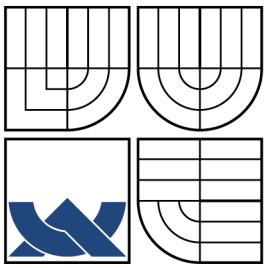
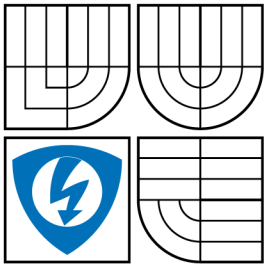


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

INTELIGENTNÍ PŘEVODNÍK NI ODPOROVÝCH SENZORŮ S ROZHRANÍM ETHERNET

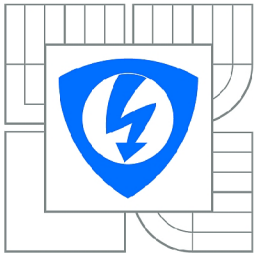
NI SENSORS TO ETHERNET CONVERTOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. ONDŘEJ KUBÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR
Doc. Ing. ZDENĚK BRADÁČ, Ph.D.

BRNO 2010



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. Ondřej Kubíček	***ID:***	78520
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Inteligentní převodník Ni odporových senzorů s rozhraním Ethernet

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte koncepci elektroniky převodníku, který umožní připojit sadu standardních teplotních Ni sensorů na Ethernet. Vlastní elektronika bude založena na procesoru Rabbit. Implementujte WWW stránky pro konfiguraci a poskytování dat, Modbus TCP pro přenos informací a základní funkčnost dataloggeru.
Realizujte hardwarový návrh, osaďte DPS a oživte HW. Vytvořte programové vybavení a demonstrujte plnou funkčnost zařízení. Sepište diplomovou práci.

DOPORUČENÁ LITERATURA:

Pavel Herout: Učebnice jazyka C, KOPP, 2004, IV. přepracované vydání, ISBN 80-7232-220-6
Dle zadání vedoucího práce.

Termín zadání:	8.2.2010	***Termín odevzdání:***	24.5.2010

Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.

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

UPOZORNĚNÍ:

ABSTRAKT

Práce ze zabývá návrhem hardwarového řešení a softwarového vybavení mikrokontrolérem Rabbit řízeného převodníku standardních odporových Ni senzorů teploty s výstupem na Ethernet. První část práce se zabývá teoretickým úvodem do měření teploty, ve stručnosti jsou představeny základní principy snímačů pro měření teploty. V další části jsou představeny komunikační standard Ethernet a aplikační protokol Modbus. Zbytek práce se již věnuje vlastnímu řešení převodníku teploty. Je popsáno vlastní zapojení převodníku a programové vybavení jak samotného převodníku teploty tak programové vybavení PC.

KLÍČOVÁ SLOVA

Datalogger, Dynamic C, Ethernet, miniSD, Modbus, Odporové snímače teploty, Převodník teploty, Rabbit

ABSTRACT

This work deals with hardware and software solution for the temperature converter of the standard rezistance temperature Ni sensors with the output through the Ethernet, driven by the Rabbit microcontroller. First part of this thesis deals with the theory of the temperature measurement. There are introduced basic principles of the temperature sensors. In the next part of the work are introduced communication standard and application protocol Modbus. The rest of the work deals with the hardware concept of the converter, the control program of the converter and the program developed for the PC.

KEYWORDS

Datalogger, Dynamic C, Ethernet, miniSD, Modbus, Rabbit, RTD, Temperature converter

KUBÍČEK, O. *Inteligentní převodník Ni odporových senzorů s rozhraním Ethernet*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 105 s. Vedoucí diplomové práce Doc. Ing. Zdeněk Bradáč, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Inteligentní převodník Ni odporových senzorů s rozhraním 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 ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Doc. Ing.Zdeňku Bradáčovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

Teplota patří v dnešní době mezi jednu z nejčastěji měřených fyzikálních veličin. Její přesné určení se stalo nezbytností v celé řadě odvětví průmyslové výroby, pro příklad uveďme potravinářský, farmaceutický nebo hutní průmysl. V průběhu let bylo vyvinuto velké množství principů, pomocí nichž lze teplotu měřit. Tyto principy od sebe vzájemně odlišují, jak podmínky, za kterých se dají použít, tak jejich výrobní náročnost a s tím spojenou cenu. Princip využívající změny odporu v závislosti na teplotě patří k nejstarším. I přesto, díky svým výhodným vlastnostem, se tzv. odporové snímače teploty dnes řadí mezi nejvíce používané snímače na trhu.

Zařízení nazývaná převodník teploty pak představují prostředek, jak převést signál snímače nesoucí informaci o měřené teplotě do unifikované, v dnešní době nejčastěji digitální podoby, jež může být následně zpracována nadřazeným systémem. Jelikož jednotlivé typy odporových snímačů mají vzájemně odlišné parametry, měl by převodník teploty poskytovat základní možnosti konfigurace, aby jej bylo možné použít ve spojení s co největším množstvím teplotních snímačů. V současné době se častěji setkáváme i s požadavkem prezentace měřených dat a konfigurace zařízení přes webové rozhraní.

1.1 Cíle diplomové práce

Cílem této diplomové práce je návrh zapojení, jeho realizace a vytvoření softwarového vybavení pro převodník standardních NI odporových teplotních senzorů s výstupem na Ethernet (dále v této práci zkráceně označovaný, již pouze jako převodník, případně převodník teploty). Návrh zapojení a jeho vlastní fyzická realizace proběhla v rámci semestrálních projektů MM1K a MM2K. V tomto semestru bylo hlavním úkolem vytvořit softwarové vybavení převodníku, navrhnout webové stránky pro prezentaci naměřených dat a základní konfiguraci některých parametrů převodníku a vytvořit obslužný software na PC, jež bude sloužit, jak ke konfiguraci, tak k vyčítání naměřených dat z převodníku.

Aby převodník byl z hlediska možnosti připojitelných senzorů co nejflexibilnější, bude umožněno do něj nahrát parametry různých teplotních snímačů, mezi nimiž bude uživatel moc následně přepínat. PC aplikace sloužící pro kalibraci pak musí být schopna

vypočíst konstanty pro jednotlivé snímače, a tyto konstanty které budou sloužit k výpočtu teploty z hodnoty získané z A/D převodníku následně do převodníku nahrát.

Vytvoření programového vybavení v tomto semestru tedy zahrnovalo vytvoření Modbus serveru na straně mikrokontroléru Rabbit a Modbus klienta na straně PC pro vzájemnou komunikaci mezi těmito dvěma zařízeními. Vytvoření rutin pro správnou komunikaci po síti Ethernet. Implementaci funkčnosti dataloggeru spočívající v pravidelném zaznamenávání naměřených teplot na připojenou paměťovou kartu a umožnění následného vyčítání těchto hodnot a jejich následného zapsání do souboru typu csv, jež lze vizualizovat například v programu Excel. Dále zmíněné webové stránky a obě PC aplikace pro ovládání převodníku.

2 MĚŘENÍ TEPLOTY

2.1 Základní rozdělení

V technické praxi se teplota řadí mezi nejdůležitější parametry zajišťující optimální chod technologického procesu a jak již bylo řečeno v úvodu, její měření patří v technické praxi mezi jednu z nejrozšířenějších úloh. Základní rozdělení metod pro měření teploty můžeme provést podle toho, zda-li se měřící snímač nachází nebo nenachází v přímém kontaktu s měřeným prostředím, případně tělesem na:

- kontaktní
- bezkontaktní

2.2 Kontaktní metody měření teploty

U kontaktního principu měření teploty dochází k ustálení tepelné rovnováhy mezi měřeným prostředím a vlastním tělem snímače. Dosažení tepelné rovnováhy znamená, že tělo snímače má stejnou teplotu jako prostředí, v němž měří. Informace o měřené teplotě je pak nesena některou z charakteristických vlastností snímače, která se mění spolu s teplotou. Příkladem může být tepelná roztažnost látek využívaná u dilatačních teploměrů (například známý rtuťový teploměr, případně bimetalový teploměr tvořený páskem složeným ze dvou kovů o různé teplotní roztažnosti) nebo změna tlaku v plynu při jeho konstantním objemu. [1] Významnou nevýhodu uvedených principů tvoří skutečnost, že informace o teplotě není v těchto případech nesena přímo elektrickou veličinou, ale je potřeba ji na elektrickou veličinu pomocí dodatečného snímače teprve převést. Z tohoto hlediska jsou velmi výhodné jevy jako závislost elektrického odporu vodiče, či polovodiče na teplotě, nebo termoelektrický jev, kdy lze teplotu přímo elektricky vyhodnotit.

2.2.1 Odporové senzory

Princip snímačů spočívá ve změně elektrického odporu v závislosti na teplotě. Pro vyhodnocení teploty využíváme známého vztahu mezi elektrickým odporem, napětím a proudem. Aby odporovým snímačem procházel po celou dobu konstantní proud, představuje ideální způsob napájení, napájení z proudového zdroje. V případě, že dojde ke zvýšení či snížení měřené teploty, elektrický odpor snímače se změní, tím dojde i ke změně úbytku napětí na snímači. Velikost úbytku napětí na snímači jsme již schopni elektricky vyhodnotit a tím pádem i změřit teplotu.

Kovové odporové snímače

Základní parametr představuje u kovových odporových snímačů teploty, vedle hodnoty odporu při 0 $^{\circ}C$ označovaném jako R_0, materiálová konstanta α, též nazývaná jako teplotní součinitel odporu. Tento parametr představuje hodnotu, o kterou se zvýší elektrický odpor snímače při nárůstu teploty o 1 $^{\circ}C$ a lze ho určit vztahem

$$\alpha = \frac{R_{100} - R_0}{100 * R_0} \tag{2.1}$$

Pro malý rozsah teplot $0 - 100^{\circ}\ C$, můžeme vztah mezi teplotou a elektrickým odporem vyjádřit pomocí lineární rovnice [2]

$$R_t = R_0(1 + \alpha t) \tag{2.2}$$

Pro vyšší rozsahy měřených teplot již nelze tento lineární vztah použít a pro vyjádření elektrického odporu potřebujeme polynomickou rovnici, příkladem může být rovnice

$$R_t = R_0(1 + At + Bt^2) \tag{2.3}$$

[2], kde A a B jsou normované teplotní koeficienty snímače (uvedený příklad se týká platinového čidla v teplotním rozsahu $0 - 850\ ^{\circ}C$) [2]. Nejrozšířenějším materiálem, ze kterého se kovové odporové snímače vyrábějí, je platina. Dalšími využívanými materiály jsou nikl, měď, molybden. Pro měření velmi nízkých teplot se využívají i slitiny kovu jako Pt-Co nebo Rh-Fe [2].

Měřící snímače vyrobené z platiny se vyznačují chemickou netečností, časovou stálostí a vysokou teplotou tání. Základní a nejčastěji využívanou hodnotou odporu u

platinových snímačů představuje 100Ω, vyrábějí se i snímače se základními hodnotami odporu R_0 rovnými $50, 200, 500, 1000, 2000\ \Omega$ [2]. Pomocí platinových snímačů lze realizovat měření teplot v rozsahu -200 až $850\ ^\circ C$. Pro vyjádření odporu v rozsahu -200 až $0\ ^\circ C$ se využívá rovnice [2]

$$R_t = R_0(1 + At + Bt^2 + C(t - 100)) \tag{2.4}$$

Rovnice pro rozsah 0 až $850\ ^\circ C$ má podobu, jež byla uvedena výše.

