

Univerzální webový řídicí a monitorovací systém

General Web Control and Monitoring System

Bc. Oldřich Vykydal

2012



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Oldřich VYKYDAL**
Osobní číslo: **A10924**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Bezpečnostní technologie, systémy a management**
Téma práce: **Univerzální webový řídicí a monitorovací systém**

Zásady pro vypracování:

1. Seznamte se s problematikou vestavěných monitorovacích a řídicích systémů.
2. Specifikujte požadavky pro práci takového systému se vzdáleným přístupem.
3. Po dohodě s vedoucím práce se seznamte s vybraným vývojovým kitem na ARM platformě.
4. Zprovozněte na kitu základní periferie související se sběrem dat z technologického prostředí.
5. Zprovozněte síťovou komunikaci a vytvořte univerzální komunikační rozhraní.
6. Ověřte na reálné úloze Vámi vyvinuté zařízení a specifikujte jeho uplatnitelnost.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:


1. LIČEV, Lačezar; MORKEŠ, David. *Procesory – architektura, funkce, použití*. Computer Press, Praha, 1999, ISBN 80-7226-172-X.
2. VÁŇA, V.: *Arm pro začátečníky*, BEN technická literatura, Praha, 2009, ISBN-80-7300-2.
3. http://pandatron.cz/?542&at91sam7se512._schema_zkusebni_desky.
4. http://pandatron.cz/?606&at91sam7s._1.dil:seznamenis_obvody.
5. DOLINAY, J.: *Programové moduly řídicích a identifikačních algoritmů pro mikropočítač 68HC11*. Diplomová práce. FT Zlín, 2002.
6. DOSTÁLEK, P.: *Aplikace mikropočítačů Motorola pro řízení technologických procesů*. Diplomová práce. FT Zlín, 2002.
7. VLACH, J.: *Řízení a vizualizace technologických procesů*. BEN technická literatura, Praha, 1999, ISBN 80-86056-66-X.
8. SROVNAL, V.: *Operační systémy pro řízení v reálném čase*. VŠB-TU Ostrava, 2003, ISBN 80-248-0503-0.

Vedoucí diplomové práce: **prof. Ing. Vladimír Vašek, CSc.**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **24. února 2012**

Termín odevzdání diplomové práce: **15. května 2012**

Ve Zlíně dne 24. února 2012


prof. Ing. Vladimír Vašek, CSc.
děkan




doc. RNDr. Vojtěch Křesálek, CSc.
ředitel ústavu

ABSTRAKT

Cílem této diplomové práce je navrhnout a realizovat univerzálně použitelný řídicí a monitorovací systém určený pro vestavěné zařízení. Systém bude navrhnout s důrazem na modulární přístup tak, aby ho bylo možno relativně jednoduše rozšiřovat o vrstvy zapouzdřující konkrétní typy použitých hardwarových prvků a také o samotné monitorovací a řídicí aplikace.

Klíčová slova:

Vestavěný systém, monitorování, řízení, mikrokontrolér, ARM Cortex-M3, LPCXpresso, TCP/IP, ethernet, webový server, 1-Wire, DS18B20

ABSTRACT

The aim of this thesis is to design and implement universally applicable control and monitoring system for embedded devices. The system will be designed with focus on a modular approach so that it could be relatively easily extended it with layers encapsulating the specific types of hardware components as well as monitoring and control applications.

Keywords:

Embedded systém, monitoring, control, microcontroller, ARM Cortex-M3, LPCXpresso, TCP/IP, ethernet, web server, 1-Wire, DS18B20

Poděkování

Rád bych na tomto místě poděkoval vedoucímu své bakalářské práce panu prof. Ing. Vladimíru Vaškovi CSc. za cenné rady a pomoc, kterou mi věnoval při práci na tomto projektu.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- **že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.**

Ve Zlíně

.....

podpis diplomanta

OBSAH

ÚVOD.....	10
TEORETICKÁ ČÁST.....	11
1 VĚSTAVĚNÁ ZAŘÍZENÍ.....	12
1.1 ÚVOD DO VESTAVĚNÝCH ZAŘÍZENÍ.....	12
1.2 OBECNÁ CHARAKTERISTIKA.....	12
1.3 HISTORIE.....	13
1.4 BĚŽNÁ STRUKTURA.....	14
1.4.1 MIKROKONTROLÉR.....	14
1.4.2 VSTUPNĚ/VÝSTUPNÍ ROZHRANÍ.....	18
1.4.3 UŽIVATELSKÉ ROZHRANÍ.....	19
1.5 VYUŽITÍ V TECHNOLOGICKÝCH PROCESECH.....	20
2 ARM ARCHITEKTURA.....	21
2.1 ÚVOD.....	21
2.2 ZÁKLADNÍ CHARAKTERISTIKA.....	21
2.3 INSTRUKČNÍ SOUBOR.....	23
2.4 PŘEHLED ARM MIKROKONTROLÉRŮ.....	24
2.5 VÝVOJ APLIKACÍ NA ARM.....	25
2.5.1 CMSIS.....	26
3 SÍŤOVÁ KOMUNIKACE.....	28
3.1 ÚVOD.....	28
3.2 ISO/OSI REFERENČNÍ KOMUNIKAČNÍ MODEL.....	28
3.3 PŘEHLED KOMUNIKAČNÍCH TECHNOLOGIÍ.....	30
3.3.1 GSM.....	31
3.3.2 ETHERNET IEEE 802.3.....	33
PRAKTICKÁ ČÁST.....	39
4 NÁVRH SYSTÉMU.....	40
4.1 ZÁKLADNÍ POŽADAVKY NA SYSTÉM.....	40
4.2 SPECIFIKACE SYSTÉMU.....	40
4.3 NÁVRH ŘEŠENÍ SYSTÉMU.....	40
5 VÝVOJOVÝ KIT.....	42
5.1 ÚVOD DO VÝBĚRU VÝVOJOVÉHO KITU.....	42

5.1.1 VÝVOJOVÝ KIT OBECNĚ.....	42
5.2 PŘEHLED VHODNÝCH VÝVOJOVÝCH KITŮ.....	43
5.2.1 MBED.....	43
5.2.2 LUMINARY MICRO LM3S696 ETHERNET EVALUATION KIT.....	44
5.2.3 LPCXPRESSO 1769.....	47
5.3 PRÁCE S LPCXPRESSO.....	47
5.3.1 ÚVOD DO VÝVOJOVÉHO PROSTŘEDÍ LPCXPRESSO.....	47
5.3.2 DEBUGOVÁNÍ LPCXPRESSO.....	49
6 TCP/IP STACK.....	51
6.1 ÚVOD.....	51
6.2 VÝBĚR IMPLEMENTACE TCP/IP.....	52
6.2.1 TESTOVACÍ TCP/IP z CMSIS BALÍKU.....	52
6.2.2 NICHELITE TCP/IP.....	52
6.2.3 LWIP.....	53
6.3 PRAKTICKÉ VYUŽITÍ LWIP V PROJEKTU.....	54
6.3.1 DRUHY LWIP API.....	54
6.3.2 NAsAZENÍ LWIP.....	54
6.3.3 WEB SERVER.....	56
7 MONITOROVÁNÍ A ŘÍZENÍ.....	57
7.1 MONITOROVÁNÍ.....	57
7.2 ŘÍZENÍ.....	57
7.3 REFERENČNÍ APLIKACE.....	58
7.3.1 1-WIRE.....	58
7.3.2 SENZOR TEPLoty DS18B20.....	58
8 PROPOJENÍ ČÁSTÍ SYSTÉMU DO CELKU A TESTOVÁNÍ.....	60
8.1 MODULÁRNÍ STRUKTURA SYSTÉMU.....	60
8.2 TESTOVÁNÍ.....	61
9 ZHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ A MOŽNOSTI VYUŽITÍ NAVRŽENÉHO ŘEŠENÍ.....	63
9.1 ZREALIZOVANÉ CÍLE.....	63
9.2 MOŽNÁ ROZŠÍŘENÍ.....	63
ZÁVĚR.....	64
ZÁVĚR V ANGLIČTINĚ.....	65
SEZNAM POUŽITÉ LITERATURY.....	66
SEZNAM POUŽITÝCH SYMBOLU A ZNAČEK.....	69

SEZNAM OBRÁZKU.....	70
SEZNAM TABULEK.....	71
SEZNAM PŘÍLOH.....	72

ÚVOD

V rámci této diplomové práce bude zpracován projekt na vytvoření univerzálně využitelného základu pro systém na monitorování a řízení technologických procesů. Realizovaný systém bude směřovat do oblasti vestavěných zařízení, pracujících v prostředí počítačových sítí. Bude se tak jednat o zařízení umožňující vzdálený přístup i správu. Půjde o kombinovanou vývojovou práci s hardware i software.

V první kapitole budou obecně probrány vestavěné zařízení, následně bude představen mikrokontrolér jakožto jejich stěžejní součást. V druhé kapitole se budeme věnovat podrobněji ARM mikroprocesorové architektuře, která tvoří jednu z nejzajímavějších platforem pro mikrokontroléry. Třetí kapitola uvede některé síťové komunikační technologie, které budou použity pro vzdálený přístup. Následující čtvrtá kapitola bude zároveň první praktickou kapitolou a budou zde shrnuty požadavky na vyvíjený systém a bude provedený návrh architektury systému. V páté kapitole se budeme zabývat výběrem vývojového kitu, který bude sloužit jako hardwarová část projektu. V další šesté kapitole budou představeny implementace TCP/IP rodiny protokolů a bude předvedena práce s vybranou implementací. Sedmá kapitola se bude věnovat realizaci monitorovací a řídicí části. V osmé kapitole se všechny komponenty spojí do funkčního systému. A v poslední deváté kapitole bude uvedeno resumé.

I. TEORETICKÁ ČÁST

1 VĚSTAVĚNÁ ZAŘÍZENÍ

V této kapitole se věnuji vestavěným zařízením obecně, informuji zde o jejich historii, popisuji jejich základní charakteristiky a dále upřesňuji jejich místo v technologických procesech.

1.1 Úvod do vestavěných zařízení

Termín „vestavěné zařízení“, „vestavěný systém“, nebo také „zabudované zařízení“, či „embedded system“, si nebude většina lidí příliš schopna přesněji představit. Přitom se v dnešní době jedná o zcela běžnou věc. Vždyť téměř každý z nás dennodenně používá domácí elektrospotřebiče, cestuje nějakým dopravním prostředkem, pracuje s počítačem, nebo třeba jen komunikuje s dalšími lidmi pomocí mobilních telefonů. Život bez nich si dnes lze jen těžko představit, kdo by pro nás automaticky reguloval ohřev v automatické mikrovlnné troubě, nebo řídil funkci motoru v automobilu? Či dokonce jak bychom dokázali pracovat s počítači, když např. myš, klávesnice, nebo monitor jsou ukázkovými vestavěnými zařízeními? Myslím, že by to šlo velmi těžko, protože právě pro usnadnění opakujících se úkonů, které jsou jasně omezeny, jsou vestavěná zařízení primárně využívána.

1.2 Obecná charakteristika

Vestavěné zařízení lze definovat jako „jednoúčelové“ zařízení, ve kterém je řídicí logika (většinou reprezentována nějakým typem mikrokontroléru) do tohoto zařízení přímo zabudována. Samotná „jednoúčelovost“ tohoto zařízení může být přitom relativním pojem, protože spektrum činností vykonávaných v dnešní době vestavěnými zařízeními již může být poměrně široké. Ovšem na rozdíl od běžného počítače (PC – Personal Computer), který je obecně univerzální (může vykonávat nejrůznější a hlavně předem neomezené činnosti), je vestavěné zařízení běžně konstruováno pro předem jasně definované činnosti a aplikace. Vestavěné zařízení je konkrétní aplikaci již takřikajíc přímo „šito na míru“ a možnost změny dané funkce v průběhu životnosti vestavěného zařízení je poměrně omezená. [1],[6]

Běžně uváděné skutečnosti u vestavěných systémů:

- díky předem definovanému určení aplikace lze vestavěné zařízení pro tuto konkrétní aplikaci lépe optimalizovat
- lze tak zvýšit efektivitu výroby a jelikož se vestavěné zařízení většinou vyrábějí ve velkých sériích, lze takto podstatně snížit cenu
- také jde většinou o efektivnější zařízení ve vztahu ke spotřebě energií, ale i k výkonosti než je tomu u univerzálních systémů
- u vestavěných aplikací nelze mimo výrobu tak snadno měnit jejich funkčnost a mnohdy to nebývá možné vůbec
- některá vestavěná zařízení pracují v real-time režimu, kdy musí být ve vztahu k době vykonání přesně daná doba odezvy systému

1.3 Historie

Jako spousta jiných technických vynálezů byly vestavěné systémy v jejich počátcích vyvíjeny pro vojenské a strategické účely. Jedním z prvních obecně známých moderních vestavěných systémů byl navigační počítač využitý ve vesmírném programu Apollo[8]. Nazýval se Apollo Guidance Computer (AGC) a byl vyvinut na začátku šedesátých let 20. století v MIT v Instrumentation Laboratory pod vedením Charlese Starka Drapera. AGC byl vytvořen z cca 4 100 integrovaných obvodů každý obsahující třívstupovou NOR logické hradlo.

Prvním hromadněji vyráběným vestavěným zařízením byl vojenský navigační systém D-17[6] vyrobený firmou Autonetic pro rakety Minuteman, uvedený v roce 1961. Byl vytvořen z jednotlivých tranzistorů, jeho nástupce o 5 let později již využíval integrovaných obvodů. Díky hromadné výrobě těchto navigačních počítačů časem znatelně klesla cena jak integrovaných obvodů, tak i vestavěných zařízení, která tyto obvody využívala. To byl krok směrem k masovému využití vestavěných zařízení i v komerčním sektoru.

Jak postupovala integrace čipů do stále menších rozměrů, tak také probíhal přesun původně externích součástek jako např. různé druhy pamětí apod. na jeden čip spolu s procesorem, což dalo vzniknout nové součástce, která se začala označovat jako mikrokontrolér nebo jednočipový počítač, či mikropočítač. Jednalo se o zásadní průlom v oblasti vestavěných zařízení, které umožnilo jejich opravdu masové nasazení do mnoha

oblastí lidské činnosti. Vestavěná zařízení se tak hlavně v podobě nejrůznější spotřební elektroniky stala zcela běžnou součástí všedního života. A tento trend dále pokračuje.

1.4 Běžná struktura

Obecně bývají vestavěná zařízení složena z řídicí části (hlavně mikrokontroléry) a části navazující na fyzické prostředí (různá průmyslová rozhraní a sběrnice). Tyto dvě části jsou v běžných vestavěných systémech obsaženy opravdu vždy, protože řídicí logika bez vazby na nějaký fyzický systém, by postrádala smysl a obráceně o systémech, které by měly pouze vazby na fyzické prostředí bez řídicí logiky, se nemá asi cenu ani dále zmiňovat. U vestavěných zařízení se ještě může vyskytovat další část nějakým způsobem komunikující s lidskou obsluhou. Jedná se o různé ovládací a zobrazovací jednotky. Sice jde o jistě důležitou součást, ale i tak existuje spousta vestavěných zařízení, která jsou schopna po celou dobu jejich životnosti pracovat bez jakéhokoliv zásahu zvenčí.

1.4.1 Mikrokontrolér

Někdy nazývaný jednočipovým počítačem, nebo MCU. Jedná se o nejdůležitější prvek ve většině vestavěných zařízení, mnohdy v sobě přímo integruje celou aplikaci. Hlavním rysem mikrokontrolérů je spojení procesorové jednotky, různých druhů pamětí (ROM, RAM, Flash), řadičů fyzických rozhraní a dalších podpůrných zařízení (oscilátor, řadič přerušení, čítače, watchdog časovač apod.) do jednoho fyzického pouzdra – jednoho čipu. Jelikož se MCU vyznačují zejména velkou kompaktností a spolehlivostí, tak bývají nasazovány primárně právě do vestavěných zařízení. MCU je potom nedělitelnou součástí, jejíž adekvátní výběr následně zásadně ovlivňuje celou vestavěnou aplikaci [6].

