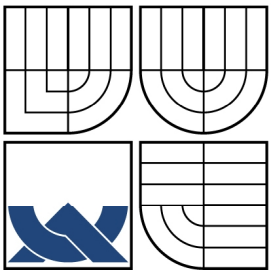
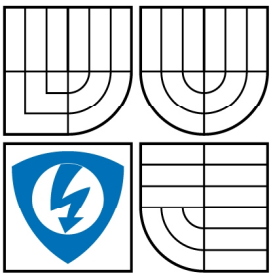


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

SYSTÉM MĚŘENÍ TEPLOTY S VÝSTUPEM ETHERNET

TEMPERATURE MONITORING SYSTEM WITH ETHERNET OUTPUT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

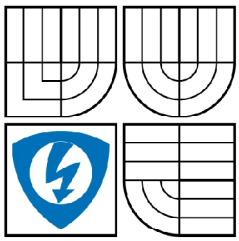
AUTOR PRÁCE — Bc. VÁCLAV HANÁK

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR FIEDLER, Ph.D.

SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Václav Hanák	***ID:***	83774
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Systém měření teploty s výstupem Ethernet

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a následně realizujte modul pro připojení měřící desky k Ethernetu. Implementujte základní protokoly, web server pro vizualizaci měřených hodnot a nastavení parametrů pro měření teploty. Pro komunikaci s řídícím systémem implementujte protokol MODBUS/TCP.

DOPORUČENÁ LITERATURA:

Dokumentace na www. rabbit.com
Konzultace u vedoucího diplomové práce

Termín zadání:	9.2.2009	***Termín odevzdání:***	25.5.2009

Vedoucí práce: Ing. Petr Fiedler, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Vysoké učení technické Brno

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Systém měření teploty s výstupem Ethernet

Diplomová práce

Studijní obor: Kybernetika, automatizace a měření

Student: Bc. Válav Hanák

Vedoucí práce: Ing. Petr Fiedler, Ph.D.

Abstrakt:

Tato diplomová práce se zabývá problematikou měření teploty v průmyslových aplikacích. Vzhledem k rozmanitosti a rychlému rozvoji průmyslu se jedná o velmi širokou problematiku. Různá odvětví průmyslu často kladou specifické požadavky na vlastnosti a parametry teplotních senzorů. Také požadavky na výstupní signál převodníků teploty mohou být velmi odlišné. Obvyklé je použití převodníků s unifikovaným analogovým výstupem, zde je pro své výborné vlastnosti používána nejčastěji proudová smyčka 4 až 20 mA. Velmi rozšířené jsou také inteligentní převodníky SMART s nejrůznějšími druhy digitálních sběrnic, nebo také s kombinací analogového i digitálního výstupu.

První část práce je zaměřena na seznámení se s principy měření teploty pomocí kovových teplotních sensorů. Práce dále popisuje návrh a realizaci inteligentího převodníku teploty s digitálním výstupem RS485. Zahrnut je jak detailní návrh obvodového řešení a desky plošných spojů, tak i vytvoření firmware pro použitý mikrokontrolér a obslužný software pro osobní počítač PC.

Druhá část práce se zabývá úvodem do komunikčního standardu Ethernet 802.3, jsou nastíněny různé možnosti implementace Ethernetu pro průmyslové převodníky fyzikálních veličin. Závěr práce se zabývá návhrhem a praktickou realizací připojení inteligentního převodníku teploty k Ethernetu.

Klíčová slova: RTD senzor, RS485, ModbusRTU, ModbusTCP, Ethernet, Rabbit, Měření teploty

Brno University of Technology

Faculty of Electrical Engineering and Communication
Department of Control, Measurement and Instrumentation

Temperature monitoring system with Ethernet output

Master‘s Thesis

Specialisation of study: Cybernetics, Control and Measurement
Student: Bc. Válav Hanák
Supervisor: Ing. Petr Fiedler, Ph.D.

Abstract:

This master‘s thesis deals with temperature measurement in industrial environment. In general this is broad issue. Different kinds of industrial production require specific characteristics and parameters of tepmerature transducers. Also the output signal of temperature transmitters is dependant on application. We can find temperature transmitters with unified analog output, most used is current loop 4 to 20 mA, it has excelent characteristics. There are also inteligent transmitters SMART with many types of digital output or with combination of both analog and digital output.

The first part of this thesis is aimed at studing basic principles of temperature measurement using resistive temperature detectors. Next parts describe design and practical implementation of inteligent temperature transmitter with digital output RS485. This incudes detailed design of circuit solution, printed circuit board, firmware for used microcontroller and user‘s software for personal computer.

The second part studies basics of the comunication standard Ethernet (IEEE 802.3), it discusses nowadays possibilities of Ehternet implementation to industrial trasmitters. Last parts of the thesis are aimed at design and practical realization of Ethernet communication module.

Keywords: RTD sensor, RS485, ModbusRTU, ModbusTCP, Ethernet, Rabbit, Temperature measurement

HANÁK, V. *Systém měření teploty s výstupem Ethernet.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 80 s. Vedoucí diplomové práce Ing. Petr Fiedler, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Systém měření teploty s výstupem Ethernet jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne: **25. května 2009** ……

podpis autora

1. OBSAH

1.1 SEZNAM OBRÁZKŮ

2. ÚVOD

Velmi častým úkolem, který je nutno řešit v průmyslových aplikacích, je měření teploty. S požadavky na přesné měření teploty se setkáváme v mnoha odvětvích, např. energetický, hutní, petrochemický, farmaceutický a potravinářský průmysl, výroba stavebních a průmyslových hmot atd. Z rozdílných požadavků jednotlivých odvětví průmyslu vyplývá, že rozpětí měřených tepot je velmi široké. K měření teploty proto není možné použít pouze jediný fyzikální princip měření. Mezi nejrozšířenější průmyslové senzory teploty patří kovové senzory na bázi platiny. Vykrývají měřící rozsah dostačující pro kryogenní techniku (-200 °C) až po oblasti energetického průmyslu a teplárenství (850 °C). Ve standardním provedení však mívají tyto snímače rozsah -50 až 400 °C. Konkurencí těchto snímačů jsou termoelektrická čidla (využívají Seebeckova jevu), která pokrývají rozsah od absolutní nuly až po 2200 °C. Jejich nevýhodou je však nutnost kompenzace studeného konce, značná nelinearita převodní charakteristiky, nižší přesnost a také nízká citlivost. U snímačů pro vysoké teploty je nutné počítat s vyšší cenou (používají se drahé kovy) a nízkou životností snímače. V menšinovém zastoupení jsou v průmyslu používány také termistory, především v oblasti vzduchotechniky a klimatizační techniky. Výhodou použití v této oblasti je vysoká citlivost, malé rozměry a nízká cena. Termistory však nelze používat pro přesná měření. V dnešní je stále více oblíbené bezdotykové měření teploty. Pro speciální aplikace je možno použít snímačů na bázi kapalných krystalů, senzory dilatační, ultrazvukové a teploměrné barvy. V některých aplikacích je třeba dbát na to, aby proces měření teploty nemohl ovlivnit bezpečnost daného provozu. V prostředích s nebezpečím výbuchu (Ex), jako jsou např. důlní průmysl nebo plynárenství, je proto nutné dbát na to, aby měření teploty nemohlo způsobit žádné ohrožení (výbuch). Kromě speciálně ošetřených RTD snímačů se pro tato prostředí používají také optické vláknové snímače.

Průmyslové převodníky teploty jsou dodávány s různými typy výstupů. Klasické analogové převodníky s unifikovaným výstupem jsou postupně vytlačovány převodníky digitálními. Obvyklé je použití sběrnic RS232C, RS485, protokoly Hart,

Modbus, Profibus atd. V poslední době se začínají objevovat také převodníky s výstupem Ethernet.

2.1 FORMULACE CÍLŮ DIPLOMOVÉ PRÁCE

Diplomová práce tématicky navazuje na semestrální projekty LM1K a LM2K. V rámci těchto projektů byly studovány principy měření teploty pomocí kovových teplotních senzorů, byl navržen a realizováno inteligentní modul pro měření teploty s výstupem RS485. Cílem diplomové práce je seznámit se komunikačním standardem Ethernet, nastudovat současné možnosti řešení připojení průmyslových převodníků fyzikálních veličin a jednoduchých vestavných systémů k Ethertnetu. Hlavním cílem práce je pak navrhnout a prakticky realizovat komunikační modul, pomocí kterého bude možno připojit měřící modul k Ethernetu. Návrh zahrnuje tyto body:

- výběr optimální platformy, důraz je kladen na konečnou cenu, náklady na vývojové prostředky, proudovou spotřebu a rozměry komunikačního modulu
- vytvoření firmware - komunikace se modulem pro měření teploty, web server pro vizualizaci naměřených hodnot a nastavení parametrů zařízení, implementace protokolu Modbus/TCP
- praktická realizace funkčního prototypu, ověření jeho funkcí

Pozn.: Kapitoly 3, 4 a 5 této diplomové práce rozšiřují a doplňují kapitoly 4, 5 semestrálních projektů LM1K a LM2K.

3. KOVOVÉ TEPLOTNÍ SENZORY

3.1 ÚVOD

Měřícím principem odporových kovových teplotních senzorů je změna odporu kovu v závislosti na teplotě. Kov si lze představit jako soubor kladných iontů, umístěných v mřížkových bodech krystalové mřížky a tzv. elektronového plynu, tvořeného souborem chaoticky se pohybujících elektronů [1]. Obecně by bylo možno použít pro měření teploty téměř libovolný kov, z praktického hlediska se však používají kovy platina (Pt), nikl (Ni), měď (Cu), molybden (Mo). Z nich je nejčastěji používaným kovem je platina, která je nepsaným průmyslovým standardem.

3.2 PARAMETRY RTD SENZORŮ

Kovové teplotní senzory jsou definovány těmito technickými parametry:

- R_0 – odpor senzoru při teplotě 0°C. Tento parametr bývá zpravidla 100Ω, 200Ω, 500Ω, 1000Ω, 2000Ω, 10000Ω.
- α – teplotní součinitel odporu. Parametr α stanovuje přírůstek odporu senzoru při změně teploty o 1°C. Je definován pro rozsah 0 až 100°C, udává se v jednotkách [ppm/K]. Parametr α je stanoven dle vztahu 3.1.

$$a = \frac{R_{100} - R_0}{100 \cdot R_0} \qquad (3.1)$$

Závislost odporu na teplotě v rozsahu 0 až 100°C je přibližně vyjádřena následujícím lineárním vztahem 3.2.

$$R_t = R_0 \cdot (1 + a \cdot t) \quad [\Omega] \qquad (3.2)$$

- W – tento parametr udává poměr odporů senzoru při teplotě 100°C (R_{100}) a při teplotě 0°C (R_0). Parametr W je určen dle vztahu 3.3.

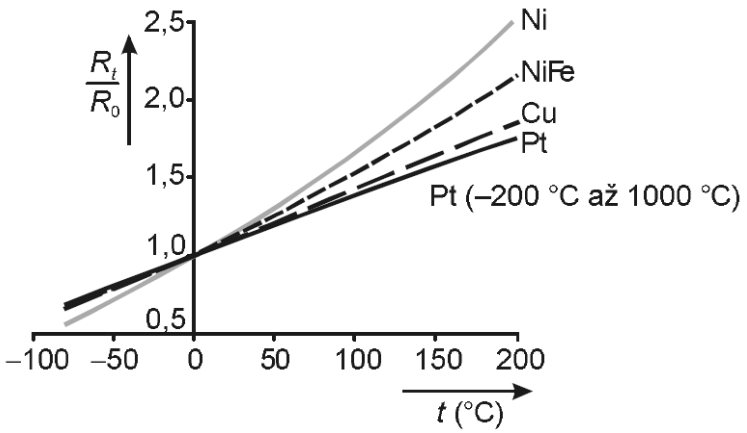
$$W_{100} = \frac{R_{100}}{R_0} \quad [-] \qquad (3.3)$$

3.3 MATERIÁLY POUŽÍVANÉ PRO RTD SENZORY

Nejčastěji používanými kovy jsou platina (Pt), nikl (Ni), měď (Cu), Balco (Ni-Fe), molybden (Mo). Pro extrémně nízké teploty (od 0,5 K) se používají slitiny kovů Rh-Fe, Pt-Co a jiné. Srovnání parametrů jednotlivých materiálů RTD senzorů udává tabulka č. 1 a obrázek č. 1 [1].

materiál	$\alpha \cdot 10^2$ (K^{-1})	teplotní rozsah (°C)	poměr odporů W_{100}
platina	0,385 až 0,391	–20 až 850	1,3850
nikl	0,617 až 0,675	–70 až +150 (+200)	1,6180
Ni-Fe	0,518 až 0,527	–100 až +200	1,462
měď	0,426 až 0,433	–50 až +150	1,4260

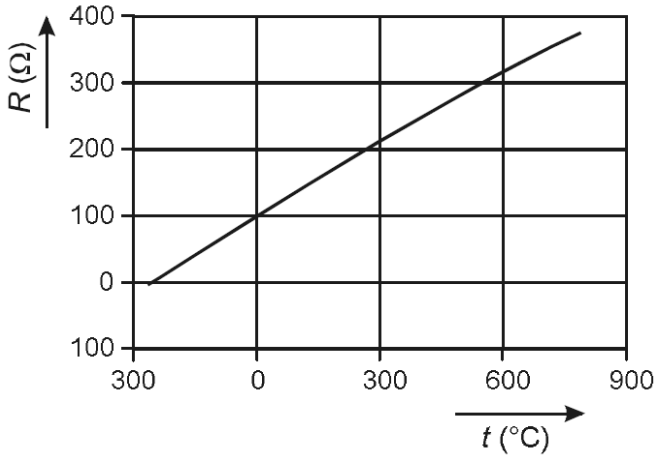
Tabulka 1. Srovnání parametrů jednotlivých materiálů [1]



Obrázek 1. Srovnání teplotní závislost jednotlivých materiálů [1]

3.4 PLATINOVÉ TEPLOTNÍ SENSORY

Mezi hlavní výhody platinových senzorů patří časová stálost, vysoká teplota tavení a chemická netečnost. Norma IEC-751 předepisuje pro provozní platinové snímače parametr W_{100} = 1,385. Parametr R_0 obvykle bývá 100Ω, vyrábějí se však i verze s R_0 50, 200, 500, 1000, 2000Ω. Závislost odporu platinového senzoru na teplotě je uvedena na obrázku č. 2 [1].



Obrázek 2. Závislost odporu platinového senzoru na teplotě [1]

Dle normy IEC-751 jsou provozní platinové senzory děleny do dvou tříd přesnosti. Třída A je určena pro měřící rozsah -200 až 650°C, třída B pro měřící rozsah -200 až 850°C. Závislost odporu platinového senzoru na teplotě v rozsahu -200 až 0°C je vyjádřena vztahem 3.4, pro rozsah od 0 do 850°C platí vztah 3.5 [1].

$$R_t = R_0 \cdot \left[1 + At + Bt^2 + Ct^3(t-100)\right] \quad [\Omega] \qquad (3.4)$$

$$R_t = R_0 \cdot \left[1 + At + Bt^2\right] \quad [\Omega] \qquad (3.5)$$

3.5 NIKLOVÉ TEPLOTNÍ SENZORY

Mezi hlavní výhody niklových senzorů patří velká citlivost, rychlá odezva a malé rozměry. Nevýhodou je omezený měřící rozsah. Ve srovnání s platinovými senzory vykazují niklové senzory větší nelinearitu převodní charakteristiky, horší dlouhodobou stabilitu a odolnost vůči vlivům prostředí. Parametr R_0 obvykle bývá 100Ω jako u platiny, vyrábějí se i verze s R_0 200, 500, 1000, 2000Ω. Závislost odporu niklového senzoru na teplotě je vyjádřena vztahem 3.6 [1].

$$R_t = R_0 \cdot \left[1 + At + Bt^2 + Ct^4 + Dt^6\right] \quad [\Omega] \qquad (3.6)$$

Hodnoty koeficientů A, B, C, D v rovnici 3.6 určuje norma DIN43760.

3.6 MĚDĚNÉ TEPLOTNÍ SENZORY

Měděné teplotní senzory se používají v rozsahu -200 až 200°C. Závislost odporu měděného senzoru na teplotě v rozsahu -50 až 150°C je vyjádřena lineárním vztahem 3.2, pro jiné rozsahy je nutno opět použít polynomickou interpolaci. Vzhledem k malé rezistivitě a snadné oxidaci mědi se tyto snímače běžně nepoužívají. Výhodnou aplikací teplotní závislosti mědi je měření teploty měděného vinutí elektrických motorů měřením odporu jejich vinutí (při vypnutém napájení) [1].

3.7 TECHNOLOGIE VÝROBY RTD SENZORŮ

RTD senzory jsou k dispozici ve třech provedeních.

- **Vinuté** – na keramickém tělísku je navinutý odporový drát o takové délce, aby jeho odpor odpovídal parametrem R_0. Tento celek je po té zataven a zapouzdřen do kovové jímky. Výhodou je široký měřící rozsah -200 až 850°C, možnost výroby zákaznických provedení (parametr R_0 si určí zákazník) a vysoká přesnost. Nevýhodou je vyšší cena a rozměry.
- **Tlustovrstvé** – na keramickém substrátu je (z pravidla) sítotiskem nanesen odporový meandr z kovové pasty. Laserovým trimováním je pak dosažena požadovaná hodnota parametru R_0. Přívodní vodiče bývají přitmeleny. Výhodou oproti vinutému provedení je nižší cena, nevýhodou menší časová stálost a užší měřící rozsah (-200 až 400°C).
- **Tenkovrstvé** – na keramický substrát je naprášena tenká vrstva kovu (magnetronové naprašování). Trimování se provádí opět laserem. Výhodami jsou malé rozměry (pouzdro 1206), nízká cena, velmi rychlá odezva, vysoká časová stabilita. Nevýhodou je úzký měřící rozsah -50 až 400°C.

4. NÁVRH INTELIGENTNÍHO MODULU PRO MĚŘENÍ TEPLOTY

Na inteligentní modul pro měření teploty jsou kladeny následující požadavky.

