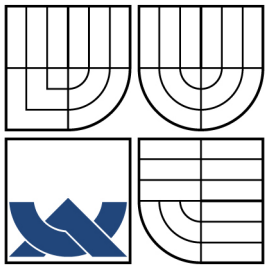
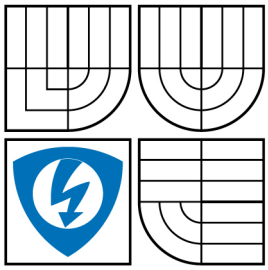


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

OVLÁDÁNÍ ELEKTRONICKÝCH SYSTÉMŮ PŘES WEBOVÉ ROZHRANÍ

ELECTRONIC DEVICE REMOTE CONTROL VIA A WEB INFERFACE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. LADISLAV DUFEK
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. VÁCLAV ZEMAN, PH.D.
SUPERVISOR

BRNO 2011

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Ladislav Dufek
Ročník: 2

ID: 11028
Akademický rok: 2010/11

NÁZEV TÉMATU:

Ovládání elektronických systémů přes webové rozhraní

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte systém pro dálkové ovládání elektronických zařízení. Celý systém dálkového ovládání musí mít možnost monitorovat stav připojeného elektronického zařízení, musí umožňovat přenos jednoduchých řídicích povelů a přístup by měl být řešen prostřednictvím webovského rozhraní na základě autentizace. K realizaci systému využijte vhodného jednodeskového mikropočítače.

DOPORUČENÁ LITERATURA:

[1] Technologic Systems Inc. www.embeddedarm.com [online]. July 23 2009. U.S. Arizona: July 23 2009, [cit. 2010-12-15]. Dostupné z WWW: http://www.embeddedarm.com/products/board-detail.php?product=TS-7350
[2] IETF Documents : RFC 2617 - HTTP Authentication: Basic and Digest Access Authentication [onli-ne]. W3C: July 30, 1997, June 1999 [cit. 2010-12-15]. Dostupné z WWW: <http://tools.ietf.org/html/rfc2617>.

Termín zadání: 7.2.2011 ***Termín odevzdání:*** 26.5.2011

Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

ABSTRAKT V ČEŠTINĚ

Předkládaná diplomová práce se zabývá návrhem a realizací systému pro ovládání elektronických zařízení. Nejprve je stanovena koncepce celého systému a proveden rozbor možných hardwarových platforem. Na základě tohoto rozboru je zpracován návrh systému, vytvořeno obvodové schéma a zhotoven funkční vzorek zařízení.

Další část práce je věnována programovému vybavení, je popsána funkce operačního systému, řídicí aplikace a webového rozhraní. Poté objasněna problematika autentizace a navrhnuto několik autentizačních technik, z nichž je odvozeno vlastní řešení autentizace při přihlášení na webové stránky.

ABSTRACT IN ENGLISH

This master thesis goes into problematics of design of an electronic device control system and its realization. First of all its overall conception is specified and potential hardware platforms are analyzed afterwards. Based on this analysis the system's final conception is drawn up, the device's circuit layout is created and finally a functional prototype is manufactured.

The later part of this thesis describes the software accessories and tools. The principles of a chosen operating system, management application and web-based interface are described along with the problems of authentization to solve. The proposed solutions for the authentization-specific tasks gave rise to the final implementation of the authentization methods and techniques.

KLÍČOVÁ SLOVA

ARM, SBC, Linux, Debian, TS-7350, ISO485, SHA-1, Technologic Systems, Tecomat.

KEY WORDS

ARM, SBC, Linux, Debian, TS-7350, ISO485, SHA-1, Technologic Systems, Tecomat.

DUFEK, L. *Ovládání elektronických systémů přes webové rozhraní.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2011. 55s., 10s. příloh. Vedoucí diplomové práce doc. Ing. Václav Zeman, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Ovládání elektronických systémů přes webové rozhraní jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 26. května 2011 .....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce doc. Ing. Václavu Zemanovi, Ph.D., za velmi užitečnou metodickou pomoc a cenné rady při zpracování práce.

V Brně dne 26. května 2011 .....

podpis autora

Obsah

Úvod **10**

1 Koncepce systému **11**

1.1 Základní požadavky na ovládací systém **11**

1.2 Návrh ovládacího systému **11**

1.3 Přehled možných řešení **12**

1.3.1 Řešení na základě embedded modulů 12

1.3.2 Programovatelné automaty 13

1.3.3 Jednodeskové mikropočítače 14

1.3.4 SBC Technologic Systems 14

1.4 Výběr vhodné platformy **16**

2 Návrh ovládacího systému **16**

2.1 Rozšiřující modul systému **16**

2.2 Schéma zapojení rozšiřujícího modulu **19**

2.2.1 Digitální vstupy a výstupy modulu 19

2.2.2 Vnější obvody vstupů modulu 20

2.2.3 Přizpůsobení napěťových úrovní 3,3V a 5V 21

2.2.4 Měření spotřeby elektrické energie 23

2.2.5 Napájení obvodů elektroměru 24

2.3 Návrh desky plošných spojů **26**

2.3.1 Osazení a oživení DPS 27

3 Programové vybavení **28**

3.1 GNU/Linux distribuce Debian **28**

3.1.1 Tvorba instalačního disku 28

3.1.2 Životnost SD karty v systému 30

3.1.3 Spuštění systému Debian 30

3.1.4 Konfigurace základních služeb 31

3.2 Tvorba ovládacího programu **33**

3.2.1 Struktura navrženého software 33

3.2.2 Modul ovládání IO Controls 35

3.2.3 Modul Modbus Control 35

3.2.4 Modul SPI Control 36

3.2.5 Modul TCP Routines 36
4 Webové rozhraní 38
4.1 Autentizace uživatele 38
4.1.1 HTTP Autentizace 38
4.1.2 Digest Access Authentication 38
4.1.3 Autentizace pomocí asymetrické šifry 40
4.1.4 Autentizace osobním certifikátem 40
4.2 Autentizace uživatele ve webovém rozhraní 40
4.2.1 Popis provedení autentizační procedury 41
4.3 Popis webové aplikace 44
4.3.1 Synchronizace hodnot 45
4.3.2 Změny nastavení 45
4.4 Komunikace mezi webovým rozhraním a ovládací aplikací 46
4.4.1 Otevření socketu 47
4.4.2 Navázání komunikace 47
4.4.3 Odeslání příkazu get/set na socket 48
4.4.4 Uzavření socketu 48
4.4.5 Odhlášení uživatele 49
5 Závěr 50
6 Seznam literatury 52
7 Seznam použitých zkratek, veličin a symbolů 54

Seznam obrázků

Obr. 1.1: Blokové schéma systému 12
Obr. 1.2: Jednodeskový mikropočítač TS-7350 15
Obr. 2.1: Blokové schéma vnitřního zapojení SBC a připojení vstupů a výstupů 17
Obr. 2.2: Rozšiřující karta sériové linky RS-485 18
Obr. 2.3: Blokové schéma zapojení posuvných registrů 19
Obr. 2.4: Časový diagram ovládání posuvných registrů 20
Obr. 2.5: Schéma zapojení digitálního vstupu 20
Obr. 2.6: Schéma přizpůsobení napěťových úrovní mezi 3,3V a 5V logikou 22
Obr. 2.7: Schéma zapojení napájecího zdroje elektroměru 23
Obr. 2.8: Schéma zapojení napájecího zdroje elektroměru 24
Obr. 2.9: Rozšiřující deska ovládacího systému 26
Obr. 2.10: Osazený a oživený prototyp ovládacího systému 27
Obr. 3.1: Struktura oddílů na SD kartě (z programu GParted) 28
Obr. 3.2: Diagram částí ovládacího software 33
Obr. 4.1: Obrazovka webové aplikace – stránka vstupů a výstupů 44

Seznam tabulek

Tabulka 4.1: Stručný přehled jednotlivých souborů 49

Úvod

V oblasti moderní spotřební elektroniky se stále častěji setkáváme se spojením tradičních výrobků s informačními technologiemi. Celou řadu přístrojů jakými jsou televize, videopřehrávače, set-top-boxy, přídavná síťová úložiště a další, lze vzájemně propojit pomocí lokální počítačové sítě a rozšířit tak možnosti běžných uživatelů o využití dalších funkcí, známých především z oblasti osobních počítačů. Prohlížení internetu, stahování programů, sledování on-line videa, vzdálený monitoring a sledování stavu zařízení, rovněž přispívají ke zvýšení komfortu obsluhy a údržby.

Stále však lze nalézt širokou skupinu výrobků, u kterých bychom si rovněž dovedli podobné funkce, jako je vzdálené monitorování, zpracování a sdílení dat, představit, avšak výrobce podobnou možnost nenabízí. Tyto zařízení bývají mnohdy vybaveny pouze základním ovládáním pomocí tlačítek a kontrolek s informacemi o stavu. Modernější zařízení pak již disponují sériovým rozhraním s vlastním protokolem.

Předkládaná diplomová práce si klade za cíl vytvořit systém pro ovládání a monitorování stavu takovýchto zařízení. S použitím vhodného jednodeskového mikropočítače, s integrovaným webovým rozhraním, umožnit přenos jednoduchých řídicích povelů a prostřednictvím okamžité zpětné vazby získat aktuální informace ze zařízení.

Při řešení bude zvláštní důraz kladen na vysokou míru zabezpečení vzdáleného webového přístupu na základě autentizace. Samotné řešení bude založeno na architektuře ARM ve spojení s operačním systémem GNU/Linux. Periferie použitého jednodeskového počítače budou rozšířeny zkonstruováním základní desky (nového modulu) o řadu vstupů, výstupů, sériových rozhraní (RS 232/485) a obvodů měření elektrické energie. Výhodou je, že tím získáme robustní a transparentní systém schopný odolávat potenciálním pokusům o zneužití, jehož budeme moci v budoucnu dále rozšiřovat, nebo aktualizovat jak softwarové tak hardwarové části.

1 Koncepce systému

1.1 Základní požadavky na ovládací systém

Dříve než začneme s výběrem možné hardwarové a softwarové platformy, stanovíme nejprve základní požadavky na navrhovaný ovládací systém:

- Základním požadavkem je vzdálená správa, prostřednictvím které bude připojené elektronické zařízení monitorováno a ovládáno.
- Vstupní a výstupní periferie pro přenos řídicích povelů.
- USART rozhraní pro sériovou komunikaci.
- Dlouhá doba životnosti výrobku minimálně 10let, a s tím související dlouhodobá dostupnost náhradních dílů.
- Možnost budoucí průběžné modernizace a rozšiřování funkcí, v průběhu technického života systému, ucelený vývojový systém s volnou licencí, s otevřenými zdrojovými kódy operačního sytému, minimálně pro nekomerční užití – kupř. pod licencí GPL a implementace standardních síťových protokolů TCP/IP.
- Autentizace uživatele pro přístup k webovému rozhraní a vysoký stupeň zabezpečení komunikace mezi uživatelem a systémem.
- A v neposlední řadě nízká cena konečného řešení.

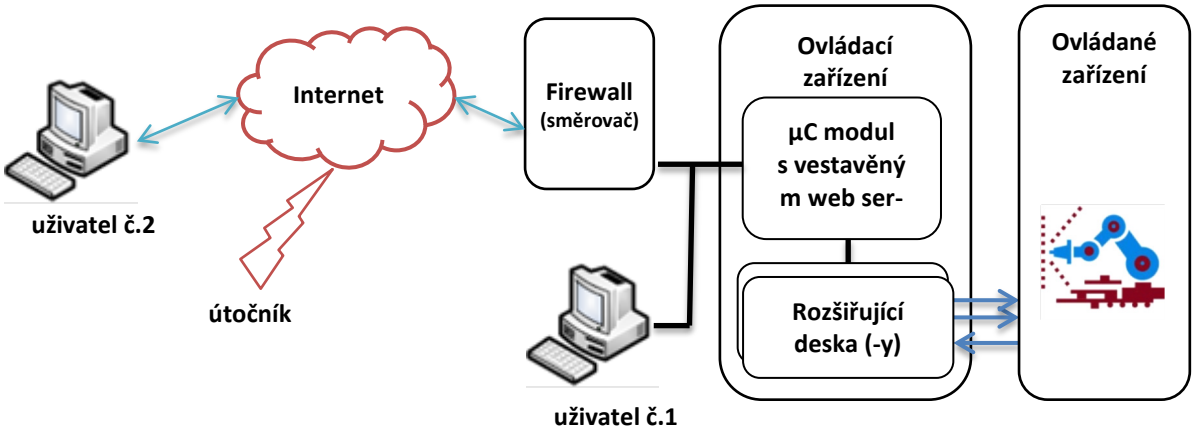
1.2 Návrh koncepce ovládacího systému

Na základě požadavků uvedených v kapitole 1.1 byla navrhnuta základní koncepce systému, která je uvedena na obr. 1.1. Ovládací zařízení tvoří hlavní procesorový modul a rozšiřující moduly, prostřednictvím kterých je ovládané zařízení připojeno. Procesorový modul vykonává obslužný program a mj. poskytuje webovou službu, pomocí které je prováděna veškerá vzdálená obsluha. Případné další služby (SSH, FTP, SFTP,…) mohou sloužit k servisním nebo diagnostickým účelům na modulu.

Prostřednictvím integrovaného webového rozhraní může oprávněný uživatel č. 1 z lokální počítačové sítě, nebo uživatel č. 2 připojený přes síť Internet, provádět obsluhu.

Přítomnost integrovaného webového rozhraní sebou nese riziko spočívající v přístupu do systému neoprávněným uživatelem (útočník), který by tak mohl získat kon-

trolu nad ovládáním systému, což je zcela nepřípustné. V řadě vyráběných modulů s vestavěným webovým rozhraním však nelze zabezpečit komunikační kanál, lze zajisti pouze bezpečné přihlášení, např. pomocí jedné z autentizačních technik. O zabezpečení komunikačního kanálu např. virtuální privátní sítí (VPN), firewallem, atd., se musí v takovýchto případech postarat zařízení třetích stran.



Obr. 1.1: Blokové schéma koncepce systému

1.3 Přehled možných řešení

V současné době mají konstruktéři mikroprocesorových zařízení na výběr z širokého sortimentu nabízených elektronických součástek a embedded modulů. Vhodný a nadčasový výběr platformy je třeba založit na důkladně promyšlených požadavcích. Špatná volba součástkové základny může způsobit, že konstrukčně i programově zdařilou realizaci bude třeba v budoucnu přepracovat pro nedostupnost použitých komponent. Rovněž nízký výpočetní výkon nebo malá operační paměť způsobí, že zařízení nebudeme moci doplnit o nové funkce. Volba hardwarové platformy tedy je jedním s hlavních rozhodnutí předurčující vlastnosti celého systému.

Před zahájením práce byly v úvodní části prostudovány některé možné směry řešení, ať již prostřednictvím malých embedded modulů, nebo PLC, aby nakonec byla zvolena varianta s použitím výkonného jednodeskového mikropočítače.

1.3.1 Řešení na základě embedded modulů

Embedded moduly jsou kompaktní, elektronické jednotky řešící jednoduché úlohy s řadou analogových a digitálních vstupů a výstupů. Vybavenější modely pak poskytují i sériová rozhraní (USB, RS-232, …). Řada výrobců k těmto modulům resp. k mikroprocesorům v nich obsažených, dodává jako součást podpory vývoje i potřebnou sestavu

TCP/IP protokolů. Tato řešení jsou zvláště vhodná tam, kde chceme minimalizovat výrobní náklady a nevadí nám nízký výpočetní výkon a malá rozšiřitelnost. Vestavný software modulů pracuje jako malý webový server, má svoji IP adresu a poskytuje webovou službu na portu 80, která zobrazuje platnou HTML stránku. Zařízení jsou většinou schopná obsloužit současně pouze několik málo požadavků o spojení. Příkladem embedded modulů mohou být jednotky Charon I a II [1] od firmy HW Group s.r.o. či open source projekt Ethernut [2].

1.3.2 Programovatelné automaty

Programovatelné automaty dále jen PLC jsou elektronické řídicí systémy určené k řízení pracovních strojů a procesů v průmyslovém prostředí. S rozvojem „inteligentní domovních elektroinstalací“ řada výrobců vytvořila kategorii levných modulárních PLC, vhodných pro široké spektrum použití. Z českých výrobců lze vyzdvihnout například výrobky firmy Teco a.s. Jejich PLC série Foxtrot [3] poskytuje již v základním provedení, díky 32bit RISC procesoru Motorola ColdFire, solidní výpočetní výkon (1000 instrukcí za 0,2ms). PLC disponuje zálohovanou pamětí 192kB pro program, 64kB pro tabulky proměnných a slot pro paměťové SD karty. Základní modul je standardně osazen WEB serverem pro vizualizaci řízeného procesu. Binárními a analogovými vstupy/výstupy a sériovou linkou RS-232.

Výhodou PLC Foxtrot je, že v systému je již implementována vlastní autentizace. Pro uživatele lze nastavit deset dvojic jméno – heslo. Tyto údaje jsou vyžadovány při přístupu k webserveru přes internetový prohlížeč. Každé dvojici lze nastavit úroveň přístupu (vyšší číslo značí vyšší práva) a výchozí stránku, která se objeví při přihlášení. Úroveň přístupu, vyjadřují práva přihlášeného uživatele. Uživatel může zobrazit a editovat všechny objekty, které jsou stejné nebo nižší úrovně než jeho vlastní. Objekty vyšší úrovně může uživatel pouze zobrazit. Na stránky vyšší úrovně nemůže uživatel přistoupit a nejsou ani zobrazené v menu.

Pro autentizaci využívá PLC zvláštní stránku, oddělenou od ostatního obsahu. Heslo přenášeno jako SHA-1 otisk s přidanou „solí“. Po správném vyplnění hesla je provedeno přesměrování zpět na výchozí webovou adresu. Platnost přihlašovací stránky je půl minuty. Pokud dojde k přihlášení po této době, je zadání jména a hesla vyžadováno znovu. Během platnosti přihlašovací stránky vrací webserver na všechny ostatní požadavky chybu 403 – přístup odepřen. Přihlášení k webserveru vyprší během dvou minut od posledního přístupu. Po vypršení přihlášení je při pokusu o přístup na webový server uživatel přesměrován opět na přihlašovací stránku. U stránek s periodicky obnovovanými

hodnotami (výchozí stav), se počítá každé obnovení jako přístup, tudíž dokud jsou data obnovována a k vypršení nedochází.

Jistou slabinou zabezpečení je nepřítomnost integrovaného firewallu. Ethernetový port tak není chráněn před útoky typu na odepření služby (DoS), při vyčerpání spojení dojde k nedostupnosti webové služby. Nedochází však k ovlivnění služeb, jednotlivé sockety na Ethernetovém portu nejsou dynamicky alokovány, ale každá služba má pevný počet socketů. Na Ethernetovém rozhraní vždy běží komunikační služby EPSNET (UDP 61682) a ModbusRTU (TCP/UDP 502), které nevyžadují autorizaci, jsou dobře zdokumentované a mohou tak být zneužity útočníkem.

Při nahrávání řídicího programu z vývojového prostředí Mosaic jsou hesla přenášena v čitelném tvaru, zapouzdřená komunikačním protokolem EPSNET. Hesla k přístupu přes webové rozhraní jsou následně uložena v prostém textu na SD kartě systému. Soubor je uložen v místě, které není přístupné z uživatelské aplikace ani jej není možné vyčíst pomocí komunikačních služeb. Jediná služba, která má k souboru přístup pro čtení je funkce firmware, která zajišťuje autorizaci uživatelů. Hesla je tak možné vyčíst pouze při fyzickém vyjmutí SD karty a přečtení v PC.

1.3.3 Jednodeskové mikropočítače

Jednodeskové mikropočítače dále jen SBC (Single Board Computer) představují spojení vysokého výpočetního výkonu a operačního systému např. Linux nebo Windows CE. Jsou navrhovány tak, aby mohly být zapojeny a využívány bez složité instalace („out-of-the-box“). Tím dochází ke zkrácení doby vývoje a finální aplikace má zkrácenou dobu uvedení na trh. Jsou vysoce kompaktní, mají klíčová systémová rozhraní a funkcionality integrovány přímo na procesorové desce. To znamená, že pro finální řešení již zbývá pouze osadit konkrétní I/O obvody nutné pro danou aplikaci.

U SBC vybavených standardní systémovou sběrnicí (PC/104, PCI-E), je možné použít přídavné karty a systém tím rozšířit o nová rozhraní či nové funkce.

1.3.4 SBC Technologic Systems

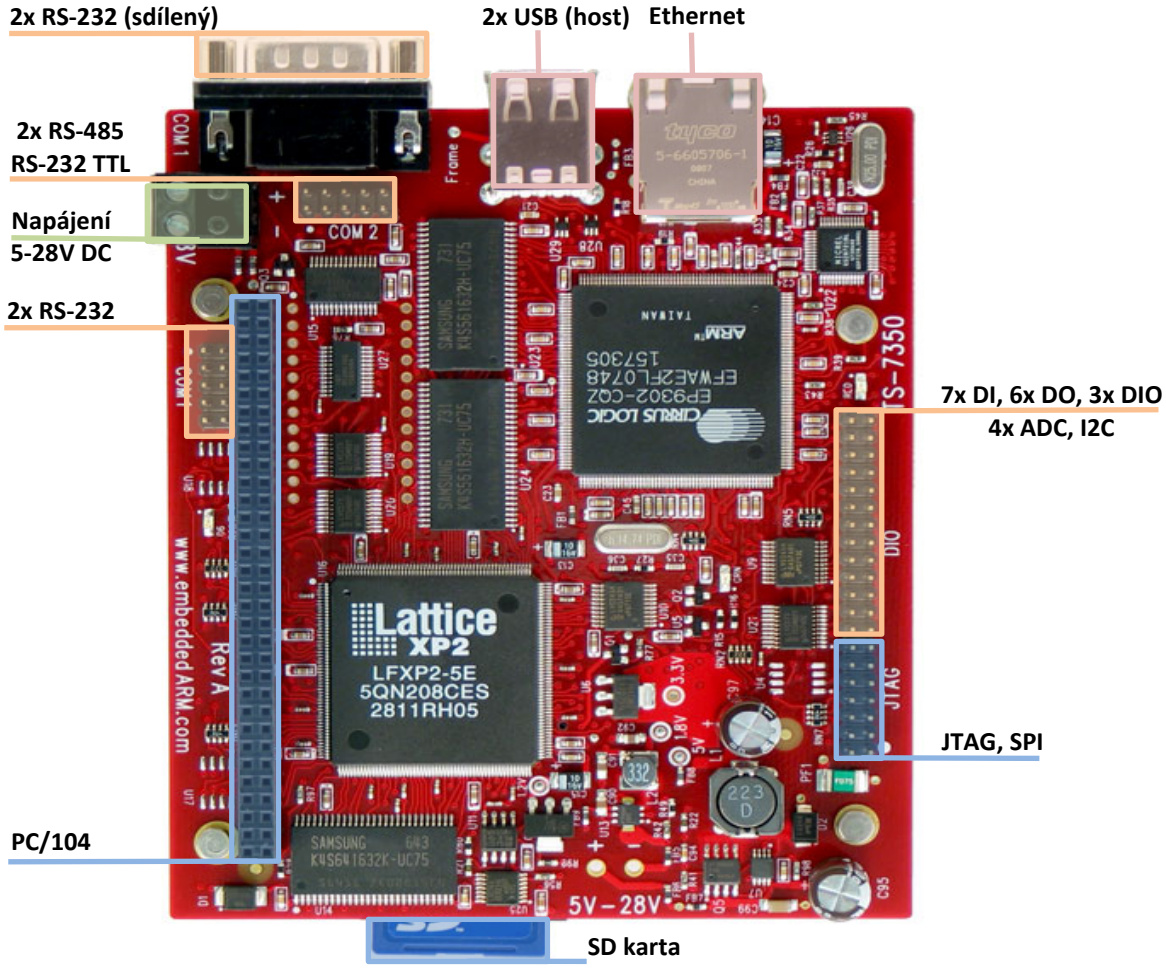
Společnost Technologic Systems je jednou z mnoha významných firem zabývajících se vývojem jednodeskových počítačů a jejich periferií. Obrázek 1.2 ukazuje model TS-7350 a jeho základní výbavu. TS-7350 [4] je modelem střední výkonnostní třídy z portfolia výrobce, postavený na procesoru Cirrus Logic EP9302 200MHz architektury ARM9 s operačním systémem Linux 2.6. Základní jádro OS Linux s minimem spuštěných proce-

sů je schopné startu už za 1,5 sekundy. S postupně se rozšiřující funkcionalitou (Apache2-SSL, SSH, NFS) se start systému může prodloužit na 20 i více sekund.