Hlavním kritériem při výběru MCU z pohledu návrháře vestavěných systémů je kromě ceny (u velkých sérií je to asi opravdu nejdůležitější parametr ovlivňující celkovou komerční úspěšnost dané aplikace, která se samozřejmě přímo odvíjí od možností daného MCU), výběr šířky slova. Lze vybírat mezi 4 až 64 bitovými mikrokontroléry, přičemž se uvádí [7], že nejčastěji využívané MCU jsou 8 bitové. Výběrem šířky slova určíme maximální možnou velikost paměti, kterou budeme moci adresovat a tím pádem i v podstatě určíme možnou „obsáhlost“ celé vestavěné aplikace.

Víme-li kolika bitovou architekturu budeme chtít využívat, můžeme se zabývat jiným stěžejním tématem při výběru MCU a tím je volba frekvence. Frekvence, laicky uváděná jako „rychlost“ MCU či procesoru, nám vyjadřuje takt jádra MCU. Tedy jak dlouho trvá jeden hodinový cyklus. Historie vývoje PC by se do příchodu multiprocesorového programování dala nazvat historií honby za co nejvyšší frekvencí procesorů. Frekvence byla dlouho tím hlavním parametrem, který určoval výkon aplikací. Toto tedy již není tak úplně pravda, kdybychom chtěli porovnávat výpočetní sílu procesorů, tak bychom se museli zabývat i jinými parametry (kromě multiprocesorovosti, tak i architekturou práce s instrukcemi, pipeliningem apod.). U vestavěných zařízení je situace odlišná od situace v PC ještě v tom, že jelikož jsou vestavěná zařízení konstruovaná pro předem daný záměr, tak mnohdy ani nemusí být žádný důvod k využití rychlejších MCU. Vyšší rychlost MCU by prostě danou aplikaci nemuselo vůbec nijak vylepšit. Právě i naopak, protože od frekvence MCU se odvíjí i spotřeba a tím pádem, by rychlejší a většinou i dražší MCU konzumoval více elektrické energie, což není v oblasti vestavěných zařízení v žádném případě zanedbatelný parametr. V hrubých rysech ovšem frekvenci jádra můžeme vzít alespoň za jakési vodítko k představení si toho, jak by mohl být daný MCU výkonný, ale čistě jen podle frekvence se rozhodovat nelze.

Dalším kritériem je rozložení a velikost paměti v mikrokontroléru. Rozeznáváme dvě různé architektury organizace paměti – Von Neumannovou a Hardvardskou [7]. Hlavní rozdíl je v oddělení, či neoddělení dynamických dat programu od kódu programu. MCU potom tedy bude mít buď jednu společnou datovou sběrnici, nebo dvě na sobě nezávislé datové sběrnice. Dnes lze ovšem už říci, že jde o rozdíl spíše minoritní, protože i když moderní MCU využívají hlavně oddělenou paměť pro data a pro kód, tak uživatelé většinou zprostředkují abstrakci, kdy se celkový datový prostor MCU jeví jako lineární. Následuje tabulka s porovnáním architektur [7].

Von Neumannova architektura	Hardvardská architektura
společná paměť pro data i program	oddělená paměť pro data a program
jedna datová sběrnice	více než jedna datová sběrnice
jeden druh instrukcí pro přístup k paměti dat a k paměti programu	více druhů instrukcí pro přístup k paměti dat a k paměti programu

jednodušší implementace	složitější implementace
menší rychlost přenosů po datové sběrnici	větší rychlost přenosů po datové sběrnici

Tabulka [1] – Rozdíly mezi architekturou Von Neumannovou a Harvardskou

Co se týká možností výběru MCU dle samotné velikosti paměti, tak tam je situace velmi dobrá a výběr je zde opravdu obsáhlý. Jelikož se častěji využívá Harvardská architektura, tak se rozlišuje mezi pamětmi RAM a ROM (flash). Kde paměti typu RAM jsou určeny pro uložení zpracovávaných dat programu, jsou volatilní – tj. drží si data jen v případě přiloženého napájení. A paměti typu ROM, které se dají pouze číst a slouží pro uložení kódu programu, který se v průběhu jeho vykonávání nemění. ROM paměti jsou také nevolatilní – tj. data v těchto pamětech zůstanou i po odpojení napájení. Dnes se jako paměť pro uložení kódu používají nejčastěji paměti typu flash, které lze libovolně programovat. MCU vybíráme tedy podle toho, kolik paměti bude třeba použít k uložení kódu (flash) a kolik paměti pro běh programů (RAM). Přičemž bychom neměli používat MCU se zbytečně velkými paměťmi když už víme, že je nebudeme potřebovat. Rozdíl ve velikostech paměti dělá poměrně podstatný rozdíl mezi cenami MCU, které jsou jinak parametrově stejné.

Dále lze u MCU rozlišovat mezi CISC [9] (Complex Instruction Set Computer) a RISC [10] (Reduced Instruction Set Computer) typem instrukční sady. Jde o to jakým způsobem se pracuje s programovými instrukcemi, jestli na jeden druh operace stačí jedna složitější instrukce, nebo se využije několika jednodušších.

CISC instrukční sada	RISC instrukční sada
více druhů a formátů instrukcí, obecně složitější instrukce	jednoduchá a vysoce optimalizovaná sada instrukcí
vyšší hustota kódu programu	nižší hustota kódu programu
podstatně složitější dekodér instrukcí	jednoduchý dekodér instrukcí
pomalejší zpracování instrukcí	rychlejší zpracování instrukcí

Tabulka [2] – Rozdíly mezi CISC a RISC instrukční sadou

Dle potřeby spolupráce s jinými fyzickými zařízeními (řízení, monitoring, komunikace, atd.) se při návrhu vestavěné aplikace vybere v ideálním případě MCU takové, které bude

mít všechny řadiče fyzických rozhraní již přímo implementováno uvnitř MCU. O různých druzích vstupně/výstupních rozhraní bude pojednávat další samostatná podkapitola. Teď následuje krátký přehled nejznámějších mikrokontrolérů.

Název	Šířka instrukcí	Architektura
ARM	32	RISC
Atmel AVR	8	RISC
AVR32	32	RISC
Freescale 68HC11	8	CISC
Intel 8051	8	smíšená
TI MSP430	16	RISC

Tabulka [3] – Přehled nejznámějších MCU

1.4.2 Vstupně/výstupní rozhraní

Mikrokontroléry dnes zcela běžně obsahují velké množství vstupně/výstupních rozhraní. Právě jejich přítomností nebo naopak absencí je dosaženo velké škály dostupných MCU a možností jejich výběru. Výrobci běžně dodávají několik základních řad MCU, jejichž zástupci mají společný typ jádra (možnost stejné frekvence) a liší se mimo velikosti paměti také tím, že jeden konkrétní typ MCU má v sobě integrován řadič pro určité fyzické rozhraní či sběrnici (např. řadič sítě ethernet, řadič sběrnice CAN, USB apod.) a jiný ho naopak nemá. Tyto řadiče lze téměř vždy implementovat i mimo samotný MCU, ale pokud je na trhu dostupný MCU, který má námi požadovaný řadič přímo v sobě a v ostatních parametrech nám také vyhovuje, tak je lepší a většinou i levnější, takový MCU přímo použít. Lze si tak ulehčit při implementaci, protože práce s takovýmto řadičem bude přímo možná přes registry MCU místo nutnosti implementovat ještě další komunikaci mezi MCU a řadičem (např. Pomocí SPI apod.). Následuje přehled běžně využívaných rozhraní v MCU.

Název	Použití
GPIO	obecné V/V porty, lze je např. naprogramovat pro implementaci onewire
RS-232	sériová linka, primárně pro komunikaci mezi PC a spotřební elektronikou
RS-485	sériová linka, využívaná hlavně v průmyslovém prostředí

A/D	analogově/digitální převodník
D/A	digitálně/analogový převodník
SPI	sériové periferní rozhraní, komunikace mezi MCU a jinými integrovanými obvody
USB	univerzální sériové rozhraní – moderní způsob připojení periférií
Ethernet	rodina protokolů pro vytvoření lokálních počítačových sítí
CAN	controller area network – průmyslová komunikace MCU a nejrůznějších senzorů
I ² C	inter-integrated circuit – dvou vodičová sériová komunikace pro připojování nízkorychlostních periférií

Tabulka [4] – Přehled nejpoužívanějších V/V rozhraní

1.4.3 Uživatelské rozhraní

Mnoho vestavěných zařízení s uživatelským rozhraním vůbec nepočítá a pro zabezpečení své funkce ho ani nepotřebuje, ale je i dost aplikací, kde je nějaká interakce s uživatelem vhodná, nebo dokonce přímo nutná. Takže se napříč vestavěnými systémy můžeme potkávat se zařízeními kompletně bez uživatelského rozhraní, přes jednoduchá rozhraní s tlačítky, informačními diodami, jednoduchými segmentovými display, jednoduchými klávesnicemi až po zařízení s plnohodnotnými monitory a např. dotykovým ovládním.

Samostatnou a velmi zajímavou kapitolou v oblasti uživatelských rozhraní u vestavěných systémů je interakce MCU v těchto zařízeních přes různá komunikační rozhraní se vzdálenými zařízeními, která jsou přizpůsobena pro zobrazování informací a ovládním vzdálených vestavěných zařízení. Častou implementací v této oblasti je připojení MCU ve vestavěném zařízení do lokální počítačové sítě pomocí řadiče ethernetu. Ve vestavěném zařízení je implementován TCP/IP protokolový stack, díky němuž je poté možno v lokální počítačové síti s vestavěným zařízením komunikovat pomocí standardních síťových protokolů. Tato komunikace může vypadat tak, že ve vestavěném zařízení běží nějaký „odlehčený“ http server, který přijímá požadavky od klientského softwaru (běžného webového prohlížeče) a na tyto požadavky adekvátně reaguje. Ať už zobrazením webové stránky s požadovanými údaji z nitra vestavěného systému, nebo vzdáleným vyvoláváním

vnitřních funkcí vestavěného systému (např. pomocí vyplňování a odesílání webových formulářů apod.). Je-li lokální síť, do které je vestavěné zařízení připojeno, navíc napojena na internet, tak se potom dá vestavěné zařízení ovládat obecně z celého světa. Tato skutečnost je velkým plusem, ovšem také není zadarmo. Implementace síťové komunikace na vestavěných zařízeních není úplně přímočará – musíme většinou velmi úzkostlivě pracovat se systémovými zdroji, protože komunikace bývá na systémové zdroje docela náročná. A u systémů, které mají zdroje poměrně omezené to může být obecně problémem.

1.5 Využití v technologických procesech

Velká možnost využití vestavěných systémů tkví v oblasti řízení a monitoringu obecně jakýchkoliv technologických procesů. Podstatnou informací pro určení toho, jestli vybraný technologický proces půjde řešit v rámci aplikace vestavěných systémů, je hlavně dostupnost senzorů pro získávání relevantních údajů z daného technologického procesu a následně dostupnost ovládacích (výkonných) prvků, kterými je tento tech. proces možno automatizovaně řídit, korigovat, usměrňovat, či přerušovat apod. Máme-li tedy senzory a výkonné prvky, můžeme začít uvažovat o vytvoření vestavěného systému pro interakci s technologickým procesem.

Tech. procesy lze jen pasivně sledovat, dozorovat či monitorovat, tedy získávat z nich informace o jejich aktuálním stavu, spotřebovávaných zdrojích, hlásit poruchy obsluze apod. V oblasti vestavěných systémů jsou ale častější aplikace, kdy se do tech. procesů nějakým způsobem přímo vstupuje. Můžeme potom hovořit o tom, že nějaký vestavěný systém daný tech. proces řídí, či reguluje.

2 ARM ARCHITEKTURA

Na tomto místě uvádím ARM platformu jako předního zástupce mikropočítačů používaných ve vestavěných zařízeních. Dále zde přibližuji zásadní věci z ARM architektury samotné a popisuji co z toho plyne pro programátory a jak se s ARM mikroprocesory dá pracovat.

2.1 Úvod

ARM je běžně užívanou zkratkou z označení Advanced RISC Machine. Je zde vyjádřeno, že jde o pokročilé mikroprocesory vycházející z RISC architektury. ARM procesorová architektura je vyvíjena britskou firmou ARM Limited. Vývoj odstartoval již počátkem 80. let minulého století, přičemž první mikroprocesor byl vydán v roce 1984. Již v té době šlo o 32 bitový mikroprocesor, což v 80. letech nebylo ani zdaleka standardem. Doménou mikroprocesorů založených na ARM architektuře jsou zejména vestavěná zařízení, kde se využívá relativně vysokého výkonu ARM platformy při malé provozní spotřebě. Nejvýkonnější mikroprocesory se už ovšem začaly objevovat i v klasických PC.

Postupem času přestala firma ARM sama mikroprocesory vyrábět a začala se soustředit pouze na jejich vývoj. To umožnilo vzniknout zajímavému obchodnímu modelu, kdy většina známých výrobců polovodičů platí firmě ARM za licence a sami poté tyto mikroprocesory s různými modifikacemi vyrábí. Prodejní čísla mikroprocesorů založených na ARM architektuře a zejména naprostá dominance v segmentu mobilních vestavěných zařízení ukazuje, že jde zřejmě o správnou cestu k úspěchu [12].

2.2 Základní charakteristika

Jedná se již od prvopočátku o 32 bitové mikroprocesory s Load/Store architekturou. Architektura je optimalizovaná pro velmi rychlou práci s registry. Pokud se ovšem musí provádět výpočty nad daty uloženými v paměti, tak narazíme na podstatná omezení. Všechna data se z paměti musí nejprve přenést do registrů, tam zpracovat a následně přenést zpět. Instrukce pro přímou práci s daty v ARM architektuře neexistuje. Přesuny z paměti do registrů a zpět tvoří velkou část zdrojového kódu (v assembleru) ve většině reálných aplikací. Přístupy do paměti však lze většinou optimalizovat, čímž nám tento princip ve výsledku přinese více, než nám vezme [11].

ARM architektura umožňuje několik režimů provádění mezi nimiž se lze přepínat. Původně šlo o tyto režimy:

- uživatelský – nepriviligovaný běžný režim
- FIQ – fast interrupt – pro přerušení s vysokou prioritou
- IRQ – interrupt – pro přerušení s běžnou prioritou
- supervisorský – pro softwarové přerušení a pro použití po resetu
- ukončující – při problémech s přístupem do paměti
- nedefinovaný – v případě nedefinovaných instrukcí

V generaci ARM4 přibyl ještě:

- systémový – privilegovaný režim využívající stejné registry jako uživatelský

V rodině Cortex-M3 jsou ovšem používané režimy pouze dva [11]:

- thread – vláknový, může být privilegovaný i nepriviligovaný (ten je právě něco jako uživatelský režim)
- handler – pro obsluhu přerušení, je vždy privilegovaný

Co se týká registrů v mikroprocesoru, tak všechny jsou samozřejmě 32 bitové, jejich funkce se může v různých generacích lišit (hlavně ve vztahu s různými režimy provádění). Univerzálně ovšem platí, že ve všech režimech provádění je vždy dostupné 15 registrů od R0 do R14 a registr R15, který slouží jako programový čítač. Dále jsou přítomny stavové registry CPSR, případně i SPSR. Pro přesnou informaci je vždy potřeba nahlédnout do datasheetu konkrétního mikroprocesoru.

Mikroprocesory ARM umožňují také dva druhy adresace:

- pomocí čítače instrukcí – implicitní mód, hodnota čítače ukazuje na následující instrukci
- pomocí báze adresy – uložené v jednom z obecných registrů, adresa další instrukce se zjistí po přičtení offsetu k bázi

2.3 Instrukční soubor

Jelikož je ARM architektura přísně RISCová, tak všechny instrukce v ARM instrukční sadě mají stejnou šířku 32 bitů. To umožňuje řetězení do pipeline, z čehož plyne, že v jednom okamžiku může být zpracováváno vícero instrukcí. Samozřejmě se každá z nich musí nacházet v jiné fázi zpracování. V případě ARM7 a nižších a poté rodiny Cortex-M to jsou FETCH (načtení instrukce z paměti), DECODE (dekódování instrukce) a EXECUTE (vykonání instrukce) [22].

