rozhrání tak, aby data mohla proudit co možná nevyšší rychlostí, kterou podporují obě fyzická zařízení.

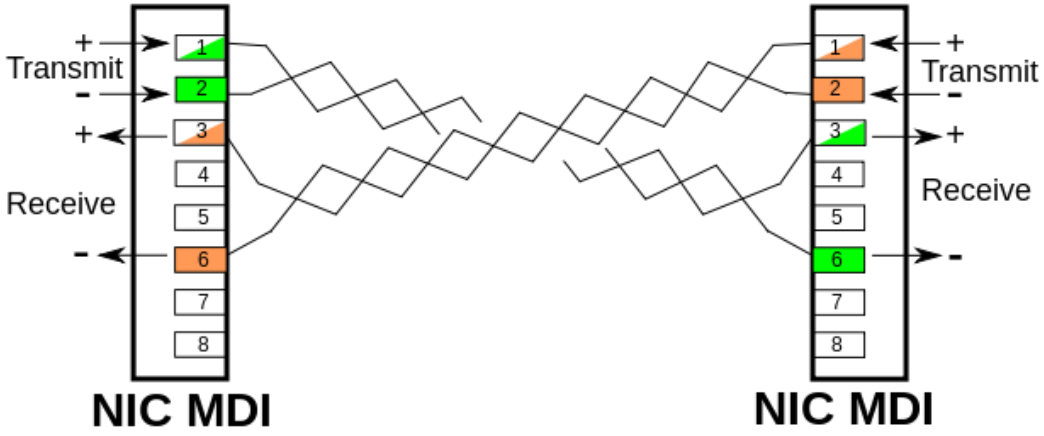
Dalším podstatným prvkem zajišťujícím kompatibilitu je *Autocrossing*. Rozhraní Tx a Rx jsou na přepínačích umístěny zrcadlově v porovnání s koncovými stanicemi, takže Tx přepínače je připojeno na Rx terminálu, což je správně. V případě, že bychom chtěli propojit 2 přepínače, došlo by k nesprávnému zapojení. Z toho důvodu musíme použít tzv. překřížený kabel. Díky funkci autocrossing už dnes ve většině případů nemusíme řešit, jakým kabelem propojujeme zařízení v síti.

Funkce *autonegotiation* a *autcrossing* jsou vázány s pojmem MDI (Medium dependent interface), jenž definuje, jaké vlastnosti má daný Ethernetový port. Většina dnešních síťových prvků je těmito funkcemi vybavena. Pokud by tomu tak však nebylo, bylo by nezbytné důsledně dodržovat zapojení křížovými a nekřížovými kabely dle toho, které prvky propojujeme.



Obrázek 12 – Spojení MDI a MDI–X (cross) nekříženým kabelem (Wikipedia.org, [b.r.])

Další pojem, jenž se dá s MDI zaměnit, je MII (Media independent interface). MII souvisí s propojením mezi MAC a PHY (Frazier, jr., 1995), tedy fyzickou a spojovou vrstvou, které bylo definováno firmou Xerox z důvodu velkého počtu různých variant fyzické vrstvy.



Obrázek 13 – Spojení dvou MDI portů kříženým kabelem (Wikipedia.org, [b.r.])

2.7 Pasivní síťové prvky

V průmyslu jsou zapotřebí nejen aktivní prvky odolné proti vibracím, vlhkosti, prašnému a agresivnímu prostředí, ale i prvky pasivní. Organizace IAONA zabývající se platformou průmyslového Ethernetu vydala příručku *Planning and Instalation Guide for Industrial Ethernet*. Tato příručka definuje základní vlastnosti průmyslových prvků, jež rozdělila do dvou tříd.

Třída	**Light Duty**	**Heavy Duty**
Krytí	IP20 podle IEC 60529, EN 60529	IP67 podle IEC 60529, EN 60529
Provozní teplota	0 až +55 °C	–20 až +65 °C
Rázy	15 *g*/11 ms (IEC 60068-2-27, EN 60068-2-27)	
Vibrace	5 *g* při 10 až 150 Hz (IEC 60068-2-27, EN 60068-2-27, krit. A)	

Tabulka 2 – Třídy kabelů do průmyslového prostředí a jejich vybrané charakteristiky (Zezulka a Hynčica, 2007)

2.7.1 Konektory

Type	IEC standard	Also recognized by [1)]	
M12-4, 4 pole with D-coding	IEC 61076-2-101-A1	ODVA, PNO	
RJ45-IP67 circular bayonet locking	IEC 61076-3-106 (Variant 01)	ODVA	
RJ45-IP67 push-pull locking	IEC 61076-3-106 (Variant 06)	IDA, Interbus-Club	

[1)] For details see documentation of each organization.

Tabulka 3 – Konektory doporučované organizací IAONA (IAONA, 2003)

V současnosti existuje na 20 druhů Ethernetových konektorů pro průmyslové aplikace. Jak již bylo uvedeno výše IAONA rozdělila použití kabeláže a konektorů do dvou tříd, a to Heavy Duty odpovídající IP67, a Light Duty odpovídající IP20, na které klade rozdílné nároky z hlediska vibrací, EMC aj. (viz Tabulka 2).

2.7.2 Kabely

V současnosti je v průmyslovém prostředí postačující kabeláž kategorie 5 oproti prostředí kancelářskému, kde se používají kabely o kategorii výše. Nicméně může být zajímavé srovnání kabelů metalických, optický kabelů se skleněným vláknem a kabelů s umělým světlovaný vláknem (Zezulka a Hynčica, 2007).

- SI–POF 980/1000 – polymethylmethakrylátové jádro s PE obalem
- HCS (Hard Clad Silica) – jádro z pevného oxidu křemičitého s obalem z PUR nebo TPE

Určení	Kabely pro instalace	Spojovací kabely
Průřez jádra	AWG 24/1 až AWG 22/1	AWG 26/7 až AWG 24/7
Norma	EN 50288-2-1	EN 50288-2-2
Počet párů	2 nebo 4	
Frekvenční rozsah	kategorie 5 (100 MHz)	
Průměr kabelu (čtyři páry)	6 až 8,5 mm (Light Duty), 7 až 9,5 mm (Heavy Duty)	5 až 6 mm (Light Duty), 6 až 7 mm (Heavy Duty)
Materiál pláště	nespecifikován	
Stínění	společné měděné stínění nebo fólie plus měděné stínění	
Maximální délka kabelu	100 m	60 m, popř. 50 m pro spolehlivý přenos
Hořlavost	IEC 60332-1 (jednoduchá zkouška hořlavosti)	
Tvorba solí	podle normy IEC 60754-2	

Tabulka 4 – Metalické kabely specifikované organizací IAONA (Zezulka a Hynčica, 2007)

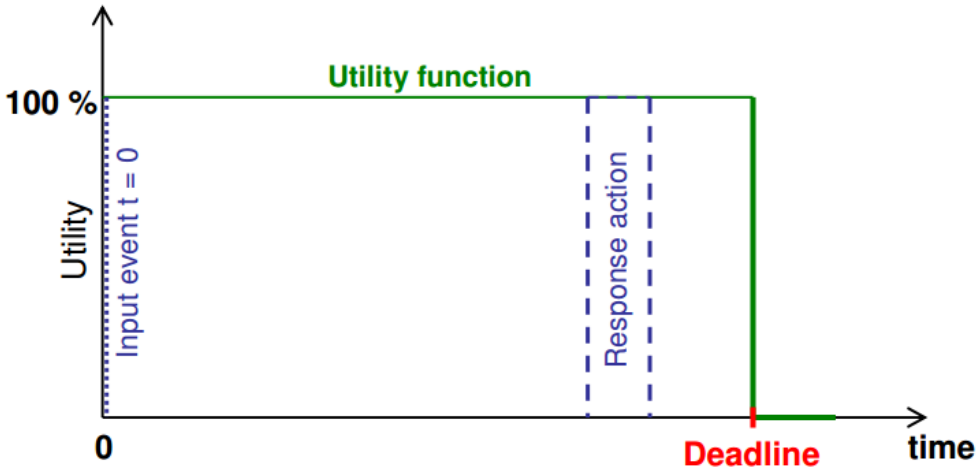
Typ kabelu	POF/HCS	Skleněný světlo-vodič	Metalický vodič
EMC	velmi dobrá		špatná
Galvanické oddělení	ano		ne
Riziko iniciace výbuchu	ne		ano
Malá hmotnost	ano		ne
Malý rádius ohybu	ano	ne	ano
Snadná montáž	ano (nejlepší)	ne	ano
Velká šířka pásma	ne	ano (nejlepší)	ano
Útlum signálu	malý	největší	velký
Ekonomika	dobrá	špatná	nejlepší

Tabulka 5 – Porovnání metalický a optických kabelů pro průmyslový Ethernet (Zezulka a Hynčica, 2007)

2.8 Princip průmyslového Ethernetu

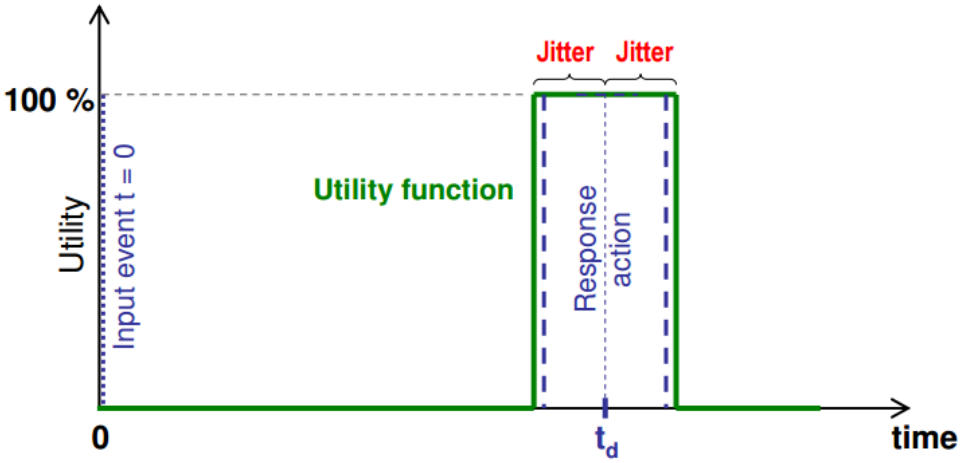
Ethernet 802.3 s přístupovou metodou CSMA/CD je nedeterministickým systémem, a tudíž se nehodí pro průmyslové aplikace a pro práci v reálném čase, kde je právě determinismus velice důležitý. Determinismus pro oblast komunikačních technologií můžeme chápat jako vlastnost systému zpracovávat data, která splňuje dva základní požadavky (Zezulka, [b.r.]):

- Včasnost (timeliness) – řídící/komunikační systém odpoví do určené doby
- Současnost (synchronism) – schopnost synchronně zasílat data z jednotlivých subjektů s určitou přesností (jitter)



Obrázek 14 – Včasnost (Zezulka, [b.r.])

Metoda CSMA/CD, která přiděluje přístup na principu „best effort“ není jediným nedostatkem standardního Ethernetu, protože i skupina protokolů TCP/IP prohlubuje zpoždění a nejednoznačnost odeslání dat (Zezulka a Hynčica, 2007).

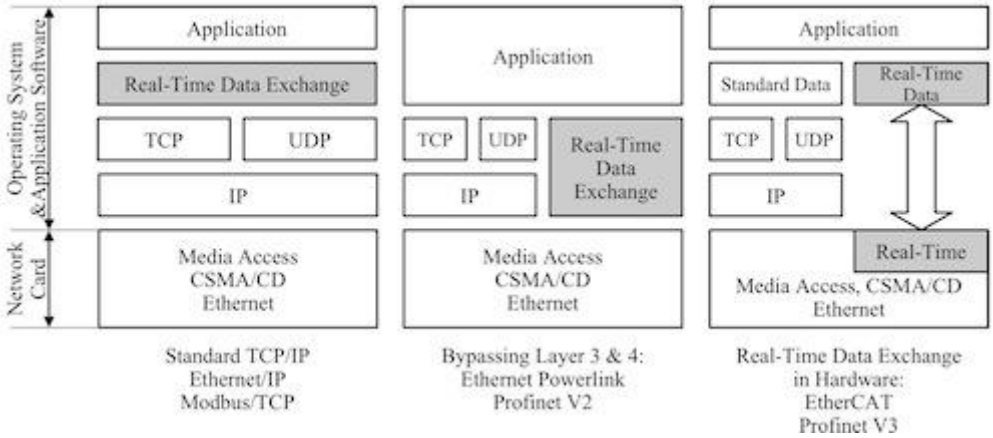


Obrázek 15 – Současnost (Zezulka, [b.r.])

Aby bylo vůbec možné se přiblížit požadavkům průmyslových aplikací, je nezbytné posílit vlastnosti reálného času. Je více způsobů, jak tohoto lze docílit, a to ať už použitou konfigurací a topologií sítě, nebo změnou HW či standardního TCP/IP stacku.

Vzhledem k požadavkům na reálný čas a náklady jsou použity technologické principy, které lze rozdělit do 3 tříd (EtherCAT.org, [b.r.])

- **Třída A** používá nemodifikovaný HW a TCP/IP stack a pro přenos kritických zpráv používá stack TCP/UDP/IP. Tato třída funguje na pricipu „best effort“ a její výkon je limitován nepředvídatelným zpožděním.
- **Třída B** používá nemodifikovaný HW, ale standardní TCP/IP stack se používá pouze v kontrolovaných a omezených případech. Hlavní komunikace je zajištěna jiným protokolem posílaným přímo v Ethernetovém rámci.
- **Třída C** pro dosažení požadovaného výkonu musí použít k tomu určený HW a vlastní protokoly. TCP/IP se vůbec nepoužívá. Příkladem této třídy může být například PROFINET IRT.



Obrázek 16 – Porovnání základních struktur průmyslových Ethernetových protokolů (Voss, 2013)

2.8.1 Úpravy standardního Ethernetu TCP/IP pro průmyslové použití

Tyto změny odpovídají Třídě A, a můžeme se říci, že by měly vždy předcházet změnám HW a SW, protože tyto změny zachovávají 100% kompatibilitu s Ethernetem použitým pro kancelářské použití a nad to bývají častěji méně finančně náročné.

2.8.1.1 Vysokorychlostní přenos

Zvýšení rychlosti o desetinásobek sníží dobu přenosu jednoho paketu o jednu desetinu. To v případě zvýšení rychlosti z 10 Mb/s, kde přenos jednoho paketu délky 500 bajtů trvá 0,4ms, na rychlost 100Mb/s znamená snížení doby přenosu na 40 µs. Zkrácení doby přenosu zpráv je žádoucí pro dosažení vlastností systému reálného času

2.8.1.2 Použití plného duplexu

V současnosti už není prakticky možné se setkat s polovičním duplexem, neboť se dnes již používá pouze kabeláž s kroucenými páry. Nicméně i toto je cesta, jak zvýšit rychlost přenosu.

2.8.1.3 Přepínání

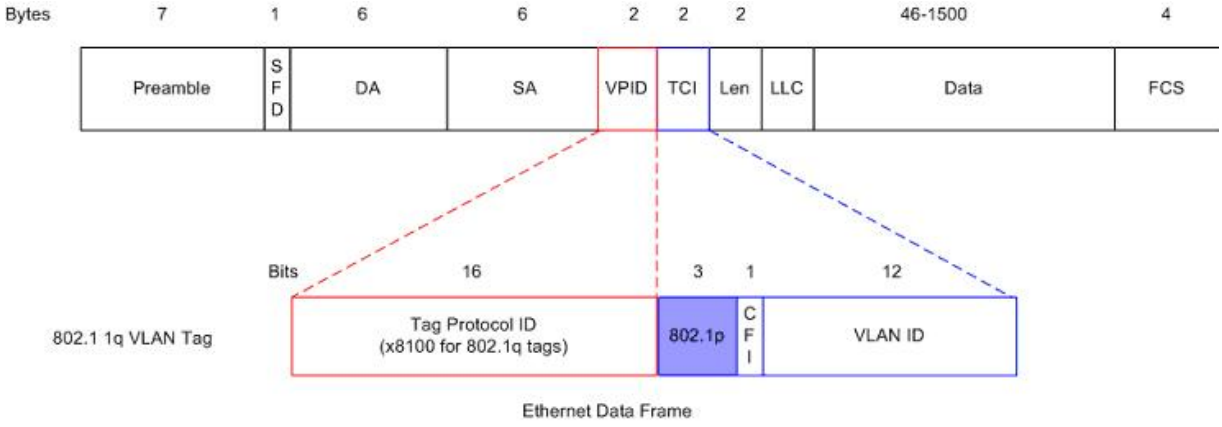
Oproti rozdělovačům nám použití přepínačů umožní snížit pravděpodobnost kolize, neboť rozdělují kolizní doménu na mnohem menší segmenty.

2.8.1.4 Protokol UDP

Protokol TCP zaručuje ze svého principu doručení dat. Toho se dosahuje pomocí navazování komunikace trojnásobným „handshakem“, potvrzování každé přijaté zprávy a také složitějším ukončování komunikace. Všechny tyto kroky znamenají časové zpoždění, a pokud není jedna zpráva doručena, tak tuto nelze poslat hned v dalším cyklu. Protokol UDP je mnohem jednodušší a tím efektivnější co do rychlosti přenosu. Tento protokol nenavazuje spojení, netrvá na potvrzení dat a hodí se pro cyklické posílání dat. Abychom zajistili doručení dat pomocí protokolu UDP, je zapotřebí vytvořit kontrolní mechanismy na vyšších vrstvách.

2.8.1.5 Priority v rámci Ethernet

Standardem IEEE 802.1q byla rozšířena hlavička Ethernetového rámce o podporu VLAN (Sonicwall, [b.r.]). Rozšířená hlavička obsahuje navíc 4 bajty. První 2 bajty obsahují hodnotu



Obrázek 17 – Ethernetový rámec dle 802.1q (Sonicwall, [b.r.])

0x8100 a slouží jako identifikace toho, že následující 2 bajty nesou informace o samotné VLAN. V těchto 2 bajtech se nachází 3 bitové pole nazývající se „Priority Code Point“ (PTP) dle IEEE P802.1p. Hodnota 0 značí nejmenší prioritu a hodnota 7 prioritu nejvyšší. Pokud přiřadíme časově kritickým zprávám vysokou prioritu, tak tím zkrátíme jejich dobu přenosu a podpoříme tím determinismus sítě.

2.8.1.6 Rozdělení sítě

Rozdělením sítě na dvě části a propojení přepínače s firewallem, může zlepšit vlastnosti reálného času. V určitém segmentu sítě pak nebudou kritická data omezována daty, které pro ně nejsou určeny. Firewall může na základě určitých pravidel filtrovat určité broadcastové nebo multicastové zprávy, anebo povolit přístup do sítě jen určitým uživatelům pro zvýšení bezpečnosti.

2.8.1.7 Producer–consumer / publisher–subscriber

Jedná se o metody pro zasílání dat většímu množství příjemců pomocí multicastu. Cílem je vytvořit z příjemců skupiny, kterým by se mohly zasílat totožná data, a také synchronizovat tok dat. Co se týče metody Producer–consumer, je tato založena na vysílání zpráv Producerem, které jsou opatřeny identifikátorem a pouze Consumer, pro kterého je zpráva určena ji akceptuje. Metoda Publisher–subscriber funguje na pricipu, kdy se Subscriber přihlásí o odběr dat a Publisher odesílá data, pouze tomu, kdo je zaregistrován. Registrace je časově omezena, a pokud Subscriber nepotvrdí po nějakém čase svůj odběr, je z tabulky příjemců vymazán.

2.8.1.8 Synchronizace

Synchronizace je jednou z důležitých vlastností deterministického systému, kterou Ethernet z podstaty přístupu k médiu CSMA/CD postrádá. Sběrnice typu fieldbus mnohdy řeší synchronizaci metodou master–slave, anebo použitím dalšího kanálu, jako například samostatného vodiče udávajícím hodiny. To v případě Ethernetu nepřipadá v úvahu, avšak i v případě použití všech výše uvedených vylepšení není dosaženo takových kvalit reálného času, který je potřeba pro řízení pohonů os obráběcích strojů, atd. V sítích LAN se pro jednoduchou synchronizaci používají protokoly NTP a SNTP, ale tyto nejsou dostatečně přesné pro průmyslové aplikace. Z těchto důvodu bylo potřeba přidat do průmyslového Ethernetu kvalitní a přesný synchronizační prostředek.

2.8.1.8.1 PTP

Takovým je právě protokol PTP dle standardu IEEE 1588, který umožňuje extrémně vysoký stupeň synchronizace. Tento dokonce dosahuje vyššího stupně synchronizace než sběrnice typu fieldbus (Zezulka a Hynčica, [b.r.]). Jedná se o efektivní a přístupnou metodu, jež dokáže plně nahradit ostatní metody synchronizace pomocí přijímačů GPS, anebo pomocí přídavného komunikačního kanálu. Tento protokol se také díky svým vlastnostem stal základem průmyslových protokolů Powerlink, Profinet nebo CIPsync.

Princip PTP spočívá v tom, že si všechny podřízené stanice synchronizují čas dle stanice master a pomocí speciálních zpráv si změří zpoždění přenosu a dopočítají odchylku od reálného času. Časová nejistota pak dosahuje hodnot menších než 1µs za použití protokolu TCP/UDP/IP.

Synchronizace mezí účastníky probíhá uspořádáním do struktury master–slave podle toho, kdo má „nejkvalitnější hodiny“. Hodiny jsou vybrány algoritmem „Best Master clock“, který porovnává vlastnosti hodin všech účastníků na základě zprávy Sync (Announce v PTPv2). Stanice ve stavu slave se na základě této zprávy rozhodne, zda přejde do stavu master nebo ne a naopak se stanice master na základě zprávy Sync může rozhodnout přejit do stavu slave. Algoritmus je totožný pro všechny stanice a je navržen tak, aby nemohlo dojít ke konfliktu. Zpráva Sync obsahuje tyto informace seřazené dle priorit od nejvyšší (preference, primární zdroj času, přesnost, stabilita, vzdálenost, unikátní identifikátor).

Po sestavení struktury master–slave přichází dvě fáze, které eliminují zpoždění přenosem. V první fázi si podřízená stanice určí posun hodin na základě příchozí zprávy Sync, jež je vysílána v pravidelných intervalech. Tato zpráva může obsahovat čas odeslání, pokud to však možné není, je tento čas odeslán v následující zprávě Follow_Up. V druhém kroku následuje samotné měření přenosového zpoždění a to tak, že slave odesílá zprávu Delay_Req a master odpovídá zprávou Delay_Resp s časem přijetí zprávy Delay_Req. Na základě této zprávy si stanice slave dopočítá hodnotu zpoždění vlivem přenosu.

Je potřeba zmínit, že časové značky protokolu PTP jsou vkládány v rozhraní MII (Media Independent Interface), aby se zamezilo zpoždění způsobené průchodem zpráv vrstvami protokolu. Vkládání značek v rozhraní MII vyžaduje HW podporu v síťovém rozhraní.

2.9 Funkční bezpečnost

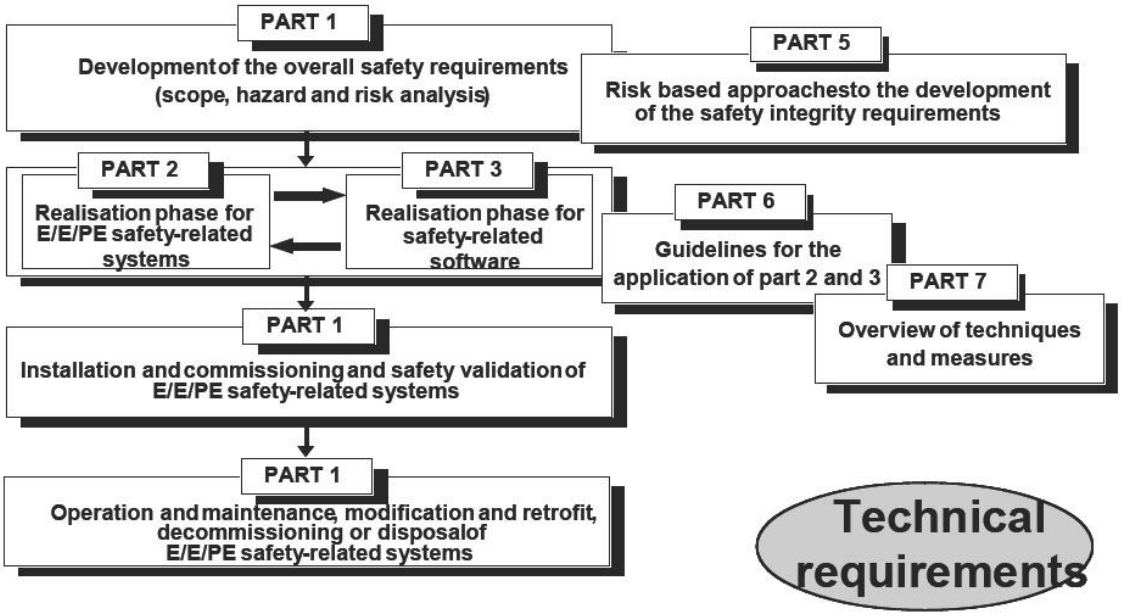
Tato část je zpracována na základě Zezulka a Hynčica (2007), Abdullah [b.r.], Ladkin (2008), IAONA (2006).

