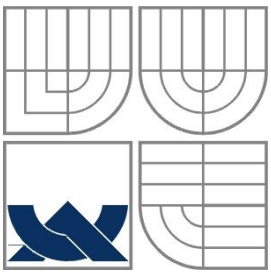
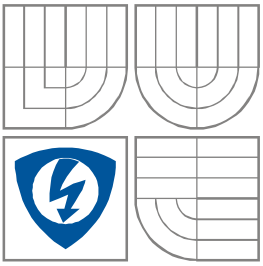


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

PŘEVODNÍK USB JOYSTICKU NA ETHERNETOVÉ ROZHRANÍ PRO ŘÍZENÍ ROBOTU

AN USB-ETHERNET JOYSTICK CONVERTOR FOR MOBILE ROBOT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

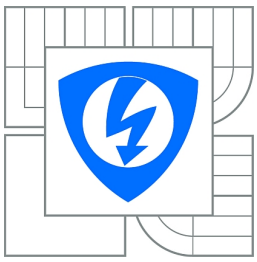
AUTOR PRÁCE
AUTHOR

Bc. LIBOR ŠUTERA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. FRANTIŠEK BURIAN

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Libor Šutera	***ID:***	109734
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Převodník USB joysticku na ethernetové rozhraní pro řízení robotu

POKYNY PRO VYPRACOVÁNÍ:

Na základě předcházející semestrální práce odladěné na vývojovém kitu navrhněte plošný spoj s procesorem STM32F107C, na kterém lze spustit kód předchozí semestrální práce. Práce by měla realizovat aplikaci převodníku USB joysticku na Ethernetové rozhraní, který je komunikačním protokolem UDP, kompatibilní s robotickým modulárním systémem používaným v Laboratoři Teleprezence a Robotiky. Předpokládá se použití této diplomové práce jako autonomního ovladače pro řízení reklamního robotu FektBot.

DOPORUČENÁ LITERATURA:

AXELSON, Jan. USB complete: the developer's guide. 4th ed. Madison, WI: Lakeview Research, c2009, 504 s. ISBN 19-314-4808-6.

USB ORG. Universal Serial Bus: HID Specifications [online]. [cit. 2012-01-30]. Dostupné z: http://www.usb.org/developers/hidpage

Termín zadání:	6.2.2012	***Termín odevzdání:***	21.5.2012

Vedoucí práce: Ing. František Burian
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Diplomová práce se zabývá návrhem a konstrukcí převodníku USB joysticku na Ethernetové rozhraní. V první části jsou teoreticky rozebrány oba protokoly použité pro převodník. Následuje rozbor současných mikroprocesorů na trhu a jejich možnosti uplatnění v tomto projektu, včetně specifikace použitého mikroprocesoru a jeho možnosti programování a ladění. Další část práce se zabývá podrobným návrhem hardwarové části převodníku. Poslední část popisuje softwarové vybavení mikroprocesoru a závěrečné zhodnocení práce.

Klíčová slova

Mikrokontrolér, Ethernet, USB, ARM, Hardware, Vývojový kit, STM32f107, PHY, ISO/OSI

Abstract

This thesis describes the design of construction of the converter USB joystick on the Ethernet interface. In the first part are theoretically analyzed both protocols used for the converter. An analysis of current microprocessor trade and their possible using for application in this project. Next part including the specification of used microprocessor and all options of programming and debugging of the microprocessor. Another part deals with the detailed design of the hardware interface. The last part describes the software equipment of microprocessor and the final appreaciation of work.

Keywords

Microprocessor, Ethernet, USB, ARM, Hardware, Development kit, STM32f107, PHY, ISO/OSI

Bibliografická citace:

ŠUTERA, L. *Převodník USB joysticku na ethernetové rozhraní pro řízení robotu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 72 s. Vedoucí diplomové práce Ing. František Burian.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Převodník USB joysticku na ethernetové rozhraní pro řízení robotu jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **18. května 2012** ……
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Františku Burianovi za účinnou metodickou, pedagogickou a odbornou pomoc a další velmi cenné praktické rady, které uplatním nejen při zpracování mé diplomové práce.

V Brně dne: **18. května 2012** ……
podpis autora

OBSAH

ÚVOD

Cílem práce je navrhnout a zkonstruovat zařízení, které bude schopné přijímat data ze zařízení USB HID a ty dále odesílat po sběrnici Ethernet. Diplomovou práci jsem logicky rozčlenil do několika kapitol. V první části projektu jsem popsal základní vlastnosti použitých protokolů a jednotlivých vrstev. Protokoly jsou popsané z pohledu referenčního modelu ISO/OSI, přičemž se zabývám jen vrstvami, které jsou skutečně použité na převodníku. Další část se zabývá výběrem a specifikací konkrétního mikrokontroléru a jeho možnostmi programování a laďění. Velice podrobně je popsaný hardwarový návrh celého zařízení, který byl rozšířen o další periferie. Aplikace převodníku byla nejdříve realizována na vývojovém kitu firmy Hitex – STM32ComStick. V poslední části projektu je popsán firmware převodníku a celá práce je zhodnocena.

Uplatnění převodníku se předpokládá na reklamním robotu Laboratoře Teleprezence a Robotiky. Na robotu FektBot bude převodník použitý pro ruční navigaci robota pomocí USB joysticku. Před testováním na robotu, jsem měl možnost testovat jednotlivé fáze komunikace na paralelně vyvíjeném převodníku CAN – Ethernet.

V práci jsou použity výrazy a některé anglické technické termíny, které nejsou překládány, protože nemají český překlad nebo je dané označení srozumitelnější. Pro vyjádření logické úrovně používám jednotku *bit* s označením malé *b*. V popisu firmware vyjadřuji hexadecimálním hodnoty značkou *0x* a spojení 8bitů jako bajt (Byte) s označením velké *B*.

Při návrhu hardwaru převodníku využívám velké množství technické dokumentace použitých součástek, která není citována, protože je používána pouze pro potřeby návrhu hardwaru.

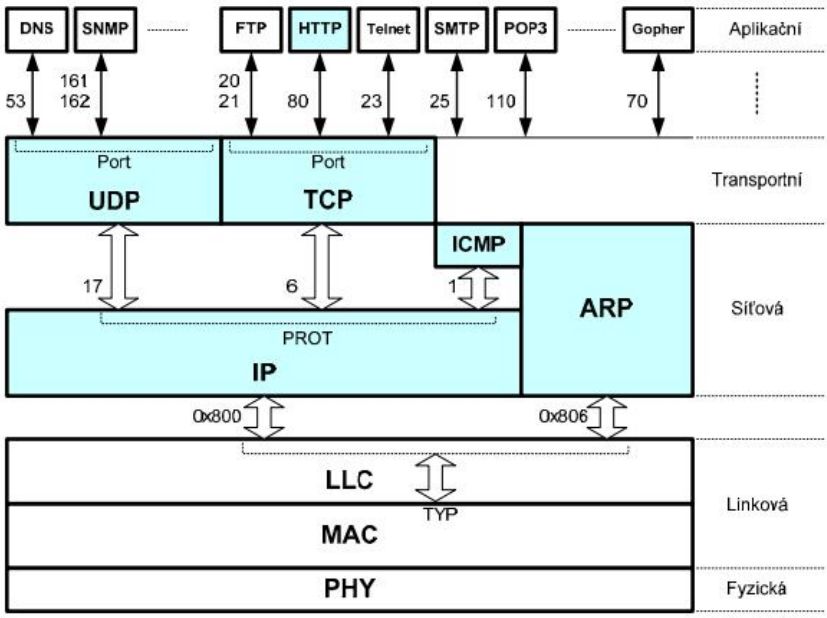
1 ETHERNET

1.1 Obecný popis a vznik Ethernetu

Ethernet je sériová sběrnice a její standard vypracovala organizace IEEE. Standard byl vydán v roce 1985 s označením „IEEE 802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifikation“, kde byla popsána metoda mnohonásobného přístupu a detekce kolize.

Technologie sítí Ethernet, ale byla vyvinuta již začátkem 70. let ve firmě Xerox, která si zaregistrovala značku Ethernet. Měl rychlost 2,94Mb/s a používal koaxiální kabel (70 ohmů) až 1km dlouhý. Pozdější Ethernet se poněkud liší od původního včetně označení „Ethernet“ (dále však používám stejné označení). První standart Ethernetu vytvořila organizace IEEE díky tomu, že skupina firem DIX – DEC, Intel a Xerox se rozhodly neponechat si Ethernet pouze jako své proprietární řešení, ale naopak předaly jeho specifikace a nechaly jej standardizovat. Z návrhu tak vznikl standard IEEE 802.3. Tímto způsobem postupně vznikla celá řada standardů Ethernetu majících různé parametry, různá přenosová média a různé způsoby provozu.

Kromě organizace IEEE se normalizací datové komunikace zabývá i mezinárodní normalizační úřad ISO, který vypracoval takzvaný referenční model OSI. Rozděluje všechny úkony potřebné k výměně dat do sedmi vrstev, které jsou realizované hardwarovými nebo programovými prostředky. Jednotlivé vrstvy zajišťují funkčnost vyšších vrstev. Mezi vrstvami jsou definovány mezivrstvou protokoly, které slouží pro komunikaci jednotlivých vrstev navzájem. Ethernet je dnes standardizován normou ISO 8802/3.



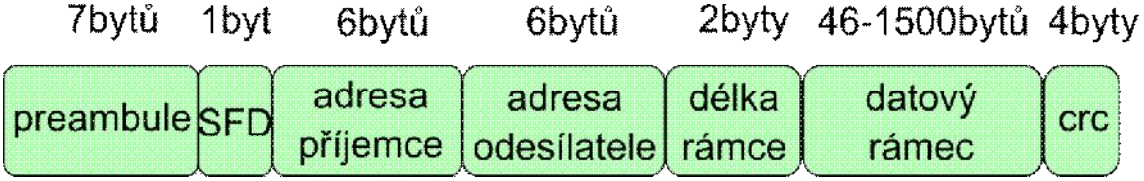
Obrázek 1: Hierarchická struktura protokolových vrstev datových sítí [21] – zjednodušený model ISO/OSI

1.2 Fyzická vrstva

V současnosti se používají rychlosti 10, 100 a 1000Mbit/s. Ethernet je definován jak pro klasické kabely (kroucený pár nebo koaxiální kabel), tak i pro bezdrátový přenos nebo pro optický kabel. Optický kabel dokáže přenášet data na velké vzdálenosti s rychlostí více jak 1Gbit/s. Použitá fyzická vrstva Ethernetu (kapitola 7.2) podporuje současně verze 10BASE-T i 100BASE-TX, které používají pro přenos kroucený pár zakončený konektory RJ45 a dokáže pracovat v polo/plně duplexním režimu.

1.2.1 Ethernetový rámec

Data jsou odesílána v rámcích, v kterých jsou zabaleny pakety. Rámce se dělí na dva základní typy, které jsou schopné pracovat spolu na stejné síti. Rozdíl mezi rámcem Ethernet II a IEEE 802.3 CSMA/CD je ve formátu preambule a ve významu 2bytů před datovou oblastí. V případě Ethernetu II tyto byty označují protokol vyšší vrstvy a v případě IEEE 802.3 délku přenášených dat.



Obrázek 2: Rozdělení rámce IEEE 802.3

Ethernetový rámec obsahuje hlavičku, kde je cílová a zdrojová MAC adresa, typ paketu. Pro synchronizaci vysílací a přijímací strany je rámec uvozený 7bytů dlouhou sekvencí střídajících se 1 a 0. Následuje SFD, který odděluje začátek rámce. Datová oblast je proměnné délky, ale standardně bývá 46-1500 bytů dlouhá. V případě, že jsou data kratší než nejmenší možná délka 46bytů, tak bude datový rámec doplněn nedefinovanými znaky nebo definovanou sadou znaků. Jako poslední v rámci je tzv. zabezpečení rámce – 4byty obsahující kontrolní součet CRC pro zjištění vadných rámců.

MAC adresa je unikátní identifikační číslo ethernetového zařízení. Skládá se z 48bitů, které jsou zapisovány v podobě šesti dvojic dvojtečkou oddělených hexadecimálních čísel například: 01:02:03:04:05:06. U zařízení u kterých se nepředpokládá připojení do veřejné sítě s jinými zařízeními je tato MAC adresa konfigurovatelná. Ale předpokládá se, že bude zajištěné, že nenastane kolize dvou zařízení se stejnou MAC adresou.

1.2.1.1 Postup sestavení ethernetového rámce

V první části, ve vlastní aplikaci (v případě kódu pro mikroprocesor), připojíme vlastní hlavičku, která může obsahovat v našem případě například typ zprávy. V modelu TCP/IP je toto práce aplikační vrstvy. V dalším kroku se zajistí přidání hlavičky komunikačního protokolu (TCP,UDP) a síťového protokolu IP. Dle specifikace modelu ISO/OSI se přidávají i kontrolní součty pro jednotlivé protokolové vrstvy. V poslední části vytváření rámce se přiřadí hlavička linkové a fyzické vrstvy a ukončení – kontrolní součet.

Postup při přijímání a dekódování rámce je stejný, ale s obráceným postupem – tj. postupné odebírání hlaviček, dekódování adresy příjemce, kontrola pomocí CRC až k samotným datům.

1.3 Linková vrstva

Ve specifikaci Ethernetu je linková vrstva rozdělena na dvě podvrstvy – LLC a MAC.

Horní podvrstva LLC je označována jako nezávislá vrstva řízení logického spoje a zajišťuje vytváření a rušení spojení linkových spojení, zajišťuje řízení datového toku a realizuje detekci chyb. Představuje rozhraní mezi MAC vrstvou a vyššími protokolovými vrstvami. LLC nabízí dvě základní přenosové služby – UCS (datagramová služba) a COS (služba virtuální spojování).

Druhá podvrstva MAC řídí přístup stanice ke sdílené sběrnici. V případě použitého mikrokontroléru pomocí metody CSMA/CD. Tato metoda definuje, jakým způsobem se bude komunikovat, pokud bude stanic více. V daném časovém okamžiku může vysílat pouze jedna stanice a ostatní stanice monitorují provoz na sběrnici, a pokud právě

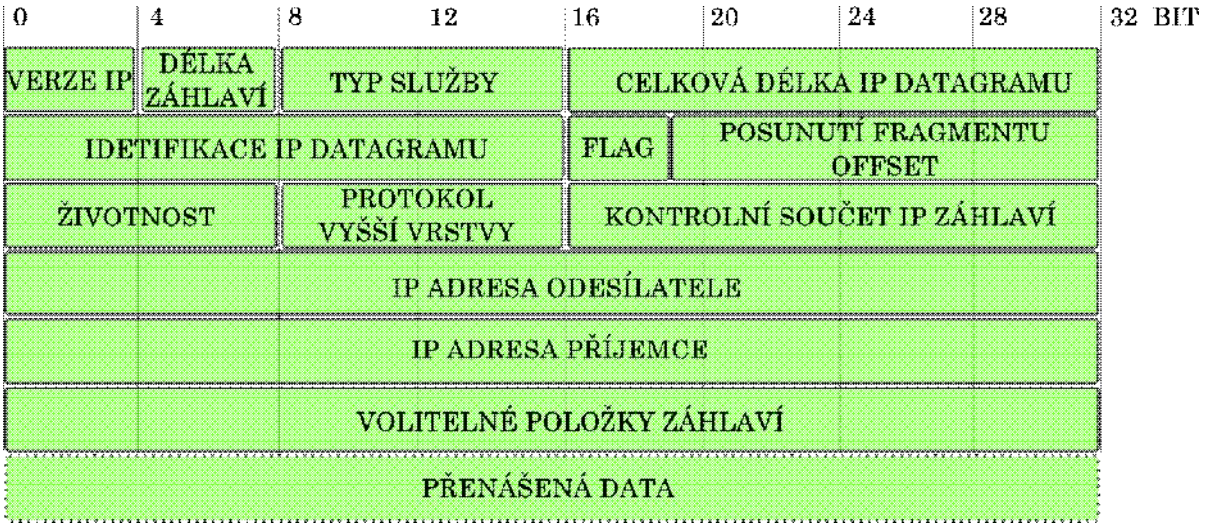
probíhá vysílání tak jsou blokovány. Jakmile stanice, která potřebuje vysílat, detekuje klidový stav sběrnice, může vyslat požadavek na vysílání dat následně začít vysílat data. Pokud je však ke sběrnici připojeno více zařízení, může dojít při vysílání dat ke kolizi. Kolize je stav, kdy se pokouší dvě různé stanice vystavit data na sběrnici, ve stejný okamžik. V takovém případě první stanice, která detekuje kolizi, začne vysílat kolizní okno neboli JAM. JAM má definovanou délku a hodnotu. Jakmile ostatní stanice identifikují kolizní okno, přeruší svoji aktivitu a po náhodně zvoleném čase se pokusí znovu odeslat svůj rámec dat. Pokud se operace odeslání rámce nepodaří 16 pokusy, tak je nahlášena vyšší vrstvě chyba a pokračuje se dalším rámcem.

1.4 Síťová vrstva

Tato vrstva definuje adresování jednotlivých zařízení a způsob směrování datagramů. Směrování provádí směrem od zdrojového uzlu k cílovému uzlu pro případ, že zařízení nejsou přímo spojeny. Tato vrstva nezajišťuje doručení datagramu, ale pouze správnou adresaci cílové stanice. Pro směrování datagramu bylo navrhnuto několik protokolů. Nejpoužívanější je Internet Protokol, který existuje ve dvou verzích – IP4 a IP6. Hlavní důvod vývoje nové verze IP6 je 32bitová adresa, jelikož se předpokládalo, že během dvou následujících let, se vyčerpají volné adresy protokolu IP4. Jako další směrovací a adresovací protokoly se používají ARP, ICMP, IGMP, IGRP.

1.4.1 IP protokol

Každý datagram je samostatný člen odesílaných dat, který musí obsahovat informace o odesílateli, adresátovy a identifikátor. Datagram má vlastní protokolovou hlavičku, která je typicky velikosti 20 byte. IP záhlaví je umístěno na začátku oblasti *data* v ethernetovém rámci (za hlavičku LLC vrstvy). [21]



Obrázek 3: Formát IP datagramu

1.4.2 Popis záhlaví IP protokolu

Verze IP

Verze IP protokolu pro IP4= 4b.

Délka záhlaví

Délka se udává jako počet 32 bitových slov, což je důvod předchozího způsobu zobrazení IP záhlaví. Typicky = 5b.

Typ služby

Udává vyšší protokolové vrstvě, jakým způsobem má zpracovat datagram z hlediska priority, zpoždění, propustnosti.

Celková délka IP datagramu

Udává celkovou délku datagramu v násobcích 32 bitů.

Identifikace IP datagramu

Slouží k podpoře fragmentace.

Flag

Příznaky fragmentace datagramu. Délka 3b.

Posunutí fragmentu

Má délku 13 bitů a udává vzdálenost fragmentu v původním datagramu od jeho začátku

Životnost

Zabezpečovací mechanismus proti bloudícím datagramům na síti – hodnota je inkrementována každým průchodem směrovačem. Pokud dosáhne dané hodnoty je datagram zahozen.

Protokol vyšší vrstvy

Identifikace protokolu vyšší vrstvy. Pro UDP = 17b.

Kontrolní součet IP záhlaví

Zabezpečovací součet záhlaví pro kontrolu správnosti údajů záhlaví přeneseného datagramu.

IP adresa odesílatele

IP adresa příjemce

IP adresy jsou 32 bitové adresy zdrojového a cílového zařízení (směrovače).

Volitelné položky záhlaví

Možnost vlastního nastavení, doplnění dalších informací IP záhlaví.

Přenášená data

Vlastní přenášená data síťové vrstvy.

1.5 Transportní vrstva

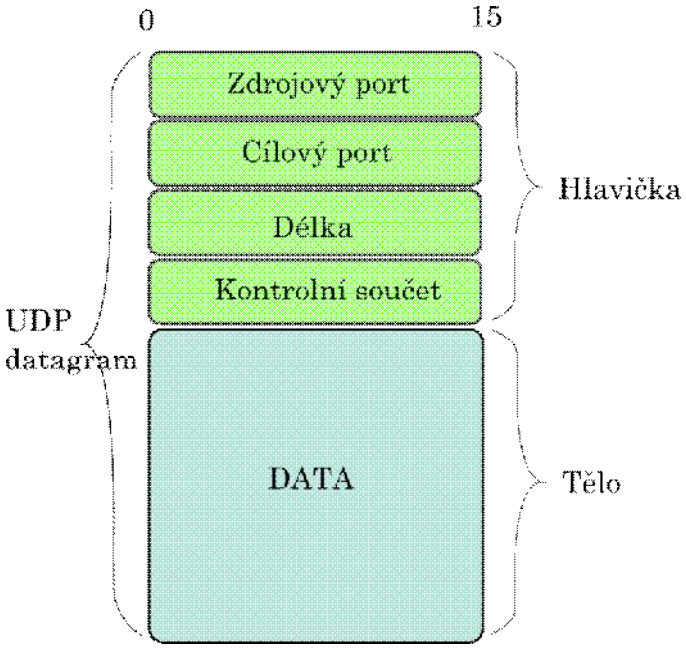
Hlavním úkolem transportní vrstvy je poskytovat efektivní přenosovou službu pro vyšší vrstvu. Vrstva může poskytovat spojovanou i nespojovanou přenosovou službu a pracuje jako rozhraní mezi poskytovateli přenosových služeb a jejich uživateli. Ve zvoleném modelu ISO/OSI zastupuje transportní vrstva i další dvě vyšší vrstvy modelu, které nejsou použity – což jsou Relační a Prezentační vrstva. Takže transportní vrstva plní úlohu sjednocení, tlumočení rozdílností mezi komunikujícími zařízeními. Mezi protokoly transportní vrstvy patří NBP, DCCP, RTP a nejpoužívanější TCP, UDP.

1.5.1 UDP

Komunikační protokol využívá transportní vrstva jako rozhraní mezi síťovou a aplikační vrstvou. Pro naši aplikaci byl vybrán protokol UDP hlavně z důvodu jeho jednoduchosti a toho, že je již implementován v celém projektu. U UDP protokolu každá aplikace přebírá zodpovědnost za úroveň spolehlivosti přenosu. UDP má charakter nespojovaného a nespolehlivého protokolu. Nezajišťuje správné pořadí doručovaných datagramů a neprovádí eliminaci duplicitních datagramů. O odeslaných datagramech si neudržuje informace. Ovšem pro aplikaci převodníku je tento protokol zcela dostačující i díky vyšší rychlosti než ostatní protokoly. Data, která jsou

převodníkem přenášena, nejsou kritická. To znamená, že v případě ztráty (zahození) méně jak 20 paketů za 1 vteřinu (což představuje 1/5 celkového počtu), nebude ohrožený chod zařízení ani bezpečnost aplikace, ve které bude pracovat.