ARM procesory rozlišují instrukce pro:

- zpracování dat – aritmetické a logické operace, porovnání, přenos dat mezi registry
- instrukce skoku – skok, skok s odkazem
- posuvy – pouze přes válcový posouvač
- práce s pamětí – přenos dat mezi jedním registrem a pamětí, blokový přenos dat, prohození dat
- přerušení – přechody mezi režimy

Každá z ARM instrukcí obsahuje na čtyřech nejvyšších bitech podmínkové pole, které určuje jakým způsobem a jestli vůbec bude daná instrukce provedena. Následuje tabulka s popisem toho, jak a kdy se jednotlivé příznaky mění.

Příznak	Logická instrukce	Aritmetická instrukce
Negative N = 1	příznak se nemění	byl nastaven 31 bit výsledku, indikuje záporný výsledek u operací se znaménkem
Zero Z = 1	výsledkem jsou jen nuly	výsledek operace byl záporný
Carry C = 1	po operaci posuvu kdy 1 zůstala v carry	výsledek byl větší než 32 bitů
oVerflow V = 1	příznak se nemění	výsledek byl větší než 31 bitů, indikuje ztrátu znaménka u operací se znaménkem

Tabulka [5] – Význam příznakových bitů v instrukcích ARM

Kromě 32 bitové instrukční sady mají některé ARM mikroprocesorové rodiny ještě tzv. Thumb instrukční sadu. Tato instrukční sada je 16 bitová a obsahuje nejpoužívanější instrukce z klasické 32 bitové sady pouze zmenšené na polovinu. Při práci s 16 bitovými instrukcemi stačí napájet pouze polovinu sběrnice, v některých aplikacích tak můžeme dosáhnout zajímavého ušetření ve spotřebě elektrické energie. Před provedením Thumb instrukce je potřeba 16 bitovou instrukci „dekomprimovat“ na 32 bitů.

2.4 Přehled ARM mikrokontrolérů

Vzhledem k tomu, že se zde zabýváme vestavěnými zařízeními, tak se dále budeme bavit pouze o mikroprocesorech, které jsou pro tyto účely přímo určené. Jedná se o moderní rodinu mikrokontrolérů ARM Cortex-M [11].

Tato rodina je založena na ARM verze 6M a 7M/ME, které oproti předchůdcům přenesly několik zajímavých vylepšení:

- NVIC – Nested Vectored Interrupt Controller – umožňuje vylepšenou obsluhu přerušení s menší odezvou, již není třeba vůbec využívat assembler, i vektory přerušení lze definovat v čistém C
- WIC – Wake-up Interrupt Controller – lepší a méně náročná obsluha přerušení v módech spánku
- Thumb-2 – ARMv7 navíc přinesl ještě vylepšenou podporu pro zkrácené instrukce
- CMSIS – Cortex Microcontroller Software Interface Standard – abstrakční vrstva nad hardwarem použitelná pro rychlý vývoj

Následuje tabulka s přehledem členů rodiny Cortex-M:

Architektura	Jádro	Zajímavé funkce	Výkonnost
ARMv6-M	Cortex-M0	Thumb instrukce, hardwarové násobení	0,9 DMIPS / MHZ
	Cortex-M1	Thumb instrukce, hardwarové násobení	0,8 DMIPS / MHZ
ARMv7-M	Cortex-M3	Thumb-2 instrukce, hardwarové násobení a dělení, možnost přístupu k jednotlivým bitům v paměti	1,25 DMIPS / MHZ
ARMv7-ME	Cortex-M4	Thumb-2 instrukce, hardwarové násobení a dělení, DSP funkčnost, možnost FPU, možnost přístupu k jednotlivým bitům v paměti	1,25 DMIPS / MHZ

Tabulka [6] - Přehled členů rodiny Cortex-M

2.5 Vývoj aplikací na ARM

Jako první věc při vývoji vestavěného zařízení na ARM platformě je nejvhodnější začít pořízením nějakého vývojového kitu. Tento kit by měl ideálně obsahovat stejný mikrokontrolér, který potom bude použit v reálném zařízení. Vývojové kity poskytují nejrůznější periferie, které můžeme použít k rychlému prototypování. Je vhodné pořídit si všechny prvky jako jsou nejrůznější senzory, tlačítka, klávesnice, display apod. (pokud je vývojový kit už neobsahuje) a adekvátně je propojit. Následně vytvořit a odladit firmware, který by měl být pouze s drobnými modifikacemi použitelný i přímo ve vyráběném vestavěném systému. Celou aplikaci si tak lze vyzkoušet „na sucho“ před vytváření návrhu desky plošných spojů, její výrobou, osazováním apod.

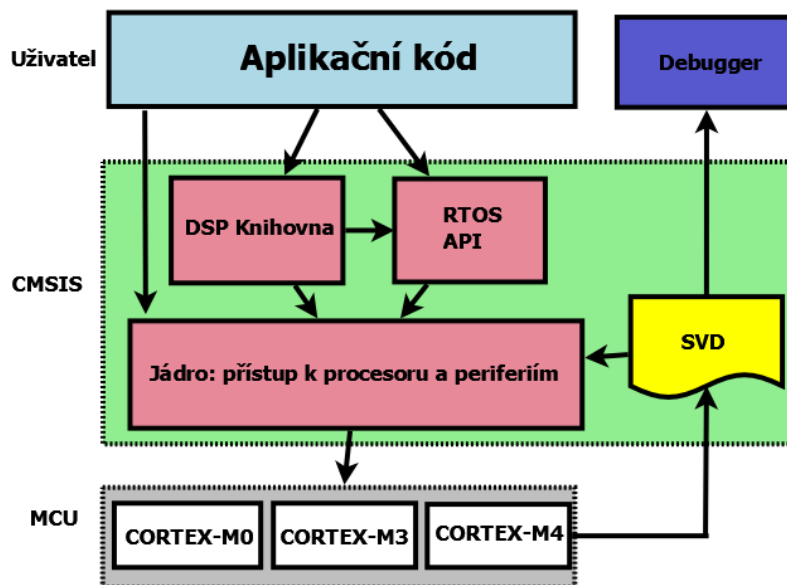
Máme-li nějakým způsobem vyřešen výběr hardware, můžeme se zabývat vývojem software. Existuje mnoho prostředků pro vývoj aplikací na ARM platformě, od možnosti použití čistého jazyka symbolických instrukcí (assembleru) zapsaném např. běžným textovým editorem, až po velmi sofistikovaná vývojová prostředí umožňující nejenom vytváření kódu v některém z vyšších programovacích jazyků, ale i správu flash paměti, debugování, online nápovědu apod. Většina hlavních výrobců ARM mikrokontrolérů poskytuje pro vývoj na svých produktech právě taková integrovaná prostředí, která dokáží uživateli (programátorovi) velice zjednodušit a zefektivnit práci. Navíc lze tyto prostředky

používat bez toho, aniž bychom se úplně připravili o možnost nízkoúrovňového přístupu, který nám poskytuje assembler a který bývá nutný ve výpočetně kritických aplikacích. V takovýchto případech může být vhodné vytvořit si některé rutiny v assembleru a raději se nespolehat jen na kompilér vyššího jazyka. Toto lze elegantně řešit, protože kód assembleru je možno vložit do kódu vyššího jazyka. Linker se posléze postará, aby všechny komponenty byly správně spojeny do funkčního programu, resp. do firmware, který se nahrává do flash paměti mikrokontroléru. A ten je tímto „programem“ následně ovládán. Toto lze všechno řešit přímo z integrovaných vývojových prostředí, kde se tedy nepřipravujeme o žádné funkce zefektivňující práci a přitom si ponecháváme možnost využití „síly“ assembleru.

2.5.1 CMSIS

S příchodem rodiny mikrokontrolérů ARM Cortex je potřeba využívat assembler při vývoji firmware pro vestavěné zařízení ještě menší než kdy dříve. Byla totiž uvedena obecná, samotný hardware abstrahující vrstva, která se jmenuje CMSIS - Cortex Microcontroller Software Interface Standard. Poskytuje jednoduché softwarové rozhraní pro přístup k mikrokontroléru a jeho perifériím a umožňuje tak jednodušší a rychlejší vývoj. CMSIS byl vyvinut ve spolupráci se všemi hlavními výrobci ARM mikrokontrolérů a byl nimi přijat za standard v této oblasti. Každý výrobce adaptoval CMSIS pro svůj hardware s tím, že softwarové rozhraní zůstává kompatibilní napříč ARM mikrokontrolérovým trhem. CMSIS každý z výrobců distribuuje zdarma v podobě knihovny (v jazyce C), která se připojí do aktuálního projektu. CMSIS knihovna bývá stažitelná z webu výrobce nebo ze stránek firmy ARM. Následuje obrázek znázorňující způsob využití CMSIS [13].

Využití CMSIS



Obrázek [1] - Znáznornění komunikace v ISO/OSI modelu

CMSIS se interně skládá z těchto částí:

- CMSIS jádro: poskytuje přístup k registrům procesoru a periferií
- DSP knihovna – množství funkcí pro operace se signály v plovoucí desetinné čárce
- RTOS API – standardizované programové rozhraní pro real-time operační systémy
- SVD – definice programátorského pohledu na kompletní mikrokontrolér a periferie

3 SÍŤOVÁ KOMUNIKACE

Zde uvádím přehled používaných technologií pro komunikaci u vestavěných zařízení, zmiňuji hlavní výhody a nevýhody. Dále se zaměřuji na technologii ethernet, uvádím hlavní informace týkající se její architektury a následně informuji o obecných věcech týkající se TCP/IP komunikace v ethernetu, tak abych v praktické části mohl navázat výběrem TCP/IP stacku pro svůj projekt.

3.1 Úvod

Vestavěná zařízení bývají mnohdy izolována od ostatního světa a mimo svou vlastní aplikaci s jinými zařízeními nekomunikují. Existuje ale i poměrně hodně vestavěných zařízení, která v rámci své funkce komunikovat s jinými zařízeními přímo potřebují. Navíc roste i počet vestavěných zařízení, která byla dlouho izolovaná, ale v poslední době se začínají připojovat do rozsáhlých komunikačních sítí a plní tak funkce dříve nevídané. Však kdo by si mimo sci-fi literaturu před několika málo lety představil, že by se jeho automobil (potažmo v něm vestavěný řídicí systém) sám například objednával na pravidelný servis přes GSM síť? Nebo že by lednice v domácnosti (jinak spolu s mnoha domácími elektrospotřebiči běžná aplikace řídicích vestavěných systémů) komunikovala s dodavatelem potravin přes internet a automaticky objednávala potraviny, které chybějí? Na podobné otázky asi neexistuje univerzální odpověď, jedno je ovšem jisté, vzdálená komunikace je u vestavěných zařízení čím dál častější a právě jí se budeme v následujících kapitolách věnovat.

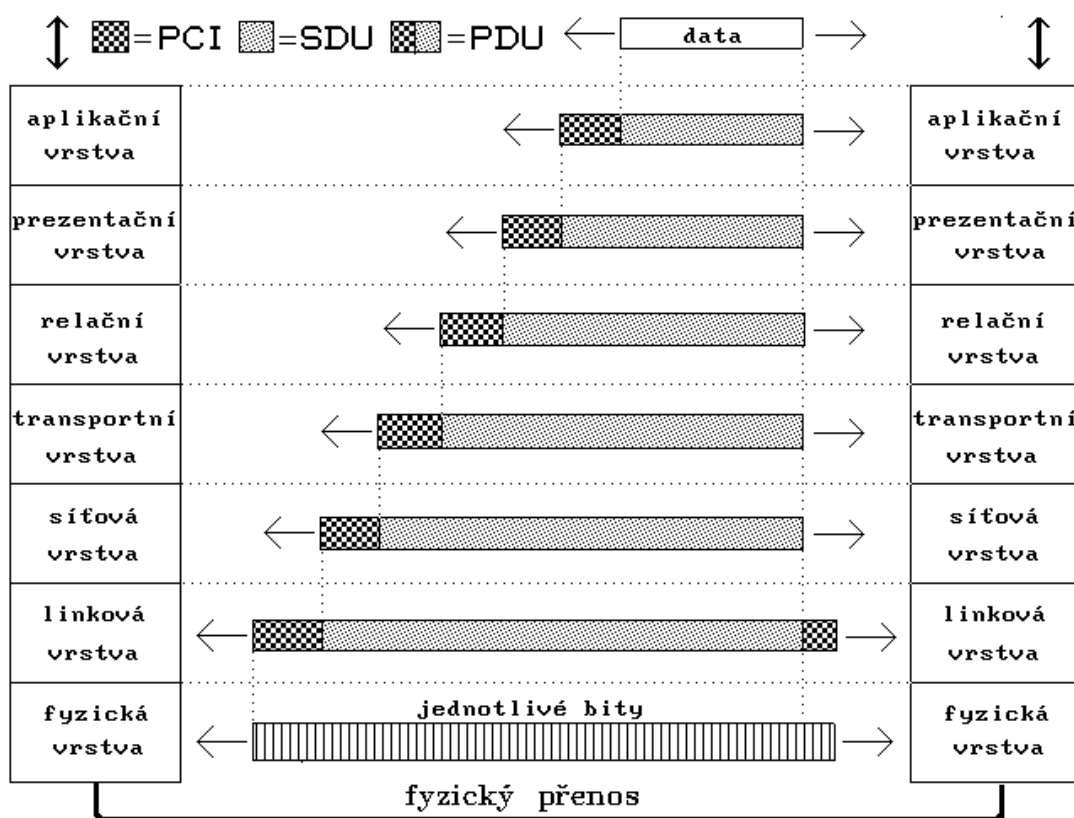
3.2 ISO/OSI referenční komunikační model

Abychom se dokázali bavit obecně o komunikaci, tak je vhodné nejprve vyložit referenční komunikační model, který slouží jako základ pro tvorbu komunikačních protokolů a poslouží také k jejich porovnání. Díky němu se také obecně lépe vysvětlují komunikační technologie jako takové.

ISO/OSI model je organizován do sedmi vrstev [14], u kterých specifikuje jejich funkce a služby, nikoliv implementaci – ta je na programátorech. Musí být dodrženo, že vrstvy na jednom systému mezi sebou komunikují pomocí jasně definovaných rozhraní a to vždy jen nejbližší sousedící. Komunikace mezi stejnými vrstvami různých systémů se zase

řídí příslušnými síťovými protokoly. Jednotlivé vrstvy potom pro zabezpečení své činnosti využívají služeb své sousední nižší vrstvy. Svě vlastní služby pak poskytují sousední vyšší vrstvě. Podle normy není dovoleno komunikovat vzdálenějším vrstvám přímo. Následuje obrázek [15] znázorňující umístění jednotlivých vrstev v modelu.

Referenční ISO/OSI model



Obrázek [2] - Znárodnění komunikace v ISO/OSI modelu

Vysvětlivka:

- SDU - Service Data Unit – užitečná data
- PCI - Protocol Control Information – řídicí informace, hlavičky protokolů
- PDU - Protocol Data Unit - protokolární datová jednotka – užitečná data + hlavička

Na počátku komunikace vznikne požadavek na přenos dat v nějaké aplikaci, tento požadavek (SDU) je v aplikační vrstvě ISO/OSI modelu obalen hlavičkou (PCI) daného aplikačního protokolu a takto nově vzniklá datová jednotka (PDU) je předána nižší vrstvě

ke zpracování. Ta opět přidá svoji hlavičku a celý proces se opakuje. Až je na nejnižší vrstvě (fyzická) proveden vlastní přenos bitů. Předávání datových jednotek nižším vrstvám a přidávání řídicích informací dané vrstvy se nazývá zapouzdřování. Na druhé komunikační straně je tento proces obrácen.

Následuje tabulka s přehledem jednotlivých vrstev ISO/OSI modelu.