V otázkách bezpečnosti týkající se funkce systému, vycházíme z normy IEC 61508, která se zabývá bezpečnostní elektrických, elektronických a elektronických programovatelných systémů E/E/EP. Pro tuto normu česká verze překlad ČSN EN 61508.

IEC 61508 se skládá ze 7 částí (IEC 61508–1 až IEC 61508–7) z nichž první 4 jsou normativní a části 5–7 jsou informativní.

2.9.1 Základní principy IEC 61508

Je nutno uvést, že z důvodu nejednoznačného překladu, existuje mnoho výkladů pojmů a verzí, které se dle mého názoru odchylují od skutečného výkladu, a to i v české verzi normy ČSN IEC 61508. Nejedná se však o nic tak závažného, neboť norma má sloužit jako návod a předloha k eliminování rizik, která neobsahuje zcela konkrétní kroky. Ty jsou vždy na výrobcích zařízení. Následující text je můj volný překlad z normy IEC 61508.

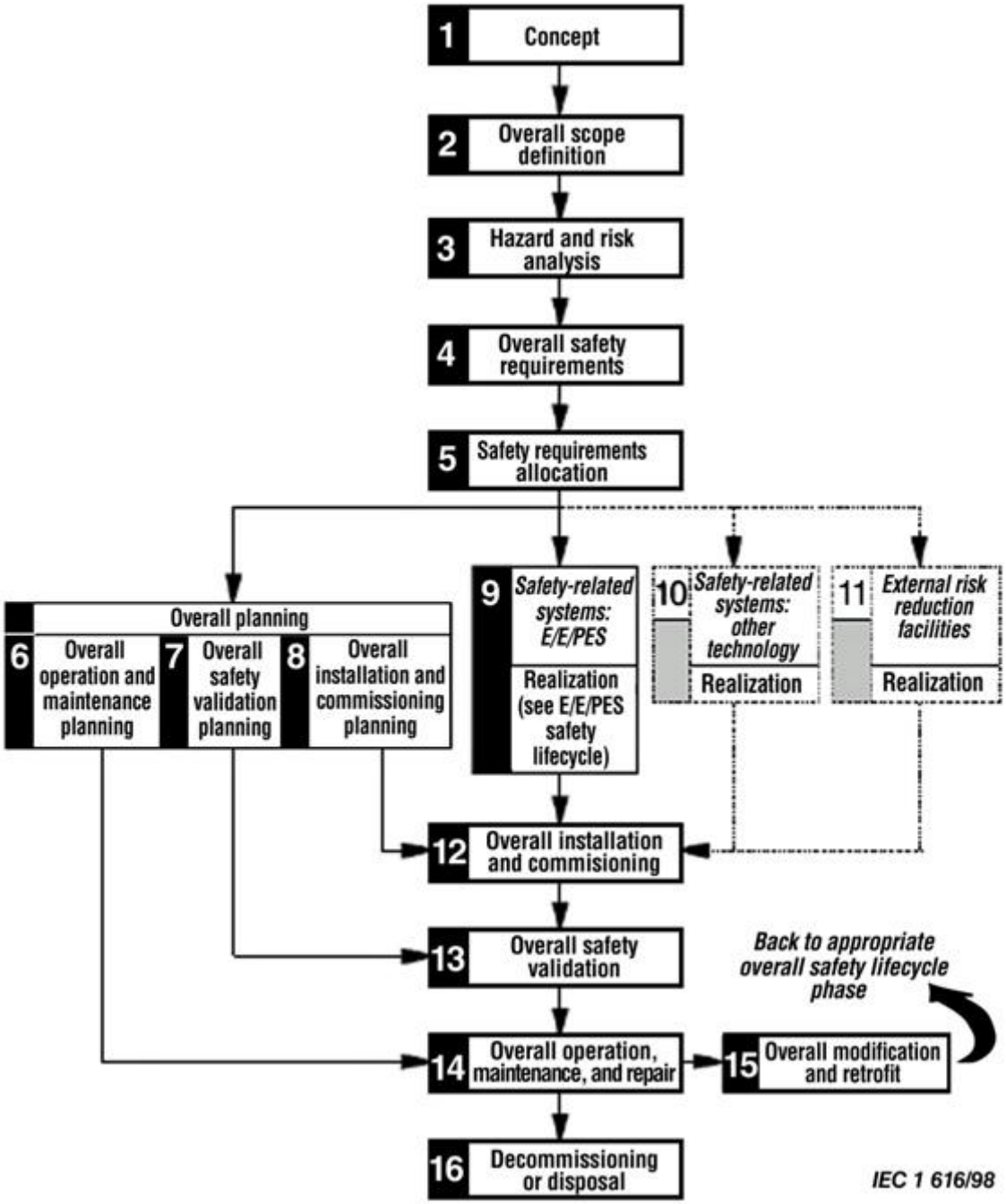


Obrázek 18 – Hlavní části IEC 61508 (Abdullah, [b.r.])

2.9.1.1 Životní cyklus systému

Detailní popis životní cyklu systému od návrhu po jeho vyřazení. Typický životní cyklus obsahuje informace o požadavcích, specifikacích, kódování, údržbě a likvidaci. Jsou 3 požadavky pro životní cyklus dle IEC 61508

1. Pro finální produkt
2. Pro dokumentaci
3. Pro zdroje



Obrázek 19 – Životní cyklus systému dle IEC 61508 (Abdullah, [b.r.])

2.9.1.2 Funkční bezpečnost

- Bezpečnost – znamená nepodléhání nepřijatelnému riziku fyzického zranění, poškození lidského zdraví, anebo přímého nebo nepřímého poškození majetku či prostředí
- Funkční bezpečnost – je část obecné bezpečnosti závisející na správné činnosti systému nebo zařízení na základě vstupů

2.9.1.3 Riziko a jeho minimalizace

Neexistuje nulové riziko. Bezpečností funkce jsou uvažovány pro snížení rizika. Existují 3 druhy rizika

- EUC (Equipement under Control) – jde o druh rizika vznikající od EUC nebo s jeho interakcí;
- Akceptovatelné riziko – jde o riziko akceptovatelné v daných souvislostech založených na současných hodnotách společnosti;
- Zbytkové riziko – je částí rizika, jenž zbývá po aplikování ochranných opatření;

2.9.1.4 Dělení (pod)systému

- EUC (Equipment Under Control) – je podsystémem skládající se z částí, které provádějí některé nebo všechny funkce, pro něž byl systém navrhnut;
- EUCCS (EUC Control Systém) – je systémem reagujícím na vstupní řídící signály od procesů nebo operátora a generující výstupní signály způsobující řádné fungování EUC;
- SRS (Safety related system) – implementuje nezbytné bezpečnostní funkce a jeho záměrem je dosáhnout nezbytné bezpečnostní spolehlivosti pro požadované bezpečnostní funkce;

2.9.1.5 SIL (Safety Integrity Level)

Je to diskrétní rozdělení spolehlivosti na 4 stupně označující spolehlivost bezpečnostních funkcí v E/E/PE SRS. SIL 1 je nejnižší úroveň a SIL 4 je úrovní nejvyšší.

Safety Integrity Level	Probability of failure on demand, average (Low Demand mode of operation)	Risk Reduction Factor
SIL 4	$>=10^{-5}$ to $<10^{-4}$	100000 to 10000
SIL 3	$>=10^{-4}$ to $<10^{-3}$	10000 to 1000
SIL 2	$>=10^{-3}$ to $<10^{-2}$	1000 to 100
SIL 1	$>=10^{-2}$ to $<10^{-1}$	100 to 10

Tabulka 6 – Rozdělení tříd SIL dle pravděpodobnosti selhání vzhledem k množství požadavků na systém (IAONA, 2006)

Safety Integrity Level	Probability of dangerous failure per hour (Continuous mode of operation)
SIL 4	$>=10^{-9}$ to $<10^{-8}$
SIL 3	$>=10^{-8}$ to $<10^{-7}$
SIL 2	$>=10^{-7}$ to $<10^{-6}$
SIL 1	$>=10^{-6}$ to $<10^{-5}$

Tabulka 7 – Rozdělení tříd SIL dle pravděpodobnosti selhání vzhledem k souvislé době běhu systému (IAONA, 2006)

2.9.1.6 ALARP (As Low As Reasonably Practicable)

Jedná se o rizika, která lze hodnotit jako zanedbatelná, a rizika, která jsou naopak zcela nepřípustná. Mezi těmito dvěma extrémy může být riziko akceptováno či nikoliv na základě hodnoty možného zisku, nebo nákladů na redukci rizika. Riziko v této oblasti by mělo být „ALARP“ = „tak nízké jak je rozumně proveditelné“

Důležitě definice:

- Újma – fyzické zranění nebo poškození lidského zdraví, anebo také přímého nebo nepřímého poškození majetku či prostředí;
- Nebezpečí – potenciální zdroj újmy;
- Bezpečnost – nepodlehnutí nepřijatelnému riziku;
- Bezpečností funkce – funkce implementována v E/E/PE SRS, jiná technologie SRS, nebo omezení zdrojů vnějších rizik se záměrem dosáhnout bezpečného stavu pro EUC s respektování specifických rizikových událostí;
- Bezpečností integrita – pravděpodobnost uspokojivé funkce bezpečnostní části systému;
- Nehoda – nežádoucí a neplánována událost, jejímž výsledkem je určitá úroveň ztrát;
- Bezpečnostní životní cyklus – nezbytné aktivity spojené s implementací SRS zabývající se časovým období od samotného začátku návrhu konceptu, fázi projektu a ukončením, kdy všechny z E/E/EP SRS, další související

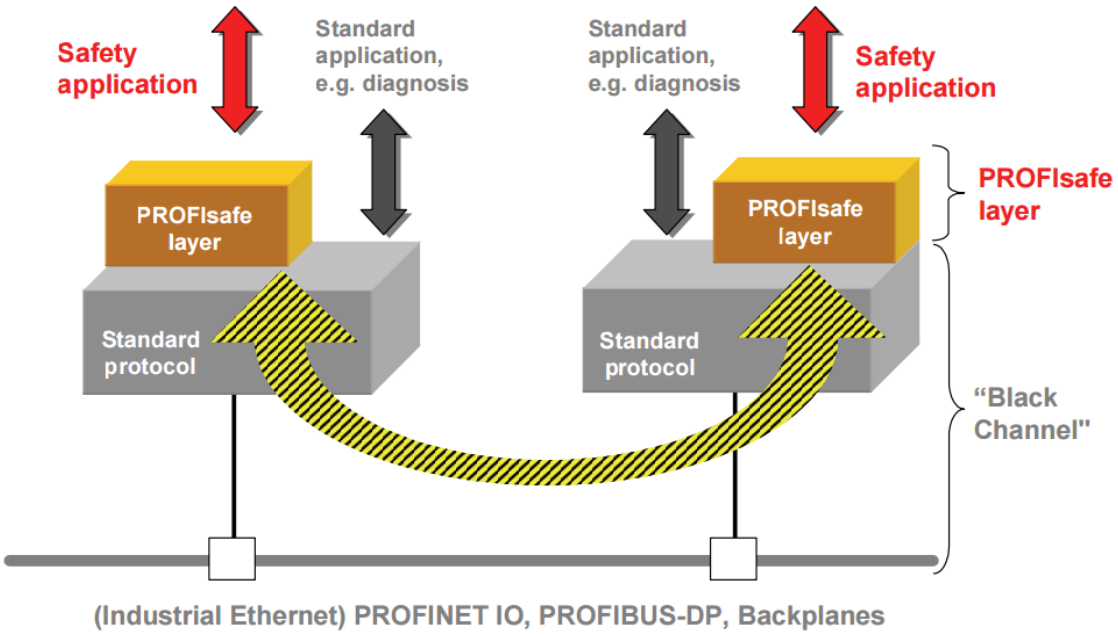
bezpečnostní technologie a minimalizace vnějších rizik už není nadále možné používat;

- Riziková událost – riziková situace vedoucí k újmě;
- Riziková situace – okolnost, kdy je osoba vystavena riziku;
- Závažnost – úroveň ztrát.

2.9.2 „Black Channel“

Ethernet je ve svém standardním provedení dosti robustní metoda, která odpovídá třídě SIL 3, ale i přesto je třeba se bezpečností v průmyslové sítí Ethernet zabývat. Základní požadavek je neměnit nic na HW, takže se bezpečností funkce musí implementovat do aplikační vrstvy.

Princip „black channel“ tkví v tom, že je na něj nahlíženo jako na nespolehlivý datový



Obrázek 20 – Princip „Black Channel“ (PI, [b.r.])

kanál, jenž funguje blíže nespecifikovaným způsobem. Mohou se v něm vyskytovat poruchy a také chyby v přenosu. Pro zajištění přenosu bezpečných dat je proto nezbytné implementovat mechanismus, jenž minimalizuje možná rizika. Tento bezpečnostní mechanismus je často proveden formou bezpečností vrstvy, která zajišťuje bezchybnost přenosu bezpečných dat. Chyby, které se mohou vyskytnout, a možnosti jejich eliminace jsou uvedeny v tabulce 8. Pro ostatní data jsou používány standardní protokoly. Je nutné si uvědomit, že tyto bezpečností prvky mají nevýhodu ve zvýšené režii, zvýšeném zpoždění, což není akceptovatelné pro aplikace vyžadující tzv. „Hard real time“. Proto například protokoly Powerlink a Profinet IO nepoužívají potvrzování příjmů.

Možné chyby dat při přenosu	Metody eliminace chyb						
	sekvenční číslování dat	časová značka dat	potvrzení příjmu dat	identifikace odesílatele a příjemce	zálohování dat	redun-dance dat	kontrola platnosti dat
opakování	x	x				x	
ztráta	x		x			x	
vkládání	x		x	x		x	
špatné pořadí	x	x				x	
nekonzistence			x		x		x
zpoždění		x					
propojení safe a non-safe			x	x			x
přetečení paměti směrovače		x					

Tabulka 8 – Chyby v přenosu a metody jejich eliminace (Zezulka a Hynčica, 2007)

2.9.3 Porovnání s protokoly fieldbus

V protokolech skupiny fieldbus bylo kolikrát nezbytné vést provozní data jednou sítí a bezpečná data ve zvláštní síti. To v Ethernetu není vůbec potřeba díky velké šířce pásma, velké EMC odolnosti a velkému stupni robustnosti.

Přesto všechno však zůstává mnohdy v odborné veřejnosti zastáván názor, že je nezbytné oddělovat sítě i v případě průmyslového Ethernetu. Během ARC fóra v roce 2014 dostal David Loveridge (senior control engineeer and partner at ICR Engineering) otázku na stejné téma a odpověděl:

„I don't see any reason why you would want to have a dedicated safety network for a safety systém" (Greenfiled, 2014)

2.10 Informační bezpečnost

Následující část je zpracována na základě Zezulka a Hynčica (2007) a IAONA (2006).

V současnosti je informační bezpečnost velmi diskutované téma a dá se říci, že zabezpečení Ethernetu v normálním prostředí je na velmi dobré úrovni. V dřívějších dobách byly průmyslové komunikační sítě vytvářeny na proprietárních protokolech, upravené pro lokální použití a také tato síť nepřesahovala hranice areálu. S pronikáním Ethernetu do průmyslových sítí vyvstala otázka jeho zabezpečení, neboť právě kompatibilita se standardní sítí, sběr dat a centrální řízení patří mezi hlavní výhody, ale také to představuje potenciální riziko.

Organizace IAONA vydala příručku *„IAONA Handbook Network Security"*, které se zabývá bezpečnostními kritérii a také specifikuje metodiku strategie obrany.

2.10.1 Bezpečnostní kritéria

2.10.1.1 Dostupnost (Availability)

Dostupnost znamená, že neautorizované osoby nemohou zamezit vstup, nebo znemožnit použití systému autorizovaným uživatelům. Porušení této dostupnosti může způsobit problémy s bezpečností, ztrátu kontroly a monitorování systému, což může vést k tomu, že systém neodpoví ve stanoveném čase.

2.10.1.2 Ochrana třetích stran (Third party protection)

V tomto bodě je hlavním cílem zamezit rozšíření poškození z jednoho systému, kterým došlo neočekávaným nebo nezamýšleným chováním, do dalších systémů, případně do systémů třetích stran. Úspěšný útok může vést k ovládnutí systému a jeho použití proti systémům třetích stran.

2.10.1.3 Integrita (Integrity)

Cílem je, aby přenášená data nemohla být modifikována, zpožděna nebo podvrhnuta neautorizovanými osobami nebo systémy.

2.10.1.4 Dohledatelnost (Audiability)

Cílem je umožnit zrekonstruování kompletní historie akcí systému na základě všech platných záznamů – logů. Důraz je kladen na zjištění a porozumění důvodů, které vedly k chybě systému. K tomu je potřeba kompletní zmapování všech okolností, kontext událostí a požadavků.

2.10.1.5 Autorizace (Authorization)

Slouží k zamezení přístupu osobám či systémům k systému, anebo k jeho částem bez příslušného povolení. Systém autorizace musí mít mechanismy, aby mohl rozlišit mezi oprávněným a neoprávněným uživatelem, kvůli všem ostatním bezpečnostním kritériím: Důvěrnost, Integrita a další.

2.10.1.6 Autentizace (Authentification)

Autentizace slouží k jednoznačnému určení uživatele, který se snaží pracovat se systémem. Tito uživatelé mají vytvořené účty, ke kterým se přihlašují obvykle pomocí přístupových jmen a hesel.

2.10.1.7 Neseriózност (Non–reputability)

Podstatou je, aby bylo možné poskytnout jednoznačný důkaz třetím stranám o tom, kdo učinil nějaký zásah do systému. To je velmi důležité pro přenesení nákladů spojených s odstraněním škod na konkrétního viníka. Je zapotřebí znemožnit padělaní logů.

2.10.1.8 Důvěrnost (Confidentiality)

Zamezuje vyzrazení důvěrných informací neoprávněným osobám a systémům. Většinou se tak činí vhodným šifrováním, jež zajistí, že data nemohou být čtena třetími stranami.

	Classification			
	none	low-medium	high	very high
Integrity	log data failure events	repeat data, use checksum	rare failure acceptable, production loss	no failure acceptable, severe production loss
Non-repudation	no measures	store access information to log files	use authorization and backtrace	use of certificates and secure servers
Confidentiality	data are public available, not protected	use basic mechanisms, single failure may occur	secure data chanels, active protection	encrypted data, failures are not acceptable
Availability	no measures, downtime: some hours	using backup, downtime: <1h	quick replacement, downtime: <5min	redundant system, no downtime
Authentication	without any acces s control	using passwords	using server based user authentication	use of certificates, smart-cards, etc.

Tabulka 9 – Třídy závažnosti dle IAONA (2006)

2.10.2 Metodika IAONA

Tato metodika v 6 krocích nastaví základní parametry zabezpečení sítě, kterými by se pak síťoví administrátoři nebo integrátoři měli řídit.

Pro jednodušší aplikování této metody byl vypracován souhrn tabulek SDS (Security Data Sheets), která mají sloužit projektantům pro podrobné specifikování vlastností zabezpečení sítě.

2.10.2.1 Klasifikace požadavků na zabezpečení

Tato část definuje požadavky na zabezpečení vzhledem k případným ztrátám v případě špatné funkce systému.

- Dopad na výrobu – popisuje dopad špatného fungování systému na výrobu;
- Uživatelská bezpečnost – určuje dopad chyby vzhledem k bezpečnosti uživatele;
- Dopad na soukromí – zohledňuje, jakým způsobem špatná funkce ovlivní zabezpečení citlivých informací;

- Dopad na obraz firmy – popisuje případné ztráty z hlediska pověstí organizace;
- Finanční ztráty – specifikuje velikost ztrát v případě poruchy;
- Znehodnocení kontraktů/práv – popisuje, případný vliv poruchy na patentová práva a důvěrná data.

Dále je třeba na základě dopadů stanovit úroveň zabezpečení sítě. Úrovně zabezpečení jsou: *„žádné, nízké–střední, vysoké, velmi vysoké“*.

Příklad: Porucha v komunikačním kanálu, může vést k vysokých ztrátám ve výrobě, ale k žádnému úniku soukromých dat. Dle toho nastavíme vysokou úroveň bezpečnosti pro kategorii „Dopad na výrobu“ a nízkou pro „Dopad na soukromí“.

2.10.2.2 Komunikační vztahy

V další části definujeme jednotlivé komunikační vztahy pro správné nastavení zabezpečení.

Number	Relation	Comment	Classification
1	Office ↔ Internet	There is no need to access the internet, however it can be allowed (as usual)	Optional
2	Office ↔ Factory	Communication SAP to MES	Necessary
3	Office ↔ Remote Factory	Only one factory	Not applicable
4	Factory ↔ Factory	Only one factory	Not applicable
5	Office ↔ Office	Only one factory	Not applicable
6	Remote Maintenance ↔ Factory	Not planned yet	Optional
7	Home Office/Field Staff ↔ Office	No need for that	Not applicable
8	Remote access of technical service ↔ Factory	Due to security considerations	Forbidden
9a	Within factory	Ethernet is used as communication bus	Necessary
9b	Within factory	Ethernet is used as communication bus	Necessary
10	Within office	Ethernet is used in the office	Necessary

Tabulka 10 – Příklad stanovení komunikačních vztahů (IAONA, 2006)

2.10.2.3 Obranná strategie

Na základě výsledků předchozí části nyní můžeme zvolit obrannou strategii. Na výběr máme dvě základní strategie: *Hard–perimeter* a *Defense–in–depth*

2.10.2.3.1 Hard–perimeter

Jedná se o strategii, která staví síťovou obranu na jednom, případně na skupině zabezpečovací prvků (firewall). Veškerá komunikace do internetu prochází tímto prvkem,

jenž funguje jako neproniknutelná zeď okolo systému. Tato strategie je vhodná pro sít, jež má menší počet internetových bran, menší složitost sítě a menší riziko poškození.

2.10.2.3.2 Defense–in–depth

Tato strategie má rozdělené prvky obrany na jednotlivé vrstvy, což zvyšuje pořizovací náklady a spravování, ale na druhou stranu u této strategie je potřeba delší doby k prolomení její obrany, navíc je možné kombinovat různé obranné prvky a v neposlední řadě je také možné odhalit průnik v přímém přenosu. Tato metoda je vhodná při větším počtu přístupových bodů a také pokud hrozí riziko vyšších ztrát.

2.10.2.4 Obranná struktura

Po zvolení obranné strategie můžeme vybrat konkrétní prostředky zabezpečení a vhodně je rozmístit. Prostředky zabezpečení se rozumí směrovače, přepínače, firewall, filtry paketů, atd. Zvolení prvků a jejich rozmístění by mělo odpovídat také požadavkům na komunikační vztahy a požadavkům na vlastnosti dané sítě. To je vhodné například pro průmyslové aplikace vyžadující přenos v reálném čase. V tomto případě je zřejmé, že nelze používat filtry paketů ani šifrování, které by příliš zpomalilo datový přenos. V těchto případech je nejvhodnější tuto síť oddělit.

2.10.2.5 Zařízení / Protokoly

Další krok je definování pravidel pro filtrování zpráv na základě jednotlivých protokolů. Pro kancelářskou síť lze očekávat použití protokolů HTTP,HTTPS, SMTP, POP3, FTP a další a v průmyslové síti to například bude Modbus TCP/IP, PTP a SNMP. Na základě těchto pravidel a kontextu lze efektivně nastavit obranné prvky.

V doporučení od IAONA lze také nalézt doporučení a bezpečností klasifikace jednotlivých protokolů.