Uvedený model není vybaven multimediálními výstupy (zvuková karta, monitor a klávesnice), hodí se tak spíše jako embedded modul pro vestavbu do zařízení.

Technické parametry TS-7350:

- Procesor Cirrus Logic EP9320 - 200MHz ARM9 CPU.
- Paměť 32MB SDRAM + 8MB RAM Frame Buffer.
- Ethernet port 10/100MB.
- USB 2.0 host (12Mbit/s max.).
- SD-Card slot (s rychlostí 6MB/s) pro operační systém.
- Sériové linky RS-232, RS-232 TTL, RS-485.
- 40-pin universální rozhraní s ADC, SPI, I2C, DIO.
- Široký rozsah pracovních teplot: od -20° do +70°C a
- napájení 5-28V DC.



Obr. 1.2: Jednodeskový mikropočítač TS-7350 [4] a jeho periferie

1.4 Výběr vhodné platformy

Po zvážení možných variant řídicích desek a jejich příslušenství, byl pro svou snadnou dostupnost a kvalitu provedení vybrán jednodeskový mikropočítač TS-7350. Tento mikropočítač poskytne dostatečný výpočetní výkon pro provoz OS GNU/Linux, webového serveru, SSH serveru, firewallu i šifrování komunikace. Potřebné periferie lze zakoupit jako karty pro sběrnici PC/104, avšak kvůli požadavku na nízkou cenu a kompaktnost řešení, bude v další části práce zkonstruován univerzální **rozšiřující modul**, který vytvoří základní desku pro osazení mikropočítače TS-7350 a zároveň poskytne dostatek periferií stanovených v základní koncepci.

2 Návrh ovládacího systému

Návrh ovládacího systému vychází z představené koncepce a zvoleného řešení. Pro komunikaci systému se svým okolím byly navrženy následující periferie:

- 16× Digitální vstup (z toho 8× galvanicky oddělený a 8× TTL).
- 16× Digitální výstup (z toho 4× relé 24V, 4×relé 230V a 8× otevřený kolektor).
- 3× Linka RS-232 (z toho 1× pro diagnostiku a 1× TTL).
- 3× Linka RS-485 (2× galvanicky oddělená).
- Měření spotřeby elektrické energie (nejlépe třífázové).

Tyto periferie budou sloužit pro přenos jednoduchých řídicích povelů (digitální I/O). V případě složitějších elektronických zařízení bude k přenosu povelů (komunikaci) použita některá ze sériových linek. Měření spotřeby elektrické energie přinese uživateli doplňující informaci o ovládaném zařízení a přispěje ke zvýšení ekonomiky provozu.

2.1 Rozšiřující modul systému

Rozšiřující modul je rozdělen na tři galvanicky oddělené části, každá se svým vlastním napájením. Na obr. 2.1 je zobrazeno navržené uspořádání modulu a vzájemná návaznost jednotlivých dílů a mikropočítače (SBC). Tučně jsou vyznačeny šipky představující paralelní propojení (sběrnice), tenké šipky znázorňují sériové sběrnice.

První část tvoří SBC, řídicí elektronika, RS-232, RS-485 a část I/O. Napájení vnitřní části obvodů je provedeno ze SBC, to především z důvodu úspory nákladů na druhý zdroj v této části. Spínaný zdroj osazený na SBC je dostatečně dimenzovaný vzhledem k mož-

nosti použití rozšiřujících karet PC/104, dle katalogu [4] 5V/2,5A (I_{SBC} = 0,5 A). Přidanou proudovou zátěž modulu I_{CM}, lze spočítat jako součet proudů (hodnoty proudů převzaty ze specifikací součástek [6, 7, 8 , 9 a 10]).

$$I_{CM} = 16I_{LED} + 4I_{RE} + 4I_{K} + 7I_{C\,IO} + 8I_{GO} = \\ = 16 \cdot 2 + 4 \cdot 40 + 4 \cdot 50 + 7 \cdot 0{,}1 + 24 \cdot 1{,}7 \cong 434 \ \ (\text{mA}) \,, \tag{2.1}$$

kde I_{LED} … je proud LED diodou (I_{LED} = 2 mA),

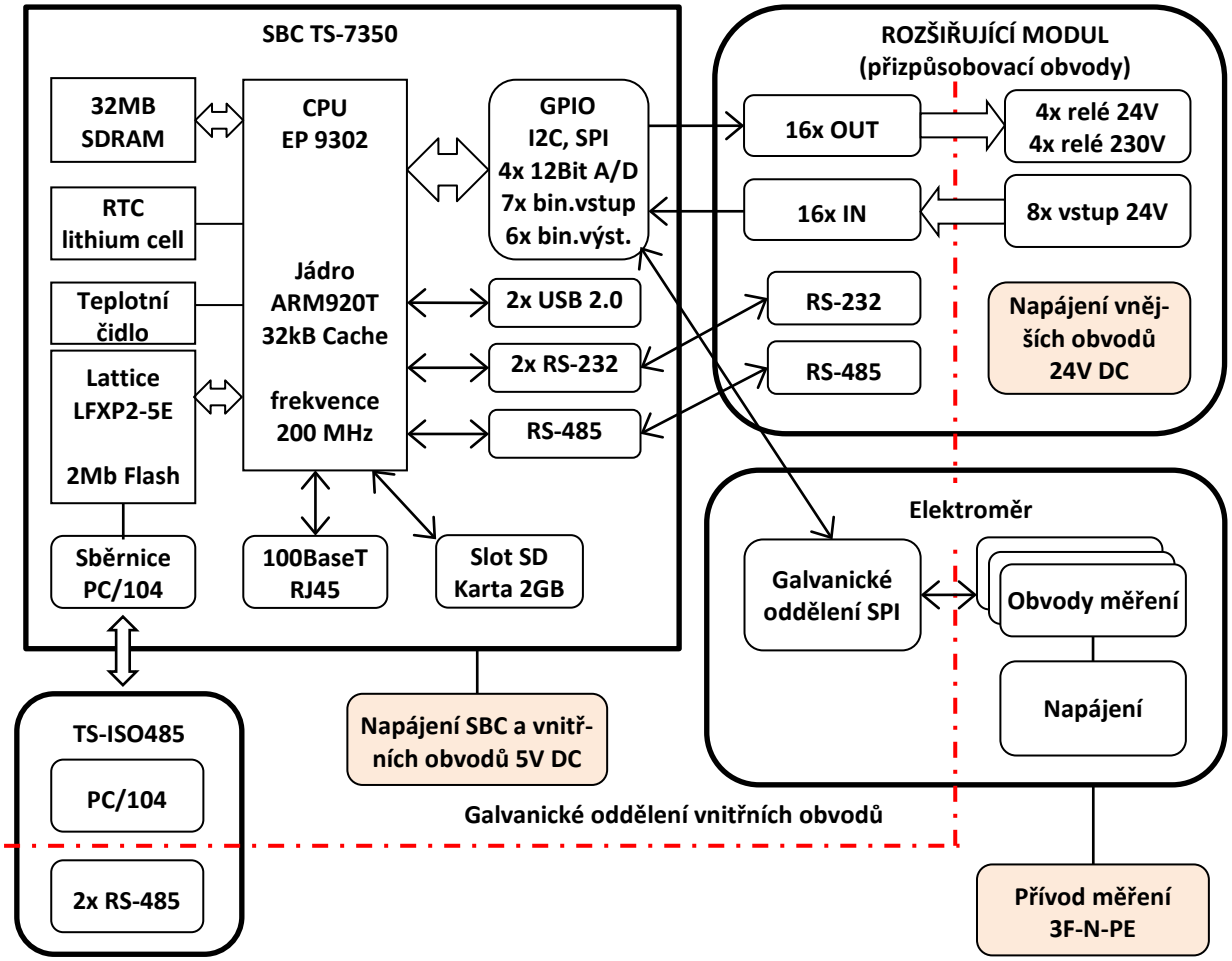
I_{RE} … proud relé RE_1-RE_4 (I_{RE} = 40 mA),

I_{K} … proud relé K_1-K_4 (I_{K} = 50 mA),

$I_{C\,IO}$ … klidový proud IO (pro zjednodušení počítáno 100µA/IO),

I_{GO} … max. proud budiči relé a galvanického oddělení při 5V (I_{GO} = 1,7 mA).

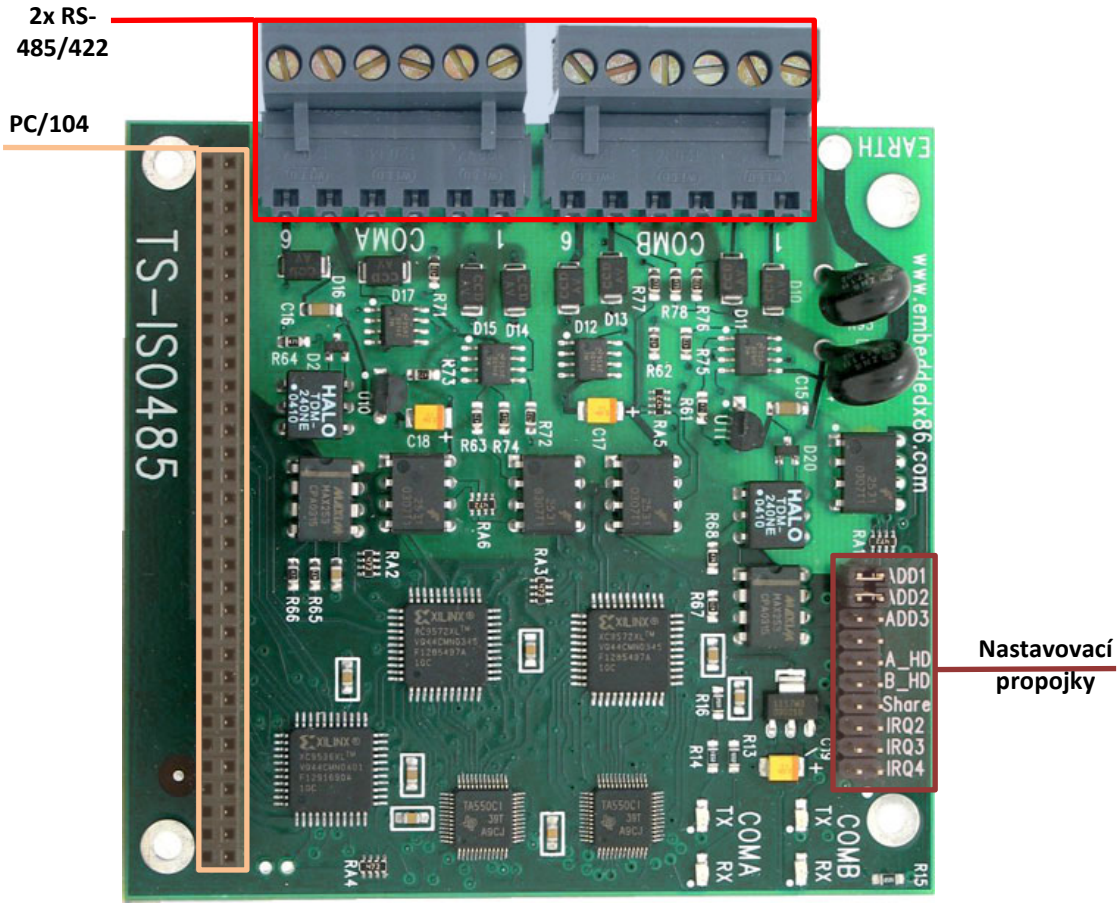
Druhou část tvoří vnější obvody vstupů, napájené z odděleného zdroje 24V. Třetí část, galvanicky spojenou se sítí 230V, tvoří obvody měření spotřeby elektrické energie.



Obr. 2.1: Blokové schéma vnitřního zapojení SBC a připojení vstupů a výstupů

Pro komunikaci s externím zařízením po lince RS-485 byla zvolena možnost osazení externí přídavné karty na sběrnici PC/104. Důvodem je především integrované galvanické oddělení a automatický režim přepínání (Rx/Tx), který usnadní tvorbu případného komunikačního programu. Třetí linka RS-485 vedená z SBC na konektor rozšiřujícího modulu byla v návrhu ponechána bez galvanického oddělení.

Rozšiřující karta TS-ISO485 (obr. 2.2) je PC/104 kompatibilní [5] přídavné zařízení. Obsahuje dvě galvanicky oddělené linky RS-485/RS-422 schopné plně-duplexního a automatického přepínání polo-duplexního režimu. V mnoha implementacích linky RS-485, musí uživatelský software řídit režim transceiveru (vysílání Tx, nebo příjem Rx). Vzhledem k tomu, že rozhraní UART má FIFO paměť a velikost tohoto vysílací bufferu je 16 bytů, může být pro některé aplikace obtížné rozpoznat kdy je třeba z režimu Tx přejí na Rx. Integrovaná logika desky, toto rozpozná a přepne z Tx na Rx přesně ve správný čas, kde je vyslán poslední bit. Maximální přenosová rychlost obou portů činí 115kbd. Konfigurace se provádí pomocí propojek na desce. Pro naši aplikaci je potřeba nastavit piny: ADDR2, A_HD, B_HD, Share, IRQ2, IRQ4.



Obr. 2.2: Rozšiřující karta sériové linky RS-485 [5]

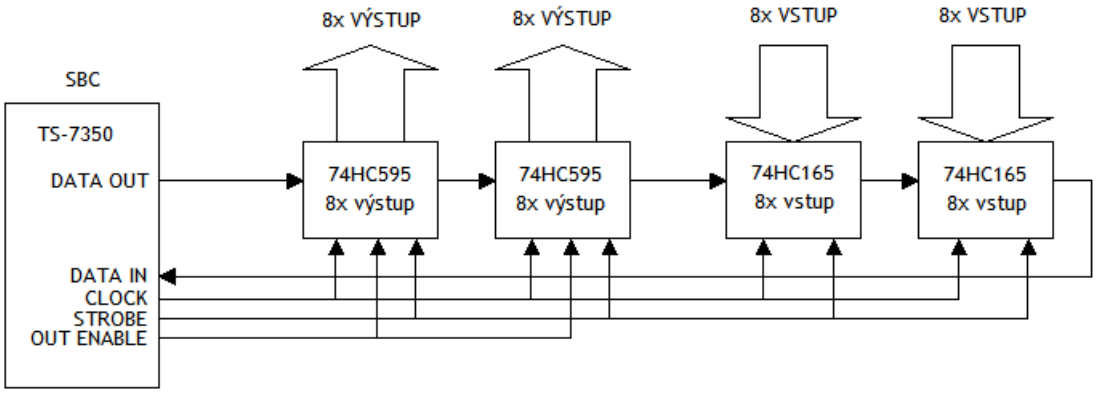
2.2 Schéma zapojení rozšiřujícího modulu

Navržené schéma zapojení rozšiřující desky ovládacího systému je kvůli svému rozsahu uvedeno v příloze B. V několika následujících kapitolách, budou nejdůležitější části zapojení popsány podrobněji.

2.2.1 Digitální vstupy a výstupy modulu

Konstrukce vstupů a výstupů je provedena pomocí kaskádního zapojení dvou 8mi bitových posuvných registrů SIPO typ 74HC595, následovaných dvěma 8mi bitovými posuvnými registry PISO typu 74HC165. Tímto způsobem lze získat 16 binárních vstupů a 16 binárních výstupů. Nevýhodou zapojení je nižší perioda vzorkování, která je dána poměrem mezi frekvencí hodinových pulsů CLOCK a délkou řetězce. Naopak výhodou je použití pouze 5ti vodičů k ovládání. Blokové schéma kaskádního zapojení je patrné z obr. 2.3, detailní schéma zapojení včetně připojení souvisejících obvodů je uvedeno v příloze B na straně 5.

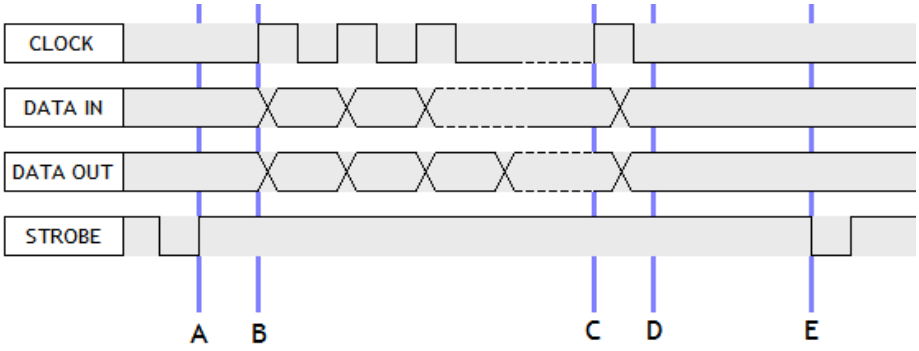
Po přivedení napájení jsou registry výstupních obvodů v nedefinovaném stavu, což by mohlo způsobit nahodilé sepnutí výstupů relé. Aby se tomuto jevu zabránilo, jsou výstupy po startu vypnuty (signálem OUT ENABLE) dokud nedojde k nastavení vnitřních KO na počáteční hodnoty. U vstupů jsou výchozí hladiny definovány připojenými pull-up rezistory.



Obr. 2.3: Blokové schéma zapojení posuvných registrů

Ovládání kaskádního zapojení popisuje časový diagram na obr. 2.4. Signálem STROBE v bodě A přepneme z log.0 na log.1, aplikací náběžné hrany se stavy vstupů přenesou do záchytných posuvných registrů, a současně se přenesou stavy z výstupních registrů do výstupních pinů. Aplikací náběžných hran vstupu CLOCK se údaje v posuvných

registrech posouvají z leva doprava, tím postupně zapisujeme bity do stínových výstupních registrů (bod B), současně čteme signálem DATA IN vstupy ze záchytných posuvných registrů. Protože vstupních i výstupních registrů je stejný počet, čtení i zápis je ukončen v bodě C, po šestnácté náběžné hraně. Nakonec pulsem STROBE (v bodě E) přesuneme hodnoty ze stínových SIPO registrů na výstupní piny obvodu 74HC595.

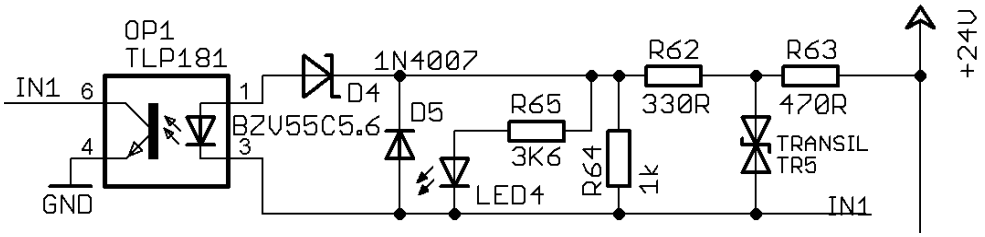


Obr. 2.4: Časový diagram ovládání posuvných registrů

Paralelní výstupy obvodů 74HC595 jsou napojeny na dvě tranzistorová pole ULN2803A poskytující dostatečný proud pro buzení cívek 5V relé. Paralelní vstupy 74HC165 jsou napojeny na vnější obvody vstupů modulu.

2.2.2 Vnější obvody vstupů modulu

Pro rozšiřující digitální vstupy bylo navrženo celkem 8 opticky oddělených vstupů, zapojených podle obrázku 2.5, zbylých 8 vstupů bylo vyvedeno na konektor JP_2 pro případné budoucí rozšíření. V zapojení byl navržen poměrně vysoký proud uzavřené smyčky (kdy je aktivní vstup) cca 20 mA, který však zajistí, že optočlen OP_1 nebude otevírán při výskytu napětí indukovaných na přívodní kabeláži.



Obr. 2.5: Schéma zapojení digitálního vstupu

Vstupní napětí 24V je přivedeno na vstup IN_1, kde je zapojen obousměrný ochranný transil s nominálním napětím 27V, omezující napěťové špičky. Případný impulzní proud je snížen ochranným rezistorem R_{63}. Rezistor R_{64} snižuje vstupní odpor zapojení.

Pro indikaci stavu vstupu je zapojena R_{65} a LED_4. Zvýšení šumové imunity vstupu je provedeno osazením zenerovy diody D_4 v závěrném směru, pomocí které optočlenem OP_1 nepoteče prakticky žádný proud, až do dosažení U_{BR} (tedy 5,6V). Kdyby místo D_4 byl v obvodu zapojen pouze prostý odpor, tak při zvyšování U_{IN1} byl OP_1 otevírán úměrně se zvyšujícím proudem, a mohlo by dojít k přiotevírání optočlenu a s tím souvisejícím nežádoucím zákmitům. Proud optočlenem je nastaven odporem R_{62} a R_{63}. D_5 je protisměrně polarizovaná ochranná dioda, jež chrání LED_4 a LED optočlenu před výskytem zvýšeného závěrného napětí, které by zničilo polovodičové přechody těchto součástek (kupř. při záměně polarity na vstupu, či při výskytu střídavé složky).

Při maximální míře zjednodušení a s uvažováním ideálních součástek dojdeme s použitím Ohmova a Kirchhoffových zákonů k následujícím vztahům popisující stejnosměrné poměry v obvodu.