- napájecí napětí v rozsahu 10 až 24 VDC
- přesnost měření 0,1% z rozsahu
- zpracování signálu RTD senzoru řady PT100, PT1000, NI100, NI1000
- počet měření 5/sec
- komunikace s osobním počítačem pomocí sériové linky RS485
- komunikační protokol Modbus/RTU.
- galvanické oddělení komunikační linky RS485
- možnost lokální konfigurace pomocí tlačítek a LCD displeje

Schéma zapojení a motiv desky plošného spoje inteligentního modulu pro měření teploty jsou uvedeny v příloze č. 1 a č. 2. Modul může být rozdělen na následující funkční bloky:

- napájecí zdroj
- digitální část
- vstupní obvody pro připojení RTD senzoru
- galvanicky oddělená komunikační linka RS485

4.1 NÁVRH NAPÁJECÍHO ZDROJE

Při návrhu napájecího zdroje je nutné vycházet z celkové proudové spotřeby navrhovaného zařízení a z velikosti výkonové ztráty, která vzniká na hlavním regulačním prvku zdroje při maximální hodnotě napájecího napětí. Dle těchto parametrů lze pak posoudit, jestli je možné použít klasický lineární stabilizátor, nebo v případě příliš velké výkonové ztráty volit variantu spínaného zdroje. V takovém případě je nutné brát v úvahu fakt, že spínané zdroje mívají ve srovnání s lineárními stabilizátory řádově vyšší úroveň zvlnění výstupního napětí. Tato vlastnost je

obvyklá zejména u modulových DC/DC převodníků, u kterých je tento parametr v řádu až stovek milivoltů. Pokud aplikace vyžaduje co nejnižší úroveň zvlnění, může být vhodným řešením použití kaskádního zapojení spínaného a lineárního stabilizátoru. Spínaný zdroj je v takovém případě využit jako předstabilizátor, eliminující výkonovou ztrátu. Na jeho výstup je v kombinaci s vhodným typem dolnopropustného filtru zařazen lineární stabilizátor, který pak napájí vlastní obvody zařízení citlivé na zvlnění.

Požadovaný rozsah napájecího napětí modulu pro měření teploty je 10 až 24 V. K napájení digitální a analogové části modulu je třeba interní stabilizované napětí 5 V, pro napájení sběrnice RS485 pak další, galvanicky oddělené napětí 5 V. Pro referenční napětí A/D převodníku je zapotřebí stabilizované a teplotně nezávislé napětí 2,5 V.

4.1.1 Určení proudové spotřeby modulu

Celková spotřeba modulu je dána proudovým odběrem použitých součástek, který je uveden v tabulce č. 1. V úvahu byla brána maxima hodnot proudů, uvedená v katalogových listech použitých součástek.

Součástka	Odběr proudu @ 5V [mA]
XE88LC05 @ 1.8432 MHz	2
RTD senzor	< 1
ADM2483	4,5
DC/DC převodník	5
Buzení optočlenů	3
Předpětí a zakončení RS485	3,3

Tabulka 2. Proudový odběr jednotlivých součástek modulu

Výpočet spotřeby modulu: $I_{ODHAD} = 2+1+4{,}5+5+3+3{,}3 \cong \underline{19mA}$

Připočítání rezervy: $I_{CELK} = I_{ODHAD} + 0{,}2 \cdot I_{ODHAD} = 19 + 0{,}2 \cdot 19 \cong \underline{\underline{23mA}}$

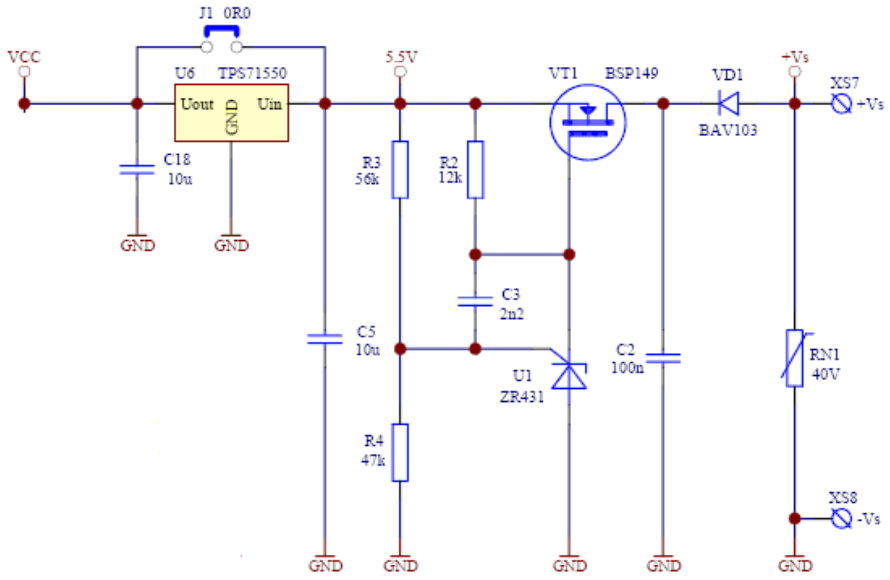
Po připočítání 20. % rezervy musí být napájecí zdroj dimenzován na 23 mA a výkonovou ztrátu určenou dle následujícího vztahu:

$$P_{TOT} = (U_{InMax} - U_{Out}) \cdot I_{Celk} = (24 - 5{,}5) \cdot 0{,}023 \cong \underline{\underline{0{,}45W}}$$

Maximální výkonová ztráta není nijak významná, proto je možné použít klasický lineární stabilizátor, bez nutnosti chlazení.

4.1.2 Návrh zapojení napájecího zdroje

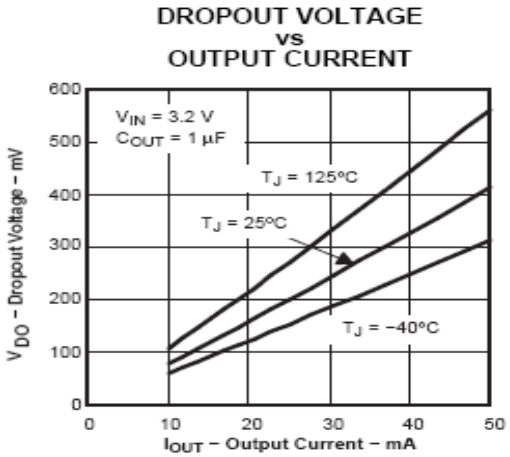
Vzhledem k výpočtům uvedeným v kapitole 3.1.1 a požadavku na široké rozpětí vstupního napájecího napětí bylo zvoleno následující řešení zdroje.



Obrázek 3. Detail zapojení napájecího zdroje modulu pro měření teploty

Napájecí zdroj se skládá ze dvou kaskádně řazených stabilizátorů. První z nich, předstabilizátor, je tvořen tří-svorkovým regulátorem ZR431 a unipolárním transistorem BSP149. Úkolem tohoto předstabilizátoru je chránit hlavní stabilizátor před přepětím a výkonovou ztrátou. Varistor RN_1 zajišťuje ochranu dalších součástek proti napěťovým špičkám. Jeho jmenovité napětí je 40 V. Je-li toto napětí překročeno, vnitřní odpor varistoru prudce klesá a zdroj rušivých impulsů je tak varistorem zatížen nakrátko. Tento typ ochrany je účinný proti krátkodobým výkonovým impulsům typu Burst a Surge. Dioda VD_1 zajišťuje ochranu proti přepólování napájecího napětí. Rezistor R_2 tvoří zdroj proudu pro regulátor ZR431. Výstupní napětí předstabilizátoru je dáno poměrem odporů rezistorů zpětnovazebního děliče R_3 a R_4. Kondenzátor C_3 tvoří zápornou zpětnou vazbu a

zabraňuje rozkmitání regulátoru. Na pozici hlavního stabilizátoru je tří-svorkový monolitickým Low-Dropout stabilizátor TPS71550 firmy Texas Instruments. Jeho vlastní spotřeba je pouze 3,2 µA, takže je vhodný také pro použití v nízkospotřebových a bateriově napájených zařízeních. Maximální hodnota vstupního napětí tohoto stabilizátoru je 24 V. Minimální hodnotu je třeba určit z katalogového listu. Následující obrázek udává závislost minimální hodnoty napájecího napětí stabilizátoru v závislosti na proudovém odběru zátěže.



Obrázek 4. Závislost Dropout napětí TPS71550 na výstupním proudu [2]

Celková proudová spotřeba modulu pro měření teploty byla v kapitole 3.1.1 stanovena na 23 mA. Stabilizátor TPS71550 je však reálně zatížen pouze odběrem mikrokontroléru a obvodů pro připojení RTD senzoru. Ostatní části modulu jsou napájeny přímo z předstabilizátoru. Z grafu uvedeného na obrázku č. 2 je patrné, že pro hodnotu proudu 10 mA je nutné, aby vstupní napětí stabilizátoru bylo minimálně 5,1 V. Výstupní napětí předstabilizátoru bylo proto zvoleno na hodnotu 5,5 V. Tato volba zahrnuje také dostatečnou rezervu na vliv teploty a stability.

4.1.3 Výpočet hodnot součástek zdroje

Výpočet odporu rezistoru R_2

Rezistor R_2 pracuje jako proudový zdroj pro regulátor ZRA431. Dle katalogového listu [3] musí být tento proud v minimálně 50 µA. Hodnotu tohoto proudu byla proto zvolena na 100 µA.

$$R_2 = \left(U_{out} - U_{ref}\right) \cdot I_N = (5{,}5 - 2.5) \cdot 100 \cdot 10^{-6} = 30k\Omega \Rightarrow \underline{\underline{33k\Omega}}$$

Výpočet odporu rezistorů R_3 a R_4

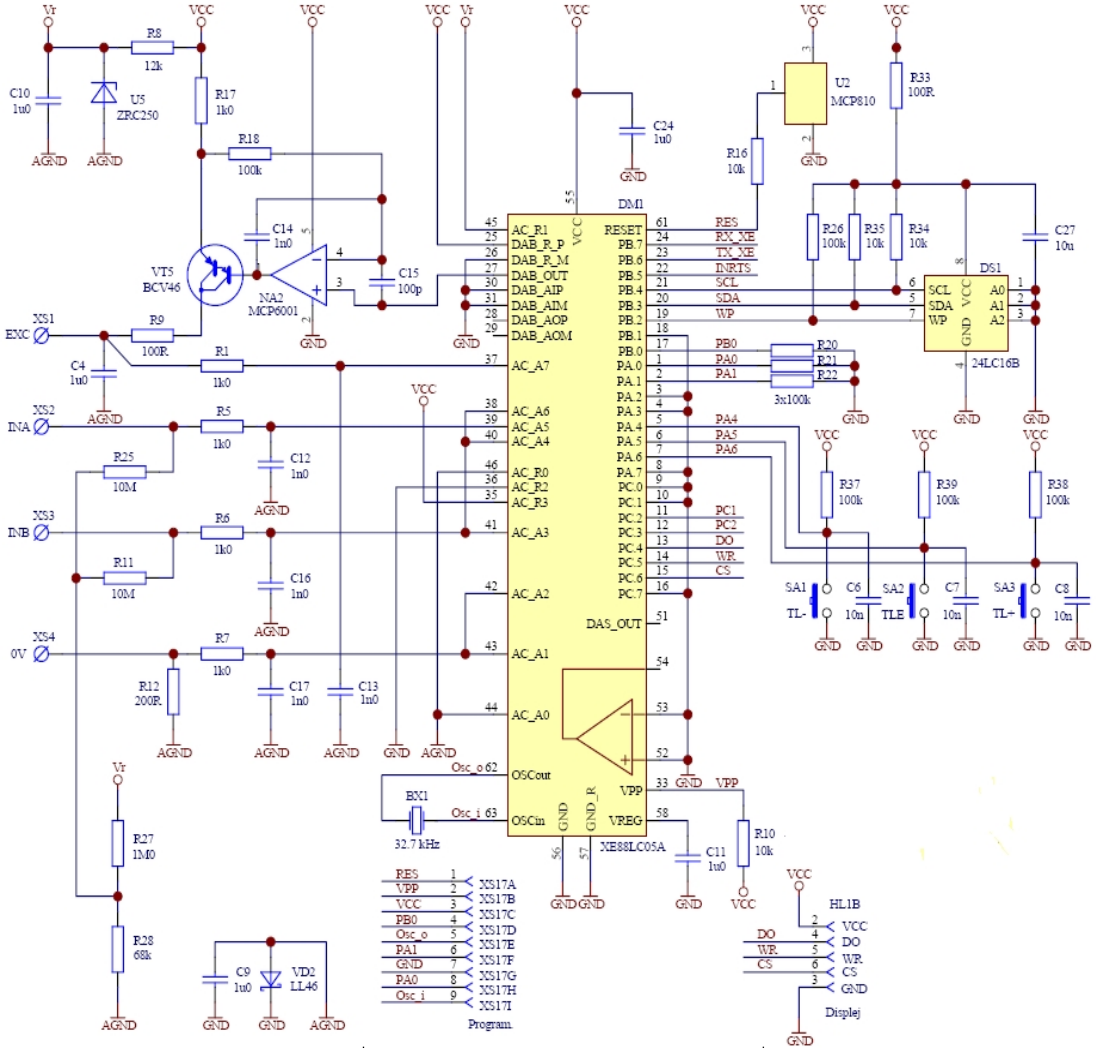
Rezistory R_3 a R_4 tvoří zpětnovazební dělič, pomocí kterého se nastavuje výstupní napětí předstabilizátoru. Vstupní proud referenčního vstupu regulátoru ZR431 je dle katalogového listu [3] maximálně 1 µA. Z ohledem na tento údaj byl zvolen proud zpětnovazebního děliče na 50 µA.

$$R_4 = \frac{U_{ref}}{I_{FB}} = \frac{2{,}5}{50 \cdot 10^6} = 50k\Omega \Rightarrow R_4 = \underline{\underline{51k\Omega}}$$

$$U_{out} = \left(1 + \frac{R_3}{R_4}\right) \cdot U_{ref} \Rightarrow R_3 = R_4 \cdot \left(\frac{U_{out}}{U_{ref}} - 1\right) = 51 \cdot 10^3 \cdot \left(\frac{5{,}5}{2{,}5} - 1\right) = 61{,}2k\Omega \Rightarrow R_3 = \underline{\underline{62k\Omega}}$$

4.2 DIGITÁLNÍ ČÁST MODULU

Digitální část modulu zajišťuje vlastní měření teploty (A/D převod, filtrace a linearizace signálu RTD senzoru), komunikaci s řídícím systémem přes sběrnici RS485 protokolem ModbusRTU, komunikaci s uživatelem pomocí lokálního rozhraní tvořeného třemi tlačítky a LCD displejem. Schéma zapojení digitální části je uvedeno na obrázku č. 5.



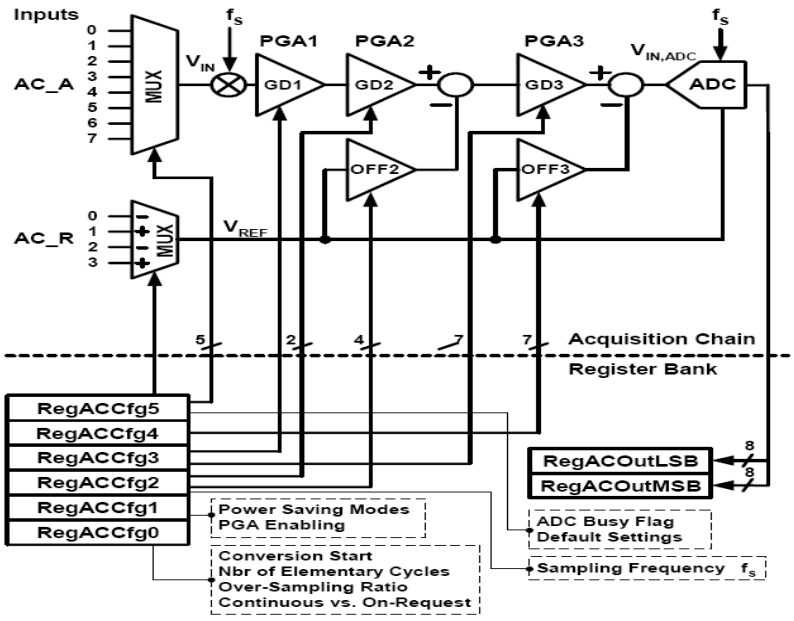
Obrázek 5. Detail zapojení digitální části modulu

4.2.1 Mikrokontrolér XE8805A

Základním prvkem digitální části je jednočipový mikrokontrolér Semtech XE8805A. Jedná se o 8. bitový mikrokontrolér s redukovanou instrukční sadou (RISC) založený na jádře CoolRisc816. Mikrokontrolér je určen pro aplikace v oblastech senzoriky (měření tlaku, teploty atd.), kde umožňuje realizaci převodníků s výstupem 4 až 20 mA pomocí jediného čipu. Vhledem k nízké spotřebě (300µA/MIPS) je vhodný také pro bateriově napájené přenosné aplikace. Klíčové vlastnosti mikrokontroléru jsou:

- 22 kB paměti programu (8k instrukcí)
- 520 bytů paměti dat
- rozsah napájecího napětí 2,4 až 5,5 V
- spotřeba 300µA/MIPS
- 16. bitový A/D převodník + PGA s nastavitelným zesílením a s možností vyrovnání offsetu před vstupem do A/D převodníku
- 16. bitový PWM D/A převodník
- operační zesilovač pro řízení proudové smyčky 4 až 20 mA
- 8. bitový D/A převodník + operační zesilovač pro realizaci zdroje proudu (nebo napětí) pro buzení senzorů
- krystalový 32768 kHz oscilátor
- programově nastavitelný RC oscilátor 100 kHz až 2 MHz
- čtyři 8. bitové čítače/časovače (capture/compare módy)
- asynchronní sériové rozhraní UART
- 3 vstupně/výstupní 8. bitové brány

Jednou z hlavních výhod tohoto mikrokontroléru je integrovaný blok ZoomingADC, který je předřazen A/D převodníku. Struktura tohoto bloku je uvedena na následujícím obrázku.