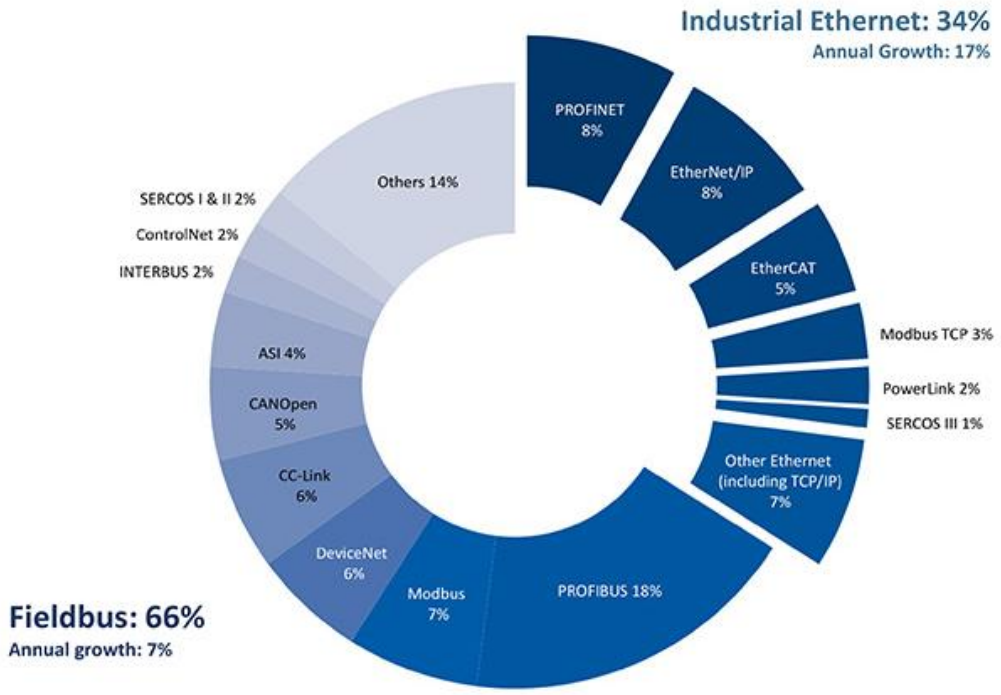
2.10.2.6 Obranná měření

V poslední kroku je nezbytné kontrolní měření ověřující splnění charakteristik obranného systému definovaného v předchozích krocích. Dále je také nezbytné vytvořit organizační pravidla a pravidla pro obsluhu a administrátory nejen během doby instalace a uvádění do provozu, ale i pro běžné používání sítě.

2.11 Současné standardy

Tato část je zpracována na základě Zezulka a Hynčica (2007), EtherCAT.org [b.r.], HMS Industrial Networks [b.r.] a Hoske (2014).

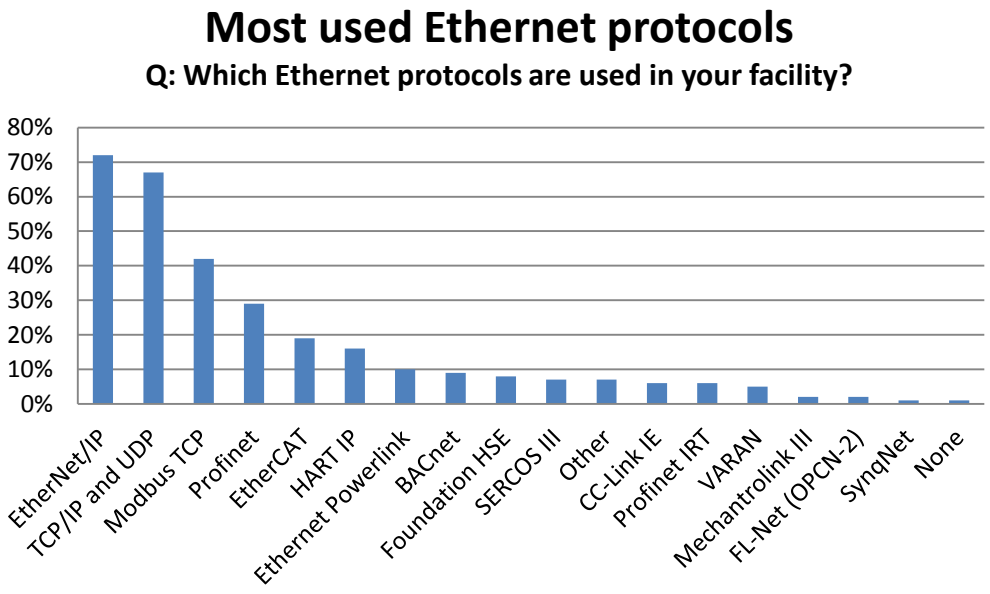
Standardy průmyslového Ethernetu procházejí dynamickým obdobím, v kterém se některým podařilo uspět a rozšířit řady podporovaných produktů, ale i na druhou stranu nalezneme takové, o kterých už nyní víme, že neuspěly.



Obrázek 21 – Porovnání průmyslových Ethernetových protokolů s protokoly fieldbus (HMS Industrial Networks, [b.r.])

Z následujících grafů je zřejmé, že tempo růstu průmyslového Ethernetu je dvojnásobné oproti skupině protokolů fieldbus. Zastoupení jednotlivých průmyslových protokolů je také velmi podobné a v následující části přiblížíme strukturu nejvýznamnějších z nich.

Mezi protokoly, které stabilně vystupují, jako nejvíce používané můžeme zmínit Ethernet/IP, Modbus TCP, Profinet, EtherCAT a PowerLink. Na stabilní pozici jsou i Sercos III a HSE, ale tyto nejsou tak značně zastoupeny jako předešlé protokoly. Dále můžeme jmenovat protokoly HART IP, BACnet, CC–Link IE, které jsou dříve méně známé protokoly, jenž si dokázaly probít cestu. Na druhé straně jsou protokoly EPA, P–net on IP, TCnet, Vnet/IP, Modbus RTPS (DDSI RTPS), které již prakticky nenajdeme v dnešních aplikacích.



Obrázek 22 – Graf rozšíření jednotlivých Ethernetových protokolů (Hoske, 2014)

2.11.1 PowerLink

Tento protokol můžeme řadit do třídy B, jež jsme definovali v sekci 2.3 Principy průmyslové Ethernetu. Z toho vylpývá, že sice nevyžaduje úpravy HW avšak, že došlo k modifikacím na programové výbavě. PowerLink disponuje dvěma módy komunikace, a to standardním TCP/IP stackem a módem „powerlink“ pro aplikace reálného času. Přístupová metoda byla nahrazena multicastem (polling) na principu Consumer/Producer. Co se týče módu Powerlink, tento funguje na principu přidělení časových oken, což zajišťuje deterministický přenos. Na Obrázku 23, je vidět rozdělení přenosu na izochronní část, jež je rozdělena do jednotlivých časových oken a část asynchronní, pro přenos nekritických dat, kde se uplatní přístupová metoda CSMA/CD.

2.11.1.1 Používané verze

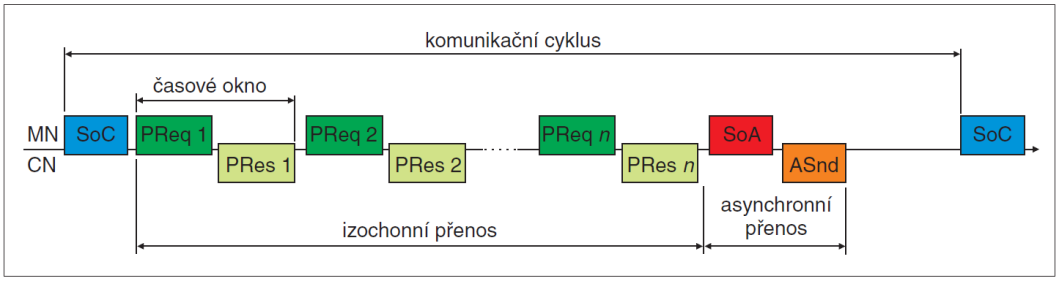
Powerlink se vyskytuje ve více verzích, jejichž specifika a vlastnosti jsou uvedeny v tabulce 11.

Ve verzi 2 jsou používány rozbočovače místo přepínačů a z toho důvodu je komunikace pouze half duplex. Zpoždění na rozbočovači je kolem 500ns, avšak u přepínačů pracující metodou „store and forward“ dosahuje zpoždění pro krátké rámce minimálně 10µs.

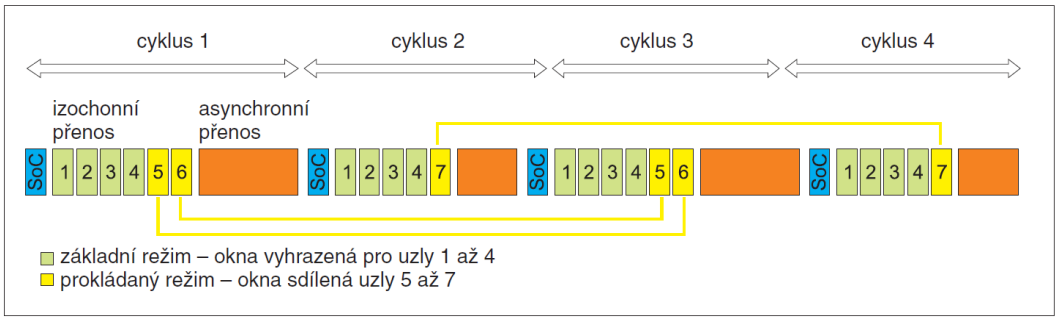
Powerlink version 1	• Protected mode only • Half Duplex Polling (hubs)	Available by B&R only
Powerlink version 2	• Network Management • New Frame structure • MAC–Addresing • Asynchronous Channel • TCP/IP Support • Bridge / Router Support • Profile Support (CANopen)	Spec: 2003 Devices Shipping: 2007
Powerlink version 3	• New protocol principle: Burst Polling • Switched Gbit Ethernet Based • IEEE 1588 synchronisation (PTP)	Announced 2006 First Outline 2009 Spec: ? Devices Shipping: ?
Powerlink version 4	• Poll Response Chaining • Still half duplex, 100 Mbit	Spec: 2012 Devices Shipping (B&R)

Tabulka 11 – Verze a vlastnosti protokolu Powerlink (EtherCAT.org, [b.r.])

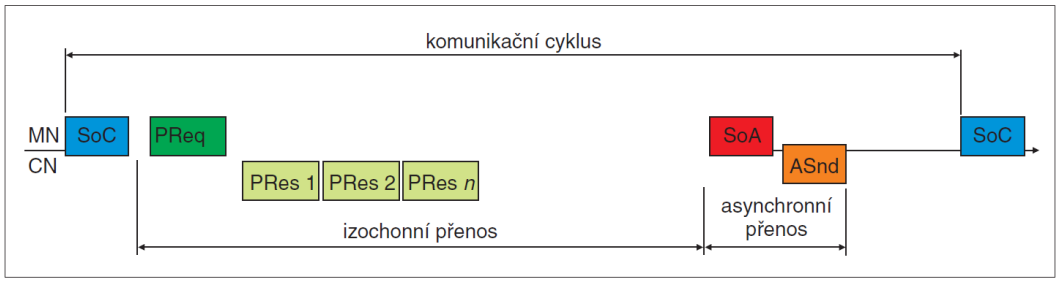
Vzhledem k neuspořádanosti jednotlivých verzí, budeme pokračovat verzí 4, jež je nástupcem verze 2, neboť pro Powerlink verze 3 nebyla dosud vydána specifikace. Ve verzi 4 je největší změna v metodě Poll Response Chaining, díky níž dosáhneme kratšího cyklu, neboť pouze na základě jedné žádosti dojde k získání dat od všech podružných zařízení. Mezi nevýhody patří větší riziko kolizí, neboť komunikace není přidělována striktně jednotlivým stanicím stanicí master. Zpoždění není tak pouze suma zpoždění mezi master a slave, ale je třeba také brát úvahu zpoždění na každém prvku sítě. Z tohoto důvodu je složitější odhadnout výkonnost tohoto protokolu.



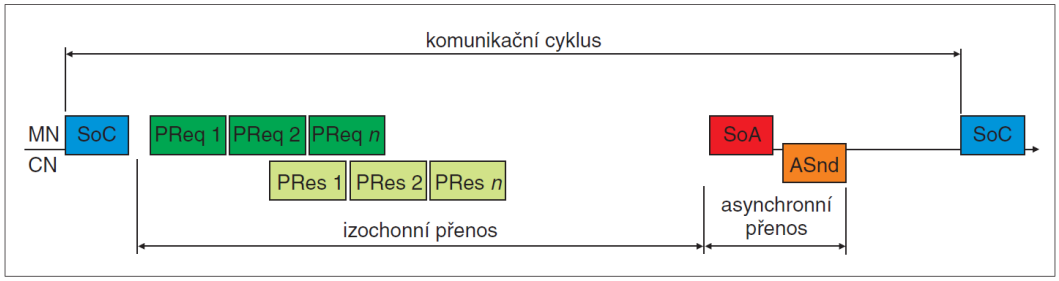
Obrázek 23 – Přenosový cyklus v módu Powerlink verze 2 (Zezulka a Hynčica, 2008)



Obrázek 25 – Prokládaný režim v módu Powerlink (Zezulka a Hynčica, 2008)



Obrázek 24 – Přenosový cyklus v módu Powerlink verze 4



Obrázek 33 – Přenosový cyklus v módu Powerlink verze 3

Ve verzi 3 jsou rozbočovače nahrazeny přepínači, dále je zavedena metoda Burst Polling a komunikace je full duplex. Burst Polling funguje na principu vysílání požadavků ze stanice master na stanice slave, která odpoví, jak jen rychle jí umožní HW. Výhody a nevýhody jsou obdobné jako pro verzi 4.

2.11.1.2 Obecné vlastnosti a výkonové charakteristiky

Tento protokol by vyvinut primárně pro Ethernetová zařízení „se standardními Ethernetovými čipy“, a tudíž klade velký důraz na to, že HW úpravy nejsou potřeba. Faktem však zůstává, že rychlost používaných čipů má výrazný vliv na rychlost přenosu a pro zlepšení vlastností se začínají implementovat FPGA do HW pro zvýšení výkonu, a to v důsledku znamená přesun z třídy B do C.

Powerlink není omezen použitím pouze určité topologie, ale přesto se tu vyskytuje jedno omezení co do počtu použitých zařízení v liniové topologii. To je logické a velmi důležité v případě použití 3. a 4. verze.

Co se týče výkonových parametrů, pro řízení 8 zařízení zapojených v topologii sběrnice přes rozbočovače, Powerlink V2 dosahuje doby jednoho cyklu kolem 300µs. V případě řízení 90 zařízení v několika sběrnicových větvích je doba jednoho cyklu kolem 2,4ms, pokud odpověď jednoho zařízení trvá 8µs. Pro porovnání můžeme uvést protokol EtherCAT, jenž dosahuje v obou případech méně jak desetiny doby cyklu protokolu Powerlink.

Další zajímavou vlastností Powerlink je, že podporuje protokol CANopen, který je jedním z nejrozšířenějších protokolů průmyslové automatizace.

2.11.2 ProfiNet

Protokol vyvinutý společností Siemens, který existuje ve 3 verzích, jež se liší vlastnostmi a použití

a) Verze 1 („Component Based Automation“) – Třída A
b) Verze 2 („(Soft) Real time“) – Třída B
c) Verze 3 („Isochronous real time“) – třída C

2. a 3. verze, tedy RT a IRT se pro zjednodušení sloučili a dnes je najdeme pod označení Profinet I/O, ale ne všechny zařízení podporují IRT.

2.11.2.1 Profinet CBA

Jedná se o koncept obsahující více než pouhý komunikační protokol. Vychází se z předpokladu, že celý automatizovaný celek lze rozdělit do samostatně pracujících částí, jež mohou být popsány svými vlastnostmi v jazyku XML a následně vloženy do databáze PCD (Profinet Comonent Description). Na základě této databáze se pak vytváří jednotlivá spojení.

2.11.2.2 Profinet I/O

Verze RT a IRT mohou být kombinovány v případě podpory mastera, použitím IRT přepínačů a dostatečnou šířkou pásma. Rozdíly mezi RT a IRT jsou vysvětleny níže.

2.11.2.2.1 RT

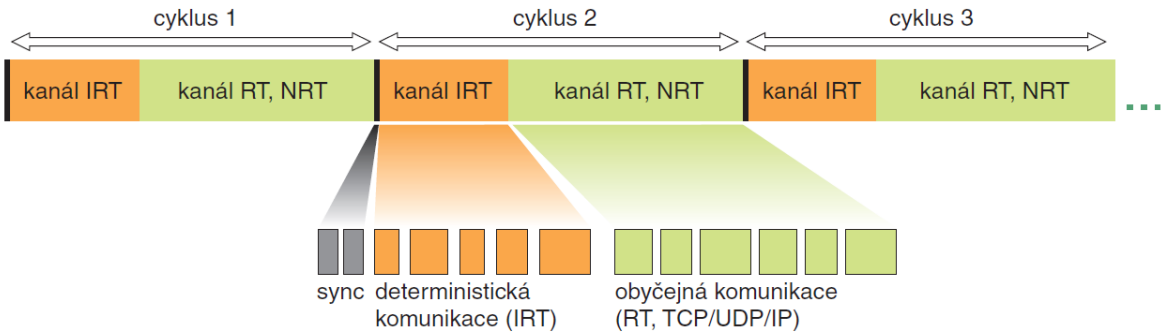
Verze 2 využívá upravený stack, jenž je rozdělený na dvě části. Kromě klasické TCP/UDP/IP je zde i část RT, zajišťující přenos časově kritických dat. Tento protokol měl sloužit výhradně pro PLC a měl mít podobné vlastnosti jako Profibus, ten je však rychlejší a podporuje synchronizaci.

Princip této verze spočívá v tom, že jednotlivé zařízení posílají cyklicky data stanici master. Toto zasílání dat probíhá bez jakékoliv synchronizace a spouštěč tohoto odesílání je vždy lokální časovač. Mezi nevýhodami můžeme zmínit, že se nejedná o deterministické řešení, neboť odesílání dat závisí na aktuálním stavu kapacity sítě, jelikož síť není rozdělena do časových slotů.

Mezi doporučení patří používání hvězdicové topologie a naopak se nedoporučuje liniová topologie.

2.11.2.2.2 IRT

Verze 3, tedy IRT používá speciální čipy (ASICS) ve všech připojených zařízeních a je zaměřena na řízení pohonů. K tomu používá rozdělování přenosového cyklu na izochronní (IRT) a neizochronní část (RT, NRT).



Obrázek 26 – Rozdělení přenosového cyklu na přenos IRT a RT (Zezulka a Hynčica, 2008)

Doba cyklu se pohybuje od 250µs do 4ms s nejistotou 1µs. Výkon sítě závisí velmi silně na zvolené topologii, která má jediné omezení. Tímto omezením je počet zařízení, které mohou být zapojeny v jedné linii, těchto může být maximálně 25.

2.11.2.3 Profinet Verze 2.3

Vylepšená verze IRT, která si klade za úkol zlepšit vlastnosti v liniových topologiích a snížit dobu jedno cyklu na 31,25 µs. Dále jsou zkráceny rámce, fragmentování rámců a také je změněna implementace MAC aby se snížila doba přenosu na přepínačích. Takový výkon může být zajištěn pouze za předpokladu, že všechny prvky v síti budou podporovat protokol 2.3 a budou vybaveny novým HW.

V současnosti se čeká na první dostupné produkty podporující tento protokol.

2.11.3 Ethenet/IP

Tento standard klade důraz na 100% kompatibilitu s protokolem Ethernet dle normy IEEE 802.3. To znamená, že využívá standardního TCP/UDP/IP stacku a samotný protokol se odlišuje až v aplikační vrstvě, jež je tvořena protokolem CIP (Common Industrial Protocol), který můžeme najít též v průmyslových protokolech typu fieldbus, konkrétně DeviceNet, ControlNet, CompoNet.

Ethernet/IP používá typ komunikace Producer / Consumer, jež je uplatňována v klasické přepínané síti 10Mbit/100Mbit/1Gbit, kde je doporučovaná topologie hvězda.

2.11.3.1 CIP

Jedná se o spojově orientovaný protokol, který reprezentuje každé zařízení v síti určitou skupinou objektů obsahující atributy (data), služby (příkazy) a specifikace jednotlivých funkcí (reakce). Každé zařízení používající protokol CIP musí obsahovat minimálně tyto objekty:

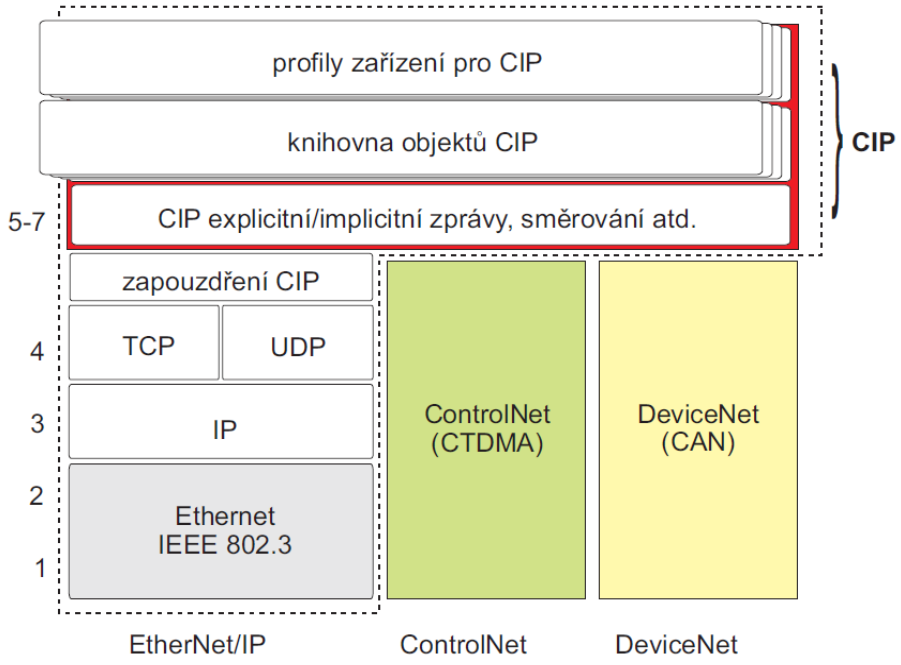
- *Identification object* – indetifikace zařízení
- *Age router object* – specifikace komunikace
- *Connection object* – správa připojení
- *Link object* – konfigurace síťového rozhraní

Další objekty jsou volitelné a mohou sloužit pro rozšíření funkcí zařízení. Všechny objekty dostupné v zařízení jsou sestaveny do elektronického popisu (Electronic Device Sheets – EDS), které obsahují základní informace ohledně možností konfigurace.

Od roku 2009 se můžeme setkat se CIP 3.0, který už podporuje protokol TPT dle normy IEEE 1588

2.11.3.2 Princip komunikace

Komunikaci můžeme rozdělit na explicitní a implicitní. Pro explicitní přenos, který primárně slouží pro přenos konfiguračních dat, je používána skupina protokolů TCP/IP a pro implicitní jsou používány protokoly UDP/IP, které se zas používají pro cyklický přenos dat.



Obrázek 27 – Struktura protokolu Ethernet/IP (Zezulka a Hynčica, 2008)

Každé spojení pomocí protokolu CIP je charakterizováno identifikátorem pro každý směr přenosu. Navíc implicitní zprávy mohou být odeslány nejenom jednotlivých účastníkům (unicast), ale je možné je poslat i více účastníkům najedou (multicast).

Ve specifikaci Ethernet/IP jsou definovány 3 třídy zařízení:

- Messaging class – zařízení, které podporují pouze explicitní přenos zpráv
- Adapter class – výstupní či vstupní periferie pracující v režimu reálného času. Nemohou zahajovat spojení.
- Scanner class – Zařízení zřizující spojení pro přenos dat v režimu reálného času.

2.11.3.3 Výkonové charakteristiky

Tento protokol není primárně určen pro aplikace v reálném času. Chybí zde dostatečná míra determinismu, který je možné zlepšit pomocí CIP Sync, ale i s tímto vylepšením to nemusí být dostatečné pro mnoho úloh.

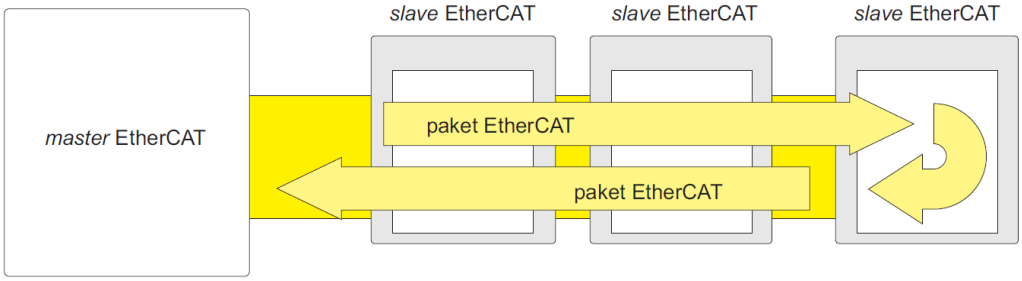
Co se týče doby jednoho cyklu, tak ten je velmi závislý na počtu, druhu zařízení a počtu současných CIP spojení. Například pro běžné zařízení (Scanner) odpovídající výkonnostnímu průměru při 16 spojení je doba cyklu 6,4ms a pro 32 spojení je to už 12,8ms. Pokud bychom použili vysoce výkonná zařízení (Scanner), tak lze zkrátit doby cyklu na polovinu. Pro zvýšení výkonu se používají čipy FPGA.