Z katalogového listu OP_1 [6] zjistíme hodnotu: $U_{OP} = 2{,}1$ V,

$$U_{R64} = U_{D4} + U_{OP1} = 5{,}6 + 2{,}1 = 7{,}7 \ \text{(V)}, \tag{2.2}$$

$$I_{R64} = \frac{U_{R64}}{R_{64}} = \frac{7{,}7}{1000} = 7{,}7 \ \text{(mA)}. \tag{2.3}$$

Proud signalizační LED diodou:

$$I_{R65} = I_{LED4} = \frac{U_{R64} - U_{LED4}}{R_{65}} = \frac{7{,}7 - 2{,}1}{3600} = 1{,}5 \ \text{(mA)}. \tag{2.4}$$

Proud optočlenem:

$$\begin{aligned} I_{OP1} &= I_{D4} = I_{R62} - I_{R63} - I_{R64} - I_{R65} = \\ &= \frac{U_{IN} - U_{R64}}{R_{62} + R_{63}} - I_{R64} - I_{R65} = \frac{24 - 7{,}7}{800} - 7{,}7 - 1{,}5 = 11{,}2 \ \text{(mA)}. \end{aligned} \tag{2.5}$$

Proud při aktivaci vstupu je pak dán součtem (2.3)(2.4)(2.5):

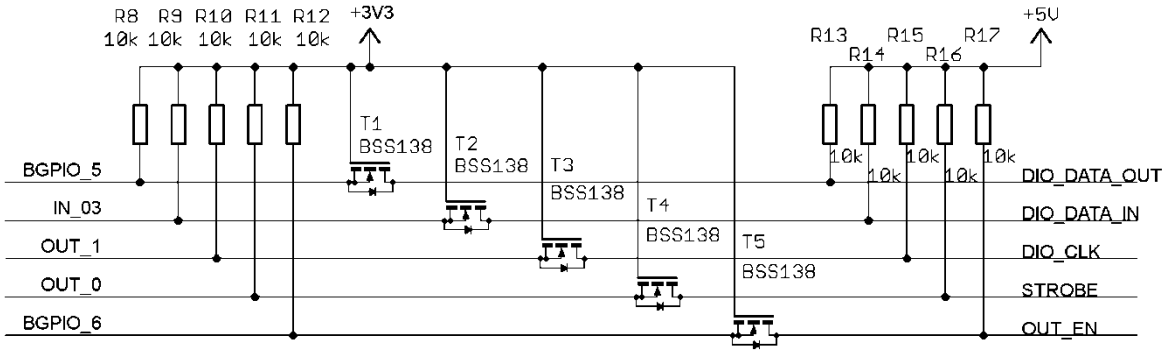
$$I_{IN1} = I_{R64} + I_{R65} + I_{OP1} = 7{,}7 + 1{,}5 + 11{,}2 = 20{,}4 \ \text{(mA)}. \tag{2.6}$$

2.2.3 Přizpůsobení napěťových úrovní 3,3V a 5V

Protože obvody procesoru Cirrus Logic EP9302 na desce TS-7350 pracují s napětím 3,3V je nutné propojit 3,3V logiku procesoru EP9302 s 5V logikou rozšiřujícího modulu. Toto propojení je možné provést pomocí některého z integrovaných obvodů například 74LVC4245, ovšem vzhledem k malému počtu řídicích signálů bylo zvoleno řešení (dle obr. 2.6), s použitím běžných součástek podle aplikační poznámky AN10441 [11] fy. NXP.

Na každé straně napájecích úrovní 3,3V i 5V jsou zapojeny pull-up rezistory. Zařízení na obou stranách používají jako výstup, otevřený kolektor a vstupní logika je přizpůsobena jejich napájecím napětím. Z toho plyne, že úroveň log. 1 je vytvořena pull-up rezistorem a úroveň log. 0 je vytvořena otevřením výstupního tranzistoru, který stáhne úroveň signálu ke GND.

Základem zapojení jsou MOSFET tranzistory s N-kanálem BS138. Ovládací hradlo G (Gate) je připojeno na stranu nižšího napájecího napětí (3,3V). Rovněž na stranu nižšího napětí jsou dále připojeny ovládací signály procesoru SBC, konkrétně na S (source). Třetí vývod tranzistorů D (Drain) je připojen ze strany vyššího napájecího napětí na ovládání posuvných registrů DIO obvodů.



Obr. 2.6: Schéma přizpůsobení napěťových úrovní mezi 3,3V a 5V logikou

Pokud žádná strana negeneruje úroveň log. 0, pull-up rezistory zabezpečují úrovně log. 1 na obou stranách. Vývody MOSFET tranzistoru G a S, připojené na stranu s nižším napájecím napětím jsou na stejné úrovni. Tzn., že nepracují, nejsou sepnuté. Tím jsou obě strany od sebe oddělené, ale každá má svou správnou úroveň napětí.

Když na straně nižšího napětí (3,3V) stáhne SBC digitální vývod ke GND, dojde k otevření tranzistoru, protože se mezi G a S objevilo napětí. Tímto dojde přes sepnutý tranzistor k připojení pull-up rezistoru na straně vyššího napájecího napětí rovněž ke GND. To má za následek změnu úrovně na straně 5V zařízení taktéž na log. 0.

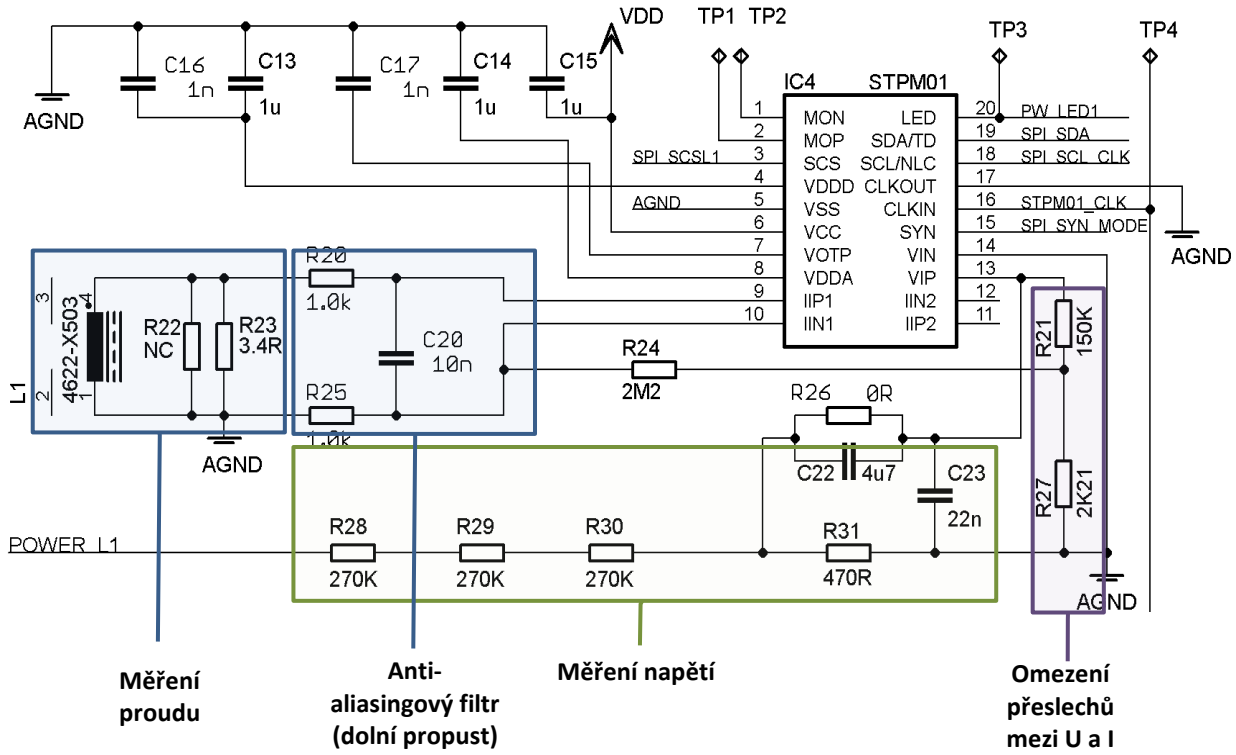
Když zařízení na straně vyššího napětí (5V) stáhne digitální vývod ke GND, dojde k využití interní diody v MOSFET tranzistoru, která je tímto v propustném směru a připojí pull-up rezistor na straně 3,3V napětí ke GND. To má za následek změnu úrovně na 3,3V straně zařízení rovněž na log. 0.

2.2.4 Měření spotřeby elektrické energie

V průběhu řešení projektu vyvstal požadavek na měření spotřeby elektrické energie ovládaného zařízení. Za tímto účelem byl návrh doplněn o tři samostatné obvody měření elektrické energie typu STPM01 od STMicroelectronics.

STPM01 [12] je IO určený pro měření činného, zdánlivého i jalového výkonu, efektivní hodnoty napětí a proudu. Může pracovat samostatně a dávat impulsní výstup úměrný množství odebrané energie ve W/h, nebo přes sériovou sběrnici SPI pomocí vlastního komunikačního protokolu lze číst vnitřní registry okamžitých hodnot měření. O A/D převod se starají 16ti bitové ∑Δ A/D převodníky pro měření proudu a 11ti bitové, pro měření napětí, deklarovaná chyba měření dle katalogu výrobce je 0,1%.

Bylo navrženo zapojení, jehož schéma je na obrázku 2.7. Zapojení vychází z referenčního návrhu výrobce. Vodiče sběrnice SPI, galvanicky spojené se síťovím napětím, bylo třeba oddělit od sběrnice SPI řídicích obvodů a SBC. K oddělení byl použit obvod ADuM3401BRWZ [8], který současně provádí napěťové přizpůsobení 3,3V a 5V úrovní.



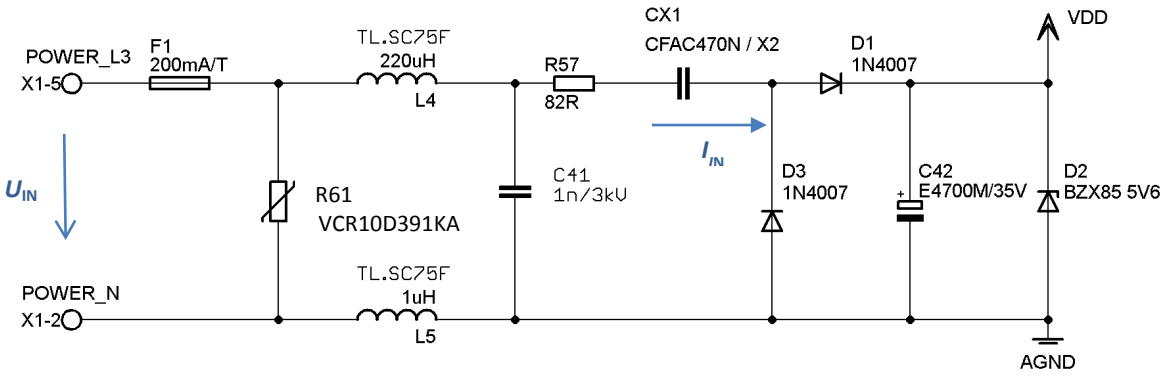
Obr. 2.7: Schéma zapojení napájecího zdroje elektroměru

STPM01 má dva snímací kanály měření proudu primární a sekundární, druhý kanál není v zapojení použit, obvykle slouží pro druhý rozsah měření (malých proudů), nebo pro měření proudů protékaných středním vodičem. Měřicím čidlem je proudový transformátor T60404-E4622-X503 s poměrem 1:2000.

Anti-aliasingový filtr je dolní propust, jejímž cílem je snížit zkreslení způsobené odběrem vzorků tím, že jsou odstraněny vyšší harmonické frekvence. Výrobcem navržený RC-filtr má útlum -20dB/dek. Měření napětí je provedeno odporovým děličem mezi třemi sériově zapojenými odpory R_{28}, R_{29}, R_{30} a R_{31}. Zapojení tří odporů do série je voleno pro zvýšení izolačního odporu mezi sítovým napětím a měřícím obvodem. Kondenzátory C_{23} a C_{22} popř. R_{26} mají vytvořit filtr, který zabraňuje EMI rušení. Měřené napětí má značnou amplitudu, což z ní činí potencionální zdroj šumu, který snadno proniká do obvodů měření proudu, kde interferuje s měřeným signálem. Pokud jsou napětí a proud ve fázi *(*účiník *cos φ = 1)* přeslechy mezi napětím a proudem se projeví jen jako zvýšení zisku, které může být snadno eliminováno kalibrací. Pokud napětí a proud nejsou ve fázi přeslech má nelineární charakter a nemůže být kalibrován. Zapojení R_{21}, R_{24} a R_{27} slouží pro omezení těchto přeslechů, odečte signál úměrný napětí na vstupu z proudového vstupu.

2.2.5 Napájení obvodů elektroměru

Existuje několik způsobů jak napájet měřicí elektroniku elektroměru. Tradiční způsob je použití transformátoru a usměrňovacích a filtračních obvodů. Nicméně z důvodů úspory místa na DPS bylo přistoupeno k možnosti napájení obvodů elektroměru z kapacitního napájecího zdroje, galvanicky spojeného s napájecí sítí 230V dle obrázku 2.8. Zapojení je odvozeno z aplikační poznámky AN2317 [12] firmy STMicroelectronics.



Obr. 2.8: Schéma zapojení napájecího zdroje elektroměru

Jako přepěťová ochrana zařízení, které je připojeno přímo na vstup střídavého napětí 230V je použit varistor R_{61}, v případě zvýšení napětí dojde k jeho otevření a obvodem začne protékat zvýšený proud, následkem čehož je přepálena pojistka F_1. Tlumivky L_4, L_5 a kondenzátor C_{41} spolu tvoří vstupní EMC filtr. Rezistor R_{57} omezuje vysoký vstupní proud při nabíjení kondenzátoru C_{X1}.

Jeho sériový odpor pro střídavé napájecí napětí vyjadřuje kapacitní reaktance X_{C1}, kterou vypočítáme:

$$X_{\mathrm{C1}} = \frac{1}{2\pi f C_{\mathrm{X1}}} = \frac{1}{2 \cdot \pi \cdot 50 \cdot 470 \cdot 10^{-9}} \cong 6773 \ \ (\Omega) . \tag{2.7}$$

Abychom spočítali proud I_{IN} kondenzátorem C_{X1}, vyjdeme ze vztahu pro horní půlvlnu efektivní hodnoty napětí:

$$U_{\mathrm{IN\ RMS}} = \frac{1}{2} \frac{U_{\mathrm{MAX}} - U_{\mathrm{Z}}}{\sqrt{2}} \ \ (\mathrm{V}) , \tag{2.8}$$

kde U_{MAX} …… je špičková hodnota napájecího napětí,

U_{Z} …… součet úbytků napětí na diodách (pro D_1 a D_3 typ. 0,7V),

$$U_{\mathrm{Z}} = U_{\mathrm{D1}} + U_{\mathrm{D2}} + U_{\mathrm{D3}} = 0{,}7 + 5{,}5 + 0{,}7 = 6{,}9 \ \ (\mathrm{V}) , \tag{2.9}$$

$$I_{\mathrm{IN}} = \frac{U_{\mathrm{MAX}} - U_{\mathrm{Z}}}{2\sqrt{2} \cdot \left(X_{\mathrm{C1}} + R_{57}\right)} = \frac{\sqrt{2} \cdot U_{\mathrm{RMS}} - U_{\mathrm{Z}}}{2\sqrt{2} \cdot \left(X_{\mathrm{C1}} + R_{57}\right)} = \frac{\sqrt{2} \cdot 230 - 6{,}9}{2\sqrt{2} \cdot \left(6{,}773 + 82\right)} \cong 15{,}7 \ \ (\mathrm{mA}) . \tag{2.10}$$

Z rovnice (2.10) můžeme spočítat výkonovou ztrátu na diodě D_2 a dostatečně ji dimenzovat.

$$P_{\mathrm{D2}} = U_{\mathrm{D2}} \cdot I_{\mathrm{IN}} = 5{,}6 \cdot 0{,}0157 \cong 0{,}09 \ \ (\mathrm{W}). \tag{2.11}$$

Jednocestně usměrněné napětí diodou D_1, musí být stabilizováno kondenzátorovým filtrem. Hodnota kondenzátoru mj. závisí na maximálním přípustném zvlnění při I_{IN}, kterého chceme dosáhnout. Dle doporučení AN2317 vypočítáme hodnotu kondenzátoru:

$$C_{42} = \frac{I_{\mathrm{IN}}}{\Delta U_{\mathrm{DD}} \cdot f} = \frac{0{,}0157}{0{,}1 \cdot 50} = 3140 \ \ (\mu\mathrm{F}) , \tag{2.12}$$

kde ΔU_{DD} … je maximální zvlnění (100 mV),

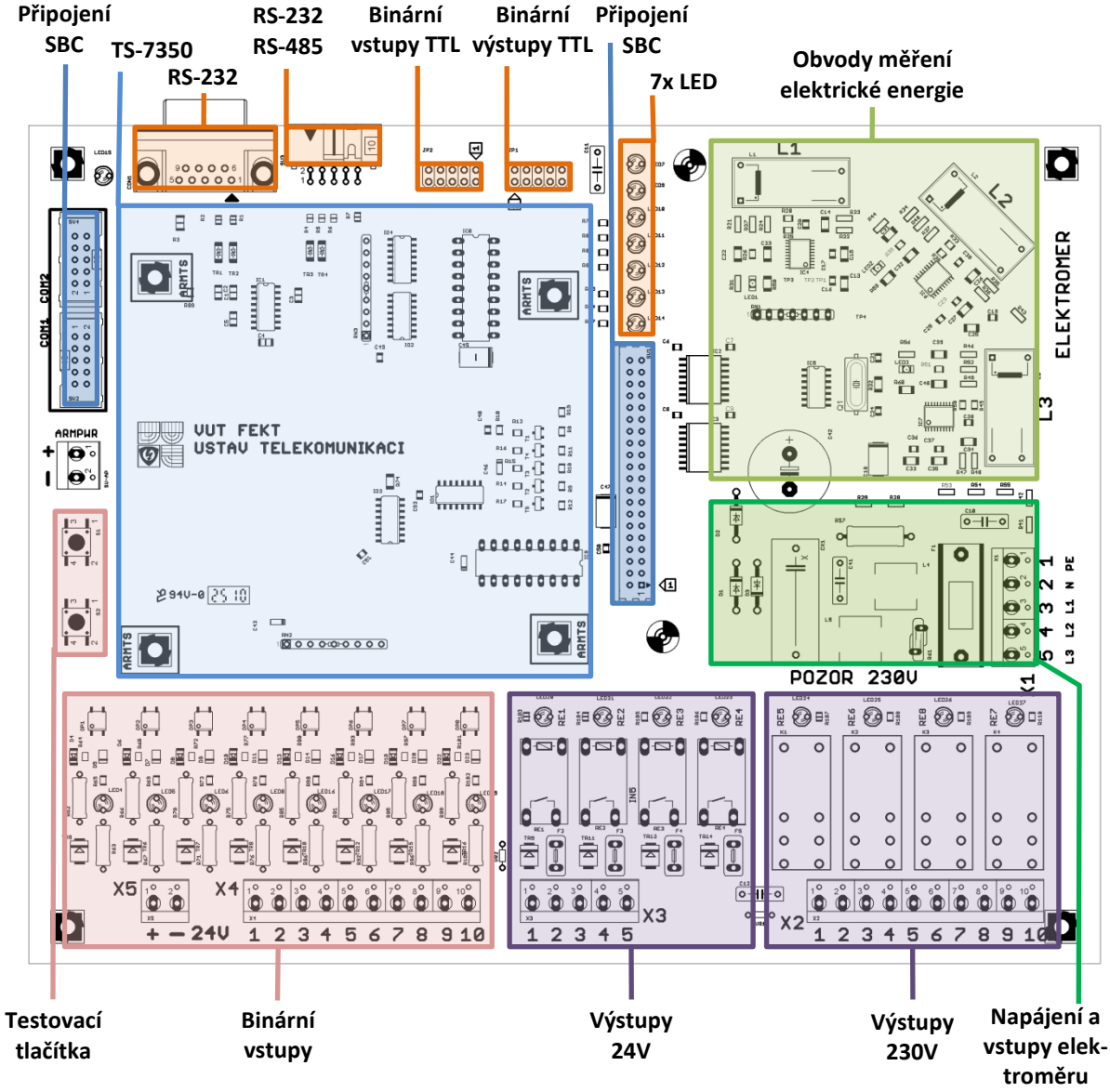
f … frekvence napájecího napětí (Hz).

Pro lepší filtraci byla v zapojení použita přiměřeně vyšší hodnota C_{42} = 4700 μF.

2.3 Návrh desky plošných spojů

Na základě schématického návrhu, byla vytvořena oboustranná, prokovená deska plošných spojů. Při návrhu byla respektována základní pravidla návrhu DPS [14] a opatření k zajištění elektromagnetické kompatibility (EMC) dle [15]. Na obrázku 2.9 je možné vidět rozmístění součástek, které bylo zvoleno s důrazem na dostatečné mezery mezi jednotlivými galvanicky oddělenými částmi zařízení. Deska SBC TS-7350 se nachází nad částí, kterou sama napájí, ostatní galvanicky oddělené obvody se nachází okolo.

Výkresová dokumentace je dostupná v příloze D, avšak pozor: Obrazy DPS zde uvedené nejsou v měřítku 1:1. Při výrobě je třeba vždy používat přímo zdrojové soubory z programu Eagle nebo Gerber a Exelon data, které jsou elektronickou přílohou!



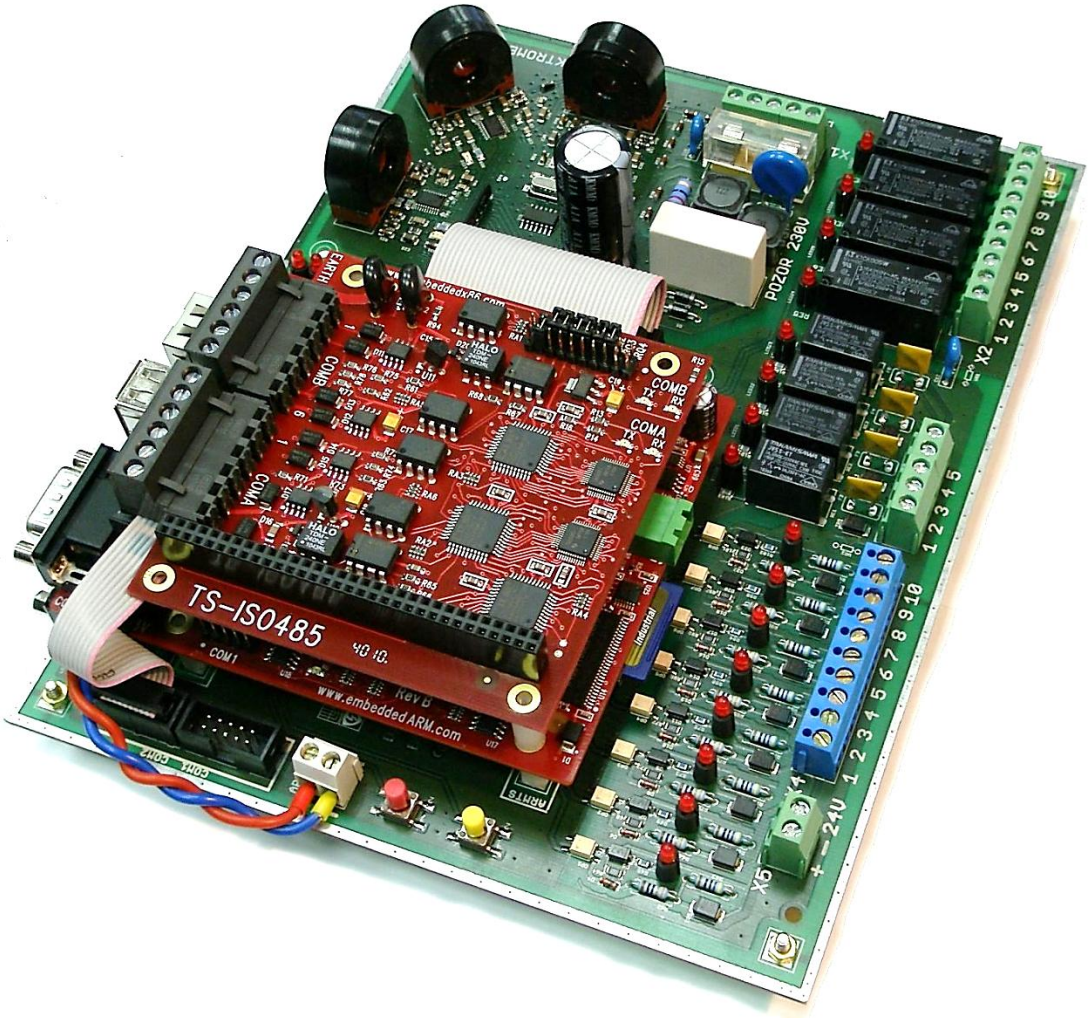
Obr. 2.9: Rozšiřující deska ovládacího systému

2.3.1 Osazení a oživení DPS

Při osazování funkčního vzorku byly nejprve do pájecí pasty ručně osazeny všechny SMD součástky ze strany TOP a zapájeny přetavením, poté byly osazeny vývodové součástky a zapájeny na pájecí vlně bezolovnatou slitinou typu SAC. Další drobné součástky a strana BOTTOM, byly dopájeny ručním pájedlem. Fotografie osazené desky je na obr. 2.10.