Další často používaný materiál pro výrobu odporových snímačů teploty představuje nikl. Mezi výhody těchto snímačů se řadí velká citlivost, rychlá odezva, malá časová konstanta, malé rozměry a také oproti platině nižší cena. Mezi nevýhody patří výrazná nelinearita, horší dlouhodobá stabilita a nižší odolnost vůči vlivu prostředí. Obvyklé základní hodnoty těchto snímačů bývají stejné jako u snímačů platinových [2]. Odpor snímače udává rovnice

$$R_t = R_0(1 + At + Bt^2 + Ct^4 + Dt^6) \tag{2.5}$$

Měď lze použít v rozsahu teplot -200 až $200\ ^\circ C$, přičemž v rozsahu -50 až $150\ ^\circ C$ lze využít linearizovaný vztah. Měděné snímače se z důvodů snadné oxidace a malé rezistivity používají pouze výjimečně. Jednou z aplikací je měření teploty vinutí elektrických motorů, kdy měříme přímo odpor vinutí motoru [2].

Termisotory

Termistory využívají, stejně jako kovové odporové snímače, závislosti odporu na teplotě. V závislosti na materiálu, z něhož je vyroben, má termistor, buď velký záporný teplotní součinitel nebo kladný teplotní součinitel. Výhodou termistorů je velká teplotní citlivost, malé rozměry, jednoduchý převod elektrického odporu na napětí nebo proud a vysoká hodnota elektrického odporu. Nevýhodou výrazně nelineární charakteristika a časová nestálost.

Negastory (NTC) mají záporný teplotní součinitel, odpor takového snímače se vzrůstající teplotou klesá. Vyrábějí se práškovou technologií z oxidů kovů, jako je oxid chromu, kobaltu, železa nebo manganu. Teplotní rozsah negastorů může dosahovat vysokých teplot okolo $1000\ ^\circ C$. Závislost odporu na teplotě udává vztah

$$R_T = Ae^{(\frac{B}{T})} \tag{2.6}$$

Kde koeficitent A představuje geometrickou konstantu snímače závislou na geometrickém tvaru a materiálu a koeficient B představuje teplotní konstantu snímače, závislou na materiálu z něhož je snímač vyroben [3]. Elektrický odpor pozistoru (PTC) se stoupající teplotou nejprve mírně klesá. Od určité teploty (označované jako Curieova) dochází ke strmému nárůstu rezistivity. Po nárůstu o několik řádů pak hodnota odporu opět mírně klesá. Jako pracovní se využívá oblasti se vzrůstajícím odporem, kde je charakteristika snímače přibližně lineární. Snímače realizované pomocí pozistorů se zpravidla používají jako detektory překročení teploty. Materiálem ze kterého se tyto termistory vyrábí je například titaničitan barnatý [2].

Monolytické polovodičové snímače teploty

Monolitické PN senzory teploty jsou založeny na teplotní závislosti napětí PN přechodu v propustném směru. Tento princip je často použit v integrovaných obvodech sloužících k měření teploty, které kromě vlastního snímače obsahují přímo i elektroniku pro vyhodnocení měřeného signálu. Využívány jsou PN přechody diod nebo tranzistorů. Tyto obvody jsou oblíbené pro svou dobrou stálost a skutečnost, že výstup z obvodu podává již přímo informaci o měřené teplotě[2].

2.2.2 Termoelektrický jev

Další možný způsob kontaktního měření teploty představují termoelektrické snímače. Princip těchto snímačů je založen na Seebeckově jevu. Termoelektrický článek je složen ze dvou vodičů z odlišného materiálu (musí mít různé Seebeckovi koeficienty), které jsou spolu jedním koncem spojeny. Pokud nespojené konce vodičů budeme udržovat na stejné teplotě(označované jako referenční) a bod v němž jsou oba vodiče spojeny vystavíme působení teploty, jíž chceme měřit, lze mezi nespojenými konci vodičů naměřit termoelektrické napětí. Toto napětí je úměrné měřené teplotě[3]. Měřené napětí dosahuje malých hodnot v řádu milivoltů a tudíž je náchylné na zarušení [2].

2.3 Bezkontaktní metody měření teploty

Bezdotykové měření teploty spočívá v měření povrchové teploty těles na základě tělesem vysílaného elektromagnetického záření, které je měřeno pomocí snímače záření. Vlnové délky se mohou pohybovat v rozsahu od $0,4\ \mu m$ do $25\ \mu m$ [2]. V tomto rozsahu jsou zahrnuty, jak oblasti viditelného spektra, tak blízkého infračerveného až dlouhovlnného infračerveného spektra, přičemž v oblasti vlnových délek $2-25\mu m$ mluvíme o tepelném záření. Uvedené rozsahy vlnových délek obsahují měření teplot v rozsahu od -40 až $10000\ ^{\circ}C$ [2].

Měření pomocí bezdotykových metod je výhodné zejména z důvodu prakticky nulového vlivu měřícího přístroje na měřený předmět a možnosti měřit teplotu na pohybujících se nebo rotujících tělesech. Teplotu je navíc možné měřit z dostatečné vzdálenosti, aniž by došlo k ohrožení osoby provádějící měření (například zásahem vysokým napětím)[2].

Nevýhodu bezkontaktních snímačů teploty představuje možnost zkreslení měření odraženým zářením z okolního prostředí na měřený objekt a potřeba zajistit minimální tepelný odpor mezi snímačem a povrchem měřeného tělesa [2].

3 POUŽITÉ KOMUNIKAČNÍ TECHNOLOGIE

3.1 Komunikační síť Ethernet

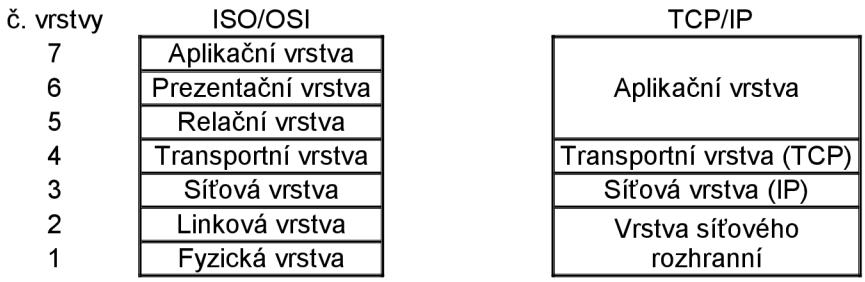
Tato komunikační síť bylo původně vyvinuta firmou Xerox, jež v sedmdesátých letech minulého století hledala způsob, jak připojit několik počítačů k jedné tiskárně. K firmě Xerox se v následujících letech připojili společnosti DEC a Intel a společně pokračovali ve vývoji této sítě. Jejich společné úsilí pak počátkem osmdesátých let vedlo k ustanovení Ethernetu jako komunikačního standardu IEEE 802.3. V současné době představuje Ethernet nejrozšířenější způsob propojení místních sítí VAN. V posledních letech pak tato technologie čím dál více proniká i do průmyslové automatizace. [4]

3.1.1 Přístup k přenosovému médiu

Pro přístup k přenosovému médiu je využito techniky CSMA/CD. Tato přístupová metoda je založena na tom, že zařízení, jež chce vysílat data, odposlouchává stav přenosového média. Pokud na přenosovém médiu v danou chvíli komunikuje jiná stanice, zařízení počká na ukončení přenosu a pak samo zahájí komunikaci. Pokud ve stejný okamžik zahájí komunikaci vícero zařízení, dojde ke kolizi, jež trvá po celou dobu přenosu. Stanice, která jako první tuto kolizi detekuje, vysílá tzv. Jam signál, po jehož odvysílání se všechny vysílající stanice odmlčí. Po uplynutí náhodného časového okamžiku se stanice opět pokusí o vysílání.

3.1.2 Ethernet TCP/IP

Samotný Ethernet, tak jak byl definován na počátku osmdesátých let standardem IEEE 802.3, představuje první dvě vrstvy referenčního ISO/OSI modelu. Tedy vrstvu fyzickou a linkovou. Na vyšších vrstvách modelu je použito síťových protokolů TCP a IP, kde protokol IP přibližně odpovídá třetí síťové vrstvě ISO/OSI modelu, zatímco protokol TCP představuje čtvrtou transportní vrstvu. Vrstvy pět a šest jsou vynechány a jejich funkce zastává sedmá aplikační vrstva. [5] Vztah mezi ISO/OSI modelem a Ethernetem TCP/IP zobrazuje obrázek 3.1



Obrázek 3.1: Vztah komunikačních modelů ISO/OSI a TCP/IP [6]

Protokol Ip slouží pro přenos dat mezi dvěma zařízeními reprezentovanými jejich IP adresami. Tento protokol nezaručuje, že odeslaný blok dat (datagram) skutečně dorazí. To musí být zajištěno některým z protokolů vyšších vrstev. [6]

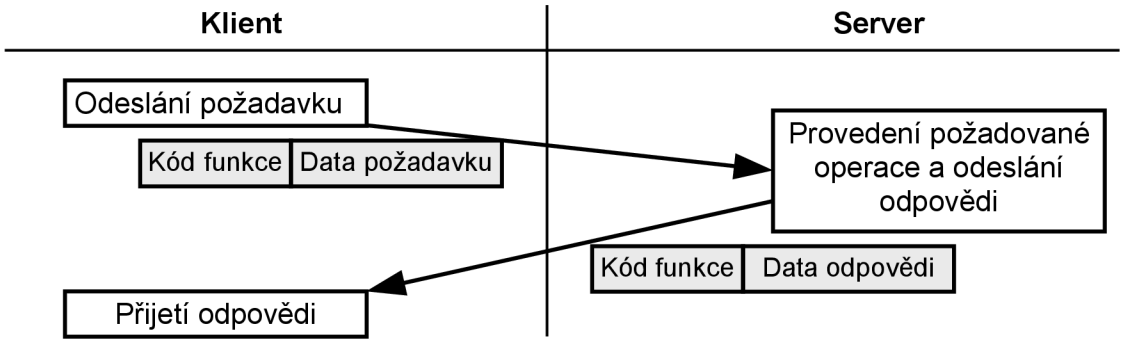
TCP představuje spojově orientovaný protokol, který zaručuje, že odeslaná data dorazí k příjemci. Tímto protokolem mezi sebou aplikace běžící na jednotlivých stanicích navazují spojení reprezentované číslem portu, přes nějž komunikace probíhá. Tento protokol rovněž pomocí kontrolního součtu ověřuje, zda nedošlo k poškození dat při přenosu. [6]

Obdobu protokolu TCP představuje protokol UDP, který rovněž spadá pod transportní vrstvu. Tento protokol na rozdíl od protokolu TCP není spojově orientovaný a tudíž nezaručuje, že data budou v pořádku doručena příjemci. Protokol UDP se obecně hodí pro aplikace, kde není požadováno, aby byla korektně doručena veškerá odeslaná data, nýbrž více záleží na rychlosti komunikace (protokol se nezdržuje potvrzováním a opakovaným odesíláním zpráv). [6]

3.2 Komunikační protokol Modbus

Jedná se o sériový komunikační protokol vyvinutý firmou Modicon zveřejněný v roce 1979. Velkou výhodu tohoto protokolu, oproti konkurenčním řešením, představuje jeho otevřenost a s ní spojený fakt, že specifikace tohoto protokolu je volně dostupná a za implementaci se neplatí žádné licenční poplatky. Díky této skutečnosti a jednoduchosti samotného protokolu je i přes své stáří v dnešní době stále velmi rozšířený, o čemž svědčí skutečnost,že ho stále podporuje většina výrobců programovatelných automatů.