1.5.1.1 Sestavení UDP datagramu



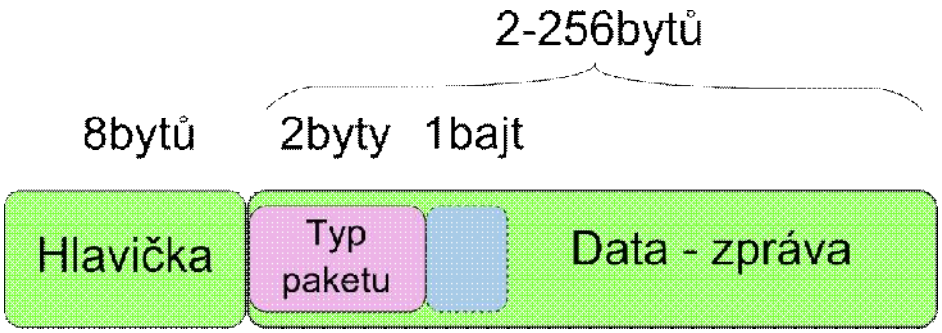
Hlavičku UDP datagramu tvoří 4 položky v rozsahu 16 bitů, které po řadě vyjadřují číslo portu odesilatele a příjemce, délku UDP datagramu, a kontrolní součet.

Číslo portu příjemce je základní informací, podle které se protokol UDP na straně příjemce rozhoduje, komu má přijatý datagram doručit. Protože UDP je bezstavový a odesilatel nemusí vyžadovat odpověď, zdrojový port je volitelný. Číslo portu odesilatele je nepovinné, vyplňuje se v případě, kdy je požadována odpověď (v opačném případě se tato položka zaplňuje nulami). Čísla portu v rozmezí 0 – 1023 jsou pevně daná a jsou vyhrazená pro nejběžnější služby. V rozsahu 1024 – 49151 jsou porty, které se při použití musí registrovat u příslušné autority. Zbývající porty jsou volně použitelné pro soukromé použití a nejsou přidělena žádné aplikaci.

Po číslech portů následuje povinná délka UDP paketu včetně dat, v bytech. Minimální hodnota je 8 bytů. Maximální hodnota je dána maximální hodnotou části hlavičky Lenght, což je 16 bitů – to znamená maximum 65535 bytů

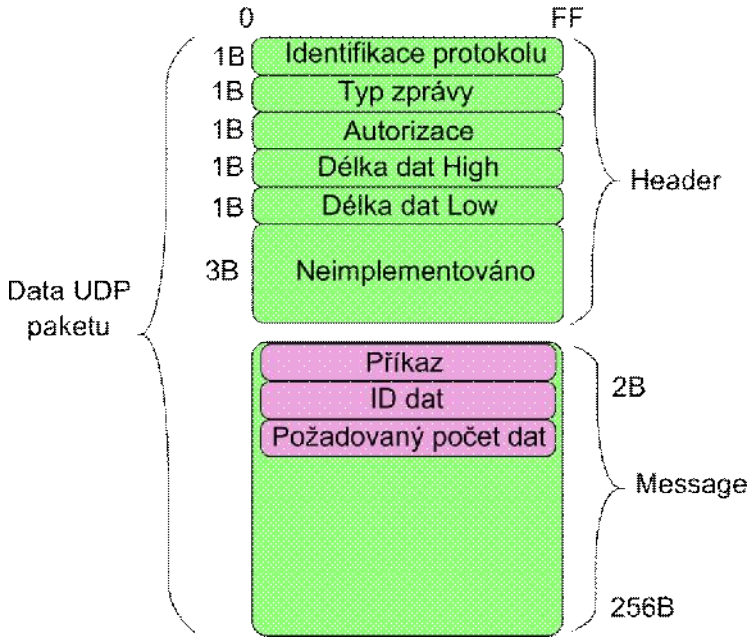
Položka LENGTH vyjadřuje délku UDP datagramu, měřenou v oktetech (tj. osmicích bitů) - včetně vlastní hlavičky. Minimální délka UDP datagramu je proto 8, což je právě velikost hlavičky, s prázdnou datovou částí. [21]

1.6 Vlastní protokolová vrstva



Obrázek 4: Struktura protokolu FektBot

Nad UDP protokolem – vrstvou, je dle potřeby kompatibility s robotickými systémy Laboratoře Teleprezence a Robotiky implementována další protokolová vrstva – FektBot. Tento protokol je přenášen jako data UDP protokolu. Protokol obsahuje vlastní hlavičku – header a vlastní přenášená data – message.



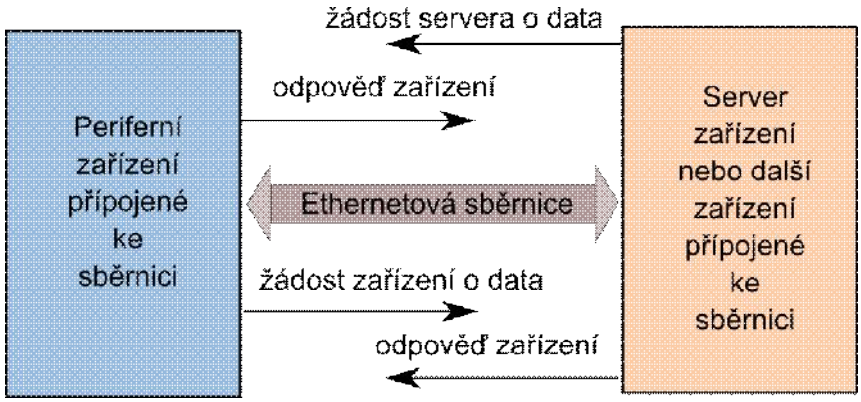
Obrázek 5: Popis protokolu FektBot

Hlavička protokolu je sestavena z 1B polí. První pole určuje identifikátor protokolu. Další byte určuje ty zprávy – v naší aplikaci budeme užívat pouze typ „datová zpráva“. Dále je třeba provést autorizaci žadatele o data a nastavení jeho přístupových práv (pouze Read, Read + částečný Write, Read + Write). Jako v každém protokolu je na dalších dvou bytech počet přenášených, dat pro případný kontrolní součet, který je rozdělen na LenHi a LenLow. Následuje vlastní zpráva, která obsahuje už konkrétní data. Na začátku datové části jsou povinné 2B, které rozlišují, zda se jedná o žádost

nebo odpověď a určují typ operace s daty. Poslední byte je nepovinný a je použitý pouze v případě sekvenční operace s daty.

1.7 Aplikační vrstva

Pro demonstraci možných způsobů komunikace jsem implementoval dva různé druhy komunikace na ethernetové sběrnici. Druhů komunikace na ethernetové sběrnici je mnohem více, ale tyto jsou uplatněny na robotu FektBot. Jedná se o komunikaci typu Server – Klient, která je upravená takovým způsobem, že i klient může žádat kterékoliv zařízení připojené na síti o data. Během komunikace se budou přenášet provozní data jednotlivých zařízení, připojených ke sběrnici. Pokud nemají zařízení programovatelné ethernet rozhraní, tak se použije převodník jako v případě USB zařízení. Každé zařízení má u sebe tabulku provozních hodnot a v tabulce má přidělené místo pro ukládání – aktualizaci vlastních proměnných. Hodnoty v tabulce jsou adresovány hexadecimálně hodnotami 0x00h až 0x3F, což znamená 64 adres celkově pro celou síť. Na každé adrese může být 1,2 nebo 4 bytová data. Jedno zařízení může mít přiděleno několik adres – například motory budou mít zvlášť buňky (adresy) pro rychlost motoru, proud motoru, teplota motoru apod. Tyto hodnoty, budou na žádost jiného zařízení nebo na žádost servera odesílat do sítě. Zjednodušené schéma komunikace je na následujícím obrázku. Konkrétní popis komunikace je v další kapitole. Tabulka je uvedena v kapitole Přílohy.



Obrázek 6: Upravená komunikace Server-Klient

1.7.1 Komunikace Server - Klient

Server - Klient je druh komunikace, který je uplatněný na robotu FektBot. Pracuje na principu žádostí a odpovědí na žádosti. V sítí existuje Server, který periodicky odesílá žádosti o data jednotlivým zařízením na sítí. Žádosti odesílá postupně, tak aby nedošlo ke kolizi na síti. Žádosti budou vypadat například takto: pro USB joystick - požadovaný směr pohybu, pro měniče motorů – velikost proudů apod. Server má v paměti uloženou

zmiňovanou tabulku proměnných a jednotlivými dotazy si ji stále aktualizuje, takže je možné při pozorování této tabulky – například testovacím programem Monitor (vytvořený v Laboratoři Teleprezence a Robotiky), zjistit jakoukoliv provozní hodnotu.

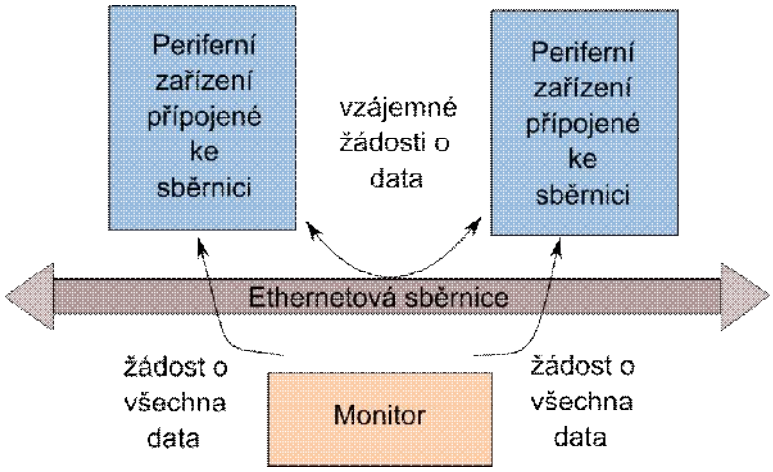
Tento monitor byl využitý pro ověření funkce a screenshot programu je v přílohách kap. 9. Hexadecimální čísla představují adresu - ID daného registru. V možnostech nastavení je volitelná IP adresa, komunikační port a interval vysílání požadavku. Pole *Statistics* zobrazuje počet odeslaných/přijatých paketů a jejich rozdíl tedy ztracených.

1.7.2 Komunikace Klient – Server (Klient)

Tento druh komunikace je implementován pro zařízení, které vyžadují určitá data potřebná pro plnění své funkce. Například měniče motorů budou žádat informace o směru otáčení, rychlosti. Komunikaci jsem otestoval připojením druhého ComSticku ke sběrnici a převodníky jsem nastavil tak, aby si navzájem periodicky odesílaly žádosti o data. Takže převodníky si navzájem aktualizují tabulku dat, kterou je opět možné sledovat programem Monitor.

1.7.3 Otestování funkčnosti komunikace

Oba druhy komunikace jsem otestoval současně. Každý z ComSticků odpovídá na žádosti programu Monitor. Odpovídá tak, že odešle celý registr dat zpět na port a IP adresu odkud přišla žádost. Dále v intervalech 250ms odesílá žádosti na pevně danou IP adresu a na určený port, ve které je žádost na čtení jedné adresy z registru dat. Na této domluvené adrese se inkrementuje hodnota pro ověření funkčnosti. Tuto hodnotu si tedy žádající cyklicky zapisuje do svého registru. Stejně to funguje i naopak, jen inkrementace má jiné parametry. Oba ComSticky současně odpovídají na žádosti Monitoru, který má tedy úplný registr dat.



Obrázek 7: Blokové schéma testovacího spojení

2 USB

Universal Serial Bus (USB) je sériová sběrnice, určená pro přenos dat na kratší vzdálenost. Specifikace standardu USB definuje především architekturu sběrnice, elektrické a mechanické vlastnosti sběrnice, přenosový protokol a s ním spojený datový tok na sběrnici. V současné době se začíná uplatňovat verze USB3.0, která je zpětně kompatibilní s nižšími verzemi.

První specifikace USB1.1 byla určena pro připojení počítačových periferií (klávesnice, myš) a pracovala na rychlosti *Low-speed* 1,5Mb/s a dále byla rozšířena na zařízení pro přenos dat velkých objemů *Full-speed* 12Mb/s.

Specifikace USB2.0 je určena pro zařízení s velice rychlým přenosem dat *High-speed device* 480Mb/s.

Na sběrnici je jedno zařízení Server, které se nazývá Host a řídí komunikaci. Ostatní zařízení připojená na sběrnici se nazývají Device, které specifikace USB rozděluje do tříd, podle toho o jaký typ zařízení se jedná. Používané zařízení joystick pro převodník patří do třídy Human Interface Device (HID) spolu s klávesnicí a myší. Označení této třídy je 0x03h. [23]

2.1 Fyzická vrstva USB

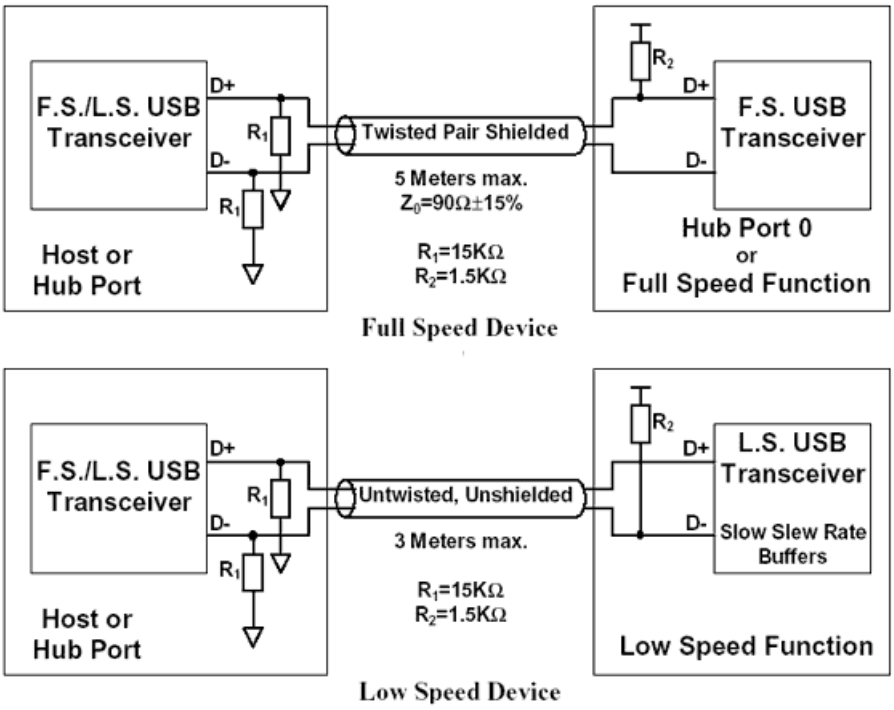
Fyzická vrstva je nejnižší v přenosovém modelu USB, která definuje architekturu sběrnice, elektrické a mechanické vlastnosti přenosového média sběrnice a kódování dat. Podrobné informace lze nalézt v [3].

Topologie sítě je stromová a na jejím začátku je vždy jeden hostitel. Jednotlivá zařízení připojená ke sběrnici jsou identifikována unikátní adresou. Více zařízení je možné připojit ke sběrnici pomocí rozbočovačů a to až 127 zařízení k jednomu hostiteli. Připojená zařízení se logicky dělí do vrstev a ve specifikaci USB2.0 je jejich maximální počet roven 7. Každé koncové zařízení obsahuje několik bran (endpoints), s vlastními FIFO pamětmi, které jsou sdruženy do rozhraní (interfaces), přes které komunikuje s hostitelem.

Zařízení připojená ke sběrnici USB se dělí na 3 kategorie z hlediska napájení. Pro každou skupinu je definovaný maximální odebíraný proud a minimální napětí, při kterém, musí zařízení spolehlivě fungovat. V první kategorii jsou zařízení napájená ze sběrnice s odběrem maximálně 100mA, v druhé jsou zařízení s maximálním odběrem 500mA a poslední kategorie jsou zařízení, které mají vlastní napájení.

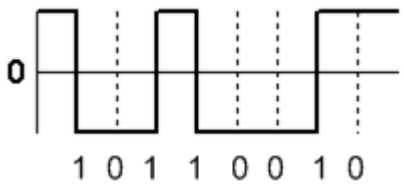
Přenos signálu po sběrnici USB probíhá pomocí dvou datových vodičů D+ a D-. Dle specifikace jsou datové vodiče schopny vydržet trvalý zkrat proti druhému vodiči, napájení sběrnice, GND a stínění kabelu. Hodnota signálu je vyhodnocována diferenciálně, kde nízká úroveň je vyhodnocena pokud $V_{OL} \leq 0{,}3V$ při zátěži $R_{PU}=1{,}5k\Omega$ proti napětí 3,6V a vysoká úroveň pokud $V_{OH} \geq 2{,}8V$ při zátěži $R_{PD}=15k\Omega$ proti napětí

0V. Na následujícím obrázku je zapojení budičů dle specifikace USB1.1 pro zařízení *Full/Low speed*, kde rezistory R_{PD} zaručují, že bez připojeného zařízení bude napětí na datových vodičích D+ a D- nulové. Připojované zařízení má na jednom z datových vodičů připojený rezistor R_{PU} proti 3,6V a po připojení k hostiteli se změní stav tohoto napětí. Pokud je rezistor připojen k D+ je detekováno zařízení *Full-speed*, pokud k D- jedná se o *Low-speed*.



Obrázek 8: Blokové schéma zapojení USB budičů převzato ze specifikace USB1.1[3]

Při přenosu dat používá USB kódování NRZI, kdy logická jednička znamená neměnný stav signálů a logická nula změnu do opačného stavu.



Obrázek 9: Příklad kódování NRZI

Z použitého kódování vyplívá, že sekvence logických jedniček nemusí způsobit změnu signálu po dlouhou dobu a protože sběrnice je synchronizována právě změnou stavu je zavedený *bit stuffing*. *Bit stuffing* znamená vkládání nuly do sekvence jedniček po každém 6 bitu, čímž dojde ke změně stavu a sychnronizaci sběrnice. Přijímač tuto vloženou nulu poté vypustí.

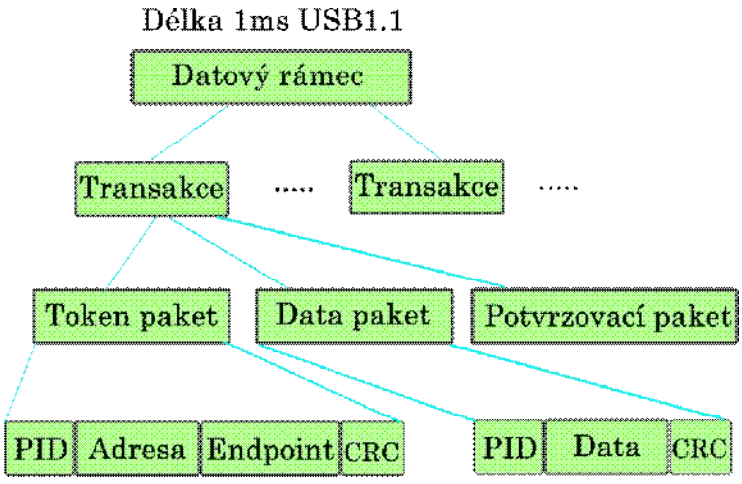
2.2 Linková vrstva

Linková vrstva se stará o organizaci časových rámců, v nichž se přenášejí informace formátované v paketech. Každý rámec začíná přenosem paketu SOF a délka rámce je pro specifikaci USB1.1 - 1ms±500ns a pro USB2.0 - 125μ±62,5ns.

Tabulka 1: Základní typy přenášených paketů [14]

Skupina paketů	Typ paketu	Popis
Pověřovací (Token)	OUT	Přenos dat od hostitele k zařízení
	IN	Přenos dat ze zařízení k hostiteli
	SETUP	Konfigurační přenos od hostitele
	SOF	Označuje začátek rámce
Datový (Data)	DATA0	Sudý datový paket (první v přenosu
	DATA1	Lichý datový paket
Potvrzovací (Handshake)	ACK	Potvrzení správného příjmu dat
	NAK	Data nebyla přijata nebo vyslána
	STALL	Brána zařízení je pozastavena nebo konfigurační požadavek není podporován
Speciální (Special)	PRE	Označuje LS přenos tak, aby rozbočovače zapnuly LS zařízení k nim připojené

Tabulka uvádí základní typy paketů pro revizi USB1.1, pro vyšší verze se tabulka rozšiřuje o další formáty datových, potvrzovacích paketů. Každý paket začíná synchronizační sekvencí SYNC 0x80 a po ní následuje identifikátor paketu PID, který není v tabulce uvedený a dále má pak definovanou strukturu podle toho, o jaký typ se jedná.



Obrázek 10: Příklad toku dat na USB

Pomocí pověřovacích paketů (Token) zahajuje USB Host jednotlivé přenosy. Pokud odesílá data nebo konfiguruje zařízení na síti, používá pakety typu OUT nebo SETUP. Za těmito pakety následují DATA0, DATA1, které se střídají pro případ chyby přenosu. Podobným způsobem putují data od zařízení k hostiteli uvozená paketem IN. [2]

Potvrzovací pakety ACK slouží pro hostitele i zařízení k potvrzení správného přijetí datového paketu. NAK vysílá zařízení hostiteli v případě, že není připraveno přijmout data. Paket STALL vysílá hostitel, pokud není přijatý řídicí požadavek podporován.

Speciální paket PRE slouží pro komunikaci se zařízeními *Low-speed* přes rozbočovač.

2.2.1 Typy přenosů na USB

Specifikace definuje čtyři druhy přenosů:

- Řídící přenos (Control transfer)
- Časovaný přenos (Interrupt Transfer)
- Izochronní přenos (Isochronous Transfer)
- Hromadný přenos (Bulk Transfer)

Každý typ přenosu, dlouhý zmiňovaných 1ms, má definovanou funkci, typ a posloupnost paketů, které může přenášek.

Řídicí přenos slouží ke konfiguraci a řízení USB zařízení.

Časovaný přenos probíhá v předem popsaných časových intervalech uvedených v deskriptoru zařízení. Typické zařízení používající tento typ přenosu je počítačová klávesnice nebo myš.

Hromadný přenos je určený pro přenos velkého objemu dat. Přenos zaručuje bezchybné doručení dat, pomocí kontroly kontrolních součtů každého paketu.

Izochronní přenos se používá pro přenos souvislého toku dat, přičemž je zaručená přenosová rychlost. Následek je nezaručení bezchybnosti dat. Tento přenos používají například zvukové karty. [14]

2.3 Deskriptory

Deskriptory jsou datové struktury definovaného formátu, kterými je popsáno USB zařízení. Celkový deskriptor zařízení je dále hierarchicky rozdělený na deskriptory konfigurace, rozhraní, brány a textového řetězce. Deskriptor zařízení má velikost 18 bajtů a v každém zařízení je pouze jeden. Deskriptor je první informace, kterou se snaží hostitel od zařízení zjistit. Obsahuje informace o verzi USB standardu, kód třídy, identifikátor výrobce a zařízení, které jednoznačně identifikují zařízení.