Oživení začalo vnějšími binárními vstupy. Na konektoru X_5 bylo postupně zvyšováno napětí až na hodnotu 24V s omezením proudu do cca 20mA, při tom by kontrolován odebíraný proud, který v klidovém stavu (kdy žádný vstup není aktivní) nesmí překročit 5mA. Následně byl na jednotlivé vstupy přiveden signál GND a opět kontrolován odebíraný proud cca 20mA/vstup. Obdobným způsobem se oživila část napájená 5V z SBC.

Napájecí zdroj elektroměru byl oživen za pomocí oddělovacího autotransformátoru. Napětí bylo postupně zvyšováno až na maximální přípustnou hodnotu 275V, přitom byla kontrolována hodna V_{DD}, která nesmí překročit 6V. Po oživení zdroje byl na testovacím bodu TP_4 zkontrolován kmitočet měřicích obvodů F_{OSC} = 4,194430 MHz.



Obr. 2.10: Prototyp ovládacího systému včetně SBC

3 Programové vybavení

3.1 GNU/Linux distribuce Debian

Základem programového vybavení, zvolené hardwarové platformy použitého mikropočítače, je operační systém GNU/Linux [16], [17] distribuce Debian (dále jen Linux). Tvoří ho jádro 2.6.21-ts kompilované pro architekturu ARM9 se speciálními úpravami výrobce SBC, které usnadňují obsluhu některých HW periferií bez přítomnosti ovladače. Přístup k nim je pak dovolen přímím zápisem do paměti, což je možné díky instalaci zařízení */dev/mem.* Nad vrstvou jádra operačního systému jsou provozovány veškeré další služby systému. Řada z nich je již v distribuci výrobce, nebo je lze přes balíčkovací systém *apt-get* doinstalovat.

3.1.1 Tvorba instalačního disku

Hlavním paměťovým médiem pro uložení veškerého obsahu je SD karta o maximální velikosti 2GB (SBC nepodporuje standard SDHC). Instalace systému na kartu se prování pouhým zápisem RAW obrazu v programu **dd**, nejedná se tedy o klasickou instalaci v pravém slova smyslu, ale o obnovení předem připraveného systému. Zdroj předem připravených image souborů systému GNU/Linux různých verzí, je FTP server výrobce: *ftp.embeddedarm.com*.

Struktura oddílů ve výchozích instalacích však nebývá optimální. Na základě svých zkušeností s provozem paměťových karet v embedded zařízeních, byla v programu **dd** a **GParted** vytvořena nová struktura, kterou je možné vidět na obr. 3.1.

Partition	File System	Size	Used	Unused	Flags
unallocated	unallocated	7.84 MiB	---	---	
/dev/sdb1	jfs	996.22 MiB	4.48 MiB	991.73 MiB	
unallocated	unallocated	15.69 MiB	---	---	
/dev/sdb2	unknown	2.50 MiB	---	---	
unallocated	unallocated	13.19 MiB	---	---	
/dev/sdb3	ext2	1.50 MiB	1.29 MiB	213.00 KiB	
unallocated	unallocated	14.19 MiB	---	---	
/dev/sdb4	jfs	713.83 MiB	445.77 MiB	268.06 MiB	
unallocated	unallocated	196.11 MiB	---	---	

Obr. 3.1: Struktura oddílů na SD kartě *(z programu GParted)*

Volba oddílů v embedded zařízeních má svá specifika, daná především konečným počtem R/W operací na paměťové kartě. Z toho důvodu například nepoužíváme odkládací oddíl /swap na kartě, ale virtuálně jej mountujeme do paměti.

Použitá karta */dev/tssdcarda* obsahuje 4 oddíly, každý oddíl pro jiný účel:

- */dev/sdb1:* 996MB – JFS oddíl pro uživatelská data, databáze, pro častý přepis.
- */dev/sdb2:* 2,5MB – RAW obraz jádra Linux 2.6.21, (pouze pro čtení).
- */dev/sdb3:* 1,5MB – EXT2 oddíl slouží jako zdrojový obraz pro ramdisk (*/initrd = initial ramdisk)*, který jádro používá po zavedení jako svůj kořenový systém, než se spustí plnohodnotný Debian.
- */dev/sdb4:* 713MB – JFS oddíl obsahující kompletní systém OS Debian.

Nealokovaný prostor mezi oddíly je ponechán zcela záměrně. Slouží v případech, kdy zapisujeme/přepisujeme jednotlivé oddíly samostatně, ty mohou pocházet z různých zdrojů a nemusí mít přesně shodnou velikost s velikostí původního oddílu. Pokud by mezi oddíly neexistoval dostatečný nealokovaný prostor, přepis počátku dalšího oddílu by v něm zničil všechna data. Volné místo na konci je ponecháno kvůli různé velikosti SD karet od různých výrobců. Rozdíly mezi deklarovanou a skutečnou velikostí mohou činit desítky MB, podle toho kolik prostoru výrobce vyčlenil na náhradu vadných buněk. Zápis obrazu na kartu může vypadat třeba takto:

```
# dd bs=512 if=<zdrojový soubor>.dd of=/dev/tssdcarda
```

Verze použitého obrazu je založena na verzi *2gbsd-sep1710.dd* ze dne:
17. října 2010.

Pro oddíly na pracovních svazcích */dev/sdb1* a */dev/sdb4* byl vybrán moderní žurnálovací souborový systém JFS vyvinutý firmou IBM pod licencí GPL.

Důvody pro jeho výběr jsou především:

A) Žurnálování/logování diskových operací s garantovanou konzistencí dat.
B) Rychlé zotavení souborového systému po havárii (díky žurnálovým zápisům). Což je v podstatě po každém vypnutí, protože nepředpokládáme, že bude zařízení vypínáno standardním příkazem *shutdown –h now*.

3.1.2 Životnost SD karty v systému

Na trhu existuje rozsáhlá nabídka SD karet, v různých cenových a typových kategoriích, kdy výrobce udává životnost karty v řádech 10 000 zápisů (technologie MLC) až 100 000 zápisů (technologie SLC) na buňku. Budeme-li se řídit tímto tvrzením (bez ohledu na technické pozadí problému) je zapotřebí omezit zápisy souborového systému do stále stejného místa. K rovnoměrnému rozložení mazacích operací na čipu se využívá strategie *static wear-leveling* [19], která se ukazuje jako zatím optimální z hlediska rovnoměrného využití paměťových buněk a tím i životnosti Flash disků.

Pro naše použití byla vybrána SD karta: Transcend 2GB Ultra Industrial SD s technologií SLC. Typ TS2GSD80I. Udávaná MTBF: 2 500 000 hodin (cca 285 let). Bude-li při garantovaných 100 000 zápisech, přepisován kompletní obsah média 10× denně, odhaduje výrobce karty její životnost na cca 27 let.

3.1.3 Spuštění systému Debian

Start zařízení probíhá v několika krocích (A-F). Po ukončení posledního kroku, resp. zastavení logovacích služeb je proces spuštění plného operačního systému dokončen.

Během procesu spouštění dochází k vykonávání řady skriptů, zpravidla se jedná o inicializační skripty pro spuštění modulů systému. Tyto skripty byly pro ovládací systém upraveny s úmyslem zajistit maximální efektivitu, robustnost a rychlost při startu. Předem se počítá se selháním hlavního i pracovního oddílu Debianu. Jádro a inicializační ramdisk (tedy */dev/sdb2 a /dev/sdb3* „ukryté“ uprostřed SD karty) však musí zůstat funkční. V takovém případě budou fungovat všechny kroky do bodu **E** a to včetně aplikace ovládacího systému, která běží nad jádrem. Základní úkony Linuxového stroje při spouštění je možné shrnout jako posloupnost kroků:

A) Inicializace hardware a spuštění zavaděče systému boot sektoru.
B) Nahrání obrazu jádra z obrazu */dev/sdb2* do operační paměti.
C) Spuštění jádra s minimem funkcí.
D) Dekomprese obrazu */dev/sdb3,* do paměti a vytvoření kořenové složky */initrd*. Od této chvíli se veškeré souborové operace dočasně provádí nad ramdiskem. Sekvence končí bodem F – spuštěním plného Debianu. Pro práci je k dispozici busybox (balík malých programů) připravený výrobcem.
E) Spuštění inicializačního skriptu *linuxrc*, který ukazuje na *linuxrc-maco*.
    a. Inicializace telnet serveru (pro servisní režim),
    b. kontrola konzistence diskových oddílů po předchozím vypnutí,

c. namountování diskových oddílů */dev/sdb1* , */dev/sdb4,*
d. inicializace rozhraní (Ethernet, PC/104, konzole a dalších),
e. inicializace netfiltru (firewallu),
f. spuštění dropbear (SSH), ukončení telnet serveru,
g. spuštění aplikace ovládacího systému vytvořeného v kapitole 3.2!

F) Následují operace nutné ke spuštění plné distribuce Debian. Kořenová složka je nyní přesunuta na */dev/sdb4* a je povolen zápis.
Start Debianu…
a. Spuštění skriptů */usr/sbin/chroot* , */sbin/init, /etc/inittab,*
b. start plánovače, paměťového manažera a dalších,
c. inicializace */swap* do paměti,
d. nastavení systémového času hodin z obvodů RTC,
e. reinicializace Ethernet rozhraní,
f. kontrola souborového systému (změna na read-write),
g. spuštění runlevel: 2 scriptu:
i. Start *syslog* deamona pro monitorování startu systému,
ii. start *kernellog* deamona pro monitorování zavádění jádra,
iii. start *init.d*,
iv. start periodického plánovače *crond*,
v. start webového serveru Apache2,
vi. zastavení *syslog* a kernel log deamonů (systém běží).

3.1.4 Konfigurace základních služeb

Abychom mohli Linuxový systém využívat pro potřeby ovládacího systému, je nutné provést nastavení dvou základních programů.

A) SSH server

SSH server **dropbear** je díky svým malým nárokům na přidělené výpočetní prostředky s oblibou využíván v embedded systémech. Slouží pro připojení zabezpečené konzola a prostřednictvím protokolu SCP k přenosu souborů. Po prvním spuštění je nezbytné vygenerovat nový veřejný klíč, příkazem:

```
# dropbearkey -t rsa -f /etc/dropbear/dropbear_rsa_host_key
```

B) Apache2 server

Konfigurační možnosti webového serveru Apahe2 jsou dosti široké [20], pro funkční a zabezpečené webové prostředí (HTTPS) ovládacího systému je zapotřebí provést řadu nastavení. Chceme-li aby byl přístup umožněn výhradně pomocí zabezpečeného protokolu https a nikoliv protokolu http (80), je nezbytné nastavit pravidlo přepisující všechny HTTP požadavky, na požadavky HTTPS na portu 443. Toto umožňuje mj. modul **mod_rewrite**. Ten je možno nastavit přímo v *httpd.conf* přidáním následujících direktiv.

```
<IfModule mod_rewrite.c>
      RewriteEngine on
      RewriteCond %{SERVER_PORT} !^443$
      RewriteRule ^.*$ https://%{SERVER_NAME}%{REQUEST_URI} [L,R]
</IfModule>
```

Dalšími dílčími kroky v postupu jsou:

1) Vygenerovat klíč a certifikát pro vlastní certifikační autoritu:

```
# openssl req $@ -new -x509 -days 3650 -nodes -out
/etc/apache2/ssl/apache-crt.pem -keyout
/etc/apache2/ssl/apache.key
```

2) Omezit přístupová práva k soukromému klíči CA:

```
# chmod 600 /etc/apache2/ssl/apache.key
```

3) V souboru */etc/apache2/sites-available/default* změnit hodnoty virtual host:

```
<VirtualHost *:443>
      SSLEngine on
      SSLCertificateFile /etc/apache2/ssl/apache-crt.pem
      SSLCertificateKeyFile /etc/apache2/ssl/apache.key
```

4) Upravit soubor */etc/apache2/ports.conf,* změnit řádek na:

```
Listen 443
```

5) Aktivovat virtual host a **mod_ssl** pomocí příkazu: # `a2enmod ssl`

Pozn.: Vzhledem k tomu, že certifikát byl podepsán vlastní certifikační autoritou, která je pro prohlížeče nedůvěryhodná, bude prohlížeč upozorňovat na problém s nedůvěryhodným certifikátem. Řešením je certifikát dočasně přijmout, nebo přidat vydávající CA do důvěryhodných certifikačních autorit.

3.2 Tvorba ovládacího programu

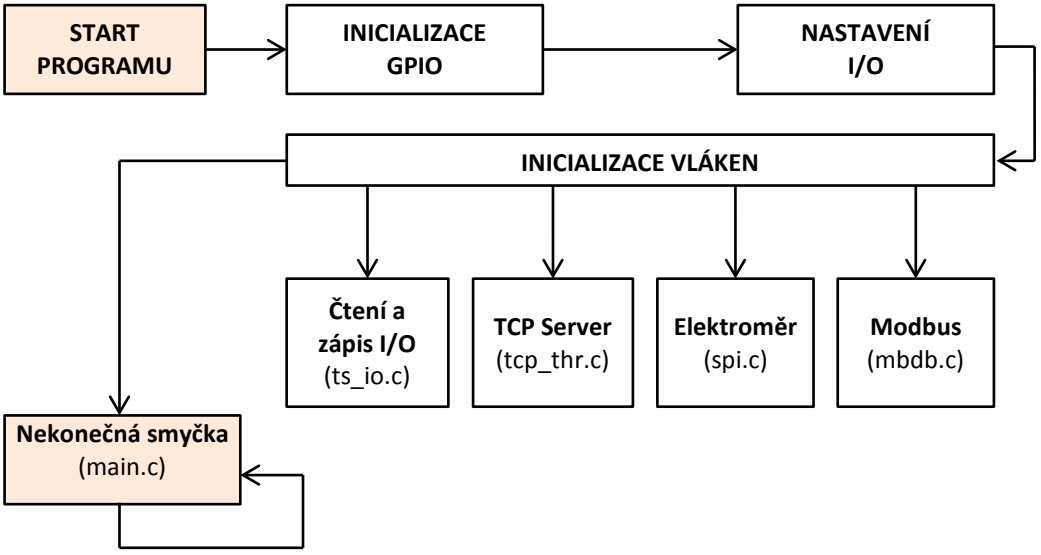
Při tvorbě vlastního programu provádějícího obsluhu hardwaru, uchovávání stavů a jejich zpřístupňování webovému serveru byl zvolen procedurální přístup, tj. data jsou uchovávána ve formě globálních nebo lokálních proměnných a metody a funkce nejsou s daty svázány. To odpovídá principům a programování v jazyce C. Vzhledem k rozsahu a obsahu řešení je tento přístup také optimální z pohledu efektivity implementačního času (na rozdíl od zvažovaného objektově orientovaného přístupu).

3.2.1 Struktura navrženého software

Program je rozdělen do čtyř základních modulů, které jsou provázány globálními proměnnými (strukturami jazyka C), které obsahují data vyměňovaná mezi jednotlivými částmi. Tyto části jsou tvořeny:

- IO Control *(ts_io.c)* – ovládání a monitorování binárních I/O.
- Modbus Control *(mbdb.c)* – rozhraní protokolu Modbus.
- SPI Control *(spi.c)* – komunikace po sběrnici SPI a čtení dat z elektroměru.
- TCP Routines *(tcp_thr.c)* – TCP server pro komunikaci s webovým serverem.

Vlastní běh programu je obsažen v metodě *main()*, mimo výše uvedené moduly. Tato metoda je vstupním bodem celého programu. Schématické zobrazení jejího běhu obsahuje obr. 3.2.



Obr. 3.2: Diagram částí ovládacího software

V metodě *main()* jsou v první řadě zpracovány argumenty z příkazové řádky předané programu systémem v parametrech *argc* a *argv*. Další běh programu je řízen načtenými hodnotami. Modifikátory programu jsou:

- *-d* pro běh jako daemon (na pozadí),
- *-v* pro nastavení výpisu (diagnostických) informací do okna konzole,
- *-h* pro vypsání nápovědy do okna konzole.

Dalším krokem je, má-li program běžet na pozadí – rozdvojení procesu pomocí speciálního systémového volání *fork()*. Toto volání vytvoří dva identické procesy, z nichž ten původní, který není procesem běžícím na pozadí, je ukončen. Vyděděný proces je ten, jemuž je vráceno ID procesu s hodnotou 0; tento proces dále pokračuje prováděním inicializačních úkonů pro rozběh programu a je jediným, který tvoří aplikaci daemona.

Výpis kódu 4.3: main.c – spuštění aplikace na pozadí

```
pid_t pid, sid;                    // deklarace ID pro proces
pid = fork();                      // rozdvojeni procesu
      if (pid < 0) {
            exit(EXIT_FAILURE);    // ukončení při chybě rozdvojení
      }
      if (pid > 0) {
                  exit(EXIT_SUCCESS);    // ukončení procesu rodiče
      }
// dále pokračuje pouze vyděděný proces jako deamon
// (služba na pozadí)
```

Posledním krokem při rozběhu programu je inicializace kritických sekcí chránících sdílené struktury pro výměnu dat mezi vlákny proti kolizním situacím násobného přístupu (*mutex* – mutual exclusion, vzájemné vyloučení přístupu). A dále vytvoření vláken v rámci aktuálního procesu aplikace – každý z modulů obsahuje funkci tvaru

```
void * nazev_funkce(void * parametr); // deklarace vlákna
```

která je výkonnou rutinou vlákna daného modulu. Odkazy na tyto funkce jsou předány systémovým voláním pro vytvoření samostatného vlákna, čímž dojde k automatickému započetí jejich vykonání.

3.2.2 Modul ovládání IO Controls

Tento modul je rozdělen do částí *TS7350* a *TS_IO* jimž odpovídají shodně pojmenované zdrojové soubory řešení. Část *TS7350* obsahuje funkce pro samostatnou modifikaci jednotlivých GPIO pinů procesoru a počáteční inicializaci ukazatelů na I/O registry. Obecně je přístup k požadovaným bitovým hodnotám implementovaným v jednotlivých funkcích možné shrnout jako posloupnost kroků:

A) nalezení bázové adresy registru (definovány v hlavičce *TS7350.h*),
B) přístup k paměti registru,
C) bitový posun dle požadované pozice,
D) modifikace / vrácení hodnoty.

Část TS_IO obsahuje funkce pro práci se „stínovým" registrem. V tomto registru jsou udržovány kopie hodnot/stavů jednotlivých hardwarových linek.

Funkce

```
void ChainUpdate(void);
```

z části *TS_IO* provádí aktualizaci hodnot ve stínovém registru podle skutečných stavů vstupů a nastavení výstupů podle hodnot ve stínovém registru. Využívá při tom funkcí pro obsluhu jednotlivých I/O procesoru z části *ts7350.c* a implementuje proces práce s vstupy a výstupy jehož popis byl proveden v odstavci 2.2.1. V této části jsou také definovány struktury pro výměnu informací s ostatními vlákny a funkce pro jejich inicializaci.

Funkce, která slouží jako vstupní bod výkonné rutiny vlákna, pracuje tak, že v pravidelných intervalech kopíruje hodnoty ze struktur pro výměnu dat s ostatními vlákny do stínového registru a naopak. Místa přístupu ke strukturám pro výměnu dat jsou ve všech modulech chráněna kritickou sekcí.

3.2.3 Modul Modbus Control

Protokol **Modbus** patří v průmyslové automatizaci k nejrozšířenějším komunikačním standardům a může tak být využit pro komunikaci ovládacího systému z řadou zařízení (např. PLC, různé regulátory, atd.). Tento modul implementuje otevřený komunikační protokol **Modbus RTU/ASCII** [22] pomocí knihovny *libmodbus*.

Knihovna definuje strukturu zprávy a kódy funkcí, z nichž v modulu jsou využity funkce:

- **`Mb_open_device()`**… pro nastavení odpovídajícího USART a otevření spoje.
- **`Mb_master()`** … pro odeslání dotazu na zařízení typu *slave*,
- **`Mb_slave()`** … pro přijetí dotazu od zařízení typu *master,*
- **`Mb_read_coils()`** … pro čtení jednoho nebo více bitů,
- **`Mb_read_registers()`** … pro čtení jednoho nebo více 16b registrů,
- **`Mb_write_coils()`** … pro zápis jednoho nebo více bitů,
- **`Mb_write_registers()`** … pro zápis jednoho nebo více 16b registrů,
- **`Mb_close_device()`**… pro řádné ukončení otevřeného komunikačního zařízení v systému.

Formáty rámců a významy zpráv, které se odesílají a přijímají tímto modulem (přes knihovnu *libmodbus*), jsou definovány veřejně dostupným protokolem DIXELL [23] společnosti Dixell S.p.A., předního evropského producenta PLC a regulátorů. Pomocí této definice lze k ovládací aplikaci připojit PLC pro řízení tepelného čerpadla.

Vykonávání obsluhy komunikace je prováděno ve zvláštním vlákně procesu aplikace, otevřeném v metodě *main().* Přístup k hodnotám (čtení) požadovaných bitových registrů, je možné shrnout jako posloupnost kroků:

A) otevření a navázání spojení se zařízením typu slave,
B) sestavení rámce dotazu, a odeslání požadavku funkcí *Mb_master()*,
C) přijetí odpovědi a dekódování zprávy (stavu),
D) přenesení zprávy (stavu) do globální struktury programu.

3.2.4 Modul SPI Control

Modul tvoří dvě části *SPI* a *ENGM* jimž odpovídají shodně pojmenované zdrojové soubory řešení. Část *SPI* obsahuje funkce pro obsluhu SPI sběrnice procesoru EP9302 a ENGM pak metody pro čtení a zápis hodnot registrů měřicích obvodů, dle protokolu výrobce [13].

3.2.5 Modul TCP Routines

Většina vlastní funkčnosti spojené s tímto modulem (předávání dat klientům pomocí TCP socketu) je obsažena ve vlastní rutině pro běh vlákna:

```
void * tcp_thr(void * param);
```

Pozn.: Socket je označením aplikačního programového rozhraní pro práci s TCP/IP protokoly a zároveň označením koncového bodu komunikačního spojení pomocí TCP/IP. Se sockety lze pracovat pomocí standardních knihoven operačního systému. V zásadě existují dvě možnosti obsluhy socketů: v tzv. blokujícím nebo neblokujícím režimu.