Vrstva	Datová jednotka	Funkce
Aplikační	Data	Poskytuje přístup aplikacím ke komunikačnímu systému
Prezentační	Data	Transformace dat do formátu, který používají konkrétní aplikace – šifrování, komprimace, konverze
Relační	Data	Zprostředkuje relaci mezi aplikacemi
Transportní	Segmenty	Zajišťuje přenos dat mezi koncovými uzly (spojovaně i nespojovaně)
Síťová	Pakety/Datagramy	Směrování v sítích a síťová adresace
Linková	Rámce	Poskytuje spojení mezi dvěma sousedními systémy, fyzická adresace
Fyzická	Bity	Specifikuje fyzickou komunikaci. Aktivuje, udržuje a deaktivuje fyzické spoje

Tabulka [7] - Přehled vrstev ISO/OSI síťového modelu

3.3 Přehled komunikačních technologií

Komunikaci si pro účely tohoto výkladu omezíme na komunikaci vzdálenou. Vzdálenou komunikací budeme chápat takovou komunikaci, která není pevně omezena na určitou malou lokalitu a lze ji i díky propojování do rozsáhlejších sítích využít ve větším, klidně i globálním měřítku. Navíc se ještě omezíme na komunikační technologie běžně využívané i v jiných oblastech než jsou vestavěné systémy, např. telekomunikace a internet. A to proto, že se chceme zaměřit jen na ty případy vzdálené komunikace s vestavěnými zařízeními, které lze provádět pomocí např. mobilního telefonu, nebo PC připojeného do počítačové sítě.

3.3.1 GSM

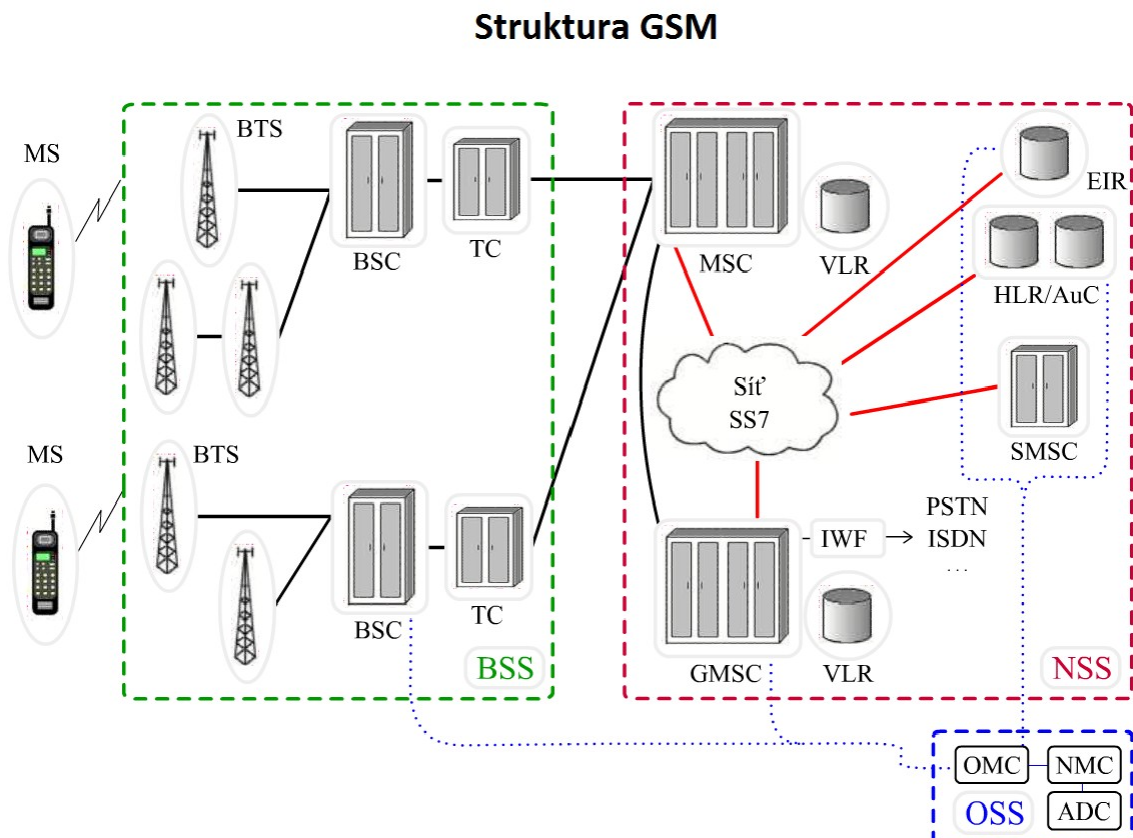
Zkratka GSM [16] znamená **G**lobální **S**ystém pro **M**obilní komunikaci. V současnosti se jedná o celosvětovou síť pro mobilní hlasové i datové komunikace. V jejích počátcích, které sahají do roku 1982 se zkratkou GSM myslel název pracovní skupiny, která měla na starost specifikaci standardu – francouzsky „Groupe Spécial Mobile“. V této době již v několika Evropských zemích existovaly mobilní telekomunikační sítě, které ovšem vzájemně nebyly kompatibilní. Znamenalo to, že komunikační zařízení z jedné země nebylo v jiné zemi použitelné. Tato skutečnost velmi brzdila jakýkoliv rozvoj, a proto bylo rozhodnuto, že bude vytvořen nový celoevropský standard mobilní telekomunikace, který výše zmíněné nedostatky odstraní. První verze standardu byla dostupná v roce 1990 a o rok později začala fungovat první komerční síť na tomto standardu postavená. Významným rokem pro GSM byl rok 1997, kdy byl standard rozšířen o datovou komunikaci GPRS. O dva roky později přibylo vylepšení ve formě definování rychlejších datových komunikací EDGE a UMTS.

Síť GSM je sítí buňkového typu, kde jsou digitálně kódovány jak signalizační kanály, tak i hovorové. Buňková, neboli cellular topologie sítě, udává způsob, jakým se mobilní telekomunikační zařízení (nejčastěji mobilní telefon) připojuje do sítě a komunikuje v ní. Síť je rozdělena na mnoho buněk – lokalitou omezené části. Buňky bývají až desítky km veliké, které mají malou hustotu – hlavně v málo obydlených oblastech až po buňky malé jen několik desítek metrů s vysokou hustotou, které se používají uvnitř budov. Každou buňku obhospodařuje jedna základová stanice BTS – Base Transceiver Station. BTS se vzájemně liší výkonem anténního subsystému (dle velikosti buňky) a několik BTS je ovládáno základnovou řídicí jednotkou BSC – Base Station Controller. BSC se napojuje do mobilní radiotelefonní ústředny MSC – Mobile Switching Centre, která přepíná hovory mezi jednotlivými BSC, případně je předává do jiných sítí.

Součástí GSM sítě je i několik druhů registrů mobilních zařízení, přes které se mobilní zařízení autentizují do sítě. V tomto procesu je důležitý Subscriber Identity Module – známý jako SIM karta. Ten je vložen do mobilního zařízení a obsahuje informace sloužící k identifikaci účastníka v síti. Tato identifikace je uložena v podobě IMSI – International Mobile Subscriber Identity, což je na celém světě unikátní číslo.

Existuje-li tedy požadavek na připojení vestavěného zařízení do sítě GSM, musí toto vestavěné zařízení obsahovat také GSM modul, v něm vloženou kartu SIM a nějaký anténní subsystém.

Propojení všech prvků do GSM sítě je vidět z následujícího obrázku [17], poté následuje výčet prvků GSM sítě.



Obrázek [3] - Znárodnění struktury GSM sítě

GSM subsystémy:

- BSS - Base Station Subsystem - Subsystém základnových stanic
- NSS - Network Switching Subsystem - Síťový spojovací subsystém
- OSS - Operational and Support Subsystem - Operační a podpurný subsystém

BSS:

- BTS - Base Transceiver Station - Základnová stanice
- BSC - Base Station Controller - Základnová řídicí jednotka
- TC - TransCoder - Transkodér

NSS:

- MSC - Mobile Switching Centre - Mobilní radiotelefonní ústředna
- HLR - Home Location Register - Domovský lokační registr
- VLR - Visitor Location Register - Návštěvnický lokační registr
- AuC - Authentication Centre - Centrum autentičnosti
- EIR - Equipment Identity Register - Identifikační registr mobilních stanic
- IWF - Inter-Working Functionality - Jednotka spolupráce s externími sítěmi
- SMS Centrum

OSS:

- OMC - Operational and Maintenance Centre - Provozní a servisní centrum
- NMC - Network Management Centre - Centrum pro řízení sítě
- ADC - Administrative Centre - Administrativní centrum

Další prvky:

- MS - Mobilní zařízení - Mobile Equipment
- Síť SS7 – Signalizační síť
- ISDN, PSTN – veřejné telefonní síť

3.3.2 Ethernet IEEE 802.3

Jde o vedoucí technologii pro tvorbu drátových LAN sítí, ve které se snoubí velká jednoduchost protokolu, snadnost údržby i montáže (díky tomu nízká cena) s dobrými přenosovými rychlostmi. Mohlo by se zdát, že do výčtu komunikačních technologií pro vzdálenější přenosy úplně nepatří, když rozsah jednotlivých segmentů těchto sítí může být dle konkrétní normy jen řádově stovky metrů, jenže skutečnost je taková, že se tyto sítě běžně spojují do větších a po připojení k internetu lze hovořit o globálně dosažitelné síti. Díky jejímu opravdu masovému rozšíření je to možná první technologie, která nás většinou při nutnosti síťové komunikaci napadne [3].

V rámci modelu ISO/OSI ethernet představuje vrstvu fyzickou (dráty a signály) a linkovou (výměna datových paketů). Původní Ethernet využíval sběrníkovou topologii – sdílené médium, kde všichni účastníci komunikace slyší všechno a v každém okamžiku může vysílat jen jeden z nich. Jednotlivé stanice jsou identifikovány svými hardwarovými adresami (MAC adresa – směrování na linkové vrstvě). Když stanice obdrží paket s jinou než vlastní adresou, zahodí jej.

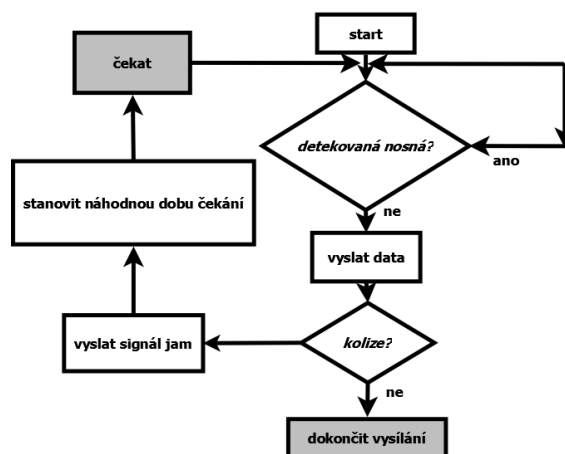
Pro přístup ke sdílenému přenosovému médiu (sběrnici) se používá metoda CSMA/CD - Carrier Sense with Multiple Access and Collision Detection, česky metoda mnohonásobného přístupu s nasloucháním nosné a detekcí kolizí.

Stanice, která potřebuje vysílat, naslouchá prostřednictvím síťové karty co se děje na přenosovém médiu. Pokud je na sběrnici klid, začne stanice vysílat. Může se stát (v důsledku zpoždění signálu), že dvě stanice začnou vysílat přibližně ve stejný okamžik. Jejich signály se pak navzájem nekontrolovatelně smísí. Tato situace se nazývá kolize a vysílající stanice ji poznají podle toho, že během svého vysílání zároveň zjistí příchod cizího signálu. Stanice, která detekuje kolizi, vyšle krátký signál (jam o 32 bitech). Poté se všechny vysílající stanice odmlčí a později se pokusí o nové vysílání.

Mezi opakovanými pokusy o vysílání stanice počká vždy náhodnou dobu. Interval, ze kterého se čekací doba náhodně vybírá, se během prvních deseti pokusů vždy zdvojnásobuje. Stanice tak při opakovaných neúspěších prodlužuje dobu mezi svými pokusy o vysílání a zvyšuje tak pravděpodobnost, že se o sdílené médium úspěšně podělí s ostatními stanicemi. Pokud se během šestnácti pokusů nepodaří rámec odvysílat, stanice své snažení ukončí a ohlásí nadřazené vrstvě neúspěch.

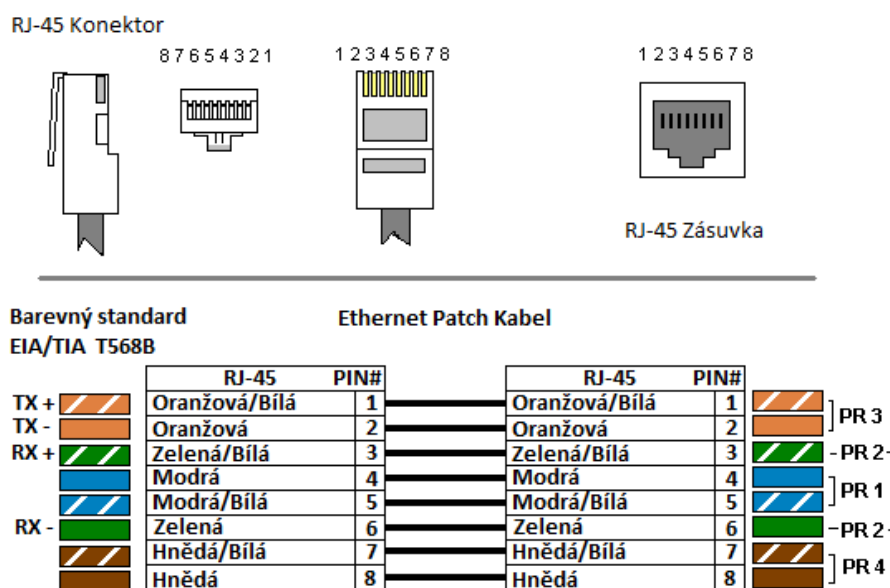
Ke kolizi může dojít jen v době, která uplyne od začátku vysílání do okamžiku, kdy signál vysílaný stanicí obsadí celé médium (pak již případní další zájemci o vysílání zjistí, že médium není volné a počkají na jeho uvolnění). Tento interval se nazývá kolizní okénko a musí být kratší, než je doba vysílání nejkratšího rámce. Jinak by mohlo docházet k nezjištěným kolizím (dvě vzdálené stanice odvysílají krátké rámce, které se na kabelu protnou a zkomolí, ale obě stanice ukončí vysílání dříve, než k nim dorazí kolidující signál). Princip CSMA/CD je dobře vidět i na následujícím obrázku [3].

Princip CSMA/CD



Obrázek [4] - Princip CSMA/CD

Sběrníková topologie se využívala v době využívání verzí ethernetu s označením 10Base5 a 10Base2, jako vodič se používal koaxiální kabel a byly dosahováno přenosové rychlosti 10 Mbit/s. Dnes jsou nejvyužívanějšími drátovými verzemi technologie ethernet tzv. Fast Ethernet ve specifikaci 100Base-TX s přenosovými rychlostmi 100 Mbit/s a tzv. Gigabit Ethernet ve specifikaci 1000Base-T s přenosovými rychlostmi 1 Gbit/s. Pro výstavbu těchto sítí je nejpoužívanějším druhem kabeláže kroucená dvojlinka - UTP - unshielded twisted-pair. Jde o osmi žilové kabely složené ze čtyř párů vodičů, které se v rámci jednoho páru kroučí do sebe. To má za důsledek lepší odolnost proti elektromagnetickému rušení, protože se při přenosu signálu vyhodnocuje rozdíl mezi dvěma vodiči v páru a ten by měl zůstat stejný i při rušení. Rušení totiž ovlivňuje oba do sebe zakroucené vodiče stejně. Jako ukončující prvek kabelů se používá konektor RJ-45, přičemž ve 100 Mbit verzi se reálně využívá 4 vodičů a ve verzi 1 Gbit se využívá všech osmi vodičů. Existuje ještě stíněná verze kabelu - STP - shielded twisted pair, která je ještě odolnější proti elektromagnetickému rušení. Následuje obrázek s nákresem UTP kabelu.