Tabulka 2: Standardní deskriptor zařízení [23]

Offset	Název	Velikost	Popis
0	*bLength*	1 bajt	Velikost deskriptoru v bajtech
1	*bDescriptorType*	1 bajt	Typ deskriptoru (0x01)
2	*bcdUSB*	2 bajty	Verze USB standardu
4	*bDeviceClass*	1 bajt	Kód třídy
5	*bDeviceSubClass*	1 bajt	Kód podtřídy
6	*bDeviceProtocol*	1 bajt	Kód protokolu
7	*bMaxPacketSize*	1 bajt	Maximální velikost paketu endpointu 0
8	*idVendor*	2 bajty	Identifikátor výrobce
10	*idProduct*	2 bajty	Identifikátor zařízení
12	*bcdDevice*	2 bajty	Verze zařízení
14	*iManufacturer*	1 bajt	Jméno výrobce
15	*iProduct*	1 bajt	Popis názvu výrobku
16	*iSerialNumber*	1 bajt	Sériové číslo zařízení
17	*bNumConfigurations*	1 bajt	Počet možných konfigurací

2.4 Enumerace zařízení

Enumerace je proces rozpoznání aktuálně připojeného zařízení. Jak jsem již zmínil, po připojení dojde ke změně na jedné z datových linek a podle toho se určí, o jaký typ zařízení se jedná. V tomto okamžiku provede Host reset sběrnice podržením obou linek v logické 0 po dobu 10ms, po které přejde do stavu obecného zařízení a jeho proudový odběr je maximálně 100ms. V tomto stavu má zařízení výchozí adresu 0, kam pomocí řídicího přenosu zašle hostitel novou adresu. Poté si přečte první bajty deskriptoru zařízení, aby zjistil, jakou délku mohou mít datové pakety. Nyní si hostitel vyčte všechny další deskriptory zařízení, z nichž zjistí možné konfigurace zařízení a jednu z nich mu přiřadí. Zařízení tedy přešlo do zkonfigurovaného stavu.

Během konfigurace zařízení prochází *připojené* (zařízení je připojeno, ale ne napájeno), *napájené* (zařízení je napájené), *obecné zařízení* (zařízení je resetováno a má výchozí adresu 0), *adresované* (zařízení byla přidělena hostitelem adresa), *zkonfigurované* (zařízení přijalo nastavenou konfiguraci).

3 MIKROKONTROLÉR

3.1 Volba mikrokontroléru

Stěžejní komponenta převodníku je řídící mikrokontrolér. Při jeho výběru je třeba brát ohled na jeho periferie, velikosti programové a RAM paměti a maximální takt mikrokontroléru.

Hledisek pro posouzení je samozřejmě více, ale ty nejdůležitější pro moji aplikaci jsou:

- Ethernetové rozhraní
  - Mikrokontrolér má implementovanou MAC i PHY
    V tomto případě je použití velmi výhodné z hlediska úspory místa, minimalizaci chyb při návrhu a jednoduššího firmware mikrokontroléru.

  - Mikrokontrolér má implementovanou MAC
    Nejčastější případ vybavení mikrokontrolérů pro síťovou komunikaci. K tomuto je třeba připojit externí fyzickou vrstvu PHY, která bude s mikrokontrolérem komunikovat po výše popsaných interface MII (RMII). V tomto případě je složitější návrh DPS.

  - Mikrokontrolér nemá implementované rozhraní pro ethernet
    U těchto mikrokontrolérů je nutný externí ethernetový řadič. Toto řešení není vhodné z hlediska úspory místa na DPS a množství softwarové režie.

- USB rozhraní
  - Pokud má mikrokontrolér implementovaný USB řadič, tak v převážné většině umožňoval více druhů komunikace. Nejpoužívanější jsou USB Device, USB Host, USB OTG.

- Ostatní interface
  - Vzhledem k tomu, že se přepokládá využití navržené DPS jako vývojové desky pro další aplikace, musel jsem při výběru procesoru zohlednit další komunikační a vstupně/výstupní interface. Je třeba, aby zvolený procesor měl implementované rozhraní RS-485, CAN, I/O porty, A/D převodníky

- Velmi důležitým rozhraním je programovací a debuggovací typu JTAG,SWD.

- Softwarová podpora
  - Velmi výhodná je softwarová podpora výrobce mikrokontroléru. Díky univerzálně vytvořeným knihovním funkcím lze jednoduše používat všechny periferie mikrokontroléru. Existují firmy, například STMicroelectronics, které vydávají komplexní příklady Eth_stack, USB_stack a jiné, které lze bez větších problému upravit a použít pro vlastní aplikaci.

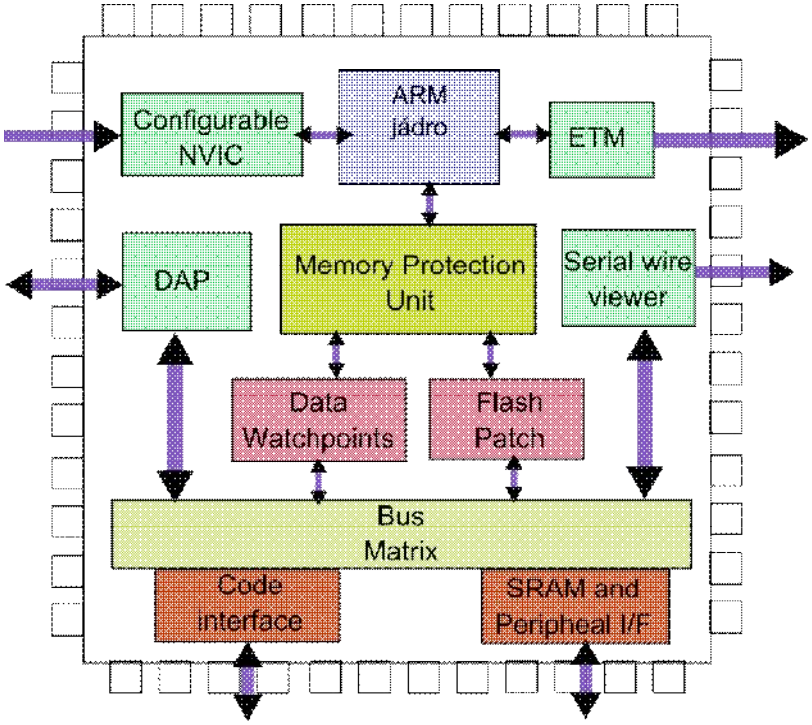
- Cena a výkon
  - Je třeba zvážit také finanční stránku v poměru s nabízeným výkonem jednotlivých, uvažovaných mikroprocesorů.

Z předchozích hledisek pohledů na pro výběr procesoru se jeví jako výhodný procesor, který má implementovanou Ethernetovou MAC vrstvu, USB řadič a další komunikační periferie. V současné době se na trhu prosazují mikrokontroléry s jádrem ARM, které už je velice rozšířené mezi výrobci procesorů. Mezi klasické výrobce patří STMicroelectronics, Atmel, Samsung, Texas Instruments, NXT.

Na základě těchto předpokladů byl vybrán mikroprocesor společnosti STMicroelectronics STM32f107, který je postavený na jádře Cortex M3.

3.2 Architektura mikroprocesorů Cortex M3

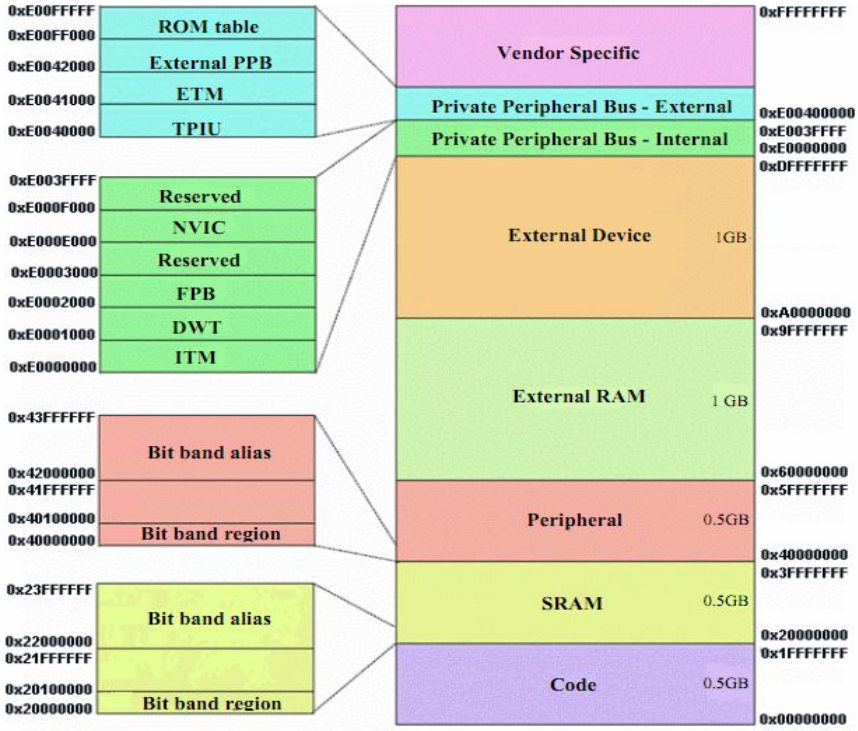
Rodina procesorů založená na architektuře ARMx7, která se dělí na 3 základní typy: ARMv7 je pro velice náročné aplikace, které běží na složitých operačních systémech, dále typ R, který je určen pro real-time aplikacích a nejrozšířenější typ M, který je optimalizován pro použití v levných embedded zařízeních. Efektivní 32bitový Cortex-M3 je použitelný pro průmyslové řídicí systémy, automobilovou elektroniku, drátové i bezdrátové komunikační systémy a podobně.



Obrázek 11: Blokové schéma procesoru ARM Cortex M3 [12]

Procesory Cortex-M3 mají k dispozici dekodér pro standardní i pro novou sadu příkazů (instrukční soubor Thumb-2), která umožňuje lepší využití především paměti, rozhraní, řídící logiky a tím programátorům umožní dosažení až o 70% vyššího výkonu na 1MHz. Jádro je založené na Harvardské architektuře a může využívat 3 úrovňový pipeline a tím je schopen vykonávat několik instrukcí současně.
Jádro obsahuje 13 registrů pro uživatelské použití, registr vazby, programový čítač, dva ukazatele zásobníků, stav rejstříku a sadu speciálních registrů. ALU také podporuje hardwarové násobení a dělení. Jádro podporuje dva operační režimy – Thread, Handler a dvě úrovně přístupu ke kódu privilegovaný – bez privilegií.

Procesor má jednoduchý pevný paměťový prostor s maximálním adresovatelným prostorem až 4GB. Součástí paměti je pevně zamapovaná paměť pro Flash paměť, pracovní RAM paměť, paměťová místa pro externí paměť RAM, externí zařízení – Periferie. Rozdělení paměťového prostoru je na následujícím obrázku. [6]



Obrázek 12: Mapování paměťového prostoru architektury Cortex M3 [6]

3.3 Základní specifikace STM32F107 [6]

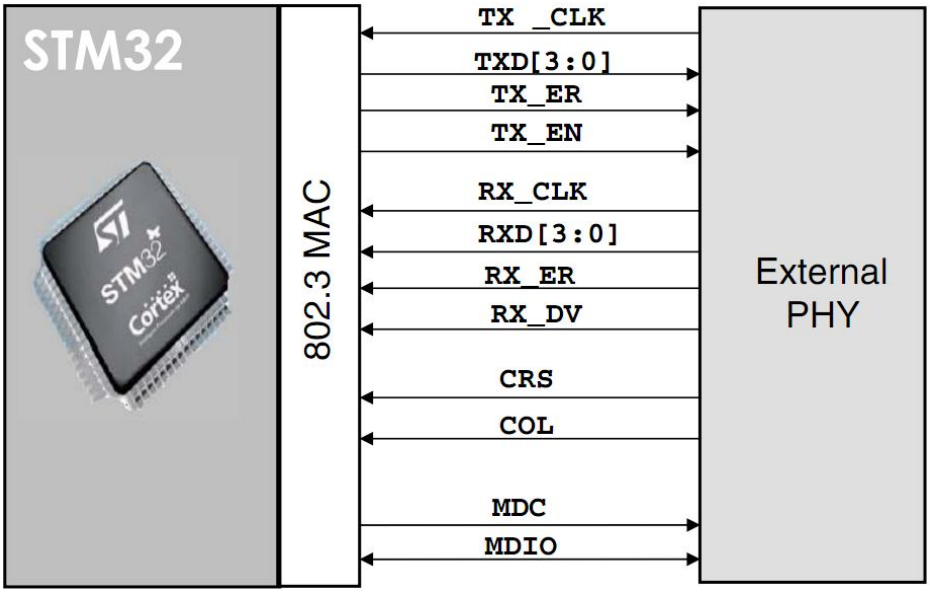
- Architektura ARMv7 – Harvardský koncept, 0.18 micron. technologie
- Instrukční soubor Thumb, Thumb-2
- 3 - úrovňová pipeline
- Maximální frekvence 72MHz
- Výkon 1.25 DMIPS/MHz, napájení 2.0 až 3.6V
- 4GB adresovatelné paměti
- 32 bitové instrukce, podpora hardwarového dělení a násobení (v jednom cyklu, znaménkové 2-12 podle velikosti operandů)
- Konfigurovatelných 1 až 240 přerušení s 8 až 256 úrovněmi priorit
- MPU – Ochrana paměti
- 256Kb Flash paměti, 64Kb Ram paměti
- Periferie: 2x12b A/D, D/A převodník, 12 kanálový DMA řadič, JTAG interface, 80 I/O pinů, 16b časovač, 2x Watchdog, 2xI2C, 5 USART, 3xSPI, 2xCAN, USB 2.0, 10/100 Ethernet
- Za zmínku také stojí obvody Watchdog, které hlídají činnost obvodu a hlídač podpětí.

3.4 Rozhraní ethernet

3.4.1 Základní vlastnosti rozhraní

- Vybraný procesor STM32f107 podporuje dva základní standardy vnitřního propojení MAC rozhraní s fyzickou vrstvou ethernetu - MII a RMII.
- Ethernet je kompatibilní s normami IEEE 802.3-2002 pro ethernet MAC a IEEE 1588-2002 standart pro hodinovou synchronizaci.
- Podporuje rychlost výměny dat 10/100Mbit/s. Podpora full-duplex i half-duplexního režimu přenosu dat.
- Vytváří ethernetový rámec, tak jak je popsáno v předchozí kapitole tj. přidání preambule, SFD, provádí automatickou kontrolu CRC. Má programovatelnou délku rámce.
- Dále má konfigurovatelných několik úrovní kontroly odeslaných, přijatých rámců.

3.4.2 Media independent interface MII



Obrázek 13: Signály propojující MAC s fyzickou vrstvou pomocí MII [5]

Specifikaci je možné rozdělit do 3 skupin. Signály určené pro přenos datového toku[5]:

MII_TX_CLK: Hodinový signál, který slouží pro synchronizaci vysílání dat. Nastavení je 2.5 MHz pro rychlost 10Mbit/s a 25MHz pro 100Mbit/s
MII_RX_CLK: Hodinový signál, který slouží pro synchronizaci přijímání dat. Nastavení je 2.5 MHz pro rychlost 10Mbit/s a 25MHz pro 100Mbit/s

MII_TX_EN: Indikuje zapnutí vysílání, generuje se zároveň s prvním niblem preambule (synchronně s vysílacími hodinami) a do klidové úrovně se uvede, jakmile jsou vyslány všechny nibbly.
MII_TXD[3:0]: Data pro vysílání jsou svázána do 4 datových vodičů (nibble), které jsou řízeny synchronně, a vystavovány jako platná data právě signálem MII_TX_EN. Pokud nebudou tímto signálem potvrzena, tak nebudou přenesena. Platí, že MII_TXD[0] je nejméně významový bit a MII_TXD[3] je nejvíce významový bit.
MII_RXD[3:0]: Signály pro přenos přijímaných dat jsou uspořádány podobně jako pro vysílání. Opět mají potvrzovací signál – platná data MII_RX_DV, který je generován opět synchronně. Nejméně a nejvíce významové bity jsou uspořádány stejně jako pro vysílání. Pokud je signál MII_TX_EN pro vysílání shozený a signál MII_RX_ER v aktivní úrovni, tak jsou signály pro příjem MII_RXD[3:0] řízeny fyzickou vrstvou.
MII_RX_ER: Signál, který je aktivní více než jeden clock hodin a indikuje chybu v přijímaném rámci. Stav přijímaného rámce, případně chybu lze rozeznat podle vzájemné kombinace signálů ER a DV a vystaveného niblu.

Další skupina jsou signály, které sledují stav komunikační linky:
MII_CRS: Signál je aktivován fyzickou vrstvou, pokud je příjem nebo vysílání neaktivní. Pokud jsou neaktivní obě akce a zároveň je v klidové úrovni. Fyzická vrstva zajišťuje aktivaci signálu po dobu trvání kolize přenosu dat. Toto platí v případě jednostranného přenosu, v případě obousměrného přenosu tento signál není významný. Signál nemusí být synchronizovaný hodinovými signály vysílání, příjem.
MII_COL: Signál, který generuje fyzická vrstva po dobu trvání kolize přenosu. Opět není synchronizován.

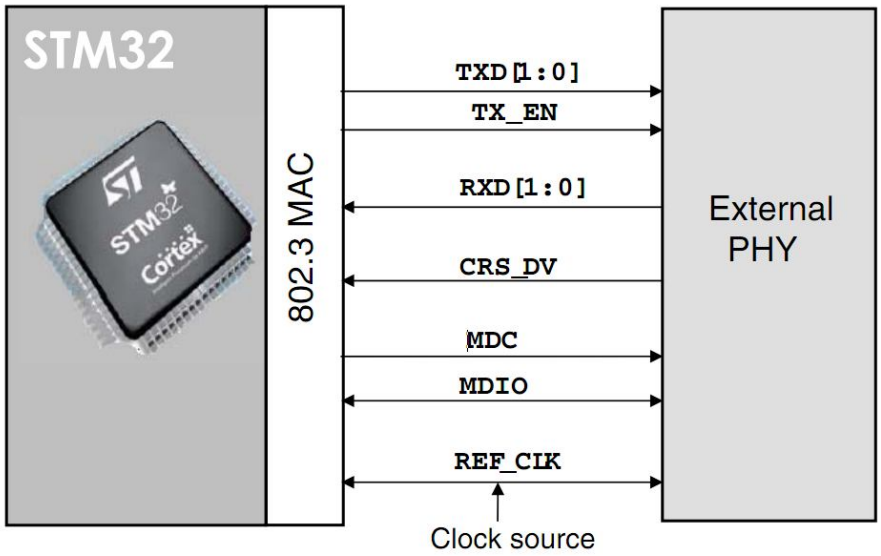
A poslední skupina jsou signály pro komunikaci mezi MAC vrstvou a PHY označovaná jako SMI (Station management interface). Komunikace je obousměrná a slouží k přenosu konfigurační, řídicích a stavových informací. Jedná se o synchronní linku, kde MDC je periodický Clock na frekvenci 25MHz a MDIO je datový vodič.

Výběr mezi MII a RMII se provádí nastavením bitu MII_RMII_SEL v registru AFIO_MAPR. Nastavení se provádí během resetu ethernetu nebo před zapnutím hodinového signálu.

Hodinový signál je generován externím oscilátorem o frekvenci 25MHz a dále prochází děličkou /2, /20 podle požadované rychlosti ethernetu.

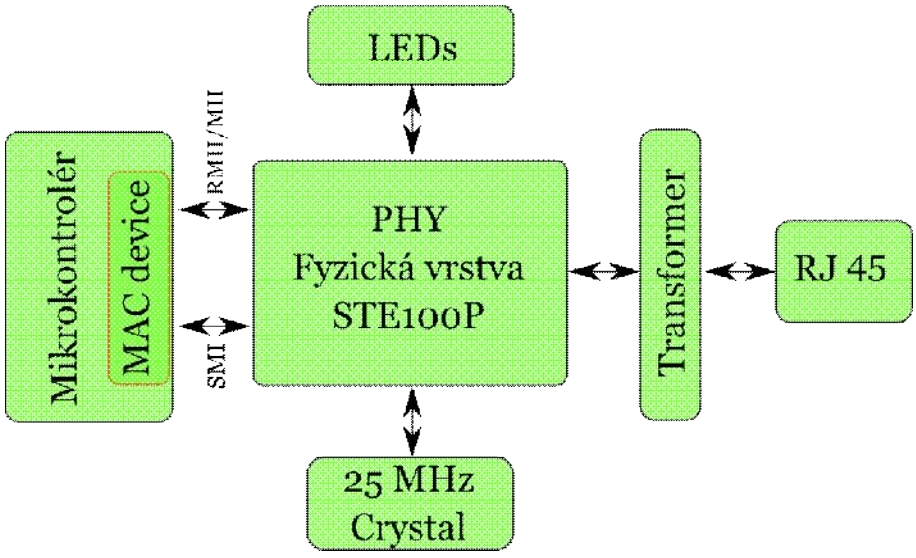
3.4.3 Reduced media independent interface RMII

Rozdíl mezi MII a RMII je v počtu pinů propojující MAC podvrstvu s fyzickou vrstvou Ethernetu. V případě MII, který používám pro převodník, je zapotřebí 16 pinů a v redukovaném MII pouze 7. Další rozdíl je ve frekvenci hodin pro přenos dat. Pro MII 25MHz a pro RMII 50MHz.



Obrázek 14: Signály propojující MAC s fyzickou vrstvou pomocí RMII [5]

Při výběru fyzické vrstvy ethernetu jsem vybíral z nabídky firmy STMicroelectronics, z důvodu co největší kompatibility s použitým mikroprocesorem. Vybraná fyzická vrstva STE100p bude popsána v kapitole 7.2.



Obrázek 15: Blokové schéma MAC a PHY [5]

3.4.4 Nastavení ethernetového rámce pro STM32f107

Preambule je 7 bytů a je určená pro synchronizaci. Nastavený pattern je hexadecimálně: 55-55-55-55-55-55-55. Následuje 1byte dlouhý startovací pattern: D5. Dále již popisované zdrojové a cílové adresy. U cílové adresy nejméně významový bit značí: 0 – jeden příjemce, 1 – skupina příjemců (ale i žádný). Druhý bit rozhoduje, jestli je cílová adresa spravovaná lokálně – 1 nebo globálně – 0. Pro broadcast odesílání (odesláním všem zařízením) se nastavuje 1.