Ve zvoleném implementačním přístupu je využit blokující režim socketů, který zajišťuje synchronní provedení všech operací, tj. volání funkcí pro práci se sockety blokuje další postup ve výkonu vlákna do doby dokončení operace spojené s volanou funkcí. Tento přístup je z implementačního pohledu výhodný kvůli zřetelnosti sekvence použitých funkcí, zjednodušení řešení a efektivnosti využití času CPU.

Obsah podprogramu lze rozdělit na dvě části: nastavení a inicializace socketu a obsluha přijatých připojení. V části inicializace se provádí:

A) vytvoření socketu typu daného makrem AF_INET (TCP spojová služba),
B) navázání socketu na lokální TCP port (20000),
C) vytvoření fronty požadavků na spojení.

V části obsluhy spojení se provádí:

A) přijetí spojení od klienta (vytvoření socketu pro komunikaci),
B) nastavení maximální doby příjmu dat od klienta,
C) příjem a odeslání dat podle specifikace protokolu v příloze E.

Část obsluhy spojení se provádí opakovaně ve smyčce pro každé přijaté spojení přímo ve vláknu serveru – tj. server v jeden okamžik obsluhuje pouze jednoho klienta (jedná se o tzv. jednovláknový server). Toto řešení lze použít, neboť obsluha klienta bude pouze krátkým spojením z generujícího skriptu PHP webového serveru. V případě přijetí dvou požadavků (téměř) současně bude jeden z nich obsloužen s nepatrným zpožděním daným rychlostí odezvy obsluhy socketů.

Nastavení maximální doby příjmu dat od klienta se provádí z bezpečnostních důvodů pro případ, kdy např. vlivem chyby operačního systému dojde ke stavu ukončení běhu klienta před řádným uzavřením spojení (socketu). V takovém případě se uplatní nastavený time-out (15s) a spojení je uzavřeno. Okamžité uzavření přijatého spojení se provádí také vždy, když autorizační mechanismus definovaný v protokolu odepře autorizaci klienta.

4 Webové rozhraní

Webové stránky tvoří základní dorozumívací prostředek mezi uživatelem a připojeným zařízením. Musí poskytovat informace dostatečně rychle a v dostatečné kvalitě, aby mohl uživatel objektivně zhodnotit aktuální stav připojeného zařízení.

Webové rozhraní je definováno prostřednictvím značkovacího jazyka HTML a kaskádových stylů (CSS), resp. layoutů. Požadovaná funkčnost webového rozhraní je dosažena kombinací technologií PHP, JavaScript a Ajax popsané v literatuře [21]. Při pokusu o přístup k webovému rozhraní je nejprve vyžadováno ověření identity uživatele – autentizace.

4.1 Autentizace uživatele

Autentizaci lze považovat za jednu ze základních funkcí při implementaci přístupových práv. K ověření identity uživatele, který se pokouší přihlásit na webový server, existuje několik v praxi často používaných technik.

4.1.1 HTTP Autentizace

HTTP autentizace je přímou součástí HTTP protokolu. Implementovány jsou dvě metody Basic a Digest [18]. Použit je obecný princip sdíleného tajemství, kdy toto tajemství – přihlašovací heslo je známé oběma stranám. Server vyzve klienta k ověření, obdrží jeho dopověď, kterou porovná s očekávaným výsledkem. Nalezne-li shodu, odešle požadovanou stránku, v opačném případě spojení ukončí.

Největším nedostatkem autentizace Basic je skutečnost, že je jméno a heslo předáváno přímo v textové podobě. Následně putují počítačovou sítí ve standardním TCP/IP paketu bez jakéhokoliv zabezpečení a mohou tak být snadno odposlechnuty. Ve snaze vyřešit tento nedostatek byla vyvinuta technika Digest Access Authentication RFC 2617.

4.1.2 Digest Access Authentication

Je rozšíření základní HTTP autentizace implementované v protokolu HTTP/1.1, funguje na stejném principu, avšak klient zasílá MD5 kontrolní součet (hash) vytvořený ze jména, hesla a řetězce, který obdržel od serveru v rámci požadavku na autentizaci.

Autentizace probíhá v následujících krocích:

Krok 1) server odešle v hlavičce požadavek na autentifikaci

```
WWW-Authenticate: Digest realm="Private", domain
="/localhost", nonce="Ay1yLzIwMDIgMzoyNjoyNCBWTW", algo-
rithm=MD5, qop="auth,auth-int"
```

Klíčové slovo **nonce** je pseudonáhodné číslo vygenerované serverem, které klient použije při sestavení odpovědi. **Qop** znamená kvalitu ochrany (Quality of protection), parametr **auth** – pouze ověření, **auth-int** se rozumí ověření plus ochrana integrity.

Krok 2) klient provede výpočet odpovědi

```
HA1 = MD5(A1) = MD5(username:realm:password)
```

Odpověď pro parametr **auth**

```
HA2 = MD5(A2) = MD5(metod:digestURI)
```

Odpověď pro parametr **auth-int**

```
HA2 = MD5(A2) = MD5(metod:digestURI:MD5(entityBody))
response = MD5(HA1:nonce:nonceCount:clientNonce:qop:HA2)
```

a odešle požadavek

```
GET /index.html HTTP/1.1
Host: localhost
Authorization: Digest username="guest",
realm="Private", qop="auth-int", algorithm="MD5",
uri="/index.html", nonce="Ay1yLzIwMDIgMzoyNjoyNCBWTW",
nc=00000001, cnonce="cc7e5189556f949675b070dab8e5277a",
response="adc10c6115a14e2811a433c19a143782"
```

Krok 3) Po přijetí odpovědi má server stejné informace jako klient a může provést ověření stejným výpočtem.

Požadavky klient čísluje v parametru **nc** (**nonceCount**), toto číslo může být použito pro ochranu proti opakovaným útokům. **cnonce** (**clientNonce**) je klientem náhodně generovaná hodnota pro zamlžení MD5.

Důležitá zlepšení algoritmu Digest jsou:

- Heslo není nikdy přenášeno jako text,
- server může kontrolovat hodnotu **nc**, a použít ji proti opakovaným útokům,

- server může omezit platnost hodnoty `nonce` po omezenou dobu, jen pro konkrétního klienta, nebo jen pro určitou žádost.

Kompromitace algoritmu MD5 nalezením kolize během několika sekund, má výrazný vliv na bezpečnost autentizace Digest.

4.1.3 Autentizace pomocí asymetrické šifry

Protokol HTTP nezajišťuje důvěrnost přenosu ani integritu spojení. Proto se obvykle používá transportní vrstva, která zajistí důvěrnost/integritu spojení a taktéž autentizaci klienta/serveru pomocí asymetrické kryptografie. V současné době je v případech kdy záleží na utajení přenášení informace nejčastěji používán transportní protokol SSL/TLS. Princip SSL spočívá ve vytvoření zabezpečeného kanálu mezi klientem (prohlížeč) a serverem (Apache2). SSL umožňuje bezpečnou výměnu informací při inicializaci TCP/IP spojení, při kterém si klient a server ustavují konkrétní bezpečnostní parametry pro následný provoz a vzájemně se autentizují certifikáty. Po ustavení těchto parametrů je veškerá komunikace (http požadavky i http odpovědi) plně šifrována a to včetně URL, které klient požaduje, webových formulářů apod.

4.1.4 Autentizace osobním certifikátem

Použití SSL umožňuje autentizaci založenou na osobním certifikátu (PKI), který klient obdrží od certifikační autority. S tímto certifikátem se pak může autentizovat v rámci skupiny serverů, které této certifikační autoritě důvěřují. Za možnou nevýhodu lze považovat to, že na rozdíl od jména/hesla, které je možno si lehce zapamatovat, certifikát nemusí mít uživatel vždy po ruce.

4.2 Autentizace uživatele ve webovém rozhraní

Při výběru autentizační techniky vyvstal problém jakou použít, aby byla dostatečně robustní a bezpečná, a zároveň příliš neobtěžovala uživatele. Basic a Digest jsou příliš slabé a snadno prolomitelné, autorizace osobním certifikátem zase vyžaduje po uživateli nosit s sebou elektronický soubor s certifikátem.

Z těchto důvodů bylo přihlašování uživatelů do webového rozhraní založeno na vlastním řešení pomocí PHP skriptu s využitím metody *session* a hashovací funkce SHA-1.

Integrita spojení je zajištěna vytvořením zabezpečeného kanálu mezi klientem a serverem na portu 443 (HTTPS).

4.2.1 Popis provedení autentizační procedury

V případě, že klient požádá o zobrazení kterékoliv stránky aplikace, provede se nejprve kontrola, zda je v *session* uložena informace o jeho přihlášení. Toto je zajištěno tak, že v záhlaví každého skriptu aplikace se nejprve provede inicializace *session* pomocí PHP funkce *session_start()* a dále se provede test pomocí PHP funkce *isset()* na existenci proměnné s uloženým loginem v rámci „superglobální" proměnně **$_SESSION**.

Pokud tomu tak není, provede se pomocí PHP funkce *header()* přesměrování na přihlašovací skript *(prihlaseni.php),* který obsluhuje přihlašování a odhlašování uživatelů a další zpracování původního skriptu se zastaví pomocí funkce *exit().* Současně se odesílá také proměnná s informací o aktuální adrese stránky, kterou klient požadoval. Tyto údaje se získají ze „superglobální" proměnné **$_SERVER**. Skript pak tuto proměnnou využije po úspěšném přihlášení k automatickému přesměrování na stránku, kterou uživatel původně požadoval.

Výpis kódu 4.1: index.php - test přihlášení a zajištění přesměrování na přihlašovací skript.

```
session_start();
if (!isset($_SESSION["maco"])){
      header("Location: http://".$_SERVER['SERVER_NAME'].
      "/prihlaseni.php?url=".$_SERVER['REQUEST_URI']);
      exit;
}
```

Pokud nejsou autentizačnímu skriptu v rámci superglobální proměnné **$_POST** odeslány přihlašovací údaje (což skript ověřuje testem pomocí PHP funkce *isset(),* zobrazí se přihlašovací HTML formulář, který obsahuje 2 vstupní formulářová pole pro zadání uživatelského jména/loginu (typ text) a hesla (typ password).

Pro zvýšení bezpečnosti přihlašování je v autentizačním procesu pro účely odesílání zadaného hesla mezi klientem a serverem využívána technika výzva – odpověď (request - response). Server vygeneruje náhodnou výzvu (sůl), kterou odešle i klientovi. Klient potom místo toho, aby odeslal heslo serveru přímo v plain textu, tak na své straně nejprve k tomuto heslu připojí uvedenou výzvu a serveru potom odešle jenom otisk tohoto spojení. Server na své straně provede totožný proces, a pokud se výsledky shodují, tak uživatele přihlásí, jinak ho odmítne. Bezpečnost tohoto řešení spočívá v tom, že server

generuje novou výzvu při každém zobrazení přihlašovacího formuláře. Takže i když se útočníkovi podaří zachytit odpověď klienta, tak mu k ničemu neposlouží, protože stejnou výzvu už server nikdy nepošle.

Konkrétní implementace této techniky je v rámci aplikace provedena tak, že při zobrazení přihlašovacího formuláře nejprve skript vygeneruje s využitím funkce *mt_rand()* náhodnou výzvu (sůl) v délce 4 znaků, kterou jednak uloží na serveru do souboru v definovaném (pomocí souboru *config.php*) a na úrovni práv operačního systému zabezpečeném adresáři a také ji zapíše uživateli do skrytého pole přihlašovacího formuláře.

Výpis kódu 4.2: prihlaseni.php – skript pro generování náhodné výzvy (soli)

```
function salt() {

      $kod ='';
      $index = 0;
      for($i=0; $i<4; $i++) {

            $predesly_znak = $index;
            $znaky = 'ABCDEFGHJKMNPQRUVWXUZabcdef-
            ghjkmnpqruvwxuz123456789';
            $index = mt_rand(0,strlen($znaky)-1);
            $tento_znak = substr($znaky,$index,1);
            $kod .= $tento_znak;
      }
      return $kod;
}
```

Události formuláře *„onsubmit“* je přiřazena obslužná javascriptová funkce, která nejprve vytvoří otisk zadaného hesla pomocí algoritmu SHA-1 (aplikace ve všech částech z důvodu bezpečnosti pracuje pouze s SHA-1 otisky hesel nikoliv přímo heslem jako takovým) a následně pomocí hashovací funkce HMAC (HMAC–SHA-1) se vypočítá otisk hesla a výzvy (soli), který se následně společně s loginem uživatele odešle na server. Z důvodu že funkce algoritmy SHA-1 a HMAC nejsou v Javascriptu přímo implementovány, je pro účely této aplikace využita externí javascriptová knihovna.

Výpis kódu 4.3: prihlaseni.php – funkce pro sestavení otisku odesílaného hesla

```
function sha1form(f) {

      f['heslo_hmac'].value =
      Crypto.HMAC(Crypto.SHA1,Crypto.SHA1(f['heslo'].value),
      f['salt'].value);
      f['heslo'].disabled = true;
      f.submit();
      f['heslo'].disabled = false;
      return false;
}
```

Pozn: Celé řešení je koncipováno tak, aby bylo funkční i v případě, kdy uživatel nemá v prohlížeči spuštěn Javascript (různé internetové kavárny, mobilní telefony, atd.). V tomto případě se pak na server odesílá login společně s heslem v plain textu. Na zvýšené bezpečnostní riziko je uživatel upozorněn pomocí zobrazení výstrahy nad přihlašovacím formulářem, ve kterém je mu doporučeno Javascript povolit!

Na straně serveru je nejprve ověřeno, zda uživatel odeslal všechny povinné údaje (login i heslo). Pokud ne, je o této skutečnosti zobrazeno chybové hlášení a uživateli se zobrazí nový přihlašovací formulář s předvyplněným loginem.

V dalším kroku se ověřuje, zda je zaslané heslo HMAC otisk nebo pouze plain text (z důvodu vypnutého Javascriptu na straně klienta). Podle této skutečnosti je případně upraveno do požadované podoby. Následně je ze souboru načtena výzva (sůl), která byla klientovi odeslána. Protože výzva již byla tímto klientem použita, provede přihlašovací skript pomocí funkce *ulink()* odstranění tohoto dočasného souboru.

Skript dále provede porovnání zadaného loginu a otisku hesla uživatele se správnými údaji. Uživatelské loginy a hesla (otisky hesel) se pro účely autentizace nacházejí v souboru **macopasswd**. Cesta k tomuto soboru, který je zabezpečen na úrovni práv operačního systému, se definuje v souboru *config.php*.

Navržená struktura souboru **macopaswd**

```
LOGIN;HESLO;<SHA-1 hash>;<Konstantní sůl>;<SHA-1 s konstantní solí>
```

K získání údajů z tohoto souboru jsou využívány standardní PHP funkce pro práci se souborem – *fopen(), fgets(), fclose().* Pro zpracování samotného seznamu, ve kterém jsou jednotlivé položky oddělené jasně definovaným znakem „**;**“ je použita PHP funkce *explode().* Nejprve je zjišťováno, zda se v tomto souboru nachází uživatel s tímto loginem a následně je u tohoto uživatele porovnáván otisk hesla.

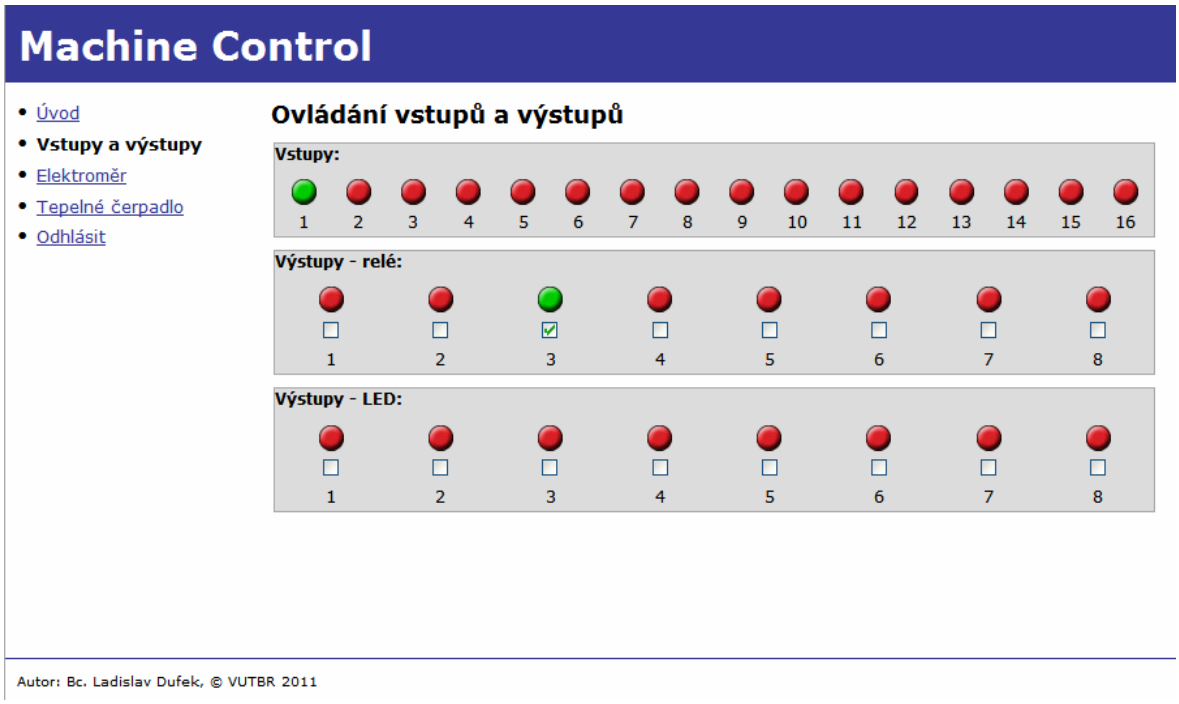
V případě, že porovnání loginu (uživatel se zadaným loginem se v souboru nenachází) nebo hesla (otisk hesla spočítaný na straně serveru je odlišný od zaslaného otisku) selže, je o této skutečnosti klient informován chybovým hlášením, proces autentizace se ukončí a uživateli se zobrazí nový přihlašovací formulář s předvyplněným loginem. V opačném případě se autentizace dokončí zapsáním příslušného hodnoty (loginu) do session a pomocí funkce *header()* se provede přesměrování na stránku, ze které byl proces autentizace vyvolán.

4.3 Popis webové aplikace

Struktura uživatelského okna je poměrně jednoduchá a jednotná pro všechny části aplikace a je definována v souboru *index.php*. Prostředí stránky na obr 4.1 je rozděleno na hlavičku, samotný obsah a patičku. Část se samotným obsahem je potom rozdělena na levou část s menu pro přepínání mezi jednotlivými podstránkami a pravou část se samotným výstupem příslušné stránky. Pro odkazy na jednotlivé podstránky je využívána technologie mod_rewrite (modul Apache2).

Webová aplikace je rozdělena do 4 základních podstránek:

- úvod (*welcome.php*) – úvodní (uvítací) stránka,
- vstupy a výstupy (*iocontrol.php*) – ovládání a monitorování bin. I/O,
- elektroměr (*energymt.php*) – monitorování stavu elektroměru,
- tepelné čerpadlo (*heatpump.php*) – monitorování stavu PLC.



Obr. 4.1: Obrazovka webové aplikace – stránka vstupů a výstupů

Obslužné skripty pro jednotlivé stránky se nachází v adresáři **include** – *energymt.php*, *heatpump.php*, *iocontrol.php*, *welcome.php* a *menu.php*. Jednotlivé skripty jsou proti zneužití útočníkem zabezpečeny v úvodu kontrolou, zda je klient přihlášen. Toto je zajištěno tak, že se nejprve provede inicializace session pomocí funkce *session_start()* a dále se provede test pomocí funkce *isset()* na existenci proměnné s uloženým loginem

v rámci „superglobální“ proměnně **$_SESSION**. Pokud tomu tak není, zpracování skriptu se zastaví pomocí funkce *exit.*

Ukázka zabezpečení skriptu před neoprávněným přístupem:

```
session_start();
if (!isset($_SESSION["maco"]["login"])) exit;
```

V první části obslužného skriptu dojde k naplnění pole **$vychoziHodnoty**, kde názvy jednotlivých klíčů pole odpovídají názvům jednotlivých položek v externí aplikaci ovládající hardware (názvy položek jsou uvedeny v příloze E). Naplnění pole je zajištěno pomocí PHP skriptu *getDataVychozi.php* (v adresáři **function**).

Další část skriptu již zajistí zobrazení příslušné tabulky se zobrazenými výchozími hodnotami. Samozřejmostí je ošetření situací, kdy se výše uvedenému skriptu nepodaří příslušná data získat.

4.3.1 Synchronizace hodnot webové aplikace

Pro pravidelnou synchronizaci hodnot zobrazených na stránce, s externí aplikací, je v příslušném skriptu vždy vytvořena javascriptová funkce *updateStranka(),* která využívá technologii Ajax (konkrétně objekt **XMLHttpRequest**) ke komunikaci s TCP serverem aplikace ovládající hardware. Samotnou komunikaci potom zajišťuje PHP skript *getData.php* v adresáři **function**.

Prvotní vyvolání funkce *updateStranka()* je zajištěno událostí stránky **onload** a následně je vyvolávána v pravidelných intervalech pomocí základní javascriptové funkce *setTimeout().* Časový interval tohoto opětovného vyvolávání v milisekudnách je možné pro jednotlivé podstránky odděleně nastavit v souboru *config.php*.

Funkce *procesRequest()* následně zajistí přepsání jednotlivých údajů bez překreslování celé stránky. V případě, že má webový klient **Javascript** zakázaný, zobrazí se mu pouze na začátku výchozí aktuální data, ale nedochází již k pravidelné aktualizaci ve stanoveném časovém intervalu.

4.3.2 Změny nastavení webové aplikace

V rámci podstránky vstupy a výstupy *(iocontrol)* je možné kromě zobrazování aktuálních dat (proměněných) provádět na webové stránce změny nastavení, resp. přímé ovládání hardware.

K tomuto účelu jsou tyto výstupy opatřeny formulářovými prvky typu checkbox (zatržítko), které mají k události **`onClick`** přiřazenou javascriptovou funkci *nastavVystup()*, která využívá technologii Ajax (opět objekt **XMLHttpRequest**) ke komunikaci s TCP serverem v ovládací aplikaci. Funkci je předáván název proměnné a podle stavu checkboxu (true/false) se odešle buď hodnota 1, nebo 0. Samotnou komunikaci se serverem externí aplikace zajišťuje potom PHP skript *setData.php* v adresáři **function**.

4.4 Komunikace mezi webovým rozhraním a ovládací aplikací

Komunikace probíhá prostřednictvím TCP protokolu na vnitřní smyčce 127.0.0.1 na portu 20000. Díky absolutnímu oddělení webového rozhraní od ovládací aplikace prostřednictvím proprietárního protokolu (příloha E) je možné spustit webové rozhraní odkudkoli, třeba na vzdáleném serveru.

Adresa serveru (**hostname**) a číslo portu se definuje v souboru *config.php*. Před **hostname** je možné předřadit *ssl://* nebo *tls://* pro použití SSL nebo TLS spojení na vzdálený stroj přes TCP/IP. Veškerá komunikace je textově orientovaná a kromě navázání spojení neprobíhá žádné potvrzování zpráv. Potvrzením přijetí je samotná odpověď na dotaz, popř. odpověď na stav s již novou hodnotou. Pro účely komunikace se v protokolu využívají dva základní příkazy:

- **`get`** – pro získání hodnot a
- **`set`** – pro nastavení hodnot.