[7] V rámci modelu ISO/OSI se tento protokol nachází na sedmé, čili aplikační vrstvě. Pro vlastní přenos se využívá celé řady komunikačních sběrnic a sítí, zahrnujících RS-232, RS-485, optická vlákna, rádiové vlny, nebo Ethernet s využitím TCP/IP[7]. Protokol Modbus pracuje na principu požadavek/odpověď, přičemž ke sběrnici jsou připojeny dva typy zařízení - klienti a servery. Komunikace poté probíhá způsobem, kdy klient vyšle požadavek (například žádost o zapsání či přečtení určitých dat), zatímco zařízení typu server, kterému je tento požadavek adresován, vykoná požadovanou operaci a následně zašle odpověď podávající informaci o úspěchu či neúspěchu požadované operace, případně požadovaná data. V protokolu Modbus jsou tak definovány celkem tři typy zpráv, požadavky (request), odpovědi (response) a záporné odpovědi (exception response).

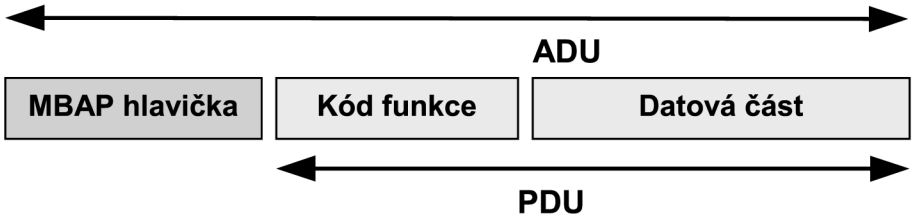


Obrázek 3.2: Průběh komunikace [7]

3.2.1 Formát zprávy

Samotné zprávy tohoto protokolu mají následující podobu. Základ zprávy představuje PDU (protocol data unit). Ta sestává z jednoho bytu, v němž je uložen kód funkce udávající, co za operaci se má provést a proměnného počtu datových bytů. Počet a význam datových bytů závisí na typu požadované operace. Tato část zprávy poskytuje serveru dodatečné informace k provedení operace. Například pokud serveru zašleme příkaz ke čtení dat, datová část obsahuje informaci o počáteční adrese z níž chceme data číst a počtu

hodnot, jež mají být přečteny. U některých typů operací pak mohou být datové byty zcela vynechány.



Obrázek 3.3: Podoba zprávy na protokolu MODBUS TCP/IP [7]

V závislosti na komunikační síti nad níž je tento protokol použit, přidávají se k PDU další části a výsledný celek pak tvoří ADU (application data unit). Ve variantě Modbus TCP/IP, která je použita v této práci se k části PDU připojuje takzvaná MBAP hlavička (ModBus Application Protocol header). Tato hlavička obsahuje sedm bytů, s následujícím významem.

Identifikátor transakce - délka dva byty. Tento indentifikátor pomáha na straně klienta ke každému vyslanému požadavku jednoznačně přiřadit odpovídající odpověď serveru. Server tento prvek kopíruje z požadavku klienta do své odpovědi. [9]

Identifikátor protokolu - délka dva byty. Tento prvek má v případě zprávy protokolu Modbus vždy nastavenu hodnotu 0. Je zde přítomen z důvodu možnosti multiplexování uvnitř systému, pokud jsou využity různé aplikační protokoly. Server tento prvek kopíruje z požadavku klienta do své odpovědi. [9]

Délka - délka 2 byty. Tento prvek udává počet bytů ve zbytku zprávy, čili počet bytů PDU + jeden byte identifikátoru zařízení. [9]

Identifikátor zařízení - délka 1 byte. Pokud některé zařízení v síti funguje jako brána mezi sítí Ethernet TCP/IP a další podsítí založenou například na RS-485, pak prvek identifikátor zařízení slouží k identifikaci zařízení v této podsíti. Server tento prvek kopíruje z požadavku klienta do své odpovědi. [9]

Název tabulky	Délka položky	Přístup	Adresa
Diskrétní vstupy (discrete inputs)	1-bit	čtení	10000-19999
Cívky (coils)	1-bit	čtení/zápis	20000-29999
Vstupní registry (input registers)	16-bitů	čtení	30000-39999
Uchovávací registry (holding registers)	16-bitů	čtení/zápis	40000-49999

Tabulka 3.1: Datový model protokolu Modbus [7]

Další přídavné části, používané např. při komunikaci po RS-232, jako například kontrolní součet CRC, nejsou potřeba, jelikož jsou zajištěny přímo na úrovni TCP/IP protokolu, případně jsou odpovídající údaje obsaženy v MBAP hlavičce.

Maximální délka PDU části zprávy je z historických důvodů omezena na 253 bytů, což při velikosti MBAP hlavičky sedm bytů, dává maximální velikost zprávy protokolu Modbus TCP/IP 260 bytů. Hodnota 253 bytů souvisí s historicky první implementací protokolu Modbus na sériové lince RS-485, kde byla maximální délka zprávy (ADU) definována na 256 bytů. To při dvou bytech vyhrazených pro CRC kód a jednom bytu vyhrazeném pro adresu serveru činí 253 bytů pro PDU. Tento formát se v protokolu Modbus při komunikaci po sériové lince používá dodnes. [7]

3.2.2 Datový model

Datový model protokolu Modbus se skládá ze čtyř tabulek, z nichž každá má určitá specifika. Tyto tabulky jsou umístěny v serveru a slouží pro výměnu dat mezi klientem a serverem. Jednotlivé tabulky se základními informacemi jsou zobrazeny v 3.1 Dvě z tabulek obsahují jednobitové diskrétní hodnoty, přičemž z tabulky diskrétních vstupů může klient pouze číst. Data obsažená v této tabulce, tak mohou být měněna jedině serverem. Do tabulky cívek, tvořené opět jednobitovými hodnotami, může již klient, jak zapisovat, tak z ní číst. Tuto tabulku tak lze použít pro předání dat od klienta serveru. Obdobný význam nastává u tabulek registrů, kdy z tabulky vstupních registrů může klient opět pouze číst, zatímco do tabulky uchovávacích registrů může již i zapisovat. Délka jednoho registru činí 16 bitů. Dle specifikace může každá z tabulek obsahovat až 65536 hodnot. Z důvodu zpětné kompatibility bývá ale adresní prostor rozdělen na bloky o velikosti 10000

položek viz. tabulka 3.1[7]. Způsob, jakým budou tyto tabulky implementovány na straně serveru, není specifikací určeno. Způsob mapování adres předaných po protokolu Modbus na reálné fyzické adresy je zcela závislý na výrobci zařízení. Tabulky mohou mít v paměti zcela oddělené adresní prostory, stejně tak je přípustné libovolné vzájemné překrývání.

3.2.3 Kódy funkcí

Operace, jež se má provést, je v každém požadavku specifikována pomocí kódu funkce, který může nabývat hodnot od 1 do 128. Specifikace těmto hodnotám přiřazuje jim odpovídající funkce. Ty jsou v oficiální specifikaci děleny do tří kategorií.

Veřejné funkce - zabírají většinu funkčního rozsahu, přičemž není ke každému kódu přiřazena funkce, tyto jsou vyhrazeny pro budoucí použití. Všechny veřejné funkce jsou veřejně dokumentované a je garantována jejich jedinečnost, navíc je k nim dostupný test ověřující validitu implementace. Pomocí tohoto testu lze zaručit kompatibilitu s jinými zařízeními implementujícími tento protokol. [8]

Uživatelské funkce - pro potřeby zařízení jimž nepostačují funkce nacházející se ve specifikaci, jsou vyhrazeny kódy v rozsazích 65 až 72 a 100 až 110. Po projednání s Modbus organizací je možné tyto funkce přesunout mezi veřejné funkce [8].

Rezervované funkce - kromě veřejných a uživatelských funkcí existuje ještě třetí kategorie rezervovaných funkcí, které jsou používány některými společnostmi v jejich zařízeních a nejsou veřejně k dispozici [8].

Základní veřejné funkce protokolu Modbus pokrývají většinou čtení z a zápis dat do jednotlivých datových tabulek. K hodnotám v těchto tabulkách můžeme přistupovat, jak jednotlivě, tak po větších blocích, počet hodnot které můžeme přečíst či zapsat najednou je omezen pouze maximální délkou datové části Modbus zprávy.

3.2.4 Příklad komunikace

Komunikace započne vysláním požadavku od klienta serveru. Pokud server vykoná klientem požadovanou operaci, aniž by došlo k chybě, pak do odpovědi zkopíruje MBAP hlavičku požadavku s výjimkou prvku udávajícího délku zbytku zprávy, jež se mění podle

Kód funkce	1 byte	0x03
Počáteční adresa	2 byty	0x0000 - 0xffff
Počet registrů	2 byty	1 - 125

Tabulka 3.2: Požadavek [8]

Kód funkce	1 byte	0x03
Počet bytů	1 byte	2*N
Hodnoty registrů	2*N bytů	

Tabulka 3.3: Odpověď [8]

počtu datových bytů odpovědi, dále do odpovědi zkopíruje kód funkce, kterou vykonal a k tomu připojí datové byty závislé na typu provedené operace. Například datová část požadavku a odpovědi pro čtení více uchovávacích registrů má podobu uvedenou v tabulkách 3.2 a 3.3.

Pokud při vykonávání operace dojde k chybě, server zašle zápornou odpověď. Ta je konstruována tím způsobem, že ke kódu funkce se přičte konstanta 0x80 (128 v decimálním vyjádření), čímž vznikne kód v rozsahu 129-255, jež je pro chyby vyhrazen a do datové části se vloží chybový kód vyjadřující, co bylo příčinou selhání. Ukázka možné podoby záporné odpovědi pro čtení více uchovávacích registrů je zobrazena v tabulce 3.4. Jednotlivé chybové kódy a jejich význam jsou určeny specifikací.

Prakticky komunikace mezi klientem a serverem funguje tak, že klient čte z a zapisuje data do datových tabulek, přičemž obě zařízení musí znát přesný význam dat na konkrétní adrese. Provedení určité operace, která není specifikací určena, aniž bychom definovali vlastní uživatelskou funkci, lze realizovat tím způsobem, že vyhradíme určitou adresu v uchovávacích registrech, či diskrétních cívkách, jejíž nastavení na určitou hodnotu bude serverem vyhodnoceno jako požadavek na vykonání zvolené nestandardní funkce.