Obrázek 6. Schéma zapojení bloku ZoomingADC [4]

Vstupní multiplexer má 8 analogových vstupů (AC_A0 až AC_A7) a 4 referenční vstupy (AC_R0 až AC_R3). V aplikaci tak mohou být dva nezávislé zdroje referenčního napětí pro A/D převodník, v příslušném registru se pak programově zvolí jeden z nich. Vstupní multiplexer dále umožňuje přepínat mezi 8. analogovými vstupy, přičemž jsou možné tyto konfigurace:

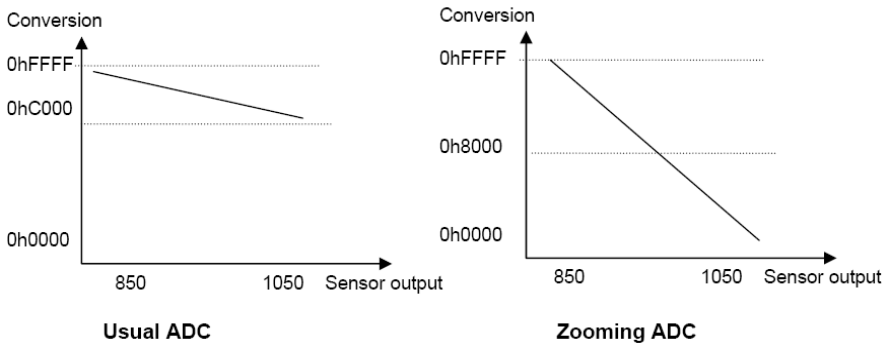
AMUX[4:0] (RegAcCfg5[5:1])	V_{INP}	V_{INN}
00x00	AC_A(1)	AC_A(0)
00x01	AC_A(3)	AC_A(2)
00x10	AC_A(5)	AC_A(4)
00x11	AC_A(7)	AC_A(6)
10000	AC_A(0)	AC_A(0)
10001	AC_A(1)	
10010	AC_A(2)	
10011	AC_A(3)	
10100	AC_A(4)	
10101	AC_A(5)	
10110	AC_A(6)	
10111	AC_A(7)	

AMUX[4:0] (RegAcCfg5[5:1])	V_{INP}	V_{INN}
01x00	AC_A(0)	AC_A(1)
01x01	AC_A(2)	AC_A(3)
01x10	AC_A(4)	AC_A(5)
01x11	AC_A(6)	AC_A(7)
11000	AC_A(0)	AC_A(0)
11001		AC_A(1)
11010		AC_A(2)
11011		AC_A(3)
11100		AC_A(4)
11101		AC_A(5)
11110		AC_A(6)
11111		AC_A(7)

Obrázek 7. Možnosti konfigurace analogových vstupů [4]

Analogové vstupy tedy mohou být konfigurovány jako 4 diferenční páry, nebo jako 7 unipolárních vstupů, přičemž vztažný potenciál je definován vstupem AC_A0. Signál ze vstupního multiplexeru je dále zpracováván kaskádou 3.

operačních zesilovačů, které umožňují flexibilní nastavení zesílení v rozsahu 0,5 až 1000. Stupeň PGA1 umožňuje hrubé nastavení zesílení, stupně PGA2 a PGA3 umožňují jemnější nastavení zesílení a také vyrovnání offsetu. Vyrovnání offsetu je velmi důležité zejména u posunutých rozsahů měřené veličiny, nebo u senzorů, které nemají vykompenzovaný offset. Výstupní napětí takových senzorů není při počátku rozsahu měřené veličiny nulové, může dosahovat desítek, v extrémních případech až stovek procent citlivosti senzoru. Při zpracování signálu s nenulovým offsetem klasickým A/D převodníkem tak dochází ke značné ztrátě rozlišení. Blok ZoomingADC obvodu XE8805A umožňuje posunout offset i rozpětí vstupního signálu tak, aby byl (v ideálním případě) shodný s referenčním napětím A/D převodníku. Teprve takto upravený signál je přiveden na vstup A/D převodníku a nedochází k žádné ztrátě rozlišení. Nastavení zesílení i offsetu se provádí programově, zápisem do příslušných řídících registrů. Použití ZoomingADC A/D převodníku je znázorněno na následujícím obrázku.



Obrázek 8. Použití ZoomingADC A/D převodníku [4]

Obrázek č. 8 porovnává zpracování výstupního napětí barometrického tlakového senzoru pomocí běžného (graf vlevo) a pomocí ZoomingADC A/D převodníku (graf vpravo). Zatímco běžný A/D převodník je využit pouze na ¼ rozsahu (14 bitů), ZoomingADC A/D převodník dosahuje za stejných podmínek plného 16. bitového rozlišení.

4.2.2 Datová paměť EEPROM

Jedinou významnou nevýhodou mikrokontroléru XE8805A je, že nemá integrovánu paměť EEPROM. Pro uchování kalibračních hodnot i uživatelského nastavení modulu je proto třeba využít externí paměť. 24LC16B je sériová paměť EEPROM s kapacitou 2kB vyráběná firmou Microchip. Díky širokému rozsahu napájecího napětí a nízké spotřebě je tato paměť ideální pro široký rozsah použití. K mikrokontroléru je připojena pomocí standardní dvojvodičové sběrnice I2C. Základní parametry paměti 24LC16B:

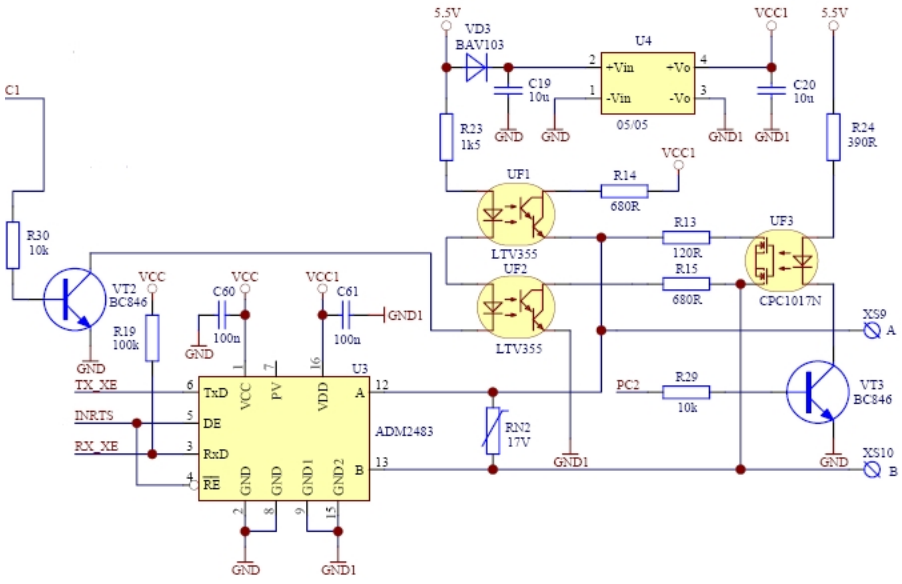
- Kapacita: 2048 bytů
- Rozsah napájecího napětí: 2,5 až 5,5 V
- Maximální proud při zápisu: 3 mA
- Maximální proud při čtení: 1 mA
- Maximální proud při nečinnosti: 1 µA
- Maximální frekvence hodinového signálu: 400 kHz
- Doba zápisu bytu nebo stránky: 5 ms

4.2.3 Resetovací obvod

Mikrokontrolér XE8805A je vybaven interním obvodem Power-on Reset (POR). Tento obvod je aktivován pouze náběžnou hranou napájecího napětí, což nelze v průmyslových aplikacích vždy zaručit. V reálné aplikaci může dojít k tomu, že napájecí napětí nabíhá pozvolna, nebo k jeho kolísání. V takovém případě interní blok POR nedetekuje hranu, nezpůsobí generování reset signálu a mikrokontrolér se může dostat do nedefinovaného stavu. V tomto stavu může dojít k trvalému poškození zařízení, například nechtěným přepsáním obsahu externí paměti EEPROM. Pro spolehlivý start mikrokontroléru je proto využit externí průmyslový resetovací obvod MCP810. Tento obvod neustále porovnává velikost napájecího napětí systému s interním referenčním napětím. Při poklesu napájecího napětí systému pod hodnotu prahového napětí obvodu MCP810 je generován reset puls.

4.2.4 Galvanicky oddělené komunikační rozhraní RS485

Rozhraní RS485 je realizováno budičem řady iCoupler ADM2483 firmy Analog Devices. Výhodou je, že tento obvod zároveň řeší galvanické oddělení vodičů sběrnice od obvodů modulu pro měření teploty. Galvanické oddělení je v průmyslových aplikacích velmi důležité. V praxi se obvykle stává, že je ke sběrnici RS485 připojeno více převodníků, které jsou nasazeny na různých místech technologického procesu a jsou napájeny z více nezávislých zdrojů napětí. U rozsáhlých procesů, jako jsou velké výrobní haly, může být v jednotlivých místech instalace převodníků rozdílný zemní potenciál. Pokud převodníky nejsou navzájem galvanicky odděleny, dochází vlivem vyrovnávání zemního potenciálu k nežádoucímu průtoku proudu přes vodiče linky RS485. Tento jev může zapříčinit chybu měření, nebo v krajním případě zničení elektronických obvodů převodníků. Z těchto důvodů je vhodné realizovat galvanické oddělení, je také dosaženo vysoké odolnosti proti zničení modulu přepětím nebo rušením, indukovaným do vodičů sběrnice. Schéma zapojení rozhraní RS485 je uvedeno na následujícím obrázku.



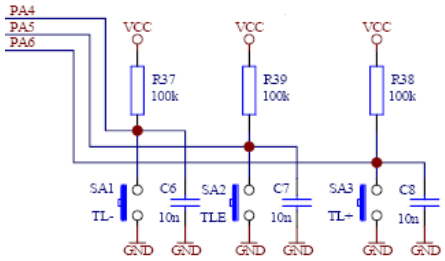
Obrázek 9. Detail zapojení komunikačního rozhraní RS485

Obvod ADM2483 vyžaduje napájení ze dvou zdrojů. Vstupní část obvodu, která je připojena k mikrokontroléru je napájena z hlavního stabilizátoru TPS71550. Výstupní část obvodu je napájena z pomocného stabilizátoru, realizováno DC/DC

převodníkem U_4. Modul pro měření teploty umožňuje také programové zapínání/vypínání předpětí a zakončení linky RS485. Zapínání/vypínání předpětí je realizováno pomocí optočlenů UF_1 a UF_2, které jsou spínány tranzistorem VT_2. Zakončení linky RS485 je realizováno obdobně, pomocí optočlenu UF_3 a tranzistoru VT_3. Vypnutí/zapnutí předpětí je ovládáno stavem výstupů PC_1 a PC_2 mikrokontroléru XE8805A.

4.2.5 Připojení tlačítek a displeje

K lokálnímu ovládání modulu slouží tři tlačítka a LCD displej.



Obrázek 10. Detail zapojení rozhraní tlačítek a displeje

Tlačítka jsou připojena k pinům portu A mikrokontroléru XE8805A. Tento port je pouze vstupní, každý z jeho pinů může při náběžné nebo sestupné hraně vstupního signálu vyvolat požadavek externího přerušení. Rezistory R_{37}, R_{38}, R_{39} (Pull-Up rezistory) zajišťují nastavení klidové úrovně vstupů. Další užitečnou vlastností pinů portu A je „digital debouncer“ – funkce pro potlačení zákmitů na kontaktech.

DebFast	Clock filter
0	256 Hz
1	8 kHz

Table 11-7: debounce frequency selection

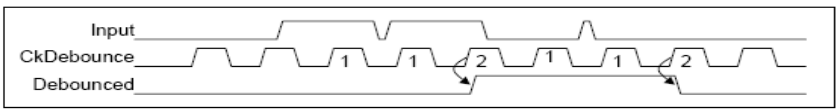


Figure 11-2: digital debouncer

Obrázek 11. Funkce pro potlačení zákmitů (Debouncer) na pinech portu A [4]

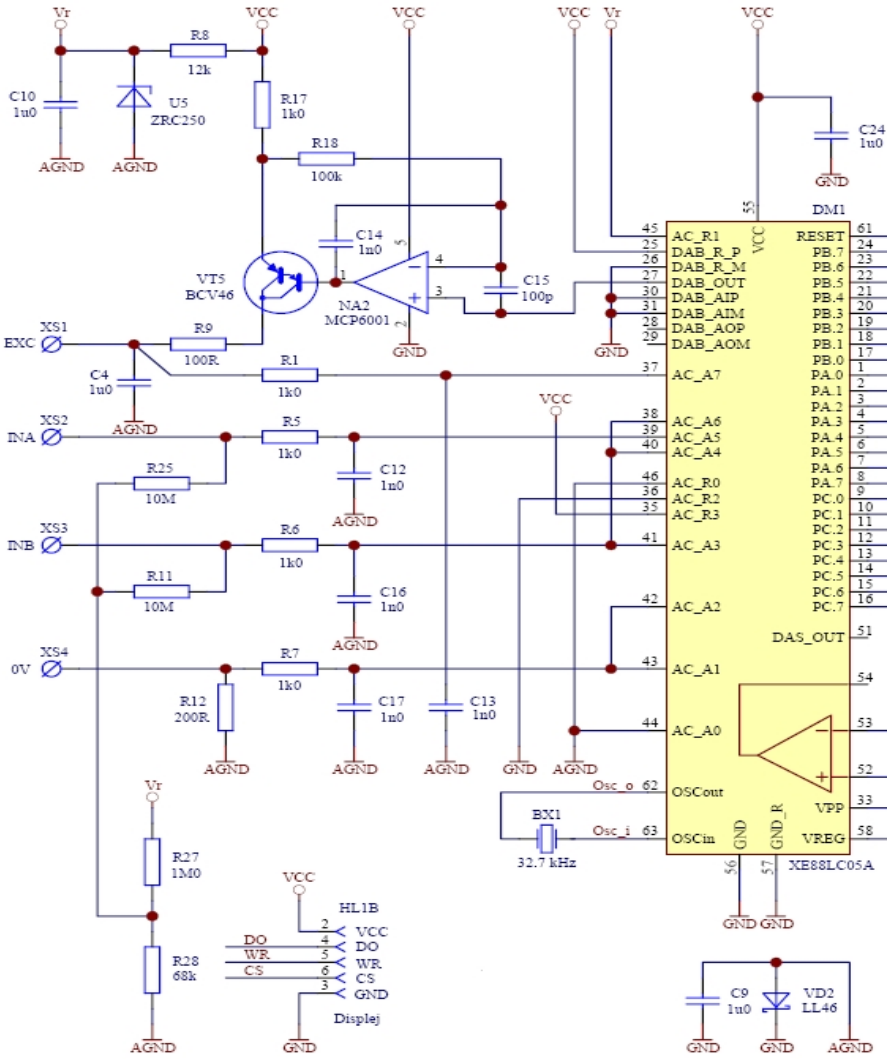
Pokud je tato funkce povolena stav každého pinu je sledován s volitelnou vzorkovací frekvencí. Změna na vstupech je akceptována pouze za podmínky, že je hodnota

tohoto vstupu neměnná v průběhu dvou po sobě jdoucích period vzorkovacího kmitočtu. Synchronizace probíhá náběžnou hranou vzorkovacího signálu.

Externí LCD displej, řízený integrovaným řadičem Holtek HT16220 je k mikrokontroléru XE8805A připojen tří-vodičovým sériovým rozhraním. Zobrazovací možnosti displeje jsou: 5 sedmi-segmentových znaků, 8 alfanumerických znaků a bargraf s 52. dílky.

4.3 OBVODY PRO PŘIPOJENÍ RTD SENZORU

Schéma zapojení vstupní části je uvedeno na následujícím obrázku.



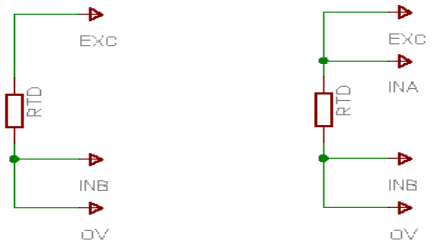
Obrázek 12. Detail zapojení vstupní části

Modul umožňuje tří-vodičové i čtyř-vodičové připojení RTD senzoru. Svorky EXC a 0V jsou výstupy pro napájení senzoru, svorky INA a INB jsou vstupy určené k měření úbytku napětí na RTD senzoru. Zdrojem referenčního napětí pro A/D převodník je dvou-svorková reference U_5 ZRA250. Pracovní bod této reference je nastaven rezistorem R_8.

4.3.1 Zdroj proudu pro RTD senzor

RTD senzor je napájen stabilizovaným zdrojem proudu, který je tvořen operačním zesilovačem NA_2, transistorem VT_5. Hodnota výstupního proudu pro RTD senzor je nastavitelná pomocí interního 8. bitového D/A převodníku mikrokontroléru XE8805A. Jeho výstupní napětí, přivedené na neinvertující vstup operačního zesilovače NA_2, slouží jako referenční napětí, kterým je řízen výstupní proud. Zpětná vazba je uzavřena přes snímací rezistor R_{17}. Procházející proud na něm vytváří úbytek napětí, který je přiveden na invertující vstup operačního zesilovače NA_2. Operační zesilovač nastavuje takovou hodnotu výstupní proud, aby došlo k vyrovnání potenciálů mezi jeho vstupy.

4.3.2 Připojení RTD senzoru



Obrázek 13. Tří a čtyř-vodičové připojení RTD senzoru

U čtyř-vodičového připojení jsou použity dva vodiče pro napájení senzoru a dva vodiče pro přivedení signálu do měřícího modulu. Výhodou tohoto zapojení je, že je tak eliminován úbytek napětí, vznikající průchodem budícího proudu na parazitním odporu přívodních vodičů. Nevýhodou je vyšší cena. Proto je rozšířené i tří-vodičové zapojení. U tohoto zapojení se vychází z toho, že všechny tři vodiče mají stejný parazitní odpor. Kompenzace probíhá tak, že je vhodným nastavením

vstupního multiplexeru A/D převodníku nejprve změřen rozdíl napětí mezi svorkami INB a OV, poté je vstupní multiplexer opět přepnut a změřen rozdíl napětí mezi svorkami EXC a INB. Odečtením těchto dvou napětí je pak eliminován úbytek na napájecích vodičích. Výhodou je ušetření jednoho vodiče, nevýhodou složitější algoritmus a delší doba měření. Pro detekci přerušení přívodu slouží rezistory R_{27} a R_{28}. Při přerušení kteréhokoliv vodiče dojde k saturaci A/D převodu a na displeji je indikován chybový stav.

5. IMPLEMENTACE FIRMWARE MODULU PRO MĚŘENÍ TEPLOTY

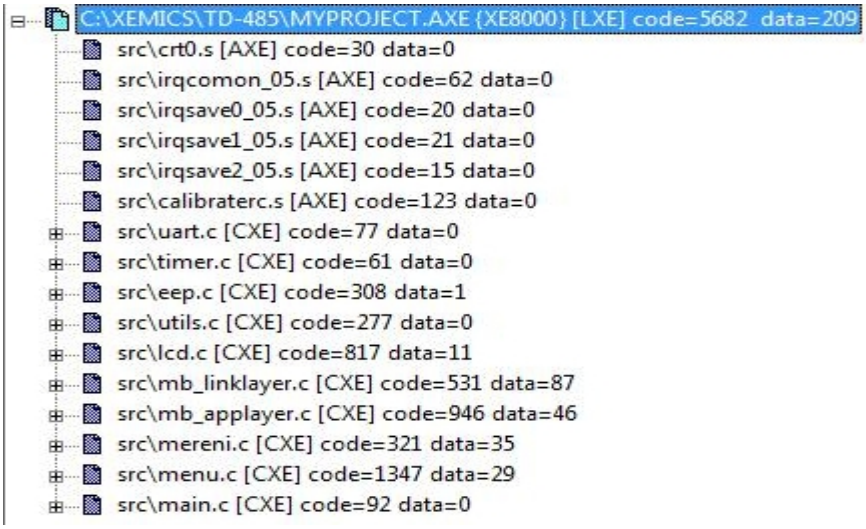
5.1 ÚVOD

Softwarové vybavení (firmware) mikrokontroléru XE8805A bylo kompletně vytvořeno a odladěno ve vývojovém prostředí RAISONANCE RIDE [5]. Firmware se skládá z několika zdrojových souborů naspaných v jazyce C a assembleru. Každý ze zdrojových souborů (modul) realizuje konkrétní funkce, například ovládá určitou periferii. Takové rozčlenění umožňuje lepší přehlednost a orientaci v celém projektu jak pro autora, tak pro další osoby.