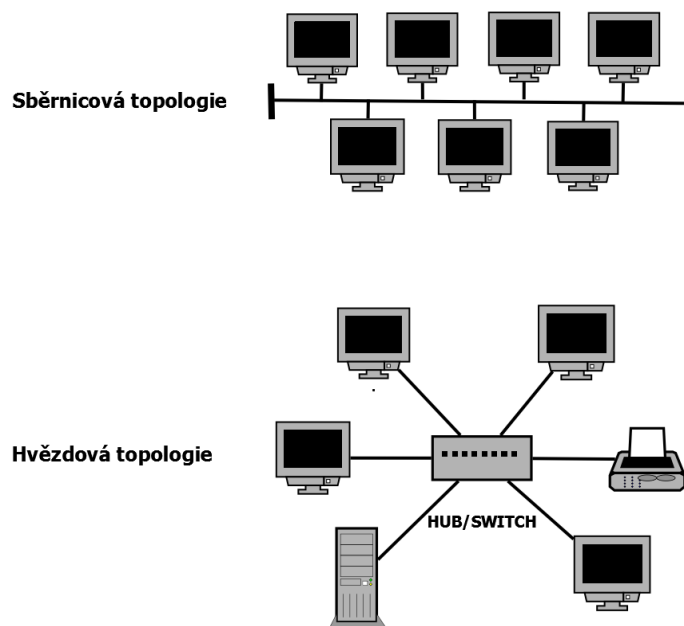


Obrázek [5] - Uspořádání vodičů UTP kabelu a RJ-45 konektor

Nasazením verzí s vyššími přenosovými rychlostmi a využitím kroucené dvojlinky se změnila topologie ze sběrnice na hvězdicovou. Ústředním prvkem se stal rozbočovač – hub. Na koncích jednotlivých spojů jsou připojené počítače. Chování sítě napodobuje sběrnici – rozbočovač kopíruje signál přicházející z jednoho rozhraní do všech ostatních. Data vysílaná jednou stanicí jsou proto rozšířena všem ostatním, stejně jako v případě jejich přenosu po sdílené sběrnici. Z důvodu dalšího snižování možnosti zahlcení sítě se rozbočovač uprostřed hvězdicové topologie nahradil za přepínač – switch. Přepínač pracuje na rozdíl od rozbočovače o jednu vrstvu v ISO/OSI modelu výše, tj. na linkové. Při přijetí ethernetového rámce ho totiž nejprve analyzuje a z MAC (fyzické) adresy jeho příjemce určí na které rozhraní má být odvysílán. Přepínač si totiž automaticky vytváří a udržuje tabulku MAC adres s vazbou na fyzické rozhraní na které se daný adresát nachází.

Následuje obrázek s nákresem síťových topologií a tabulka s přehledem nejzajímavějších verzí ethernetu.

Síťové topologie



Obrázek [6] - Síťové topologie

Standard	Rok uvedení	Popis
Experimental Ethernet	1973	2.94 Mbit/s (367 kB/s) přes koaxiální kabel
Ethernet II (DIX v2.0)	1982	10 Mbit/s (1.25 MB/s) přes tzv. „tlustý koaxiální kabel“. Tento typ je využíván ve všech formách Ethernetových protokolů TCP/IP.
IEEE 802.3	1983	10BASE5 10 Mbit/s (1.25 MB/s) přes tzv. „tlustý koaxiální kabel“.
802.3a	1985	10BASE2 10 Mbit/s (1.25 MB/s) přes tzv. „tenký koaxiální kabel“ (neboli 10Base2)
802.3i	1990	10BASE-T 10 Mbit/s (1.25 MB/s) přes kroucenou dvoulinku.
802.3j	1993	10BASE-F 10 Mbit/s (1.25 MB/s) přes optické vlákno.
802.3u	1995	100BASE-TX, 100BASE-T4, 100BASE-FX Fast Ethernet v 100 Mbit/s (12.5 MB/s)

802.3x	1997	Plně duplexní a je zde řízení toku dat.
802.3y	1998	100BASE-T2 100 Mbit/s (12.5 MB/s) přes méně kvalitní kroucenou dvoulinku.
802.3z	1998	1000BASE-X Gbit/s Ethernet přes optické vlákno v 1 Gbit/s (125 MB/s)
802.3ab	1999	1000BASE-T Gbit/s Ethernet přes kroucenou dvoulinku v 1 Gbit/s (125 MB/s)
802.3ae	2003	10 Gbit/s (1,250 MB/s) Ethernet přes vlákno; 10GBASE-SR, 10GBASE-LR, 10GBASE-ER, 10GBASE-SW, 10GBASE-LW, 10GBASE-EW
802.3af	2003	Power over Ethernet (12.95 W)
802.3an	2006	10GBASE-T 10 Gbit/s (1,250 MB/s) Ethernet přes nestíněné UTP.
802.3at	2009	Vylepšení Power over Ethernet (25.5 W)
802.3ba	2010	40 Gbit/s and 100 Gbit/s Ethernet. 40 Gbit/s na jeden metr sběrnice.

Tabulka [8] - Přehled verzí ethernetu

Ve vztahu k připojování vestavěných zařízení ke komunikačním sítím se při použití technologie ethernet dá těžit jednak z velkého rozšíření této technologie, potom také z přímého integrování řadičů ethernetu v mnoha mikrokontrolérech a také z toho, že díky technologii POE – Power Over Ethernet se dá jednoduše vyřešit napájení vestavěných zařízení pomocí jediného UTP kabelu, který je potom společný pro data i napájení.

II. PRAKTICKÁ ČÁST

4 NÁVRH SYSTÉMU

V této kapitole přednáším požadavky na řešení webového monitorovacího a řídicího systému, dále navrhuji strukturu tohoto systému, jehož vlastní implementací se poté zabývám v dalších kapitolách.

4.1 Základní požadavky na systém

Cílem tohoto návrhu je vytvořit předlohu vestavěného zařízení, které bude schopno podle připojeného spektra detektorů monitorovat různé technologické procesy a tyto procesy bude schopno řídit. To vše v obecné rovině, v případě reálného nasazení musí být možné přidat do systému standardizovanou cestou moduly implementující spolupráci s konkrétními detektory a moduly realizující různé řídicí akce.

Dalším zásadním požadavkem na vyvíjené vestavěné zařízení je možnost komunikovat s ním vzdáleně pomocí nějakých standardizovaných nástrojů a protokolů. Zařízení musí být schopno autonomní funkce s možností vzdáleného nastavování, sledování a ovládání systému.

4.2 Specifikace systému

Vestavěné zařízení bude připojeno do komunikační sítě ethernet, bude na něm běžet webový server, který bude naslouchat požadavkům uživatele, zadaných pomocí webového prohlížeče. S vestavěným zařízením bude pracováno právě jen pomocí webového prohlížeče. Webový server uživateli zprostředkuje práci s monitorovanými, či řízenými periferiemi.

4.3 Návrh řešení systému

Hardwarová část vestavěného zařízení bude založena na mikrokontroléru ARM Cortex-M3, který bude pro snazší implementaci integrován na nějakém vývojovém kitu. Mikrokontrolér musí mít integrovaný ethernetový řadič, jehož piny budou přímo vyvedeny na vývojovém kitu. Ke kitu potažmo mikrokontroléru budou dále připojeny vhodné testovací periferie.

Celá softwarová část vestavěného zařízení bude integrovaná ve firmwaru, který se pak nahraje do hardware, jež tímto bude ovládat. Software bude naprogramován jako

vysoce modulární s částmi oddělenými jasným rozhraním, tak aby bylo možné relativně jednoduchým způsobem obměňovat či doplňovat systémové součásti a upravovat tak jeho celkovou funkčnost. Měly by zde být následující moduly:

- hlavní řídicí modul – bude ovládat všechny ostatní části
- moduly pro práci se senzory – implementace konkrétních druhů senzorů
- moduly pro práci s výstupními zařízeními – implementace řízení tech. procesů
- modul uživatelského rozhraní – implementace webového serveru
- modul síťové komunikace – implementace TCP/IP stacku
- moduly ovladačů hardware – implementace ovladače ethernetového řadiče a různých vstupně/výstupních zařízení

5 VÝVOJOVÝ KIT

V této kapitole se zabývám výběrem konkrétního vývojového kitu, který bude sloužit jako hardwarová část v mém projektu. Uvádím zde popis několika různých vývojových kitů, mezi kterými jsem si vybíral. Vysvětluji, proč jsem se nakonec rozhodl pro vývojový kit LPCXpresso. Dále popisuji jeho nejzajímavější vlastnosti a uvádím, jakým způsobem jsem s tímto kitem reálně pracoval.

5.1 Úvod do výběru vývojového kitu

Než se začneme zabývat popisem jednotlivých vývojových kitů a výběrem toho nejhodnějšího, tak je jistě užitečné říci co budeme od takového kitu očekávat a co je důvodem, že daný projekt neřešíme jinými způsoby, nýbrž právě pomocí kitu.

5.1.1 Vývojový kit obecně

Vývojovým kitem budeme rozumět hardwarové zařízení určené k rychlému a efektivnímu prototypování vestavěných zařízení. Jde o to, abychom mohli vyvíjet naši aplikaci bez toho, že se budeme muset hned od začátku věnovat návrhu desky plošných spojů, jejím osazováním, pájením a nakonec oživováním. Došel by-li vývoj našeho vestavěného zařízení až do produkčního stavu, tak se výše uvedeným činnostem samozřejmě nevyhneme. Do té doby však máme podstatně zjednodušenou práci, když si můžeme všechno hned reálně vyzkoušet. Nemít vývojový kit, tak by nás čekala spousta teoretické a simulační práce a vývoj by se stal mnohem pomalejším.

Amatérskou alternativou k vývojovým kitům by ještě mohl být vývoj v nepájivém poli, to lze ovšem dělat jen do relativně malé složitosti vyvíjené aplikace – hledat drobnou chybu ve změní desítek až stovek „drátků“ může být opravdu velmi frustrující. Navíc mikrokontroléry, kterými se budeme dále zabývat, stejně nebývá možné do nepájivých polí vůbec zapojit – nenašel jsem jediný mikrokontrolér postavený na platformě ARM Cortex-M3, který by navíc obsahoval integrovaný řadič ethernetu a byl určený do DIP nebo DIL pouzdra.

Abychom upřesnili definici z předchozího odstavce v tom, jak budeme vývojový kit chápat, tak můžeme říct že, vývojovým kitem budeme dále rozumět hardwarové zařízení, které bude složeno z mikrokontroléru, který chceme následně reálně nasadit, a budou

na něm vyvedeny všechny pro nás užitečné periferie. Dále by vývojový kit měl mít vyřešeno napájení, přívod hodinového signálu a mělo by být umožněno real-time debugování vyvíjené aplikace. Zjednodušeně se dá říci, že by na vývojovém kitu mělo být okamžitě možné začít vytvářet a v reálu ladit naši aplikaci.

5.2 Přehled vhodných vývojových kitů

V následujících podkapitolách si představíme tři konkrétní vývojové kity, které se díky svým vlastnostem dostaly do užšího výběru a byly pečlivěji posuzované. Ve všech třech případech se jedná o kity obsahující mikrokontrolér založený na platformě ARM Cortex-M3. Všechny tři mikrokontroléry mají také přímo integrován 10/100 Mbit ethernetový řadič, jehož vývody jsou přímo vyvedeny na vývojovém kitu pro snadné využití.

5.2.1 Mbed

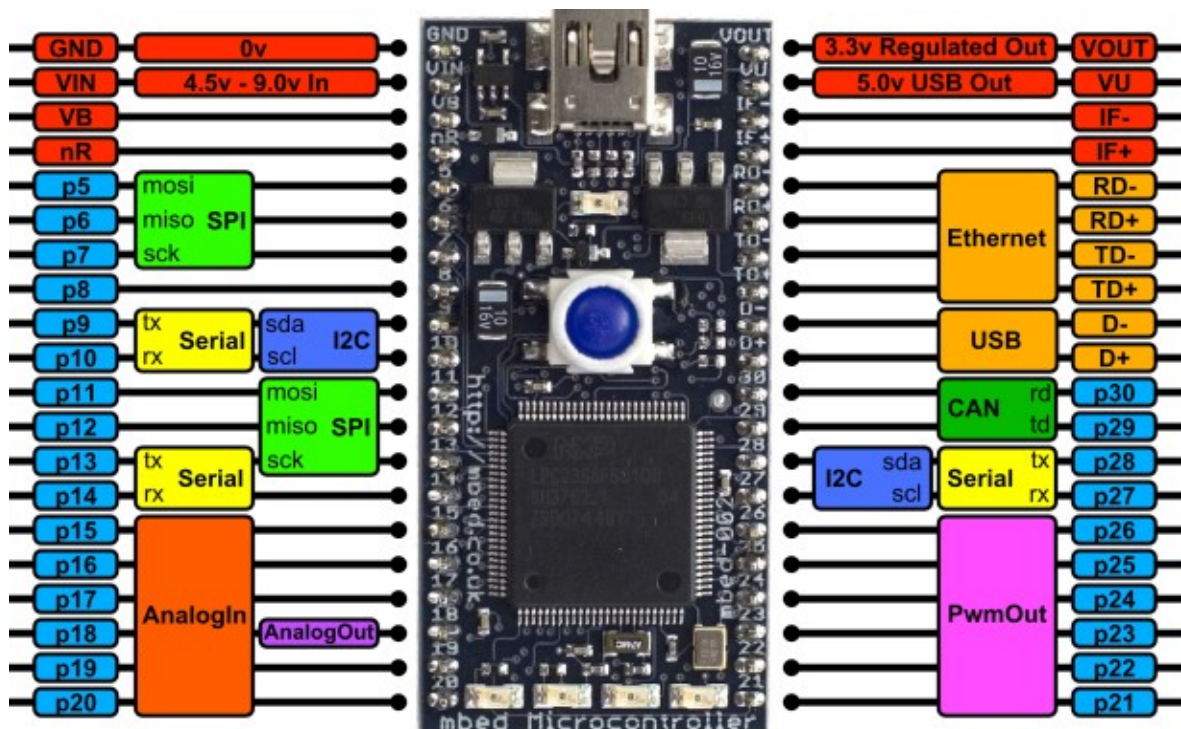
Mbed je souhrnné označení pro vývojový kit a sadu nástrojů pro vývoj na něm. Tento projekt vznikl v roce 2005 přímo ve společnosti ARM jako pokus umožnit hlavně studentům jednoduše prototypovat na mikrokontrolérech této firmy. V dnešní době má za sebou velkou komunitu uživatelů se spoustou knihoven a návodů. Opravdovou specialitou této platformy je skutečnost, že veškeré nástroje, tj. v první řadě integrované vývojové prostředí s compilerem, debuggerem apod. je čistě online charakteru. Práce s mbed probíhá tak, že přes klasický webový prohlížeč přihlásíme na domovské stránkách aplikace, v níž je pomocí běžných webových technologií (jako je hlavně javascript a AJAX) implementováno kompletní vývojové prostředí. Jde o platformu jednoduše využitelnou jak na windows, tak na mac i linuxu. Vývojový kit se k PC, ze kterého se připojujeme na web mbed připojí USB kabelem. Je také zajímavé, že celý kit jako takový je koncipován jako DIP pouzdro, pokud tedy potřebujeme využít nějaké jeho periferie, tak ho můžeme vložit do nepájivého pole. Ale už sám o sobě je plně použitelný pro základní vývoj. Jelikož se celá práce s mbed děje online na webu, tak je zde naprostá závislost na kvalitní připojení k internetu – offline se s mbed nedá nijak pracovat [18].

Parametry mbed vycházejí z mikrokontroléru, na kterém je založen a tím je v aktuální verzi LPC1768 od fy. NXP[22] (původně Phillips). Následuje přehled nejzajímavějších parametrů:

- Vysoce výkonné jádro ARM® Cortex™-M3

- 96MHz, 32KB RAM, 512KB FLASH
- Ethernet, USB Host/Device, 2xSPI, 2xI2C, 3xUART, CAN, 6xPWM, 6xADC, GPIO
- prototypovací deska – 40 pinové DIP pouzdro, 54x26mm
- napájení přes 5V USB nebo 4.5-9V externí zdroj
- Vestavěný drag 'n' drop FLASH programátor
- Odlehčené online vývojové prostředí a kompilér
- Vysoko úroňové C/C++ SDK
- Mnoho publikovaných knihoven a projektů

Vývojový kit se dá pořídit za cca 59\$, následuje zapojení vývojového kitu mbed.