Veškerá komunikace typu client-server mezi webovým rozhraním a externí aplikací ovládající hardware je zajištěna pomocí PHP skriptů:

- *getDataVychozi.php* – získání výchozích dat po načtení podstránky,
- *getData.php* – získávání dat z externí aplikace a
- *setData.php* – odesílání změny nastavení.

Tyto jednotlivé skripty jsou proti zneužití útočníkem opět zabezpečeny v úvodu kontrolou, zda je klient přihlášen. Toto je zajištěno tím, že se nejprve provede inicializace session pomocí funkce *session_start()* a test pomoci funkce *isset()* na existenci proměnné s uloženým loginem v rámci „superglobální“ proměnně **`$_SESSION`**. Pokud tomu tak není, zpracování skriptu se zastaví pomocí funkce *exit().*

4.4.1 Otevření socketu

Spojení mezi klientem a serverem se inicializuje voláním PHP funkce *fsockopen()* s parametry **hostname** a číslo portu. Pro případ, že volání selže, obsahuje tato funkce dva nepovinné parametry *errno* a *errstr* (vracející chybovou úroveň) a volitelný parametr *timeout* při případ, že by daný host nebyl dostupný.

Výpis kódu 4.4: index.php - kódu *fsockopen()*:

```
$socket = @fsockopen ($server["hostname"], $server["port"], $errno,
$errstr,5);
```

4.4.2 Navázání komunikace

Před odesláním samotného příkazu **`get`** nebo **`set`** je nutné nejprve navázat komunikaci s ovládací aplikací. Pro tento účel je nutné na socket odeslat příkaz *„maco“* a hash hesla přihlášeného uživatele (s konstantní solí), které je nutné získat ze souboru **macopasswd**. K odeslání příkazu slouží PHP funkce *fputs().* Server v ovládací aplikaci prověří hash hesla s konstantní solí a v případě, že souhlasí s heslem v jeho databázi, potvrdí spojení odesláním řetězce *„maco“.* V opačném případě spojení ukončí.

Pozn: Zde by bylo bezesporu daleko bezpečnější generovat pro každý dotaz unikátní sůl, jak je tomu při autentizaci uživatele, avšak to by znamenalo daleko vyšší režijní zátěž stroje výpočtem hešů pro každý dotaz (interval cca 100ms). Vzhledem k tomu, že zde je komunikace provozována na lokální smyčce, je nepravděpodobné, že bude odposlechnuta útočníkem.

V případě, že bylo navázání komunikace úspěšné, aplikace vrátila odpověď *„maco“,* dojde k přečtení obsahu socketu (odpovědi). Pro účely dalšího zpracování webovou aplikací slouží PHP funkce *fgets()*.

Výpis kódu 4.5: index.php - kódu pro dotaz a přijmutí odpovědi:

```
fputs ($socket, "maco".$hashHesla);
$odpoved = fgets ($socket,5);
```

4.4.3 Odeslání příkazu get/set na socket

K odeslání konkrétního příkazu **set** nebo **get** slouží opět PHP funkce *fputs().* V případě příkazu **get** se příkaz odesílá ve tvaru

get=stránka;,

kde stránka je *welcome*, *iocontrol*, *energymt* nebo *heatpump*. V případě příkazu **set** ve tvaru

set&délka_řetězce&proměnná=hodnota;,

kde se délka řetězce zjistí pomocí PHP funkce *mb_strlen().*

Výpis kódu 4.6: index.php - dotazem na stránku a příklad s nastavením proměnné:

```
fputs ($socket,"get=".$stranka.";");

// nastavení proměnné
$retezec = $promenna."=".$hodnota.";";
fputs ($fp,"set&".mb_strlen($retezec)."&".$retezec);
```

V případě, že byl v předešlém kroku odeslán příkaz **get**, aplikace na socket zpět vrátí řetězec s požadovanými daty. K jeho získání pro účely dalšího zpracování se použije PHP funkce *fgets().*

Protože má odpověď serveru na dotaz **get** tvar

get&délka_zprávy&řetězec_s_proměnnými;

rozdělí se nejprve na jednotlivé části pomocí PHP funkce *explode()* podle znaku *(&)*. Následně se provede kontrola správného rozdělení tak, že se porovná délka zprávy uvedená v tomto řetězci s hodnotou zjištěnou pomocí PHP funkce *mb_strlen().* Řetězec s proměnnými je nyní ve tvaru

proměnná=hodnota;proměnná=hodnota;

Použije se tedy znovu funkce *explode(),* která tento řetězec rozdělí podle znaku „**;**“ na jednotlivé proměnné. Stejná funkce se použije i v závěrečné fázi, kdy se řetězce s jednotlivými proměnnými rozdělí podle znaku „**=**“ na název proměnné a hodnotu a zpracují do pole, resp. formátu XML.

4.4.4 Uzavření socketu

Po dokončení komunikace, dojde k uzavření soketu pomocí PHP funkce *fclose()*.

4.4.5 Odhlášení uživatele

Odhlášení uživatele zabezpečuje skript *prihlaseni.php* odstraněním příslušné hodnoty (loginu) ze session pomocí PHP funkce *unset().* Stejný skript zajistí také zobrazení příslušného textu o odhlášení uživatele a zobrazení nového přihlašovacího formuláře.

Tabulka 4.1: Stručný přehled jednotlivých souborů

Soubor	**Stručný popis**
.htaccess	Definice pravidel pro **mod_rewrite**.
config.php	Možnost definice parametrů aplikace (názvu aplikace, autora, adresy serveru, portu, intervaly obnovování jednotlivých podstránek, cesty k souborům s loginy a hesly, cesta k adresáři pro ukládání dočasných souborů)
index.php	Základní struktura aplikace.
prihlaseni.php	Skript zajišťující přihlášení a odhlášení uživatelů.
getData.php	Skript zajišťující komunikaci s externí aplikací ovládající hardware (získání dat).
getDataVychozi.php	Skript zajišťující komunikaci s externí aplikací ovládající hardware (prvotní získání dat).
setData.php	Skript zajišťující komunikaci s externí aplikací ovládající hardware (změna hodnot proměnných).
energymt.php heatpump.php iocontrol.php welcome.php	Obslužné skripty pro jednotlivé podstránky.
menu.php	Menu aplikace na levé straně.

5 Závěr

Cílem práce bylo navrhnout a realizovat systém pro dálkové ovládání elektronických zařízení prostřednictvím jednoduchých řídicích povelů.

Základní koncepce, stanovená v úvodu práce a její požadavky posloužily jako podklad pro vytvoření základního návrhu ovládacího systému. Po prostudování možných variant řešení s použitím různých embedded modulů a PLC, došlo k výběru jednodeskového mikropočítače založeného na platformě ARM s operačním systémem GNU/Linux.

V další části práce se na základě představené koncepce, vytvořil návrh rozšiřujícího modulu, včetně stanovení typu a počtu jeho periferií a to i s výhledem do budoucna.

Tvorba softwarového vybavení začala přípravou distribuce OS GNU/Linux, se zvláštním zaměřením na kvalitu a životnost paměťového média. Po stručném popisu spouštěcího procesu se přešlo k nastavení dvou programů, nutných pro běh ovládacího systému (Apache2 a SSH).

Pomocí volně dostupných vývojových nástrojů Eclipse a gcc vznikl v jazyce C základní program pro ovládání vstupů a výstupů. Pro složitější přístroje vybavené sériovým rozhraním doplněný o softwarový modul protokolu Modbus, který poskytne styčný základ pro ovládání a monitorování PLC DIXELL dle uvedeného komunikačního protokolu. Užitečnou doplňující informací o ovládaném zařízení bude údaj z měření spotřeby odebrané elektrické energie, který jistě přispěje ke zvýšení ekonomiky provozu sledovaného zařízení a bude tak užitečným pomocníkem uživateli.

Pro vlastní komunikaci webového prostředí a ovládací aplikace vznikl návrh proprietárního komunikačního protokolu, na bázi TCP spojení po lokální smyčce. Tímto způsobem je zajištěno oddělení webových stránek od ovládacího programu. Po prozkoumání možných autentizačních technik z doporučené literatury bylo zvoleno řešení vlastní, založené na technologii PHP a JavaScript. Důvodem je především použití bezpečnější hashovací funkce SHA-1.

Výsledkem této diplomové práce je nové speciální zařízení, přizpůsobené specifickým podmínkám a požadavkům, jehož přínosem je možnost zajistit ovládání různých elektronických výrobků přes povely webového rozhraní.

Díky otevřenosti použité platformy a dostatečné výkonové i kapacitní rezervě, může být zařízení v budoucnu dále rozšiřováno o nové hardwarové i softwarové moduly. Důležitým zlepšením by zajisté bylo zpracování analogových údajů, např. z měřidel teploty, vlhkosti, atp.

Díky integrovanému TCP serveru se rovněž nabízí možné vytvoření tzv. „tlustého klienta“, nebo připojení vzdáleného webového serveru.

Samostatnou kapitolu by vytvořila změna koncepce sytému od ovládacího k řídicímu. Doplněním potřebných algoritmů by i se stávajícím hardwarem bylo možné provedení základní reléové regulace podle stavu vstupů, podle kalendáře, nebo podle jiných předem daných pravidel.

6 Seznam literatury

[1] *Charon 1 - Modul rozhraní Ethernetu* [online]. Praha, www.hwgroup.cz 2003 [cit. 2011-05-25]. Dostupný z WWW: <http://www.hw-group.com/products/charon1/index_cz.html>.

[2] *Ethernut Software* [online]. 2008 [cit. 2010-12-15]. Dostupný z WWW: <http://www.ethernut.de/en/software/index.html>.

[3] Teco a.s. Tecomat : *Základní moduly* [online]. Kolín: 1. září 2010 [cit. 2011-04-15]. Dostupné z WWW: <http://www.tecomat.com/index.php?ID=292>.

[4] Technologic Systems Inc. *Technický list TS-7350.* [online]. July 23 2009. U.S. Arizona: July 23 2009, [cit. 2011-05-25]. Dostupné z WWW: <http://www.embeddedarm.com/products/board-detail.php?product=TS-7350>.

[5] Technologic Systems Inc. *Technický list TS-ISO485.* [online]. June 05 2009. U.S. Arizona: June 05 2009, April 22 2008, [cit. 2010-12-15]. Dostupné z WWW: <http://www.embeddedarm.com/products/board-detail.php?product=TS-ISO485>.

[6] *Technický list TLP181.* [online]. 12. listopad 2009. [cit. 2011-05-25]. Dostupné z WWW: <http://download.siliconexpert.com/pdfs/2009/12/22/14/13/35/422/tos_/auto/tlp181_en_datasheet_091112.pdf>.

[7] *Technický list ULN2803A*. [online]. 1. únor 2006. [cit. 2011-05-25]. Dostupné z WWW: <http://focus.ti.com/lit/ds/symlink/uln2803a.pdf>.

[8] *Technický list ADuM3441*. [online]. Revize C. [cit. 2011-05-25]. Dostupné z WWW: <http://www.analog.com/static/imported-files/data_sheets/ADUM3440_3441_3442.pdf>.

[9] *Technický list 74HC595*. [online]. 4. leden 1998. [cit. 2011-05-25]. Dostupné z WWW: <http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf>.

[10] *Technický list 74HC165*. [online]. Revize 03. 14. březen 2008. [cit. 2011-05-25]. Dostupné z WWW: <http://www.nxp.com/documents/data_sheet/74HC_HCT165.pdf>.

[11] *Aplikační poznámka: AN10441*. [online]. 16. červen 2007. [cit. 2011-05-25]. Dostupné z WWW: <http://www.nxp.com/documents/application_note/AN10441.pdf>.

[12] *Aplikační poznámka: AN2317*. [online]. 1. Duben 2006. [cit. 2011-05-25]. Dostupné z WWW:<http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/APPLICATION_NOTE/CD00091951.pdf>.

[13] *Technický list STPM01*. [online]. Revize 7. říjen 2010. [cit. 2011-05-25]. Dostupné z WWW: <http://www.st.com/internet/analog/product/81930.jsp>

[14] ZÁHLAVA, V.: *Metodika návrhu plošných spojů.* Vydavatelství ČVUT, Praha 2002.

[15] VACULÍKOVÁ, P., VACULÍK, E. a kolektiv: *Elektromagnetická kompatibilita elektrotechnických systému.* Grada Publishing, Praha 1998.

[16] Zpracoval kolektiv autorů. 3. vydání. *Používáme LINUX.* Brno: COMPUTER PRESS, 2003. 659 s. ISBN 80-7226-698-5.

[17] Zpracoval kolektiv autorů. 1. vydání. *LINUX kompletní příručka administrátora.* Brno: COMPUTER PRESS, 2004. 828 s. ISBN 80-722-6919-4.

[18] *Wear leveling,* www.wikipedia.org [online]. 6. únor 2011. [cit. 2011-05-25]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Wear_leveling>.

[19] *IETF Documents : RFC 2617 - HTTP Authentication: Basic and Digest Access Authentication* [online]. W3C: July 30, 1997, June 1999 [cit. 2011-05-25]. Dostupné z WWW: <http://tools.ietf.org/html/rfc2617>.

[20] ARTUR MAJ.: *Apache 2 with SSL/TLS* [online]. 02. únor 2005. [cit. 2011-05-25] Dostupné z WWW: <http://www.securityfocus.com/infocus/1820>.

[21] Lecky-Thomson Ed, Nowicki D. Steven. 1. vydání. *Programujeme profesionálně.* Praha: BEN Technická literatura, 2010. 720s. ISBN 978-80-251-3127-5.

[22] MODBUS over Serial Line. *Specification and Implementation Guide* [online]. Ver.1.02. 20.12.2006. [cit. 2011-05-25] Dostupné z WWW: <http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf>

[23] *MODBUS-RTU applied to DIXELL devices* [online]. Ver.2.6. [cit. 2011-05-25] Dostupné z WWW: <http://www.dixell.de/uploads/media/ModBUS_Protokoll_v2_6_GB.pdf>

7 Seznam použitých zkratek, veličin a symbolů

AD	Analog to Digital (analogově/číslicový)
API	Application Programming Interface
ARM9	Advanced RISC Machine generace 9
BOTTOM	Spodní strana desky plošného spoje
BPLACE	Spodní strana součástek na desce plošného spoje
CGI	Common Gateway Interface
CPU	Central Processing Unit (hlavní procesorová jednotka)
ČSN	Česká státní norma
DPS	Deska Plošného Spoje
EMI	Electromagnetic Interference (elektromagnetické rušení)
GNU	General Public License
GPIO	General Purpose Input/Output (vstupy/výstupy pro obecné použití)
HTML	HyperText Markup Language
HTTP	The Hypertext Transfer Protocol [RFC 2068]
HTTPS	The Hypertext Transfer Protocol Secure
IO	Integrovaný obvod
I/O	Input/Output
I2C	Inter-Integrated Circuit
ICMP	Internet control message protocol (internetový protokol) [RFC 792]
IP	Internet protocol (internetový protokol) [RFC 791]
ISA	Industry Standard Architecture
ISP	In-System Programming (programovací rozhraní)
JTAG	Joint Test Action Group
LED	Light-Emitting Diode (světlo vyzařující dioda)
MCU	Microcontroller Unit
MD5	Message-Digest algorithm 5 [RFC 1321]
MTBF	Mean Time Between Faliture (střední doba mezi selháním)
MLC	Multi Layer Cell (technologie přístupu k paměťové buňce)
PID	Process Identifier (identifikační číslo procesu)
PHP	Hypertext Preprocessor
PLC	Programovatelný logický automat
RFC	Request for comments
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory (paměť pouze pro čtení)
RTC	Real Time Counter
RTOS	Real-Time Operating Systém
SAC	Bezolovnatá pájecí slitina (Sn 96.5%, Ag 3.0%, Cu 0.5%)
SBC	Single Board Computer
SCK	Serial Clock (hodinový signál sériové sběrnice)
SD	Secure Digital (paměťová karta)
SDHC	Secure Digital High Density (paměťová karta vyšší kapacity >2GB)
SHA-1	Secure Hash Algorithm
SIPO	posuvný registr se sériovým vstupem a paralelním výstupem (Serial In - Paralel out)
SPI	Serial Peripheral Interface (sériové periferní rozhraní)

SSL	Security Socket Layer
SLC	Single Layer Cell (technologie přístupu k paměťové buňce)
SRAM	Static Random Access Memory
TCP/IP	Transmission Control Protocol [RFC 793] / Internet Protocol
TEA	Tiny Encryptiom Algoritm
TOP	Horní strana desky plošného spoje
TPLACE	Horní strana součástek na desce plošného spoje
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol [RFC 768]
USART	Universal Synchronous Asynchronous Receiver Transmitter
USB	Universal Serial Bus (univerzální sériová sběrnice)
PISO	posuvný registr se paralelním vstupem a sériovým výstupem (Parallel In - Serial Out)

SEZNAM PŘÍLOH

PŘÍLOHA A: OBSAH PŘILOŽENÉHO CD
PŘÍLOHA B: SCHÉMA ZAPOJENÍ
PŘÍLOHA C: FOTO VÝSLEDNÉ DPS
PŘÍLOHA D: VÝROBNÍ DOKUMENTACE
PŘÍLOHA E: KOMUNIKAČNÍ PROTOKOL ŘÍDICÍHO PROGRAMU

Příloha A – Obsah přiloženého CD

Složky:

/ Podklady pro výrobu	… podklady pro výrobu DPS (soupisky materiály, Gerber data)
/ Linux skripty	… vytvořené skripty ze systému Linux
/ Výkresy DPS	… schémata a výkresy desek v programu Eagle v 5.7.0
/ Součástky	… katalogové listy použitých součástek
/ Zdrojové kódy	… zdrojové soubory programu
/ Webové stránky	… zdrojové soubory webový stránek
/ TS-7350	… katalogové listy SBC TS-7350 *)
/ TS-ISO485	… katalogové listy přídavné desky ISO485*)
/ Image SD karty	… aktuální obraz použité SD karty *)

*) není součástí elektronické přílohy, kvůli své velikosti.

VUT BRNO

VYROBNI PODKLADY

VERZE: 10.4.2011

LIST	OBSAH
1	Uvodni strana
2	Pripojení ARM TS-7350
3	Merení L1, L2
4	Merení L3, napajení mericich obvodu
5	Posuvné registry, I/O, relé

SIGNALIZACNI LED

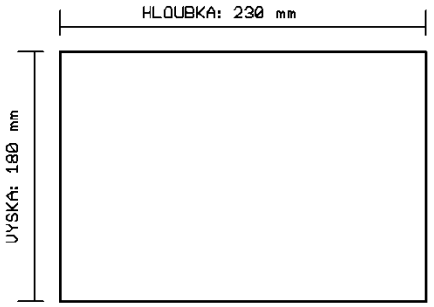
1x LED pro indikaci stavu provozu (yellow)
3x SMD LED pro indikaci mericích obvodu
7x LED pro universální pouzití
8x LED pro indikaci stavu vstupu (VST1-VST8)
4x LED pro indikaci stavu vystupu 24V (OUT1-OUT4)
4x LED pro indikaci stavu vystupu relé (OUT5-OUT8)

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

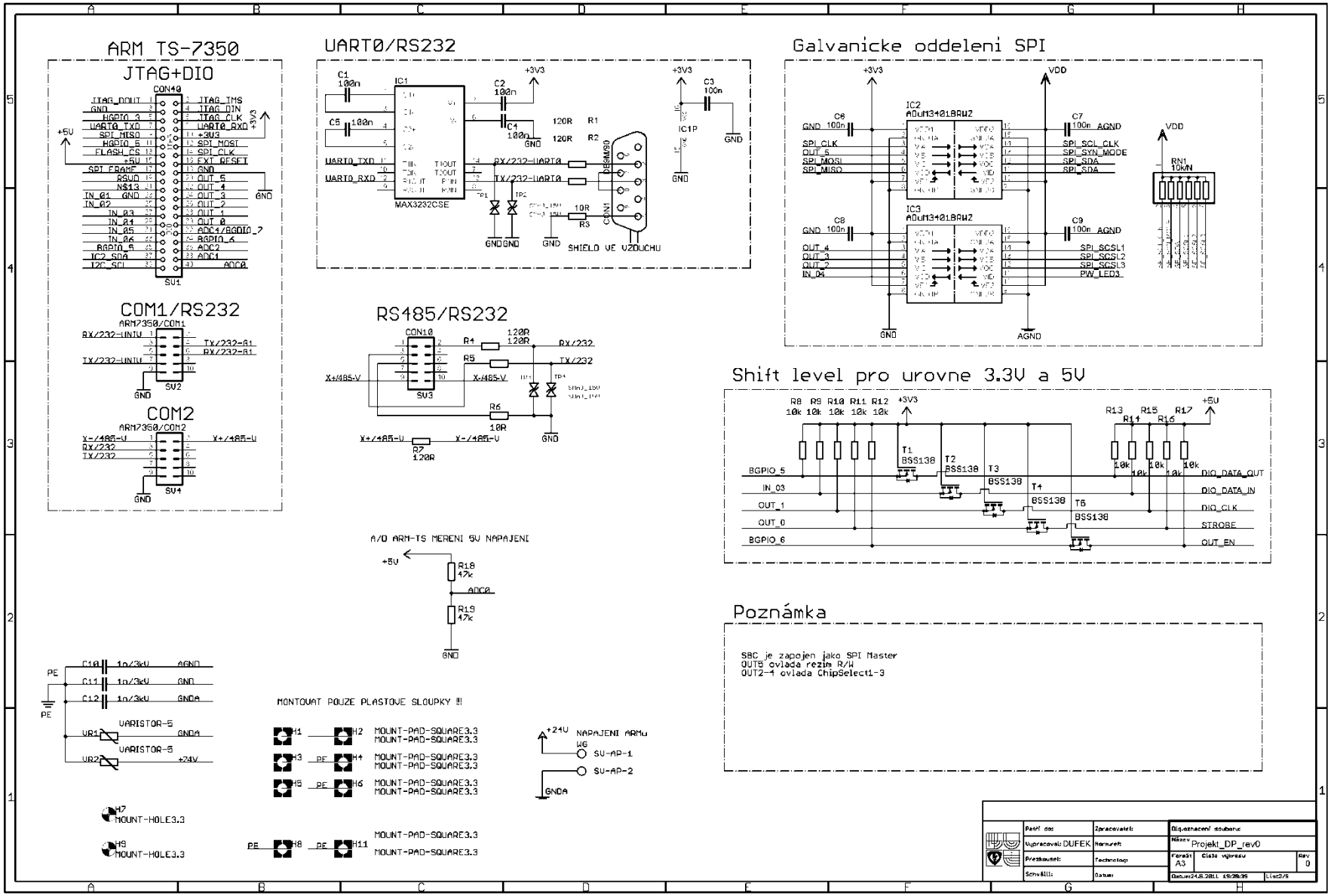
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