Pro tzv. označený rámec je možné v této části nechat vložit 4byte dlouhé sekvence, které upřesňují prioritu uživatele, CFI, Vlan indetifikátor. Další část rámce je již povinná a je 2 Byte dlouhá. Může nabývat dvou hodnot. Pokud je velikost dat větší než 1500bitů, tak bude indikovat číslo přijaté sekvence dat (posílané data jsou větší a tedy rozděleny a opatřeny indetifikátory). Druhá hodnota může být pro případ, kdy se liší typ odesílaných dat, potom značí typ protokolu.

Nyní následují data a PAD. Data mají maximální délku 1500bitů a minimální v závislosti na tom, zda se jedná o označený nebo neoznačený rámec – 46 – 42bytů. Minimální velikost rámce je daná z důvodu spolehlivé detekce kolize – v přímé závislosti s konstrukčními parametry přenosového média. PAD potom doplňuje rámec do plné velikosti v případě menší velikosti dat než 1500bitů.
Poslední přidávaná část je CRC, což je 4byte dlouhý kontrolní součet. Do kontrolního součtu se zahrnuje vše kromě preambule a SFD. [5]

3.5 Rozhraní USB

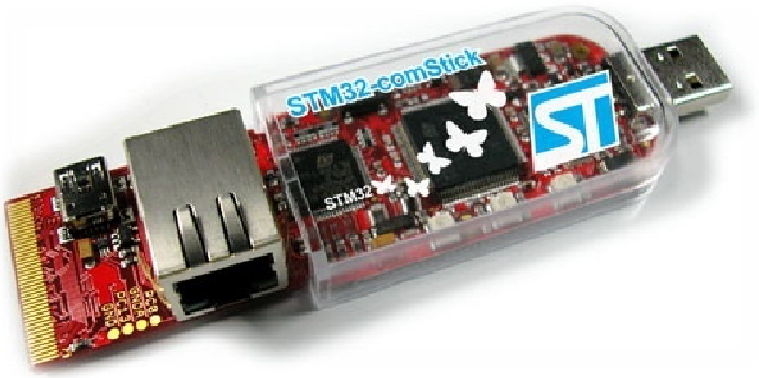
Mikroprocesor má implementovaný rozšířený standard USB – USB_OTG, který upravuje klasickou komunikace typu Master-Slave na možnost přímého spojení Point-to-point. Vznikl na základě potřeby propojovat zařízení bez přítomnosti zařízení (PC), které inicializovalo a udržovalo spojení. Hlavní rozdíl je tedy v tom, že USB_OTG zařízení se může chovat jako Host i Device. Protože při komunikaci musí být určené zařízení, které bude komunikaci řídit, byl zaveden další signál ID, který podle úrovně rozhoduje o tom, zda je zařízení Host nebo Device. Implicitně je ID pomocí vnitřního Pull-up rezistoru v logické 1, což znamená chování USB Host a naopak pokud se signál ID uzemní, zařízení bude jako Device.

4 IDE A VÝVOJOVÝ KIT

4.1 Vývojový kit

Aplikaci převodníku jsem nejdříve implementoval na dostupném vývojovém kitu od firmy Hitex s procesorem STM32f107 STM32ComStick. Výrobce ke kitu dodává několik ukázkových aplikací použití síťových prvků použitého procesoru. Komunikace s kitem probíhá přes USB2.0 rozhraní. Není třeba externího napájení, protože je napájený z USB, což znamená omezení maximálního odebíraného proudu z kitu na 350mA. Procesor má vyvedených 58pinů na 80pinovém konektoru atypického rozměru. Mezi dostupnými interface jsou CAN, UART, ADC, DAC, SPI, I/O a další standardní periferie. Přímo na desce jsou osazeny standardní konektory pro Ethernet RJ-45 s integrovanými oddělovacími transformátory a pro USB AB_Micro. Jako fyzická vrstva je použitý obvod firmy STElectronics STE100P.

Firma Hitex nabízí USB Sticky v několika verzích, které se liší převážně provedením šasi a konektorů. Dále také nabízí Evalution Boardy, které jsou podstatně na vyšší úrovni z hlediska možnosti vývoje aplikací. Evalution Board má ve výbavě procesor s jádrem Cortex-M4 s integrovanou fyzickou vrstvou Ethernetu, dále TFT LCD display, slot pro MicroSD paměťovou kartu, 3osý akcelerometr a gyroskop, obvody pro zpracování analogového signálu a mnoho dalších. K Evalution Boardu patří také softwarová podpora všech integrovaných periferií.



Obrázek 16: Vývojový kit firmy Hitex s procesorem STM32f107

4.2 Vývojové prostředí

Pro psaní a kompilování zdrojového kódu, nebylo možné použít IDE prostředí, dodané k vývojovému kitu, z důvodu omezení velikosti kódu. Protože projekt *Ethernet stack* několikanásobně převyšuje velikost 32kB, bylo třeba najít editor kódu a kompilátor, který nemá toto omezení. Těmto podmínkám vyhovovalo prostředí Ride7.

Protože jsem měl k dispozici originální programátor firmy ST pro jejich procesory ST-Link, bylo nutné najít prostředí, které tento programátor podporuje. Toto propojení je nutné k dalšímu vývoji a návrhu nových aplikací na vlastní navržené vývojové desce. Prostředí podporující ST-Link je vyrobila firma Keil – µVision.

4.2.1 Hitop

Hitop je vývojové prostředí dodávané firmou Hitex k výrobkům USB Stick. Používaná verze Hitop je omezena velikostí přeloženého kódu na 32kB. IDE má všechny standardní funkce, potřebné pro vývoj firmware procesoru. Je možné si vybrat ze tří podporovaných kompilátorů kódu – Keil, Tasking, GNU. IDE obsahuje integrovaný plnohodnotný debugger, který komunikuje pomocí druhého procesoru (programátor, debugger) osazeného na USBStick firmy Hitex, což je zároveň nevýhodou tohoto prostředí, že může programovat a debuggovat pouze výrobky firmy Hitex. Verze, kterou jsem dostal k vývojovému kitu, nebyla plně kompatibilní a bylo nutné při vytváření nového projektu, provádět úpravy konfiguračních souborů linkeru a ručně zadávat cesty ke zdrojovým souborům.

4.2.2 Ride7

Ride7 je vývojové prostředí firmy Raisonance, které nabízí všechny potřebné funkce pro vývoj firmware procesoru. Prostředí se zaměřuje na procesory rodiny ARM, ale i na 8bitové 8051(52) a využívá neomezený kompilátor GNU. Dále umožňuje programování a debuggování pomocí RLink programátoru. Ride7 je široce podporované firmou ST a proto nabízí množství ukázkových projektů volně použitelných. Naopak výhodou je opět velká softwarová podpora procesorů ARM, protože prostředí nabízí velké množství zkušebních projektů.

4.2.3 Keil µVision

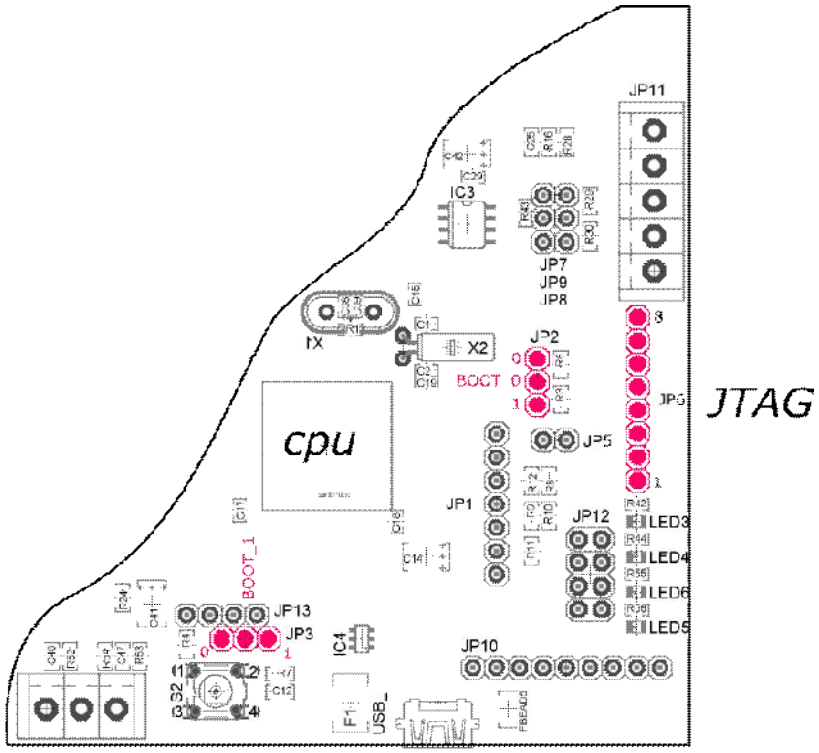
Keil µVision v4 spojuje funkce řízení projektů, editaci a laďění kódu a simulaci v jednom univerzálním prostředí. Keil podporuje velkou škálu procesorů včetně ARM a CortexM3, celý seznam je dostupný a stažitelný na [24]. Keil jsem vybral jako programovací a ladicí prostředí pro vlastní vývojovou desku v kombinaci s programátorem ST-Link. Nevýhodou je neumožnění integrace nástrojů třetí strany a možnost změny počáteční adresy paměti.

5 PROGRAMÁTOR

Programátorů procesorů ARM s jádrem Cortex-M3 je velké množství. Vlastní programátory nabízejí jak výrobci procesorů, tak i vývojových prostředí. Pro programování vlastní vývojové desky jsem zvolil zmíněný ST-Link od firmy STElectronics.

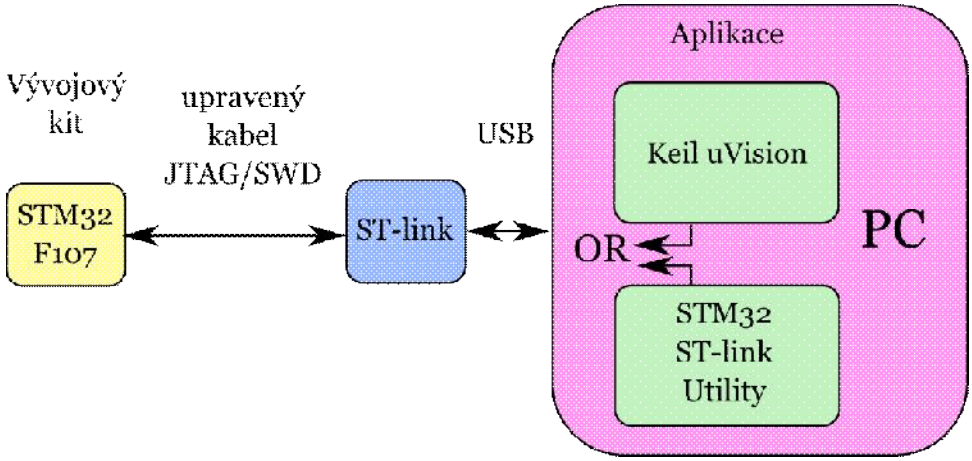
5.1 ST-Link

ST-Link je in-circuit debugger a programátor mikrokontrolérů rodiny STM8 a STM32. Pro komunikaci využívá SWIM nebo JTAG/SWD interface, kde SWD je speciální ladicí rozhraní, které je výhodnější z hlediska dvojnásobné rychlosti proti JTAG a z hlediska počtu použitých vodičů mikrokontroléru. JTAG využívá včetně signálů Vcpu, Reset a GND 8 vodičů proti 5 signálům rozhraní SWD. Nicméně na vývojovém kitu je konektor pro rozhraní JTAG s tím, že možné při použití SWD volné signály využít jiným způsobem. ST-Link je napájený 5V napětím z USB PC. Takovéto použití, je možné díky tomu, že signály SWIO a SWCLK jsou mapované na signálech JTAGu TMS a TCK.



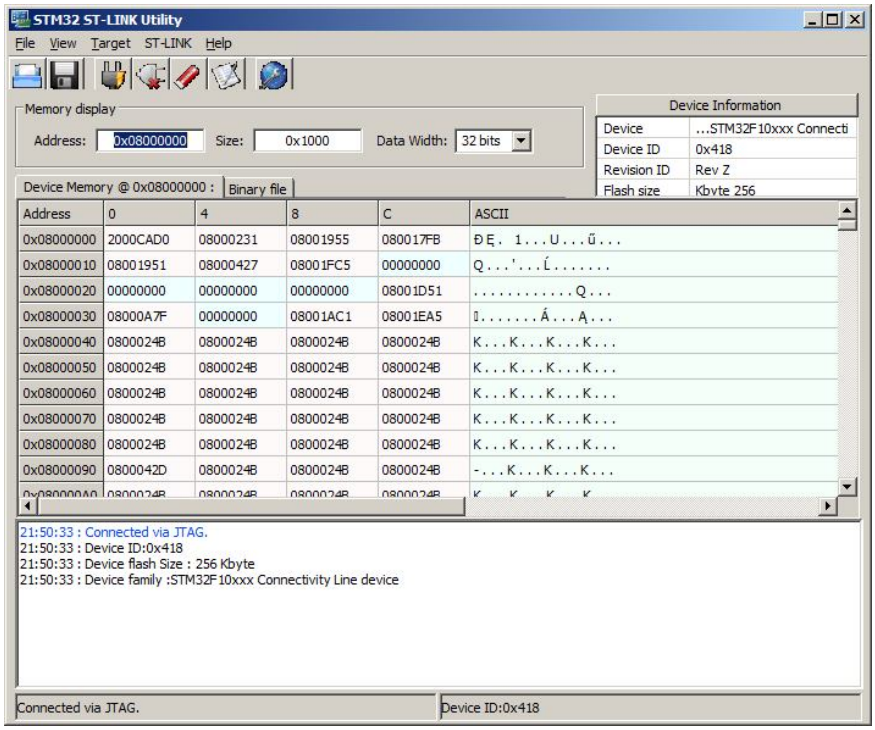
Obrázek 17: Umístění JTAG konektoru na vývojové desce

ST-Link používá 20 pinový konektor, který jsem v rámci úspory místa na DPS redukoval na potřebných 8 signálů. Ostatní vodiče se dle dokumentace [10] připojily na GND případně přes pull-down rezistor na GND. Základní stavy programátoru lze určit pomocí Status LED dle Tabulka 3.

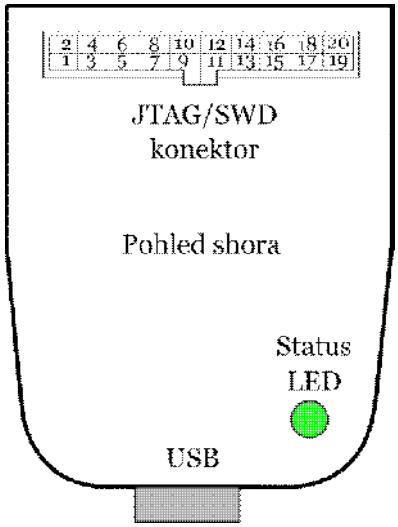


Obrázek 18: Blokové schéma připojení programátoru

V rámci aplikací v PC jsem otestoval popisované vývojové prostředí µVision a originální programovací prostředí firmy STMicroelectronics STM32 ST-link Utility jako programovací prostředky mikrokontroléru. Následující Tabulka 4 popisuje způsob propojení redukovaného kabelu pro připojení ST-Linku s vývojovou deskou dle specifikace ST-Link [10].



Obrázek 19: Screenshot připojeného programu STM32 ST-link Utility k mikrokontroléru



Obrázek 20: Horní pohled na ST-Link

Tabulka 3: Status LED ST-Linku [10]

Status LED	Stav ST-Linku
Zapnutá	Komunikace navázaná
Bliká	Probíhá výměna dat PC-zařízení
Vypnutá	Komunikace nenavázaná

Tabulka 4: Propojení ST-Link a vývojové DPS

Pin ST-Link	Signál	Kabel	Pin	Signál
1	Vcc MCU		1	Vcc_3.3
2	Vcc MCU		1	Vcc_3.3
3	TRST		3	TRST
4	GND	Připojení na GND	8	GND
5	TDO		5	TDI
6	GND	Připojení na GND	8	GND
7	TMS/SWIO		6	TMS/SWIO
8	GND	Připojení na GND	8	GND
9	TCK/SWCLK		7	TCK/SWCLK
10	GND	Připojení na GND	8	GND
11	NC	Pull-down R	8	GND
12	GND	Připojení na GND	8	GND
13	TDI		4	TDO
14	GND	Připojení na GND	8	GND
15	RESET		2	RST
16	GND	Připojení na GND	8	GND
17	NC	Pull-down R	8	GND
18	GND	Připojení na GND	8	GND
19	NC	Pull-down R	8	GND
20	GND	Připojení na GND	8	GND

6 SOFTWARE

6.1 Implementace Ethernet_stack

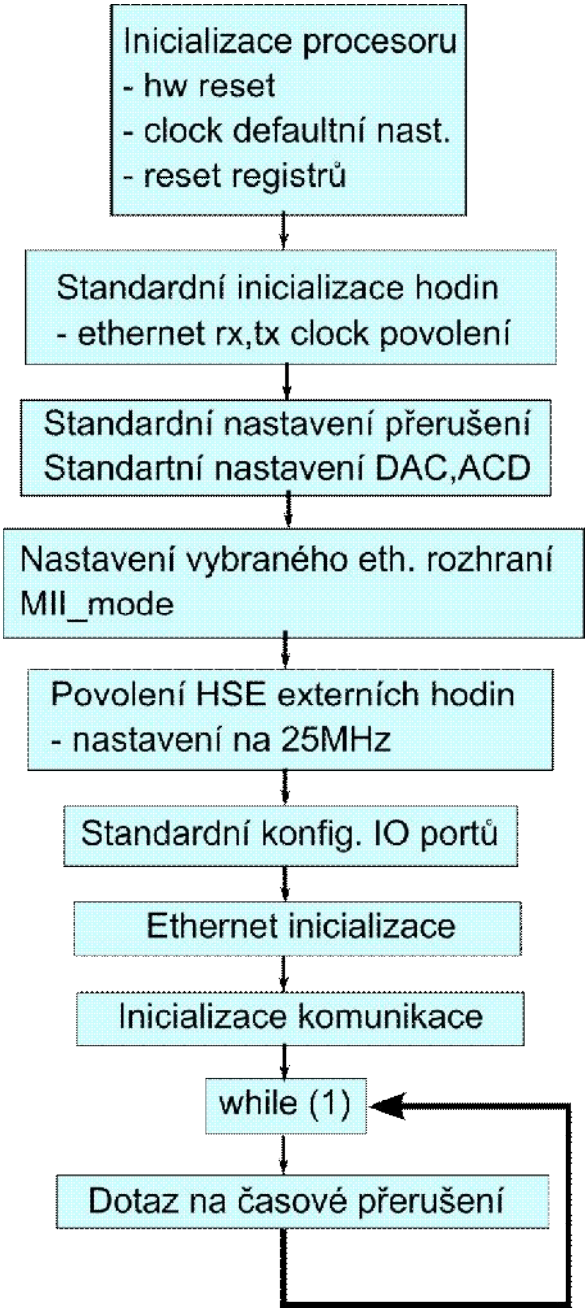
6.1.1 Softwarová část – Ethernetové rozhraní

Vzhledem k tomu, že tento typ procesoru s ethernet rozhraním je poměrně nový, tak není příliš mnoho aplikací s použitím ethernetu, které by byly dostupné. Takže nemám dostupný žádný příklad použití ethernetu pro tento procesor, odkud by bylo možné srovnat inicializaci a nastavení rozhraní. Nastavení ethernetu tedy dělám pomocí dokumentu „reference manual“, který je příručkou procesoru – popisuje možné nastavení, registry, elektrické vlastnosti jednotlivých periferií. K dispozici mám knihovnu pro práci s ethernetem od výrobce procesoru STMicroelectronics, kde jsou definované všechny potřebné registry, flagy i jednotlivé bity. Dále využívám standardní knihovny pro vstupně-výstupní porty, časovače, které přímo obsahují často používané rutiny – SW reset, inicializace časovačů, I/O.

Podle doporučeného postupu ST hlavní větev programu začíná inicializací procesoru, kde proběhne nastavení hodin procesoru – výběr a povolení HSE (externí vysokorychlostní hodinový signál), a nastavení na rychlost 25MHz – tuto rychlost vyžaduje vybrané propojení subvrstvy MII. Dále je nutné povolení Ethernet_clock, aby se začaly generovat signály TX_CLK a RX_CLK.

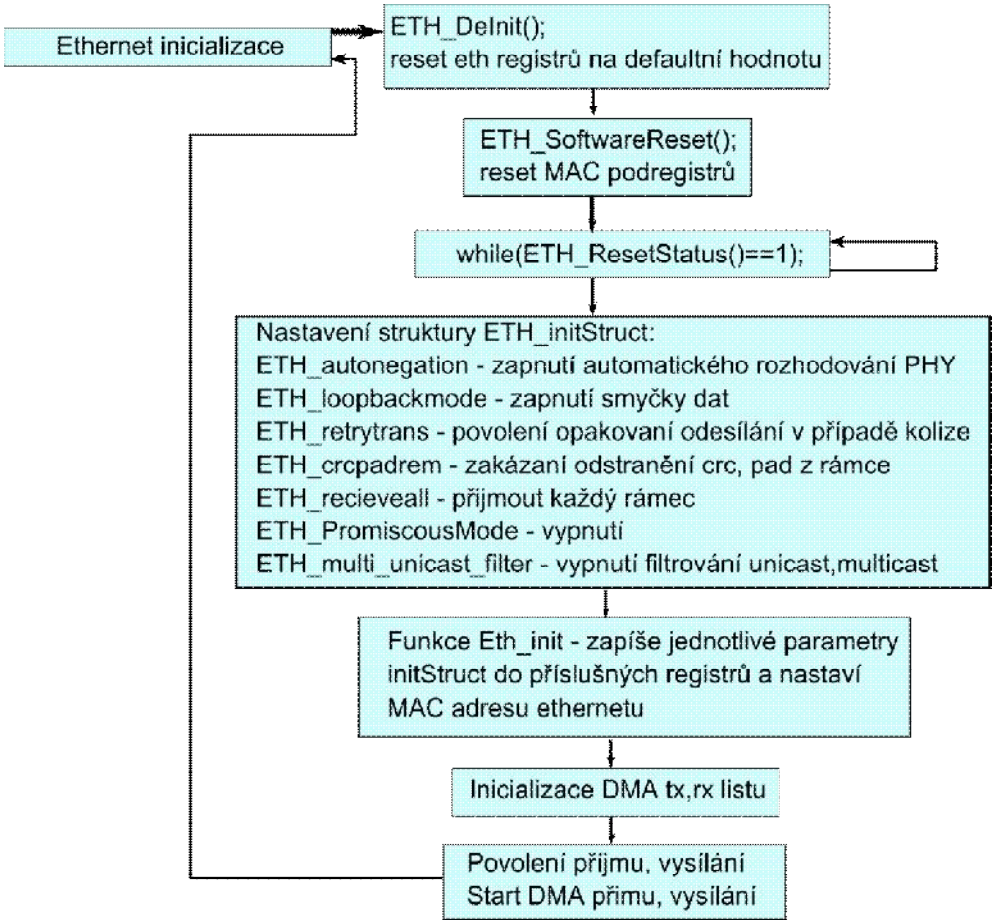
Dalším krokem hlavního programu je výběr Ethernet media interface (fyzické propojení vrstev ethernetu). Volám knihovní funkci *GPIO_ETH_MediaInterfaceConfig(GPIO_ETH_MediaInterface_MII)* s parametrem vybraného propojení. Současně se zavolá funkce, která spustí generování hodinového signálu pro ethernet *RCC_MCOConfig* s parametrem vybraného hodinového zdroje (RCC_MCO_HSE) – například pro vybraný Media interface RMII se používá jiný zdroj PLL3.