Obrázek [7] - Vývojový kit Mbed

5.2.2 Luminary Micro LM3S696 Ethernet Evaluation Kit

Jde o vývojový kit, který je aktuálně vyráběn firmou Texas Instruments, je založený na mikrokontroléru LM3S696 z řady Stellaris ARM® Cortex™-M3. Byl to první vývojový kit, který jsem objevil při hledání hardware pro svůj projekt. Zaujal mě hlavně

integrovaným hardwarovým debuggerem (komponenta, která se dá pořídit pro většinu mikrokontrolérů, které umějí komunikovat přes JTAG rozhraní, jenže jde většinou o relativně drahé zařízení) a LCD displayem s ovládacími tlačítky. LCD v tomto projektu přímo nepotřebuji, ale na druhou stranu je možná lepší mít více vybavený vývojový kit, protože se bude dávat obecně použít pro vývoj většího spektra aplikací. Problém u tohoto vývojového kitu je se softwarem, protože i když je k tomuto kitu výběr z několika možných integrovaných vývojových prostředí (s patřičným kompilerm a knihovnamy), tak ani jedno z nich není volně využitelné – všechna jsou omezena a to buď na určitý čas, kdy lze bezplatně vývojové prostředí provozovat, nebo většina funkcí funguje zdarma, ale nejde používat hardwarový debugger až po verzi, která jde využívat relativně dobře, až na to že je omezena na 32KB maximální velikost firmwaru, který lze nahrát do mikrokontroléru. Takže kromě ceny kit, která je cca 69\$ by pro serióznější vývoj bylo potřeba investovat ještě minimálně několik desítek dolarů navíc [19]. Následuje popis parametrů:

- 32-bit ARM® Cortex™-M3 50-MHz processor, 256KB FLASH, 64KB RAM
- 10/100 Mbit ethernet, SSI/SPI, 2xI2C, 3xUART. 6XPWM, 4xADC
- USB kabel poskytuje sériovou komunikaci, debugování a napájení
- OLED display s rozlišením 128 x 64 pixelů a 16-ti odstíny šedé
- navigační přepínače a tlačítka, magnetický speaker, slot na MicroSD kartu
- standardní ARM® 20-pin JTAG debugovací konektor s vstupně/výstupním módem



Obrázek [8] - Vývojový kit Luminary Micro LM3S6965

5.2.3 LPCXpresso 1769

Vývojová platforma LPCXpresso se skládá z integrovaného vývojového prostředí LPCXpresso IDE a několika typů prototypovacích desek. Prototypovací deska je složena z hardwarového JTAG debuggeru (LPC-LINK) a z desky s cílovým mikrokontrolérem. Dále se budeme bavit jen desce, která obsahuje mikrokontrolér řady ARM Cortex-M3, konkrétně jde o NXP LPC1769 s integrovaným řadičem ethernetu [20], [22]. Hlavní vlastnosti NXP LPC1769:

- Jádru ARM® Cortex™-M3
- 120MHz, 64KB RAM, 512KB FLASH
- Ethernet, USB Host/Device, 2xSPI, 2xSSP, 3xI2C, 1xI2S, 3xUART, 2xCAN, 6xPWM, 6xADC, GPIO

Vývojový kit se připojuje k PC pomocí USB kabelu, který je zapojen do části s hardwarovým JTAG debuggerem. S cílovou deskou tak nekomunikujeme přímo, nýbrž právě přes debugger. Cílová deska je od výroby spojena s debuggerem velmi lehce přerušitelnými spoji, po jejichž přerušení můžeme JTAG debugger využít k ladění jakéhokoliv LPC mikrokontroléru. K opětovnému spojení debuggeru a prototypovací desky

potom již musíme použít propojení dráty do předem připravených pinů. Jde opravdu o velmi elegantní a do detailu domyšlený návrh.

LPCXpresso lze pořídit za cenu okolo 30\$ a když si uvědomíme, že za tuto cenu vlastně dostáváme vývojový kit s velmi výkonným mikrokontrolérem + hardwarový JTAG debugger a s velmi mocným vývojovým prostředím (podrobněji viz. dále), tak lze jen těžko hledat argumenty proti koupi. Jde zřejmě o nejlevnější a přitom velmi kvalitní ARM Cortex-M3 vývojovou platformu na trhu. Pro představu dále následuje obrázek desky LPCXpresso.



Obrázek [9] - Vývojový kit LCPXpresso LPC1769

5.3 Práce s LPCXpresso

Tvorba vestavěné aplikace pomocí LPCXpresso platformy má několik částí. Jako první krok je vhodné provést fyzické propojení komponent, se kterými chceme pracovat (jako různé senzory, tlačítka, periferie apod.) s prototypovací deskou. Nejenom v tomto kroku je vhodné důsledně prozkoumat relevantní informace v datasheetu [22] a zjistit, které piny mají jakou funkci. Funkce u většiny pinů se dá měnit mezi několika možnostmi, některé jsou přímo „zadrátovány“ a měnit je nelze. Např. při vývoji tohoto projektu byly použity piny propojené s ethernetovým řadičem a obecné vstupně/výstupní piny GPIO.

Dalším krokem je vytvoření firmware ve vývojovém prostředí, o tom je následující kapitola.

5.3.1 Úvod do vývojového prostředí LPCXpresso

Vývojové prostředí LPCXpresso je založeno na velice kvalitním a rozšířeném obecném opensource vývojovém prostředí Eclipse[20]. Jeho běžné funkce byly zachovány, pouze bylo doplněno mnoho LPC specifických pluginů. Kdo se již setkal s Eclipse, tak nebude

mít se zvládnutím práce v LPCXpresso IDE problém, pro ostatní je dostupno mnoho kvalitních tutorialů [21]. LPCXpresso IDE je určeno pro všechny PC platformy – můžeme ho použít jak na windows, tak i na mac a linuxu. Je volně stažitelné ze stránek výrobce [21].

Nejzásadnější komplikací při využívání LPCXpresso IDE, tak může být limit v maximální velikosti firmware, který lze pomocí vývojového prostředí nahrát do mikrokontroléru. V základní verzi jde o 32KB kódu, tato verze však může být zdarma vylepšena pouhým registrováním se u výrobce. To je ovšem umožněno pouze majitelům LPCXpresso vývojového kitu, jelikož spolu s kitem je distribuován registrační kód. Po jeho zadání bude zvětšen limit v nahrávání firmware do mikrokontroléru na 128KB kódu. To je podstatně víc než u bezplatných verzí jiných výrobců. Tato velikost je sice pouze čtvrtinou celkové kapacity flash paměti používaného mikrokontroléru, jenomže 128KB kódu je v oblasti vestavěných aplikací poměrně velká hodnota. Kdyby to přesto nestačilo, je samozřejmě možno dokoupit rozšíření, ale i bez něho jde o špičkový bezplatný software.

Vývoj firmware pro mikrokontrolér v LPCXpresso IDE začíná u vytvoření nového projektu. Zde je nejdůležitější vybrat správný typ mikrokontroléru, pro jaký budeme vyvíjet. Podle toho budou připojeny správné CMSIS knihovny a my tak budeme moci používat vysokou míru abstrakce bez programování v assembleru, místo toho budeme využívat obecné struktury v jazyce C, které odstíní mnoho hardwarových detailů a umožní nám soustředit se jen na klíčové věci. Máme na výběr z několika typů projektů, z nichž nejzajímavější jsou tyto:

- C project – běžný samostatný projekt, může být v řetězci projektů (bude vysvětleno dále) pouze jeden, vždy obsahuje soubor *main.c*, který implementuje funkci *main()*, jež zapouzdřuje celou vestavěnou aplikaci; dále je zde přítomen soubor *cr_startup_lpc[číslo typu mikrokontroléru].c*, kde jsou definovány vektory přerušení a jejich obsluha (přerušení jsou události v systému, které vznikají asynchronně k běhu hlavního programu, reaguje se tak na vnější události – např. tik časovače, alarm, dokončení vstupně/výstupní operace apod.)
- C semihosting project – podobné jako C project, jen s tím rozdílem, že jsou použité jiné implementace standardních C knihoven; tohoto se s úspěchem využívá při ladění programu, při běžném provozu to již není vhodné; jde o to, že když je projekt nastavený jako semihosting, tak je např. možno provádět ladící výstupy

do terminálu – běžná funkce *printf()*, když by semihosting nebyl nastaven (lze přepnout i u už vytvořeného projektu), tak by se ladící výpis provedl „někam do mikrokontroléru“ a programátor by z toho neviděl nic (pravděpodobnější ovšem je, že by ani korektně neproběhla kompilace programu)

- C library project – knihovní projekt, je připojen k hlavnímu projektu a definuje některé funkce, které hlavní projekt využívá; ze samostatné knihovny nelze udělat kompletní firmware

Jak bylo mimochodem naznačeno, tak projekty lze spojovat dohromady a lze tak vytvořit efektivní modulární strukturu. Máme-li správně vytvořen projekt, tak můžeme začít pracovat na vlastním firmware. Ten může mít následující strukturu. Např. CMSIS knihovna bude jeden projekt (C library project), modul implementující komunikaci s nějakým druhem senzoru bude další projekt (opět C library project) a naposled zde bude hlavní projekt (C [semihosting] project), který bude implementovat hlavní řídicí část firmware.

Máme-li naprogramovány zamýšlené funkce, tak můžeme přejít k překladu programu. Pokud překlad proběhne správně, tak by mělo dojít k vytvoření souboru s binárním obrazem firmware. Následně můžeme nahrát firmware do flash paměti mikrokontroléru a začít debuggovat.

5.3.2 Debuggování LPCXpresso

Debuggování je obecně činnost, která má za úkol odhalit co nejvíce chyb v programu a pomoci tak k jeho co nejlepšímu odladění. Přístupů k debuggování lze použít více, takovým základním je ladění pomocí výpisů. Výpisy jsou umísťovány na důležitá místa ve zdrojovém kódu programu, jejichž prostřednictvím se dá zjistit v jakém stavu program je, jakými větvemi vykonávání prošel apod. V případě LPCXpresso je nutno použít nastavení projektu „semihosting“ viz. výše. Další možností je ladění přímo na hardware. Snažíme se li například zjistit, jestli je některý z fyzických portů periodicky spínán, můžeme k tomuto portu připojit např. diodu, která nám dá blikáním najevo v jaké stavu se port nachází. Tímto způsobem sice ověříme, že vývojový kit, potažmo hardware skutečně něco fyzicky dělá, má to však mnohá omezení. Lze tak zejména testovat děje, které je člověk schopen sledovat. Pokud se hodnota na portu změní tisíckrát za vteřinu, je tento způsob ladění bezpředmětný.

Nejlepší způsob pro zjištění chování programu je spojit ladění pomocí výpisů a hardwarové ladění dohromady. A to prostřednictvím integrovaného debuggeru. V něm navíc můžeme v reálném čase zjišťovat hodnoty všech proměnných, ale také stav všech prvků v mikrokontroléru. Jsou zde vidět stavy registrů, periferních zařízení, paměti apod. Nejlepší věcí na ladění aplikace v interním debuggeru ovšem je možnost krokování. Lze tak příkaz za příkazem ve zdrojovém kódu (kromě C jazyka lze zároveň sledovat i assembler) projít celý program a kontrolovat jednak změnu vnitřních registrů a proměnných, dále to, jak se mění stav jednotlivých periférií – jak ve strukturách (potažmo v registrech), tak i na živo v hardware.

Když budeme sledovat například změnu stavu portu GPIO2.18, tak víme že sledujeme změnu ve struktuře GPIO (obecné vstupně/výstupní porty), druhý port a 18 bit (v LPCXpresso je toto opravdu názorné – po seznámení se se systémem se v tomto programátor nebude ztrácet) a při krokování programu uvidíme, že po provedení daného příkazu či funkce se hodnota sledovaného pinu změnila např. z 0 na 1 – změny jsou dokonce viditelně zvýrazňovány. Navíc tu změnu uvidíme i na živo na hardware (samozřejmě za předpokladu, že na daném pinu máme připojen nějaký prvek, ze kterého to je poznat, např. LED), protože mezi jednotlivými kroky je hardware fyzicky pozastaven a my tak máme kontrolu nad prováděním až v úrovni jednotlivých instrukcí v něm. Hardwarový debugger je opravdu nedocenitelný pomocník pro tvůrce vestavěného systému.

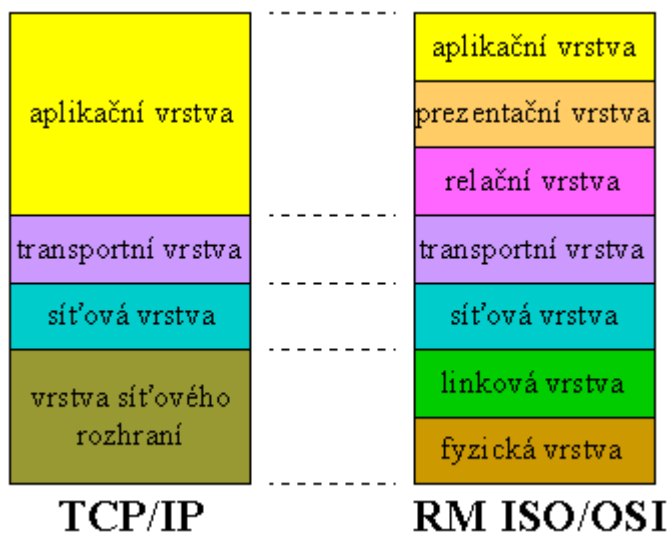
Hardwarové debugování má samozřejmě i omezení. Největším problémem je sledování více navazujících dějů, které následují po sobě a trvají pouze malé časové úseky. Zde hardwarový debugger není tolik platný. Lze ním sice zachytit některé události, jenomže pouze odděleně. Při krokování budou prodlevy mezi kroky minimálně v řádech jednotek vteřin a pokud chceme krokovat děj, který má trvání v jednotkách milisekund, tak to prostě nestihneme a časová návaznost provázaných dějů nebude dosažena. Tady je asi jedinou možností využití digitálního osciloskopu připojeného na daný port, který bude při dostatečné citlivosti sledovat děj v reálném čase. Poté ho bude možno zpětně zobrazit a zanalyzovat.

6 TCP/IP STACK

V této části práce popisují způsob jakým obecně chápat implementaci TCP/IP stacku. Uvádím, jakým způsobem jsem vybíral konkrétní implementaci TCP/IP stacku a zmiňuji zde nejzajímavější vlastnosti mnou testovaných implementací. Následně se věnuji praktickému popisu práce s finálně nasazeným TCP/IP stackem ve vztahu ke svému projektu.

6.1 Úvod

Rodina protokolů TCP/IP – Transmission Control Protocol/Internet Protocol – je hlavní sadou protokolů celosvětové sítě Internet. Tato architektura je principiálně velmi podobná architektuře použité u OSI/ISO síťového modelu, který ovšem zjednodušuje. V případě TCP/IP se také jedná o vrstevný komunikační model, který je ale na rozdíl od sedmi vrstevného ISO/OSI pouze čtyř vrstevový. Vůči ISO/OSI došlo ke sloučení některých vrstev do sebe, přičemž takto vzniklá vrstva převzala funkci těch původních. Z následujícího obrázku [15] je vidět, k jakým úpravám došlo.



Obrázek [10] - Rozdíl mezi TCP/IP a ISO/OSI

6.2 Výběr implementace TCP/IP

Implementace TCP/IP by měla být obecně modulární, kdy jeden modul, či sada funkcí zapouzdřuje danou vrstvu TCP/IP síťové architektury. Následující tabulka ukazuje, jaké protokoly (moduly) musejí být minimálně implementovány pro bezproblémovou funkčnost síťové komunikace v našem vestavěném zařízení.

Vrstva	Protokoly a forma implementace
Aplikační	HTTP – webový server
Transportní	TCP - modul
Síťová	IP, ICMP - moduly
Síťové rozhraní	ARP, Ethernet – modul + ovladač ethernetového řadiče

Tabulka [9] - Minimální implementace TCP/IP

V další části následuje krátký popis tří implementací TCP/IP, které byly testovány a uvažovány pro nasazení v tomto projektu.

6.2.1 Testovací TCP/IP z CMSIS balíku

Spolu s modulem CMSIS knihovny pro LPCXpresso [21] je také distribuován balík zkušebních programů jehož součástí je TCP/IP implementace vytvořená původně Andreasem Dannenberg na HTWK Leipzig, Univerzity aplikované vědy v Německu, upraven byl dodavatelem softwaru pro LPCXpresso. Jedná se o velmi minimalistickou implementaci, která je vhodná k prvním pokusům o síťovou komunikaci na LPCXpresso.