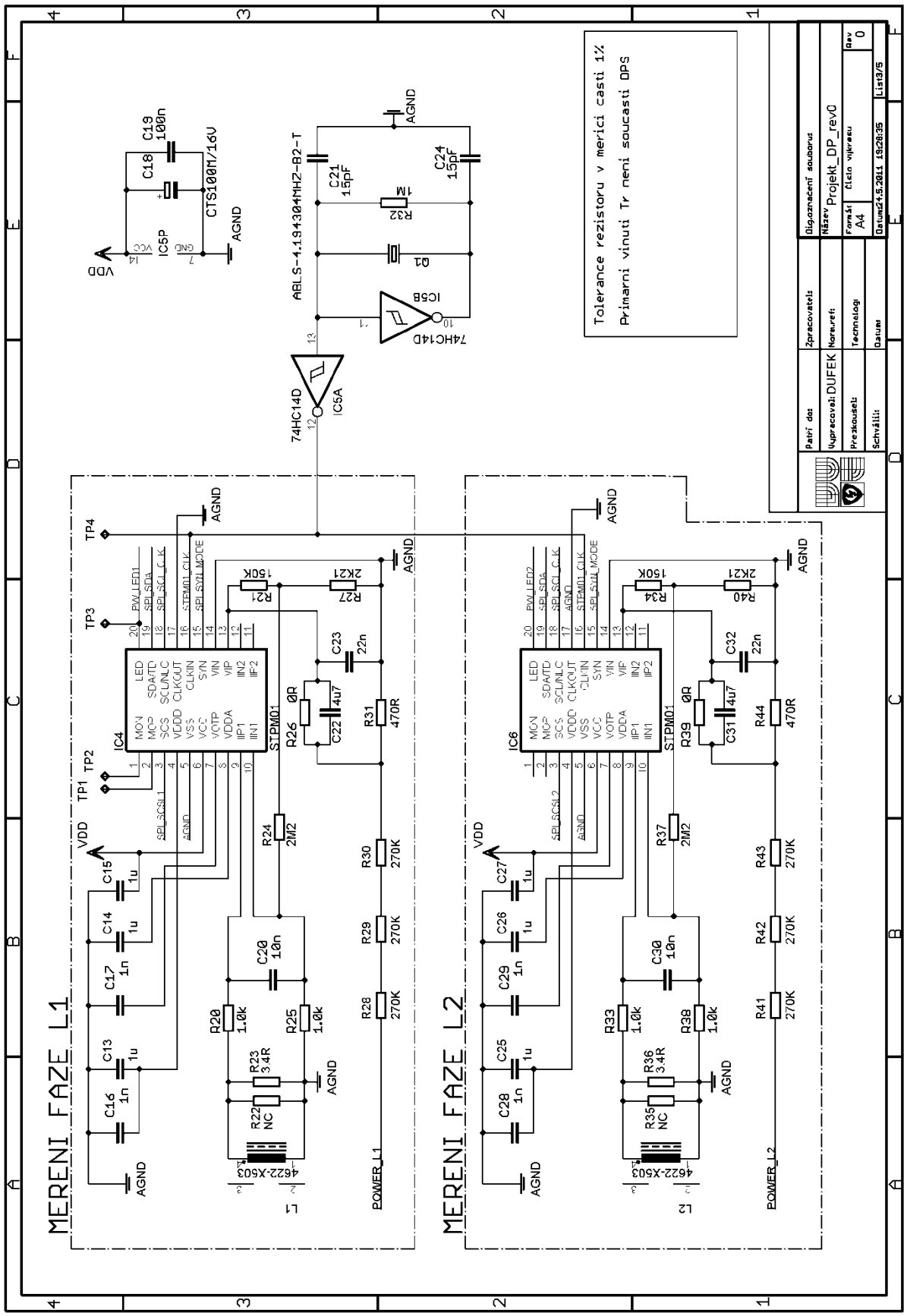
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

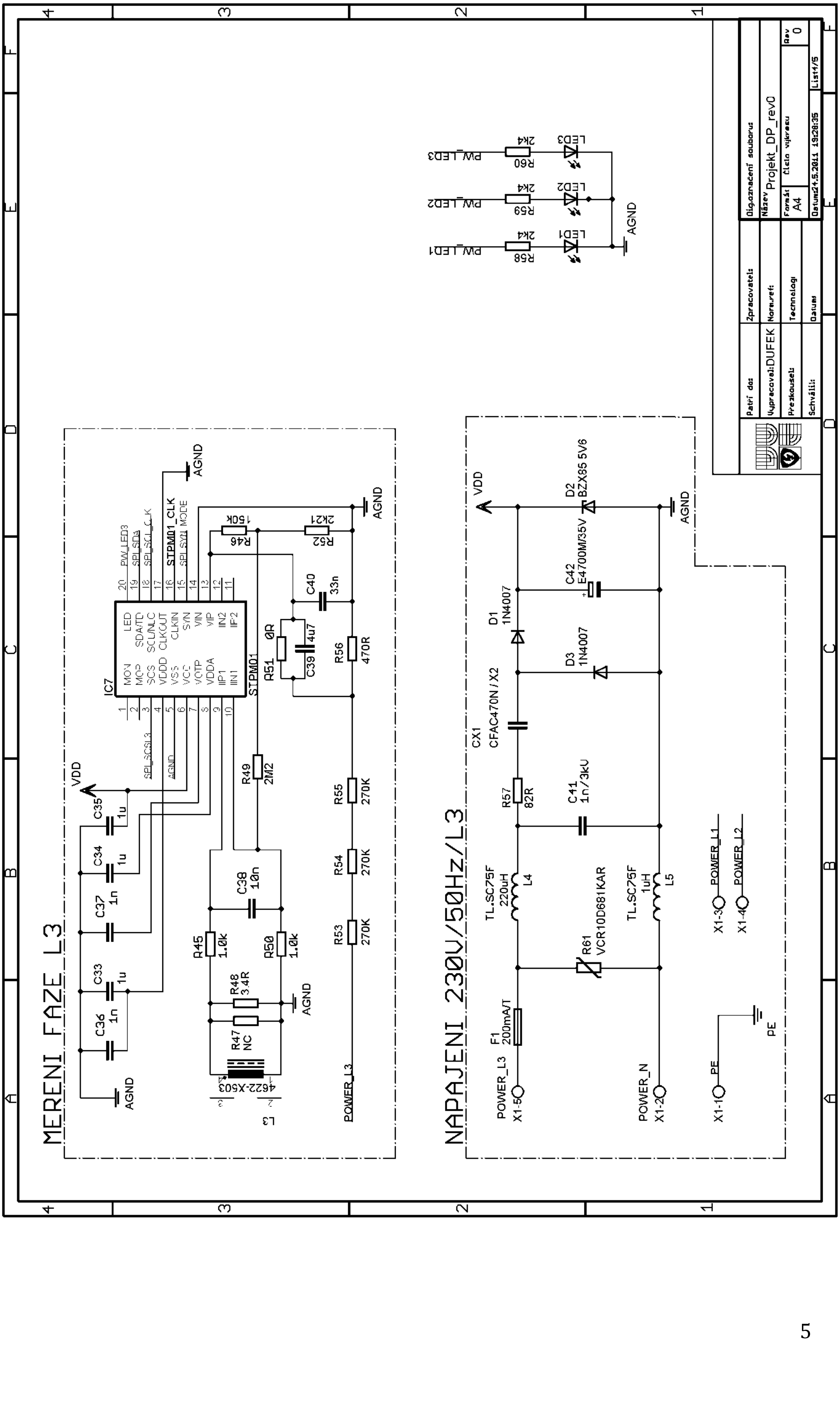
CELY PANEL PRO 1x DPS

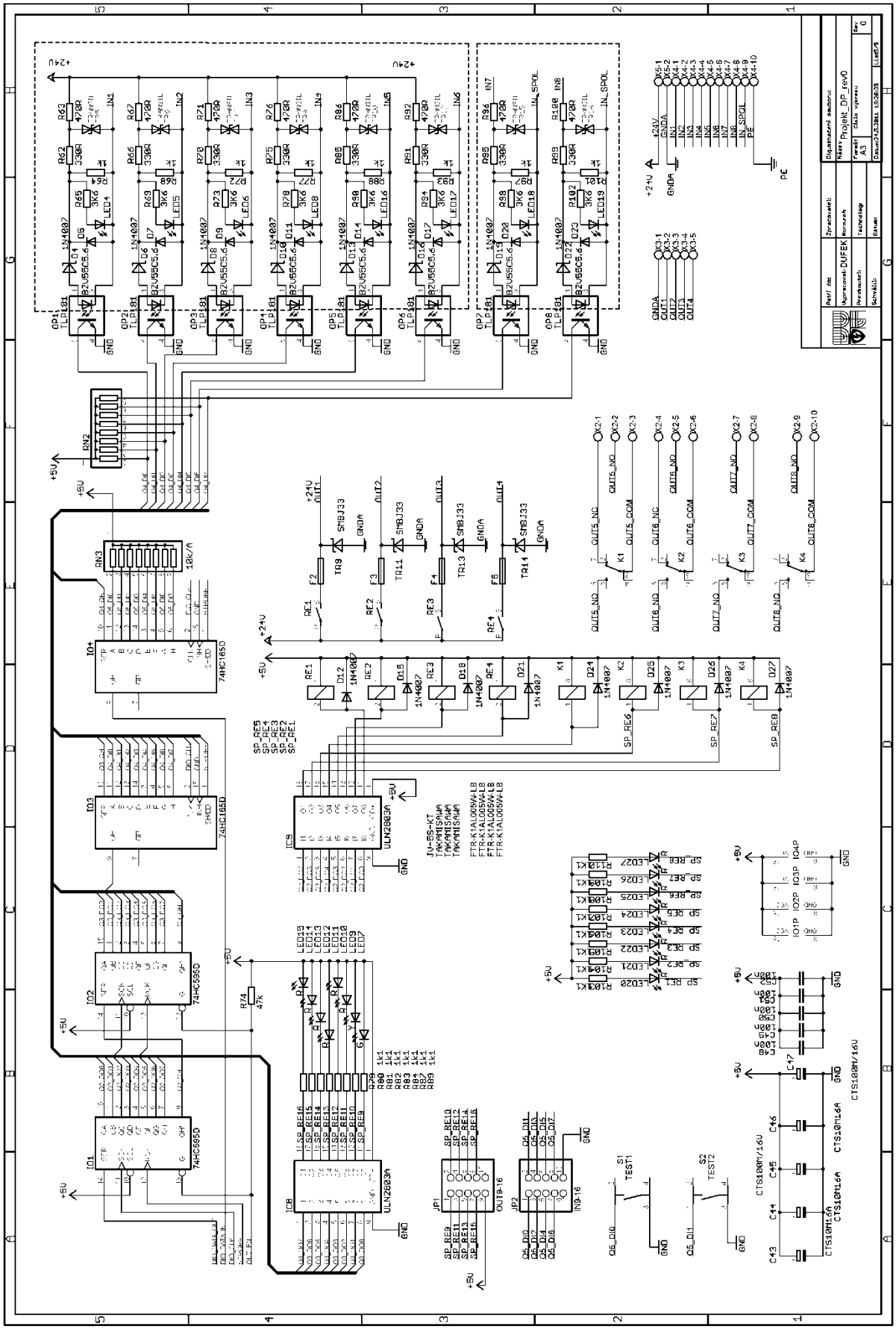


Patří do:	Zpracovatel:	Dig.označení souboru:
Vypracoval: DUFEK	Norm.ref:	Název: Projekt_DP_rev0
Přezkoušel:	Technologi	Formát: A3 / Čísla výkresu / Rev: 0
Schválil:	Datum	Datum: 24.5.2011 19:28:35 / List 1/5

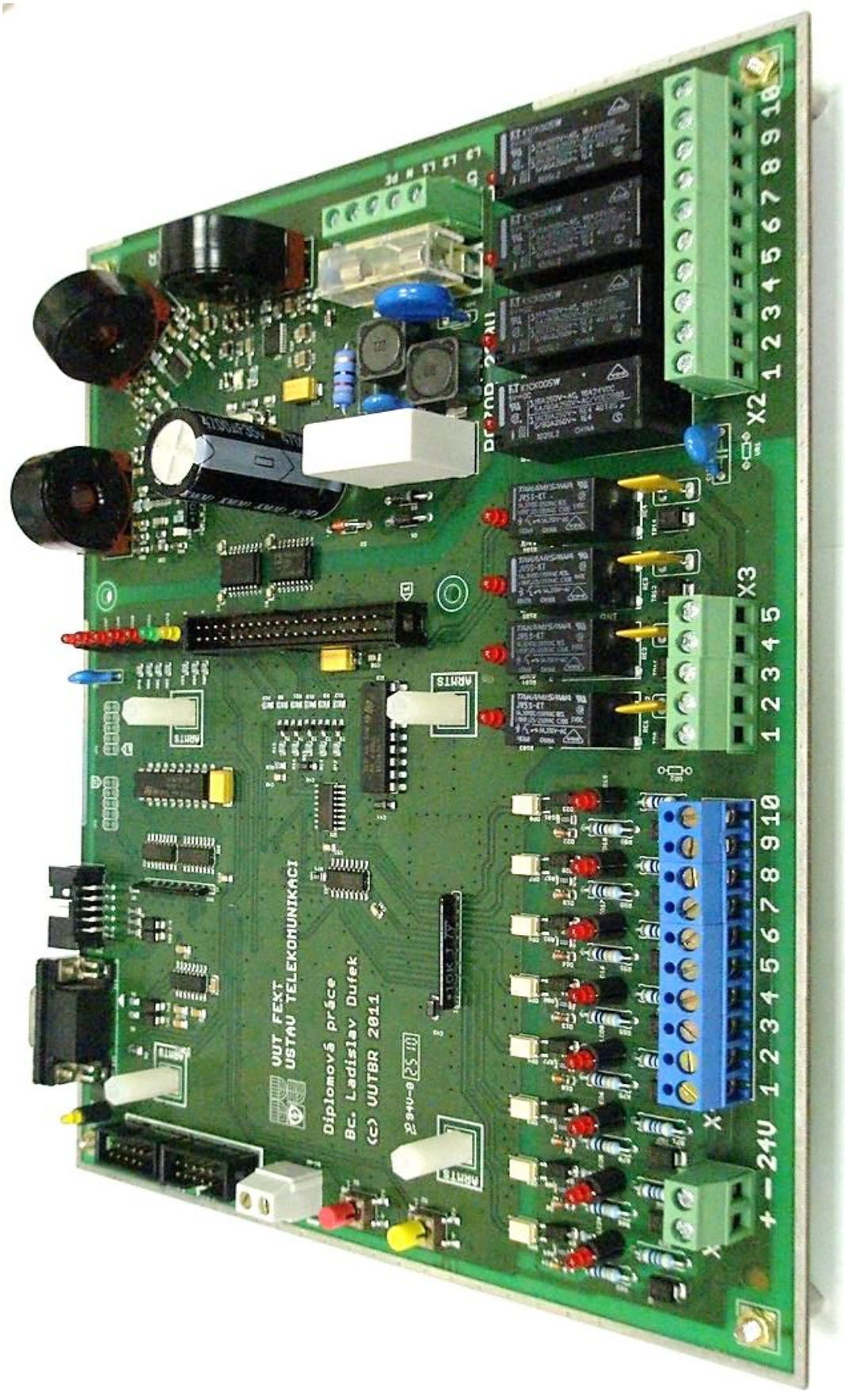

ARM TS-7350
JTAG+DIO
CON40
JTAG_TMS
JTAG_DIN
JTAG_CLK
UART0_RXD
SPI_MOST
SPI_CLK
EXT_RESET
SPI_FRAME
ADC1/AGPIO_7
AGPIO_6
ADC2
ADC1
ADC0
SV1
COM1/RS232
ARM7350/COM1
RX/232-UNIV
TX/232-UNIV
TX/232-A1
RX/232-A1
SV2
COM2
ARM7350/COM2
X-/485-V
X+/485-V
RX/232
TX/232
SV4
UART0/RS232
IC1
MAX3232CSE
C1
C2
C3
C4
C5
100n
120R
R1
R2
R3
10R
UART0_TXD
UART0_RXD
RX/232-UART0
TX/232-UART0
DSUB9
CONN
SHIELD VE VZDUCHU
RS485/RS232
CON10
R4
R5
R6
R7
SV3
Galvanicke oddeleni SPI
IC2
IC3
ADUM3401BRWZ
C6
C7
C8
C9
SPI_CLK
OUT_5
SPI_MOSI
SPI_MISO
SPI_SCL_CLK
SPI_SYN_MODE
SPI_SDA
OUT_4
OUT_3
OUT_2
IN_04
SPI_SCSL1
SPI_SCSL2
SPI_SCSL3
PW_LED3
RN1
10K/N
Shift level pro urovne 3.3V a 5V
R8 R9 R10 R11 R12
R13 R15 R16 R17
T1 BSS138
T2 BSS138
T3 BSS138
T4 BSS138
T5 BSS138
BGPIO_5
IN_03
OUT_1
OUT_0
BGPIO_6
DIO_DATA_OUT
DIO_DATA_IN
DIO_CLK
STROBE
OUT_EN
A/D ARM-TS MERENI 5V NAPAJENI
R18
R19
ADC0
Poznámka
SBC je zapojen jako SPI Master
OUT5 ovlada rezim R/W
OUT2-4 ovlada ChipSelect1-3
C10 C11 C12
VARISTOR-5
VAR1
VAR2
MONTOVAT POUZE PLASTOVE SLOUPKY !!
H1 H2 H3 H4 H5 H6 H7 H8 H9 H11
MOUNT-PAD-SQUARE3.3
MOUNT-HOLE3.3
NAPAJENI ARMu
SV-AP-1
SV-AP-2
Datum:24.5.2011 19:28:05
Projekt_DP_rev0


MERENI FAZE_L1
MERENI FAZE_L2
Tolerance rezistoru v merici casti 1%
Primarni vinuti Tr neni soucasti DPS
Navrhl: DUFEK
Projekt_DP_rev0
Datum: 24.5.2011 19:28:35
List: 3/5
Rev: 0
Format: A4
VDD
AGND
POWER_L1
POWER_L2
IC4
IC6
STPM01
74HC14D
ABLS-4.194304MHZ-B2-T
CTS100M1/16V
4622-X503
L1
L2


MERENI FAZE L3
NAPAJENI 230V/50Hz/L3
IC7
STPM01
STPM01_CLK
VDD
AGND
PW_LED1
PW_LED2
PW_LED3
LED1
LED2
LED3
R58
R59
R60
2k4
D1
1N4007
D2
BZX85 5V6
D3
1N4007
C42
E4700M/35V
CX1
CFAC470N / X2
R57
82R
C41
1n/3kV
TL.SC75F
220uH
L4
1uH
L5
R61
VCR10D681KAR
F1
200mA/T
POWER_L3
POWER_N
POWER_L1
POWER_L2
PE
X1-1
X1-2
X1-3
X1-4
X1-5
4622-X503
L3
R45
R50
1.0k
R47
NC
R48
3.4R
C38
10n
C33
C34
C35
1u
C36
C37
1n
R49
2M2
R53
R54
R55
270K
R56
470R
R51
0R
C39
4u7
C40
33n
R46
150K
R52
2K21
Označení souboru
Název
Projekt_DP_rev0
Formát
A4
Rev
0
Patří do:
Zpracovatel:
Vypracoval: DUFEK
Norm.ref:
Přezkoušel:
Technolog:
Schválil:
Datum:
List1/5



Příloha C – Foto výsledné DPS



Obr. 1.1: Fotografie výsledné DPS po osazení

Příloha D – Výrobní dokumentace

Pokyny pro výrobu DPS:

Materiál DPS: FR4

Tloušťka materiálu FR4: 1,5mm

Síla mědi: 18µm

Vrtaná DPS: ANO

Nepájivá maska: ANO

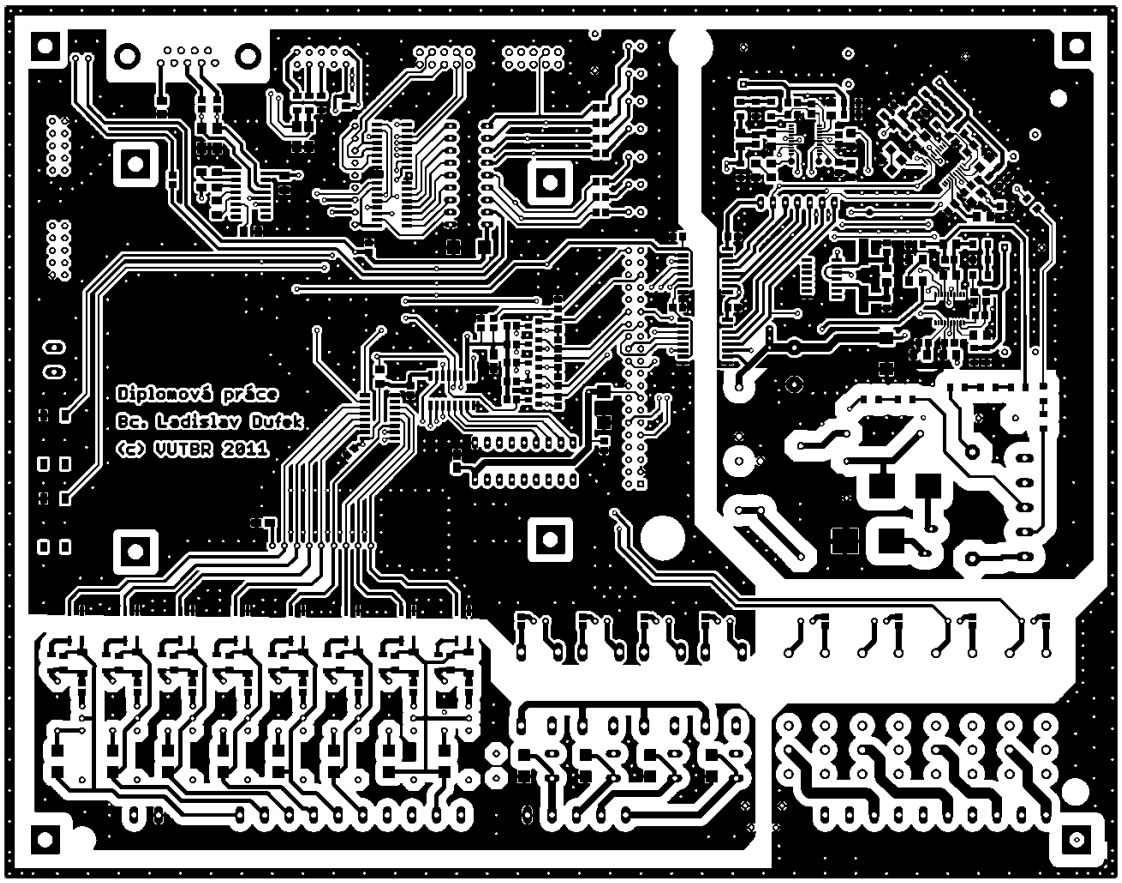
Potisk: ANO

Povrchová úprava: H.A.S.L. (bezolovnatý HAL)

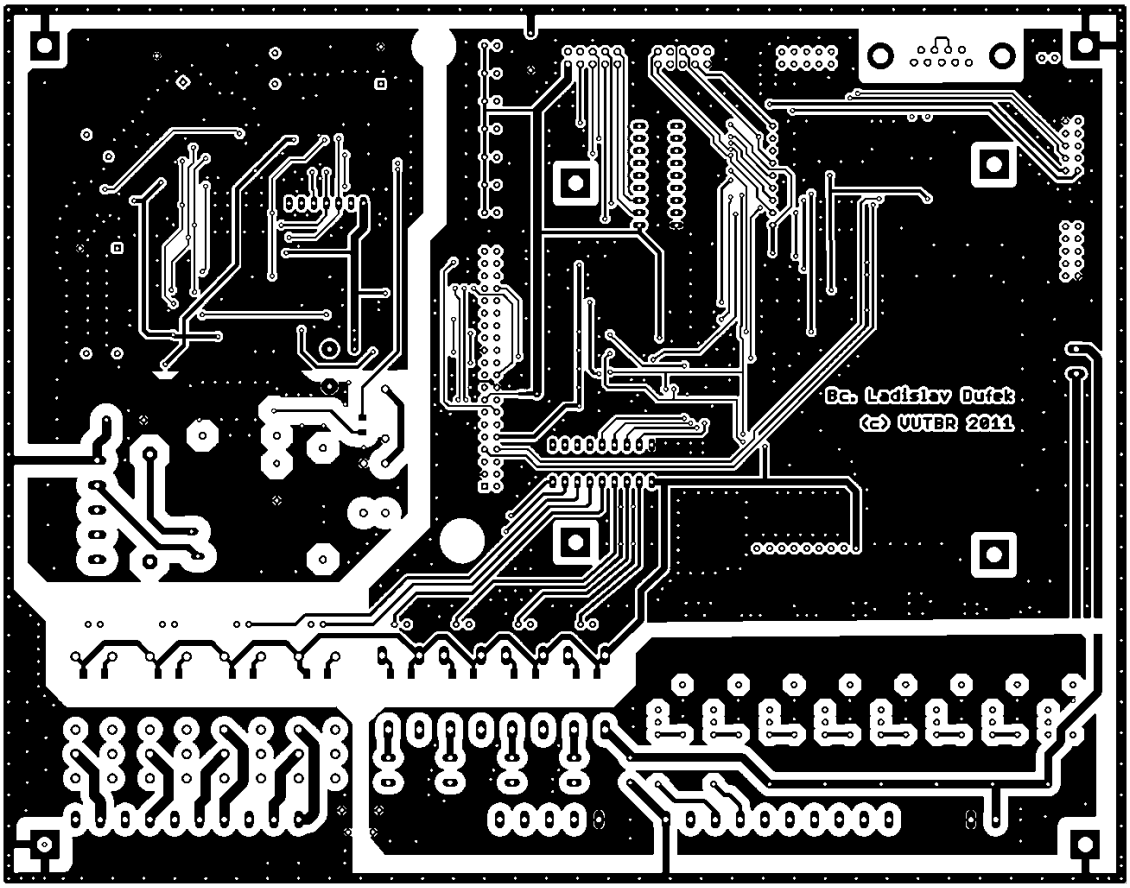
Deska je navržena jako oboustranná, oboustranně osazená, s prokovenými otvory bez drátových propojek.