Celá aplikace funguje jako stavový automat. Po inicializaci je v kontextu funkce *main* vykonávána hlavní (nekonečná) smyčka, ve které jsou cyklicky testovány stavové proměnné aplikace. Tyto proměnné jsou definovány jako globální-nestálé a jejich stavy jsou většinou měněny v obslužných rutinách přerušení. Tímto mechanismem je hlavní programové smyčce signalizováno, že došlo k nějaké asynchronní události, na kterou je třeba reagovat. Podle hodnoty stavové proměnné je pak vykonána konkrétní reakce. Výhoda tohoto systému je v tom, že obslužné rutiny přerušení jsou velmi krátké a nezabírají strojový čas procesoru. V jejich kontextu se vykonávají pouze nezbytné operace související s vlastní obsluhou přerušení a nastavují se zmíněné stavové proměnné. Samotná reakce může být náročnější na strojový čas a proto je vykonána v kontextu hlavní programové smyčky. Další výhodou je, že v době kdy hlavní smyčka vykonává patřičnou reakci, je povolen globální příznak přerušení a mohou tak být zaznamenány další události. Nevýhodou tohoto systému může být to, že reakce jsou prováděny v čase postupně a není stanoven žádný systém priorit jejich obsluhy.

5.2 SEZNAM ZDROJOVÝCH SOUBORŮ PROJEKTU

Součástí projektu jsou následující zdrojové soubory (moduly):

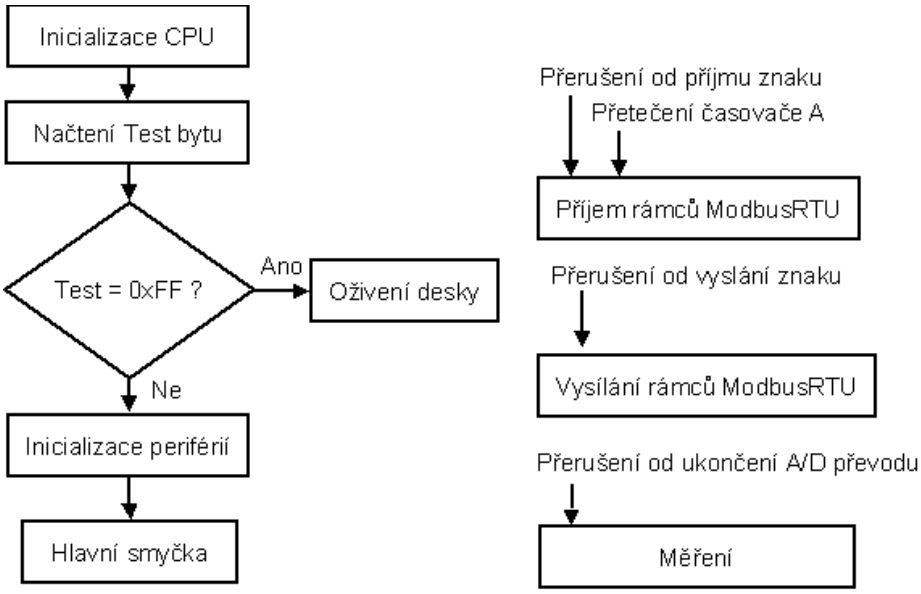


Obrázek 14. Seznam zdrojových souborů projektu

- **crt0.s** – Inicializace mikrokontroléru
- **irqcomon.s** – Obsluha přerušení
- **irqsave0_05.s** – Obsluha přerušení úrovně 0
- **irqsave1_05.s** – Obsluha přerušení úrovně 1
- **irqsave2_05.s** – Obsluha přerušení úrovně 2
- **calibraterc.s** – Kalibrace interního RC oscilátoru
- **uart.c** – Obsluha asynchronního sériového rozhraní UART
- **timer.c** – Obsluha časovačů A a C
- **eep.c** – Sw. implementace rozhraní I2C, obsluha sériové paměti 24LC16
- **utils**.c – Společné rutiny
- **lcd.c** – Funkce pro obsluhu LCD displeje
- **mb_linklayer.c** – Implementace protokolu Modbus/RTU (slave), linková vrstva
- **mb_applayer.c** – Implementace protokolu Modbus/RTU (slave), aplikační vrstva
- **mereni.c –** A/D převod, filtrace signálu, výpočet teploty
- **menu.c** – Lokální ovládání zařízení
- **main.c –** Inicializace, hlavní smyčka aplikace

5.3 POPIS FIRMWARE

Následující obrázek znázorňuje zjednodušený vývojový diagram aplikace. Detailní popis jednotlivých funkcí je uveden v následujících kapitolách.



Obrázek 15. Grafické znázornění běhu aplikace

5.4 INICIALIZACE CPU

Nízkoúrovňová inicializace CPU je provedena v modul *crt0.s*, který definuje vektory resetu a přerušení. Nízkoúrovňová inicializace zahrnuje nastavení ukazatele softwarového zásobníku, nulování obsahu paměti ram, nastavení inicializovaných C proměnných a volání funkce *main*.

5.5 NAČTENÍ HODNOTY TEST BYTU

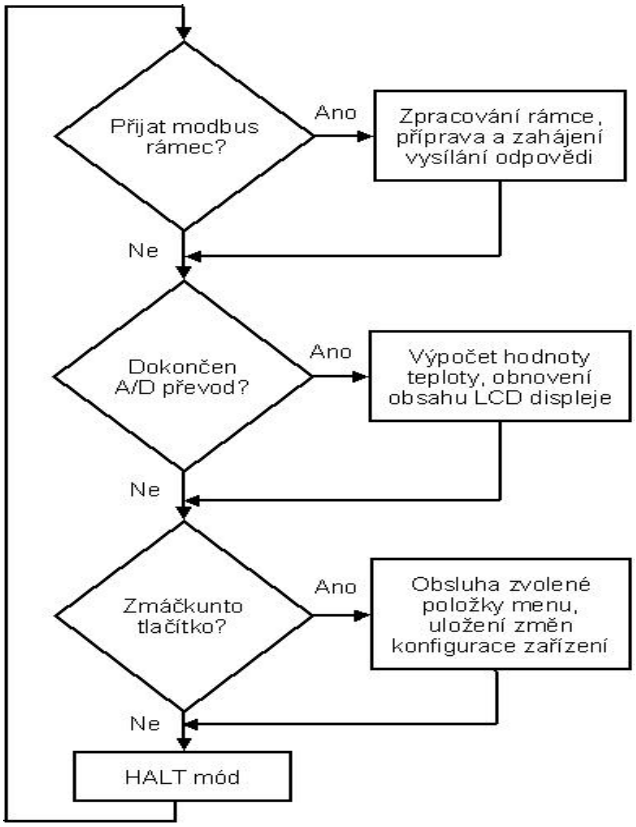
V rámci funkce main je provedeno počáteční nastavení směru a obsahu vstupně/výstupních bran. Dále je z externí paměti eeprom (poslední buňka s adresou 0x07FF) načtena hodnota test bytu. Pokud je tato hodnota rovna 0xFF (prázdná buňka eeprom), do externí paměti modulu nebyl zaveden soubor se standardními daty, modul není oživen.

5.6 OŽIVENÍ MODULU

Před prvním použitím modulu je nutno provést jeho oživení. To je provedeno v modulu *main.c*. Nejprve je pomocí funkce CalibrateRC nakalibrován interní oscilátor mikrokontroléru na hodnotu 1,2288 MHz, kalibrační konstanty jsou uloženy do eeprom. Voláním funkce *mbInit(BR_9600, 0x01, 0x00)* je inicializován komunikační proces. Modul v rámci nekonečné smyčky očekává komunikaci od PC. Na PC je spuštěn obslužný (oživovací) software, který nahraje do eeprom modulu soubor se standardními daty. Na závěr je přepsán také testovací byte na hodnotu 0x00, tím je nahrání standardních dat ukončeno. Po nahrání standardních dat je nutný reset modulu krátkým přerušením napájení. Po opětovném připojení k napájení je již modul oživen a připraven k činnosti.

5.7 HLAVNÍ SMYČKA

Sled operací hlavní smyčky je zachycen na následujícím obrázku.



Obrázek 16. Hlavní smyčka aplikace

V hlavní smyčce aplikace jsou cyklicky testovány stavové proměnné komunikace, měření a menu. Pokud není třeba obsluhovat žádnou událost, procesor je přepnut do HALT módu. Z tohoto módu se budí přijetím přerušení od přijatého/vyslaného znaku přes rozhraní UART, přerušením od ukončení A/D převodu, přetečení časovače nebo zmáčknutím každého ze tří tlačítek.

Příjem i vysílání rámců protokolu probíhá automaticky v kontextu přerušení. Po přijetí rámce protokolu Modbus je zkontrolována hodnota CRC a je porovnána adresa modulu. Pokud je rámec v pořádku a při shodě adresy, je v kontextu hlavní smyčky zpracován, připravena odpověď a zahájeno vysílání.

Po ukončení A/D převodu je v kontextu přerušení nastavena stavová proměnná procesu měření a spuštěn další A/D převod. V kontextu hlavní smyčky je proveden výpočet aktuální hodnoty teploty, aktualizace obsahu LCD displeje a

Modbus registrů. Po zmáčknutí tlačítka je v kontextu přerušení nastavena stavová proměnná menu, obsluha je provedena opět v hlavní smyčce.

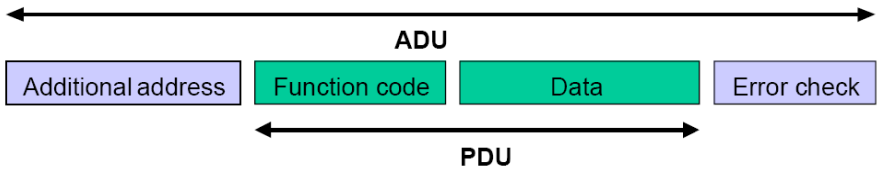
5.8 IMPLEMENTACE PROTOKOLU MODBUSRTU (SLAVE)

Protokol Modbus byl vyvinut již v roce 1979 firmou Modicon. Byl jedním z prvních komunikačních protokolů typu Master/Slave, kde komunikace probíhá formou požadavek-odpověď. V dnešní době je díky své průhlednosti a snadné implementovatelnosti stále velmi oblíbený a rozšířený v aplikacích distribuovaných systémů řízení a sběru dat [6]. Podporuje jej také většina výrobců programovatelných logických automatů (PLC). Implementaci protokolu Modbus z pohledu referenčního modulu ISO/OSI ukazuje následující obrázek.

Layer	ISO/OSI Model	
7	Application	MODBUS Application Protocol
6	Presentation	Empty
5	Session	Empty
4	Transport	Empty
3	Network	Empty
2	Data Link	MODBUS Serial Line Protocol
1	Physical	EIA/TIA-485 (or EIA/TIA-232)

Obrázek 17. Implementace prot. Modbus z hlediska ref. modelu ISO/OSI [7]

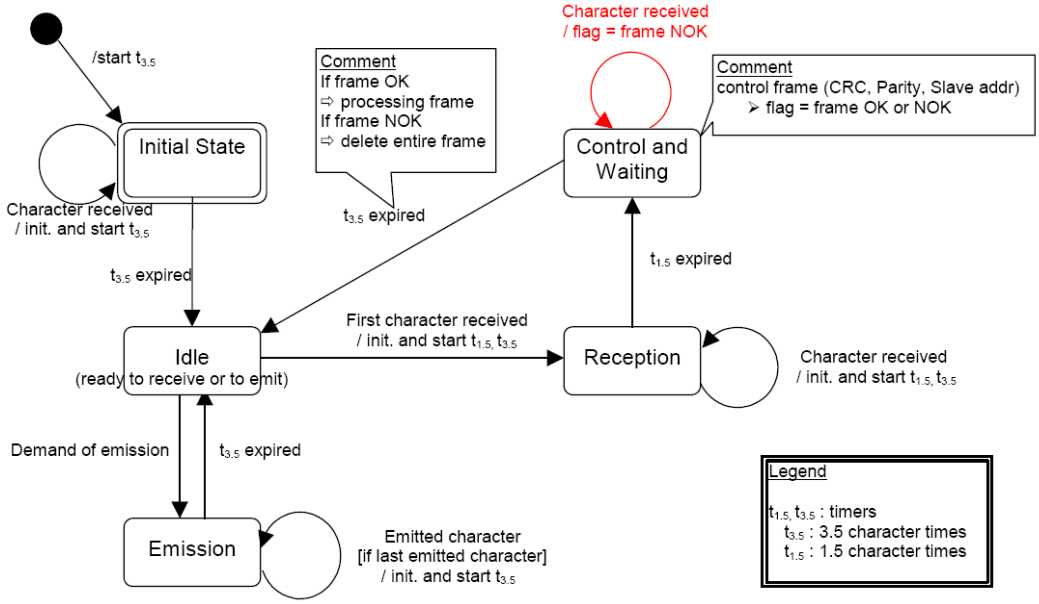
Základní datovou jednotkou linkové vrstvy je rámec, jejím úkolem je tedy práce s rámci. Základní tvar rámce protokolu Modbus je uveden na obrázku č. 18.



Obrázek 18. Základní tvar rámce protokolu Modbus [7]

Část PDU (Protocol Data Unit) obsahuje kód funkce a data, je nezávislá na linkové a fyzické vrstvě. Rozšířením PDU o adresu zařízení a kontrolní součet dostáváme tvar rámce používaný na sériových linkách.

Aplikace modulu pro měření teploty implementuje ve shodě s obrázkem č. 17 tři vrstvy referenčního modelu ISO/OSI. Fyzickou vrstvu tvoří galvanicky oddělená komunikační linka RS485. Linková vrstva je naprogramována v modulu *mb_linklayer.c*, je realizována jako jednoduchý stavový automat. Příjem i vysílání rámců probíhá zcela automaticky v kontextu obslužných rutin přerušení s minimálním nárokem na strojový čas procesoru. Stavový automat linkové vrstvy byl navržen dle následujícího obrázku.



Obrázek 19. Stavový diagram příjmu a vysílání rámců ModbusRTU [7]

Po provedení inicializace je aktivní stav Initial State, zároveň je nastaven časovač na dobu t_{35} (3,5 násobek doby přenosu jednoho znaku při zvolené přenosové rychlosti). Pokud dojde v době t_{35} k příjmu znaku, stavový automat zůstává ve stavu Initial State a časovač je opět nastaven na dobu t_{35}. Pokud uplyne doba t_{35} a nedojde k příjmu znaku, stavový automat přechází do stavu Idle (připravenost k příjmu a vysílání rámců). Prvním přijatým znakem stavový automat přechází do stavu Reception, zároveň je vynulován časovač. Pokud je čas mezi příjmem následujících

znaků menší, než t_{15} (1,5 násobek doby přenosu jednoho znaku při zvolené přenosové rychlosti), přijatý znak je uložen do bufferu a časovač je vynulován. Pokud je čas mezi příjmem znaků větší, než t_{15}, stavový automat přechází do stavu Control and Waiting (příjem rámce je ukončen, nebo nastal timeout). Ve stavu Control and Waiting je provedena kontrola CRC přijatého rámce, je porovnána adresa zařízení s adresou, obsaženou v rámci. Je-li CRC a adresa v pořádku v pořádku, přijatý rámec je zpracován, připravena odpověď a po uplynutí doby t_{35} stavový automat přechází do stavu Initial State. Je-li požadavek na vysílání rámce, stavový automat přechází do stavu Emission. V tomto stavu setrvává, dokud není odvysílán poslední byte odpovědi. Po uplynutí doby t_{35} dochází ke změně stavu na Idle, zařízení je připraveno k příjmu dalšího rámce. Detailní funkce je patrná ze zdrojového souboru mb_linklayer.c.

Aplikační vrstva je naprogramována v modulu *mb_applayer.c*. S linkovou vrstvou komunikuje pomocí funkcí aplikačního rozhraní, uvedených v hlavičkovém souboru *mb_linklayer.h*.

Implementovány jsou tyto standardní Modbus funkce:

- *Read Holding Registers*
- *Read Input Registers*
- *Write Single Register*

Tyto funkce slouží ke komunikaci s nadřazeným systémem (Master), umožňují vyčítat obsah registrů modulu (aktuální, maximální, minimální hodnota teploty atd.) Mapa registrů modulu pro měření teploty je uvedena v příloze č. 3 a č. 4.

Dále jsou implementovány následující uživatelské (nestandardní) funkce:

- *Read Eeprom* – čtení externí eeprom modulu
- *Write Eeprom* – zápis externí eeprom modulu
- *Software Reset* – programový reset modulu
- *Enable User Functions* – povolení uživatelsky definovaných funkcí

Uživatelské funkce jsou povoleny pouze během procesu výroby modulu, slouží pro jeho oživení (zavedení souboru se standardními daty do eeprom), kalibraci a diagnostiku. V běžném pracovním režimu modulu jsou tyto funkce zakázány.

5.9 ZPRACOVÁNÍ SIGNÁLU RTD SENZORU

Zpracování signálu RTD senzoru je realizováno v modulu *mereni.c*. Po připojení napájení je nejprve volána funkce *InitAdc()*. V rámci této funkce je zapnut a nastaven zdroj proudu RTD senzoru, jsou načteny kalibrační konstanty z eeprom, je inicializován A/D převodník a spuštěno měření. Vlastní měření probíhá ve třech fázích, ve kterých jsou periodicky měřeny tyto signály:

Označení	Význam	Mód A/D převodu	Nastavení MPX
D_R	Hodnota budícího proudu	Unipolární	AC_A2 proti AC_A0
D_0	Napětí při zkratovaných svorkách multiplexeru	Unipolární	AC_A0 proti AC_A0
D_A	Výstupní napětí RTD senzoru	Bipolární	AC_A7 proti AC_A6

Tabulka 3. Jednotlivé fáze měření

- D_R - údaj o hodnotě budícího proudu RTD Senzoru. Je měřen jako úbytek napětí na snímacím rezistoru R_{12} (viz příloha 1). Tento rezistor je zapojen v sérii s RTD senzorem a je tedy protékán stejným proudem jako RTD senzor.
- D_0 - údaj o hodnotě napětí při zkratovaných vstupních svorkách interního multiplexeru A/D převodníku mikrokontroléru XE8805A.
- D_A - údaj o hodnotě výstupního napětí RTD Senzoru.