Posledním standardním nastavením je konfigurace GPIO, což jsou vstupně-výstupní porty. Volám funkci *GPIO_Configuration(void),* která nastaví mód, rychlost a číslo pinu (i pro přemapované zapojení – je totiž možné využít několik pinů pro stejnou funkci).



Obrázek 21: Vývojový diagram hlavní větve programu ethernet stack

V další části programu nastavuji Ethernet rozhraní. Zavoláním funkce *ETH_DeInit()* provedu vynulování všech registrů používaných ethernetem a jejich nastavení na defaultní hodnoty. Funkcí *ETH_SoftwareReset()* provedu reset a vynulování interních registrů MAC podvrstvy. Konfiguraci samotného ethernetu provádím nastavením proměnných struktury ETH_InitStructure, kterou poté funkce *ETH_Init()* rozebere a podle jednotlivých proměnných nastaví příslušné registry rozhraní. V struktuře (která je deklarovaná v headeru ETH knihovny) je možné podrobné nastavení podvrstvy MAC (například mód přenosu, aktivace loopbacku, rychlost, automatické generování kontrolního součtu) a dále potom nastavení DMA.



Obrázek 22: Vývojový diagram inicializace ethernetu

Po inicializaci ethernetu následuje samotné nastavení komunikace – které je popsané v další kapitole a hlavní větev programu končí v nekonečné smyčce *while*. Ve smyčce se dotazuji funkcí Systém_Periodic_Handle(); zda čas, který se inkrementuje po 1ms (přerušením časovačem), nabyl určité hodnoty – je využíváno pro další funkce.

6.1.2 Softwarová část - komunikace

Po inicializaci ethernetového rozhraní se může začít pracovat s komunikačními protokoly. Ke komunikaci na síti využívám standardní knihovny výrobce STMicroelectronics v kterých je implementovaná většina funkcí potřebná pro komunikaci. Protože jsou knihovny vzájemně propojené, není třeba se zabývat podrobným nastavením a prací s nižšími protokoly. Protože přímo využívám až vrstvu UDP, tak stačí pouze přilinkovat potřebné knihovny nižších protokolů a jejich stavba sama zajistí jejich vzájemnou práci. Každému zařízení na síti je nutné přiřadit adresový prostor, což je IP adresa zařízení, maska sítě a IP adresa výchozí brány. Pro každé zařízení se vytvoří datová struktura *netif*, která obsahuje všechny důležité informace o zařízení.

Nejdůležitější parametry a jejich nastavení je následující:

- Struktura ip_addr
- Deklarace funkcí pro příjem a vysílání paketů
- Popisnou zkratku zařízení
- Ukazatel na další strukturu netif
- Různé kontrolní součty hlavičky, adresy

Inicializace se provede zavoláním funkce *LwIp_Init()*. Pokud se nebude jednat o server, který bude přidělovat adresy pomocí DHCP, tak ve funkci naplním datovou strukturu *ip_addr*, kde IP je v mém případě 192.168.0.8, maska sítě 255.255.255.0 a výchozí brána 192.168.0.1. V případě DHCP se nastaví hodnoty na 0 a provede se definování příslušného makra. Dalším důležitým nastavení je nastavení MAC adresy zařízení. MAC adresa zařízení (hardwaru ethernetu) je unikátní, avšak ComStick má adresu definovatelnou, což je rozhodně velká výhoda. MAC adresa je nastavena bezúčelně na hodnotu 0.0.0.0.0.1. K zápisu do struktury se použije funkce *Set_Mac_Adress(macaddress)*. Posledním krokem základní inicializace je zavolání funkce *netif_add(params)*, která naplní zmíněnou strukturu *netif* a přidá ji na list registrovaných zařízení.

Poslední inicializační funkcí je *server_init(void)*. Funkce obsahuje vytvoření struktury *udp_pcb*, která je nutná pro samotnou komunikaci přes udp protokol. Struktura obsahuje číslo portu, na kterém bude protokol naslouchat, číslo portu na který se budou odesílat data a callback funkci. S touto strukturou pracují všechny funkce UDP protokolové vrstvy. Následuje vytvoření soketu pomocí funkce *udp_bind(params)* s parametry *udp_pcb*, makrem IP_ADDR_ANY, UDP_SERVER_PORT. Což znamená, že budu přijímat zprávy z jakékoliv IP adresy na smluveném portu. Zbývá ještě nastavit akci, která se provede po přijetí datového paketu. To se provede nastavením pomocí funkce *udp_recv(params)*, která definuje funkci, která se zavolá a bude zpracovávat přijatý udp paket. Funkce pro zpracování paketu se jmenuje *udp_server_callback*.

Jednotlivé funkce protokolu UDP:

- Udp_new – vytvoření struktury udp_pcb
- Udp_remove – odstranění struktury udp_pcb
- Udp_bind – vytvoření soketu pro připojení
- Udp_connect – připojení k cílové IP adrese
- Udp_disconnect – odpojení
- Udp_recv – příjem dat z připojené IP adresy
- Udp_send – odeslání dat na připojenou IP adresu

6.1.3 Implementace Server - Klient

Komunikace Server – Klient probíhá tak, že určený server na síti, rozesílá žádosti jednotlivým zařízením připojených k síti. V našem konkrétním případě odešle žádost (datový paket), ve kterém je dle protokolu FektBot určené, zda bude číst sekvenční část dat nebo jen číst na určené adrese. Při sekvenčním čtení dat – Seq_Read se určí startovní adresa v registru a druhým bytem se určí požadovaný počet registrů ke čtení (jak bylo uvedeno - nepovinný byte v protokolu, je platný pouze pro Seq_Read). Pro požadavek Read se vyčítá pouze žádaná adresa registru. Při příchodu datového paketu je pomocí přerušení na ethernetovém rozhraní zavolána popisovaná funkce udp_server_callback. Této funkci se mimo jiné předají přijatá data, IP adresa z jaké byl paket vyslán a číslo portu na který byl paket doručen. K tomu, aby se rozluštil požadovaný příkaz, se implementovala funkce parse, která doručený udp paket rozdělí na hlavičku a data a dále pak provede potřebné operace s daty a paket odešle na danou adresu a port. Po provedení parsování a odeslání dat je nutné se připravit na nové navázání spojení a to zavoláním funkce udp_bind a udp_recv. Důležité je uvolnit buffer, ve kterém byla přijatá data, pomocí funkce pbuf_free(buf) , která je v knihovně pro práci s bufferem. V mém projektu takto pracuje komunikace mezi „monitorem“ a zařízením na síti. V reálné aplikaci bude program zaměněný za servera.

6.1.4 Implementace Klient – Server (Klient)

Jak jsem zmiňoval, bude nutná i zpětná komunikace mezi klientem a serverem. Například požadavek na směr, akci nebo jiná data. O tyto data může žádat servera nebo přímo jiné zařízení. Vyslání požadavku může být řízeno několika způsoby. Prvním způsobem je na požadavek některé z periferií – například senzor, který je připojený na vstupním portu procesoru, zaznamená překážku a pro ověření se může zeptat jiného sebelokalizačního zařízení na stav jeho registrů (které si aktualizuje dle skutečnosti). Pro parametry, které je nutné stále aktualizovat (zmiňované měniče motorů) je možné využít časovač procesoru a cyklicky odesílat žádosti. Tímto způsobem jsem žádání implementoval do procesoru s periodou 250ms. Žádání se provede zavoláním funkce posli_dotaz (params).

6.1.4.1 Sestavení žádosti

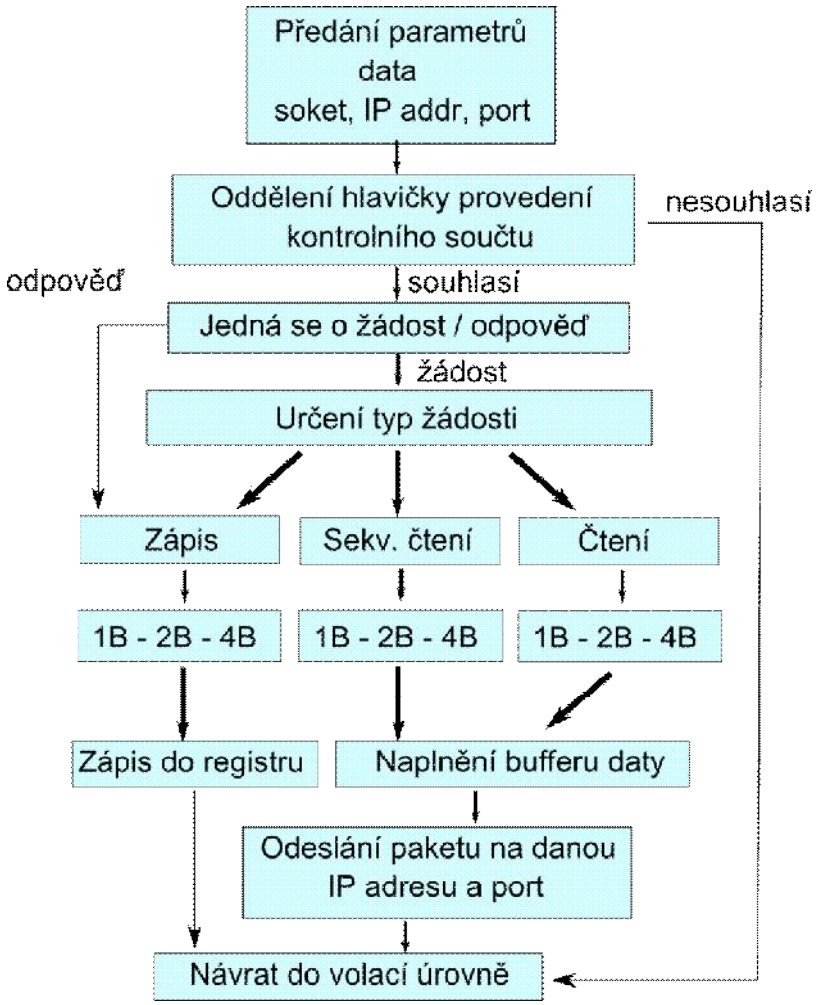
Pro poslání žádosti je nutné definovat adresní prostor a konkrétní IP adresu kam se žádost pošle (je možné použít broadcastovou adresu). Vytvoří se nový soket a datová struktura p_buf, kam se uloží data pro odeslání. Následuje sestavení samotné žádosti, které se řídí parametry protokolu FektBot a žádost obsahuje jak hlavičku, tak i samotná data. V protokolu je též implementován zápis, což je pouze jednosměrná komunikace a pouze odešleme data a místo kam se zapíšou. Dle velikosti žádosti se alokuje paměť v RAM paměti procesoru. Opět pomocí funkce knihovny pro práci s bufferem se buffer naplní žádostí. Funkcí *udp_connect* se připojíme k cílové IP adrese na definovaný port. Port je pro celou aplikaci shodný, jeho hodnota je prakticky neomezená, ale pro případ, že by na sítí probíhaly další služby, je vhodné se vyhnout již používaným portům. Číslo portu může být až 2 B velké a pro aplikaci je použité 10005. Nyní už je možné odeslat příslušný paket funkcí udp_send. Pro ukončení spojení se použije funkce udp¬¬¬_disconnect. Zbývá jen odstranit vytvořený soket a buffer.

6.1.5 Funkce *void parse(params)*

Jednou z důležitých funkcí je *void parse(params)*, která zajišťuje parsování dat udp paketu. Funkce oddělí hlavičku, kterou celou zkopíruje pro případ, že se jedná o žádost a je třeba odpovědět. Při odpovědi se v hlavičce mění bit reply na hodnotu 1 a dále velikost odesílaných dat, jinak hlavička zůstává nepozměněna. V hlavičce zkontroluje indetifikátor protokolu a velikost dat udávaných a skutečně přijatých. Je třeba dát pozor na 2-3 první byty dat, které jsou ještě řídící, ale již se počítají do velikosti dat. Funkce nemá prozatím ošetřený stav, kdy velikost dat nesouhlasí s údajem v hlavičce.

Vzhledem k periodickému charakteru zpráv, nebude nutné posílat zpětně zprávu se žádostí o opakování. Následuje určení požadované operace s daty, což se provede pomocí maskování 1 bytu dat. Jak jsem již zmínil, možnosti jsou Read – což je čtení jedné buňky registru, dle doručeného ID neboli adresa buňky v rozsahu 0x00 až 0x3F, Seq_Read – čtení určeného počtu buněk od ID, Write – zápis na dané ID. Každá z operací může pracovat s daty po 1,2 nebo 4 bytech, což určuje nižší nibl maskovaného bytu. V případě zprávy typu odpověď se rovnou provede zápis do vlastního registru na žádanou adresu. V případě žádosti se naplní paket pro odeslání požadovanými daty.

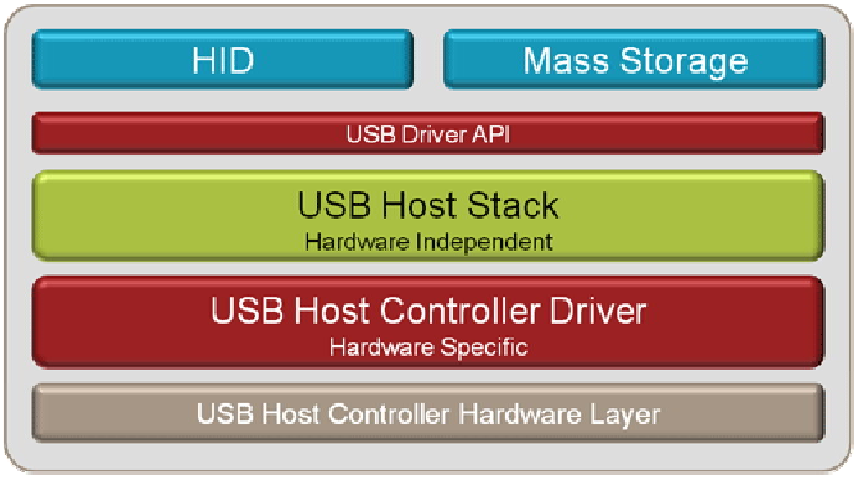
Posledním úkolem funkce je navázat spojení s předanou IP adresou a na daný port odeslat datový paket (v tomto případě hlavička a data). Funkci je možné dále rozšiřovat o jiné operace s daty.



Obrázek 23: Vývojový diagram funkce PARSE (PARAMS)

6.2 Implementace USB_Stack

Pro implementaci USB protokolu jsem použil částečně zdrojový kód firmy Keil, který je obsažený ve freeware verzi programovacího prostředí µVision. Knihovny programu jsem rozdělené do několika vrstev, podle toho do jakých částí obsluhy spadají. Grafické znázornění struktury programu je převzato od firmy Keil [12].



Obrázek 24: Grafické znázornění struktury program USB Host [12]

Knihovny, které jsou použité pro komunikaci s nejnižší vrstvou – to je přímo obsluha USB řadiče a konfiguračních registrů, jsou dodané firmou STMicroelectronics, která vyrábí použitý procesor. Kompletní dokumentace **USB OnTheGo host and device library** je dostupná na stránkách firmy STMicroelectronics [8].

Rozhodl jsem se upravit ukázkový program v nejvyšší vrstvě – HID, který je napsán univerzálně pro základní práce s klávesnicí a joystickem. V následujícím textu popíšu základní funkce nejdůležitějších knihoven aplikační vrstvy HID

6.2.1 HW_CONFIG

Knihovna obsahuje důležitou funkci pro celý procesor *void Set_system(void)*, která inicializuje hlavní systémové hodiny periferních sběrnic APB, všechny vstupně/výstupní porty používané USB sběrnicí a povolení, nastavení priorit USB přerušení.

6.2.2 USB_CONF

Knihovna **usb_conf.h** definuje pomocí makra počet koncových bodů zařízení, ke kterému je nutné započítat koncový bod pro řídicí přenos. Dále jsou zde definice velikostí vysílacího/přijímacího bufferu USB jádra. Tato paměť se dělí mezi připojené koncové body. Poslední jsou definice funkcí, které jsou volány pro požadavky na příjem nebo vysílání na daný endpoint **EPx_IN_Callback** a **EPx_OUT_Callback**. V knihovně **usb_conf.h** je také možné konfigurovat další rozšiřující události rozhraní, které vyvolají přerušení.

6.2.3 USB_DESC

V knihovně **usb_desc.h** jsou všechny potřebné definice, týkající se deskriptorů joysticku. Definice udávají ID a velikosti v bajtech dílčích deskriptorů popisovaných

v kapitole 2.3: deskriptor zařízení 0x01, deskriptor konfigurace 0x02, deskriptor řetězců 0x03, deskriptor rozhraní 0x04 a deskriptor endpointu 0x05.

Knihovna **usb-desc.c** obsahuje struktury formou 8 bitového pole: *Joystick_DeviceDescriptor, Joystick_ConfigDescriptor, Joystick_StringLangID, Joystick_StringVendor, Joystick_StringProduct, Joystick_StringSerial.* Struktury jsou přednastavené standardními hodnotami. Deskriptory se vyplní podle požadované funkce zařízení a pole PID a VID jsou unikátní sériová čísla nastavená výrobcem zařízení.

6.2.4 USB_ISTR

Hlavní funkce knihovny **usb_istr.c** void *USB_Istr(void)* je volána obsluhou přerušení při přijmu dat z USB. Obsluha přerušení *void USB_LP_CAN_RX0_IRQHandler(void)* se nachází v knihovně, která má na starosti všechny přerušení periferií a událostí procesoru **stm32f10x_it.c**.

6.2.5 USB_PROP

Knihovna obsahuje funkce, které přímo pracují s připojeným joystickem pomocí konfiguračních přenosů. Do konfiguračních paketů patří funkce, které se dotazují a nastavují deskriptory zařízení, dále provádí reset zařízení, a kontrolu statusu zařízení.

6.2.6 USB_INIT

Usb_init.c je knihovna ze souboru knihoven, obsluhujících USB jádro a definuje jednu z nejdůležitějších funkcí **void USB_Init(void)**. Tato funkce naplní struktury *pInformation* a *pProperty* informacemi o stavu linky a aktuální konfiguraci zařízení.

6.2.7 USB_PWR

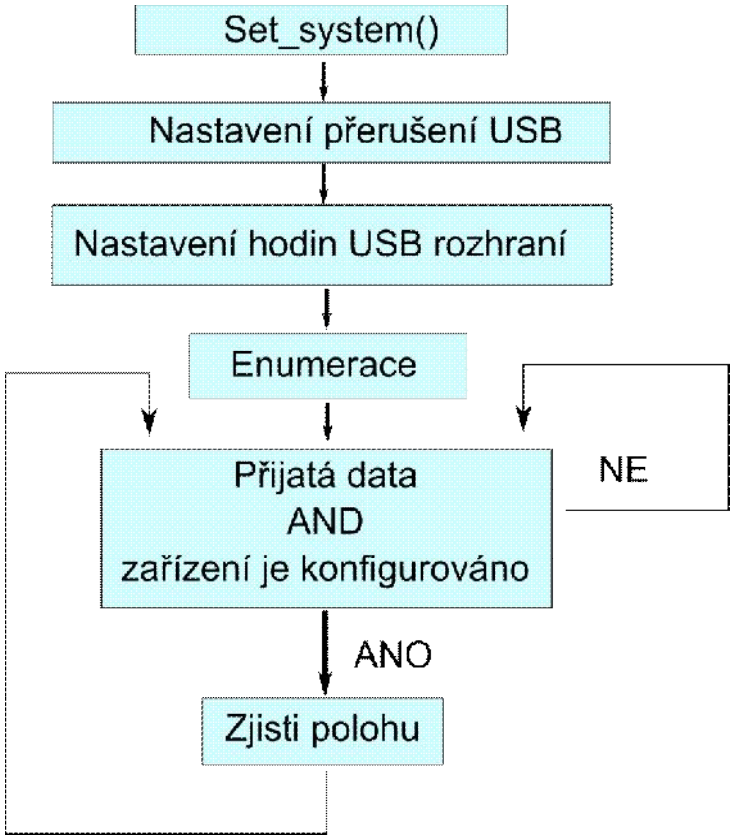
Knihovna obsluhuje napájení, případně pozastavení, probuzení sběrnice. Knihovna **usb_pwr.h** definuje stavy zařízení, do kterých se během celého programu může dostat.

6.2.8 DIODY

Knihovnu *Diody* jsem vytvořil pro ovládání digitální výstupů procesoru, konkrétně připojených LED diod. LED diody jsem používal k usnadnění ladění a také jako indikátory stavu při programování aplikace na vývojové kitu ComStick. Snadnou úpravou knihovny je možné ovládat kterýkoliv vstupní/výstupní pin. Před použitím funkcí knihovny **void nastav(u32 dioda)** a **void zhasni(u32 dioda)** je třeba v hlavní větvi programu zavolat funkci **void IO_Init(void)**, která inicializuje potřebné registry.

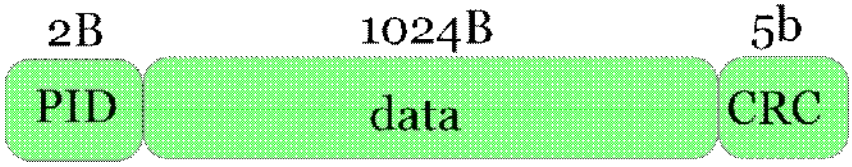
6.2.9 MAIN

Soubor **main.c** obsahuje hlavní větev programu funkci **int main(void)**. Ve funkci jsou použité výše popsané funkce. Konce funkce je zacyklený v nekončeném cyklu *while(1)*, uvnitř kterého se dotazuji, zda byla přijata data a zároveň je zařízení stále ve stavu *Configured*. Pokud jsou podmínky splněny, dekóduji přijatý datový paket pomocí funkce **u8 JoyState(void)**. Data jsou přijata do přijímacího bufferu ve struktuře *pInformation*.