Kód funkce	1 byte	0x83
chybový kód	1 byte	0x01,0x02,0x03,0x04

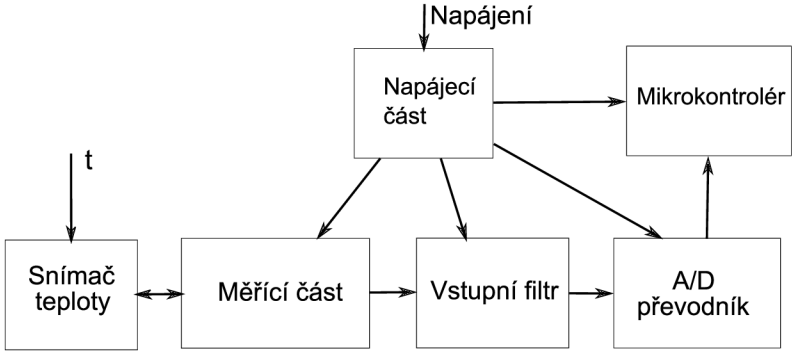
Tabulka 3.4: Záporná odpověď [8]

4 HARDWAROVÉ ŘEŠENÍ

Zapojení převodníku teploty lze rozdělit na dvě základní části. Na část analogovou, obsahující obvody pro získání a úpravu napětí nesoucího informaci o měřené teplotě a na část digitální sestávající z modulu mikrokontroléru Rabbit RCM3900 a obvodů pro konfiguraci měřící části obvodu. Při návrhu analogové části bylo čerpáno především z aplikačních poznámek [10] [11]. Výběr použitých součástek byl podřízen tomu, aby byly snadno dostupné, tzn. bylo je možné sehnat v prodejnách GM Electronic nebo Ges-Electronic. Jedinou výjimkou z tohoto pravidla se stal vstupní DC/DC měnič, jež nebyl přes tyto distribuční kanály dostupný.

4.1 Koncepce

Pokud půjdeme do větší hloubky, můžeme navržené schéma zapojení převodníku rozdělit do několika funkčních bloků viz. obrázek 4.1. Odporový snímač teploty je připojen k měřící části, ta se skládá z odporového můstku a zesilovače pro úpravu amplitudy výstupního napětí můstku. V zesíleném napětí jsou následně potlačeny vyšší frekvence aktivním filtrem a napětí se převede do digitální podoby pomocí A/D převodníku. Digitální reprezentace měřeného napětí je dále zpracována modulem RCM3900 mikrokontroléru Rabbit, sloužícím jako řídící prvek celé aplikace. Jednotlivé funkční bloky budou následně popsány.



Obrázek 4.1: Blokové schéma převodníku

4.2 Blok napájení

Převodník byl navržen pro napájení stejnosměrným napětím o velikosti 12 V. Toto napětí je potřeba snížit na hodnotu 5 V, která je v převodníku použita pro napájení některých obvodů a dále upravena na hodnotu 3.3 V a -5 V. Tabulka 4.1 zobrazuje proudový odběr v zapojení obsažených obvodů, či funkčních bloků. Největší proudový odběr v zapojení

Prvek	Označení	odběr
RCM3900	$I_{RCM3900}$	325 mA
MCP3202	$I_{A/D}$	550 μA
MPC602	I_{MCP602}	460 μA
LED dioda	I_{LED}	10 mA
Měř. obvod	I_{MER}	3.1 mA

Tabulka 4.1: Proudový odběr jednotlivých obvodů převodníku

vykazuje modul mikrokontroléru Rabbit RCM3900. Výrobce udává typickou hodnotu odebíraného proudu 325 mA. Proudový odběr měřícího obvodu I_{MER} se skládá z maximálního odběru, který byl vypočten pro měřící odporový můstek a odběru přístrojového zesilovače, jehož maximální hodnotu výrobce udává rovnu 1.3 mA. LED diod s jmenovitým proudem o hodnotě 10 mA se v zapojení nachází celkem šest, do celkového odběru je tedy nutné tuto hodnotu započítat šestkrát.

$$I = I_{RCM3900} + I_{A/D} + 6\,I_{LED} + I_{MER} + I_{MCP602} = 389.1\ mA \tag{4.1}$$

Z důvodu značného proudového odběru modulu RCM3900, nebylo možno pro snížení vstupního napětí použít běžného lineárního stabilizátoru, výkonová ztráta na tomto obvodu by v případě napájení 12 V dosahovala hodnoty

$$P = (U_{NAP} - U_{POZ}) * I = (12 - 5) * 0.389 = 2.72\ W \tag{4.2}$$

Tato hodnota navíc není konečná, jelikož do proudového odběru nebyla započítána spotřeba obvodů napájecího bloku zapojení, rovněž je nutné počítat s určitou rezervou. S přihlédnutím k těmto skutečnostem bylo k předpokládanému zatížení připočteno 30 % z vypočtené hodnoty. Výsledná výkonová ztráta v tom případě činí 3.54 W. Z tohoto

důvodu byl využit DC/DC měnič Tsr1-2450 od firmy Traco s maximálním výstupním proudem 1 A, fungující na principu spínaného regulátoru. Tento princip zajišťuje oproti lineárním stabilizátorům minimální výkonovou ztrátu. Výrobce udává účinnost tohoto měniče až 96 %. Rozsah vstupních napětí tohoto obvodu se pohybuje v rozmezí $6.5 - 32\ V$ [13].

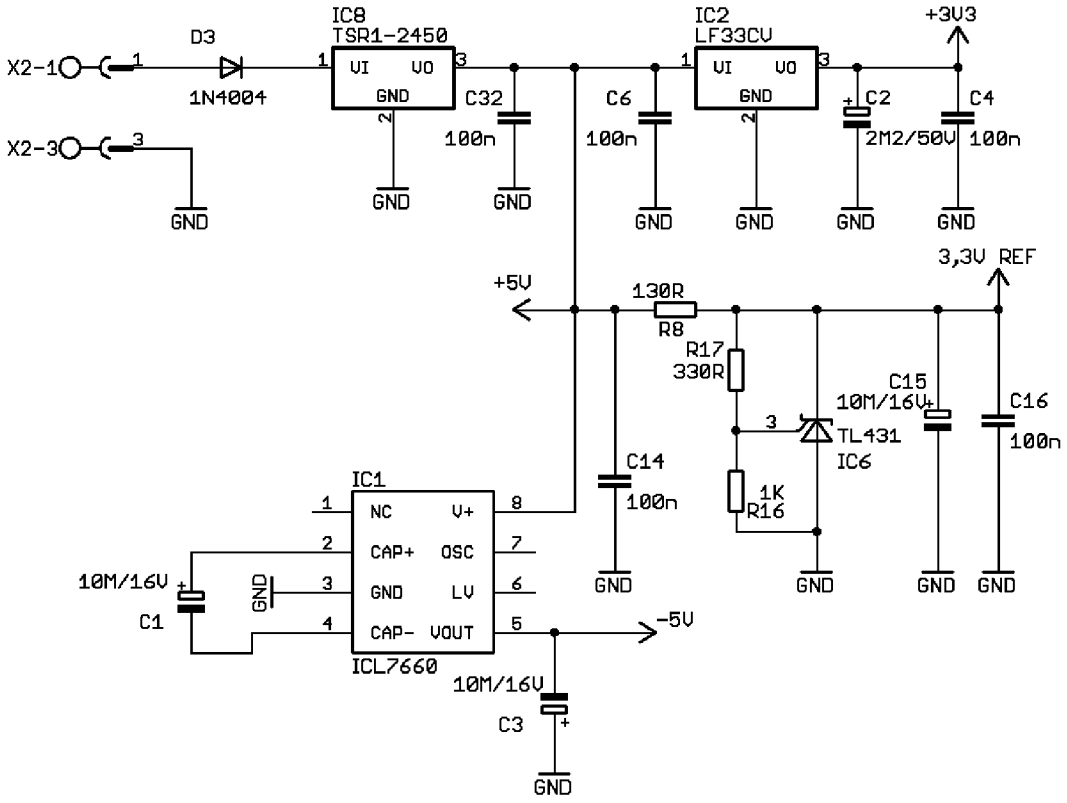
4.2.1 Uspořádání napájecí části

Na vstupu napájecí části (bezprostředně za napájecím konektorem) se nachází ochranná dioda zapojená v propustném směru. Tato dioda v zapojení slouží jako ochrana proti přepólování napájecího napětí. V zapojení musíme počítat s úbytkem napětí na této diodě, o který bude vstupní napájecí napětí sníženo. Toto napětí činí dle výrobce, v případě použité diody typu 1N4004, 1.1 V. Tato velikost se však vztahuje na případ, kdy diodou protéká maximální povolený proud o velikosti 1 A [12]. V reálných podmínkách převodníku teploty, tak úbytek napětí činí 0.8 V. Pokud úbytek na diodě přičteme k nejnižší možné vstupní hodnotě napětí DC/DC měniče (dle výrobce 6.5 V), získáme minimální napětí, na které musí být převodník připojen. Toto minimální napětí činí 7.3 V. Za ochrannou diodou se nachází DC/DC měnič Tsr1-2450 transformující vstupní napětí na hodnotu 5 V používanou dále v obvodu. Toto výstupní napětí je již bezprostředně použito pro napájení použitých zesilovačů a obvodu 70HC03 obsahujícího čtyři NAND hradla.

V zapojení potřebujeme celkem čtyři druhy napájecího napětí pro jednotlivé obvody. Napětí 3.3 V pro napájení modulu RCM3900, přesné referenční napětí 3.3 V sloužící jako napájení A/D převodníku a měřícího můstku, dále napětí 5 V pro napájení zesilovačů a klopného obvodu a napětí $-5\ V$ pro zajištění symetrického napájení přístrojového zesilovače AD620, zesilujícího měronosné napětí.

Napětí o hodnotě 3.3 V použité pro napájení modulu RCM3900 je získáno pomocí lineárního regulátoru LF33CV, snižujícího napětí z 5 V dodávaných DC/DC měničem Tsr1-2450. Tento obvod napájí pouze modul mikrokontroléru a výkonová ztráta na tomto obvodu činí

$$P = U_{ROZ} * I_{RCM3900} = 1.7 * 0.325 = 0.55\ W \tag{4.3}$$



Obrázek 4.2: Napájecí část převodníku

4.2.2 Obvod TL431

Vzhledem k pravděpodobnému zarušení napájecího napětí použitého pro napájení digitálních obvodů, není vhodné napětí 3.3 V použité pro napájení mikrokontroléru Rabbit, použít pro napájení A/D převodníku a měřícího odporové můstku. Pro získání přesné napěťové reference pro napájení A/D převodníku zapojení obsahuje napěťový regulátor TL431. Principiálně se tento obvod chová jako zpětnovazební regulátor s vnější napěťovou zpětnou vazbou a s vnitřní referencí 2,5 V [14]. Kromě nezbytných blokovacích kondenzátorů, jsou pro správnou činnost obvodu potřeba další tři pasivní součástky. Dva rezistory nastavující výstupní napětí a jeden sériový rezistor snižující vstupní napětí. Zapojení obvodu TL431 ve schématu, lze spatřit na obrázku 4.2.

Hodnoty rezistorů R16 a R17 lze stanovit ze vzorce

$$V_{OUT} = V_{REF}(1 + \frac{R17}{R16}) \tag{4.4}$$

, přičemž námi požadované napětí je $V_{OUT} = 3.3\ V$ a referenční napětí samotného obvodu

činí $V_{REF} = 2.5\ V$ [14]. Daným podmínkám vyhovují rezistory o hodnotách $R17 = 330\Omega$ a $R16 = 1k\ \Omega$. Při použití uvedených hodnot získáme z rovnice výstupní napětí rovné $U_{OUT} = 3.325\ V$. Pro přesný výpočet výstupního napětí by bylo třeba k vypočtenému napětí připočíst úbytek napětí, způsobený proudem protékajícím střední referenční elektrodou. Proud protékající referenční elektrodou dosahuje, dle výrobce běžně velikosti jednotek μA, proto byl při výpočtu zanedbán.