2.11.4 EtherCAT

Tento klade důraz a na vysoký výkon a velmi krátkou dobu cyklu. Aby mohl takového výkonu dosáhnout, tak používá upravený HW s alespoň dvěma Ethernetovými porty, díky kterým se zařízení chová jako Ethernetový rozbočovač a data mohou být zpracovávána během průchodu rámce. HW část zpracovávající Ethernetové rámce je založená na FPGA nebo na ASIC pro dosažení vysokému výkonu.

2.11.4.1 Princip komunikace a obecné vlastnosti

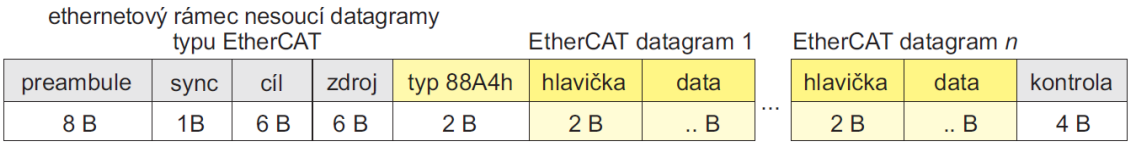
Komunikace probíhá v režimu master-slave, kdy rámec během jednoho cyklu projde všemi zařízeními slave, takže topologie je logický kruh. Rámec vždy vysílá stanice master, na který



Obrázek 28 – Průchod paketu EtherCAT (Zezulka a Hynčica, 2008)

ale nejsou kladeny vysoké HW požadavky (2 porty, FPGA,…). Fyzicky ale topologie může být prakticky jakákoliv a jediné omezení nalezneme pro liniovou topologii, kde se nemůže být připojeno více jak 65535 zařízení.

Data se posílají přímo v upraveném Ethernetovém rámci, který je směrován na základě MAC adres. Pokud je potřeba poslat rámec mimo lokální síť tak je směrován na adresu IP pomocí protokolu UDP. Pro přenos rámců je využíváno standardní 100Mbit sítě a plného duplexu.



Obrázek 29 – Ethernetový rámec EtherCAT (Zezulka a Hynčica, 2008)

Synchronizace je zajištěna vlastním protokolem, který je poměrně jednoduchý díky využití logického kruhu. Pro externí synchronizaci lze použít protokol PTP dle IEEE 1588.

Mezi výhody tohoto protokolu je nativní podpora protokolu CANopen (CANopen over EtherCAT-CoE). Dále lze volitelně přidat protokoly TCP/IP Ethernet over EtherCAT (EoE), který umožňuje chování jako uzel Ethernetu. Mezi další volitelný protokol patří Servodrive over EtherCAT (SoE) pro řízení pohonů a File Access over EtherCAT (FoE) pro přístup k souborům.

2.11.4.2 Výkonové parametry

EtherCAT dosahuje nejnižší doby cyklu ze všech výše uvedených protokolů a pyšní se extrémně nízkou dobou synchronizace (pod 1µs). Nejmenší uváděná doba cyklu je 11 µs pro 256 digitálních I/O uzlů a pro 100 digitálních podřízených, pokud každý bude mít 10 I/O, je uváděná doba 10 µs.

3 Návrh Ethernetového rozhraní pro řízení měniče

Část týkající se návrhu Ethernetového rozhraní je rozdělena na 3 části, které e zabývají propojením s měničem, návrhem desky plošného spoje a jeho komponent a v poslední řadě MCU od firmy Microchip, který převádí data z Ethernetového rozhraní na řídící signály pro jednotlivé tranzistory. MCU není schopné pouze generovat řídící signály, ale také měří napětí a proudy v měniči a detekuje chyby a je možné implementovat odeslání těchto dat do Eternetu.

3.1 Měnič IGBT

Jako vhodný měnič pro mnou navrženou desku plošného spoje BRD00001 byl vybrán IGBT měnič navržený studenty S. Tortigue a Y. Zongo (Tortigue a Zongo, 2004). Tento měnič dokáže řídit až 8 IGBT a díky různých dílčím nastavením ho lze použít pro velké spektrum aplikací.

3.1.1 Vstupy

Deska plošného spoje měniče obsahuje vstup pro napájení a také optické a digitální vstupy pro řídící signály.

Napájení je zajištěno pomocí 15V stabilizovaného zdroje, který se nalézá v levé části desky.

Řídící signály pro měnič je možné přivést samostatnými optickými kabely, anebo je možné použít 15 pinový konektor D-sub. Ten bude také sloužit pro připojení k desce BRD0001. Použité rozhodovací úrovně na desce měniče jsou 5V pro logickou „1“ a 0V pro logickou „0“ a tomu také bylo přizpůsobené komponenty na BRD00001.

3.1.2 Výstupy

IGBT měnič má 15 pinový D-sub výstup pro informování řídící jednotky o vzniklých chybách. Měnič dokáže informovat o chybě na každém IGBT tranzistoru zvlášť, ale také dohromady. Dále posílá informaci o tom, zdali jsou řídící pulzy blokovány.

Na tomto konektoru, který primárně slouží jako výstup je ale i přítomen jeden vstup, který slouží pro kvitaci chyby z řídící jednotky.

Rozhodovací hladiny jsou stejné jako pro vstupy, tedy 0V a 5V.

3.2 DPS Měření

Pro měření napětí a proudů v obvodu byla stejnými studenty vytvořena deska měření, která se připojuje do obvodu měniče (Tortigue a Zongo, 2004).

Tato deska je napájena -15/0/15V a obsahuje konektory pro připojení k měniči a dva 15 pinové konektory D-sub, které slouží připojení k měřícímu zařízení. Jeden z těchto konektorů je připojen přímo k výstupu z LEM a ten druhý má předřazené komponenty pro úpravu signálu před odeslání k měřícímu zařízení. Výstupem je proud -20mA – 20mA.

Pro připojení k BRD00001 je použit výstup bez dodatečného nastavení, protože úprava signálu pro AD převodník je zajištěna na straně BRD00001.

3.3 DPS BRD00001

DPS může být rozděleno na jednotlivé části, které mají na starosti jednotlivé úlohy pro vytvoření komplexního návrhu. Celá DPS je napájena 9V pomocí konektoru J1. Následující část je zpracována na základě datasheetů jednotlivých komponent.

3.3.1 Zdrojová část

Na DPS BRD0001 je celkem 5 zdrojů, které zajišťují činnost všech komponent. Požadovaná stabilizovaná napětí pro komponenty jsou 5V, 3,3V a -5V. DPS je chráněna proti přepětí a zkratu zenerovou diodou a pojistkou 1A.

Jako zdroj 5V stabilizovaného napětí byl použit zdroj UA7805CKCS, který je napájen z konektoru 9V.

Jako zdroj 3,3V byl vybrán obvod LM317, který má v obvodu zpětné vazby operační zesilovač MCP6H01 pro minimalizování napěťové závislosti na zatížení obvodu.

Pro vytvoření -5V je celkem použito 3 dílčích zdrojů. Obvod LM2575T je spínaný obvod generující 5V z 9V a byl použit, protože jsem se obával přílišného zatížení obvodu UA7805CKCS. Pro invertování napětí byl použit obvod MC33063AD, který vytváří -9V, které jsou posléze stabilizovány obvodem UA7905CKCS na -5V.

Zdroje UA7805CKCS, UA7905CKCS, LM2575T jsou v provedení TO-220 a jsou k nim přidány chladiče, které jsou upevněny k DPS.

3.3.2 Řídící část

Řídící obvod měniče IGBT pracujíc na hladině 5V, ale napěťová hladina výstupu MCU je 3,3V. Obvod CD40108BPW zajišťuje převod mezi těmito hladinami a odpory R47 a R50 slouží jako proudová ochrana. Konektor pro připojení řídících signálů je na desce označen jako J5.

3.3.3 Měřící část

DPS BRD00001 disponuje obvody měření z DSP měření měniče pro celkem 6 kanálů, které jsou přivedeny ke konektoru J4 a mohou být měřeny AD převodníkem MCU. Vzhledem k tomu, že výstup z obvodu LEM nabývá hodnot -20mA – 20mA, je nezbytné signál nejprve upravit. K tomu byl vybrán operační zesilovač MCP6H01 v invertujícím zapojení, který disponuje velice nízkým offsetem, aby bylo dosaženo velké přesnosti měření. Tento operační zesilovač měří úbytek na 100Ω odporu a převádí jej na hodnoty v rozmezí 0-2,5V pro AD převodník MCU.

Aby bylo možné převést napětí nabývající záporných i kladných hodnot v požadovaném rozsahu je zapotřebí napětí 1,25V. Toto napětí je získáno pomocí napěťové reference MCP1525T, která generuje 2,5V. Toto napětí je následně upraveno pomocí

operačních zesilovačů na 1,25V, které zajišťují samotný napěťový převod, ale i to, že napěťová reference MCP1525T nebude zatížena.

Protože napětí a proudy v měniči se velice rychle mění a MCU disponuje pouze jedním AD převodníkem, bylo nutné použít sample/hold obvod SMP04EZS, který zajistí, že hodnoty ze všech 6 kanálů budou zaznamenány ve stejný okamžik. Zavzorkování je řízeno výstupem z MCU.

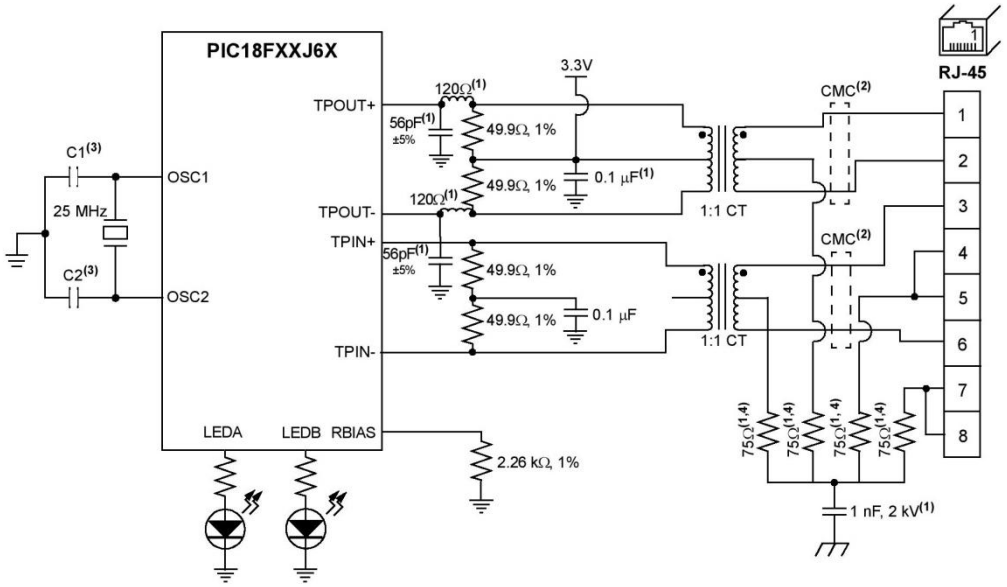
Pro ochranu operačních zesilovačů před přepětím ze strany měniče bylo použito dvojice zenerových diod.

3.3.4 Detekce chyb a blokování

V případě jakéhokoliv nestandardního chování na spínacích prvcích měniče je vyhodnocena chyba, která je přivedena přes konektor J3 do MCU. Detekce chyb je přítomna na každém spínacím prvku. Dále jsou přítomny piny, které indikují, jestli jsou nebo nejsou spínací prvky blokovány spínačem na DSP měniče a také je zde pin pro kvitaci chyb z MCU.

3.3.5 Ethernetové rozhraní

MCU PIC18F97J60 disponuje MAC a PHY, tedy vrstvy odpovídající spojové a fyzické vrstvě OSI modelu, a tak je možné s minimálním počtem součástek připojit toto MCU k Ethernetové síti.



Note 1: These components are installed for EMI reduction purposes. See **Section 19.1.5** for more information.
2: Recommended insertion point for Common-Mode Chokes (CMCs) if required for EMI reduction.
3: See **Section 3.3 "Crystal Oscillator/Ceramic Resonators (HS Modes)"** for recommended values.
4: Power over Ethernet applications require capacitors in series with these resistors.

Obrázek 30 – Elektrické schéma Ethernetového rozhraní PIC18F97J60 (MIcrochip.com, ©1998-2014)

Pro jednodušší implementaci Ethernetového rozhraní byl použit MagJack, cože je komponenta obsahující konektor RJ45, signálové oddělovací transformátory, rezistory a kondenzátor. Součástí MagJacku mohou být i LED diody signalizující stavy rozhraní Ethernet. Tyto LED diody je možné připojit k určitým portům MCU, které jsou ovládány přímo Ethernetovým modulem.

Pro správnou funkci Ethernetového modulu musí být splněny určité požadavky. Jedním z nich je krystal, anebo v mém případě oscilátor pracující na frekvenci 25MHz, připojení rezistoru 2,26kΩ k portu RBIAS a také zajistit napájení Ehernetového modulu, které je oddělené od napájení MCU.

3.3.6 Rozhraní pro MCU PIC18F97J60

Pro správnou funkci MCU je zapotřebí splnit požadavky na napájení, připojení nezbytných komponent a programovací rozhraní.

3.3.6.1 Napájení a doplňující komponenty

MCU by mělo být napájeno napětím v rozmezí 2,7-3,6V aby bylo dosaženo maximálního výkonu. MCU má celkem 5 vstupů pro napájení, které je nezbytné vybavit blokovacími kondenzátory do vzdálenosti 6mm od pouzdra mikroprocesoru.

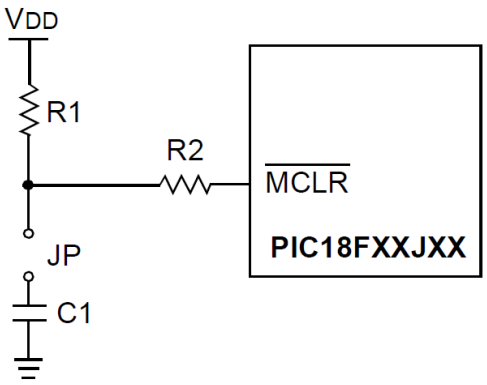
Jádro MCU potřebuje pro svůj chod napětí 2,5V, které může být generováno regulátorem přímo v MCU, anebo je možné připojit zdroj 2,5V k pinu V_{DDCORE}. Já jsem zvolil napájení pomocí interního regulátoru, a pro tento případ je nutné přidat blokovací kondenzátor k V_{DDCORE} a přivést napájení k pinu ENVREG.

Poslední pin, který je nezbytné napájet je $\overline{\mathrm{MCLR}}$. Problematika tohoto pinu je vysvětlena v následující části.

3.3.6.2 Programovací rozhraní

Pro naprogramování MCU je na BRD0001 umístěn konektor RJ11, který slouží pro připojení programátoru ICD3. Pro jeho správnou funkci jsou vyžadovány doplňující elektrické komponenty, jako jsou odpory a kondenzátor.

Programátor vyžaduje připojení pinu $\overline{\mathrm{MCLR}}$, PGC, PGD, V_{ss} a V_{DD}. $\overline{\mathrm{MCLR}}$ je pin sloužící pro dvě základní funkce: Reset a pro programování/debugování. Během normálního chodu MCU je doporučeno mít kondenzátor C1 připojený. Pro programování to je ale nevhodné kvůli přechodovým jevům a z toho důvodu je na DPS BRD0001 umístěn konektor JP1, stejně tak, jako je umístěn konektor JP v obrázku 31.



Obrázek 31 – Příklad zapojení pinu $\overline{\mathrm{MCLR}}$ (MIcrochip.com, ©1998-2014)

3.4 MCU Microchip PIC18F97J60

Jedná se o 8 bitový mikroprocesor od firmy Microchip, který disponuje Ethernetový modulem s MAC a PHY a tudíž je velmi snadná implementace Ethernetového rozhraní. Poměrně nízká cena těchto mikroprocesorů a jejich poměrně značný výkon představuje hlavní výhodu těchto mikroprocesorů.

Tento mikroprocesor má celkem 5 různých možností pro připojení oscilátoru, což umožňuje si vybrat přesně takový režim vyhovující maximálně dané aplikaci. Navíc PIC18F97J60 obsahuje interní 31kHz RC oscilátor, který nám dává další funkce, zejména pro zvýšení spolehlivosti. Maximální hodinová frekvence je 41.667MHz, které je dosaženo pomocí frekvenční násobičky.

3.5 Programové vybavení

Pro ověření vlastností MCU PIC18F97J60 je zapotřebí programového vybavení a určitých nástrojů, které budou popsány v následující části. Hlavním cílem je změřit rychlost odezvy MCU a určit tak nejmenší možný řídící cyklus, který může být použit pro kvalitní řízení měniče.

3.5.1 Programové vybavení MCU PIC18F97J60

Mikroprocesory od firmy Microchip se programují v jazyce C/C++ a využívá se zejména programovacího nástroje MPLAB. Já jsem použil konkrétně verzi MPLAB X 2.35 a kompilátor C18 ve verzi 3.47. Pro propojení MCU s počítačem pro jeho naprogramování byl použit programátor ICD3 s verzí firmwaru 01.36.10.

Programová výbava v MCU je založena na TCP/IP stacku od firmy Microchip, který zajišťuje základní Ethernetovou konektivitu. Pro ověření vlastností byla vytvořena funkce, která využívá TCP/IP stacku a na základě požadavků od řídícího počítače vykonává určité akce.

3.5.1.1 TCP/IP Stack od Microchip

Tento TCP/IP stack je volně použitelný jako základ pro nejrůznější aplikace a je volně stažitelný z Microchip.com (Microchip, [b.r.]}. Tento stack je rozdělen do jednotlivých vrstev odpovídající TCP/IP modelu a obsahuje implementaci základních protokolů a funkcí.

Hlavní vlastnosti TCP/IP stacku od Microchip

- Podporované protokoly: ARP, IP, ICMP, UDP, TCP, DHCP, SNMP, HTTP, FTP, TFTP
- Podpora socketů TCP a UDP
- SSL (Secure Sockets Layer)
- NetBIOS Name Service
- DNS (Domain Name Systém)
- Ethernet Device Discovery

Poslední verze TCP/IP stacku, která je k dispozici na Microchip.com byla uvolněna 19. 5. 2014, ale pro tuto diplomovou práci byla použita verze uvolněná 15. 6. 2013, tedy verze stacku 5.42.08.

3.5.1.2 Demo App

Microchip nedává k dispozici pouze TCP/IP stack, ale uvolňuje celou knihovnu ovladačů a ukázkových demo aplikací. V uvolněné verzi 15. 6. 2013. byla obsažena demo aplikace „Demo App“ pro PIC18F97J60, která obsahovala příklad aplikace SNTP klienta používající UDP pakety.

TCP/IP stack a Demo App musely být pro otestování vlastností MCU upraveny. Celý stack je konfigurovatelný komentováním, odkomentováním jednotlivých funkcí definovaných pomocí maker.

Demo App obsahuje HW podporu nejrůznějších vývojářských desek od Microchip (Explorer 18, PICDEM.net 2, …), kterou lze definovat v HW konfiguraci TCP/IP stacku a tím lze dosáhnout okamžité funkčnosti Ethernetu na daném HW bez dalšího programování. Vzhledem k tomu, že mnou navržená DPS BRD0001 je co se týče Ethernetový konektivity až na EEPROM totožná s PICDEM.net2, tak jsem využil její HW podpory. Pro správnou funkčnost jsem deaktivoval použití EEPROM a přemapoval jednotlivé piny využívané stackem.

Veškeré nastavení TCP/IP stacku bylo provedeno v následujích 4 hlavičkových souborech.

- HWP PICDN2_ETH97.h
- TCPIP ETH97.h
- TCPIPConfig.h
- HardwareProfile.h

Po HW nastavení byly deaktivovány všechny nevyužité funkce stacku pro dosažení maximálního výkonu. V TCPIP ETH97.h se aktivují a deaktivují jednotlivé funkce a zejména

protokoly např.: SMTP, DNS, FTP, SSL, atd. Pro otestování vlastností MCU byly aktivovány pouze dvě funkce a to: STACK_USE_BERKELEY_API a STACK_USE_ICMP

BERKELEY_API je použité jako základ funkce UDPServer a ICMP slouží pro snazší monitorování dostupnosti socketů v síti.

3.5.2 UDP Server

Tato funkce byla vytvořena pro ověření výkonových vlastností MCU PIC18F97J60. Jedná se server, který reaguje a odpovídá na požadavky klienta. Veškerá komunikace probíhá skrze Ethernet a využívá se protokolu UDP.

Kód zdrojového souboru UDPServer.c, který jev příloze E, je založen na Berkeley API a pro jeho vytvoření jsem se inspiroval ukázkovým demem BerkeleyUDPClient.c. Předpokladem pro použití UDPServer.c je povolení a inicializace Berkeley API a dostupnost zdrojových souborů UDP.c, ARP.c a Tick.c.

Základem je struktura switch, která zajišťuje vytvoření socketu, aktivování naslouchání definovaného portu, odesílání paketů a přijímaní paketů. Ve výchozím nastavení je aktivováno naslouchání na portu 6777 (UDP_CLIENT_PORT). Na základě hodnoty druhého byte příchozí zprávy je určen mód definující způsob komunikace s UDPClient.

3.5.2.1 Mód 3

Pokud MCU obdrží ve druhém byte číslo 3, tak začne s definovanou periodou „TIMEOUT“ odesílat uživatelská data UDP klientovi. Tento mód byl vytvořen pro otestování periodického odesílání dat z MCU. Ve skutečnosti by tento mód mohl být použit pro odeslání hodnoty střídy frekvenčnímu měniči a následně bychom mohli monitorovat obdržená data. Proto je také tento mód odpovídá funkci „DutyCycleAndReception“ v aplikaci UDPClient.

3.5.2.2 Mód 5

Pokud MCU obdrží ve druhém byte číslo 5, tak MCU očekává zasílání řídících pulzů pro jednotlivé spínací prvky, bez odesílání jakýchkoliv dat. Očekávané hodnoty jsou „0“ a „1“. Tento mód odpovídá funkci „SendControlPulses“ v aplikaci UDPClient.

3.5.2.3 Mód 7

Tento mód je založen na módu 5, to znamená, že po přijetí ve druhém byte číslo 7 jsou očekávány řídící pulzy s hodnotami „0“ a „1“. Po přijetí paketu s řídícími informace je navíc ihned odeslán paket s uživatelskými daty a poté MCU opět čeká na obdržení dalšího řídícího paketu. Tento mód odpovídá funkci „PulsesAndReception“ v aplikaci UDPClient.

3.5.3 UDP Client

Aplikace UDP Client byla vytvořena v jazyce Java v prostředí NetBeans IDE 8.0.2 a slouží pro komunikaci s MCU PIC18F97J60 a funkcí UDP Server popsanou výše. Zdrojem informací pro vytvoření této aplikace bylo fórum Stackoverflow.com (2011), (2012). Debugování této aplikace bylo prováděno za pomoci programů „Packet sender“ a „Wireshark“.

UDPClient obsahuje metody umožňující jednorázové odeslání paketu, periodické odesílání paketů a zobrazení příchozích paketů. Uživatel může použít pro komunikaci s UDPServer tyto metody:

- SendControlPulses
- DutyCycleAndReception
- PulsesAndReception
- Reception
- Reset

Metody „SendContro Pulses“ a „PulsesAndReception“ slouží pro vysílání řídících pulzů. Druhá jmenovaná metoda po každém odeslání paketů čeká na odpověď od MCU. Pokud paket nedorazí do 1,1*f (f = uživatelsky definované zpoždění) tak je odeslán další řídící paket. V případě že definované zpoždění bylo 10ms, tak doba mezi dvěma po sobě jdoucími pakety byla přibližně 11ms, neboť nějaký čas trvalo samotné odeslání paketu. Tyto metody odpovídají módům 5 a 7 ve funkci UDPServer.

Metoda „PulsesAndReception“ odpovídá módu 3 ve funkci UDPServer a slouží pro jednorázové nastavení MCU a periodické vyčítání dat.

Metoda „Reset“ slouží pro odeslání paketu o všech bytech nulové hodnoty pro uvedení funkce UDPServer do výchozího nastavení.