Obrázek 25: Vývojový diagram hlavní větve programu USB Host

6.2.10 Dekódování datového paketu joysticku



Obrázek 26: Datový paket Joysticku

PID je identifikátor paketu o velikosti 4bity a pro možnost kontroly správného příjmu je negovaný zopakován. Z tohoto vychází možnost identifikovat 16 typů paketů. Datový paket nenese adresu zařízení ani koncového bodu, protože komunikace již byla ustanovena. Paket je ukončený 5 bitů dlouhým kontrolním součtem.

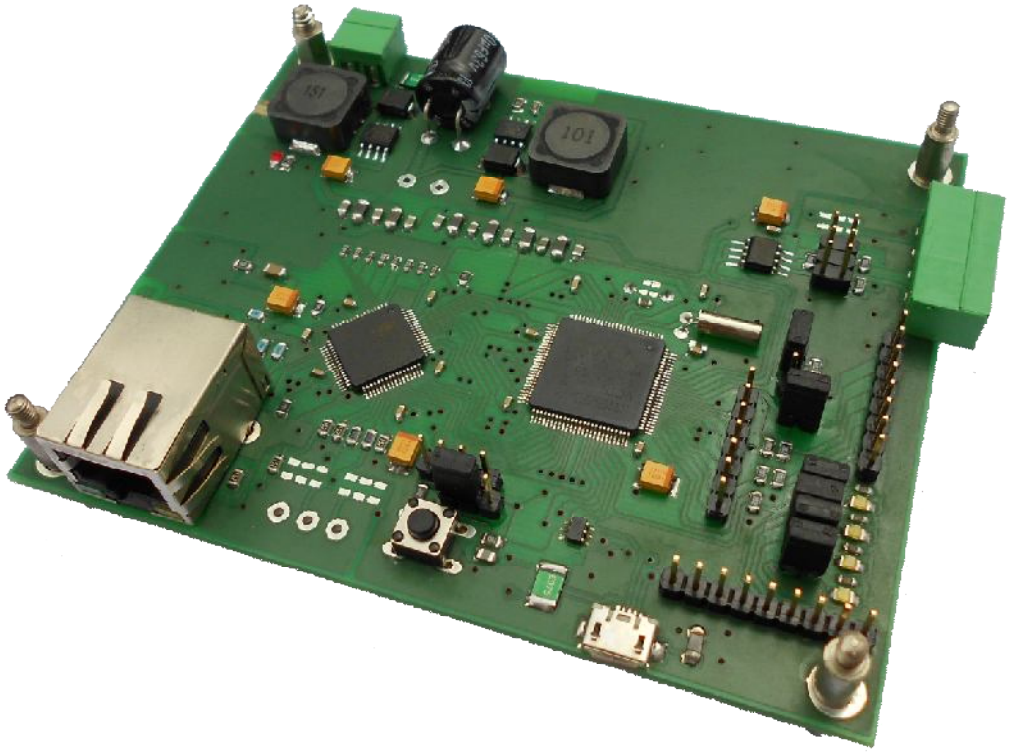
Joystick vysílá 20B dlouhý datový paket a v oblasti data je zakódovaný aktuální stav joysticku. Vytvořil jsem tabulku popisující možné stavy joysticku pro usnadnění dekódování. Hodnoty v tabulce jsou v hexadecimálním tvaru.

Tabulka 5: Tabulka dekódování datového paketu joysticku

Byte	Funkce	Hodnota					
1	PID	1B					
2	KŘÍŽ	UP 0x01	DOWN 0x02	LEFT 0x04	RIGHT 0x08	START 0x10	STOP 0x20
3	TLAČÍTKA	A 0x10	B 0x20	X 0x40	Y 0x 80	LEFT B 0x 01	RIGHT B 0x02
4	LEVÝ POTENC.	0x00 až 0xFF					
5	PRAVÝ POTENC.	0x00 až 0xFF					
6	LEVÝ JOY.	RIGHT	0x8000 – 0x0080				
7		LEFT	0x8000 – 0xFF7F				
8		NAHORU	0x8000 – 0x0080				
9		DOLU	0x8000 – 0xFF7F				
10	PRAVÝ JOY.	RIGHT	0x8000 – 0x0080				
11		LEFT	0x8000 – 0xFF7F				
12		NAHORU	0x8000 – 0x0080				
13		DOLU	0x8000 – 0xFF7F				
14	EMPTY						
15	EMPTY						
16	EMPTY						
17	EMPTY						
18	EMPTY						
19	EMPTY						
20	EMPTY						

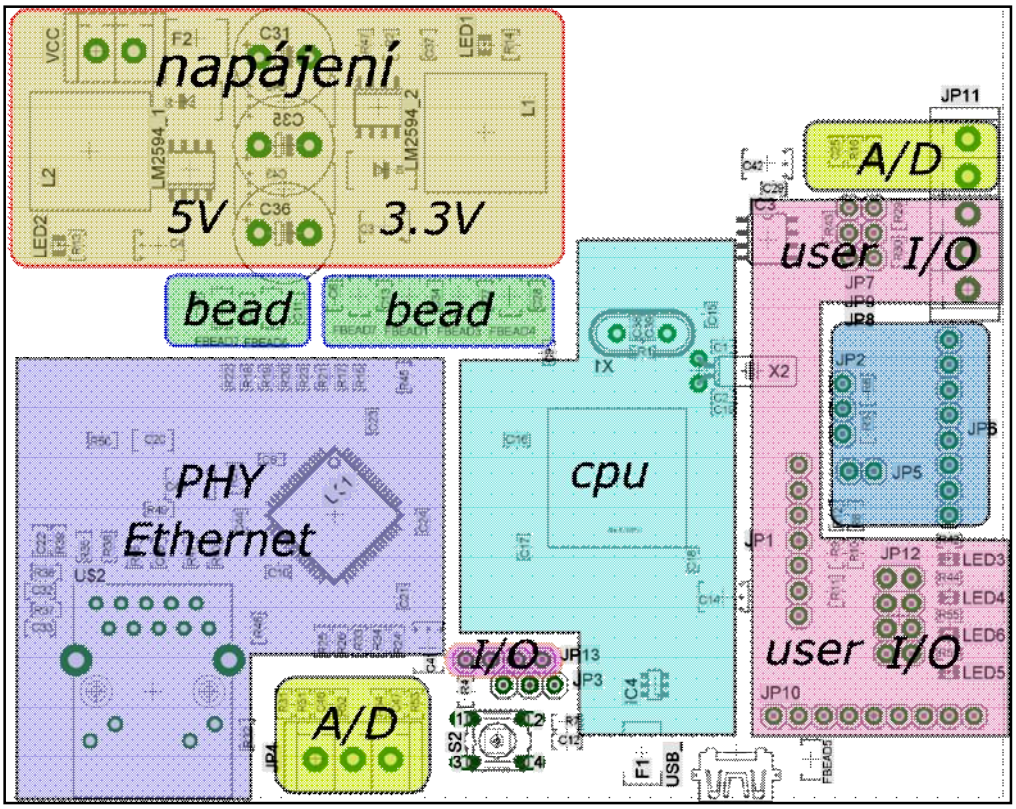
7 HARDWARE

V textu se budu odkazovat přímo na použité součástky pomocí jejich označení, které je kompatibilní s navrženým schématem zapojení umístěným v kapitole Přílohy. Dále jsou v následující části popsané části hardwaru a jejich konfigurace, která je důležitá pro další práci s převodníkem (vývojovou DPS).



Obrázek 27: Hotový výrobek převodník/vývojový kit

Původní zadání projektu převodník USB – Ethernet, jsem po dohodě s vedoucím práce rozšířil takovým způsobem, aby bylo možné na navržené DPS provádět vývoj a ladění aplikací co nejširšího spektra. Modifikace DPS spočívala převážně v umožnění použití dostupných interface, užitečných v oblasti robotiky. Celý návrh hardwarové části je optimalizován tak, aby byl z dostupných součástek a prostorově minimalizovaný. Při návrhu je použita většina součástek s SMD pouzdrem. Pasivní součástky jsou v pouzdrech SMD0805, speciálně pak blokovací kondenzátory SMD0603. Rozložení součástek na DPS je rozděleno logicky do funkčních bloků, aby bylo zapojení přehlednější při ladění HW.



Obrázek 28: Rozmístění součástek na DPS

7.1 Napájení a zemnění

Na vývojové desce jsou použité součástky s napájecím napětím 3,3V a 5V a proto jsem se rozhodl použít dva samostatné zdroje. Pro určení parametrů zdrojů, jsem vytvořil tabulku maximálního nezanedbatelného proudového odběru součástek.

Tabulka 6: Maximální proudový odběr součástek

Součástka	Napájecí napětí [V]	Max. odebíraný proud [mA]
MCU	3,3	150
PHY	3,3	90
USB_OTG	5,0	499
MAX481	5,0	1

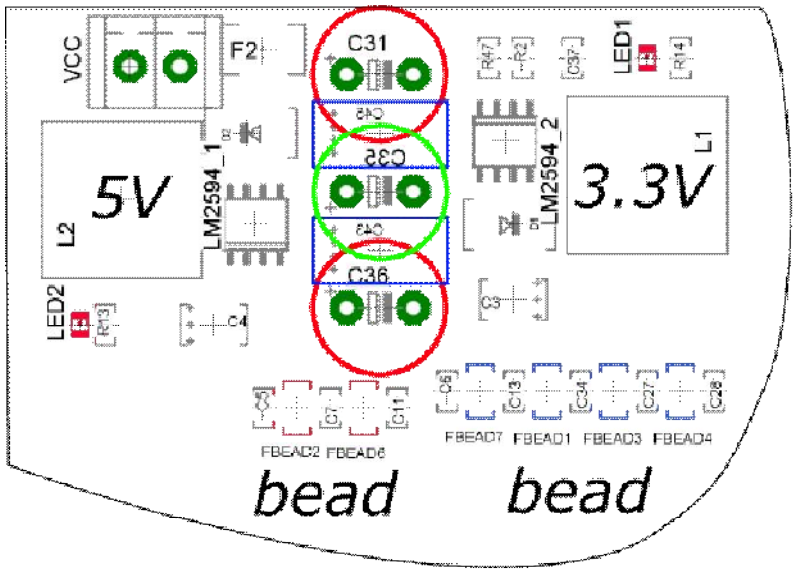
Pro napájení jsem jednoznačně zvolil integrované spínané zdroje, které mají menší rozměry a vyšší účinnost. Protože napájecí napětí převodníku bude maximálně 24V nebo 48V, podle toho v jakém robotickém systému bude použitý, bylo třeba přizpůsobit součástky napájecího zdroje tomuto napětí. Součástky, kterých se to týká, jsou filtrační kondenzátory vstupu a samotné integrované spínané zdroje. U spínaných zdrojů jsem

zvolil takový obvod, který je dostupný ve verzi 3,3V, 5V a pro obě napětí i v HV verzi (vstupní napětí do 60V) a jeho maximální zatěžovací proud byl 500mA. Při maximálním rozdílu napětí vstup/výstup jsem odečetl hodnotu teploty pouzdra 45°C, proto není třeba řešit chlazení obvodu. Tyto podmínky splňuje spínaný zdroj firmy National Semiconductor LM2594, včetně shodného pouzdra a rozložení pinů. Protože zdroj 3,3V nebyl dostupný, použil jsem stejný typ ve verzi ADJ, kde se výstupní napětí nastaví pomocí odporového děliče na zpětné vazbě obvodu. Velikosti odporů jsem stanovil dle dokumentace výrobce, podle vzorce (1.1). Kde: Vref je vnitřní referenční napětí 1,23V a R1 se vhodně volí tak, aby vypočtená hodnota R2 byla dostupná s maximální odchylkou 1%.

$$\frac{\quad}{\quad} \tag{1.1}$$

Parametry dalších součástek jsem zvolil dle dokumentace výrobce.

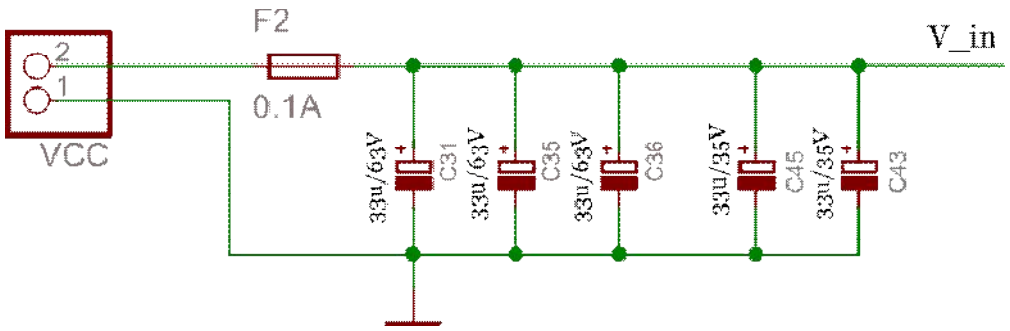
Převodník je možné použít i na napětí 45 – 60V. Na DPS je třeba vyměnit IO za jejich zmíněné HV verze a vstupní filtrační kondenzátory. Vzhledem k lepším vlastnostem – ESR, jsou použity v paralelní kombinaci C45,C43 tantalové a C35 – elektrolytický. Protože tantalové kondenzátory jsou na napětí 35V jsou v HV verzi zaměněny elektrolytickými C31,C36.



Obrázek 29: Spínané zdroje-filtrace

Na obrázku jsou vyznačeny modře tantalové kondenzátory, které se v případě HV verze zamění za červeně označené elektrolytické. Při návrhu rozložení součástek zdroje je nutné dodržet maximální vzdálenosti spojů mezi součástkami z důvodu hrozby rozkmitání zdroje. U každého zdroje je indikační LED, která současně plní funkci

minimální odběru zdroje. Z důvodu filtrace možného Vf rušení zdroje jsem použil feritové korálky (FBEAD), které lze s výhodou využít při oživování, pro postupné připojování periferii. Každá periferie má tedy vlastní přívod napájení. DPS je navržena tak, aby na straně součástek byla pouze celistvá zemnící plocha, která je logicky rozčleněna na analogovou a digitální část viz Obrázek 37. Pod cívkami L1, L2 jsem vzhledem k vertikálnímu provedení cívek provedl nařízntutí (restrict) těchto zemnících ploch, abych zabránil vzniku proudové smyčky.



Obrázek 30: Vstupní filtr zdrojů

Souvislé zemnící plochy (rozlévaná měď) minimalizují vznik proudových smyček a snižují impedanci přívodu GND (AGND) k součástce. Z toho důvodu jsem použil vícebodové zemnění, kdy je GND (AGND) pin součástky připojen nejkratší cestou k zemnící ploše. Digitální a analogová země jsou vzájemně propojené rezistorem R50 o velikosti 0R0.

Ke standardnímu zapojení součástek patří blokovací kondenzátory napájení, které nejen zvyšují spolehlivost součástek.

Filtrační blokovací kondenzátory jsou použité na vstupním napájení převodníku a eliminují vliv indukčností přívodů a kontaktních přechodových odporů napájecích konektorů.

Lokální blokovací kondenzátory slouží jako zdroje energie pro součástky a redukuje impulsní proudy. Jeho hodnota se pohybuje v rozmezí 100pF-100nF. Hodnota se určuje z hodnoty proudového impulsu, maximální změny napětí během pulsu a dobu trvání pulsu. V převodníku jsem použil lokální kondenzátory, které jsou umístěné co nejblíže napájecím pinům součástky, 100nF a filtrační kondenzátory pro každý obvod 100µF. [1]

7.2 Mikrokontrolér a PHY

Mikrokontrolér v pouzdře LQFP100 a fyzickou vrstvu Ethernetu v pouzdře TQFP64, bylo třeba usadit výhodně tak, aby vzájemné propojující, vodivé cesty byly minimální délky. Mikrokontrolér má možnost přemapování některých pinů rozhraní MII, ale i

přesto nebylo možné navržení spojů tak, aby nebyla použita průchodka. Vzhledem k tomu, že je nutné dodržet přibližné délky spojů, aby nebylo porušeno časování signálů. Pro diagnostiku zpoždění signálů na DPS jsem využil program firmy Saturn PCB Design – PCB Toolkit dostupný ve freeware verzi na [5]. Protože použitý rozměr průchodky zavádí zpoždění 100ps a bylo třeba použít 4 průchodky na jeden signál *Transfer* strany, tak jsem jako kompenzaci umístil průchodky na všechny signály strany *Transfer*. Strana *Recieve* je propojena bez průchodek. Pomocí skriptu programu Eagle *lenght.ulp* jsem vytvořil tabulku délek jednotlivých signálů a potřebné signály jsem uměle prodloužil pomocí meandrů. Jedná se hlavně o signály RxD[3:0] (TxD[3:0]) vzhledem k synchronizačnímu signálu RXCLK (TXCLK), kde je dle dokumentace fyzické vrstvy maximální dovolené zpoždění 10ns.

Resetování obvodu mikrokontroléru a fyzické vrstvy se provede přivedením logické 0 na piny RST pomocí tlačítka S2 nebo při propojeném konektoru JP5 pomocí rozhraní JTAG.

Fyzická vrstva STE100p má dvě možnosti konfigurace. Jedna možnost – pomocí systémové sběrnice popsané v kapitole 3.4.2 a druhá možnost je používaná především pro prvotní konfiguraci příslušných registrů po resetu zařízení. Jedná se o pevně nastavené vstupy PHY – mf0:mf4 pomocí pull-up (down) rezistorů R18 – R23.

Jedná se o konfiguraci sestavování rámce:

- mf0 - Auto-negace
- mf1 – Povolení konverze NRZ-NRZI
- mf2 – Povolení 4B/5B kódování
- mf3 – Scrambler
- mf4 – Volba rychlosti přenosu 10/100Mbps
- fde – Povolení plně duplexního režimu

Fyzická vrstva umožňuje signalizace stavu komunikace pomocí LED, protože jsem použil standardní konektor RJ45, takže jsem mohl využít pouze indikaci aktivity linky – zelená LED a přenosu dat na Rx,Tx – žlutá LED na konektoru RJ45. Konektor RJ45 má integrované magnetické obvody, které slouží ke galvanickému oddělení.

Konfigurační rezistory konektoru RJ45 R41, R40, R48, R35 jsou připojeny na digitální zem přes kondenzátor C20, který je dimenzovaný na 2kV a který filtruje indukované napětí na kabelu UTP (STP). Maximální délka kabelu je daná použitím fyzické vrstvy 100BASE-TX, kde je limitujícím faktorem použitá metoda CSMA/CD, kde u delšího kabelu než 300m není zaručena spolehlivá detekce kolize.

7.2.1 Bootloader

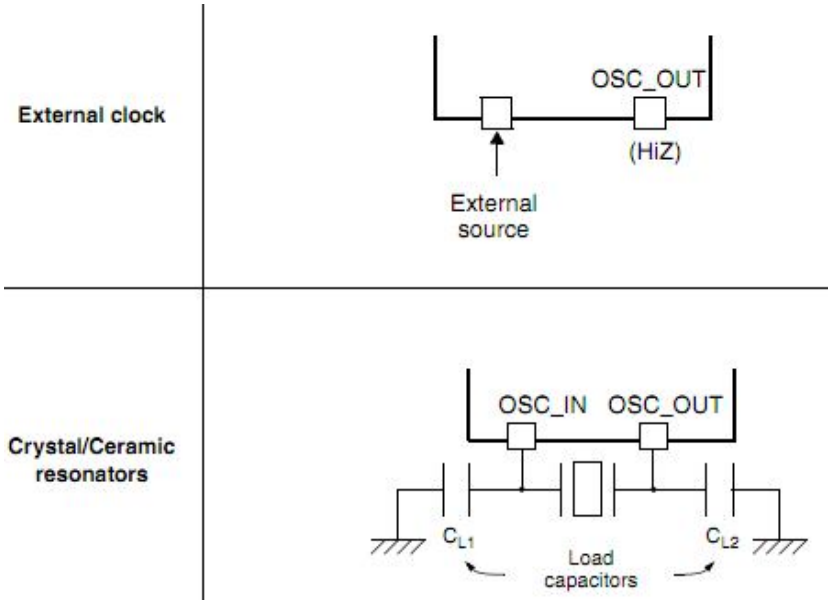
Obvod také obsahuje důležité programovací a debuggovací rozhraní **JTAG** a **SWG**, které lze použít v kombinaci s popisovanými IDE a programátorem použít jako výhodný ladicí a programovací nástroj – popisované v kapitole 4. Samotné programování je možné pomocí integrovaného bootloaderu. Bootloader podporuje programování pomocí UART, USB, ETH, CAN, ale standardně se používá UART1. Bootloader se aktivuje nastavením pinu BOOT0 na log. 1, BOOT1 na log. 0 a následného resetu obvodu. Kombinací pinů BOOT0 a BOOT1 se vybírá aktivní paměť procesoru – paměť, která se začne bootovat po resetu čipu. Vlastní program se ukládá do *Main Flash memory* a bootloader je uložený v *Systém memory*. [9]

HW konfigurace přepínačů JP3 – BOOT_1 a JP2 – BOOT_2 je znázorněna na Obrázek 17.

Tabulka 7: HW konfigurace výběru bootovací paměti

Konfigurační piny bootloaderu		Boot mode
BOOT1	**BOOT0**	
X	**0**	Main Flash memory
0	**1**	System memory
1	**1**	Embedded SRAM

7.2.2 Zdroj hodinového signálu



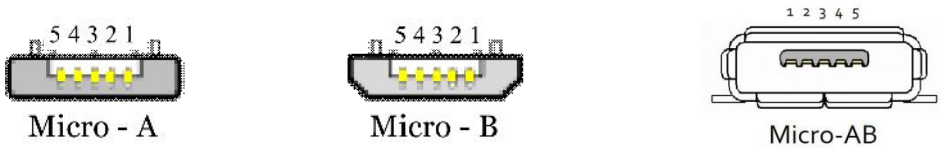
Obrázek 31: Možnosti přivedení hodinového signálu pro HSE[6]

HSE má dvě možnosti vstupního signálu. Bypass, což je vstup z externího zdroje o frekvenci 25MHz připojený na svorky pro krystal, kde výstupní pin je připojený na vysokou impedanci. Vstupní signál může být čtverec, sinus, trojúhelník se střídou 50%. Druhá možnost je krystal připojený standardním způsobem podle obrázku.

Časování mikrokontroléru zajišťuje oscilátor o frekvenci 25MHz se sériovým rezistorem 200 a časování fyzické vrstvy pomocí externího signálu X1 generovaným procesorem. Mikrokontrolér má připojený i oscilátor 32768Hz, pro případný vývoj aplikace s hodinami reálného času. Kondenzátory Cl1 a Cl2 nejsou osazeny z důvodu jejich velice nízké kapacity, kterou nahradí kapacita samotné DPS.