Pro stabilizaci výstupního napětí obvodu potřebujeme aby proud horní elektrodou (ve schématu znázorněna jako katoda zenerovy diody) dosahoval alespoň hodnoty 1 mA. Hodnota sériového snižujícího rezistoru R26 musí být koncipována tak, aby při daném úbytku napětí na něm rovném 1.7 V, tímto rezistorem procházel proud o velikosti kolem 8 mA. Tato hodnota byla určena z proudového odběru A/D převodníku a měřícího můstku, jež nepřekročí hodnotu 3 mA, proudu protékajícího odpory R16 a R17, který se rovná 2.5 mA a určité rezervy. Tyto požadavky splňuje rezistor o hodnotě 130 Ω.

4.2.3 ICL7660

Pro napájení zesilovače AD620 potřebujeme symetrické napětí $\pm 5\ V$. Napětí -5 V se v zapojení získává pomocí obvodu ICL7660, pracujícího na principu nábojové pumpy. Tento obvod při vstupním napětí $1,5$ až 12 V, vytváří na svém výstupu inverzní napětí o hodnotách $-1,5$ až $-10\ V$. Kromě toho lze tento obvod využít i jako zdvojovač, dělič nebo násobič napětí. Pro inverzi napětí, využívanou v tomto zapojení, je potřeba k obvodu připojit dva externí elektrolytické kondenzátory o kapacitě 10 μF (hodnota uvedená v datasheetu [15]).

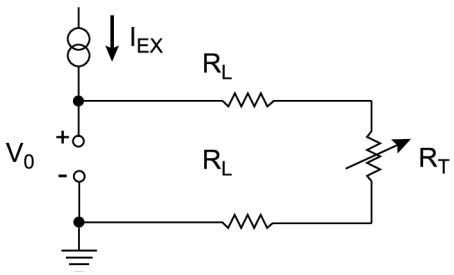
4.3 Měřící část

Jak již bylo zmíněno v kapitole zabývající se odporovými teplotními snímači, měření teploty těmito snímači je založeno na skutečnosti, že velikost elektrického odporu snímače je přímo závislá na měřené teplotě a tedy i úbytek napětí na tomto snímači je na teplotě závislý. Snímač se doporučuje napájet ze zdroje konstantního proudu, konstantní proud protékající snímačem zajistí, že závislost mezi změnou elektrického odporu snímače a

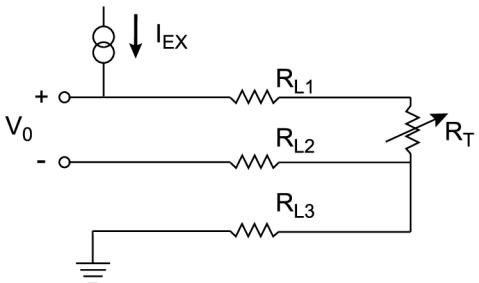
změnou úbytku napětí na snímači bude lineární. V tomto zapojení bylo zvoleno pro jednoduchost napájení z napěťového zdroje, tím sice zanášíme do měřícího obvodu určitou nelinearitu, ta však nepřekračuje linearitu samotných snímačů a spolu s ní bude číslicově potlačena.

4.3.1 Možné způsoby připojení snímače do měřícího obvodu

Pro připojení odporového snímače lze využít jedno z následujících tří uspořádání. Dvouvodičové připojení snímače využívá nejmenší počet přívodních vodičů a je tedy z možných způsobů připojení finančně nejméně nákladné, avšak výrazně se u něj projeví chyba způsobená úbytkem napětí na přívodních vodičích, jimiž snímač připojujeme do měřícího obvodu. Tato chyba klesá se zvyšujícím se jmenovitým odporem použitých snímačů a u snímačů s jmenovitým odporem 10 $k\Omega$ ji již lze považovat za zanedbatelnou. V případě snímače Pt100 tato chyba však může v závislosti na délce přívodních vodičů dosahovat i jednotek $°C$. Pokud například budeme počítat, že elektrický odpor jednoho z vodičů bude 0.3 Ω a celkový odpor obou vodičů tedy 0.6 Ω, bude chyba způsobená odporem přívodních vodičů při použití snímače Pt 100 s hodnotou $\alpha = 0.00385$ činit $1.6° C$. [10]



Obrázek 4.3: Dvouvodičové připojení [10]

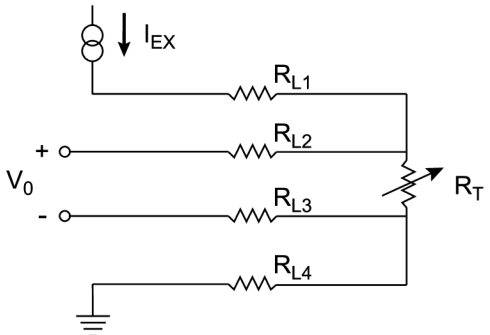


Obrázek 4.4: Třívodičové připojení [10]

Pro ušetření nákladů oproti čtyřvodičovému připojení lze použít třívodičové připojení zobrazené na obrázku 4.4

Čtyřvodičové připojení využívá dva vodiče pro napájení snímače a dva vodiče pro přenos napětí nesoucího informaci o měřené teplotě 4.5. Tento způsob připojení snímače jako jediný zajistí úplné potlačení chyby způsobené odporem přívodních vodičů. Pokud

bude vstupní odpor zařízení, na nějž je výstupní napětí snímače přiváděno vysoký, v řádu $G\Omega$, pak vodiči v obrázku označenými jako L2 a L3 bude procházet pouze minimální proud a rovněž úbytek napětí na těchto vodičích bude zanedbatelný. Nevýhodou této metody jsou pak vyšší náklady na připojení související s jedním, případně dvěma, přídavnými vodiči oproti třívodičové variantě.

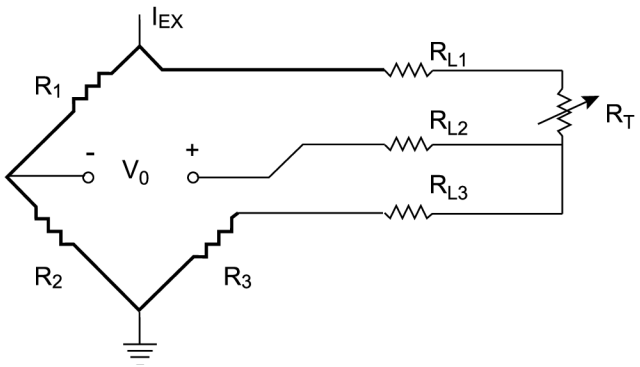


Obrázek 4.5: Čtyřvodičové připojení [10]

4.3.2 Připojení snímače do měřícího obvodu

Pro použití v této práci bylo zvoleno třívodičové připojení měřícího snímače do odporového Wheatstoneova můstku. Výstupní napětí v tomto případě představuje rozdílové napětí mezi rameny můstku označené V_0 viz. obrázek 4.6. Použití měřícího můstku v této aplikaci je výhodné zejména z důvodu možnosti vyvážení můstku tak, aby při nejnižší teplotě z měřícího rozsahu bylo výstupní rozdílové napětí V_0 nulové. V případě připojení zobrazeného na obr. 4.4 by napětí, které dále zesilujeme a převádíme do číslicové podoby, bylo větší o úbytek napětí na snímači při nejnižší teplotě rozsahu. Tento úbytek napětí je v případě zapojení do můstku odečten. Toto přídavné napětí pro naše účely nenese žádnou informaci, pouze by snížilo měřitelný rozsah teplot pro dané rozlišení přibližně o jednu třetinu.

Kromě třívodičového způsobu připojení měřícího snímače, zapojení rovněž obsahuje možnost, připojení snímače dvěma vodiči. K tomuto účelů slouží na desce přítomný jumper JP1. Pomocí něho lze připojit vstup přístrojového zesilovače k vodiči, jímž je snímač při třívodičovém způsobu připojení napájen. K připojení snímače poté stačí pouze dva vodiče.



Obrázek 4.6: Třívodičové připojení do odporového můstku [10]

4.3.3 Použité odporové snímače

Pro ověření funkčnosti převodníku byli k dispozici odporové snímače teploty od firmy SENSIT s.r.o. Konkrétně se jedná o snímače Ni1000/5000, Ni1000/5000 a Pt100/3850. Platinový snímač Pt100 byl zvolen z toho důvodu, že niklový snímač o podobné jmenovité hodnotě se v nabídce firmy nenachází.

Typ	Jmenovitý odpor(při 0 °C)	Teplotní rozsah °C	Změna odporu Ω/° C	Doporučený proud (mA)	Maximálníss proud (mA)
Pt 100/3850	100 Ω	-50 až 400	0.385	1	3
Ni 1000/5000	1000 Ω	-60 až 250	6.18	0,3	1
Ni 10000/5000	10000 Ω	-60 až 250	61.8	0,1	0,5

Tabulka 4.2: Parametry senzorů teploty [20] [19] [18]

Doporučené hodnoty měřícího proudu uvedené výrobcem se nedoporučuje překračovat, z důvodu možného ovlivnění měření samoohřevem snímače. Za samoohřev se označuje stav, kdy se snímač samovolně zahřívá vlivem procházejícího měřícího proudu.

Jelikož v době návrhu aplikace nebyl znám přesný typ snímačů, pro něž má být měřící obvod navrhnut, byla měřící část zapojení navrhována pro využití snímačů Pt100/3850, Ni1000/6180 a Ni10000/6180 v teplotním rozsahu -50 až 180 °C.