Střídání jednotlivých fází měření je realizováno v přerušení a probíhá dle následujícího schématu:

D_0	**D_R**	**D_A**	8x D_A	**D_0**	8x D_A	**D_R**	8x D_A	**D_0**	8x D_A	**D_R**

Tabulka 4. Střídání fází měření

Po resetu jsou postupně změřeny všechny tři signály D0, DR a D_A. Dále je vždy 8 krát změřen kanál D_A (výstupní napětí RTD senzoru je měřeno nejčastěji), po té je střídavě změřena hodnota D_R a D_0.

Z těchto tří údajů je dle vztahu (5.1) vypočtena bezrozměrná normovaná hodnota $D_T = f(T)$, která nese informaci o poměru úbytku napětí na RTD senzoru ku úbytku napětí na snímacím rezistoru R_{12}:

$$D_T = \frac{D_A - D_0}{D_R - D_0} \qquad [-] \qquad (5.1)$$

Rozpětí normované veličeny je přibližně $D_T \in \langle 0,1 \rangle$. Pomocí vztahu (5.1) je docíleno kompenzace driftu zdroje proudu i driftu A/D převodníku.

Hodnota D_T je v dalším kroku výpočtu násobena kalibrační konstantou R_N, čímž je vypočítána přesná hodnota odporu RTD senzoru:

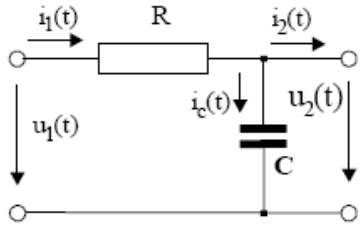
$$R = D_T \cdot R_N \qquad [\Omega] \qquad (5.2)$$

Kalibrační konstanta R_N je určena při procesu kalibrace, kdy je na vstup zařízení místo skutečného RTD senzoru připojen kalibrační normál s přesně definovanou hodnotou odporu. Je vyčtena hodnota D_T, vypočítána kalibrační konstanta R_N a uložena do eeprom. Protože závislost odporu RTD senzoru na teplotě není lineární, firmware modulu umožňuje linearizace převodní charakteristiky. Linearizovaná hodnota teploty je vypočtena pomocí algebraického polynomu:

$$T = a_4 R^5 + a_4 R^4 + a_3 R^3 + a_2 R^2 + a_1 R + a_0 \qquad [°C] \qquad (5.3)$$

Koeficienty polynomu jsou určeny dle požadovaného měřícího rozsahu při kalibraci zařízení a uloženy do eeprom.

K filtraci naměřeného údaje je použit dolnopropustný číslicový *IIR* filtr prvního řádu. Tento filtr je diskrétním ekvivalentem spojitého integračního článku.



Obrázek 20. Spojitý dolnopropustný filtr prvního řádu [8]

Přenos integračního článku v časové oblasti je popsán diferenciální rovnicí

$$RC \cdot \frac{du_2(t)}{dt} + u_2(t) = u_1(t) \qquad (5.4)$$

Časová konstanta systému je dána součinem hodnot odporu rezistoru R a kapacity kondenzátoru C.

$$t = RC \tag{5.5}$$

Převrácená hodnota časové konstanty udává zlomový kmitočet sytému. Laplaceovou transformací obou stran rovnice a vyjádřením poměru obrazu výstupu ku obrazu vstupu získáme operátorový přenos systému v oblasti komplexní roviny p.

$$F(p) = \frac{K}{tp+1} \tag{5.6}$$

Konstanta *K[-]* ve vztahu (5.6) udává statické zesílení systému, *t[s]* je časová konstanta systému. K nalezení operátorového přenosu diskrétního systému je nutné provést diskretizaci, tzn. převést přenos původního spojitého systému na ekvivalentní diskrétní přenos. Diskretizace je provedena následujícím postupem za použití tvarovače nultého řádu (Zero Order Holder).

1. Výpočet časového průběhu původního spojitého systému

$$y(t) = L^{-1}\left\{\frac{1}{p}\cdot\frac{K}{Tp+1}\right\} = K\cdot\left(1-e^{-\frac{t}{t}}\right) \tag{5.7}$$

2. Získání výstupní posloupnosti $y(k)$ navzorkováním původního spojitého signálu $y(t)$ s periodou T

$$y(k) = y(t)|_{t=KT} = K\cdot\left(1-e^{kT}\right) \tag{5.8}$$

3. Nalezení Z obrazu $Y(z)$ výstupní posloupnosti $y(k)$, výpočet ekvivalentního diskrétního přenosu pomocí vztahu (5.9).

$$F_e(z) = \frac{z-1}{z}\cdot Y(z) \tag{5.9}$$

$$Y(z) = Z\{y(kT)\} = K\cdot\left(\frac{z}{z-1} - \frac{z}{z-e^{-\frac{T}{t}}}\right) \tag{5.10}$$

$$F_e(z) = \frac{z-1}{z} \cdot K \cdot \left(\frac{z}{z-1} - \frac{z}{z-e^{-\frac{T}{t}}} \right) = K \cdot \frac{1-e^{-\frac{T}{t}}}{z-e^{-\frac{T}{t}}} \qquad (5.11)$$

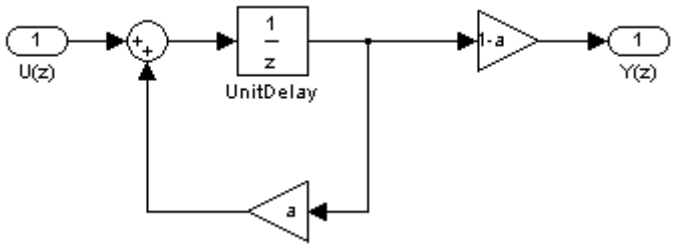
Diskretizovaný přenos původního spojitého integračního článku je tedy

$$F_e(z) = \frac{b}{z-a}, \quad \text{kde } a = e^{-\frac{T}{t}}, \; b = 1-a \qquad (5.12)$$

Z nalezeného ekvivalentního přenosu (5.12) je dále určena diferenční rovnice systému, ze které je následně odvozen algoritmus programové realizace diskrétního filtru. Diferenční rovnice systému je

$$y(k+1) = (1-a) \cdot u(k) + a \cdot y(k) \qquad (5.13)$$

Filtr byl programově realizován dle následujícího simulačního schématu.



Obrázek 21. Simulační schéma číslicového filtru

V aplikaci modulu pro měření teploty je časová konstanta filtru uživatelsky nastavitelná (položka 4 „FILTER“ lokálního menu) v rozsahu 0,1 až 100 [s]. To znamená, že hodnota koeficientu a v diferenční rovnici (5.13) musí být přepočítána dle aktuálně zadané hodnoty časové konstanty. Mezi časovou konstantou a hodnotou koeficientu a a platí následující vztah:

$a = e^{-\frac{T}{t}}$, kde T je perioda vzorkování, t je požadovaná časová konstanta (5.14)

5.10 LOKÁLNÍ MENU MODULU PRO MĚŘENÍ TEPLOTY

Ovládání a jednotlivé funkce menu jsou naprogramovány v modulu *menu.c*. K orientaci v menu přístroje slouží tři tlačítka (vlevo, vpravo, enter). Zmáčknutím každého tlačítka (sestupnou hranou) je generováno externí přerušení, v něm je nastavena stavová proměnná menu. V hlavní smyčce je dle hodnoty stavové proměnné menu rozlišeno, které tlačítko bylo zmáčknuto a je provedena patřičná reakce. Menu obsahuje celkem 11 položek.

1. **PASSW** – tato položka slouží k povolení/zakázání ostatních položek. Při aktivaci této položky musí uživatel nejprve zadat čtyřmístné číslo. Pokud je zadáno správně, může uživatel povolit nebo zakázat ostatní položky menu.
2. **DECP** – nastavení pozice desetinné tečky (počtu desetinných míst při zobrazování naměřeného údaje)
3. **FILTER** – nastavení časové konstanty dolnopropustného filtru (0,1 až 100 [s])
4. **ADDR** – interní adresa modulu pro komunikaci protokolem ModbusRTU, rozsah nastavení adresy je 1 až 247.
5. **BAUD** – nastavení přenosové rychlosti asynchronního sériového rozhraní UART. Podporovány jsou tyto přenosové rychlosti: 4800, 9600, 19200, 38400 Bd.
6. **BIAS** – zapnutí/vypnutí předpětí komunikační linky RS485.
7. **TERMIN** - zapnutí/vypnutí zakončení komunikační linky RS485.
8. **UNIT** – nastavení jednotky teploty. Podporovány jsou tyto jednotky: °C, °K, °F.
9. **OFFSET** – zadání offsetu měření teploty. Tato hodnota je vždy odečítána od naměřené hodnoty teploty. Slouží ke kompenzaci aditivní chyby měření, nebo k uživatelskému posunu offsetu měřeného údaje.
10. **TMIN** – zobrazení minimální naměřené hodnoty teploty. Po zapnutí je tato hodnota inicializována prvním naměřeným údajem.
11. **TMAX** – zobrazení maximální naměřené hodnoty teploty. Po zapnutí je tato hodnota inicializována prvním naměřeným údajem.

6. KOMUNIKAČNÍ STANDARD ETHERNET

6.1 ÚVOD

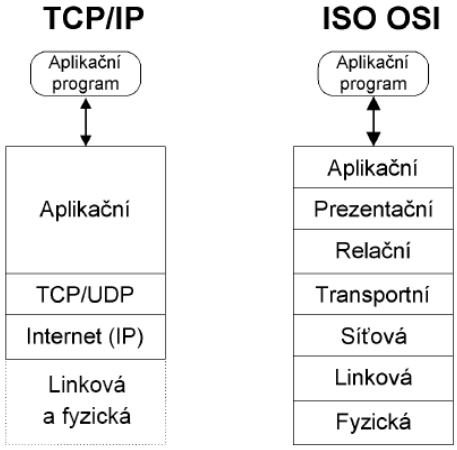
První verze Ethernetu byla vyvinuta na počátku 70. let firmou Xerox. Úkolem vývojového týmu bylo vzájemně propojit pracovní stanice a tiskárny vyráběné firmou Xerox. První verze dosahovala přenosové rychlosti 2,94 Mbps. V roce 1978 se k firmě Xerox připojili firmy DEC a Intel, které projekt společně financovali a pokračovali v dalším vývoji. Nová verze, která vznikla v roce 1980, nesla označení DIX Ethernet, přenosová rychlost byla zvýšena na 10 Mbps. Zásadní pro další vývoj bylo společné rozhodnutí firem DEC, Intel a Xerox neponechat si Ethernet jako své proprietární řešení. Společně předali veškeré specifikace standardizační organizaci IEEE, která předložený koncept v pozměněné podobě přijala a v roce 1985 uvedla jako standard IEEE 802.3 CSMA/CD. Tento standard je organizací IEEE rozvíjen v nejrůznějších podobách (fast Ethernet, Gigabit Ethernet, 10 Gigabit Ethernet) dodnes. Zajímavostí je, že firma Xerox si zaregistrovala Ethernet jako svou obchodní značku. V rámci standardů IEEE 802.3 se tedy hovoří o sítích na bázi CSMA/CD, označení Ethernet je čistě neformální [9][10][11].

Pro přístup ke sdílenému médiu využívá standard IEEE 802.3 nedeterministickou přístupovou metodu CSMA/CD – metoda mnohonásobného přístupu k médiu s detekcí nosné a kolize. Tato metoda je založena na principu odposlechu nosné (Carrier Detect). Stanice před zahájením vysílání zjišťuje (odposlouchává) stav média. Je-li detekována nosná (médium obsazeno), stanice počká na ukončení právě probíhajícího přenosu a poté okamžitě zahájí vysílání. Pokud je médium volné, stanice začíná vysílat a v průběhu vysílání kontroluje, jestli nedošlo ke kolizi (Collision Detection). Pokud zahájí vysílání dvě stanice současně, dochází ke stavu kolize, který trvá po celou dobu přenosu. V takové případě stanice vysílá “Jam signál” (krátký signál, obsahující bezvýznamná data) a vysílání ukončí. Po uplynutí náhodně vygenerovaného časového intervalu se stanice pokouší vysílání opakovat (backoff algoritmus). S vzrůstajícím zatížením sítě se zvyšuje

pravděpodobnost výskytu kolizí. Jako rozumné se ukázalo 20% zatížení Ethernetu [9][10][11].

6.2 SÍŤOVÝ MODEL TCP/IP

Architektura TCP/IP je založena na čtyřvrstvém modelu. Tímto se odlišuje od referenčního modelu ISO/OSI, který vznikl později a definuje sedm vrstev. Filosofie obou modelů je odlišná, přímé odvození modelu TCP/IP z referenčního modelu ISO/OSI proto není možné. Srovnání obou modelů je uvedeno na následujícím obrázku 19. Síťový model TCP/IP neřeší (až na výjimky, jako je SLIP) linkovou a fyzickou vrstvu [11].



Obrázek 22. Srovnání modelů TCP/IP a ISO/OSI [11]

6.2.1 IP protokol

IP protokol přibližně odpovídá síťové vrstvě referenčního modelu ISO/OSI. Úkolem IP protokolu je doprava dat mezi dvěma libovolnými síťovými uzly, které jsou určeny IP adresami. Základní datovou jednotkou IP protokolu je IP datagram, který se skládá z IP záhlaví a dat. IP datagramy jsou od zdrojového k cílovému uzlu dopravovány pomocí směrovačů (routerů), kterých může být na přenosové cestě celá řada. Úlohou každého směrovače je doprava datagramu k následujícímu směrovači nebo cílovému uzlu. IP protokol dále zajišťuje fragmentaci datagramů v případě, že velikost datagramu je větší než maximální velikost linkového rámce (MTU), který je následující uzel schopen přijmout. Velikost MTU stanoví příslušná linková vrstva. IP protokol provádí také případnou defragmentaci datagramů na straně příjemce [11].

6.2.2 TCP protokol

Protokol TCP přibližně odpovídá transportní vrstvě referenčního modelu ISO/OSI. Protokol TCP zajišťuje přenos dat mezi konkrétními aplikacemi, běžícími na síťových uzlech. Základní datovou jednotkou protokolu TCP je paket, který se skládá z TCP záhlaví a přenášených dat. Protokol TCP je spojově orientovaný. Nejprve je mezi zdrojovou a cílovou aplikací navázáno virtuální spojení, pomocí kterého se pak uskutečňuje potvrzovaný přenos dat. Zdrojová i cílová aplikace je jednoznačně určena číslem portu. Protokol TCP zabezpečuje přenos před ztrátou nebo modifikací dat, způsobenou poruchou technických prostředků [11].

6.2.3 UDP protokol

Protokol UDP spadá podobně jako protokol TCP do transportní vrstvy, je jeho jednodušší obdobou. Základní jednotkou protokolu UDP je UDP datagram. Protokol UDP není na rozdíl od protokolu TCP spojově orientovaný. To znamená, že nedochází k navazování spojení ani potvrzování přenášených dat. Zdrojová aplikace posílá datagram cílové aplikaci, protokol UDP negarantuje jeho doručení. Přenos může probíhat rychleji, než u protokolu TCP. Správnost přenosu však musí být ověřena aplikací. Zdrojová i cílová aplikace je stejně jako u protokolu TCP jednoznačně určena číslem portu [11].

6.2.4 Aplikační vrstva

Aplikační vrstva architektury TCP/IP sdružuje funkce třech nejvyšších vrstev referenčního modelu ISO/OSI. Spadá do ní velké množství protokolů, které zajišťují aplikacím přístup k různým internetovým službám. Mezi nejčastěji používané protokoly aplikační vrstvy patří:

- HTTP – přenos hypertextových dokumentů
- FTP – vzdálený přenos souborů
- SMTP – protokol elektronické pošty
- IMAP – přístup do schránek el. pošty ze vzdálených počítačů
- SNMP – správa a koordinace síťových uzlů

7. NÁVRH KOMUNIKAČNÍHO MODULU ETHERNET

7.1 SOUČASNÉ MOŽNOSTI ŘEŠENÍ

V současné době jsou možnosti připojení průmyslových převodníků a vestavných systémů k Ethernetu velmi široké. Téměř každý výrobce mikrokontrolérů (Atmel, Microchip, Freescale, Texas Instrument, atd.) nabízí hned několik řešení vyhovujících standardu IEEE 802.3. Obvyklé je použití mikrokontroléru v kombinaci s externím ethernetovým řadičem. Externí řadič řeší fyzickou (PHY) a linkovou (MAC) vrstvu, síťová a transportní vrstva je realizována pomocí speciálního software (TCP/IP stack) v mikrokontroléru. Existují však také jednočipová řešení (Ubicom, Microchip). Výrobci mikrokontrolérů poskytují plnou podporu vývoje ethernetových aplikací. Pro své platformy nabízí integrovaná vývojová prostředí, operační systémy reálného času, TCP/IP stack včetně protokolů aplikační vrstvy, hardwarové kity. Dostupné jsou také hotové moduly určené k zástavbě do zákaznické aplikace. Tyto moduly poskytují kompletní hardwarové řešení, software (firmware) může být vyvinut zákazníkem na míru dle potřeb konkrétní aplikace.

Na trhu jsou k dispozici také hotová řešení v podobě universálních převodníku běžných typů komunikačních linek (RS232C, RS485, RS422, atd.) na Ethernet. Tyto převodníky mají množství zákaznicky konfigurovatelných parametrů (profilů) pro přizpůsobení potřebám nejrůznějších aplikací. Jejich výhodou je především možnost okamžitého nasazení do provozu bez nutnosti jakýchkoliv vývojových prací a detailních znalostí. Tyto převodníky bývají obvykle vybaveny také analogovými vstupy, takže umožňují přímé připojení průmyslových převodníků fyzikálních veličin k Ethernetu. Nevýhodou je, že firmware takových zařízení nemůže být modifikován zákazníkem. To znemožňuje použití v aplikacích, které vyžadují specifickou funkcionalitu, práci se sokety, generování webových stránek s dynamickým obsahem atd.