Seznam použitých součástek, je kvůli svému rozsahu uveden v elektronické příloze.

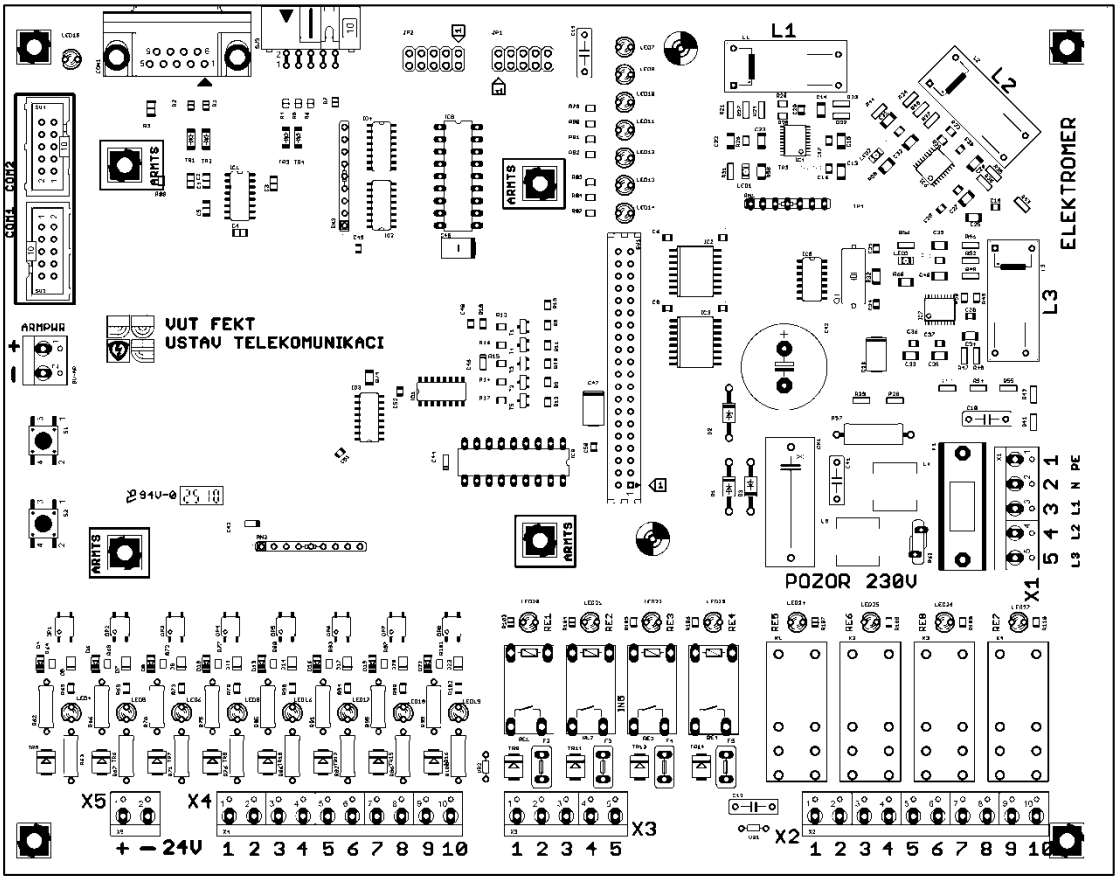
Obrazy DPS na obrázcích 1.1 – 1.4 mají jen informativní charakter (nejsou v měřítku 1:1). Při výrobě je třeba používat přímo zdrojové soubory z elektronické přílohy.



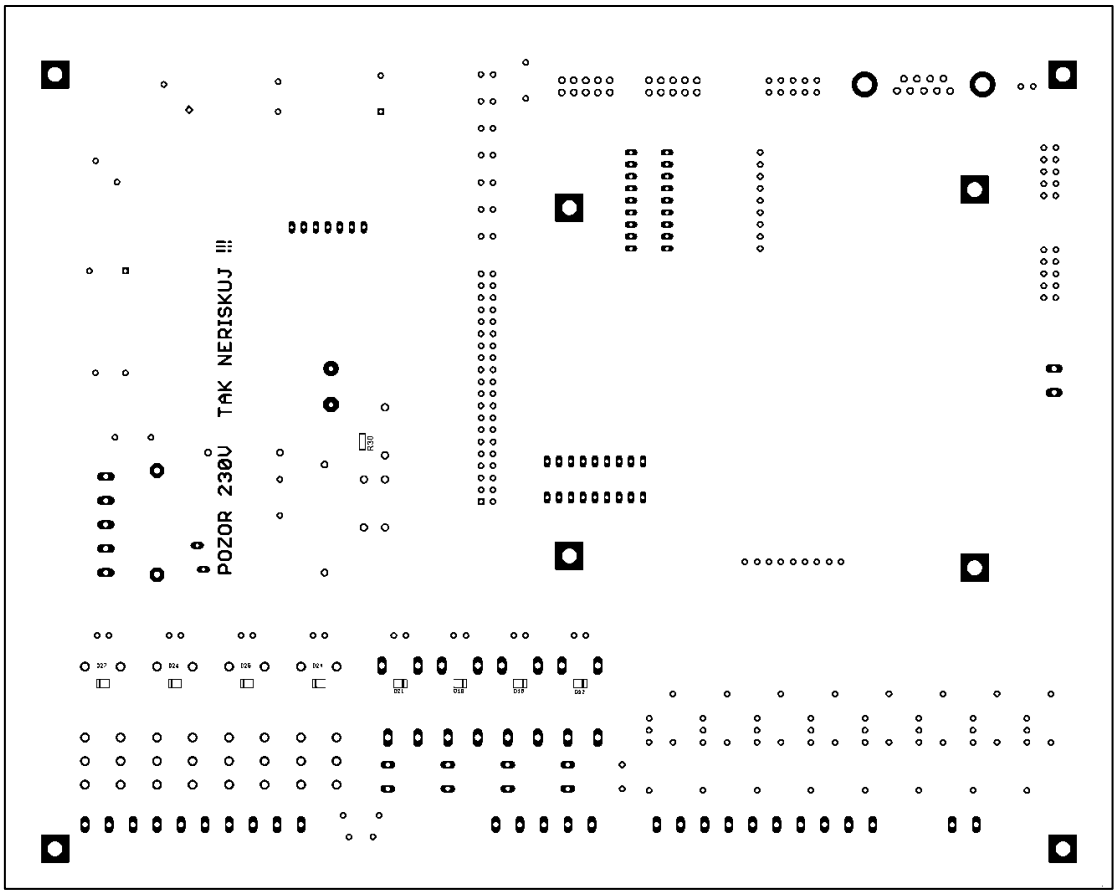
Obr. 1.1: Deska plošných spojů – vrstva TOP



Obr. 1.2: Deska plošných spojů – vrstva BOTTOM



Obr. 1.3: Osazení součástek – vrstva TPLACE



Obr. 1.4: Osazení součástek – vrstva BPLACE

Příloha E – Komunikační protokol řídicího programu

Interní komunikační protokol typu klient-server, mezi aplikací ovládající hardware a webovým interface.

Verze 1.0

Komunikace mezi webovým interface (klientem) a aplikací obsluhující hardware (serverem) probíhá prostřednictvím protokolu TCP na lokální smyčce localhost. Spojení se navazuje na portu 20000.

Protokol je textově orientovaný, nezabezpečený, kódovaný ve formátu ASCII. Kromě přihlášení zde neprobíhá žádné potvrzování zpráv na úrovni protokolu.

Navázání spojení

Navázání spojení provádí webová stránka (klient), na adrese localhost:20000 zasláním řetězce:

„maco“ + hesh hesla, který byl použit pro přihlášení na stránku

Například: **maco6d3f093e53d77348c4552d4ee2fe08de288ba02c**

Server prověří hash hesla a v případě, že souhlasí s hashem v jeho databázi, potvrdí spojení odesláním **„maco“**. V opačném případě spojení ukončí. Tvorba hashe není součástí specifikace protokolu.

Aktualizace obsahu stránky příkazem `get`

Pro aktualizaci svého obsahu posílá webová stránka opakovaný dotaz **`get`** následovaný názvem stránky, podle které server zašle stavy pouze odpovídajících vstupů.

Dotazy jsou jednotně kódovány do následujícího tvaru:

```
get=<název stránky>;
```

Příklady:

```
get=welcome;
get=iocontrol;
```

V případě, že za příkazem **get** je stránka, kterou server nezná, server odpoví:

get&<délka zprávy>&error=<název stránky>;

Pole **délka zprávy** udává vždy počet byte zprávy za znakem **&** až do posledního platného znaku zprávy, typicky „;“. Tímto je ošetřeno, že server po přejetí n-tého znaku příjem ukončí a odešle odpověď.

Například:

get&14&error=badpage;

Akceptovatelné názvy stránek jsou:

Dotaz	Název stránky
welcome	Úvodní uvítací stránka
Iocontrol	Ovládání vstupů a výstupů
energymt	Elektroměr
heatpump	Tepelné čerpadlo
error	Chybné volání stránky

Klientská stránka se na změnu stavu dotazuje v pravidelných intervalech, daných časovou konstantou ve skriptu PHP. Po odeslání dotazu vždy očekává odpověď, pokud odpověď nedorazí ve zvoleném intervalu (cca 15 sekund) uzavře spojení a vyšle dotaz znovu. Dotazy jsou však zasílány pouze v případě, že je navázáno TCP spojení s aplikací. Párování zpráv dotaz – odpověď se neprovádí. V případě, že server zašle data pro jinou stránku, nebo v neplatném formátu, informace se zahodí. Jakmile server odešle odpověď na dotaz, uzavře TCP spojení. Počet byte zprávy udává počet znaků (ASCII).

Odpověď serveru na dotaz **get** má jednotný charakter, a pro všechny stránky má následující formát:

get&<délka zprávy>&<název proměnné>=<hodnota proměnné>;

Příklad odpovědi pro uvítací stránku:

```
get&59&prg_ver=1.00;prg_name=maco;prg_datetime=26.5.2011
15:30:00;
```

Příklad odpovědi pro stránku ovládání vstupů a výstupů:

```
get&138&io_in01=1;io_in02=0;io_in03=0;io_in04=1;io_in05=0
;io_out01=0;io_out02=0;io_out03=1;io_out04=0;io_out05=1;io_out
06=0;io_out07=1;io_out08=0;
```

Názvy proměnných a hodnoty jsou řazeny přímo za sebe, oddělené středníkem. Vše je posíláno v textovém formátu, tak jak se má text zobrazit na webové stránce. Pokud server zašle název proměnné, kterou webová stránka nezná, hodnota i název se ignoruje.

Aktualizace programu příkazem `set`

Interaktivní webový formulář dovoluje uživateli provádět na webové stránce změny nastavení, resp. přímé ovládání hardware. Jakmile provede na formuláři změnu editovatelné položky, je nové nastavení odesláno do programu pomocí příkazu **`set`**.

Odeslání proběhne okamžitě pouze v případě, že webová stránka právě neočekává odpověď **`get`**, jinak je zařazeno „do fronty" a odešle se po uvolnění komunikačního kanálu.

Potvrzení provedení změny stavu výstupu se neprovádí, ke kontrole slouží skutečnost, že na další dotaz webové stránky **`get`** server odpoví již novou hodnotou.

Povely jsou jednotně kódovány do následujícího tvaru:

`set&<délka zprávy>&<název proměnné>=<hodnota proměnné>;`

Příklady: **`set&22&io_out01=1;io_out10=1;`**

`set&11&io_out02=0;`

Jedna zpráva **`set`** může nést více proměnných, ale nemusí. Dělícím a ukončovacím znakem je opět středník. V případě, že zaslaný povel aplikace nezná zašle chybovou zprávu ve tvaru:

`set&<délka zprávy>&error=<název proměnné>;`

Například:

`set&30&error=io_out20;error=io_out30;`

Zpětná chybová hlášení jsou ve webovém formuláři ignorována, slouží pro diagnostické účely, popřípadě mohou být využity pro komunikaci s jinou klientskou aplikací.

Hodnoty proměnných, které jsou použity pro jednotlivé oblasti programu, odpovídají názvům v PHP skriptu webových stránek.

Uvítací stránka

Tabulka 1: Proměnné v části Welcome

Název proměnné v C	Datový typ	popis
prg_ver	char[10]	Číslo verze programu
prg_name	char[10]	Název programu
prg_datetime	char[20]	Datum a čas SBC

Ovládání vstupů a výstupů

Tabulka 2: Proměnné v části IOControl

Proměnná v C	Datový typ	popis
io_in01	bool	Vstup číslo 1
io_in02	bool	Vstup číslo 2
io_in03	bool	Vstup číslo 3
io_in04	bool	Vstup číslo 4
io_in05	bool	Vstup číslo 5
io_in06	bool	Vstup číslo 6
io_in07	bool	Vstup číslo 7
io_in08	bool	Vstup číslo 8
io_in09	bool	Vstup číslo 9 (tlačítko)
io_in10	bool	Vstup číslo 10 (tlačítko)
io_in11	bool	Vstup číslo 11
io_in12	bool	Vstup číslo 12
io_in13	bool	Vstup číslo 13
io_in14	bool	Vstup číslo 14
io_in15	bool	Vstup číslo 15
io_in16	bool	Vstup číslo 16
io_out01	bool	Výstup relé číslo 1
io_out02	bool	Výstup relé číslo 2
io_out03	bool	Výstup relé číslo 3
io_out04	bool	Výstup relé číslo 4
io_out05	bool	Výstup relé číslo 5
io_out06	bool	Výstup relé číslo 6
io_out07	bool	Výstup relé číslo 7
io_out08	bool	Výstup relé číslo 8
io_out09	bool	Výstup LED
io_out10	bool	Výstup LED
io_out11	bool	Výstup LED
io_out12	bool	Výstup LED
io_out13	bool	Výstup LED
io_out14	bool	Výstup LED
io_out15	bool	Výstup LED
io_out16	bool	Výstup LED

Elektroměr

Tabulka 3: Proměnné v části EnergyMeter

Proměnná v C	Datový typ	popis
em_U1trms	double	Okamžitá ef. hodnota napětí u(t)/mV na fázi L1
em_U2trms	double	Okamžitá ef. hodnota napětí u(t)/mV na fázi L2
em_U3trms	double	Okamžitá ef. hodnota napětí u(t)/mV na fázi L3
em_I1trms	double	Okamžitá ef. hodnota proudu i(t)/mA na fázi L1
em_I2trms	double	Okamžitá ef. hodnota proudu i(t)/mA na fázi L2
em_I3trms	double	Okamžitá ef. hodnota proudu i(t)/mA na fázi L3
em_F1t	double	Okamžitá hodnota frekvence f(t)/Hz na fázi L1
em_F2t	double	Okamžitá hodnota frekvence f(t)/Hz na fázi L2
em_F3t	double	Okamžitá hodnota frekvence f(t)/Hz na fázi L3
em_P1t	double	Okamžitá hodnota činného výkonu P/mW na fázi L1
em_P2t	double	Okamžitá hodnota činného výkonu P/mW na fázi L2
em_P3t	double	Okamžitá hodnota činného výkonu P/mW na fázi L3
em_P1vat	double	Okamžitá hodnota zdánlivého výkonu výkonu P/mVA na fázi L1
em_P2vat	double	Okamžitá hodnota zdánlivého výkonu výkonu P/mVA na fázi L2
em_P3vat	double	Okamžitá hodnota zdánlivého výkonu výkonu P/mVA na fázi L3
em_P1vart	double	Okamžitá hodnota jalového výkonu P/mVAr na fázi L1
em_P2vart	double	Okamžitá hodnota jalového výkonu P/mVAr na fázi L2
em_P3vart	double	Okamžitá hodnota jalového výkonu P/mVAr na fázi L3
em_PF1t	double	Okamžitá hodnota účiníku ϕ na fázi L1
em_PF2t	double	Okamžitá hodnota účiníku ϕ na fázi L2
em_PF3t	double	Okamžitá hodnota účiníku ϕ na fázi L3
em_P1w	uint64_t	Hodnota odebrané energie P/Wh na fázi L1
em_P2w	uint64_t	Hodnota odebrané energie P/Wh na fázi L2
em_P3w	uint64_t	Hodnota odebrané energie P/Wh na fázi L3
em_Pw	uint64_t	Celková hodnota odebrané energie P/Wh
em_pulse_kW	uint64_t	Impuls odběru 1 imp / kWh
em_pulse_W	uint64_t	Impuls odběru 1000 imp / kWh
em_pulse_mW	uint64_t	Impuls odběru 100 imp / Wh (citlivost elektroměru 128imp/Wh)

Hodnoty odebrané energie em_P1w, em_P2w, em_P3w se neustále inkrementují.

Tepelné čerpadlo

Tabulka 4: Modbus (údaje stavu vstupů a výstupů z PLC)

Proměnná v C	Datový typ	popis
mo_device_nam	char[30]	Název zařízení na lince Modbus
mo_device_ver	char[10]	Číslo verze zařízení na lince Modbus
mo_rtc	char[20]	Hodnota RTC
mo_probe_1	char[10]	Hodnota vstupu čidla 1 (teplota)
mo_probe_2	char[10]	Hodnota vstupu čidla 2 (teplota)
mo_probe_3	char[10]	Hodnota vstupu čidla 3 (teplota)
mo_reOn	bool	Relé On/off
mo_redefrost1	bool	Relé odmrazování 1
mo_redefrost2	bool	Relé odmrazování 2
mo_realarm	bool	Relé alarmu
mo_relight	bool	Relé světla
mo_refan	bool	Relé ventilátoru
mo_reaux1	bool	Relé AUX1
mo_reaux2	bool	Relé AUX2
mo_re01	bool	Relé 1
mo_re02	bool	Relé 2
mo_re03	bool	Relé 3
mo_re04	bool	Relé 4
mo_re05	bool	Relé 5
mo_re06	bool	Relé 6
mo_re07	bool	Relé 7
mo_re08	bool	Relé 8
mo_re09	bool	Relé 9
mo_re10	bool	Relé 10
mo_re11	bool	Relé 11
mo_regeneric1	bool	Obecný výstup relé 1
mo_regeneric2	bool	Obecný výstup relé 2

Tabulka 5: Modbus stavy (číselník), údaje pro zjištění stavu PLC

Proměnná v C	Položka	Datový typ	popis
mo_stat_en	1	char[22]	Zařízení
mo_stat_defr	2	char[22]	Odmrazování
mo_stat_freez	3	char[22]	Rychlé chlazení
mo_stat_man	4	char[22]	Ruční ovládání
mo_stat_reset	5	char[22]	Reset alarmů
mo_stat_save	6	char[22]	Režim úspory energie
mo_stat_instat	7	char[22]	Digitální vstup - stav
mo_stat_holid	8	char[22]	Funkce „prázdniny“
mo_stat_aux	9	char[22]	Funkce AUX
mo_stat_light	10	char[22]	Funkce LIGHT
mo_stat_resetplc	11	char[22]	Reset PLC

Tabulka 6: Modbus příznaky (číselník), doplnění stavů z tabulky 5.

Proměnná v C	Položka	Datový typ	popis
stat_0	0	bool (char[7])	VYPNUTO
stat_1	1	bool (char[7])	ZAPNUTO

Tabulky Modbus stavy a příznaky jsou propojené. Za stavem vždy následuje příznak (ZAPNUTO/VYPNUTO) v C je typ bool, v db char.

Tabulka 7: Modbus chyby (číselník)

Proměnná v C	Položka	Datový typ	popis
mo_err_none	1	char[30]	Žádná chyba (normální stav)
mo_err_low	2	char[30]	Nezávažná chyba (alarm 1)
mo_err_high	3	char[30]	Závažná chyba (alarm 2)
mo_err_probe	4	char[30]	Chyba teplotního čidla
mo_err_modbus	5	char[30]	Chyba komunikace Modbus
mo_err_door	6	char[30]	Dveře zařízení otevřeny
mo_err_generic	7	char[30]	Obecný alarmový poplach
mo_err_rtc	8	char[10]	RTC alarm
mo_err_link	9	char[40]	Chyba jednotky výměníku
mo_err_ hipress	10	char[20]	Vysoký tlak výměníku
mo_err_ lowpre	11	char[20]	Nízký tlak výměníku
mo_err_alarm1	12	char[10]	Alarm 1
mo_err_alarm2	13	char[10]	Alarm 2
mo_err_alarm3	14	char[10]	Alarm 3
mo_err_alarm4	15	char[10]	Alarm 4
mo_err_alarm5	16	char[10]	Alarm 5
mo_err_alarm6	17	char[10]	Alarm 6
mo_err_alarm7	18	char[10]	Alarm 7
mo_err_alarm8	19	char[10]	Alarm 9
mo_err_alarm9	20	char[10]	Alarm 10
mo_err_alarm10	21	char[10]	Alarm 11
mo_err_alarm11	22	char[10]	Alarm 12