Snímač	Odpor při -50 ° C	Odpor při 200 ° C
Pt100/3850	80.31	175.86
Ni1000/5000	790.9	2137.0
Ni10000/5000	7908.8	21369.6

Tabulka 4.3: Odpor snímačů při mezních teplotách měřícího rozsahu

Rovnice popisující závislost elektrického odporu snímačů na teplotě jsou popsány, spolu s příslušnými koeficienty nutnými pro výpočet, níže. Uvedené vztahy se týkají snímačů jež byli uvedených v tabulce 4.2

Pt100: $R = 100(1 + At + Bt^2 + C(t-100)t^3)$ v rozsahu -50 až 0 °C [18]

$R = 100(1 + At + Bt^2)$ v rozsahu 0 až 400 °C [18]

$A = 3.9083x10^{-3}{}^{\circ}C^{-1}$ $B = -5,775x10^{-7}{}^{\circ}C^{-2}$ $C = -4.183x10^{-12}{}^{\circ}C^{-4}$

Ni1000: $R = 1000(1 + At + Bt^2 + Ct^3)$ [19]

Ni10000: $R = 10000(1 + At + Bt + Ct^3)$ [20]

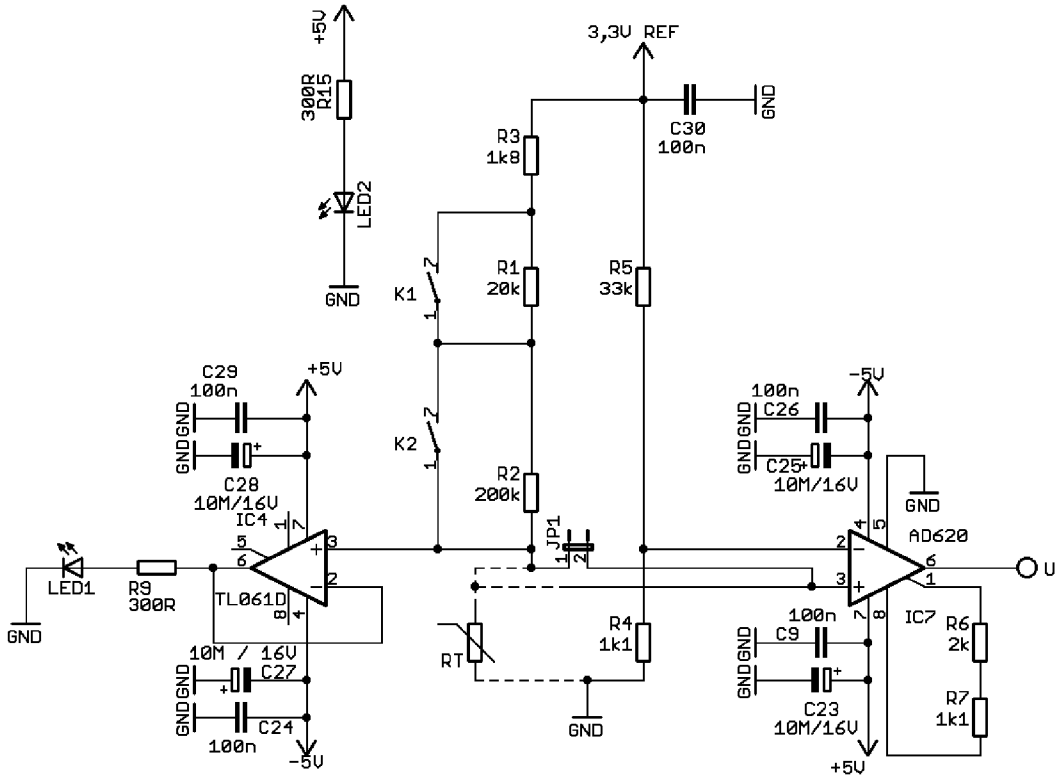
$A = 4,427x10^{-3}{}^{\circ}C^{-1}$ $B = 5,172x10^{-6}{}^{\circ}C^{-2}$ $C = 5,585x10^{-9}{}^{\circ}C^{-3}$

Tabulka 4.3 zobrazuje hodnotu odporu jednotlivých snímačů při mezních teplotách rozsahu. Uvedený rozsah se liší od rozsahu plánovaného. Důvodem je především nedostupnost pasivních součástek s přesnými hodnotami elektrického odporu, pro něž by bylo možné tento rozsah zajistit.

4.3.4 Konfigurace měřícího můstku

Aby bylo možné měřit teplotu pomocí odporových snímačů teploty s různou jmenovitou hodnotou odporu, jak je požadováno v zadání práce, bylo nutné do měřícího můstku implementovat rozhraní díky němuž by větev můstku, do níž se připojuje měřící snímač, byla konfigurovatelná z hlediska velikosti proudu touto větví protékajícího. Přizpůsobení velikosti proudu pro různé snímače je nezbytné, jednak z důvodu různých hodnot měřícího proudu, jež jsou u jednotlivých snímačů povoleny (dodržení by mělo zabránit ovlivnění měření samoohřevem snímače) a jednak proto, aby došlo k přizpůsobení výstupního napětí z měřícího můstku na přibližně stejnou úroveň pro všechny snímače a nebylo

tedy potřeba přepínací funkčnost implementovat i do dalších částí obvodu. Přepínání odporu měřící větve bylo implementováno s pomocí dvou mikrokontrolérem programově ovládaných relé, která mění hodnotu odporu levé horní větve měřícího můstku. Levá horní větev můstku, tak sestává ze tří rezistorů ve schématu označených R1, R2 a R3, z nichž dva jsou paralelně připojeny ke dvěma magnetickým relé. Pokud je relé sepnuto a elektrický proud prochází skrze sepnuté relé. Úbytek napětí na této paralelní kombinaci lze v takovém případě zanedbat. Hodnoty odporu této větve při různé konfiguraci relé zobra-



Obrázek 4.7: Měřicí část zapojení

zuje tabulka 4.4. Varianta, kdy by bylo sepnuto pouze relé K1, v zapojení nemá využití. Výsledný odpor, resp. protékající proud by se příliš nelišil od stavu, kdy není sepnuto žádné relé.

Konfigurace měřícího můstku číslo jedna je vhodná zejména pro snímače s doporučenou hodnotou měřícího proudu 1 *mA* při jmenovité hodnotě elektrického odporu 100 Ω. Aby stejný obvod pro zesílení výstupního napětí z můstku bylo možné použít i pro snímače s desetinásobnou jmenovitou hodnotou elektrického odporu, musí mít měřící

Číslo konfigurace	Sepnutá relé	Odpor větve ($k\omega$)
1	K1, K2	1.8
2	K2	21.8
3	Žádné	221.8

Tabulka 4.4: Možné konfigurace odporového můstku

proud desetkrát menší hodnotu. Konfigurace číslo dvě a tři tedy slouží pro měřící proudy o přibližné hodnotě 0.1 a 0.01 mA. V této aplikaci bude konfigurace číslo jedna použita pro měření pomocí snímače Pt100, konfigurace číslo dvě pro měření pomocí snímače Ni1000 a konfigurace číslo tři, pro měření pomocí snímače Ni10000.

4.3.5 Přístrojový zesilovač AD620

Jelikož výstupní napětí z odporového můstku se pohybuje v rozmezí $0-0.2\ V$, je toto napětí před převodem do číslicové podoby, třeba zesílit na takovou úroveň, aby byl zcela využit rozsah A/D převodníku, na nějž můžeme přivést napětí v rozmezí $0-3.3\ V$. Pro zesílení je možné použít rozdílové zapojení operačního zesilovače. Toto zapojení má však nevýhodu spočívající v malé vstupní impedanci. Pro zesílení byl proto zvolen přístrojový zesilovač AD620 od firmy Analog Devices. Principiální schéma přístrojového zesilovače je zobrazeno na obrázku 4.8. Hlavní výhoda přístrojového zesilovače, kvůli níž byl v této aplikaci použit, spočívá ve vysoké vstupní impedanci této struktury. Ta je dána vstupní impedancí na vstupu připojených operačních zesilovačů, která bývá v řádu $G\Omega$. Vodiči přivádějícími zesilované rozdílové napětí, tak protéká pouze minimální proud a úbytek napětí na nich lze zanedbat.

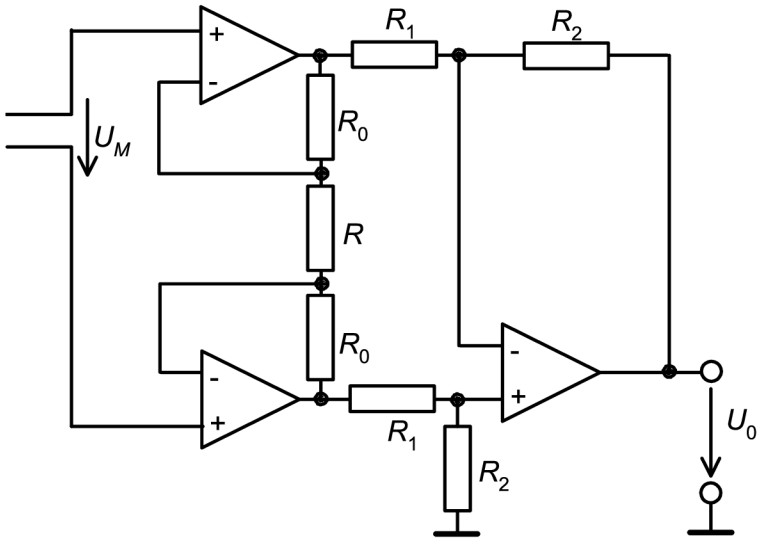
Zesílení tohoto obvodu se nastavuje pomocí rezistoru připojeného mezi vývody v [17] označené R_G(na obrázku 4.7 označeny 1 a 8). Výsledné zesílení lze spočítat z rovnice

$$G = \frac{49.4k\Omega}{R_G} + 1 \tag{4.5}$$

[17] Z toho pro výpočet velikosti rezistoru R_G získáme

$$R_G = \frac{49.4k\Omega}{G-1} \tag{4.6}$$

Pro požadované zesílení 16.9-krát vychází rezistor o hodnotě 3.1 $k\Omega$. Pro přesnější nastavení byl požadován rezistor s přesností 0.1 %, této podmínce vyhovuje kombinace rezistorů 2k a 1k1. [17]



Obrázek 4.8: Obecné zapojení přístrojového zesilovače [16]

4.3.6 Signalizační dioda

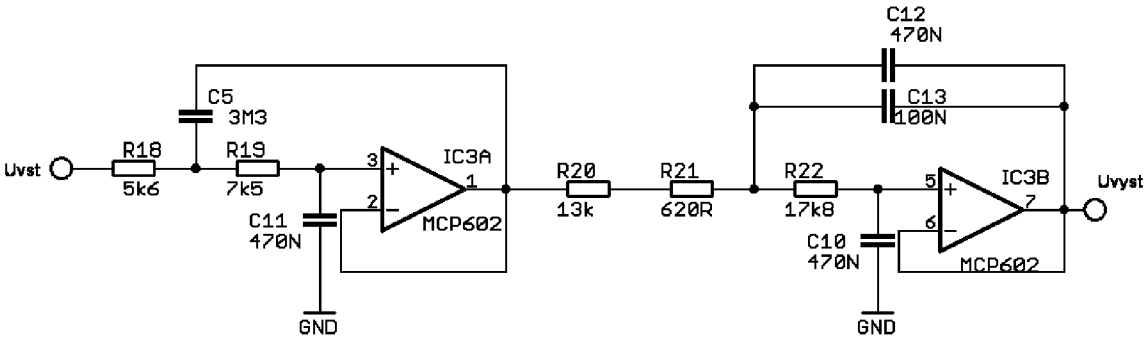
Operační zesilovač TL061, patrný na obrázku vlevo dole 4.7, je zde zapojen jako sledovač napětí a slouží k impedančnímu oddělení měřícího můstku od obvodu diody. LED dioda je napájena přes rezistor o hodnotě 300 Ω přímo z operačního zesilovače. Neinvertující vstup operačního zesilovače byl připojen doprostřed konfigurovatelného ramene můstku, do bodu, k němuž je připojen měřící snímač. Dioda slouží k výstražné signalizaci, že k převodníku není připojen žádný snímač. Pokud je k převodníku snímač připojen dioda nesvítí, to je dáno skutečností, že napětí na neinvertujícím vstupu operačního zesilovače dosahuje pouze hodnoty kolem 0.2 V (úbytek napětí na snímači). Jelikož v případě zapojení operačního zesilovače jako sledovače napětí, bude na výstupu stejná hodnota napětí jako na neinvertujícím vstupu, LED dioda nebude svítit, jelikož výstupní napětí operačního zesilovače nemá dostatečnou hodnotu (úbytek napětí v propustném směru na použité diodě činí 2.2 V). Pokud k převodníku není žádný snímač připojen, neinvertující

vstup operačního zesilovače se bude nacházet na úrovní napájecího napětí $3.3V$ a tato hodnota bude i na výstupu operačního zesilovače.