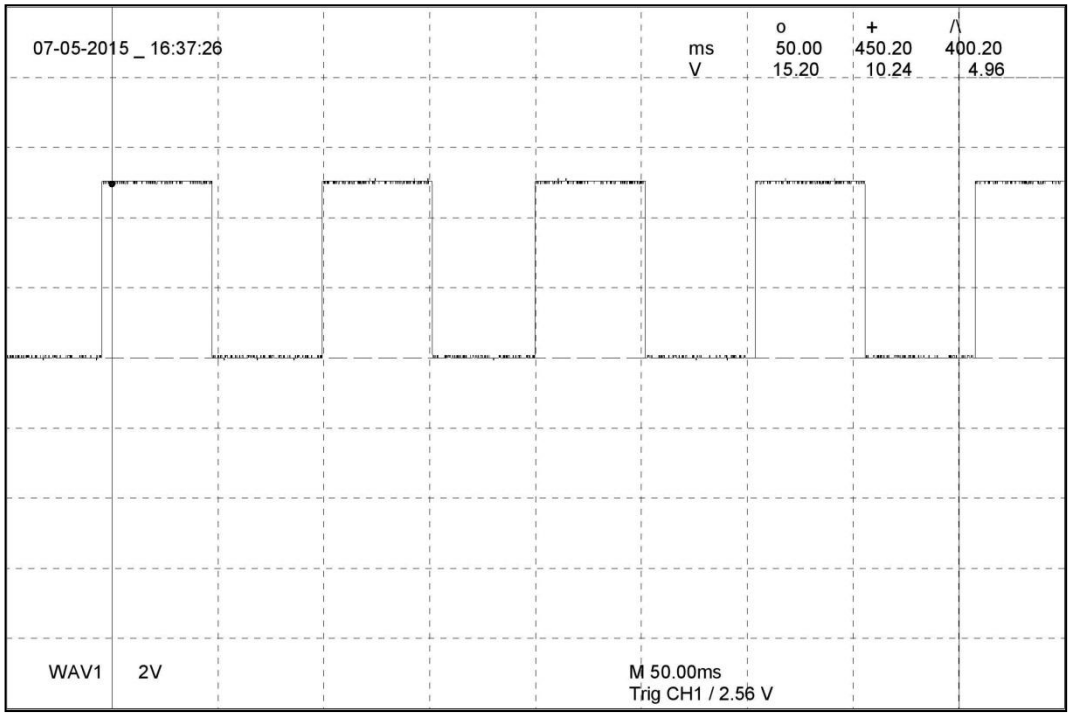
Metoda „Reception“ neodesílá žádny paket pouze aktivuje naslouchání na definovaném portu.

4 Měření parametrů komunikace PIC18F97J60

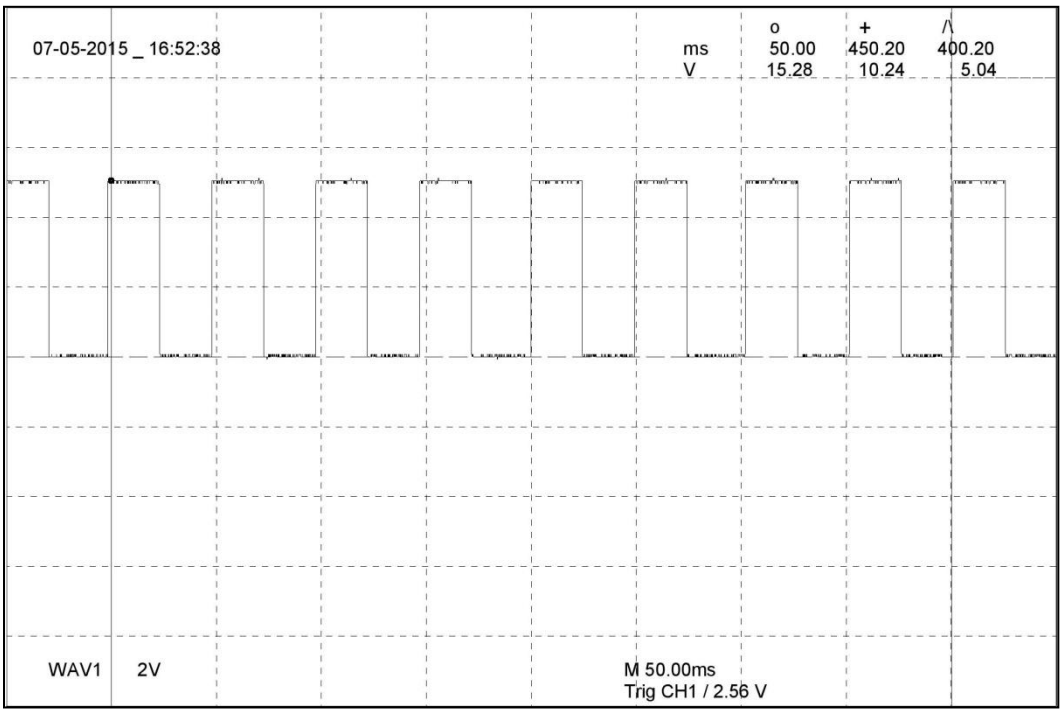
K měření bylo použito programové vybavení popsané v předchozí části, tedy UDP Server na straně MCU a UDP Client na straně řídícího počítače. Počítač byl připojen s MCU 2m dlouhým nekříženým Ethernetovým kabelem a byly nastaveny statické IP adresy: 169.254.1.1 pro MCU a 169.254.1.131 pro počítač. Bylo využíváno dvou metod v aplikaci UDP Client a to „SendControPulses“ a „PulsesAndReception“. Spínaný výstup z MCU byl přiveden na vstup osciloskopu, kde jsme ověřovali kvalitu spínání při různých frekvencích odesílání paketů. Během měření byla zachytávána veškerá komunikace programem Wireshark.

4.1 Naměřená data

4.1.1 Vysílání řídících pulzů a příjem dat – metoda „PulsesAndReception“



Obrázek 32 – metoda „PulsesAndReception“, zpoždění 50ms

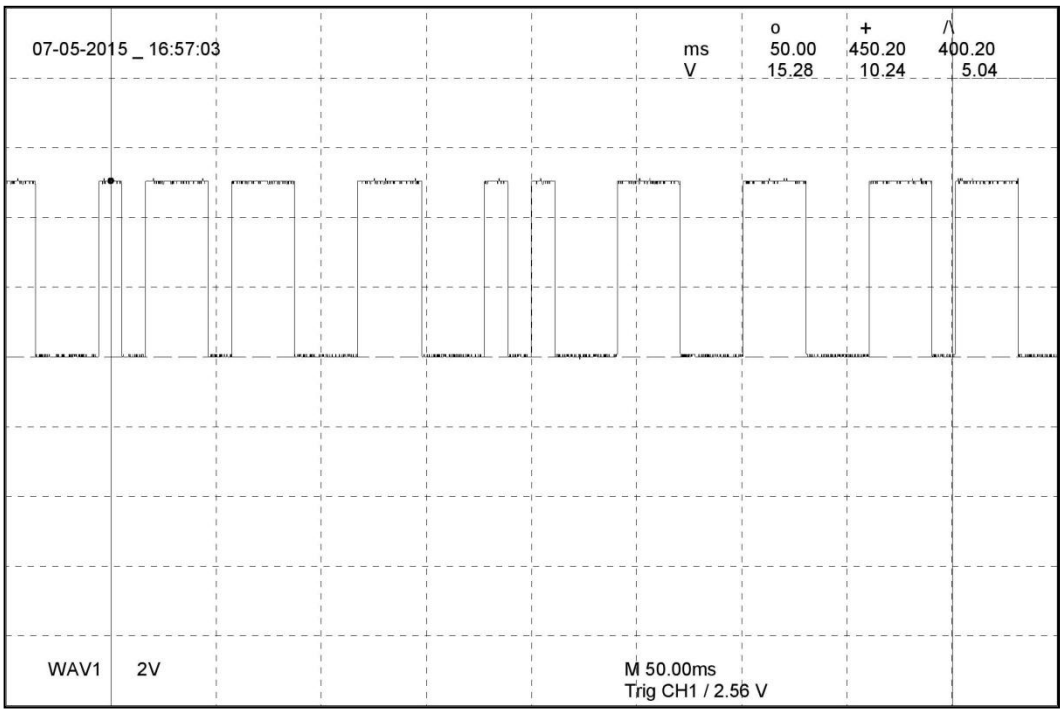


Obrázek 33 – metoda „PulsesAndReception“, zpoždění 25ms

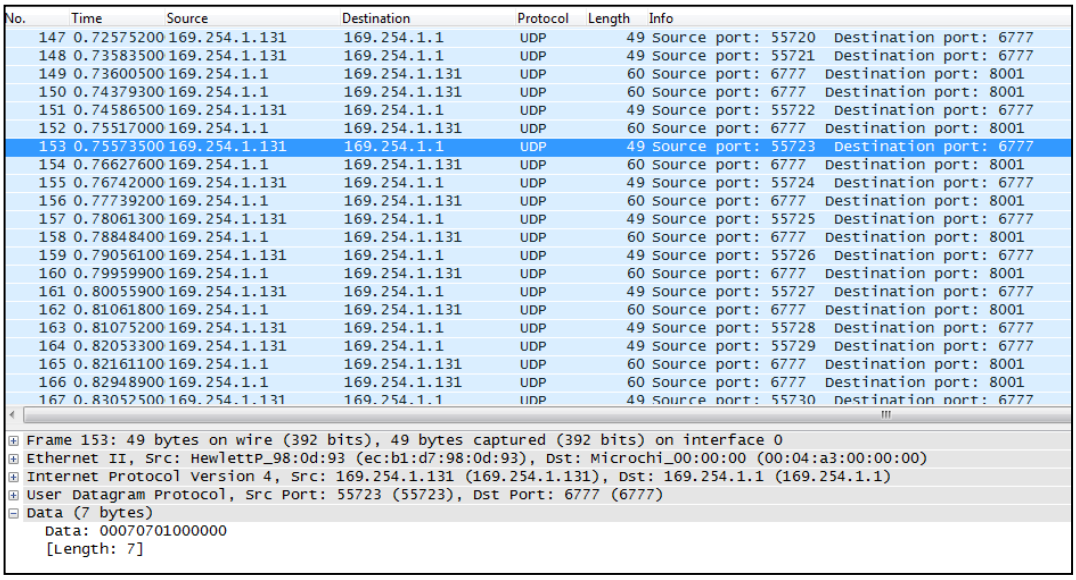
No.	Time	Source	Destination	Protocol	Length	Info
114	1.41876500	169.254.1.1	169.254.1.131	UDP	60	Source port: 6777 Destination port: 8001
115	1.43506500	169.254.1.131	169.254.1.1	UDP	49	Source port: 64041 Destination port: 6777
116	1.44352000	169.254.1.1	169.254.1.131	UDP	60	Source port: 6777 Destination port: 8001
117	1.46040000	169.254.1.131	169.254.1.1	UDP	49	Source port: 64042 Destination port: 6777
118	1.46815300	169.254.1.1	169.254.1.131	UDP	60	Source port: 6777 Destination port: 8001
119	1.48506400	169.254.1.131	169.254.1.1	UDP	49	Source port: 64043 Destination port: 6777
120	1.49301600	169.254.1.1	169.254.1.131	UDP	60	Source port: 6777 Destination port: 8001
121	1.51011900	169.254.1.131	169.254.1.1	UDP	49	Source port: 64044 Destination port: 6777
122	1.52111200	169.254.1.1	169.254.1.131	UDP	60	Source port: 6777 Destination port: 8001
123	1.53513100	169.254.1.131	169.254.1.1	UDP	49	Source port: 64045 Destination port: 6777
124	1.54585600	169.254.1.1	169.254.1.131	UDP	60	Source port: 6777 Destination port: 8001
125	1.56011200	169.254.1.131	169.254.1.1	UDP	49	Source port: 64046 Destination port: 6777
126	1.57028900	169.254.1.1	169.254.1.131	UDP	60	Source port: 6777 Destination port: 8001
127	1.58513100	169.254.1.131	169.254.1.1	UDP	49	Source port: 64047 Destination port: 6777
128	1.59521600	169.254.1.1	169.254.1.131	UDP	60	Source port: 6777 Destination port: 8001
129	1.61038400	169.254.1.131	169.254.1.1	UDP	49	Source port: 64048 Destination port: 6777
130	1.62005300	169.254.1.1	169.254.1.131	UDP	60	Source port: 6777 Destination port: 8001
131	1.63534000	169.254.1.131	169.254.1.1	UDP	49	Source port: 64049 Destination port: 6777

```
⊞ Frame 126: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: Microchi_00:00:00 (00:04:a3:00:00:00), Dst: HewlettP_98:0d:93 (ec:b1:d7:98:0d:93)
⊞ Internet Protocol Version 4, Src: 169.254.1.1 (169.254.1.1), Dst: 169.254.1.131 (169.254.1.131)
⊞ User Datagram Protocol, Src Port: 6777 (6777), Dst Port: 8001 (8001)
⊟ Data (7 bytes)
    Data: 00010707000000
    [Length: 7]
```

Obrázek 34 – metoda „PulsesAndReception“, zpoždění 25ms

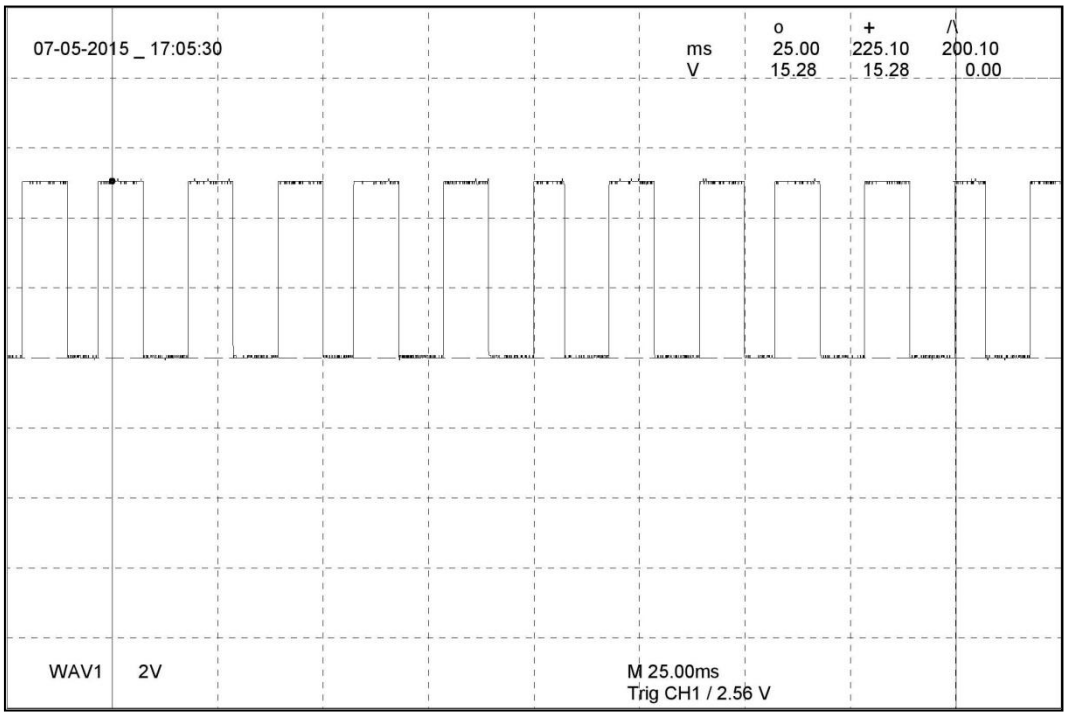


Obrázek 35 – metoda „PulsesAndReception“, zpoždění 10ms

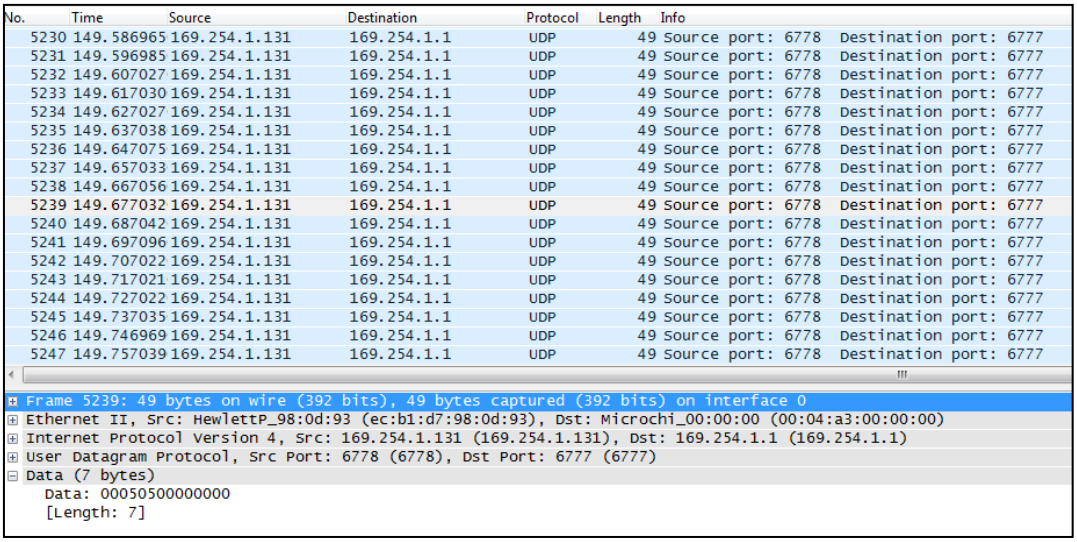


Obrázek 36 – metoda „PulsesAndReception“, zpoždění 10ms

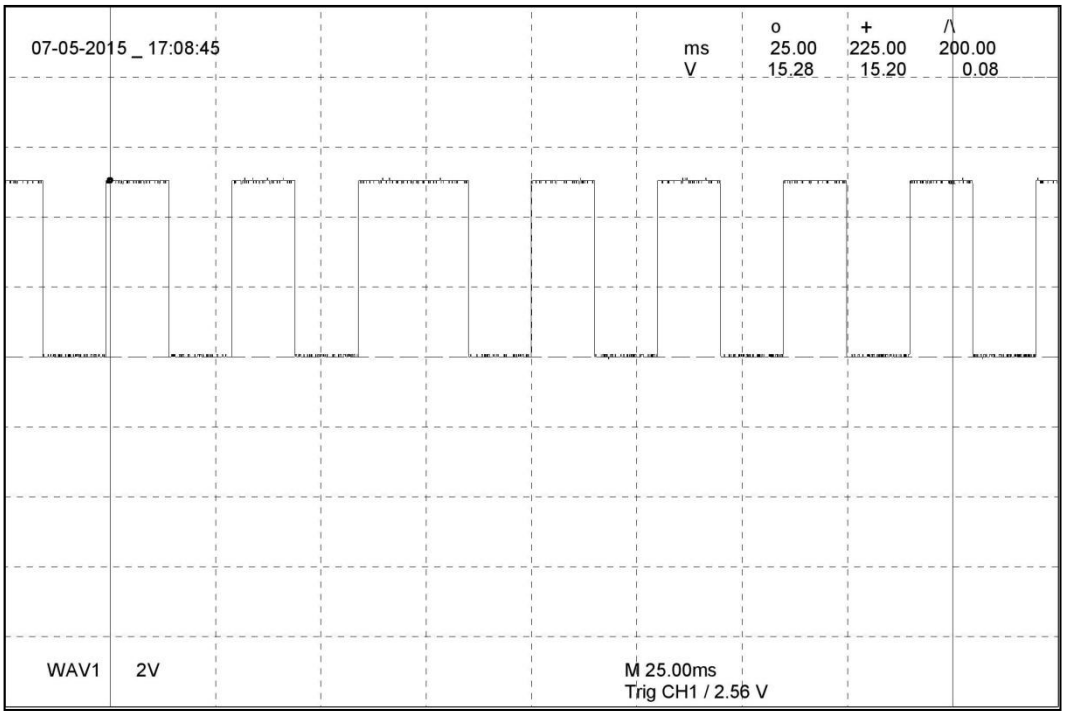
4.1.2 Pouze vysílání řídících pulzů – metoda „SendControPulses“



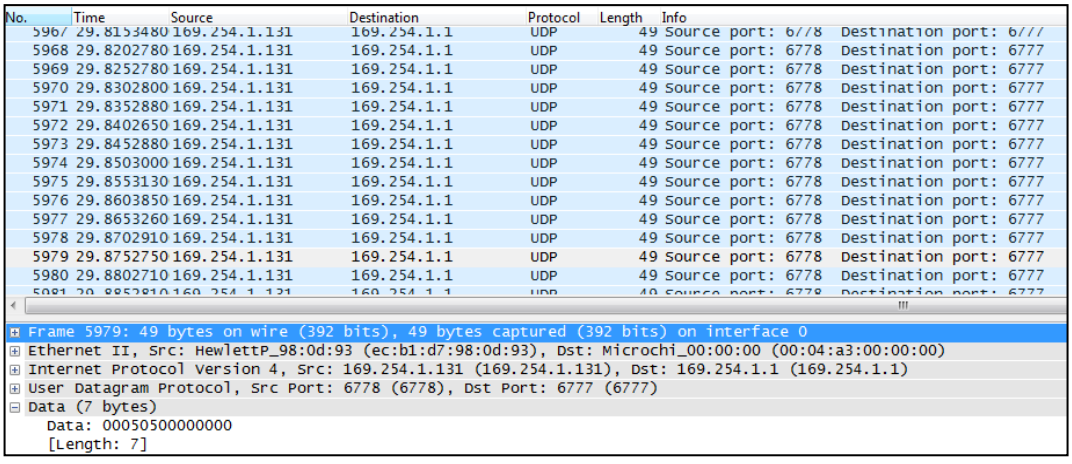
Obrázek 37 – metoda „SendControPulses“, zpoždění 10ms



Obrázek 38 – metoda „SendControPulses“, zpoždění 10ms



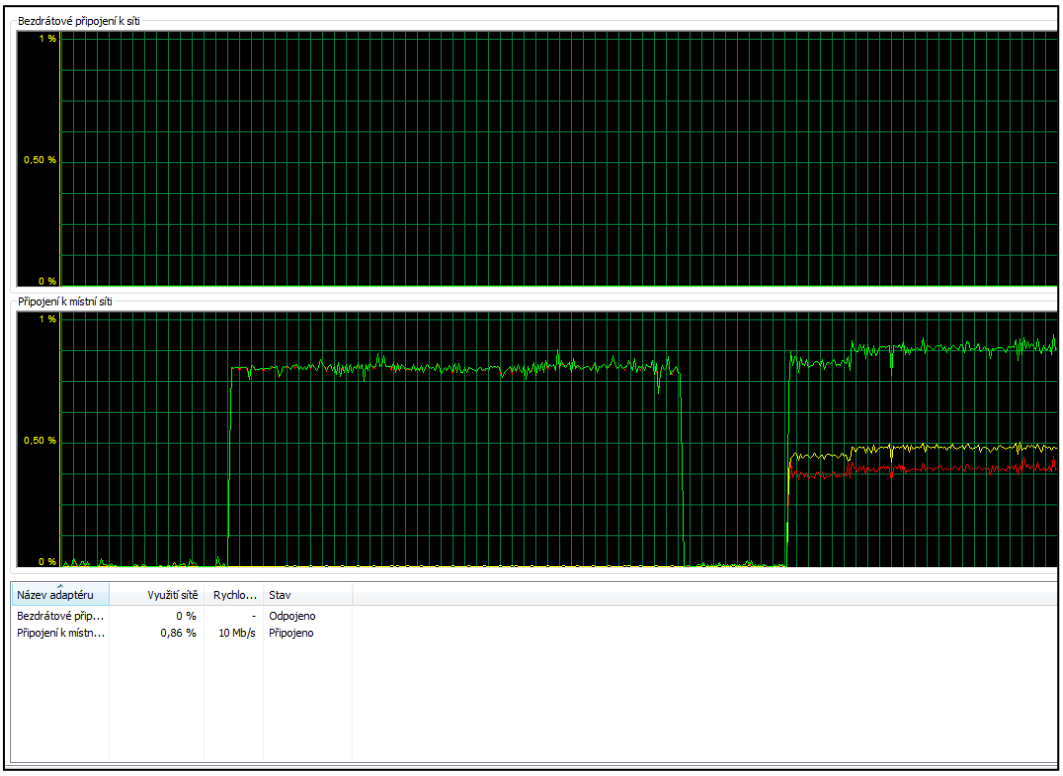
Obrázek 40 – metoda „SendControPulses“, zpoždění 5ms



Obrázek 39 – metoda „SendControPulses“, zpoždění 5ms

4.1.3 Zatížení sítě

Měření zatížení sítě bylo změřeno aplikací „Správce úloh“ systému Windows. První část odpovídá metodě „SendControPulses“ při zpoždění 5ms a druhá část „PulsesAndReception“ při zpoždění 10ms.