7.3 USB_OTG

Ze specifikace USB_OTG vyplívá, že správné připojení dvou zařízení v tomto režimu se provádí kabelem, který má na straně Device zařízení konektor typu USBMicro-A, kde je signál ID uzemněný a na straně Host konektor USBMicro-B. Konektory jsou vzájemně nezaměnitelné a jejich použití je dané specifikací USB. Jak jsem již popisoval, mikrokontrolér se může chovat jako Host i jako Device zařízení, proto jsem na DPS osadil SMD konektor USBMicro-AB, ke kterému lze připojit oba zmiňované konektory. Osazený konektor má tedy 5 vodičů včetně signálu ID, který je připojený na pin procesoru. Kostra konektoru je připojena přes feritový korálek na digitální zem.



Obrázek 32: Konektory USB_OTG [14]

Tabulka 8: Propojení USBMicro-AB s mikrokontrolérem

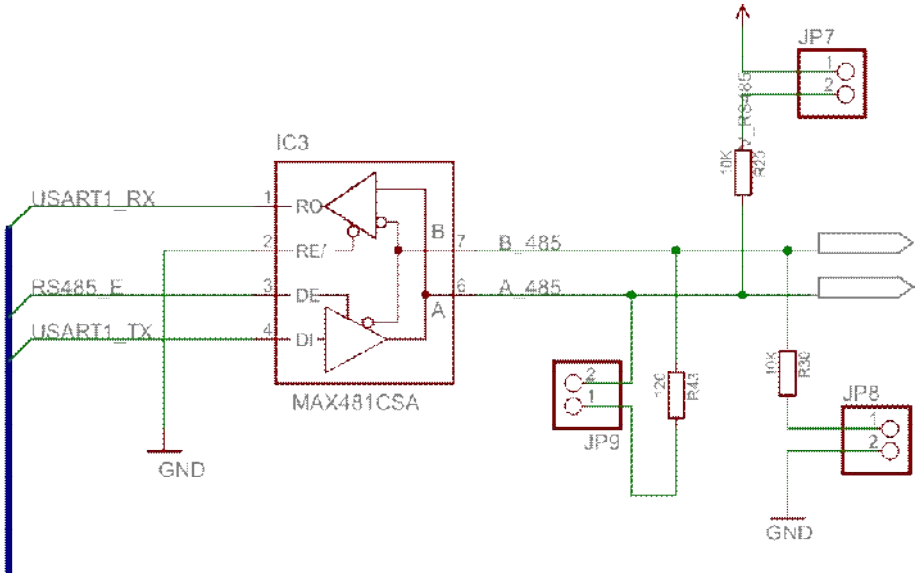
signál	pin cpu	pin USB
OTG_VBUS	+5V	1
OTG_ID	69 / PA10	4
OTG_DP	71 / PA12	3
OTG_DM	70 / PA11	2
OTG_GND	GND	5

Zapojení jsem doplnil o EDS ochranu elektrostatických výbojů USBLC6-2P6 pro vysokorychlostní komunikace a dále by bylo vhodné zařadit proudovou ochranu USB rozhraní. Vzhledem k nedostupnosti kompatibilního obvodu firmy ST jsem se rozhodl připojit signál VBUS přes vratnou pojistku F1 700mA přímo na stabilizované napětí 5V dle reference manuálu [5].

7.4 Uživatelské I/O

Jak jsem zmínil, rozhodl jsem se navrženou DPS rozšířit o další rozhraní a částečně z DPS udělat vývojový kit, aby bylo možné například aplikaci převodníku rozšířit případně vytvořit aplikaci novou.

Jedním z praktických rozhraní jsou sériová synchronní/asynchronní. Na pinheader JP1 je vyvedený USART2 a jeho datové signály USART2_RX, USART2_TX a signály hardwarového požadavku na vysílání USART2_RTS, povolení vysílání USART2_CTS. Další rozhraní USART jsem použil, pomocí převodníku firmy Maxim Max481CSA, pro sběrnici RS485. Převodník má vlastní přívod 5V napájení a jeho piny jsou standardně blokované.



Obrázek 33: Schéma zapojení převodníku UART/RS485

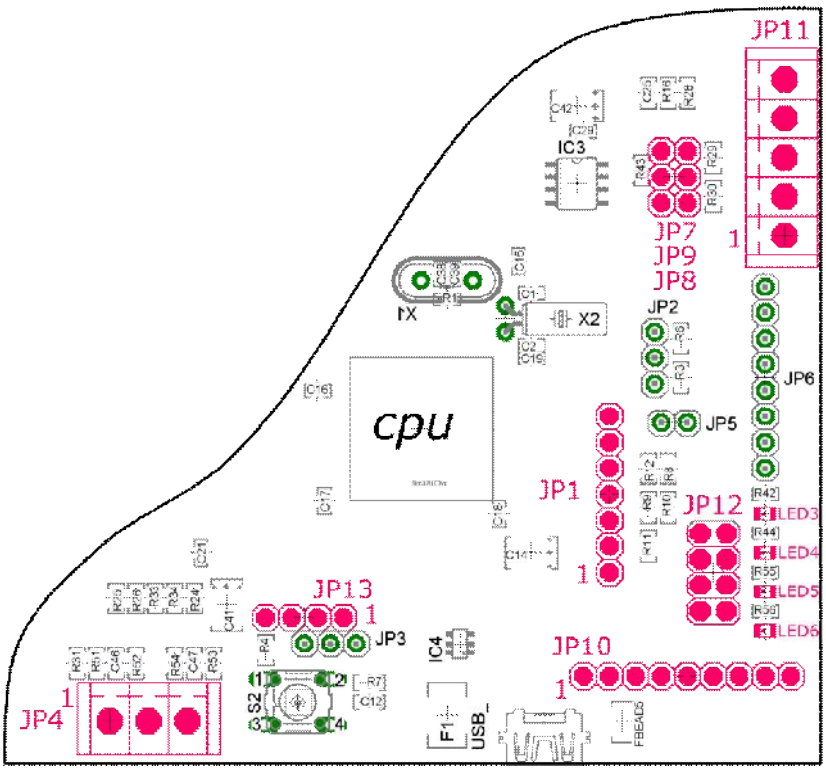
Převodník vyžaduje jeden obslužný signál RS485_Enable, který je aktivní v případě vysílání a neaktivní pro příjem. Signál je připojený na portu B a pinu 5. Sběrnice RS485 vyžaduje mezi diferenciálními signály použití terminátoru, což je rezistor, který zabraňuje odrazům na sběrnici. Protože zařízení nemusí být jako koncové, je terminátor R43 120Ω připojený přes switch JP9 (pokud se jedná o zařízení NE-koncové, na odbočce 100x kratší, než je sama sběrnice, terminátor se nepoužívá) stejně jako pull-up

a pull-down rezistory R29, R30, které v případě potřeby drží sběrnici ve stavu vysoké impedance. Rozhraní RS485 je vyvedeno na konektoru JP11 a pinech 2,3.

Další sériové rozhraní SPI3 je vyvedeno na zbylé piny pinheaderu JP1, které se používá v meziprocesorové komunikaci. Pro případ komunikace mezi více zařízeními je třeba dalších řídicích signálů *chip_select*, který udává vybrané (např. demultiplexem) stanici oprávnění vysílat. Pro tyto signály je možné použít jakýkoliv vstupně/výstupní nepoužitý pin procesoru.

Velice důležité pro robotické aplikace, jsou převodníky analogového signálu na digitální. Například pro potřebu měření stavu baterie, teploty jsem na konektory JP11 a JP4 vyvedl signál A/D převodníku včetně analogové GND. Na vstupu převodníku je připravený odporový dělič, který není osazený. Skládá se ze sériového rezistoru R28, R52, R54 a paralelního R16, R51, R53 a jejich osazení se předpokládá až dle aktuální aplikace.

Jako výhodný indikátor při ladění programů fungují LED diody, které lze připojit na vstupně/výstupní piny procesoru pomocí switchů na JP12 v opačném případě jsou piny připojené na pinheader JP10.



Obrázek 34: Umístění a uživatelských pinů

Tabulka 9: Mapování uživatelských pinů

pinheader / konektor	pozice	pin / mapování cpu	funkce	alt. Funkce
JP1	1	34 / PC5	SPI_SCK	
	2		SPI_MISO	
	3		SPI_MOSI	
	4		USART2_CTS	
	5		USART2_RTS	
	6		USART2_TX	
	7		USART2_RX	
JP4	1		AGND	
	2	34 / PC5	ADC12_IN15	
	3	36 / PB1	ADC12_IN9	TIM1_CH3N
JP7			PULL-UP	
JP8			PULL-DOWN	
JP9			TERMINÁTOR	
JP10	1	61 / PD14		TIM4_CH3
	2	62 / PD15		TIM4_CH4
	3	63 / PC6		
	4	65 / PC8		
	5	66 / PC9	LED3	
	6	81 / PD0	LED4	CAN1_RX
	7	82 / PD1	LED6	CAN2_TX
	8	83 / PD2	LED5	TIM3_ETR
	9		GND	
JP11	1		GND	
	2		RS485_A	
	3		RS485_B	
	4		AGND	
	5	15 / PC0	ADC12_IN10	
JP13	1	43 / PE12		TIM1_CH3N
	2	42 / PE11		TIM1_CH2
	3	41 / PE10		TIM1_CH2N
	4	40 / PE9		TIM1_CH1

8 ZÁVĚR

Cílem diplomové práce bylo navrhnout hardware a software převodníku USB – Ethernet. Firmware jsem na začátku projektu vyvíjel na popisovaném vývojovém kitu firmy Hitex – STM32ComStick. Tento kit není pro vývoj složitějších aplikací vhodný a to z důvodu omezení velikosti kódu programovacího prostředí, které zároveň představuje jedinou možnost debuggování mikrokontroléru. Z tohoto důvodu jsem používal pro vývoj aplikace prostředí Ride7 firmy Raisonance a později prostředí µVision firmy Keil, která široce podporuje procesory s jádrem ARM.

Pro vývoj aplikace byly použity kostry ukázkových programů výrobce mikrokontroléru, které jsem přepracoval dle mého zadání. V projektu jsem používal standardní knihovny mikrokontroléru a vlastní knihovny, určené pro práci s digitálními vstupně/výstupními piny. Ethernetové rozhraní používá komunikační protokol UDP, nad kterým jsem vytvořil další komunikační vrstvu, dle specifikace zadané vedoucím práce. Specifikace USB sběrnice je velice rozsáhlá a má široké spektrum použití, proto se v práci zabývám konkrétní modifikací USB_OTG.

Pro převodník jsem použil stejný mikrokontrolér, který byl implementovaný ve vývojovém kitu STM32ComStick a aplikace tak byla kompatibilní. Konkrétně jsem použil mikrokontrolér firmy STMicroelectronics STM32F107VCT6 založený na jádře ARM Cortex – M3

Při návrhu hardwaru jsem využíval konzultací s vedoucím práce a jeho velice rozsáhlých zkušeností s návrhem hardwaru na vysoké úrovni. Také proto jsem po otestování všech dostupných periférií převodníku/vývojového kitu nenašel žádné konstrukční a návrhové vady a základní funkce periferií se podařilo oživit. Navržená deska má implementované důležité rozhraní JTAG/SWD, které umožní vývoj dalších aplikací. Pokusil jsem se vytvořit srozumitelný návod pro použití dostupných periferií, včetně jejich softwarové obsluhy a podrobný popis navržené DPS pro usnadnění v případě, že DPS by byla potřeba modifikovat, rozšiřovat pro další aplikace.

Návrh a konstrukci jsem logicky dělal na konci práce, kdy jsem ověřil realizovatelnost převodníku. Převážnou většinu firmwaru převodníku jsem tedy vyvíjel bez možnosti debuggování a k dispozici jsem měl pouze indikační LED vývojového kitu. Přesto jsem naprogramoval funkční Ethernet Stack, který funguje i při vysokých rychlostech žádostí. Dále jsem implementoval kompletní USB Stack, který jsem z důvodu složitosti ladil až na navržené DPS.

Z časových důvodů se v době sepisování této práce nepodařilo spojit firmwarové projekty a převodník nemůže být nasazený v reálné aplikaci. Nicméně po dohodě s vedoucím práce bude převodník doprogramován a kompletně odladěný.

Literatura

[1] ZÁHLAVA, Vít. *Návrh a konstrukce desek plošných spojů: principy a pravidla praktického návrhu*. 1. vyd. Praha: BEN - technická literatura, 2010, 123 s. ISBN 978-80-7300-266-4. [cit. 2012-05-10]

[2] AXELSON, Jan. *USB complete everything you need to develop custom USB peripherals*. 3rd ed. Madison, WI: Lakeview Research, 2005. ISBN 978-193-1448-031. [cit. 2012-05-10]

[3] Compaq, Intel, Microsoft, NEC. *Universal Serial Bus Specification - revision 1.1*. USB Implementers Forum, September 23, 1998. [cit. 2012-05-10] Dostupné na URL: <http://esd.cs.ucr.edu/webres/usb11.pdf>

[4] PUŽMANOVÁ, Rita. *Moderní komunikační sítě od A do Z*. 2. aktualiz. vyd. Brno: Computer Press, 2006, 430 s. ISBN 80-251-1278-0. [cit. 2012-05-10]

[5] STMicroelectronics: *Reference manual* [online]. Verze 12, aktualizováno 2011-1-1, str. 1096. [cit. 2012-05-10] Dostupné na URL: <http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/REFERENCE_MANUAL/CD00171190.pdf>

[6] STMicroelectronics: *Datasheet* [online]. Verze 6, aktualizováno 2011-Srpen, str. 103. [cit. 2012-05-10] Dostupné na URL: < http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00220364.pdf>

[7] STMicroelectronics: *ARM-based 32bit MCU STM32F10xxx standard peripheral library* [online], aktualizováno 2011. [cit. 2012-05-10] Dostupné na URL: <http://www.st.com/internet/com/SOFTWARE_RESOURCES/SW_COMPONENT/FIRMWARE/stm32f10x_stdperiph_lib.zip>

[8] STMicroelectronics: *ARM-based 32bit MCU STM32F10xxx USB Device Full Speed Library*[online], aktualizováno 2011. [cit. 2012-05-10] Dostupné na URL: <http://www.st.com/internet/com/SOFTWARE_RESOURCES/SW_COMPONENT/FIRMWARE/um0424.zip>

[9] STMicroelectronics: *STM32™ microcontroller system memory boot mode* [online], aktualizováno Nov.2011. [cit. 2012-05-10] Dostupné na URL: <http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/APPLICATION_NOTE/CD00167594.pdf>

[10] STMicroelectronics: *ST-LINK in-circuit debugger/programmer for STM8 and STM32 microcontrollers* [online], aktualizováno Oct.2011. [cit. 2012-05-10] Dostupné na URL:

<http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/USER_MANUAL/CD00221563.pdf>

[11] STMicroelectronics: *10/100 FAST ETHERNET 3.3V TRANSCEIVER* [online], aktualizováno Oct.2011. [cit. 2012-05-10] Dostupné na URL: <http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00001918.pdf>

[12] Oficiální web výrobce: *ARM The Architecture for the Digital World* [Online], aktualizováno 2012. [cit. 2012-05-10].
Dostupné na URL: <http://www.arm.com/>

[13] Internetový server o sítích: *Portál o počítačových sítích a hardwaru* [Online], aktualizováno 2012. [cit. 2012-05-10]
Dostupné na URL: <http://site.the.cz/index.php>

[14] ŘEHÁK, J. *USB - Universal Serial Bus - Popis rozhraní* [online]. [cit. 2012-05-10]
Dostupné na URL: <http://www.hw.cz/navrh-obvodu/rozhrani/usb/usb-universal-serial-bus-popis-rozhrani.html>

[15] Internetový server o mikročipech: *Portál o počítačových sítích a hardwaru* [Online], aktualizováno 2012. [cit. 2012-05-10]
Dostupné na URL: <http://www.mcu.cz>

[16] Internetový server: *Vše o elektronice a programování* [Online], aktualizováno 2012. [cit. 2012-05-10]
Dostupné na URL: <http://www.hw.cz>

[17] Internetový server: *Embedded Development Tools* [Online], aktualizováno 2012. [cit. 2012-05-10]
Dostupné na URL: <http://www.keil.com>

[20] DVOŘÁK, Vítězslav. *Ethernet v průmyslových aplikacích*. Plzeň, 2007. 52 s. Diplomová práce. Západočeská univerzita v Plzni. [cit. 2012-05-10]

[21] KRIST, Petr. *Moderní principy průmyslových komunikací*. Plzeň, 2008. 413 s. Disertační práce. Západočeská univerzita v Plzni. [cit. 2012-05-10]

[22] USB IMPLEMENTERS FORUM, Inc. *Universal Serial Bus* [online]. 2012 [cit. 2012-05-10]
Dostupné na URL: <http://www.usb.org>

[23] USB ORG. Universal Serial Bus: HID Specifications [online]. 2012 [cit. 2012-05-10]
Dostupné na URL: <http://www.usb.org/developers/hidpage>

[24] Seznam podporovaných zařízení firmou Keil.
Dostupný na URL: <http://www.keil.com/arm/chips.asp>
[25] Podpůrný program pro návrh a konstrukci plošných spojů. *Saturn PCB Toolkit*
Dostupný na URL: <http://saturnpcb.com>

Seznam zkratek a symbolů

Vysvětlení častěji používaných zkratek a symbolů. Zkratky, které jsou použity jednorázově, jsou vysvětleny přímo v textu.

ARM Advanced RISC Machine – 32bitová architektura RISC firmy ARM

CAN Controller Area Network – komunikační standard, definující fyzickou a linkovou vrstvu, primárně vyvinut pro automobilový průmysl

CPU Central Processing Unit – centrální procesorová jednotka

CRC Cyclic redundancy Check – cyklický zabezpečovací kód

DPS Deska Plošného Spoje – slouží pro elektrické a mechanické propojení elektrotechnických součástek

ESD ElectroStatic Discharge – elektrostatický výboj

ETH Ethernet – označení komunikačního standardu IEEE 802.3

HID Human Interface Device – elektronické zařízení, které může převádět interakce člověka do elektronické podoby

HW Hardware – fyzicky existující elektronické zařízení

IDE Integrated Development Enviroment – programová aplikace efektivně spojující editor kódu, kompilátor a ladící prostředek

ID Identification – identifikace ve výpočetní technice

IEEE Institute of Electrical and Electronics Engineers – Institut pro elektrotechnické a elektronické inženýrství

IP Internet Protocol – komunikační protokol na úrovni síťové vrstvy

ISO/OSI International Standards Organization / Open System Interconnect – normalizační institut definující otevřený systém

JAM Kolizní sekvence

JTAG Joint Test Action Group – komunikační standard definovaný normou IEEE 1149.1

LED Light Emitting Diode – elektronická polovodičová součástka

LLC Logical Link Control – část linkové vrstvy specifikace IEEE 802.3

MAC Media Access Control – část linkové vrstvy specifikace IEEE 802.3, definuje řízení přístupu ke sdílenému médiu

MII Media Independent Interface – komunikační rozhraní mezi MAC a transceiverem Ethernet

NRZI Non-Return to Zero Inverted – kódování dat na úrovni linkové vrstvy

PHY Fyzická vrstva standardu ISO/OSI

RMII Reduced Independent Interface – komunikační rozhraní mezi MAC a transceiverem

SMD Surface Mount Device – součástka pro povrchovou montáž plošných spojů

SW Software – programové vybavení zařízení

SWD Serial Wire Debug – dvouvodičová alternativa standardu JTAG

TCP Transmission Control Protocol – protokol datových sítí na úrovni transportní vrstvy, poskytuje spojovanou službu

THT Through Hole Technology – způsob montáže elektronických součástek

UDP User Datagram Protocol - protokol datových sítí na úrovni transportní vrstvy, poskytuje nespojovanou službu

USB Universal Serial Bus – univerzální sériová sběrnice

USB_OTG Universal Serial Bus On The GO – modifikace sběrnice USB

Seznam použitých součástek

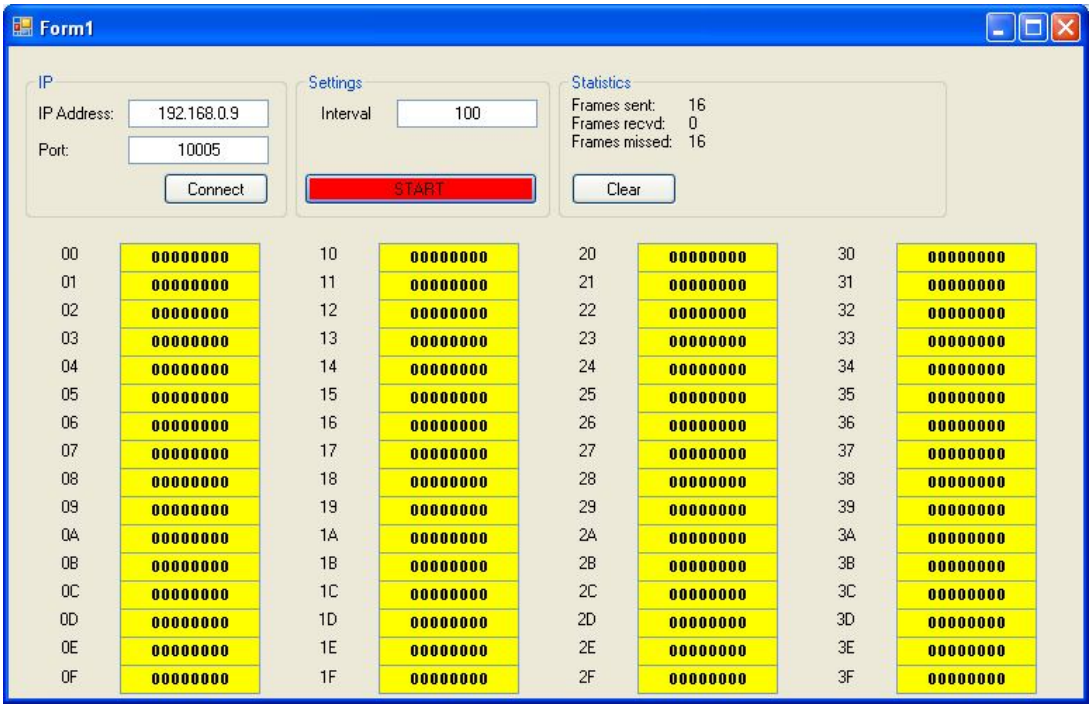
Počet	Popis	Hodnota	Pouzdro	Pozice
1 Ks		100uH/1.3A		L1
1 Ks		150uH/1A		L2
1 Ks		USB_CON_OTG	MINI USB-B R/A SMT W/ REAR	USB_con
1 Ks		STE100P	TQFP64	U$1
2 Ks	Dioda Schottkyho	1A/100V	DO214AA	D1, D2
2 Ks	Kondenzátor	12p	0603 [smd]	C1, C2
2 Ks	Kondenzátor	22p	0603 [smd]	C38, C39
13 Ks	Kondenzátor	100n	0603 [smd]	C8, C9, C10, C15, C16, C17, C18, C19, C21, C23, C24, C29, C44
1 Ks	Kondenzátor	5n6	0805 [smd]	C37
2 Ks	Kondenzátor	10p	0805 [smd]	C32, C33
14 Ks	Kondenzátor	100n	0805 [smd]	C5, C6, C7, C11, C12, C13, C22, C25, C26, C27, C28, C34, C46, C47
1 Ks	Kondenzátor	1n/2kV	1206 [smd]	C20
3 Ks	Kondenzátor Elektrolyt	33u/16V	RM = 5 mm, d = 10,5 mm radial	C31, C35, C36
4 Ks	Kondenzátor Elektrolyt	100u/6,3V	tantal B, 3528 [smd]	C14, C40, C41, C42
2 Ks	Kondenzátor Elektrolyt	100u/16V	tantal B, 3528 [smd]	C3, C4
2 Ks	Kondenzátor Elektrolyt	33u/16V	tantal D, 7343 [smd]	C43, C45
1 Ks	Konektor		PV05-3,81-H-P	JP11
4 Ks	Konektor 2 PIN		Lámací lišta	JP5, JP7, JP8, JP9
1 Ks	Konektor 2 PIN		PV02-3,81-H-P	VCC
2 Ks	Konektor 3 PIN		Lámací lišta	JP2, JP3
1 Ks	Konektor 3 PIN		PV03-3,81-H-P	JP4
1 Ks	Konektor 4 PIN		Lámací lišta	JP13
1 Ks	Konektor 4 PIN dvouřadý		Lámací lišta dvouřadá	JP12
1 Ks	Konektor 7 PIN		Lámací lišta	JP1
1 Ks	Konektor 8 PIN		Lámací lišta	JP6
1 Ks	Konektor 9 PIN		Lámací lišta	JP10
1 Ks	Krystal	32,768KHz	CRYSTAL	X2
1 Ks	Krystal	25MHz	HC49	X1
6 Ks	LED		Hyper CHIPLED Hyper-Bright LED	LED1, LED2, LED3, LED4, LED5, LED6
4 Ks	MOUNTING PAD, round	MOUNT-PAD-ROUND3.2	MOUNTING PAD 3.2 mm, round	H1, H2, H3, H4
1 Ks	OMRON SWITCH		OMRON SWITCH	S2
1 Ks	Pojistka	0.1A	1812 [smd]	F2
1 Ks	Pojistka	0.7A	1812 [smd]	F1