Na tomto místě je nutné podotknout, že tato část zapojení související se signalizační LED diodou je v obvodu z dnešního pohledu zcela zbytečná. Rozsvícení LED diody v situaci, kdy k převodníku není připojen žádný teplotní snímač, lze zajistit i programově, A/D převodník v této situaci bude saturován, čili výstupem bude maximální hodnota, což lze v řídícím programu detekovat a LED dioda LED1, by tak mohla být k převodníku připojena stejným způsobem jako diody LED3,4,5,6. Navíc původní záměr počítal se skutečností, že tato LED dioda bude svítit pouze v případě, kdy k převodníku není připojen žádný snímač. K rozsvícení však dojde i za situace nevhodně nakonfigurovaného měřícího můstku. Pokud například bude připojen snímač Ni10000 a měřící můstek bude v konfiguraci pro použití snímače o jmenovité hodnotě 100 Ω (všechna relé se tedy budou nacházet v sepnutém stavu), poměr odporů ve větvích můstku bude přibližně 2:10 a na neinvertujícím vstupu operačního zesilovače se tak bude nacházet napětí blízké napájecímu napětí můstku.

4.4 Filtrace měřeného napětí

Pro potlačení vyšších frekvenčních složek v měřeném signálu byl mezi zesilovač a A/D převodník zařazen aktivní filtr čtvrtého řádu. Tento filtr sestává z obvodu MCP602 obsahujícího dva operační zesilovače a několika pasivních součástek. Filtr byl navržen pomocí programu FilterLab zdarma poskytovaným firmou Microchip. Filtr byl navržen tak, aby potlačoval, již brum sítě o frekvenci 50 Hz. Jedná se o dolní propust a frekvence zlomu (vstupní signál o této frekvenci bude potlačen o -3 dB) nastává na frekvenci 20 Hz. Jelikož celý převodník je koncipován pro měření teploty soustav s velkými časovými konstantami, v nichž ke změně teploty dochází pouze pomalu, nároky na filtr se v této aplikaci týkají spíše potlačení rušení, než dodržení vzorkovacího teorému.



Obrázek 4.9: Aktivní filtr

4.5 Převedení napětí do číslicové podoby

Napětí z výstupu Anti-aliasing filtru je přivedeno na kanál Ch0 A/D převodníku MCP3202. Rezistor R23, připojený mezi výstup filtru a vstup A/D převodníku, slouží k omezení velikosti procházejícího proudu ve chvíli, kdy k převodníku není připojen žádný snímač. V případě rozpojeného odporového můstku se na výstupu přístrojového zesilovače AD620 objeví jeho napájecí napětí o velikosti 5 V. Uvnitř struktury A/D převodníku se na vstupu nachází dioda o úbytku 0.6 V připojená v propustném směru proti zemi [21]. Připojení napětí 5 V na vstup by způsobilo značný zkratový proud protékající touto diodou.

K saturaci přístrojového zesilovače dojde rovněž v okamžiku, kdy k převodníku připojíme snímač o nižším jmenovitém odporu než odpovídá aktuální konfiguraci přepínacího rozhraní.

4.5.1 A/D převodník MCP3202

Jedná se o 12-bitový dvoukanálový A/D převodník od firmy Microchip, mezi jehož základní vlastnosti patří [21]:

- Přesnost 1 LSB
- Napájecí napětí v rozmezí $2.7 - 5.5$ V
- Nízký proudový odběr 550 μA, ve stavu stand-by pouze 500 nA

- Max. frekvence vzorkování 100 kHz při napájení 5 V a 50 kHz při napájení 2.7 V

Kromě napětí přiváděného na jeden ze dvou vstupních kanálů, může být převáděno i rozdílové napětí vstupních kanálů. Kanály lze poté nakonfigurovat jako vstupy IN+ a IN-, přičemž se odečítá napětí kanálu IN- od kanálu nakonfigurovaného jako IN+. Nevýhoda v případě tohoto A/D převodníku spočívá ve skutečnosti, že napětí přivedené na IN- může dosahovat pouze úrovně ± 100 mV oproti napětí referenčnímu. [21] 12-bitové rozlišení převodníku nám poskytuje 4096 kvantizačních úrovní. Pokud budeme počítat s teplotním rozsahem -50 až 200 $°C$ pak z $250/4096$ získáme maximální možné rozlišení teploty rovnající se 0.06 $°C$. Při přesnosti převodníku 1 LSB můžeme teplotu teoreticky měřit s přesností 0.12 $°C$.

4.5.2 SPI komunikační rozhraní A/D převodníku

Jako komunikační rozhraní využívá tento A/D převodník sériové rozhraní SPI. Pro plně duplexní datovou komunikaci slouží piny v kontextu rozhraní SPI běžně označované jako MISO a MOSI. Pokud zařízení vystupuje v roli mastera, tak pin označený MISO, tohoto zařízení, slouží jako vstup a pin MOSI jako výstup. U zařízení pracujícího v režimu slave je tomu právě naopak. A/D převodník pracuje v režimu slave a piny pro komunikaci jsou u něj označeny názvy DIN a DOUT. Vývod SCK slouží jako zdroj hodinového signálu využívaného pro synchronizaci komunikace s převodníkem. Posledním standardním pinem SPI komunikačního rozhraní je CS sloužící k vybrání zařízení, jež bude po tomto rozhraní komunikovat. To se využívá hlavně v případě připojení více zařízení k datové SPI sběrnici. V případě tohoto převodníku daný pin nese označení CS/SHDN (Chip-Select/SHutDowN). A/D převodník se nachází v aktivním stavu, pokud je na tento pin přivedena napěťová úroveň log. 0.

4.6 Modul RCM3900

Pro řízení převodníku byl použit modul Rabbit RCM3900. Hlavním přínosem tohoto modulu je skutečnost, že modul již obsahuje integrované komunikační rozhraní Ethernet, které má převodník využívat pro přenos dat. Modul využívá mikrokontroléru Rabbit řady

3000 a rovněž obsahuje slot pro připojení karty standardu miniSD. Celý modul má navíc poměrně malé rozměry 47 mm x 69 mm [22].

Základní vlastnosti modulu RCM3900:

- Obsahuje 8-bitový mikrokontrolér Rabbit řady 3000, běžící na frekvenci 44,2 MHz.
- Pracovní rozsah teplot -20 až 85 $°C$.
- Napájecí napětí 3,3V, je však kompatibilní s 5 V logikou.
- Podpora paměťových karet miniSD do velikosti 1 GB.
- Integrované komunikační rozhraní Ethernet 10/100Base-T.
- Integrovaný obvod hodin reálného času.
- 512 kB rychlé SRAM paměti pro data a dalších 512 kB pro program + 512 kB flash paměti.
- Šest sériových komunikačních portů. Čtyři z nich je možno nakonfigurovat pro komunikaci po rozhraní SPI.

4.6.1 Mikrokontrolér Rabbit 3000

Mikrokontrolér Rabbit řady 3000, jež je v modulu obsažen, vychází z procesorů rodiny Zilog Z80/Z180. Byl koncipován pro řízení embedded aplikací, s čímž souvisí jeho nízké emisní vyzařování. V nabídce firmy jsou již k dispozici vyšší řady mikrokontrolérů 4000, 5000 a 6000, jež pracují na vyšším kmitočtu a nabízejí více integrovaných periferií. Mikrokontroléry této řady neobsahují paměť na čipu, pro paměť dat a programu je potřeba použít externí paměti, jež se v tomto případě nacházejí přímo na těle modulů [22].

Mikrokontrolér má sedm vstupně/výstupních paralelních portů označených písmeny abecedy A až G. Každému portu přísluší osm pinů, celkem tedy má tento mikrokontrolér 56 vstupně/výstupních pinů (ne všechny jsou na desce RCM3900 vyvedeny), které mohou sloužit jako binární vstup nebo výstup. Většina těchto pinů není pouze vstupně/výstupní, ale je sdílena s některým z v mikrokontroléru integrovaných periferních zařízení. [22]

Obrázek 4.10: Modul RCM3900 [23]

4.6.2 Programování mikrokonotroléru

Pro programování mikrokontroléru firma Rabbit semiconductor poskytuje zdarma vývojové prostředí Dynamic C. Součástí vývojového prostředí je celá řada knihoven pokrývající většinu ze zadání vyplývajících potřeb vyvíjeného převodníku teploty. V práci jsou konkrétně využity knihovny pro komunikaci přes komunikační rozhraní SPI a Ehernet, TCP/IP stack, knihovna HTTP serveru, knihovna pro prácí s paměťovou kartou miniSD a knihovna pro smtp. Pro připojení programátoru modul obsahuje deseti-pinový konektor. Na straně PC se programovací kabel připojuje k sériovému portu (existuje i varianta s programovacího kabelu pro připojení přes USB). Pro přepnutí mikrokontroléru do programovacího módu stačí pouze připojit programátor k modulu a restartovat. Modul mikrokontrolér se po restartu spustí v programovacím módu automaticky (rozpoznání v jakém módu se má procesor spustit se odehraje na základě pinu SMODE, na kterém se při připojeném programátoru nachází úroveň log.1, zatímco pokud programátor není připojen tento pin je na úrovni log.0) [22]

4.6.3 Ethernetový port

Modul lze k síti ethernet připojit skrze standardní konektor RJ-45. Převodník podporuje standard sítě 100Base-T, umožňující komunikaci rychlostí až 100Mb/s. Ethernetový čip nacházející se v modulu podporuje automatické rozlišení, zda je k modulu připojen křížený či přímý ethernetový kabel, to umožňuje nezávislost převodníku na typu použitého kabelu a k převodníku lze připojit libovolný utp nebo stp kabel[22].

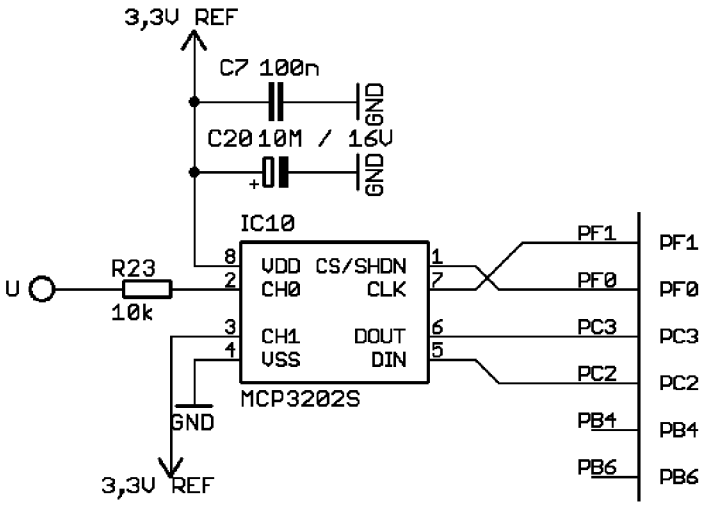
4.6.4 Připojení paměťové karty k převodníku

Pro možnost záznamu velkého objemu dat je na modulu přítomen slot pro připojení paměťové karty typu miniSD. Paměťové karty z rodiny karet SD, miniSD a microSD jsou v dnešní době velmi často používány jako záznamové médium v digitálních fotoaparátech nebo mobilních telefonech. V embedded zařízeních je jejich použití výhodné zejména z důvodu snadné a rychlé výměny a vzhledem k malým rozměrům snadné přenositelnosti. V aplikaci převodníku teploty bude paměťová karta použita pro sběr hodnot teploty a tím realizovat zadáním požadovanou funkčnost dataloggeru. Karty typu miniSD již v dnešní době nejsou, na rozdíl od variant microSD a SD, běžně dostupné.Proto byla na místě paměťové karty pro záznam naměřených teplot použita paměťová karta typu microSD, tu lze do slotu pro miniSD kartu zapojit při použití adaptéru mezi těmito dvěma typy karet. Rozdíly mezi kartami typu SD, miniSD a microSD jsou pouze mechanického charakteru a při použití korektního adaptéru je tedy možné kartu microSD za kartu miniSD zaměnit.