Obrázek 41 – Graf vytížení sítě

5 Závěr

V této práci bylo vytvořeno Ethernetové rozhraní umožňující řízení měniče, jehož základem je MCU PIC18F97J60. Byl použit volně dostupný TCP/IP stack a Demo App od firmy Microchip a na tomto základě byla vytvořena funkce UDPServer sloužící pro komunikaci s aplikací UDPClient a tím bylo dosaženo kompletního řešení pro řízení měniče. V poslední části byly změřeny vlastnosti Ethernetového rozhraní MCU PIC18F97J60 a z těchto hodnot se dají usoudit limity tohoto řešení.

Doba trvání jednoho cyklu, tedy čas mezi odeslaným paketem a odpovědí na tento paket, je spolu s kvalitní synchronizaci nejdůležitější faktor určující pro použitelnost daného řešení. Co se týče MCU PIC18F97J60 tak bylo měřeními zjištěno. Že nejmenší doba cyklu, která ještě zajišťuje dostatečnou kvalitu řízení, to znamená, že nedochází k velkému počtu chyb je cca 25ms pro příjem i vysílání a 10ms pouze pro vysílání. V případě vyšších frekvencí zasílání paketů už PIC18F97J60 není schopen odpovídat dostatečně rychle a dochází ke ztrátě paketů a chybám ve spínání, jak je zřejmé z naměřených grafů.

Na základě změřených dat nám tedy vychází, že maximální možná spínací frekvence, pokud bude měnič řízen přímo z řídícího počítače, je 10Hz. To je frekvence, která je zcela nedostačující. Vzhledem ke zkušenostem s 8bitovými mikroprocesory od Microchip vím, že je možné dosáhnout spínací frekvence PWM s definovaným zatěžovatelem kolem 1kHz. Nedomníval jsem se, že Ethernetový modul a jeho programová výbava bude tolik zatěžovat MCU natolik, že se sníží spínací frekvence na tak nízkou úroveň. Z grafu zatížeností sítě, kde je možné odečíst hodnotu 0,87% je zřejmé, že není limitující 10Mbit Ethernetové rozhraní ale samotná rychlost MCU PIC18F97J60.

Z těchto měření je zřejmé, že rychlost přenosu a tedy použitelnost pro aplikace reálného času je velice silně závislá na použitém HW. Ze studia používaných Ethernetových protokolů jako jsou Powerlink nebo EtherNet/IP využívající standardní strukturu a HW jsem se domníval, že dosahované časy se budou pohybovat v řádech desítek či stovek µs.

Pro dosažení požadovaných vlastností pro řízení měniče je tedy třeba zvolit mnohem výkonnější 32bitový MCU, pracující na vyšší frekvenci nebo využít FPGA, což je prvek, který se ve velkém využívá v HW využívající protokol EtherCAT, který dosahuje nejnižší doby cyklu. Druhá možnost, která se nabízí pro zachování stávajícího HW, je posílat skrze Ethernet pouze změny například zatěžovatele a nechat generování PWM čistě na MCU.

Tyto výsledky však neznamenají, že tento mikroprocesor nemůže být součástí průmyslových řešení. V průmyslových aplikacích, kde je kladen důraz na uniformnost řešení zcela určitě najdeme i prvky, které nemusí odesílat a přijímat data v reálném čase, ale musí být schopny komunikovat daným protokolem nebo musí podporovat protokol PTP. To jsou naopak funkce, které PIC18F97J60 může zajistit a nižší výkon naopak vyváží velice příznivou cenou. Mezi další výhody také patří právě možná podpora protokolu PTP, který vyžaduje

přístup k MII pro zapisování času a minimalizovaní časových nejistot vzniklých procházení dat jednotlivými vrstvami.

Literatura

[1] ZEZULKA, František a Ondřej HYNČICA: *Průmyslový Ethernet I: Historický úvod*. Automa, 2007, roč. 13, č. 1, s. 41‑43

[2] ZEZULKA, František a Ondřej HYNČICA: *Průmyslový Ethernet II: Referenční model ISO/OSI*. Automa, 2007, roč. 13, č. 3, s. 86–90.

[3] ZEZULKA, František a Ondřej HYNČICA: *Průmyslový Ethernet III: Fyzické provedení sítě Ethernet*. Automa, 2007, roč. 13, č. 6, s. 40–44.

[4] ZEZULKA, František a Ondřej HYNČICA: *Průmyslový Ethernet III: Fyzické provedení sítě Ethernet*. Automa, 2007, roč. 13, č. 6, s. 40–44.

[5] ZEZULKA, František a Ondřej HYNČICA: *Průmyslový Ethernet V: Bezpečná komunikace po Ethernetu*. Automa, 2007, roč. 13, č. 12, s. 58–61.

[6] ZEZULKA, František a Ondřej HYNČICA: *Průmyslový Ethernet VI: Informační bezpečnost*. Automa, 2008, roč. 14, č. 1, s. 58–62.

[7] ZEZULKA, František a Ondřej HYNČICA: *Průmyslový Ethernet VII: Přehled současných standardů*. Automa, 2008, roč. 14, č. 2, s. 26–29.

[8] ZEZULKA, František a Ondřej HYNČICA: *Průmyslový Ethernet VIII: Ethernet Powerlink, Profinet*. Automa, 2008, roč. 14, č. 5, s. 62–66.

[9] ZEZULKA, František a Ondřej HYNČICA: *Průmyslový Ethernet IX: Ethernet/IP, EtherCAT*. Automa, 2008, roč. 14, č. 10, s. 60–64.

[10] The 40th Anniversary of Ethernet. *IEEE-SA* [online]. [cit. 2015–05–02]. Dostupné z: http://standards.ieee.org/events/Ethernet/index.html

[11] FELSER, Max a Thilo SAUTER. *Standardization of Industrial Ethernet – the Next Battlefield?*. In: IEEE.[online]. 2004 [cit. 2015–05–02]. Dostupné z:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1377762&tag=1

[12] OSTERLOH, Heather. *TCP/IP: kompletní průvodce: použitelný pro veškeré operační systémy*. Praha: SoftPress, 2003, 512 s. ISBN 80–864–9734–8.

[13] PUŽMANOVÁ, Rita. *TCP/IP v kostce: kompletní průvodce : použitelný pro veškeré operační systémy*. 1. vyd. České Budějovice: Kopp, 2004, 607 s. ISBN 80–723–2236–2.

[14] GRYGÁREK, Petr. *Protokoly TCP/IP*. TECHNICAL UNIVERSITY OF OSTRAVA,. [online]. [cit. 2015–05–02]. Dostupné z:http://www.cs.vsb.cz/grygarek/PS/lect/tcpip.html

[15] Rozdíly.cz. *Iso–osi–model.gif.* [online]. [cit. 2015–05–02]. Dostupné z: http://www.rozdily.cz/w/index.php?title=Soubor:Iso–osi–model.gif&limit=20

[16] FRAZIER, JR., Howard M. *Media Independent Interfce*. [online]. In: IEEE. 1995 [cit. 2015–05–02]. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=485301&tag=1

[17] QoSSONICWALL, Inc. *Configuring SonicWALL* [online]. [cit. 2015–05–02]. Dostupné z: http://help.mysonicwall.com/sw/eng/216/ui2/29/config/qos.html

[18] BOUŠKA, Petr. *Cisco IOS 9 – Spanning Tree Protocol*. In: www.samuraj-cz.com [online]. [cit. 2015–05–02]. Dostupné z: http://www.samuraj–cz.com/clanek/cisco–ios–9–spanning–tree–protocol/

[19] Cisco.com, Inc. *Understanding Rapid Spanning Tree Protocol (802.1w)* [online]. 2006 [cit. 2015–05–02]. Dostupné z:http://www.cisco.com/c/en/us/support/docs/lan–switching/spanning–tree–protocol/24062–146.html

[20] BELDEN.com. *Product Analysis: HiPER Ring vs. RSTP: Redundancy Process with Hirshmann Switches* [online]. [cit. 2015–05–02]. Dostupné z:http://www.belden.com/docs/upload/hiper_ring_vs_rstp_wp.pdf

[21] PI - Profibus and Profinet International. *PROFIsafe System Description: Technology and Application* [online]. [cit. 2015–05–02]. Dostupné z:http://www.profibus.com/uploads/media/profisafe_system_description_v_2010_english.pdf

[22] EtherCAT.org. *Industrial Ethernet Technologies* [online]. [cit. 2015–05–02]. Dostupné z:http://www.ethercat.org/download/documents/Industrial_Ethernet_Technologies.pdf

[23] ABDULLAH, Mohammad Tarique. *Generic Standard IEC-61508*. In: Slideshare.net [online]. [cit. 2015–05–02]. Dostupné z:http://www.slideshare.net/tarique83/generic–standard–iec–61508

[24] LADKIN, Peter B. *An Overview of IEC 61508 on E/E/PE Functional Safety*. In: Causalis Limited [online]. 2008 [cit. 2015–05–02]. Dostupné z:http://www.causalis.com/90–publications/IEC61508FunctionalSafety.pdf

[25] HOSKE, Mark T. *Most used Ethernet protocols*. In: Control Engineering [online]. 2014 [cit. 2015–05–02]. Dostupné z: http://www.controleng.com/single–article/most–used–Ethernet–protocols/d73fc7d3aaed00e427e44c140e20fc0d.html

[26] HMS Industrial Networks. *Technology trends driving our business* [online]. [cit. 2015–05–02]. Dostupné z: http://www.hms–networks.com/about/technology–trends–driving–our–business

[27] IAONA E.V. *The IAONA Handbook for Network Security*. Magdeburg, 2006. 5th edition. Dostupné z: http://www.ininet.ch/vpi–initiative/download/IAONA–Security–Guide–15–draft.pdf

[28] IAONA E.V. *Industrial Ethernet Planning and Installation Guide*. Magdeburg, 2003. 4th edition. Dostupné z: http://www.pacontrol.com/download/Industrial–Ethernet–Planning–and–Installation–Guide.pdf

[29] GREENFIELD, Dave. *Industrial Ethernet: Safety over Ethernet*. In: Automation World [online]. [cit. 2015–05–02]. Dostupné z:http://www.automationworld.com/industrial–Ethernet–safety–over–Ethernet

[30] BOUŠKA, Petr. *Cisco IOS 10 – Rapid Spanning Tree Protocol*. In: www.samuraj-cz.com [online]. [cit. 2015–05–02]. Dostupné z: http://www.samuraj–cz.com/clanek/cisco–ios–10–rapid–spanning–tree–protocol/

[31] ZEZULKA, František a Ondřej HYNČICA: *Synchronizace v distribuovaných řídicích systémech: Precision Time Protocol (PTP) podle IEEE 1588*. Automa [online]. [cit. 2015–05–02]. Dostupné z:http://automa.cz/synchronizace–v–distribuovanych–ridicich–systemech:–precision–time–protocol–ptp–podle–ieee–1588–40557.html

[32] PAVELKA, Jiří. *Elektrické pohony*. Vyd. 1. Praha: Nakladatelství ČVUT, 2007, 222 s. ISBN 978–80–01–03588–7.

[33] PAVELKA, Jiří a Zdeněk ČEŘOVSKÝ. *Výkonová elektronika*. Vyd. 2., přeprac. Praha: ČVUT, Elektrotechnická fakulta, 2000. ISBN 80–010–2094–0.

[34] IEEE-SA. *IEEE 802.3™ - 2012 - IEEE Standard for Ethernet*. [online]. Editor Rajkumar Buyya. [cit. 2015–05–02]. Dostupné z: https://standards.ieee.org/about/get/802/802.3.html

[35] Microsoft.com . *Chapter 2 – Architectural Overview of the TCP/IP Protocol Suite: Figure 2–1 The architecture of the TCP/IP protocol suite* [online]. [cit. 2015–05–03]. Dostupné z: https://technet.microsoft.com/en–us/library/Bb726993.caop0201_big(l=en–us).gif

[36] Microchip.com: *PIC18F97J60* [online]. [cit. 2015-05-08]. Dostupné z: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026439

[37] Microchip.com: TCP*/IP Stack for PIC18, PIC24, dsPIC & PIC32* [online]. [cit. 2015-05-08]. Dostupné z: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE

[38] TORTIGUE S. a ZONGO Y.: *IGBT Driver.* Department of Electric Drives and Traction (K13114). Czech Technical University in Prague, 2004

[39] TORTIGUE S. a ZONGO Y.: *Measurement card*. Department of Electric Drives and Traction (K13114). Czech Technical University in Prague, 2004

[40] ZEZULKA, František: *Industrial Ethernet* [online]. [cit. 2015-05-08]. Dostupné z: http://webuser.hs-furtwangen.de/~spale/forall/PES/Vorlesung/ppt/zezulka.pdf

[41] Wikipedia: *Medium-dependent interface* [online]. [cit. 2015-05-08]. Dostupné z:http://en.wikipedia.org/wiki/Medium-dependent_interface#/media/File:Ethernet_MDI_to_MDIX.svg

[42] VOSS, Wilfried F.: *Industrial Ethernet and the Definition of Deterministic, Real-Time Performance*. In: http:bitstream24.com [online]. 2013. [cit. 2015-05-08]. Dostupné z: http://bitstream24.com/industrial-Ethernet-and-the-definition-of-deterministic-real-time-performance/

[43] Stackoverflow: *Need to send a UDP packet and receive a response in Java* [online]. 2012. [cit. 2015-05-08]. Dostupné z:

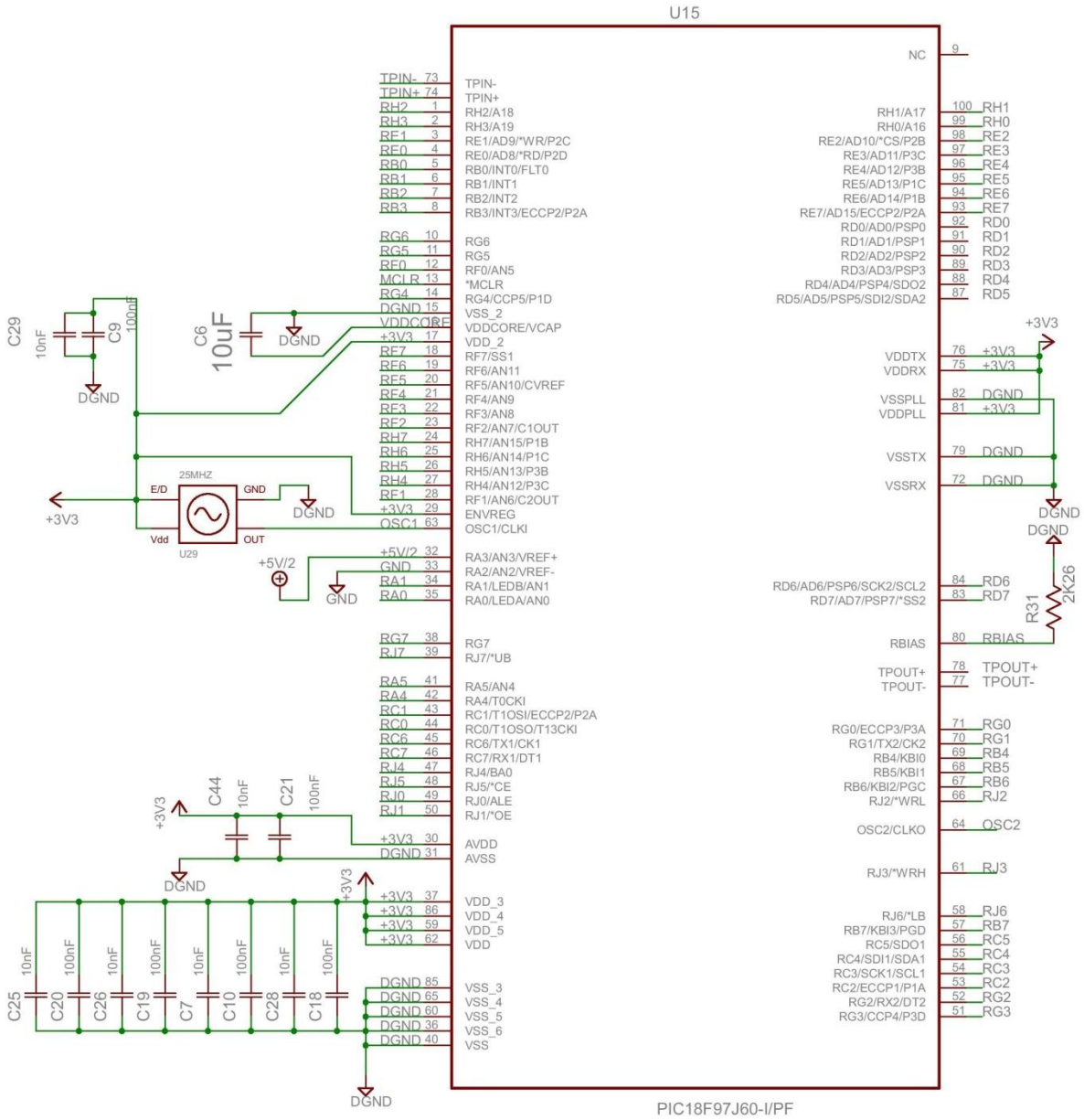
http://stackoverflow.com/questions/8562689/need-to-send-a-udp-packet-and-receive-a-response-in-java

[44] Stackoverflow: *sending and receiving UDP packets using Java?* [online]. 2011 [cit. 2015-05-08]. Dostupné z: http://stackoverflow.com/questions/10556829/sending-and-receiving-udp-packets-using-java