7.2 VÝBĚR OPTIMÁLNÍHO ŘEŠENÍ

Při výběru platformy pro připojení k ethernetu je nutné vycházet z požadavků zadání konkrétní aplikace. Na základě těchto požadavků pak zvážit tyto klíčové body:

- nároky na výpočetní výkon
- nároky na paměťový subsystém
- kusová cena konečného řešení
- celkové náklady na vývoj
- rozměry
- proudová spotřeba

Součástí řešení komunikačního modulu Ethernet je implementace protokolu ModbusTCP a webového serveru. Z těchto požadavků vyplývá, že vybraná platforma musí umožňovat vytvoření zákaznického firmware. Z výběru jsou proto předem vyloučena zařízení typu převodníky linek na Ethernet. Z důvodu usnadnění vývoje budou při výběru preferována hotová hardwarová řešení v podobě vestavných modulů. Odpadne tak nutnost vývoje hardware, bude použito odladěné a otestované řešení. Další výhodou vestavných modulů je, že jsou dodávány včetně MAC adresy. Bylo vybíráno z těchto produktů:

1. XPort Architekt, Lantronix
2. FoxBoard LX, HW Server
3. Charon II, HW Server
4. Spinet, Asix
5. Digi Connect ME® 9210, DIGI
6. RCM3200, Rabbit
7. Ethernut II, Ethernut

Po zvážení požadavků aplikace se jako nejvýhodnější ukázalo použití vestavného modulu RCM3200 firmy Rabbit. Výkon i velikost pamětí tohoto modulu jsou pro aplikaci zcela dostatečné, vyhovující je také jeho cena, rozměry i proudová spotřeba. Další výhodou je, že firma Rabbit poskytuje zdarma vývojové prostředí Dynamic C, čímž se minimalizují vývojové náklady.

7.2.1 Modul Rabbit RCM3200

RCM3200 je vestavný modul určený pro připojení zákaznických zařízení k Ethernetu. Základní vlastnosti modulu jsou uvedeny v následujícím seznamu.

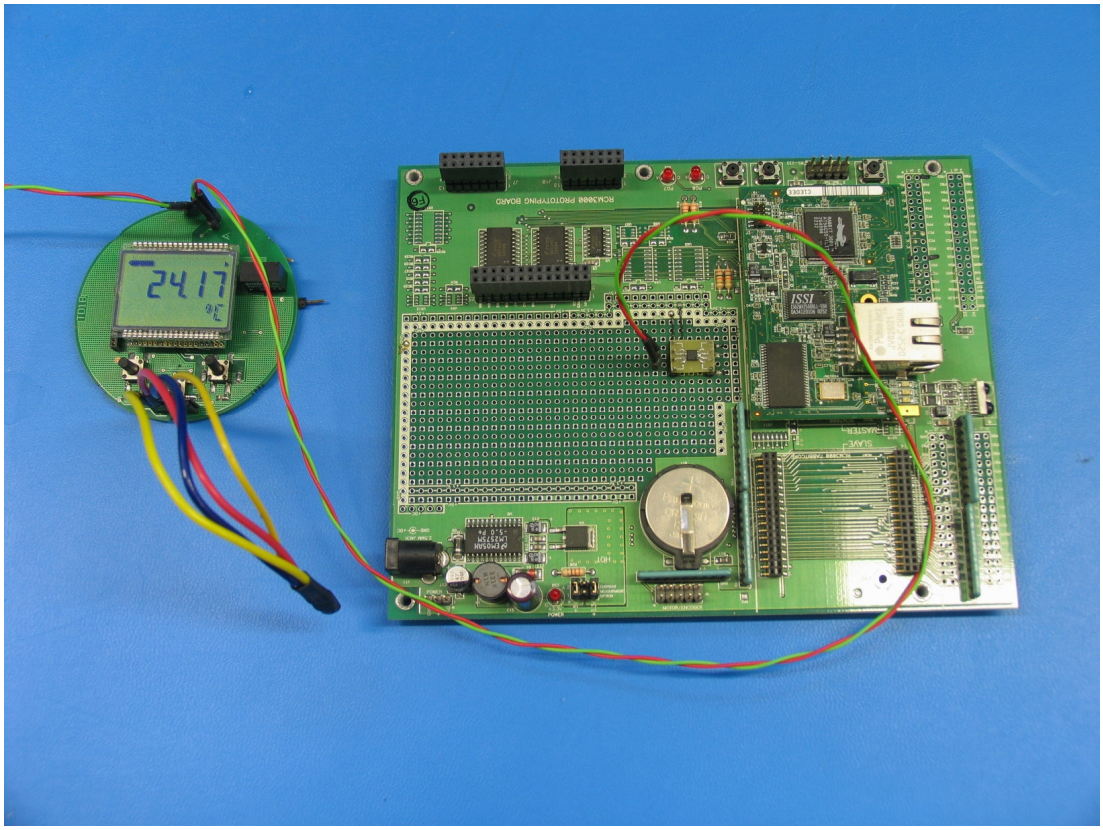
- mikroprocesor Rabbit3000 @ 44,2 MHz
- Ethernet port 10/100Base-T
- 52 vstupně/výstupních linek
- 512kB paměti flash, 512kB programové SRAM, 256kB datové SRAM
- hodiny reálného času zálohované baterií
- 6 univerzálních sériových portů (UART, SPI, SDLC/HDLC)
- rozměry 47x69x22 mm
- maximální odběr 255mA

Obrázek 23. Modul RABBIT RCM3200 [12]

Modul RCM3200 je velmi flexibilní a kompaktní, může být použit v mnoha typech aplikacích, např. převodníky komunikačních linek na Ethernet, ovládání vzdálených periferií a zařízení přes Ethernet, docházkové a dohledové systémy, datové koncentrátory a další. Existují i výkonnější modely, postavené na mikroprocesorech Rabbit4000 a Rabbit5000. Kromě Ethernetu jsou podporovány také bezdrátové komunikační technologie WiFi a ZigBee.

7.3 NÁVRH A REALIZACE HARDWARE KOMUNIKAČNÍHO MODULU EHTERNET

Pro realizaci hardware byla použita vývojová deska Rabbit RCM30/31/32XX, do které byl zasunut modul RCM3200. Do pájivého pole vývojové desky byl osazen budič sběrnice RS485 MAX3471 a připojen k asynchronnímu sériovému rozhraní UART D modulu RCM3200.

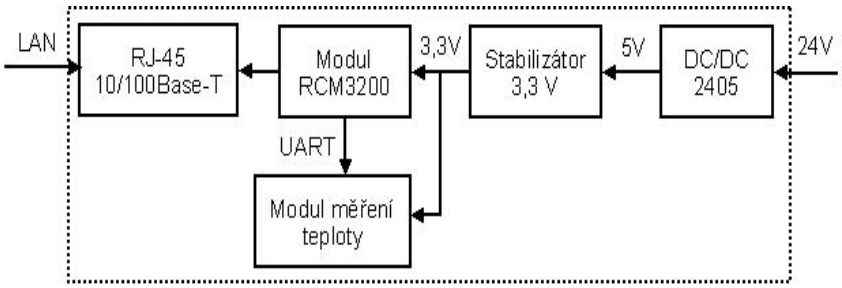


Obrázek 24. Realizace prototypu na vývojové desce

Tato sestava není určena k nasazení do skutečného provozu, slouží pouze k vývoji a realizaci prototypu pro účely této diplomové práce. Pro nasazení do průmyslového provozu připadají v úvahu dvě varianty popsané v kapitolách 7.3.1 a 7.3.2. Detailní obvodové řešení není zahrnuto, jsou uvedena pouze bloková schémata.

7.3.1 Varianta č. 1, snímač teploty s výstupem Ethernet

V tomto provedení je modul měření teploty integrován s komunikačním modulem RCM 3200 na jedné desce plošných spojů a vestavěn do společného pouzdra. Komunikace mezi oběma moduly probíhá pomocí asynchronního rozhraní UART protokolem ModbusRTU.

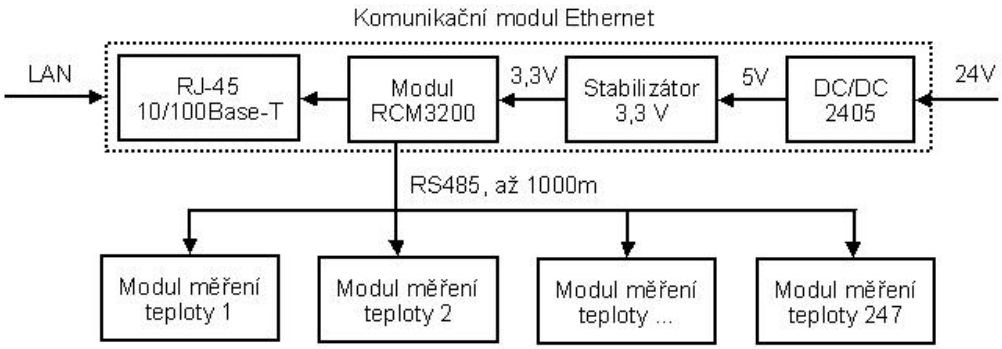


Obrázek 25. Blokové schéma varianty č. 1

Tato varianta je určena pro použití v aplikacích typu řízení budov (monitorování teplot v kancelářích, servrovnách, atd.), kde je již instalována síť LAN (intranet). Výhodou této varianty je kompaktnost provedení, nevýhodou je neefektivní využití modulu RCM3200 (jeden síťový uzel pro jedno měřené místo) a z toho vyplývající vyšší cena v přepočtu na jedno měřené místo.

7.3.2 Varianta č. 2, převodník protokolů ModbusRTU/ModbusTCP

V tomto provedení je modul pro měření teploty oddělen od komunikačního modulu. Modul pro měření teploty je určen k přímé instalaci na měřeném místě, je napájen z vlastního zdroje. Komunikační modul určen k instalaci do rozváděčové skříně, s modulem pro měření teploty je propojen sběrnicí RS485. Komunikace mezi moduly probíhá protokolem ModbusRTU.



Obrázek 26. Blokové schéma varianty č. 2

Hlavní předností této varianty je možnost připojení více modulů pro měření teploty (Slave) k jednomu komunikačnímu modulu (Master). Počet měřených míst je 1 až 247. Tento počet je dán specifikacemi protokolu ModbusRTU. Pro slave adresu je v rámci protokolu ModbusRTU rezervován pouze jeden byt, tzn. maximálně 256 zařízení. Specifikace však dále stanovuje, že rozsah adresy pro slave zařízení je právě 1 až 247.

8. IMPLEMENTACE FIRMWARE KOMUNIKAČNÍHO MODULU ETHERNET

8.1 ÚVOD

Firmware komunikačního modulu byl vytvořen a odladěn v prostředí Dynamic C [13]. Dynamic C je volně stažitelné integrované vývojové prostředí (IDE) pro mikroprocesory Rabbit. Prostředí podporuje standardní funkce editace, kompilování linkování, nahrávání a ladění kódu. Součástí vývojového prostředí jsou tyto softwarové moduly:

- operační systém reálného času uC/OS-II
- TCP/IP stack
- protokoly aplikační vrstvy HTTP, FTP, TFTP, SMTP, SNMP, SNTP, Telnet, ModbusTCP
- souborový systém FAT
- podpora pro externích pamětí dataflash, paměťových karet SD
- podpora dotykových displejů
- komunikace s GPS moduly

Výhodou vývojového prostředí Dynamic C je podpora operačního systému reálného času (uC/OS-II). V prostředích pro mikroprocesory (mikrokontroléry), které nemají dostatečné technické prostředky pro implementaci operačního systému není možné rozdělit běh programu do více úloh. Celá aplikace je spuštěna jako jediná úloha, běžící v kontextu hlavní programové smyčky. Hlavní smyčka je obvykle realizována jako stavový automat, který periodicky testuje stavové proměnné aplikace. Hodnoty stavových proměnných jsou modifikovány v obslužných rutinách přerušení. Tímto způsobem byl realizován firmware modulu pro měření teploty popsaný v kapitole 5. V platformách s operačním systémem reálného času je filosofie programování odlišná. Běh programu je možné rozdělit do procesů, kterým jádro operačního systému přiděluje systémové prostředky na základě jejich priorit (kooperativní operační systém). Firmware komunikačního modulu Ethernet využívá služeb

operačního systému uC/OS-II. Firmware byl vytvořen s ohledem na obě hardwarové varianty popsané v kapitolách 7.3.1 a 7.3.2.

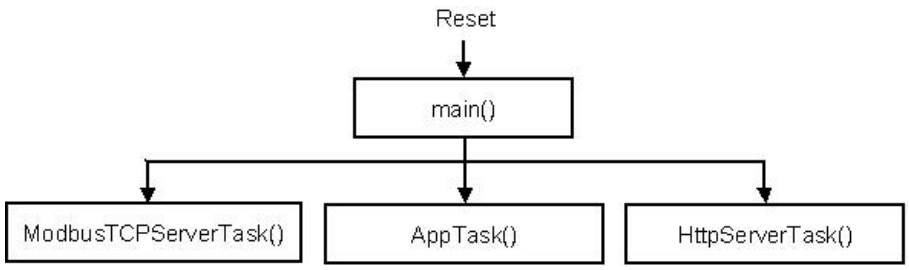
8.2 SEZNAM ZDROJOVÝCH SOUBORŮ FIRMWARE

Součástí firmware komunikačního modulu jsou následující zdrojové soubory.

- **main.c** – Inicializace a spuštění procesů aplikace.
- **application.c** – hlavní proces aplikace, realizující periodické vyčítání registrů měřícího modulu a aktualizaci registrů ModbusTCP serveru, HTTP server pro zobrazení naměřených hodnot a nastavení parametrů zařízení přes webové rozhraní.
- **modbus_rtu_master.c** – implementace ModbusRTU Master
- **modbus_tcp_server.c** – implementace ModbusTCP serveru. Podporováno je 5 klientských připojení.
- **modbus_utils.c** – společné funkce, převod čísel reprezentovaných ve tvaru malého endiánu na velký a naopak

8.3 START APLIKACE

Po resetu a provedení nízkoúrovňové inicializace je volána C funkce *main*. V rámci funkce *main* je provedena inicializace vývojové desky RCM30/31/32XX, operačního systému uC/OS-II a TCP/IP stacku. Po té jsou inicializovány a spuštěny procesy aplikace.



Obrázek 27. Grafické znázornění startu aplikace

8.4 HLAVNÍ ČÁST APLIKACE

Hlavní část aplikace je naprogramována v souboru *application.c*, je tvořena dvěma procesy. Proces *AppTaks* periodicky vyčítá hodnoty vstupních (Input) a udržovacích (Holding) registrů z modulu měření teploty. Tento proces má nejvyšší prioritu. Ke komunikaci využívá služeb souboru *modbus_rtu_master.c*, ve kterém je programově realizována jednotka Master protokolu ModbusRTU. Hodnoty jsou vyčítány každých 500 ms a ukládány do datových struktur *InputRegs* a *HoldingRegs* (identické se strukturami měřícího modulu).

```
//----------------------
//Hlavni uloha aplikace
//----------------------
void AppTask(void *ptr)
{
   auto uint8 rc, i;

   //Periodicke vycitani Input Registers z merici desky TD485
   while(true)
   {
      //Povoleni uzivatesky definovanych funkci
      rc = mbEnableUserFce(0x01);

      //Vycteni vsech vstupnich registru mericiho modulu TD485
      rc = mbReadInputRegs(0x01, 0x0000, 0x0021, (uint16*)&InputRegs);
      if(!rc)
      {
         //Zmena poradi wordu 32. bitovych hodnot
         SwapUint32((uint16*)&InputRegs.SerialNr, 0x09);
         SwapUint32((uint16*)&InputRegs.Dn, 0x06);
      }

      //Vycteni vsech pridrznych registru z merici desky TD485
      rc = mbReadHoldingRegs(0x01, 0x0000, 0x005, (uint16*)&HoldingRegs);

      //Uspani ulohy na 500 ms
      OSTimeDlyHMSM(0, 0, 0, 500);
   }
}
```

Obrázek 28. Hlavní proces aplikace

V souboru *application.c* jsou dále implementovány funkce aplikační vrstvy protokolu ModbusTCP. Funkce *ReadHoldingRegs* a *ReadInputRegs* umožňují nadřazenému systému (ModbusTCP klient) vyčítat hodnoty vstupních (struktura *InputRegs*) a udržovacích (struktura *HoldingRegs*) registrů komunikačního modulu, které obsahují údaje vyčtené z měřícího modulu. Tyto funkce jsou volány ze souboru *modbus_tcp_server.c*.

Druhým procesem je proces *HttpServerTask*. Nejprve je inicializován HTTP server a rezervován TCP port 80. Po té je v rámci nekonečné smyčky periodicky volána funkce *http_handler*, která zajišťuje provoz HTTP serveru.

```
//-----------------
//Http Server task
//-----------------
void HttpServerTask(void *ptr)
{
   //Inicializace HTTP serveru, rezervace portu 80
   http_init();
   tcp_reserveport(80);

   while(true)
      http_handler();
}
```

Obrázek 29. Proces HTTP serveru

8.5 IMPLEMENTACE PROTOKOLU MODBUSRTU (MASTER)

Jednotka Master protokolu ModbusRTU je realizována ve zdrojovém souboru *modbus_rtu_master.c*. Nejprve je nutné provést inicializaci voláním funkce *InitModbusRTUMaster()*. V této funkci je otevřen sériový port D mikroprocesoru Rabbit3000. Parametry komunikace jsou 38400 Bd, 8 datových bitů, lichá parita, 1 stop bit. Řízení přenosu linky RS485 je nastaveno na příjem. Zdrojový soubor *modbus_rtu_master.c* umožňuje aplikaci komunikaci s modulem pro měření teploty (a jinými Slave zařízeními). Poskytuje funkce na úrovni linkové vrstvy, sestavuje, vysílá a přijímá rámce protokolu ModbusRTU.

8.5.1 Funkce WriteDownstreamDevice

*uint8 WriteDownstreamDevice(uint8 *Buffer, uint8 *Len)*

WriteDownstreamDevice je základní funkce, která slouží pro komunikaci (vysílání a příjem rámců) se Slave zařízeními připojenými k lince RS485 komunikačního modulu. Argumenty funkce jsou ukazatele na požadavek (Slave adresa + PDU) a jeho délku. Funkce *WriteDownstreamDevice* nejprve vypočte kontrolní součet a doplní jej do požadavku, čímž dokončí sestavení ADU protokolu Modbus pro sériové linky. Funkce dále vynuluje oba kruhové buffery sériového portu D. Po té je směr přenosu linky RS485 přepnut na vysílání, do vysílacího kruhového bufferu je zkopírován obsah požadavku a zahájen přenos. Funkce dále periodicky testuje, kolik bytů zbývá v kruhovém bufferu k vyslání. Je-li počet bytů vyšší, než nula (vysílání požadavku stále probíhá), volající proces je blokován na dobu 2 ms (testování cyklicky pokračuje, dokud není kruhový buffer prázdný). Doba 2 ms byla zvolena s ohledem na dobu přenosu jednoho znaku. Pokud je počet bytů v kruhovém bufferu roven nule, funkce *WriteDownstreamDevice* provádí cyklický test bitu 2 stavového registru SDRS sériového kanálu D. Teprve, až je tento bit vynulován je zaručeno, že byl odvysílán i poslední byt požadavku a je možné bezpečně přepnout řízení přenosu linky RS485 na příjem. Funkce dále čeká (blokuje volající proces) po dobu 100 ms na odpověď Slave zařízení. V případě, že je v kruhovém přijímacím bufferu dostatečný počet bytů, funkce *WriteDownstreamDevice* provede výpočet a porovnání kontrolního součtu. Je-li kontrolní součet v pořádku, odpověď Slave zařízení je zkopírována na adresu reprezentovanou argumentem *Buffer* a do proměnné *Len* je zapsán počet bytů odpovědi, návratová hodnota funkce je rovna nule. Pokud není v přijímacím bufferu dostatek bytů, Slave zařízení neodpovědělo časovém limitu (nastal Timeout) a návratová hodnota indikuje timeout.