Jde o projekt, na kterém můžeme poznat principiální způsob implementace TCP/IP stacku a zjistit jakým způsobem je vytvořena komunikace s hardwarem v LPCXpresso. K další práci s vývojovým kitem ovšem přestane velmi brzo dostačovat, na nasazení do provozu se jistě nehodí. Ale je opravdu rozumné od této implementace začít zkoumat TCP/IP. Ve flash zabírá cca 4KB paměti.

6.2.2 Nichelite TCP/IP

Jde komerční implementaci TCP/IP stacku [28], která je cíleně směřována pro vestavěná zařízení. Jedná se o kompaktní a přitom plně implementovanou verzi TCP/IP, která se

striktně drží RFC norem. Poskytuje i BSD-like odlehčené rozhraní nazývané „Mini Socktes API“. Zajme i neblokujícími verzemi všech funkcí, kdy není třeba aktivního čekání – využívá se přerušení. Tato implementace nebývá u malých implementací TCP/IP příliš častá. Ve flash zabírá cca 12KB paměti.

Je poskytována časově omezená verze, náhodně jsem ovšem na stránkách podpory LPCXpresso [21] našel informaci s odkazem na upravenou verzi nichelite, která má být zdarma dostupná pro uživatele LPCXpresso a která je na LPCXpresso použitelná bez velkých úprav.

Tato verze nichelite byla ovšem velmi problematická. Ačkoliv byla prezentovaná jako určená pro LPCXpresso, tak vůbec nebyla postavena nad knihovny dostupnými v LPCXpresso IDE. Tudíž v tomto vývojovém prostředí nešla zkompileovat. A bohužel vývojové prostředí se všemi knihovnami, které jsou používány u klasické verze nichelite, je nezanedbatelně zpoplatněno. Pokusy o přepracování této implementace, která by byla opravdu využitelná na LPCXpresso, bez nutnosti platit za další software se mi nepodařilo dotáhnout do konce. Pouze jsem tím ztratil spoustu času, a navíc by při použití takto upraveného nichelite pak byl nejspíše problém i s licenčními podmínkami.

6.2.3 lwIP

lwIP je velmi zajímavá open source implementace TCP/IP protokolového stacku. Tato implementace byla vytvořena Adamem Dunkelsem v laboratořích Švédského institutu počítačových věd - Swedish Institute of Computer Science (SICS). lwIP bylo vytvořeno za účelem implementace plnohodnotného TCP/IP stacku pro zařízení s omezenými systémovými zdroji. Toto dělá z lwIP výborného kandidáta do výkonnějších vestavěných zařízení. Ve flash zabírá cca 40KB paměti a je distribuován pod BSD licenci [24].

Zajímavé funkce:

- IP – obsahuje i směrování paketů přes vícero připojených fyzických rozhraní
- ICMP – pro síťovou správu a debuggování
- UDP – obsahuje experimentální podporu pro UDP-lite rozšíření
- TCP – implementováno i s kontrolou zahlcení, rychlým zotavením z chyb a rychlým znovu odesláním

- Možnost využití specializovaného „raw“ (nízkoúrovňové) API pro vyšší výkon
- Doplnkové BSD socket-like API
- DHCP – automatická konfigurace síťového rozhraní
- PPP – protokol pro spojení bod-bod
- ARP – protokol pro zjišťování fyzických adres v ethernetu

6.3 Praktické využití lwIP v projektu

V této kapitole budou přiblíženy některé zajímavé skutečnosti z jádra lwIP. Následně bude ukázáno jakým způsobem se v rámci tohoto projektu s lwIP pracovalo. Bude popsán i webový server, který zprostředkovává uživatelské prostředí, a který je na lwIP přímo založen.

6.3.1 Druhy lwIP API

LwIP umožňuje využití tří odlišných druhů programátorských přístupů. Jedná se o tato rozhraní [23]:

- nízkoúrovňové raw API – je využíváno vysokoúrovňovými API
- vysokoúrovňové netconn API – klasické sekvenční API (nejpoužívanější)
- vysokoúrovňové socket API – sekvenční API zaměřeno na kompatibilitu s normami posix a BSD-sockety

Jako nejvhodnější možnost pro projekt vestavěného systému se z počátku asi jeví využití sekvenčního netconn API, bohužel toto API (stejně jako socket API) je určeno pro běh nad nějakým operačním systémem, protože využívá principu vláken. S použitím operačního softwaru se v tomto projektu nikdy nepočítalo a tudíž je zvoleno nízkoúrovňové API. To je vytvořeno proto, aby běželo na zařízeních bez operačního systému. Využívá principu tzv. callbacků. Jak se tento přístup využívá v praxi je naznačeno v následující kapitole.

6.3.2 Nasazení lwIP

Jelikož jsme nuceni použít nízkoúrovňové raw API, tak se tím pádem hlavní smyčka programu (v inicializační části funkce *main()*) musí postarat o několik základních věcí:

- Nejprve musejí být inicializovány struktury lwIP, jde hlavně o paměťové buffery
- poté následuje inicializace lwIP softwarových modulů, které odpovídají vrstvám TCP/IP modelu – nejprve tedy „linkové“ ARP, poté „síťové“ IP a následně „transportní“ TCP či UDP
- pak je do systému zaregistrováno síťové rozhraní, které je v tomto kroku zároveň nastaveno (IP adresa, síťová maska a brána)
- dále je nastartován samotný fyzický řadič, jehož obslužné funkce (v implementaci ovladače) budou využívat předpřipravené struktury ve výše uvedených krocích

Po této sekvenci máme připraven plně inicializovaný lwIP TCP/IP stack. Dále se inicializují ostatní části vestavěného systému, jedním z nich je i webový server. Pro správnou funkci síťové komunikace se ovšem musíme cyklicky v určitých intervalech (dané specifikací konkrétního protokolu) neustále starat o tyto věci:

- musíme spouštět zpracování přijatého ethernetového rámce (pokud byl přítomen) lwIP stackem, po jeho zpracování se již automaticky předá řízení do vyšší vrstvy TCP/IP stacku a tak to pokračuje až do aplikační vrstvy; proběhne tak vlastně „odpouzdření“ přijatého požadavku z počítačové sítě; je zde provedena i kompletní obsluha v aplikaci (web server) a následně je poslána odpověď zpět do sítě (projde zase zapouzdřováním)
- dále je potřeba spouštět údržbové rutiny pro protokol ARP, např. čištění ARP záznamů
- následně se musí také spustit údržbové rutiny pro TCP, které řeší věci jako zpoždění paketů, znovu vyslání nepotvrzených paketů apod.

Výše uvedené je potřeba zakomponovat do nekonečného cyklu, který je běžně spouštěn po inicializacích vestavěného systému. V tomto cyklu se opakovaně spouští všechny funkce, které má vestavěné zařízení provádět. Obsluha TCP/IP stacku se stává tedy jednou z nich.

Po tomto nastavení již máme funkční lwIP stack. Mělo by být možno úspěšně otestovat spojení pomocí ICMP (klasický příkaz ping). Pokud je spuštěn i nějaký server v aplikační vrstvě lwIP (v našem případě to bude web server), měla by již také fungovat komunikace prostřednictvím serverem obsluhovaném protokolu (v našem případě HTTP).

Než ale takto můžeme začít lwIP vůbec využívat, tak ještě také musíme vyřešit otázku lwIP kompatibilních ovladačů pro náš konkrétní ethernetový řadič. Spolu s lwIP je distribuována i kostra takového ovladače. Musíme ho adekvátně upravit podle námi využívaného typu mikrokontroléru. Principiálně jde o to vytvořit funkce, které budou číst a vkládat data z a do registrů řadiče ethernetu. Toto je na ARM Cortex-M3 platformě vhodné (jako ostatně všechna komunikace s hardware) provést pomocí CMSIS – budeme tedy pracovat se strukturami v jazyce C zapouzdřující celý ethernetový řadič. Nejde o úplně triviální úkol, ale materiálů a návodů užitečných k jeho zvládnutí je dostatek [22] a [23].

6.3.3 Web server

Web server v kombinaci s webovým prohlížečem slouží v případě našeho vestavěného zařízení jako uživatelské rozhraní. Veškerá komunikace mezi uživatelem a vestavěným zařízením půjde právě přes něj. Jako komunikační protokol se používá standardní HTTP.

Za implementaci serveru byla zvolena verze, která je distribuována jako open source přídatný modul spolu s jádrem lwIP. Využívá se zde raw API a jde o poměrně robustní verzi.

V modulu web serveru jsou implementovány technologie CGI (Common Gateway Interface) a SSI (Server Side Includes) pro tvorbu dynamického obsahu. Princip je relativně jednoduchý. Webová stránka obsahuje ve formuláři, kterým se odesílají na server uživatelem zadaná data, odkaz na jméno CGI skriptu. Toto jméno je při inicializaci web serveru namapováno na obslužnou funkci v jazyce C, která se tak při přijetí a zpracování formuláře spustí. V této funkci se může vykonat relativně cokoliv (v našem případě to bude nějaká monitorovací, či řídicí akce) a po jejím ukončení je zpět uživateli předána vygenerovaná stránka. Samotné vygenerování stránky je realizováno pomocí SSI, protože přímé generování html kódu z CGI funkce není ve web serveru implementačně umožněno. SSI doplní do předpřipravené šablony dynamické údaje – ty jsou vázány na výsledky z CGI funkce.

7 MONITOROVÁNÍ A ŘÍZENÍ

Na tomto místě se věnuji způsobům tvorby obecné monitorovací nebo řídicí aplikace za použití vyvinutého systému. Na jednoduchém referenční příkladu je vysvětlen princip jakým se budou moci vytvářet konkrétní monitorovací či řídicí aplikace.

7.1 Monitorování

Monitorování nějakého technologického procesu se bude prakticky provádět získáváním údajů z připojených senzorů. Může jít jak o monitorování dlouhodobé – které bude periodicky provádět čtení údajů ze senzorů, tak i nárazové, které bude naopak cíleně vyvolané zasláním požadavků uživatele skrze CGI ve webovém serveru.

V prvním případě, tedy při dlouhodobém monitorování bude třeba připojit funkci provádějící vlastní získávání údajů ze senzorů do nekonečného cyklu v hlavním programu.

Ve druhém případě bude monitorovací funkce přímo volána z obsluhy CGI.

V obou případech bude potřeba vždy vytvořit modul, nebo sadu modulů, které v sobě budou zapouzdřovat práci s konkrétním druhem senzoru. A to jak po stránce řízení komunikační logiky, tak i po stránce ovládání hardware MCU, který tuto komunikaci bude fakticky zprostředkovávat.

7.2 Řízení

Řízení bude v kontextu našeho systému pouze rozšířením monitorování. Fakticky to znamená, že k monitorovacímu modulu bude přidána pouze nějaká forma „inteligence“. Korektněji řečeno půjde pouze o přidání určité formy řídicí logiky. Takže tím pádem to, co bylo řečeno k vlastní implementaci u monitorování, tak zůstává platné i zde. Fyzicky opět půjde o modul, který bude k snímání údajů ze senzorů přidávat i operace s nějakým druhem výstupních periférií. Spouštění řídicí funkce bude možno provádět buď z CGI, nebo pravidelně z hlavního cyklu programu – opravdu stejně jako je tomu u samotného monitorování.

Pomocí velmi jednoduchých konceptů, které byly představeny v předchozích dvou kapitolách půjde vytvořit rozsáhlé spektrum vestavěných aplikací. Jádro systému bude vždy stejné, měnit se budou pouze moduly pro monitorování či řízení. Následuje popis jednoduché praktické aplikace tohoto přístupu.

7.3 Referenční aplikace

Jednoduchá ukázková aplikace se týká měření teploty pomocí senzorů DS18B20 připojených na sběrnici 1-Wire. Přes webové rozhraní půjde inicializovat zjištění dostupných senzorů na sběrnici. Tato operace fakticky vypíše hardwarové adresy všech dostupných detektorů. Po zadání adresy bude z daného senzoru vyčtena teplota, která bude předána zpět uživateli do klientského software (webového prohlížeče). Následuje popis použitých technologií.

7.3.1 1-Wire

Jedná se o sběrnici navrženou firmou Dallas Semiconductor, která poskytuje nízko rychlostní datový přenos, signalizaci a případně i napájení (speciální parazitní mód viz. datasheet) přes jeden signálový vodič. Sběrnice má jeden řídicí obvod (master) a jeden či více ovládaných zařízení (slave). Všechny obvody jsou zapojeny jak na společnou zem, tak i na společný datový vodič. Tento datový vodič je připojen přes odpor cca 4,7k ohmů na napájecí napětí a "zdvihá" tak sběrnici v klidu do log. 1 (jde o tzv. pull up rezistor). Pokud není používán speciální parazitní mód, tak je potřeba ještě vyřešit napájení slave [25], [26], [29].

Komunikace dle protokolu 1-Wire sběrnice je prováděna na jednom pinu GPIO (obecném vstupně/výstupním). Ve vlastním modulu zapouzdřující 1-Wire komunikaci jsou implementovány základní nízkoúrovňové funkce pro reset sběrnice, zápis log. 0 nebo 1 na sběrnici a čtení ze sběrnice. Prodlevy mezi jednotlivými akcemi na sběrnici a určení tak jejich významu (různé sekvence čtení a zápisů na sběrnici – mají různý význam) je hlídáno pomocí časovačů MCU.

7.3.2 Senzor teploty DS18B20

Jde o digitální teplotní senzor připojovaný na 1-Wire sběrnici a poskytující 9-bitovou až 12-bitovou přesnost měření ve stupních Celsia. Dokáže měřit v prostředí od -55°C do $+125^{\circ}\text{C}$, přičemž v rozsahu od -10°C do $+85^{\circ}\text{C}$ měří s maximální odchylkou $\pm 0.5^{\circ}\text{C}$. DS18B20 může být napájen separátním vodičem s napětím od 3V do 5,5V nebo může využít napájení přes datový vodič (parazitní napájení).

Každý senzor DS18B20 má unikátní 64 bitový hardwarový kód, který od sebe dovoluje odlišit konkrétní senzory a umožňuje tak provoz více senzorů na jedné 1-Wire sběrnici. Díky tomu je umožněno, že jeden MCU umožňuje řídit mnoho senzorů rozmístěných po velké oblasti. Běžnými aplikacemi pro teplotní senzor DS18B20 jsou snímání teploty v budovách, přístrojích, strojích a obecně monitorovacích a řídicích systémech [27].

Vlastní softwarová implementace práce s DS18B20 pouze obaluje nízkourovňové funkce v modulu implementujícím 1-Wire komunikaci samotnou. Modul s implementací 1-Wire je tak potenciálně využitelný i jinými druhy senzorů pracujících na této sběrnici než je pouze DS18B20.

8 PROPOJENÍ ČÁSTÍ SYSTÉMU DO CELKU A TESTOVÁNÍ

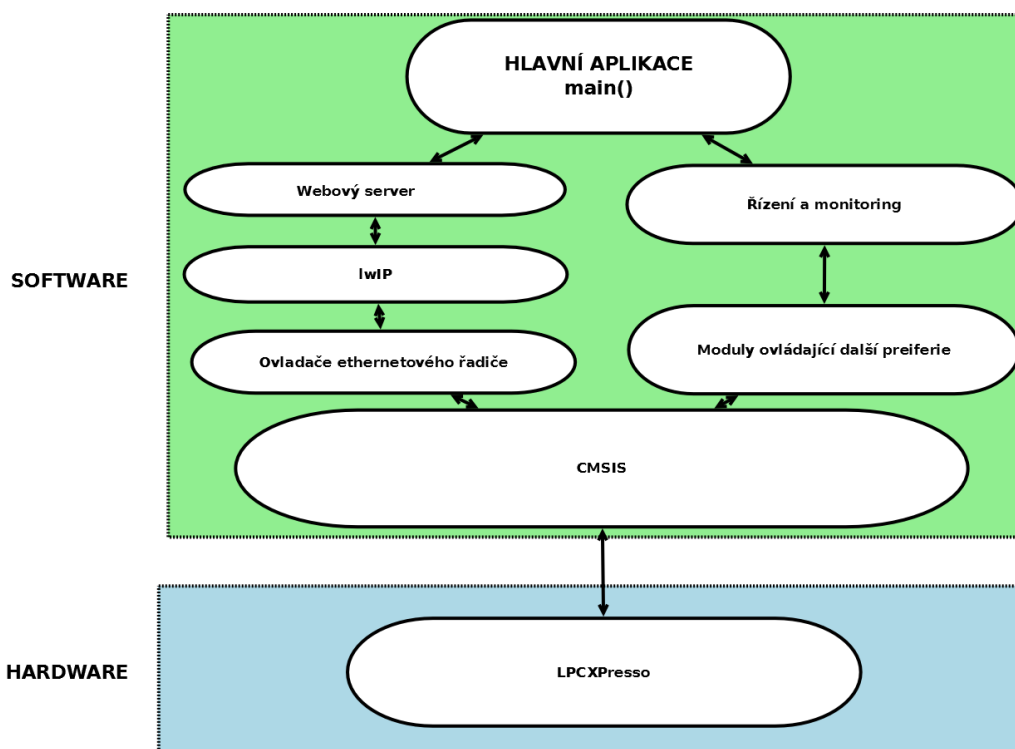
V této kapitole popisují vlastnosti a strukturu vyvinutého systému jako celku a uvádí jakým způsobem byl reálně otestován.