1 Ks	RESET CIRCUIT	USB LC6 - 2SC6	SOT-23 Package	IC4
1 Ks	RJ45 socket with LED.	RJ45+TRAFO+LED	RJ45 socket with LED ROFA	IC2
1 Ks	RS485 TRANSEIVER	MAX481CSA	Small Outline Package	IC3
8 Ks	Rezistor	10K	0603 [smd]	R15, R17, R18, R19, R20, R21, R22, R23
1 Ks	Rezistor	200R	0603 [smd]	R1
4 Ks	Rezistor	430	0603 [smd]	R42, R44, R55, R56
1 Ks	Rezistor	0R0	0805 [smd]	R39
1 Ks	Rezistor	1K2	0805 [smd]	R47
5 Ks	Rezistor	1K5	0805 [smd]	R24, R25, R26, R31, R32
1 Ks	Rezistor	2K	0805 [smd]	R2
1 Ks	Rezistor	4K99 1%	0805 [smd]	R27
12 Ks	Rezistor	10K	0805 [smd]	R3, R4, R6, R7, R8, R9, R10, R11, R12, R29, R30, R45
5 Ks	Rezistor	49R9	0805 [smd]	R35, R36, R37, R48, R49
2 Ks	Rezistor	75	0805 [smd]	R40, R41
1 Ks	Rezistor	100R	0805 [smd]	R46
1 Ks	Rezistor	120	0805 [smd]	R43
2 Ks	Rezistor	220R	0805 [smd]	R13, R14
2 Ks	Rezistor	300R	0805 [smd]	R33, R34
6 Ks	Rezistor	N/A	0805 [smd]	R16, R28, R51, R52, R53, R54
1 Ks	Rezistor	NEOSAZOVAT	0805 [smd]	R38
1 Ks	Rezistor	OR0	0805 [smd]	R50
1 Ks	Rezistor	CPU	1206 [smd]	FBEAD4
1 Ks	Rezistor	PHY	1206 [smd]	FBEAD1
1 Ks	Rezistor	RS485	1206 [smd]	FBEAD6
1 Ks	Rezistor	USB	1206 [smd]	FBEAD2
1 Ks	Rezistor	USB_CHASS	1206 [smd]	FBEAD5
1 Ks	Rezistor	VDDA_CPU	1206 [smd]	FBEAD3
1 Ks	Rezistor	VDDA_PHY	1206 [smd]	FBEAD7
1 Ks	Stabilizátor spínaný	Uout 3.3V 500mA	SO-08 [smd]	LM2594_2
1 Ks	Stabilizátor spínaný	Uout 5V 500mA	SO-08 [smd]	LM2594_1
1 Ks	Stm32f107vct6_v1.1_Šutera_Libor	STM32DEV_V2		IC1

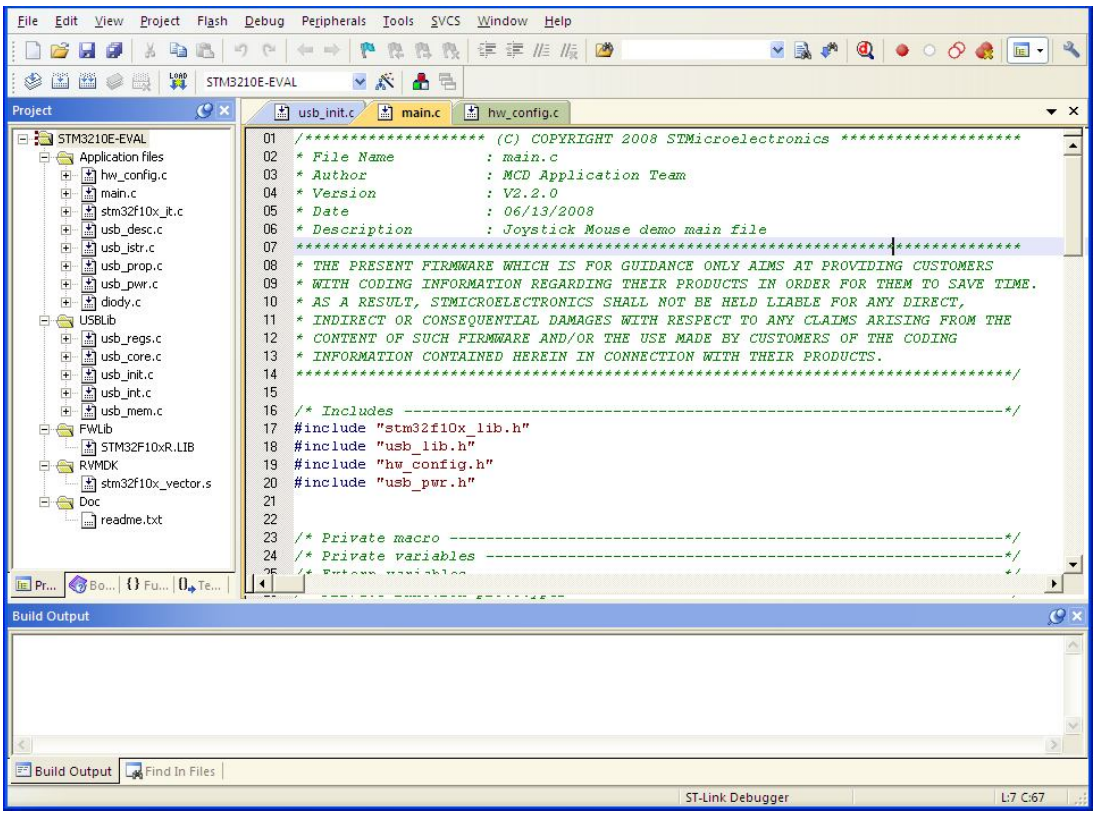
Seznam obrázků

Seznam příloh

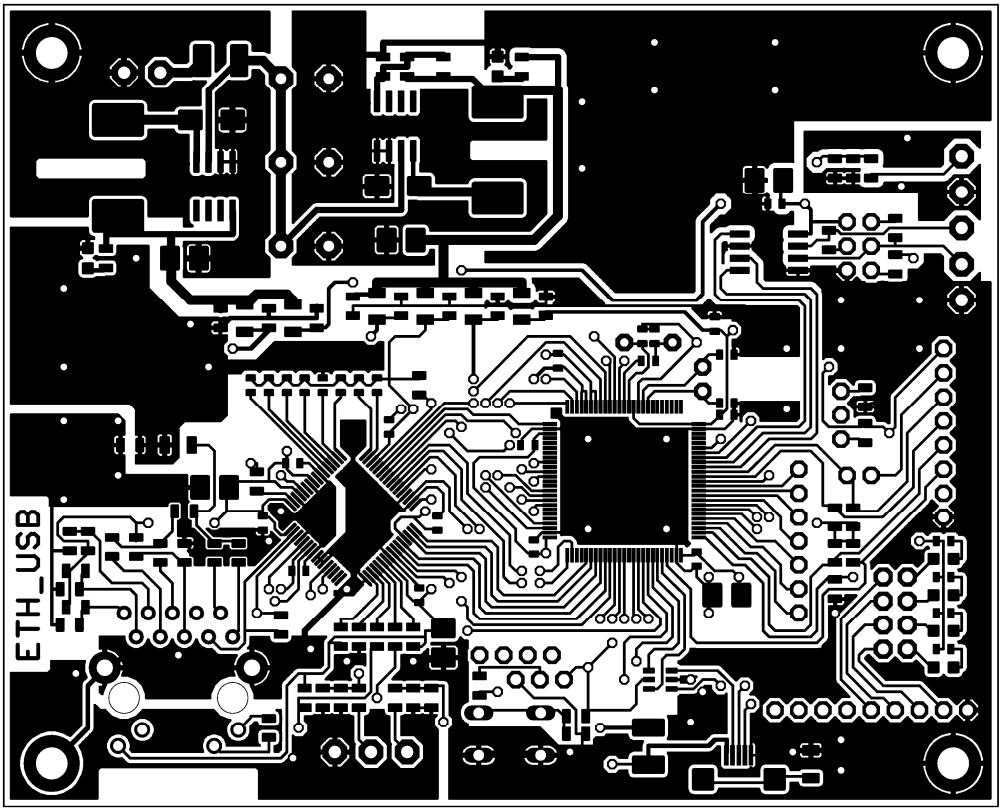
9 PŘÍLOHY



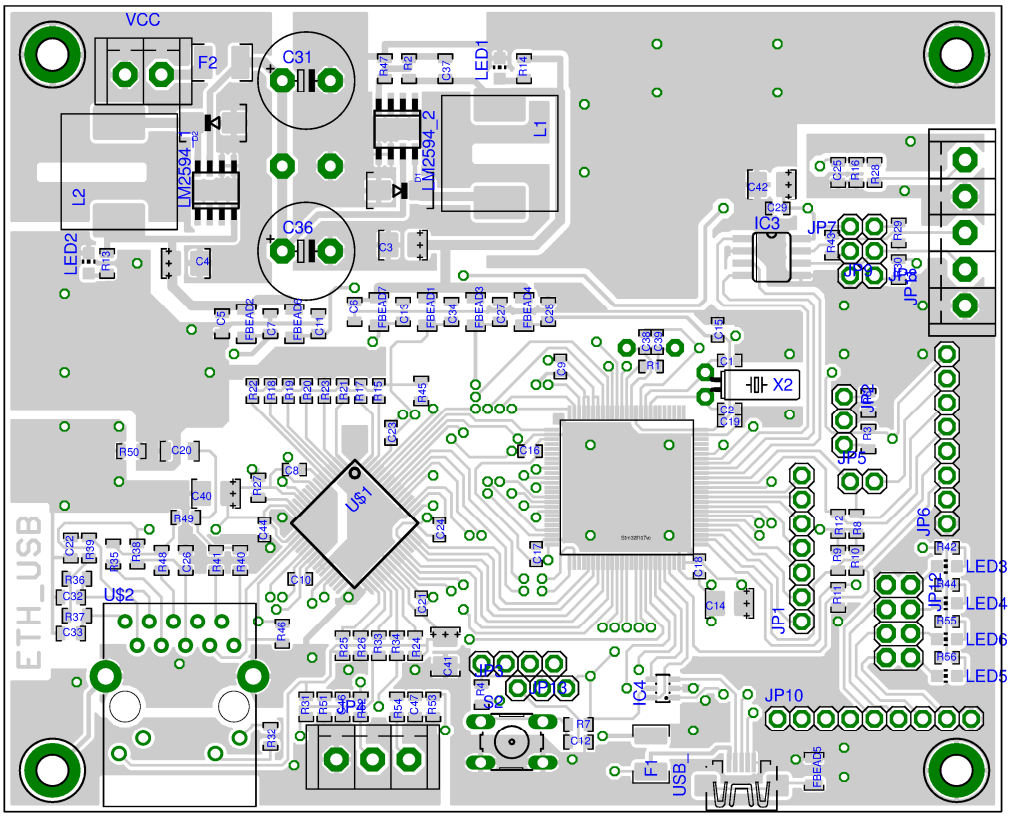
Obrázek 35: Program Monitor



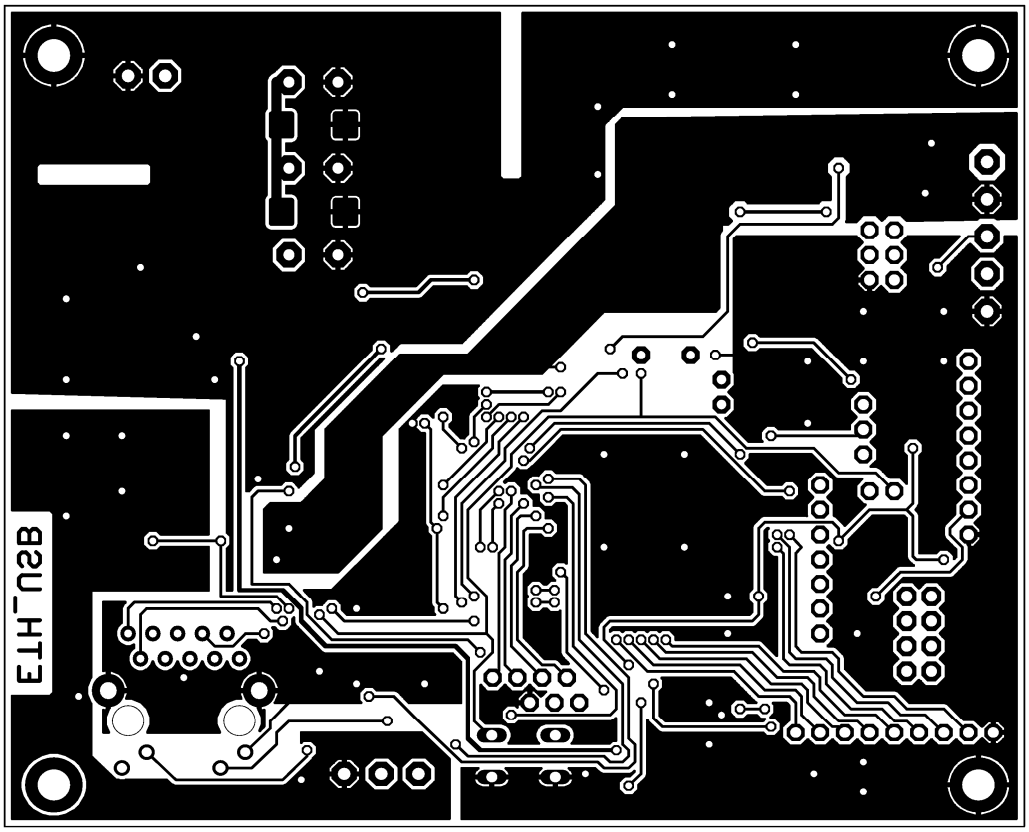
Obrázek 36: IDE Keil µVision



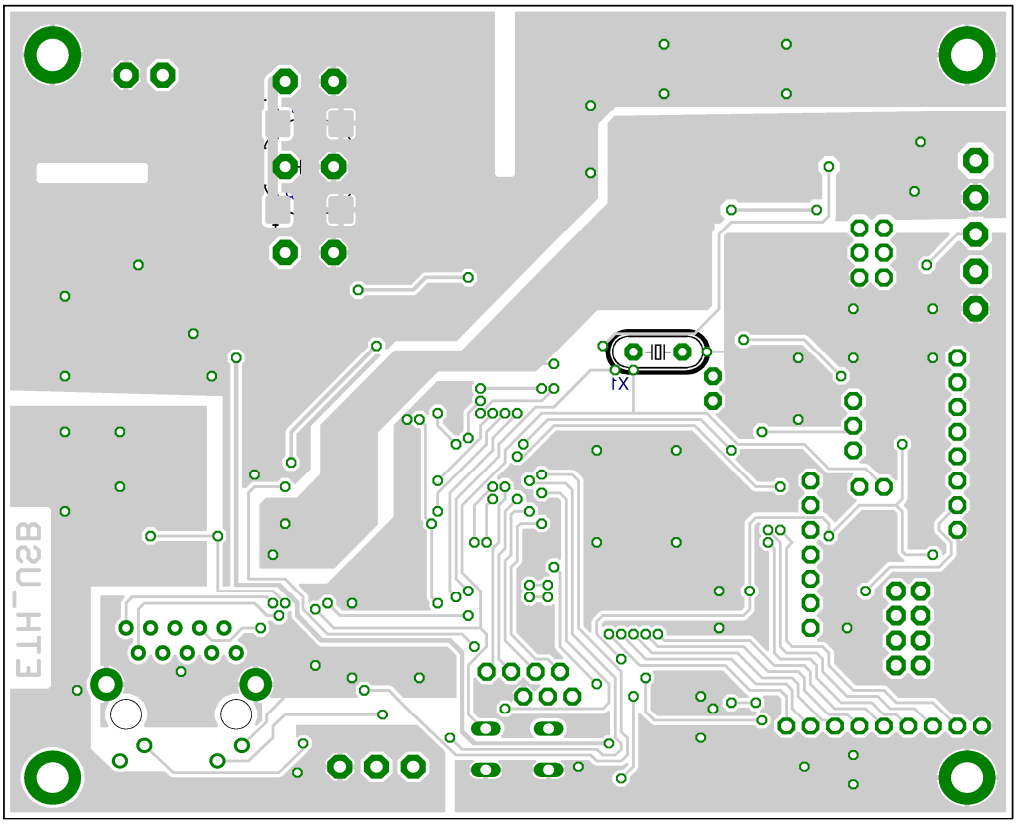
Obrázek 37: Deska plošného spoje - TOP



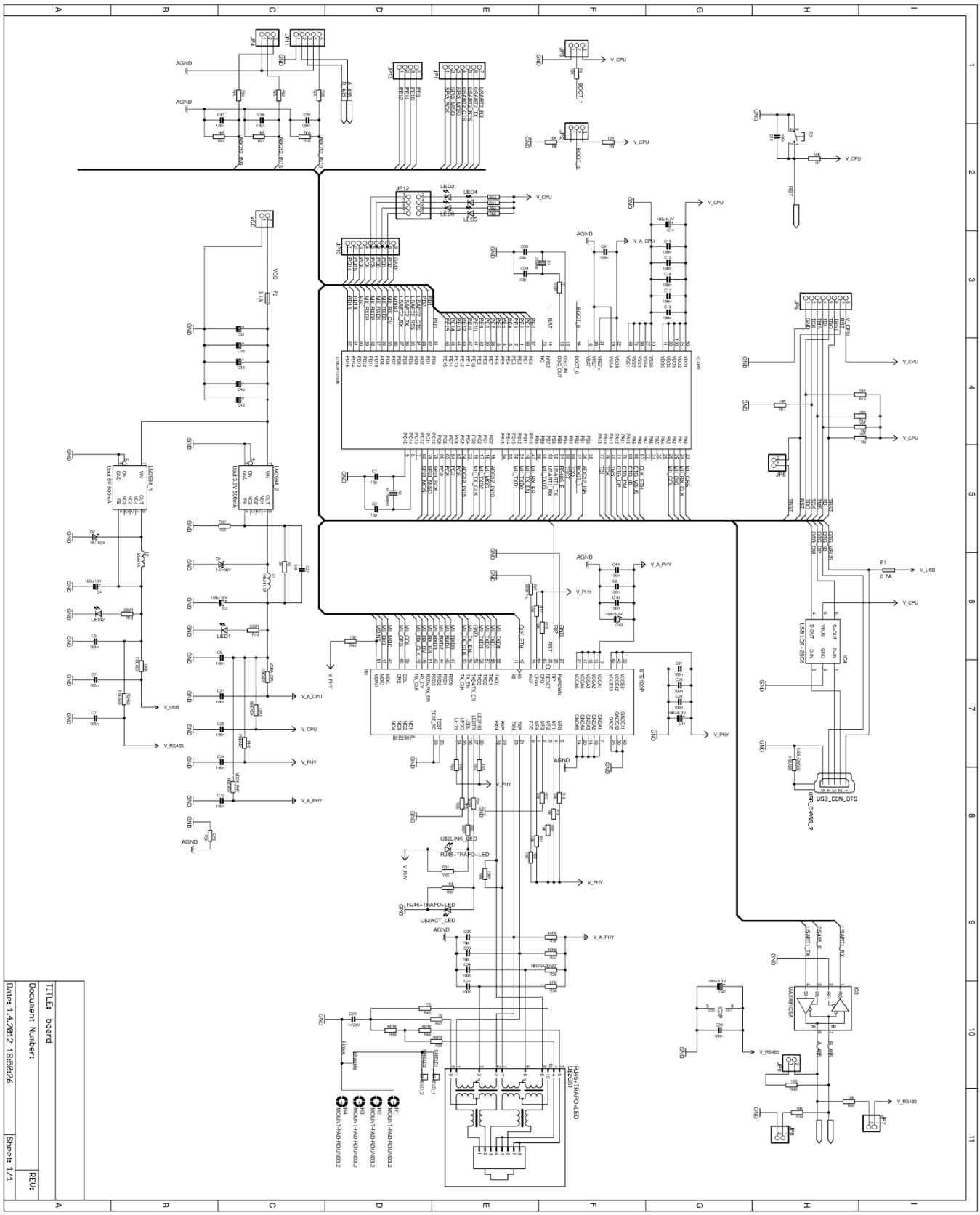
Obrázek 38: Osazovací plán strany TOP



Obrázek 39: Deska plošného spoje - BOTTOM



Obrázek 40: Osazovací plán strany BOTTOM



Obrázek 41: Schéma zapojení