8.5.2 Funkce mbReadHoldingRegs

*uint8 mbReadHoldingRegs(uint8 UnitID, uint16 Addr, uint16 Quantity, uint16 *Regs)*

Tato funkce realizuje standardní funkci #3 protokolu Modbus (funkce #3 – Read Holding Registers). Aplikaci umožňuje vyčtení hodnot uchovávacích registrů Slave zařízení. Argumenty funkce jsou adresa Slave zařízení, adresa prvního vyčítaného registru, počet registrů a adresa pro uložení hodnot vyčtených registrů. Funkce *mbReadHoldingRegs* provede na základě hodnot argumentů sestavení PDU protokolu Modbus do globální proměnné *mbFrame*. Dále je volána funkce *WriteDownstreamDevice*, která doplní rámec o kontrolní součet a provede jeho vyslání na sběrnici RS485. Pokud je odpověď Slave zařízení v pořádku (nedošlo k Timeoutu, chybě CRC ani výjimce), funkce *mbReadHoldingRegs* zkopíruje hodnoty vyčtených registrů na adresu reprezentovanou argumentem *Regs*.

8.5.3 Funkce mbReadHoldingRegs

*uint8 mbReadInputRegs(uint8 UnitID, uint16 Addr, uint16 Quantity, uint16 *Regs)*

Tato funkce realizuje standardní funkci #4 protokolu Modbus (funkce #4 – Read Input Registers). Aplikaci umožňuje vyčtení hodnot vstupních registrů Slave zařízení. Průběh transakce je shodný s funkcí *mbReadHoldingRegs*, popsanou v kapitole 8.4.3.

8.5.4 Funkce mbWriteSingleRegister

uint8 mbWriteSingleRegister(uint8 UnitID, uint16 Addr, uint16 Value)

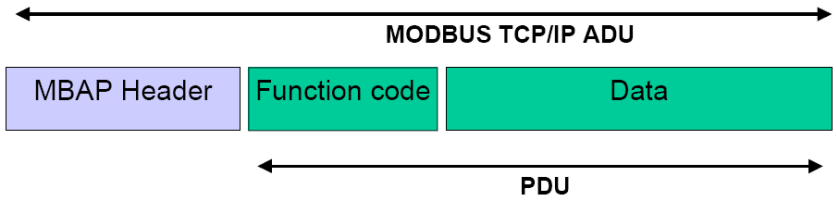
Tato funkce realizuje standardní funkci #6 protokolu Modbus (funkce #6 – Write Single Register). Aplikaci umožňuje zápis hodnot jednoho udržovacího registru Slave zařízení. Průběh transakce je obdobný, jako u předchozích funkcí.

8.6 IMPLEMENTACE SERVERU MODBUSTCP

Server protokolu ModbusTCP je realizován ve zdrojovém souboru *modbus_tcp_server.c*.

8.6.1 Úvod

Tvar požadavku (odpovědi) protokolu ModbusTCP je znázorněn na následujícím obrázku.



Obrázek 30. Tvar požadavku (odpovědi) protokolu ModbusTCP [14]

K části PDU, která je nezávislá na nejnižších vrstvách modelu ISO/OSI je přidáno aplikační záhlaví MBAP Header (Modbus Application Protocol Header). Stavba aplikačního záhlaví MBAP Header je uvedena následujícím obrázku.

Fields	Length	Description -	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the client	Recopied by the server from the received request
Length	2 Bytes	Number of following bytes	Initialized by the client (request)	Initialized by the server (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client	Recopied by the server from the received request

Obrázek 31. Stavba aplikačního záhlaví MBAP Header [14]

- Transaction Identifier – identifikátor transakce je náhodně vygenerován klientem, server tuto hodnotu kopíruje do odpovědi
- Protocol Identifier – identifikátor protokolu, je vždy nastaven klientem na hodnotu 0 (Modbus protokol), server opět kopíruje z požadavku do odpovědi
- Length – počet následujících bytů v požadavku, při odpovědi je tato položka nastavena serverem, dle počtu bytů odpovědi
- Unit Identifier – adresa zařízení. Pokud je hodnota této položky v intervalu 1 až 247, požadavek je určen pro Slave zařízení připojená k sériové lince ModbusTCP serveru. Server sestaví rámec protokolu ModbusRTU (Unit Idetifier + PDU + CRC) a vyšle jej na sériovou linku. Pokud je hodnota Unit Identifier rovna 0 nebo 255, požadavek je určen pro zpracování ModbusTCP serverem.

8.6.2 Implementace

Zdrojový soubor *modbus_tcp_server.c* poskytuje aplikaci následující funkce. Nejprve je nutné provést inicializaci voláním funkce *InitModbusTCP()*. V této funkci je rezervován TCP port 502 a vytvořen proces *ModbusTCPServerTask*. Tento proces čeká na klientská připojení naslouchánim na portu 502, současně může být připojeno maximálně 5 klientů. Proces *ModbusTCPServerTask* funguje jako stavový automat, jeho detailní funkce je patrná ze zdrojového kódu. Jednotlivá spojení jsou udržována pomocí struktury *MODBUS_CONN*.

```
//--------------------
//ModbusTCP connection
//--------------------
typedef struct
{
   tcp_Socket Sock;         //TCP socket
   uint8 State;             //Stav pripojeni
   uint8 Buffer[0x0100];    //Buffer pro prijem pozadavku a vysilani odpovedi
   uint16 ClientIddle;      //Doba neaktivity klienta
}MODBUS_CONN;
```

Obrázek 32. Definice struktury MODBUS_CONN

Člen *Sock* struktury identifikuje TCP socket, který je použit pro připojení s klientem. Tato hodnota je vždy předávána jako jeden z argumentů funkcím, určeným pro práci se sockety (*tcp_listen*, *sock_established*, *sock_bytesready*, atd.). Člen *State*

je stavová proměnná, určující stav každého klientského připojení. Člen *Buffer* (pole 256. bytů) slouží jako komunikační buffer pro ukládání požadavků a odpovědí. Člen *ClientIddle* udává, jak dlouho je klient po navázání spojení neaktivní (nezasílá požadavky). Po překročení nastavené doby je spojení s daným klientem aktivně uzavřeno.

Přijaté požadavky jsou v kontextu procesu *ModbusTCPServerTask* zpracovány funkcí *int mbProcessRequest(uint8 *Buffer)*. Na základě hodnoty položky Unit Identifier aplikačního záhlaví je nejprve stanoveno, zda je požadavek určen pro Slave zařízení připojená na linku RS485 komunikačního modulu, nebo přímo pro ModbusTCP server. V případě že je hodnota Unit Identifier v intervalu 1 až 247, je požadavek počínaje položkou Unit Identifier předán funkci *WriteDownstreamDevice* (kapitola 8.5.1). Tato funkce doplní požadavek o kontrolní součet a vyšle na sběrnici RS485. Pokud Slave zařízení odpoví v časovém limitu, je k jeho odpovědi přidáno MBAP záhlaví a vše je zasláno zpět na stanici klienta. Pokud je hodnota Unit Identifier rovna 0 nebo 255, požadavek je zpracován ModbusTCP server, který podporuje následující funkce:

- *Read Holding Registers*
- *Read Input Registers*

8.7 IMPLEMENTACE WEBOVÝCH STRÁNEK

Webové stránky umožňují vyčítání naměřených hodnot, nastavení parametrů měřícího i komunikačního modulu v prostředí webového prohlížeče. Webové stránky byly implementovány dle aplikačního dokumentu [15] a vzorových aplikací (samples), dostupných po instalaci vývojového prostředí Dynamic C. Implementace skládá ze tří kroků, importování souborů, vytvoření tabulky MIME (Multipurpose Internet Mail Extensions), vytvoření tabulky zdrojů (Resource Table).

8.7.1 Importování souborů webových stránek

Importování souborů webových stránek je provedeno pomocí direktivy *#ximport "filename" symbol*. Tato direktiva umístní soubor s názvem *filename* do paměti programu jako binární záznam. Adresa importovaného souboru je reprezentována makrem *symbol*, které je generováno kompilátorem při překladu kódu. Makro *symbol* je dále použito při vytvoření tabulky zdrojů HTTP serveru.

```
//Importovani html stranek a obrazku
#ximport "\html\index.html"          index_html
#ximport "\html\values.xml"          values_xml
#ximport "\html\nast_par.html"       nast_par_html
#ximport "\html\par_site.html"       par_site_html
#ximport "\html\tcp.html"            tcp_html
#ximport "\html\zmena_hesla.html"    zmena_hesla_html
#ximport "\html\obr.jpg"             obr_jpg
#ximport "\html\rcm30_600_o.jpg"     rcm30_600_o_jpg
#ximport "\html\body.gif"            body_gif
#ximport "\html\rozvrzeni.css"       rozvrzeni_css
#ximport "\html\script.js"           script_js
```

Obrázek 33. Importování souborů webových stránek

8.7.2 Vytvoření tabulky MIME

Vytvoření tabulky MIME je provedeno pomocí maker *SSPEC_MIME*. Tato tabulka slouží k zaregistrování typů (přípon) jednotlivých souborů, aby mohly být správně interpretovány po načtení z HTTP serveru do webového prohlížeče na stanici klienta. Začátek tabulky je uvozen makrem *SSPEC_MIMETABLE_START*, dále následuje registrace jednotlivých typu souborů. Tabulka je ukončena makrem *SSPEC_MIMETABLE_END*. V aplikaci komunikačního modulu Ethernet jsou registrovány následující typy souborů.

```
//Tabulka MIME typu
SSPEC_MIMETABLE_START
   SSPEC_MIME(".html", "text/html"),
   SSPEC_MIME(".jpg", "image/jpg"),
   SSPEC_MIME(".bmp", "image/gif"),
   SSPEC_MIME(".css", "text/css"),
   SSPEC_MIME(".js", "text/javascript"),
   SSPEC_MIME_FUNC(".xml", "text/xml", shtml_handler)
SSPEC_MIMETABLE_END
```

Obrázek 34. Vytvoření tabulky MIME

8.7.3 Vytvoření tabulky zdrojů

Vytvoření tabulky zdrojů (Resource Table) je provedeno pomocí maker *SSPEC_RESOURCE*. Tabulka zdrojů obsahuje seznam všech souborů a dynamických proměnných HTTP serveru. Soubory jsou do tabulky zdrojů přidávány makrem *SSPEC_RESOURCE_XMEMFILE(name, address)*, kde argument *name* reprezentuje název souboru, položka *address* adresu souboru v paměti.

```
//Tabulka zdroju
SSPEC_RESOURCETABLE_START
   SSPEC_RESOURCE_XMEMFILE("/", index_html),
   SSPEC_RESOURCE_XMEMFILE("/index.html", index_html),
   SSPEC_RESOURCE_XMEMFILE("/values.xml", values_xml),
   SSPEC_RESOURCE_XMEMFILE("/nast_par.html", nast_par_html),
   SSPEC_RESOURCE_XMEMFILE("/par_site.html", par_site_html),
   SSPEC_RESOURCE_XMEMFILE("/tcp.html", tcp_html),
   SSPEC_RESOURCE_XMEMFILE("/zmena_hesla.html", zmena_hesla_html),
   SSPEC_RESOURCE_XMEMFILE("/obr.jpg", obr_jpg),
   SSPEC_RESOURCE_XMEMFILE("/rcm30_600_o.jpg", rcm30_600_o_jpg),
   SSPEC_RESOURCE_XMEMFILE("/body.gif", body_gif),
   SSPEC_RESOURCE_XMEMFILE("/rozvrzeni.css", rozvrzeni_css),
   SSPEC_RESOURCE_XMEMFILE("/script.js", script_js),
   SSPEC_RESOURCE_ROOTVAR("ActTmp", (void*)&InputRegs.Temperature, FLOAT32, "%7.2f"),
   SSPEC_RESOURCE_ROOTVAR("MaxTmp", (void*)&InputRegs.MaxTemp, FLOAT32, "%7.2f"),
   SSPEC_RESOURCE_ROOTVAR("MinTmp", (void*)&InputRegs.MinTemp, FLOAT32, "%7.2f"),
   SSPEC_RESOURCE_ROOTVAR("UnitCode", (void*)&HoldingRegs.UnitsCode, INT16, "%d"),
   SSPEC_RESOURCE_ROOTVAR("UpperRange", (void*)&InputRegs.UpperRange, FLOAT32, "%7.2f"),
   SSPEC_RESOURCE_ROOTVAR("LowerRange", (void*)&InputRegs.LowerRange, FLOAT32, "%7.2f"),
   SSPEC_RESOURCE_ROOTVAR("Damping", (void*)&InputRegs.Damping, FLOAT32, "%7.2f")
SSPEC_RESOURCETABLE_END
```

Obrázek 35. Vytvoření tabulky zdrojů

8.7.4 Přenos statických dokumentů

Po přijetí požadavku z webového prohlížeče provede HTTP server porovnání názvu požadovaného dokumentu s názvy souborů v tabulce zdrojů. Pokud je požadovaný dokument nalezen, je zaslán webovému prohlížeči. Pokud není dokument v tabulce zdrojů nalezen, je vrácena chyba HTTP 404 – Not Found.Tímto způsobem probíhá přenos všech statických dokumentů.

8.7.5 Přenos dokumentů s dynamickým obsahem

Pro přenos dokumentů s dynamickým obsahem bylo využito SSI příkazů (Server Side Includes). Jedná se o jednoduchý způsob generování dynamického obsahu za pomocí vkládání speciálních direktiv do HTML kódu webových stránek. Tyto direktivy jsou při zpracování požadovaného dokumentu HTTP serverem nahrazeny příslušným obsahem (hodnota proměnné, soubor, atd.).

Všechny dynamické proměnné jsou webovému prohlížeči předávány pomocí stránky *values.xml*.

```
- <TD485>
  - <Temperature>
    - <ActValue>
        <!-- #echo var="ActTmp"  -->
      </ActValue>
    - <MaxValue>
        <!-- #echo var="MaxTmp"  -->
      </MaxValue>
    - <MinValue>
        <!-- #echo var="MinTmp"  -->
      </MinValue>
    - <UnitCode>
        <!-- #echo var="UnitCode"  -->
      </UnitCode>
    </Temperature>
  + <Params>
  + <IPConfig>
  + <ModbusRTU>
  </TD485>
```

Obrázek 36. Struktura stránky values.xml

Každý SSI povel v souboru *values.xml* je HTTP serverem nahrazen hodnotou příslušné proměnné a odeslán webovému prohlížeči. Soubor *values.xml* je do webového prohlížeče načten pomocí funkce *importXml(file, callback)* [16]. Po

načtení XML souboru je funkcí *importXml* volána funkce *callback*, která zajistí zobrazení hodnot na webové stránce. Tímto způsobem probíhá načtení a zobrazení všech dynamických proměnných.

```
//Obnoveni hodnot periodickym nacitanim souboru values.xml
function RefreshValues(xmlDoc)
{
  TD485 = xmlDoc.getElementsByTagName('TD485')[0];
  Temp = TD485.getElementsByTagName('Temperature')[0]
  ActValue = Temp.getElementsByTagName('ActValue')[0].firstChild.nodeValue;
  MaxValue = Temp.getElementsByTagName('MaxValue')[0].firstChild.nodeValue;
  MinValue = Temp.getElementsByTagName('MinValue')[0].firstChild.nodeValue;
  UnitCode = Temp.getElementsByTagName('UnitCode')[0].firstChild.nodeValue;
  Units = new Array(" °C", " °K", " °F");

  //Zobrazeni dat do textovych poli
  ActTmp.firstChild.nodeValue = ActValue + Units[UnitCode - 1];
  MaxTmp.firstChild.nodeValue = MaxValue + Units[UnitCode - 1];
  MinTmp.firstChild.nodeValue = MinValue + Units[UnitCode - 1];

  //Za sekundu vycist hodnoty
  setTimeout("importXML('values.xml', RefreshValues);", 1000);
}
```

Obrázek 37. Zobrazení aktuálních hodnot teploty pomocí Java skriptu (index.html)

8.7.6 Nastavování parametrů modulu pomocí webových stránek

K nastavování parametrů jsou použity webové stránky s formuláři (Forms). Formuláře obsahují textová pole (a jiné prvky, např. roletky, zaškrtávací pole, atd.), pomocí kterých se zadávají hodnoty parametrů. Hodnoty parametrů jsou odeslány HTTP serveru pomocí metody POST. Ke zpracování jednotlivých parametrů je určena API funkce *int parse_post(HttpState *state)* [15].

V aplikaci komunikačního modulu se z časových důvodů nastavování parametrů přes webové rozhraní nepodařilo realizovat. Všechny parametry komunikačního modulu jsou nastaveny programově a není možné je měnit (pouze změnou firmware).

9. OBSLUŽNÝ SOFTWARE PRO PC

9.1 ÚVOD

Obslužný software pro PC byl vytvořen a odladěn v prostředí Microsoft Visual C# 2008 Express Edition. Umožňuje následující funkce:

- oživení modulu pro měření teploty – zavedení souboru se standardními daty do eeprom
- kalibrace a linearizace měření teploty
- nastavení parametrů modulu pro měření teploty (výrobní číslo, datum kalibrace, Slave adresa, jednotky měření teploty, předpětí a zakončení linky RS485, přenosová rychlost)
- periodické čtení hodnot vstupních (Input) a udržovacích (Holding) registrů

Obslužný software byl vytvořen tak, aby byl nezávislý na typu použité sběrnice. Umožňuje komunikovat jak protokolem ModbusRTU po sériové lince, tak i protokolem ModbusTCP přes Ehternet.

9.2 SEZNAM ZDROJOVÝCH SOUBORŮ OBSLUŽNÉHO SOFTWARE

Obslužný software pro PC se skládá z následujících souborů.