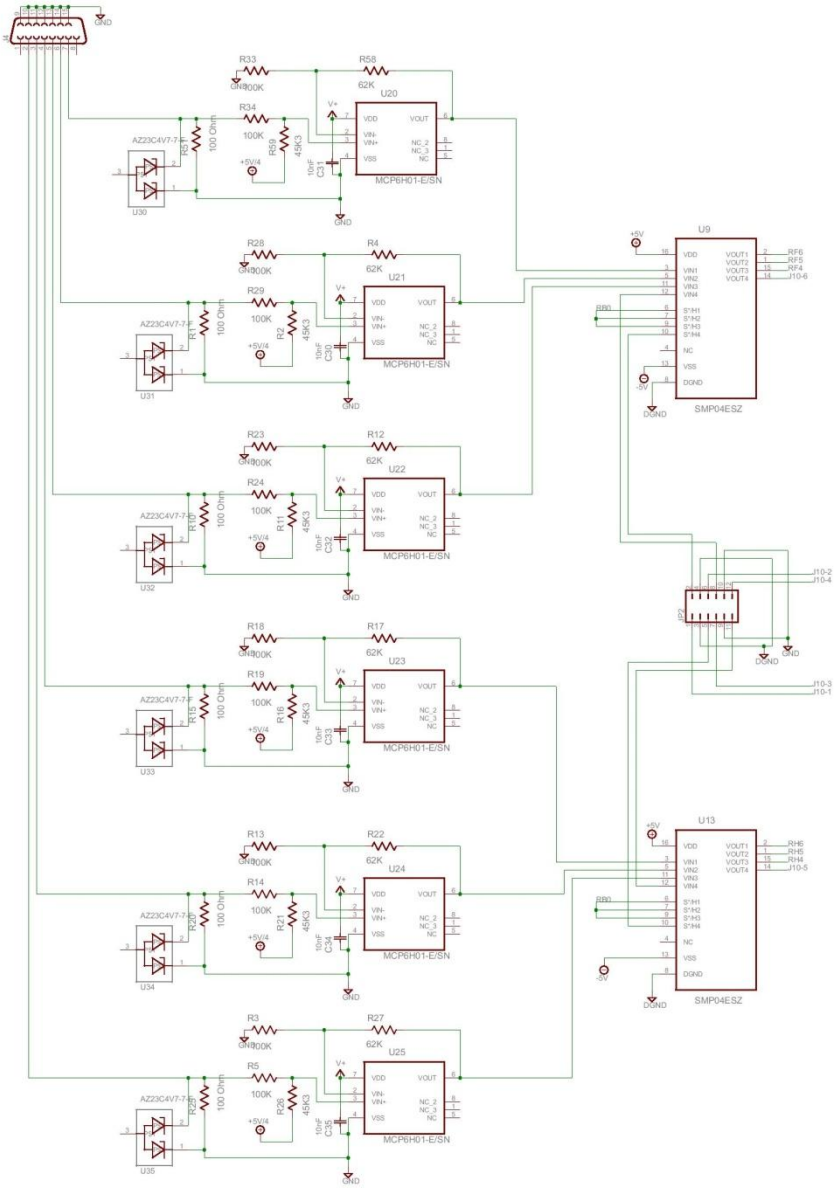
Přílohy

Příloha A – Výkresová dokumentace

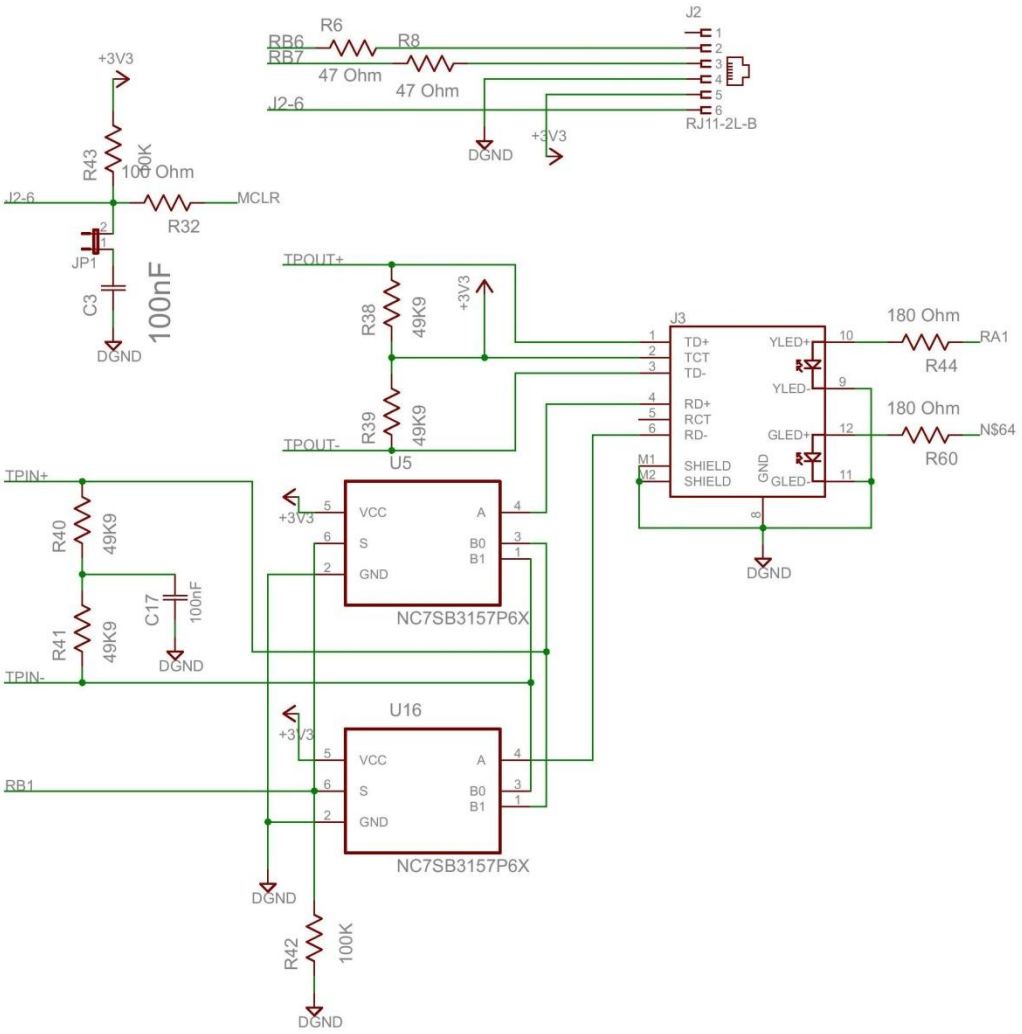
1/7



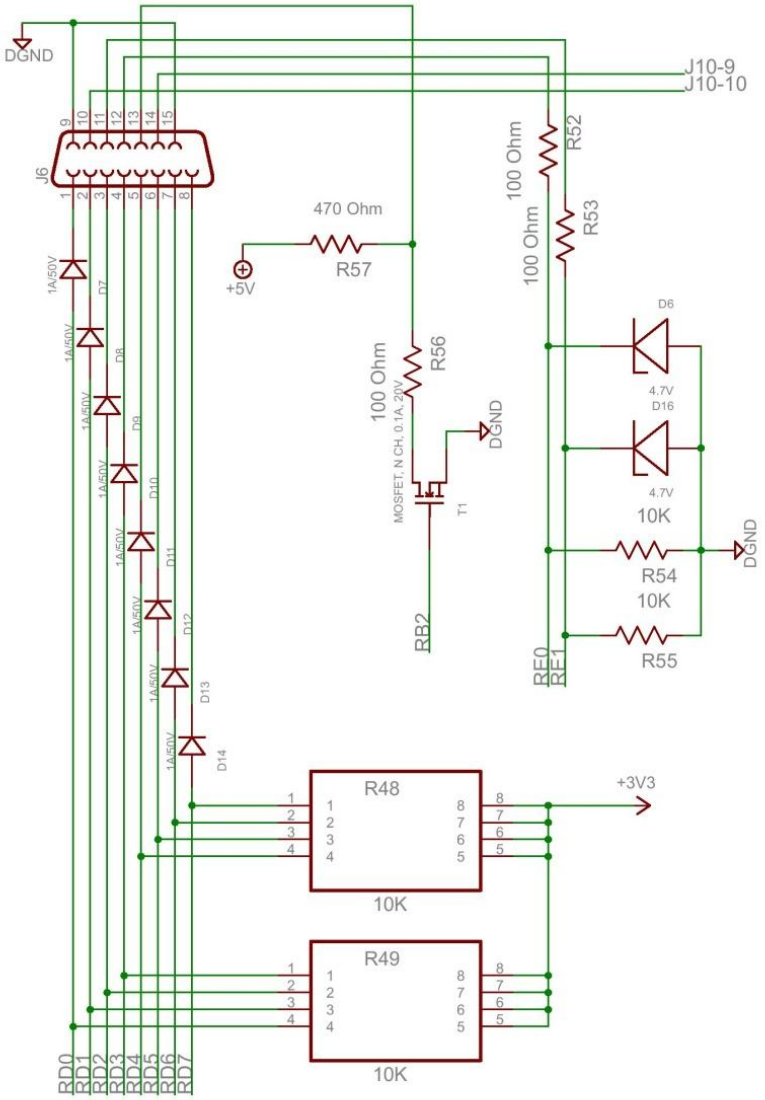
2/7

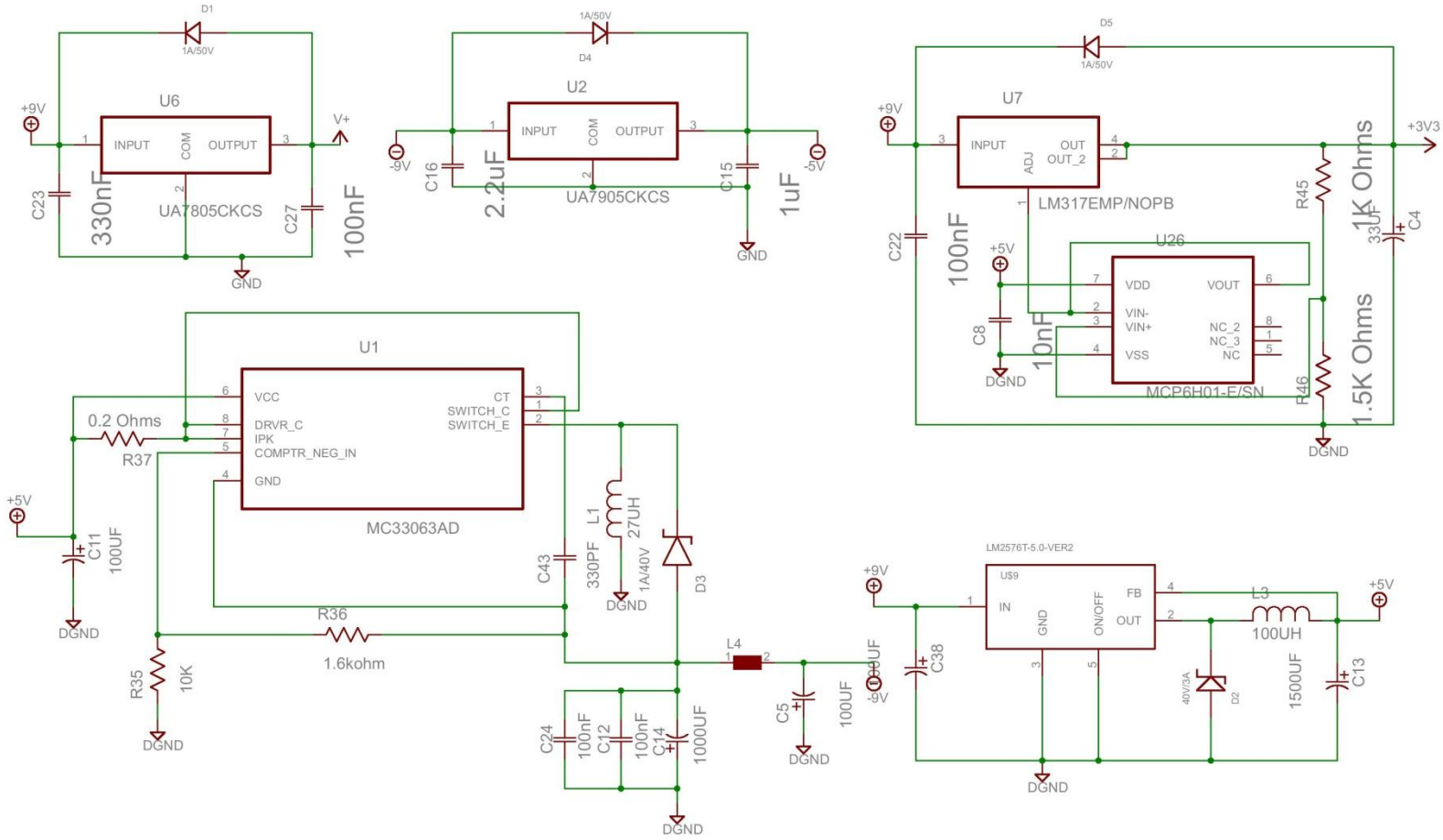

R33
R58
62K
100K
U20
R34
100K
100 Ohm
AZ23C4V7-7-F
R59
45K3
10nF
C31
MCP6H01-E/SN
U30
R28
R4
U21
R29
C30
U31
R23
R12
U22
R24
R11
C32
U32
R18
R17
U23
R19
R16
C33
U33
R13
R22
U24
R14
R21
C34
U34
R3
R27
U25
R5
R20
C35
U35
GND
+5V
-5V
U9
SMP04ESZ
U13
DGND
J10-2
J10-4
J10-3
J10-1
J10-6
J10-5

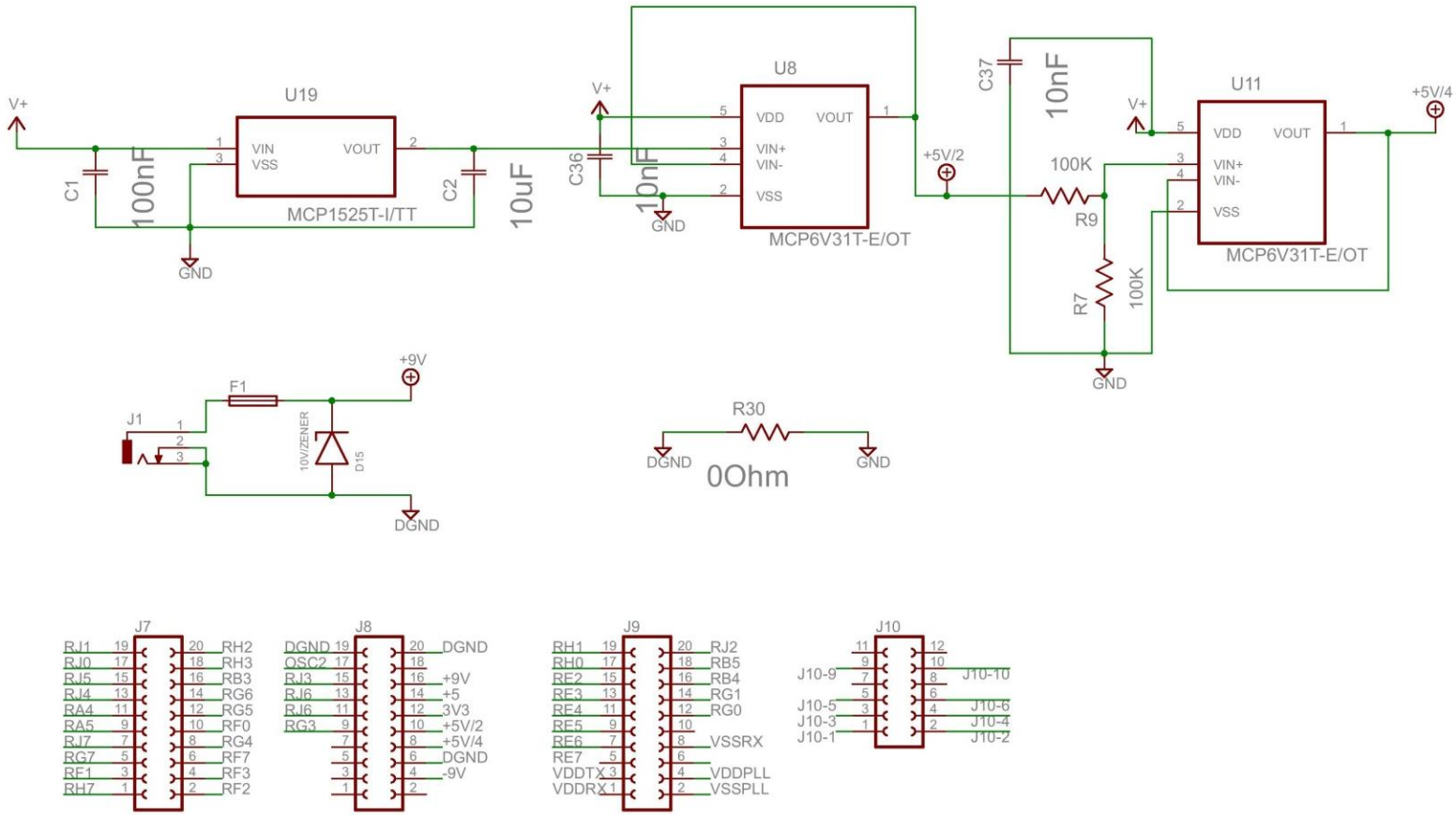
3/7


R6
RB6
RB7
47 Ohm
R8
47 Ohm
J2-6
J2
RJ11-2L-B
DGND
+3V3
+3V3
R43
100K
100 Ohm
J2-6
MCLR
R32
JP1
C3
100nF
DGND
TPOUT+
R38
49K9
+3V3
J3
TD+
TCT
TD-
RD+
RCT
RD-
SHIELD
SHIELD
YLED+
YLED-
GLED+
GLED-
GND
180 Ohm
RA1
R44
180 Ohm
N$64
R60
TPOUT-
R39
49K9
U5
TPIN+
R40
49K9
C17
100nF
R41
49K9
TPIN-
DGND
VCC
S
GND
A
B0
B1
NC7SB3157P6X
U16
RB1
DGND
R42
100K
DGND

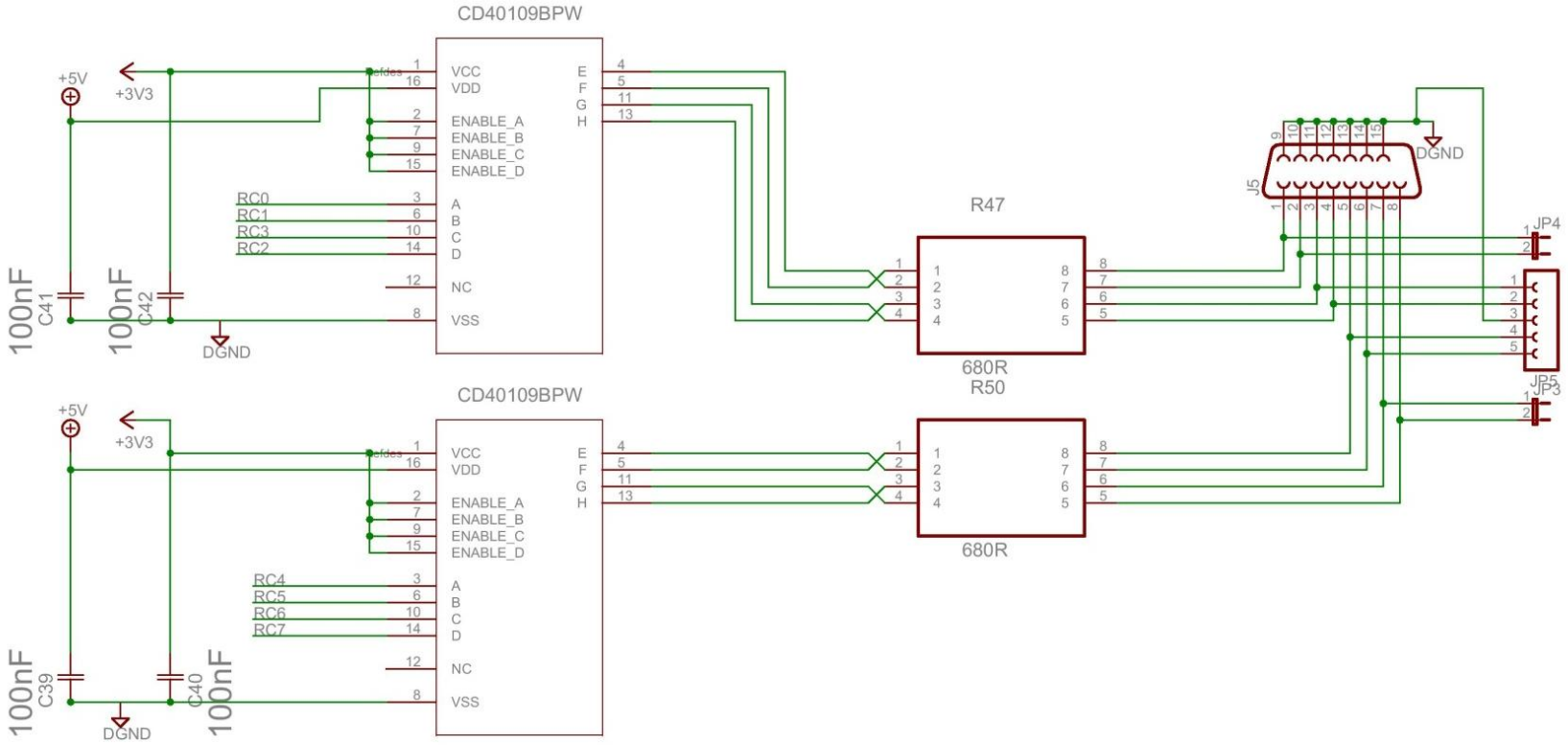
4/7


DGND
J10-9
J10-10
J6
470 Ohm
R57
+5V
100 Ohm
R52
100 Ohm
R53
100 Ohm
R56
MOSFET, N CH, 0.1A, 20V
T1
DGND
D6
4.7V
D16
4.7V
10K
R54
DGND
10K
R55
RB2
RE0
RE1
R48
10K
+3V3
R49
10K
RD0
RD1
RD2
RD3
RD4
RD5
RD6
RD7


D1
1A/50V
U6
+9V
INPUT
COM
OUTPUT
V+
C23
330nF
UA7805CKCS
C27
100nF
GND
D4
1A/50V
U2
-9V
C16
2.2uF
UA7905CKCS
C15
1uF
-5V
GND
D5
1A/50V
U7
+9V
INPUT
ADJ
OUT
OUT_2
LM317EMP/NOPB
+3V3
R45
1K Ohms
C4
33UF
C22
100nF
U26
+5V
C8
10nF
VDD
VOUT
VIN-
VIN+
VSS
NC_2
NC_3
NC
MCP6H01-E/SN
R46
1.5K Ohms
DGND
U1
VCC
DRVR_C
IPK
COMPTR_NEG_IN
GND
CT
SWITCH_C
SWITCH_E
MC33063AD
0.2 Ohms
R37
+5V
C11
100UF
DGND
C43
330PF
L1
27UH
1A/40V
D3
R36
1.6kohm
R35
10K
L4
C5
100UF
C24
100nF
C12
100nF
C14
1000UF
LM2576T-5.0-VER2
U$9
IN
GND
ON/OFF
FB
OUT
+9V
C38
100UF
L3
100UH
+5V
40V/3A
D2
1500UF
C13


V+
U19
VIN
VSS
VOUT
MCP1525T-I/TT
C1
100nF
C2
10uF
GND
U8
VDD
VOUT
VIN+
VIN-
VSS
MCP6V31T-E/OT
C36
10nF
C37
10nF
+5V/2
100K
R9
R7
100K
U11
+5V/4
+9V
F1
J1
10V/ZENER
D15
DGND
R30
0Ohm
J7
RJ1 RH2
RJ0 RH3
RJ5 RB3
RJ4 RG6
RA4 RG5
RA5 RF0
RJ7 RG4
RG7 RF7
RF1 RF3
RH7 RF2
J8
DGND DGND
OSC2 +9V
RJ3 +5
RJ6 3V3
RJ6 +5V/2
RG3 +5V/4
DGND
-9V
J9
RH1 RJ2
RH0 RB5
RE2 RB4
RE3 RG1
RE4 RG0
RE5
RE6 VSSRX
RE7
VDDTX VDDPLL
VDDRX VSSPLL
J10
J10-9 J10-10
J10-5 J10-6
J10-3 J10-4
J10-1 J10-2

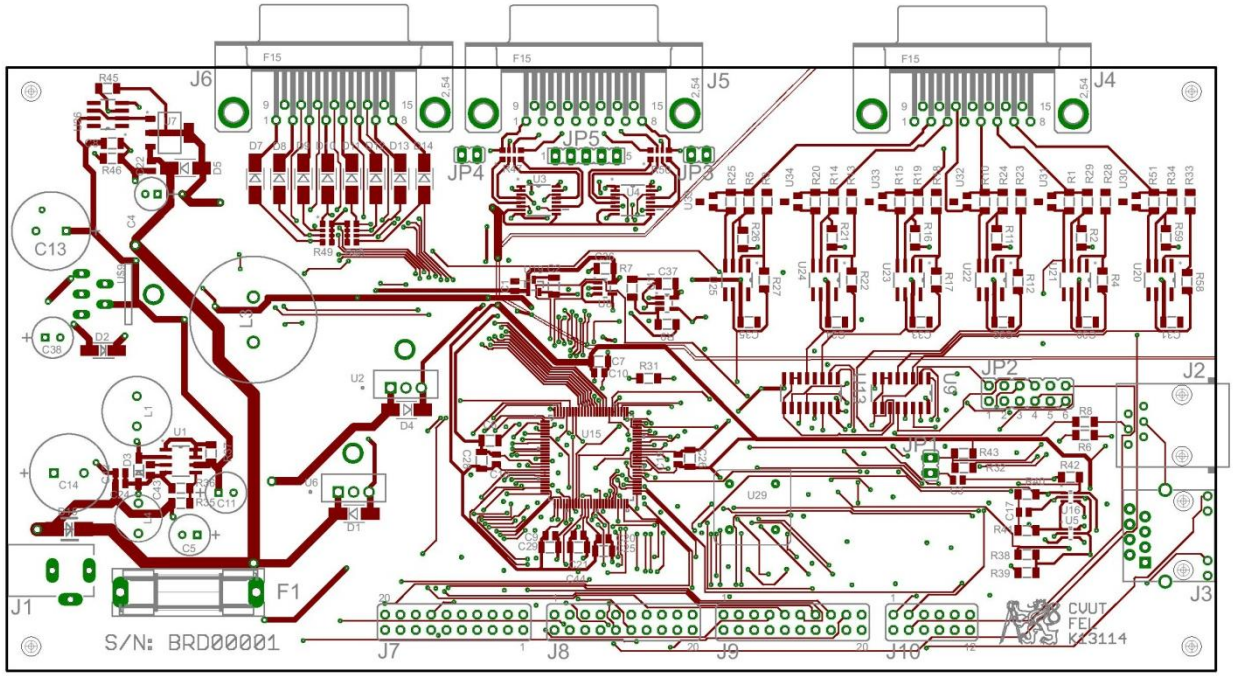
7/7


CD40109BPW
+5V
+3V3
VCC
VDD
ENABLE_A
ENABLE_B
ENABLE_C
ENABLE_D
A
B
C
D
NC
VSS
E
F
G
H
RC0
RC1
RC3
RC2
100nF
C41
100nF
C42
DGND
R47
680R
R50
J5
DGND
JP4
JP5
JP3
CD40109BPW
RC4
RC5
RC6
RC7
100nF
C39
C40
100nF
DGND
680R

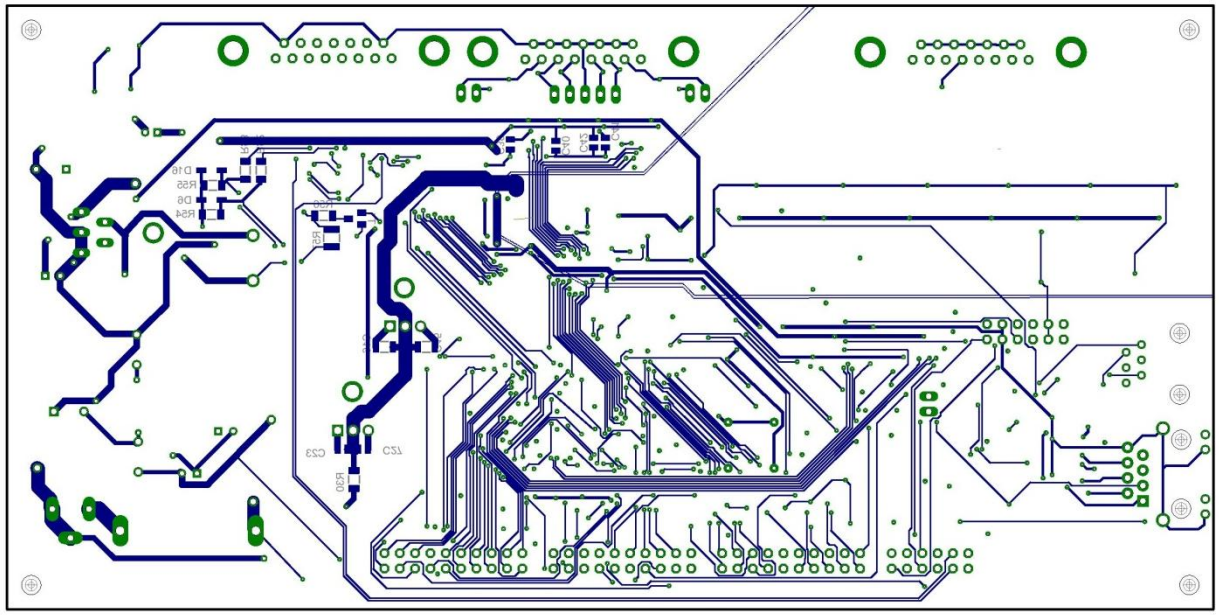
Příloha B – Pohledy DPS

Orientační pohledy DPS bez zobrazené vrstvy rozlité mědi.