8.1 Modulární struktura systému

V průběhu celého vývoje řídicího a monitorovacího systému byl kladen velký důraz na modulární přístup. Byla vyvinuta velká snaha o to, rozdělit implementaci systému do několika spolupracujících vrstev tak, aby bylo možno relativně jednoduchým způsobem rozšiřovat a upravovat výsledný systém. Změna v jedné vrstvě při zachování stejného rozhraní a způsobu využití této vrstvy, by neměla vyvolat nutnost zásahu do jiné vrstvy (modulu).

Do takto postaveného systému je možno přidávat ovladače fyzických rozhraní a implementace protokolů používaných fyzickými zařízeními, jako jsou různé senzory apod., které mohou být na tato rozhraní připojovány. Z těchto stavebních kamenů může být následně postaveno rozsáhlé spektrum monitorovacích a řídicích aplikací. Následuje náčrt modulární struktury systému s popisem vrstev, tak jak byla implementována.

STRUKTURA SYSTÉMU



Obrázek [11] – Struktura vytvořeného systému

System se dělí na softwarovou a hardwarovou část, kde hardware představuje vývojový kit s mikrokontrolérem NXP LPC1769 a software představuje firmware tohoto mikrokontroléru. Jednotlivé podčásti mají následující význam:

- LPCXpresso – vývojový kit, slouží jako hardware vestavěného systému
- CMSIS – knihovna poskytující softwarové rozhraní k hardware
- ovladač ethernetového řadiče – software přímo využívající CMSIS knihovnu pro ovládání fyzického rozhraní sítě ethernet
- moduly ovládající další periferie – implementace spolupráce s různými typy externích fyzických zařízení (senzory apod.); jedná se o zapouzdření komunikační logiky a ovládání hardware (prostřednictvím CMSIS), které tuto komunikaci fyzicky vykonává
- lwIP – modul implementující TCP/IP stack a obstarávající tak komunikaci v počítačové síti (mimo aplikační vrstvy)
- webový server – faktické uživatelské rozhraní celého systému, implementuje aplikační vrstvu síťové komunikace
- moduly provádějící řízení a monitoring – implementace součástí, které využívají ovládání senzorů a dalších fyzických prostředků nižší vrstvou k propojování těchto prvků do aplikací (funkcí), které mají řídicí, či monitorovací charakter
- hlavní aplikace – celou softwarovou část zapouzdřující modul, stará se o inicializaci systému, stejně jako o plánování chodu jednotlivých součástí

8.2 Testování

Konečné testování systému proběhlo na jednoduché monitorovací aplikaci, která využívala trojici teplotních senzorů DS18B20 zapojených na 1-Wire sběrnici v odstupech cca 15cm od sebe. Byla vyzkoušena základní komunikace s těmito senzory. Následně byla zprovozněna webová aplikace, která umožnila vzdálené zjišťování informací z jednotlivých senzorů. Přes webový prohlížeč běžící na vzdáleném počítači bylo učiněno spojení na IP adresu vestavěného zařízení a byla obdržena webová stránka informující o teplotách zjištěných jednotlivými senzory. Přes tuto webovou stránku a v ní umístěných formulářích bylo možno iniciovat spojení s vlastními senzory, kromě zjišťování teploty

bylo možno resetovat 1-Wire sběrnici a senzory na ní připojené a také vyhledávat nově přidané senzory a vypisovat jejich jedinečnou hardwarovou adresu. Velikost celého firmware po spojení všech využitých modulů a knihoven dohromady byla 83KB kódu. Jelikož je flash paměť použitého mikrokontroléru velká 512KB, tak je ještě dostupno mnoho prostoru k rozšiřování.

9 ZHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ A MOŽNOSTI VYUŽITÍ NAVRŽENÉHO ŘEŠENÍ

Na tomto místě shrnuji dosažených výsledků, uvádím co se při vývoji povedlo realizovat a co by bylo vhodné udělat jinak, zmiňuji také možnosti rozšíření.

9.1 Zrealizované cíle

System, který se nakonec povedlo vytvořit, je univerzálním systémem s širokými možnostmi rozšíření. Lze ho využít pro implementaci monitorovacích a řídicích vestavěných aplikací různého druhu nasazení. Použité techniky umožňují vytvořit malé vestavěné aplikace stejně jako ty poměrně rozsáhlé, vytvořený systém je dobře škálovatelný. Při realizaci referenční aplikace bylo použito jen velmi omezené až zanedbatelné množství hardwarových zdrojů.

Jako zbytečně složité řešení hodnotím využití web serveru z balíku lwIP, u kterého jde v případě použití na vestavěných zařízeních o zbytečně komplexní modul, v případě další práce s tímto systémem bych uvažoval o přepracování tohoto web server do jednodušší formy.

System jako celek splnil požadavky na něj kladené a při jeho vývoji se ukázaly široké možnosti využitelné při návrhu vestavěných aplikací fungujících na platformě ARM Cortex-M3.

9.2 Možná rozšíření

Jelikož je vyvinutý systém přímo vytvořen pro možnost jednoduchého rozšiřování funkčnosti pomocí modulů, tak je zde pouze vhodné zopakovat, že je možno vytvořit nejrůznější monitorovací a řídicí funkce. Je vhodné začít s implementací ovládání dalších druhů senzorů než byly 1-Wire teploměry. System by tedy bylo v prvním kroku vhodné rozšířit o spolupráci se senzory vlhkosti, tlaku a obecně dalších veličin využitelných v technologických procesech.

ZÁVĚR

Povedlo se navrhnout, implementovat a otestovat systém pro řízení a monitorování na vestavěných zařízeních. Jde o otevřenou platformu, která se skládá z hardware a základního softwarového balíku implementujícího jádro systému. Tento systém je poměrně snadno rozšiřitelný a adaptovatelný na konkrétní technologický proces, jehož monitorováním či řízením by mohl být pověřen.

Byly zde také představeny všechny důležité techniky, technologie a postupy, které mohou pomoci proniknout novým zájemcům do tvorby vestavěných zařízení s možností komunikace prostřednictvím standardizovaných protokolů z rodiny TCP/IP. Práce byla realizována na mikrokontrolérech ARM Cortex-M3 od firmy NXP, tato skutečnost ovšem nebrání v použití získaných poznatků i na jiných platformách, protože mnoho zde uvedených informací je obecně použitelných.

Celkově tedy bylo dosaženo všech cílů, kterých si tento projekt kladl za cíl splnit.

ZÁVĚR V ANGLIČTINĚ

We were able to design, implement and test the system for management and monitoring for embedded devices. This is an open platform that consists of hardware and basic software package which implements core system. This system is relatively easily extensible and adaptable to the specific technological process, which could be entrusted by monitoring or management.

There were also introduced all the important techniques, technologies and practices that can help to reach new users of embedded devices with the possibility of communication through standardized protocols of family TCP / IP. The work was carried out on microcontroller ARM Cortex-M3 from NXP company, but this fact does not prevent the use of knowledge gained on the other platforms, because many of the information which is contained here is generally applicable.

Overall the project has met all the targets.

SEZNAM POUŽITÉ LITERATURY

- [1] LIČEV, Lačezar - MOROKES, David : Procesory - architektura, funkce, použití, Praha, Computer Press, 1999, ISBN 80-7226-172-X.
- [2] VÁŇA, Vladimír : Arm pro začátečníky, BEN technická literatura, Praha, 2009, ISBN 978-80-7300-2.
- [3] PUŽMANOVÁ, Rita : Moderní komunikační sítě od A do Z, 2. aktualizované vydání, Praha, Computer Press, 2006, ISBN 80-251-1278-0.
- [4] VLACH, Jaroslav : Řízení a vizualizace technologických procesů, Praha, 1999, BEN technická literatura, ISBN 80-86056-66-X.
- [5] YIU, Josef : The Definitive Guide to the ARM Cortex-M3, Newnes, Burlington, 2012, ISBN 185617963X.
- [6] WIKIPEDIA : Embedded system [online] (duben 2012)
Dostupný z WWW <http://en.wikipedia.org/wiki/Embedded_system>
- [7] WIKIPEDIA : Microcontroller [online] (březen 2012)
Dostupný z WWW <<http://en.wikipedia.org/wiki/Microcontroller>>
- [8] JEŽEK, David : Lidská noha vkročila na měsíc před 40 lety [online] (březen 2012)
Dostupné z WWW <<http://diit.cz/clanek/apollo-guidance-computer-podrobneji>>
- [9] WIKIPEDIA : CISC [online] (březen 2012)
Dostupné z WWW <<http://cs.wikipedia.org/wiki/CISC>>
- [10] WIKIPEDIA : RISC [online] (březen 2012)
Dostupný z WWW <<http://cs.wikipedia.org/wiki/RISC>>
- [11] ARM : Cortex-M Series [online] (březen 2012)
Dostupný z WWW <<http://www.arm.com/products/processors/cortex-m/index.php>>
- [12] WIKIPEDIA : ARM [online] (duben 2012)
Dostupný z WWW <<http://cs.wikipedia.org/wiki/ARM>>

- [13] ARM : CMSIS - Cortex Microcontroller Software Interface Standard [online] (duben 2012)
Dostupný z WWW <<http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php>>
- [14] WIKIPEDIA : Referenční model ISO/OSI [online] (duben 2012)
Dostupný z WWW <http://cs.wikipedia.org/wiki/Referen%C4%8Dn%C3%AD_model_ISO/OSI>
- [15] PETERKA, Jiří : Referenční model ISO/OSI – přenos dat [online] (duben 2012)
Dostupný z WWW <<http://www.earchiv.cz/a92/a215c110.php3>>
- [16] WIKIPEDIA : Global System for Mobile Communications [online] (březen 2012)
Dostupný z WWW
<http://cs.wikipedia.org/wiki/Global_System_for_Mobile_Communications>
- [17] RICHTER, Tomáš : Historie systému GSM [online] (březen 2012)
Dostupný z WWW <<http://tomas.richtr.cz/mobil/bunk-gsm.htm>>
- [18] MBED : Rapid Prototyping for Microcontrollers [online] (březen 2012)
Dostupný z WWW <<http://mbed.org/>>
- [19] TEXAS INSTRUMENTS : LM3S6965 Ethernet Evaluation Kits [online] (duben 2012)
Dostupný z WWW <<http://www.ti.com/tool/ek-lm3s6965>>
- [20] NXP : LPCXpresso [online] (duben 2012)
Dostupný z WWW <<http://ics.nxp.com/lpcxpresso/>>
- [21] NXP : LPCXpresso support [online] (duben 2012)
Dostupný z WWW <<http://ics.nxp.com/support/lpcxpresso/>>
- [22] NXP : LPC1769/68/67/66/65/64/63 [online] (duben 2012)
Dostupný z WWW
<http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf>

- [23] WIKIA : lwIP - lightweight TCP/IP [online] (duben 2012)
Dostupný z WWW <http://lwip.wikia.com/wiki/LwIP_Wiki>
- [24] DUNKELS, Adam : lwIP [online] (duben 2012)
Dostupný z WWW <<http://savannah.nongnu.org/projects/lwip/>>
- [25] MAXIM INTEGRATED PRODUCTS : 1-Wire® Communication Through Software [online] (březen 2012)
Dostupný z WWW <<http://www.maxim-ic.com/app-notes/index.mvp/id/126>>
- [26] MAXIM INTEGRATED PRODUCTS : Interfacing the DS18X20/DS1822 1-Wire® Temperature Sensor in a Microcontroller Environment [online] (březen 2012)
Dostupný z WWW <<http://www.maxim-ic.com/app-notes/index.mvp/id/162>>
- [27] MAXIM INTEGRATED PRODUCTS : Programmable Resolution 1-Wire Digital Thermometer [online] (březen 2012)
Dostupný z WWW <<http://www.maxim-ic.com/datasheet/index.mvp/id/2812>>
- [28] InterNiche Technologies, Inc. : NicheLite TCP/IP - Compact Networking Protocols [online] (březen 2012)
Dostupný z WWW <<http://www.iniche.com/nichelite.php>>
- [29] HW server, s.r.o. : Sběrnice 1-Wire [online] (březen 2012)
Dostupný z WWW <<http://www.hw.cz/navrh-obvodu/rozhrani/sbernice-1-wiretm.html>>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PC	Personal Computer – osobní počítač
AGC	Apollo Guidance Computer – navigační počítač mise Apollo – jedno z prvních vestavěných zařízení
MCU	Microcontroller unit – jednočipový počítač, mikrokontrolér
CISC	Complex Instruction Set Computer – komplexní instrukční sada
RISC	Reduced Instruction Set Computer – redukovaná instrukční sada
RAM	Random Access Memory – volatilní paměť pro data programu
ROM	Read Only Memory – nevolatilní paměť pro kód programu
V/V	Vstupně/výstupní
ARM	Advanced RISC Machine – označení známe procesorové platformy
CMSIS	Cortex Microcontroller Software Interface Standard – abstrakční vrstva nad hardwarem použitelná pro rychlý vývoj
GSM	Globální Systém pro Mobilní komunikaci – mobilní telefonní síť
API	Application Programming Interface – programátorské rozhraní aplikace
JTAG	Joint Test Action Group – standard pro programování a debugování MCU
OSI	Open Systems Interconnection – referenční komunikační model
IP	Internet Protocol – směrovací internetový protokol
TCP	Transmission Control Protocol – Spojovaný transportní protokol
TCP/IP	Reálně používaný internetový komunikační model
ICMP	Internet Control Message Protocol – Kontrolní internetový protokol
UDP	User Datagram Protocol – Nespojovaný transportní protokol
DHCP	Dynamic Host Configuration Protocol – protokol pro aut. nastavení sítě
ARP	Address Resolution Protocol – protokol pro zjišťování fyzických adres
PPP	Point-to-Point Protocol – protokol pro přímé spojení dvou uzlů
lwIP	Lightweight TCP/IP stack – odlehčený TCP/IP stack
HTTP	Hypertext Transfer Protocol – protokol určený pro přenos webových stránek
CGI	Common Gateway Interface - protokol pro propojení externích aplikací s webovým serverem
SSI	Server Side Includes – jednoduché skriptování na straně serveru

SEZNAM OBRÁZKŮ

- [1] Znázornění komunikace v ISO/OSI modelu
- [2] Znázornění komunikace v ISO/OSI modelu
- [3] Znázornění struktury GSM sítě
- [4] Princip CSMA/CD
- [5] Uspořádání vodičů UTP kabelu a RJ-45 konektor
- [6] Síťové topologie
- [7] Vývojový kit Mbed
- [8] Vývojový kit Luminary Micro LM3S696
- [9] Vývojový kit LCPXpresso LPC1769
- [10] Rozdíl mezi TCP/IP a ISO/OSI
- [11] Struktura vytvořeného systému

SEZNAM TABULEK

- [1] Rozdíly mezi architekturou Von Neumannovou a Hardvardskou
- [2] Rozdíly mezi CISC a RISC instrukční sadou
- [3] Přehled nejznámějších MCU
- [4] Přehled nejpoužívanějších V/V rozhraní
- [5] Význam příznakových bitů v instrukcích ARM
- [6] Přehled členů rodiny Cortex-M
- [7] Přehled vrstev ISO/OSI síťového modelu
- [8] Přehled verzí ethernetu
- [9] Minimální implementace TCP/IP

SEZNAM PŘÍLOH

- [1] CD/DVD se zdrojovými soubory, dokumentací a tímto textem