- **modbus.cs** – bázová (abstraktní) třída pro komunikaci protokolem Modbus
- **modbusRTU.cs** – třída pro komunikaci protokolem ModbusRTU (sériová linka)
- **modbusTCP.cs** – třída pro komunikaci protokolem ModbusTCP (Ethernet)
- **registers.cs** – třída pro práci s registry modulu pro měření teploty
- **eeprom.cs** – třída pro práci s eeprom modulu pro měření teploty
- **convertfloat.cs** – třída pro převod čísel v plovoucí řádové čárce (float)

9.3 BÁZOVÁ TŘÍDA MODBUS

Třída *modbus* je bázová třída, ze které jsou následně odvozeny třídy *modbusRTU* a *modbusTCP*. Bázová třída *modbus* definuje abstraktní metodu *sendRec*, jejíž implementace je závislá na typu sběrnice. Tato metoda je proto implementována až v odvozených třídách *modbusRTU* a *modbusTCP*. Bázová třída *modbus* dále definuje a implementuje metody, které realizují standardní a uživatelské funkce protokolu Modbus. Tyto metody nejprve sestaví požadovaný PDU protokolu Modbus (nezávislé na sběrnici) a po té volají metodu *sendRec*, která PDU doplní o potřebné položky (vytvoří ADU) a provede přenos požadavku po sériové lince nebo po Ethernetu. Bázová třída *modbus* definuje následující metody.

9.3.1 Metody třídy modbus

- *protected abstract byte[] sendRec(byte[] req)* – abstraktní metoda pro vysílání požadavků a příjem odpovědí protokolu Modbus
- *public UInt16[] readHoldingRegisters(int unitID, int address, int quantity)* – standardní funkce #3
- *public UInt16[] readInputRegisters(int unitID, int address, int quantity)* – standardní funkce #4
- *public void writeSingleRegister(int unitID, int address, int value)* – standardní funkce #6
- *public byte[] readEeprom(int unitID, int address, int byteCount)* – uživatelská funkce pro čtení řady bytu z eeprom modulu měření teploty
- *public void writeEeprom(int unitID, int address, int byteCount, byte[] eeprom)* – uživatelská funkce pro zápis řady bytu do eeprom modulu měření teploty
- *public void resetDevice(int unitID)* – uživatelská funkce pro softwarový reset modulu měření teploty
- *public void enableUserFce(int unitID)* – uživatelská funkce pro povolení výše popsaných funkcí

9.4 TŘÍDY MODBUSRTU A MODBUSTCP

Tyto třídy jsou odvozeny z bázové třídy *modbus*, ze které dědí všechny metody. Obě třídy *modbusRTU* a *modbusTCP* implementují abstraktní metodu *sendRec*. Třída *modbusRTU* realizuje komunikaci přes sériový port, využívá systémovou třídu *SerialPort*. Třída *modbusTCP* realizuje komunikaci přes Ethernet, využívá systémovou třídu *TcpClient*.

9.5 TŘÍDA REGISTERS

Třída *registers* umožňuje aplikaci vyčítání a zápis registrů modulu pro měření teploty. Ke komunikaci využívá třída *registers* privátní člen *mb*. K vytvoření instance třídy *registers* jsou definovány tři verze konstruktorů. V závislosti na tom, který konstruktor aplikace volá, bude komunikace probíhat po sériové lince nebo po Ethernetu.

9.5.1 Konstruktor 1

```
public registers()
{
      mb = new modbusRTU("COM1", 9600, 8, Parity.None,
      StopBits.One);
}
```

Pokud aplikace volá tento konstruktor, privátní člen *mb* obsahuje instanci třídy *modbusRTU* a komunikace probíhá po sériové lince (default parametry přenosu).

9.5.2 Konstruktor 2

```
public registers(string portName, int baudRate, int dataBits, Parity
parity, StopBits stopBits)
{
      mb = new modbusRTU(portName, baudRate, 8, parity, stopBits);
}
```

Pokud aplikace volá tento konstruktor, privátní člen *mb* obsahuje instanci třídy *modbusRTU* a komunikace probíhá po sériové lince. Parametry přenosu jsou nastaveny dle hodnot argumentů konstruktoru.

9.5.3 Konstruktor 3

```
public registers(string ipAddress, int port)
{
	mb = new modbusTCP(ipAddress, port);
}
```

Pokud aplikace volá tento konstruktor, privátní člen *mb* obsahuje instanci třídy *modbusTCP* a komunikace probíhá přes Ethernet. Argumenty jsou IP adresa ModbusTCP serveru a port (obvykle 502).

9.6 TŘÍDA EEPROM

Třída *eeprom* umožňuje aplikaci zápis a čtení paměti eeprom modulu měření teploty. Stejně jako třída *registers* definuje také třída *eeprom* tři verze konstruktorů. Pro čtení a zápis základních datových typů do eeprom jsou implementovány následující metody.

9.6.1 Metody třídy eeprom

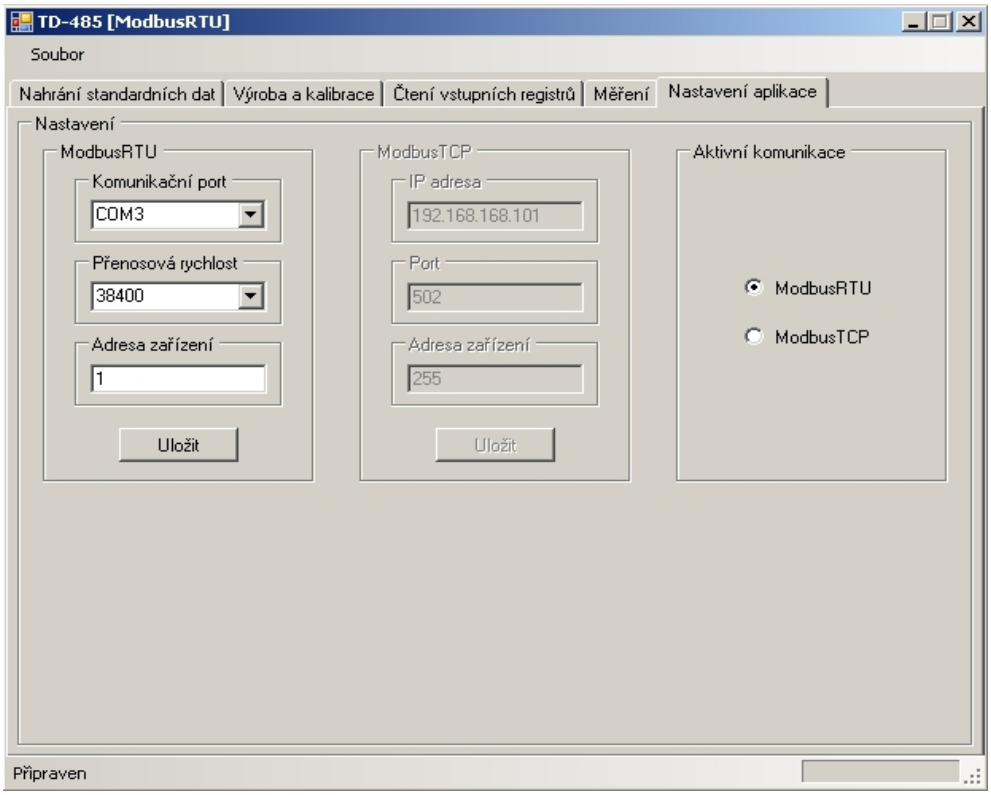
- *public byte[] readByte(int unitID, int addr, int count)* – čtení pole bytů z eeprom
- *public void writeByte(int unitID, int addr, int count, byte[] data)* – zápis pole bytů do eeprom
- *public UInt32[] readUint32(int unitID, int addr, int count)* – čtení pole 32. bitových hodnot (Uint32) z eeprom
- *public void writeUInt32(int unitID, int addr, int count, UInt32[] data)* – zápis pole 32. bitových hodnot (Uint32) do eeprom
- *public float[] readFloat(int unitID, int addr, int count)* – čtení pole float hodnot z eeprom
- *public void writeFloat(int unitID, int addr, int count, float[] flt)* – zápis pole float hodnot do eeprom

9.7 TŘÍDA CONVERTFLOAT

Třída *convertFloat* je statická třída, která implementuje metody pro převod float hodnot z formátu IEEE754 na formát používaný mikrokontrolérem XE8805A. Metody této třídy jsou volány při zápisu i čtení float hodnot do eeprom modulu měření teploty.

9.8 POUŽÍTÍ OBSLUŽNÉHO SOFTWARE

Před použitím programu musí uživatel nejprve zvolit protokol a nastavit parametry komunikace. Veškeré nastavení je uloženo do registru Windows.



Obrázek 38. Nastavení parametrů komunikace

Záložka „Nahrání standardních dat“ slouží pro oživení modulu měření teploty. Záložka „Výroba a kalibrace“ slouží ke kalibraci a nastavení parametrů. Pomocí záložek „Čtení vstupních registrů“ a „Měření“ je možné periodické vyčítání naměřených hodnot.

10. ZÁVĚR

V první části diplomové práce byly studovány základní principy měření teploty pomocí kovových teplotních senzorů. Byl navržen a následně realizován inteligentní modul pro měření teploty s digitálním výstupem RS485 a protokolem ModbusRTU. Modul je vybaven LCD displejem a tlačítky, interní menu umožňuje zákaznické nastavení všech parametrů. Všechny funkce modulu měření teploty byly ověřeny na dvou prototypových kusech. Byl vytvořen obslužný software pro PC, který umožňuje oživení, nastavení a kalibraci modulu měření teploty. Modul splňuje všechny požadavky zadání a je připraven k sériové výrobě.

V druhé části práce byly studovány základy komunikačního standardu Ethernet (IEEE 802.3) a současné možnosti řešení implementace Ethernetu pro průmyslové převodníky a vestavné systémy. Na základě technické rešerše byla pro realizaci komunikačního modulu Ethernet vybrána platforma Rabbit, modul RCM3200. Byly navrženy dvě varianty hardwarového řešení pro připojení modulu měření teploty k Ehternetu. Funkční prototyp komunikačního modulu byl realizován a testován na vývojové desce RCM30/31/32XX.

Dále byl vytvořen firmware komunikačního modulu. Firmware umožňuje periodické vyčítání hodnot registrů modulu měření teploty, pro komunikaci s nadřazeným systémem byl implementován protokol ModbusTCP. Pomocí webových stránek je možné vyčítat naměřené hodnoty v prostředí standardních webových prohlížečů.

11. SEZNAM POUŽITÝCH ZDROJŮ

[1] KREIDL, M. *Měření teploty, senzory a měřící obvody*. 1. vyd. BEN – Technická literatura, 2005. 240 s. ISBN 80-7300-145-4

[2] Texas Instrument, *TPS71550* [online], rev. 6/2007. Dostupné z: < http://focus.ti.com/lit/ds/symlink/tps71550.pdf >

[3] Zetex, *ZRA431* [online], rev. 5/2002. Dostupné z: < http://www.zetex.com/3.0/pdf/ZR431.pdf >

[4] Semtech, XE8805A [online], rev. 1/2006. Dostupné z: < http://www.semtech.com/pc/downloadDocument.do?navId=H0,C1,C193,C195,C199,P2637&id=789 >

[5] Raisonance, *RkitC816* [online], rev. 1/2006. Dostupné z: < http://www.mcu-raisonance.com/mcu_downloads.html >

[6] Ronešová A. *Přehled protokolu MODBUS* [online], rev. 05/2005. Dostupné z: < http://home.zcu.cz/~ronesova/bastl/files/modbus.pdf >

[7] Modbus-IDA, *MODBUS over serial line specification and implementation guide* [online], rev. V1.02 12/2006. Dostupné z: < http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf >

[8] Jura, P. *Signály a systémy část 2, spojité systémy* [online]. Dostupné z: < https://www.feec.vutbr.cz/et/skripta/uamt/Signaly_Systemy_S_P2.pdf?PHPSESSID=21f6635a192b679a4e8ab417eae67893 >

[9] Site.The.cz, *Historie vzniku technologie Ethernet* [online]. Dostupné z: < http://site.the.cz/index.php?id=24 >

[10] Wikipedia, *Ethernet* [online]. Dostupné z: < http://cs.wikipedia.org/wiki/Ethernet >

[11] DOSTÁLEK, L – KABELOVÁ, A. *Velký průvodce protokoly TCP/IP a systémem DNS*. 1. vyd. Praha: Computer Press, 2000. 240 s. ISBN 80-7226-323-4

[12] Rabbit, *RabbitCore RCM3200 C-Programmable Module with Ethernet User's Manual* [online], rev. 019-0118. Dostupné z: <

http://www.rabbit.com/documentation/docs/manuals/RCM3200/UsersManual/RC3200UM.pdf >

[13] ZWorld Inc, *DynamicC 9.62* [online], ver 9.62. Dostupné z: < http://www.rabbit.com/support/downloads/dc/dc9.shtml >

[14] Modbus-IDA, *MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE V1.0b* [online], rev. 10/2006. Dostupné z: < http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf >

[15] Rabbit, *Dynamic C TCP/IP User's Manual volume 2* [online], rev. 019-0144. Dostupné z: < http://www.rabbit.com/documentation/docs/manuals/TCPIP/UsersManualV2/tcpV2.pdf >

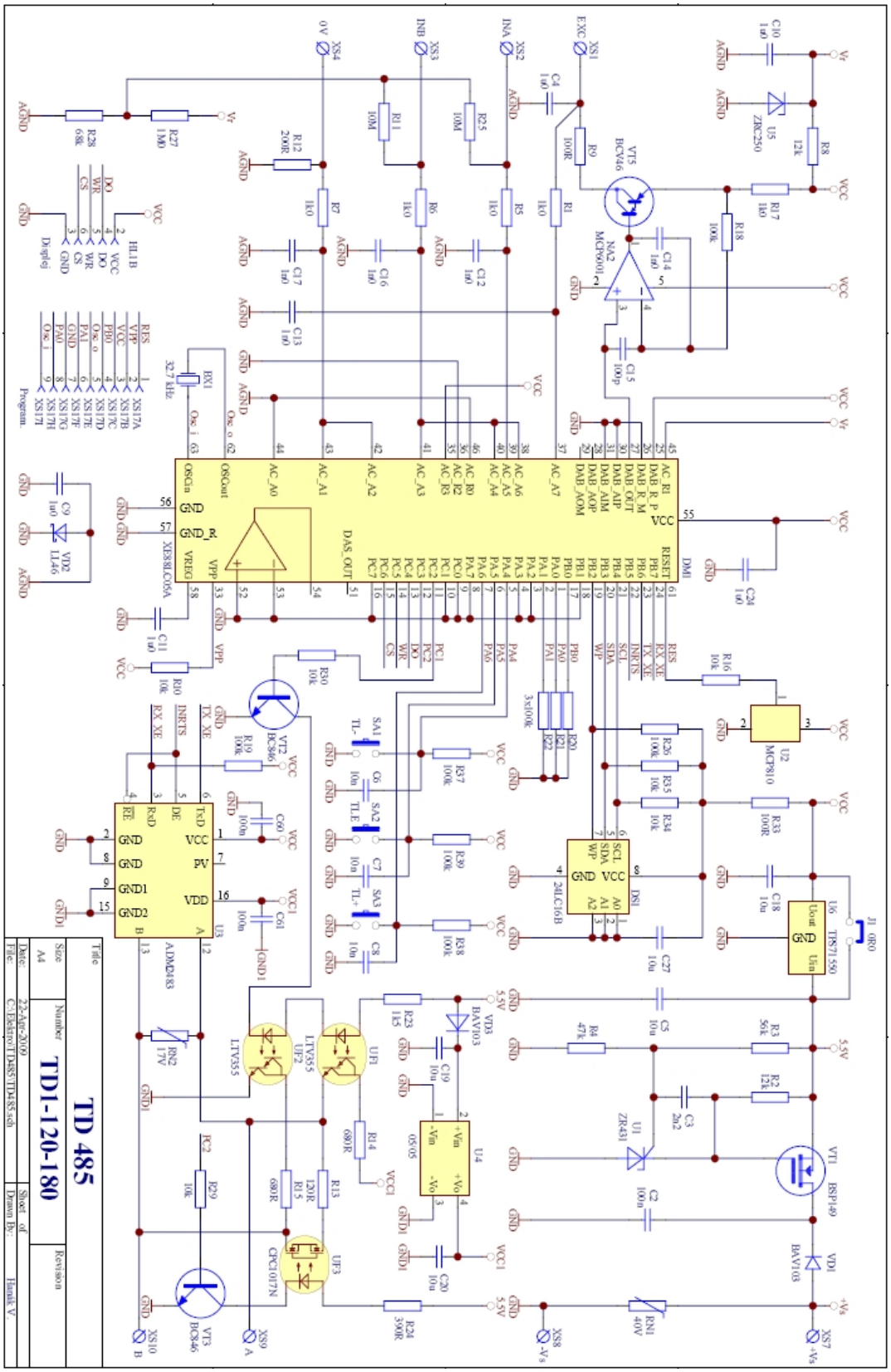
[16] www.jr.pl, Import XML Document [online]. Dostupné z: < http://www.jr.pl/www.quirksmode.org/dom/importxml.html >

12. SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

ADC	Analog to Digital Converter
ADU	Application Data Unit
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
EEPROM	Electronically Erasable Programmable Read-Only Memory
FAT	File Allocation Table
FTP	File Transfer Protocol
HART	Highway Addressable Remote Transducer
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IMAP	Internet Message Access Protocol
IP	Internet Protocol
ISO/OSI	International Organization for Standardization/Open Systems Interconnection
LCD	Liquid Crystal Display
MAC	Media Access Control
MIME	Multipurpose Internet Mail Extensions
PDU	Protocol Data Unit
PGA	Programmable Gain Amplifier
PLC	Programmable Logic Controller
POR	Power on Reset
RISC	Reduced Instruction Set Computer
RTD	Resistance Temperature Detector
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SSI	Server Side Includes
TCP	Transmit Control Protocol
UDP	User Data Protocol

13. SEZNAM PŘÍLOH

Příloha 1. Schéma zapojení modulu pro měření teploty



Příloha 2. Motiv desky plošného spoje modulu pro měření teploty

Příloha 3. Mapa vstupních registrů (Input Registers) modulu měření teploty

Adresa registru	Název registru	Popis registru	Datový typ
0x0000	SerialNr	Výrobní číslo	unsigned 32
0x0001			
0x0002	CalDate	Datum poslední kalibrace	Float, IEEE754
0x0003			
0x0004	UpperRange	Horní mez rozsahu	Float, IEEE754
0x0005			
0x0006	LowerRange	Dolní mez rozsahu	Float, IEEE754
0x0007			
0x0008	Temperature	Aktuální teplota	Float, IEEE754
0x0009			
0x000A	MaxTemp	Maximální naměřená teplota	Float, IEEE754
0x000B			
0x000C	MinTemp	Minimální naměřená teplota	Float, IEEE754
0x000D			

Příloha 4. Mapa udržovacích registrů (Holding Registers) modulu měření teploty

Adresa registru	Název registru	Popis registru	Datový typ
0x0000	UnitsCode	Kód jednotky teploty	unsigned 16
0x0001	Address	Slave adresa	unsigned 16
0x0002	Cfg485	Konfigurace linky RS485	unsigned 16
0x0003	BaudRate	Přenosová rychlost	unsigned 16
0x0004	DecPPos	Pozice desetinné tečky	unsigned 16