TOP



BOT



Příloha C – Kusovník

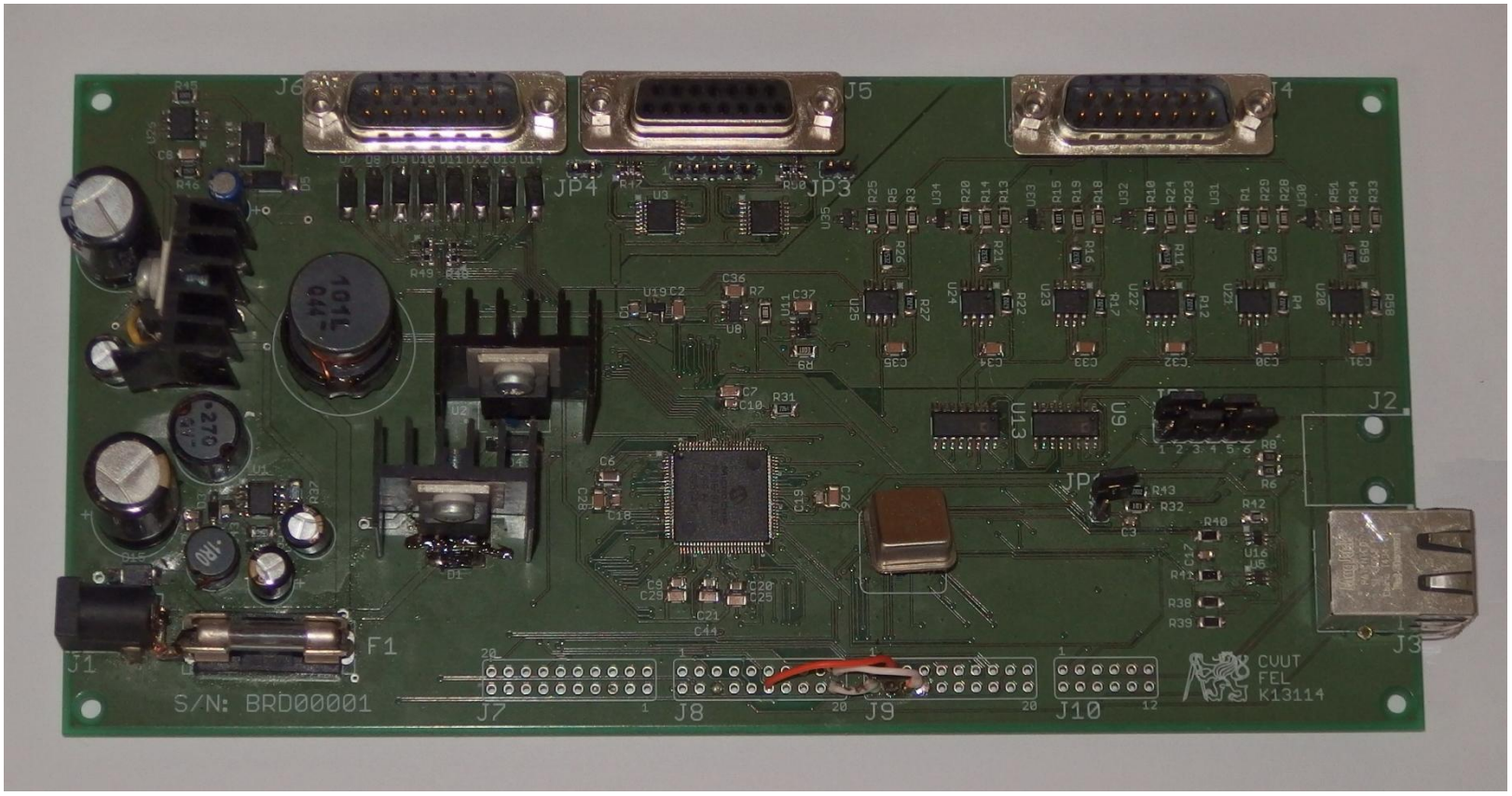
Název / Name	Typ / Type	Hodnota / Value	Pouzdro / Case	Počet / Amount	Pozice / Position	Obj. kód Farnell / Order code Farnell
Op amp	MCP6V31T–E		SOT–23	2	U11, U8	2113883
Voltage ref	MCP1525T–I		SOT–23	1	U19	9758500
MCU	PIC18F97J60–I/PF		TQFP	1	U15	1579614
Voltage reg	LM2576T–5.0/NOPB		TO–220	1	U$9	1564682
Diode Schottky	1N5819HW–7–F	1A, 40V	SOD–123	1	D3	1773475
Diode Zener	1SMB5925B	10V	SMB	1	D15	1894811
Multiplexer	NC7SB3157P6X2		SC–70	2	U16, U5	1607690
Oscilátor	ACHL–25.000MHZ–EK	25MHz	13.2mm x 13.2mm, Through Hole	1	U29	1652541
Diode Schottky	B340A–13–F	3A, 40V	DO–214AC	1	D2	1843668
Capacitor ceramic	C0805C104K5RACTU	100nF	0805	17	C1, C10, C12, C17, C18, C19, C20, C21, C22, C24, C27, C3, C39, C40, C41, C42, C9	1414664
Capacitor ceramic	C0805C334K3RACTU	330nF	0805	1	C23	1414691
Capacitor ceramic	08052A331JAT2A	330pF	0805	1	C43	1327688
Capacitor ceramic	GRM31MR71C225KA35L	2.2µF	1206	1	C16	9527753
Capacitor ceramic	C1206C105K3RACTU	1µF	1206	1	C15	9227865
Capacitor ceramic	C1206C106K4RACTU	10µF	1206	2	C2, C6	1463368

Capacitor ceramic	C1206C103K5RACTU	10nF	1206	15	C25, C26, C28, C29, C30, C31, C32, C33, C34, C35, C36, C37, C44, C7, C8	1414713
Capacitor electrolytic	ECA1CAM330X	33µF		1	C4	8767106
Capacitor electrolytic	EEUFM1E101	100µF		3	C11, C38, C5	1219466
Capacitor electrolytic	EEU–FM1E102	1000µF		1	C14	1219471
Capacitor electrolytic	EEUFR1E152	1500µF		1	C13	1800655
Connector	MJ–179PH	5A, 12V		1	J1	1737246
Diode	DO–214AC	1A, 30V	DO–214AC	11	D1, D10, D11, D12, D13, D14, D4, D5, D7, D8, D9	7277920
Connector	3–1634219–2	15WAY		2	J4, J6	5081981
Connector	3–1634223–2	15WAY		1	J5	5082067
Connector	pinhead	2, 54mm	1 row	11	JP1, JP3, JP4, JP5	1022247
Connector	pinhead	2, 54mm	2 row	6	JP2	2356134
Mosfet	SSM3K16FU	N Channel		1	T1	1714382
Inductor	ELC11D270F	27µH		1	L1	1749101
Inductor	ELC18B101L	100µH		1	L3	1308471
Connector	SI–60062–F	MagJack	RJ45	1	J3	1137983
Voltage reg	LM317EMP/NOPB		SOT–223	1	U7	1469023
Resistor	ERJ8ENF1601V	1.6kOhm	1206	1	R36	2307429
Resistor	ERA8AEB101V	100Ohm	1206	10	R1, R10, R15, R20, R25, R32, R51, R52, R53, R56	1717738

Resistor	CR1206–FX–1003ELF	100kOhm	1206	15	R13, R14, R18, R19, R23, R24, R28, R29, R3, R33, R34, R42, R5, R7, R9	2333542
Resistor	CRCW120610K0FKTA	10kOhm	1206	4	R35, R43, R54, R55	1653050
Resistor	MC0125W120612K26	2.26kOhm	1206	1	R31	2142201
Resistor	CRCW120645K3FKEA	45.3kOhm	1206	6	R11, R16, R2, R21, R26, R59	2139563
Resistor	CRGH1206J47R	47Ohm	1206	2	R6, R8	2331845
Resistor	ERA8AEB4992V	49.9Ohm	1206	4	R38, R39, R40, R41	2095156
Resistor	CRCW120662K0FKEA	62kOhm	1206	6	R12, R17, R22, R27, R4, R58	1653159
Resistor	LR1206–R20F	0.2Ohm	1206	1	R37	1100337
Resistor	CRCW12060000Z0EAHP	0Ohm	1206	1	R30	2112787
Resistor	CRCW12061K50FKEA	1.5kOhm	1206	1	R46	1653081
Resistor	CRCW12061K00FKTA	1kOhm	1206	1	R45	1653075
Resistor		180Ohm		2	R44,R60	
Resistor	CRCW1210470RFKEA	470Ohm	1206	1	R57	1653186
Resistor	EXB38V681JV	680Ohm	0603	2	R47, R50	2060099
Resistor	EXB38V103JV	10kOhm	0603	2	R48, R49	2060107
Connector	95501–2661		RJ11	1	J2	1536490
Fuse holder	0031.8201		20X5MM	1	F1	1162740
Op amp	MCP6H01–E/SN	rail to rail	SOIC	7	U20, U21, U22, U23, U24, U25, U26	1863969

Diode zener	BZT52C4V7	4.7V	SOD–123	2	D16, D6	1902435
Sample Hold	SMP04ESZ		SOIC	2	U13, U9	1661013
Voltage reg	MC33063AD		SOIC	1	U1	1053591
Level shifter	CD40109BPW		TSSOP	2	U3, U4	1741425
Voltage reg	UA7805CKCS		TO–220	1	U6	2075448
Voltage reg	UA7905CKCS		TO–220	1	U2	2342562
Inductor	744772010			1	L4	1635794
Diode dual–zener	AZ23C4V7–7–F		SOT–23	6	U30, U31, U32, U33, U34, U35	1773570
Jumper	SPC20479			5		2396301
Fuse		1A		1		

Příloha D – Fotodokumentace osazené DPS



Příloha E – UDP Server

```
/*********************************************************************
 *
 *  UDP Server performance testing module for Microchip TCP/IP
 *      Stack - implemented with Berkeley API
 *   -Waiting for operation mode
 *   -Based on the mode respond immediately or with defined delay
 *
 *********************************************************************
 * FileName:        UDPServer.c
 * Dependencies:    UDP, ARP, Tick
 * Processor:       PIC18, PIC24F, PIC24H, dsPIC30F, dsPIC33F, PIC32
 * Compiler:        Microchip C32 v1.05 or higher
 *                  Microchip C30 v3.12 or higher
 *                  Microchip C18 v3.30 or higher
 *                  HI-TECH PICC-18 PRO 9.63PL2
 *                                  or higher
 * Company:         CTU  - Faculty of Electrical Engineering
 *
 * Software License Agreement
 *
 * Copyright (C) 2015 Jiri Petrak. All rights reserved.
 *
 *
 * Author               Date        Comment
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Jiri Petrak      10/05/15                    Original
 ********************************************************************/
#define __UDP_SERVER_C

#include "TCPIPConfig.h"

#include "TCPIP Stack/TCPIP.h"

#define UDP_CLIENT_PORT         8001
#define UDP_LISTEN_PORT         6777

// Defines how frequently respond in mode 3.
#define TIMEOUT     (TICK_SECOND)

// IP address of udp client
#define CLIENT_IP_ADDRESS_BYTE1 (169ul)
#define CLIENT_IP_ADDRESS_BYTE2 (254ul)
#define CLIENT_IP_ADDRESS_BYTE3 (1ul)
#define CLIENT_IP_ADDRESS_BYTE4 (131ul)


#if defined(STACK_USE_BERKELEY_API)



// Defines the structure of an UDP packet
typedef struct
{
    struct
    {
        BYTE a  : 1;
```

```
        BYTE b  : 1;
                BYTE c  : 1;
    } flags;
    BYTE aaa;
    BYTE bbb;
        BYTE ccc;
        BYTE ddd;
        BYTE eee;
        BYTE fff;
} UDP_PACKET;

// Received data variables
static BYTE bytercv0 = 0;
static BYTE bytercv1 = 0;
static BYTE bytercv2 = 0;
static BYTE bytercv3 = 0;
static BYTE bytercv4 = 0;
static BYTE bytercv5 = 0;
static BYTE mode = 0;
static BYTE onoff= 0;

// Testing variables
int x=0;
int y=0;
int errstate=0;
int counter=0;
int errstate2=0;

void UDPServer(void)
{
static DWORD        dwTimer;
static SOCKET       bsdUdpClient;
UDP_PACKET          pkt;
int                 i;
static struct sockaddr_in   udpaddr;
int                 addrlen = sizeof(struct sockaddr_in);
static enum
    {
        SM_HOME = 0,
        SM_CREATE_SOCKET,
        SM_BIND,
        SM_UDP_SEND,
        SM_UDP_RECV,
        SM_WAIT
    } State = SM_HOME;

        switch(State)
    {
        case SM_HOME:
            dwTimer = (TickGet());
            State=SM_CREATE_SOCKET;
            break;
        case SM_CREATE_SOCKET:
            // Create a socket
             bsdUdpClient = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
            if(bsdUdpClient == INVALID_SOCKET){
                            errstate2=1;
                    }
            State=SM_BIND;
```

```
    break;

case SM_BIND:
    // Because we listen first explicit binding has to be made.
    // When we are going to send first send UDP
    // packet then send() do implicit binding.
    udpaddr.sin_port = UDP_LISTEN_PORT;
    udpaddr.sin_addr.S_un.S_addr = IP_ADDR_ANY;
    if( bind(bsdUdpClient, (struct sockaddr*)
            &udpaddr, addrlen) == SOCKET_ERROR ){
        errstate=1;
    }
    State=SM_UDP_RECV;
    break;
case SM_UDP_SEND:
    // Trasmit UDP packet based on the mode
    if(mode==7){
        memset(&pkt, 0, sizeof(pkt));
    pkt.flags.a = 2;
    pkt.flags.b = 2;
    pkt.aaa = 1;
                pkt.bbb = mode;
                pkt.ccc = 7;
    udpaddr.sin_port = UDP_CLIENT_PORT;
    udpaddr.sin_addr.S_un.S_addr = CLIENT_IP_ADDRESS_BYTE1 |
            CLIENT_IP_ADDRESS_BYTE2<<8ul |
            CLIENT_IP_ADDRESS_BYTE3<<16ul |
            CLIENT_IP_ADDRESS_BYTE4<<24ul;
    if(sendto(bsdUdpClient, (const char*)&pkt, sizeof(pkt),
        0, (struct sockaddr*)&udpaddr, addrlen)>0){

    }
     State=SM_UDP_RECV;
     dwTimer = (TickGet());
     udpaddr.sin_port = UDP_LISTEN_PORT;
    }
    if(mode==3){
    if((TickGet()) - dwTimer > TIMEOUT){
    memset(&pkt, 0, sizeof(pkt));
    pkt.flags.a = 2;
    pkt.flags.b = 2;
    pkt.aaa = 1;
                pkt.bbb = mode;
                pkt.ccc = 3;
    udpaddr.sin_port = UDP_CLIENT_PORT;
    udpaddr.sin_addr.S_un.S_addr = CLIENT_IP_ADDRESS_BYTE1 |
            CLIENT_IP_ADDRESS_BYTE2<<8ul |
            CLIENT_IP_ADDRESS_BYTE3<<16ul |
            CLIENT_IP_ADDRESS_BYTE4<<24ul;
    if(sendto(bsdUdpClient, (const char*)&pkt, sizeof(pkt),
                0, (struct sockaddr*)&udpaddr, addrlen)>0){
    }
     State=SM_UDP_RECV;
     dwTimer = (TickGet());
     udpaddr.sin_port = UDP_LISTEN_PORT;
        }
    }
    break;
case SM_UDP_RECV:
```

```
// Look for packet reception
    i = recvfrom(bsdUdpClient, (char*)&pkt, sizeof(pkt),
            0, (struct sockaddr*)&udpaddr, &addrlen);
    if(i < (int)sizeof(pkt))
{
                    counter++;
}else{
bytercv0 = (pkt.aaa);
            bytercv1 = (pkt.bbb);
            bytercv2 = (pkt.ccc);
            bytercv3 = (pkt.ddd);
            bytercv4 = (pkt.eee);
            bytercv5 = (pkt.fff);
            // Definition of sending mode
            mode=bytercv1;
            onoff=bytercv2;
            if(mode==5){
                    State=SM_UDP_RECV;
                     break;
                        }
            State=SM_UDP_SEND;

            break;
            }
    // After initialisation packet MCU starts sending
    // data with defined frequency by TIMEOUT
if(mode==3){
State=SM_UDP_SEND;
break;
}
    // After packet reception MCU switch an output and send
    //no answer
if(mode==5){
    if(onoff==1){
        IGBTOUT1_IO=1;
    }
    if(onoff==0){
        IGBTOUT1_IO=0;
    }
}
    // After packet reception MCU switch an output and send
    //immediate answer
if(mode==7){
    if(onoff==1){
        IGBTOUT1_IO=1;
        //Nop();
        //Nop();
       // Nop();
       // Nop();
     //   Nop();
     //   Nop();
     //   Nop();
     //   Nop();
      // IGBTOUT1_IO=0;

    State=SM_UDP_SEND;
    }
    if(onoff==0){
        IGBTOUT1_IO=0;
```

```
                State=SM_UDP_SEND;
                }
            }
            State=SM_UDP_RECV;
            break;
    }
}
#endif  //if defined(STACK_USE_BERKELEY_API)
```

Příloha F – UDP Client

```
/*********************************************************************
 *
 *  UDP Client performance testing app.
 *
 * This is testing app which is able to send and receive UDP packets
 * which was created for communication with MCU Microchip PIC18F97J60
 * especially with UDPServer function.
 *
 * App crash possbile when incorrect input is used
 *
 *********************************************************************
 * FileName:        UDPClient.java
 *
 * Company:         CTU  - Faculty of Electrical Engineering
 *
 * Software License Agreement
 *
 * Copyright (C) 2015 Jiri Petrak. All rights reserved.
 *
 *
 * Author               Date        Comment
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Jiri Petrak      10/05/15                    Original
 ********************************************************************/

package udpclient;

import java.io.IOException;
import java.net.*;
import java.util.Arrays;
import java.util.Scanner;

public class UDPClient
{
    // Ports definition
    static public int UdpClientOutPort=6778;
    static public int UdpClientListenPort = 8001;
    static public int UdpServerPort = 6777;

    public static void main(String[] args) throws Exception
    {
    // All possible modes of communication with MCU are separated by slash
     String mode;
     Scanner in = new Scanner(System.in);
     System.out.println("Enter: SendDutyCycle/SendControlPulses/"
             + "DutyCycleAndReception/PulsesAndReception/Reception/Reset");
     mode = in.nextLine();
     System.out.println("You entered string "+mode);
     new UDPClient().mode(mode);
    }

    // Menu structure
    public void mode(String mode) throws IOException {
    switch (mode){
    case "SendDutyCycle":
        new UDPClient().SendDutyCycle();
```

```
            break;
        case "SendControlPulses":
            new UDPClient().SendControlPulses();
            break;
         case "DutyCycleAndReception":
             new UDPClient().DutyCycleAndReception();
             break;
         case "PulsesAndReception":
             new UDPClient().PulsesAndReception();
             break;
         case "Reception":
             new UDPClient().Reception();
             break;
         case "Reset":
             new UDPClient().Reset();
             break;
         default:
            System.out.println("error ");
            break;
        }
}

    //This mode sends UDP packet with Duty Cycle
    public void SendDutyCycle() throws IOException {
        int strida;

        InetAddress sourceAddr = InetAddress.getLocalHost();
        DatagramSocket datagramSocket = new
DatagramSocket(UdpClientOutPort,
                sourceAddr);
        Scanner in = new Scanner(System.in);
        while(true){
        System.out.println("Enter Duty Cycle [%]: 1-99 ");
        strida = in.nextInt();
        System.out.println("You entered string: "+strida);

        byte[] buffer = new byte [7];
        buffer[3]=(byte) strida;
        buffer[2]=6;
        buffer[1]=5;
        buffer[0]=0;

        InetAddress address = InetAddress.getByName("169.254.1.1");
        DatagramPacket packet = new DatagramPacket(buffer, buffer.length,
address, UdpServerPort);
        datagramSocket.send(packet);
        System.out.println("Packet sent");
        }
     }

    //This mode sends periodically "0" and "1" with defined delay
    public void SendControlPulses() throws IOException {
        int f;
        InetAddress sourceAddr = InetAddress.getLocalHost();
        DatagramSocket datagramSocket = new
DatagramSocket(UdpClientOutPort, sourceAddr);
        Scanner in1 = new Scanner(System.in);
        System.out.println("Enter Freq [ms]: 1-1000");
        f = in1.nextInt();
```

```
        System.out.println("You entered string: "+f);

        byte[] buffer = new byte [7];
        buffer[3]=0;
        buffer[2]=5;
        buffer[1]=5;
        buffer[0]=0;

        int x=0;

        while(true){
        if(x==0) {
            buffer[3]=0;
            x=1;
            } else {
            buffer[3]=1;
            x=0;
            }
        InetAddress address = InetAddress.getByName("169.254.1.1");
        DatagramPacket packet = new DatagramPacket(buffer, buffer.length,
address, UdpServerPort);
        datagramSocket.send(packet);
        System.out.println("Packet sent");
        try {
            Thread.sleep(f);                //1000 milliseconds is one
second.
            }catch(InterruptedException ex) {
            Thread.currentThread().interrupt();
            }
        }
     }
    //This mode sends UDP packet with Duty Cycle and then show up all
    //incoming UDP packets
    public void DutyCycleAndReception() {
        try {
            int strida;
            InetAddress sourceAddr = InetAddress.getLocalHost();
            DatagramSocket datagramSocket = new
DatagramSocket(UdpClientOutPort, sourceAddr);
            Scanner in = new Scanner(System.in);
            System.out.println("Enter Duty Cycle [%]: 1-99 ");
            strida = in.nextInt();
            System.out.println("You entered string: "+strida);
            byte[] buffer = new byte [7];
            buffer[3]=(byte) strida;
            buffer[2]=3;
            buffer[1]=3;
            buffer[0]=0;
            InetAddress address = InetAddress.getByName("169.254.1.1");
            DatagramPacket packet = new DatagramPacket(buffer,
buffer.length, address, UdpServerPort);
            datagramSocket.send(packet);
            System.out.println("Packet sent");


            DatagramSocket serverSocket = new
DatagramSocket(UdpClientListenPort);
            byte[] receiveData = new byte[8];
```

```
            System.out.printf("Listening on udp:%s:%d%n",
                InetAddress.getLocalHost().getHostAddress(),
UdpClientListenPort);
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
                        receiveData.length);

            while(true)
                {
                serverSocket.receive(receivePacket);
                System.out.println("RECEIVED: "
+Arrays.toString(receiveData));
                }
        }catch (IOException e) {
              System.out.println(e);
        }
    }
    // This mode sends periodically "0" and "1" with defined delay and also
    // show up all incomimg UDP packets
     public void PulsesAndReception() {
        try {
            long time1 = 0,time2,timev;
            int f;
            InetAddress sourceAddr = InetAddress.getLocalHost();
            DatagramSocket datagramSocket = new
DatagramSocket(UdpClientOutPort, sourceAddr);
            Scanner in1 = new Scanner(System.in);
            System.out.println("Enter Freq [ms]: 1-1000");
            f = in1.nextInt();
            System.out.println("You entered string: "+f);

            byte[] buffer = new byte [7];
            buffer[3]=0;
            buffer[2]=7;
            buffer[1]=7;
            buffer[0]=0;

            InetAddress address = InetAddress.getByName("169.254.1.1");
            DatagramPacket packet = new DatagramPacket(buffer,
buffer.length, address, UdpServerPort);
            datagramSocket.send(packet);
            System.out.println("Initilisation packet sent");


            DatagramSocket serverSocket = new
DatagramSocket(UdpClientListenPort);
            byte[] receiveData = new byte[8];

            System.out.printf("Listening on udp:%s:%d%n",
                    InetAddress.getLocalHost().getHostAddress(),
UdpClientListenPort);
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
                                receiveData.length);
            int z=0;
            long time3 = 0;
            while(true)
            {
                //When no packet is received until 1.1*f next packet is
sent.
                if(time3*1000<1.1*f){
```

```
                    serverSocket.receive(receivePacket);

                }
                time2=System.nanoTime();
                timev=time2-time1;
                System.out.println("RECEIVED: "
+Arrays.toString(receiveData));
                System.out.printf("TIME:%d\n " ,timev);
                if(z==0) {
                    buffer[3]=0;
                    z=1;
                    } else {
                    buffer[3]=1;
                    z=0;
                    }
                try {
                    Thread.sleep(f);    //1000 milliseconds is one second.
                    }catch(InterruptedException ex) {
                    Thread.currentThread().interrupt();
                }
                InetAddress address1 =
InetAddress.getByName("169.254.1.1");
                DatagramPacket packet1 = new DatagramPacket(buffer,
buffer.length, address1, UdpServerPort);
                DatagramSocket datagramSocket1 = new DatagramSocket();
                datagramSocket1.send(packet1);
                time1=System.nanoTime();
                time3=System.nanoTime();
                System.out.println("Packet sent");

            }
        }catch (IOException e) {
              System.out.println(e);
            }
    }
    // Show up all incomimg UDP packets
    public void Reception() {
        try {

            DatagramSocket serverSocket = new
DatagramSocket(UdpClientListenPort);
            byte[] receiveData = new byte[8];

            System.out.printf("Listening on udp:%s:%d%n",
                    InetAddress.getLocalHost().getHostAddress(),
UdpClientListenPort);
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
                                receiveData.length);
            while(true)
            {
                  serverSocket.receive(receivePacket);
                  String sentence = new String( receivePacket.getData(), 0,
                                    receivePacket.getLength() );
                  System.out.println("RECEIVED: " + sentence);
            }
        }catch (IOException e) {
              System.out.println(e);
                }
```

```
    }
    // Send a packet will all values set as "0"
    public void Reset() throws SocketException, UnknownHostException,
IOException {
        InetAddress sourceAddr = InetAddress.getLocalHost();
        DatagramSocket datagramSocket = new
DatagramSocket(UdpClientOutPort, sourceAddr);

        byte[] buffer = new byte [7];
        buffer[3]=0;
        buffer[2]=0;
        buffer[1]=0;
        buffer[0]=0;

        InetAddress address = InetAddress.getByName("169.254.1.1");
        DatagramPacket packet = new DatagramPacket(buffer, buffer.length,
address, UdpServerPort);
        datagramSocket.send(packet);
        System.out.println("RESET packet sent");
    }
}
```