4.6.5 Připojení A/D převodníku k modulu mikrokontroléru

Mikrokontrolér obsahuje šest sériových komunikačních portů označených A až F. Z toho první čtyři hardwarově podporují použití rozhraní SPI. Port A je využit pro připojení programátoru, proto ho, pokud nechceme přijít o možnost využití debuggeru, nelze pro připojení A/D převodníku využít. Sériový port B je na modulu RCM3900 využit pro komunikaci s miniSD paměťovou kartou. A/D převodník tedy byl připojen k sériovému portu C. Hodinový signál pro sériový port C se nachází na pinu PF1. Pro propojení s pinem CS A/D převodníku může být použit kterýkoliv z vstupně/výstupních vývodů modulu mikrokontroléru, tento pin nastavený jako výstup bude pouze měnit své stavy mezi

hodnotou log.1 a log.0 ve chvíli, kdy chceme s A/D převodníkem komunikovat.



Obrázek 4.11: Připojení A/D převodníku k mikrokontroléru

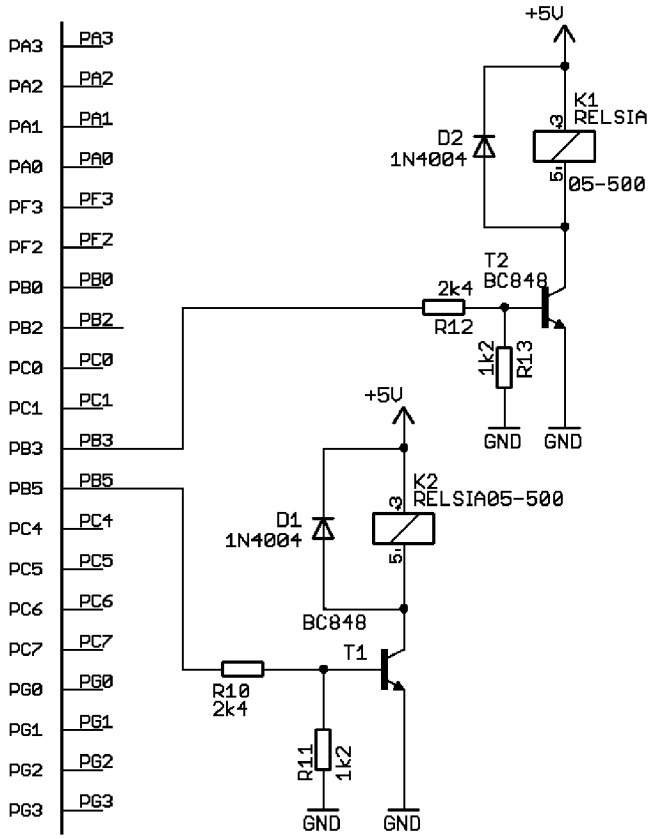
Na tomto místě je třeba podotknout, že na místě A/D převodníku by bylo vhodnější využít obvod s vyšším rozlišením než je 12-bitů převodníku MCP3202. Softwarové vybavení bylo navrhováno s ohledem na to, aby k převodníku bylo možné připojit co nejširší spektrum různých odporových snímačů teploty a právě v případě snímačů, pro něž nebyl měřící obvod přímo koncipován, se nízké rozlišení převodníku projeví, malým teplotním rozlišením. Výrazně výhodnější by bylo použití převodníku s rozlišením 16-bitů.

4.7 Připojení relé pro konfiguraci měřícího můstku

Relé sloužící ke konfiguraci odporového můstku, pro umožnění měření pomocí snímačů s různými hodnotami jmenovitého odporu, jsou ovládána piny mikrokontroléru PB3 a PB5. Pro sepnutí v zapojení použitých relé RELSIA05-500 je potřeba napětí o hodnotě 5 V, což při výrobcem udávané hodnotě elektrického odporu 500Ω znamená proud protékající relé o velikosti 10 mA. [24]

Relé jsou mikrokontrolérem spínána přes tranzistory zapojené ve spínacím režimu. Spínací režim zajistí minimální hodnotu úbytku napětí mezi emitorem a kolektorem, který by měl v případě použitých tranzistorů BC848 dosahovat hodnoty kolem 0.2 V. Aby tranzistor pracoval ve spínacím režimu musí mu být, jak proud báze, tak proud kolektoru

vnucen vnějšími obvody. Proud protékající kolektorem definuje připojená zátěž. Proud báze v zapojení nastavují rezistory R10, R11, R12 a R13, jimi nastavený bázový proud musí být větší, než by v daném pracovním bodě odpovídalo přirozenému zesilovacímu činiteli h_{21E} tranzistoru.[16]



Obrázek 4.12: Zapojení konfiguračních relé

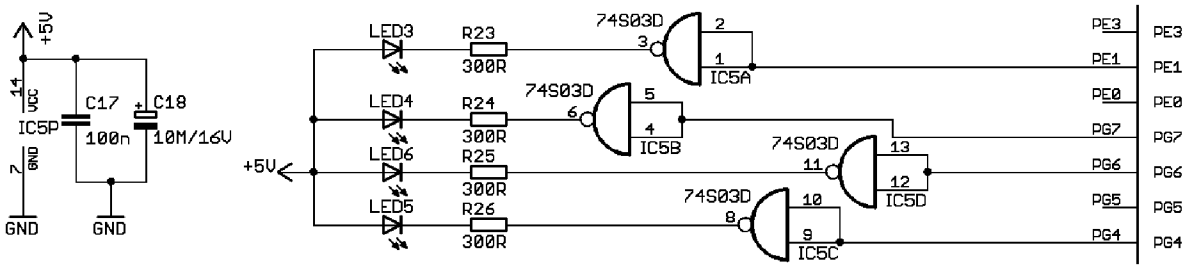
V použitém zapojení jsou báze tranzistorů propojeny přes rezistory k pinům modulu mikrokontroléru a ovládací cívky relé jsou zapojeny jako zátěž mezi napájecí napětí 5 *V* a kolektory tranzistorů. Velikosti odporů nastavujících proud do báze byli zvoleny tak, aby se proud báze pohyboval kolem hodnoty 0.5 *mA*. Tato hodnota je přímo v datasheetu tranzistorů BC848 uvedena jako doporučená pro spínání kolektorového proudu o velikosti 10 *mA*. Paralelně k ovládacím cívkám relé jsou ještě připojeny ochranné diody pro omezení přepěťových špiček při rozpínání relé. Připojení relé k mikrokontroléru zobrazuje obrázek 4.12

4.8 Připojení signalizačních LED diod

V převodníku se nachází šest signalizačních LED diod s požadovanou hodnotou proudu 10*mA*. První dvě diody v zapojení označené LED1(červená) a LED2(zelená) nejsou ovládány mikrokontrolérem. První červená dioda byla již výše zmíněna a měla by sloužit k signalizaci, že k převodníku není připojen žádný snímač. Zelená dioda je zapojena mezi zem a napájecí napětí a jejím účelem je signalizace, zda je k převodníku připojeno napájecí napětí. Zbývající čtyři LED diody jsou již programově ovládány z mikrokontroléru. Jelikož proud protékající těmito diodami, převyšuje maximální povolený zatěžovací proud výstupních pinů mikrokontroléru, bylo využito obvodu 74HC03 obsahujícího čtyři NAND hradla. Výstupy hradel jsou v provedení s otevřeným kolektorem, který má vysokou zatížitelnost z hlediska do něj vstupujícího proudu. Pro ovládání LED diod bylo využito pinů modulu mikrokontroléru označených PG4, PG6, PG7 a PE1, jež nejsou používány žádnou pro tuto aplikaci nezbytnou periferií mikrokontroléru. Tyto piny jsou připojeny na vstupy jednotlivých hradel. Jelikož se jedná o NAND hradla, oba vstupy každého hradla jsou navzájem spojeny. NAND hradlo v takovém případě pracuje jako invertor, tudíž pokud je na jeho propojené vstupy přivedena hodnota log.1, na výstupu hradla bude hodnota log.0 a naopak pokud přivedeme na vstup log.0, výstup bude ve stavu log.1.

Jednotlivé LED diody jsou připojeny anodou k napájecímu napětí 5 *V* a katodou přes rezistor k výstupům obvodu 74HC03. Dioda tedy svítí pokud se výstup hradla nachází ve stavu log.0. Při funkci hradel jako invertoru tedy dojde k rozsvícení diody pokud výstupní pin mikrokontroléru nastavíme na log.1.

Význam LED diod LED3 - LED6 ovládaných mikrokontrolérem je následující. Oranžová LED dioda LED3 se rozsvítí v případě překročení uživatelem nastavených teplotních mezí, LED6 v případě, že se A/D převodník dostane buď do saturace, či hodnota z něj získaná bude nižší než nastavené minimum (v tomto případě hodnota 64). Tímto bude signalizováno překročení měřícího rozsahu převodníku. Zelená LED dioda LED4 signalizuje chod řídícího programu. Pokud při vykonávání řídícího programu dojde k chybě (typicky některá z funkcí ze standardních knihoven vývojového prostředí Dynamic C vrátí chybový příznak) rozsvícení červené LED diody LED5.



Obrázek 4.13: Připojení signalizačních diod k mikrokontroléru

4.8.1 Praktická realizace

Zapojení je realizováno na oboustranné desce plošných spojů o rozměrech 79x105 mm. Většina součástek v zapojení použitých je v provedení SMD, v případě pasivních součástek byla pro snadné pájení, tam kde to bylo možné, zvolena velikost 1206. Z výkonového hlediska nebyli na v zapojení použité rezistory kladeny prakticky žádné nároky a v obvodu tedy postačili rezistory pro jmenovitou hodnotu zatěže 0.25 W. Z hlediska přesnosti byli u pasivních součástek kladeny vyšší nároky pouze na rezistory použité v měřícím můstku, zde byli použity rezistory s přesností 0.1%. V ostatních částech obvodu byli použity součástky s přesností 1% v případě některých kondenzátorů i nižší(10%). Modul RCM3900 mikrokontroléru Rabbit se k desce plošných spojů připojuje pomocí dvou 34-pinových konektorů, do nichž je modul nasunut. Napájení a odporový snímač teploty se k převodníku připojují pomocí tří-kontaktové násuvné svorkovnice AKZ 950/3. Kompletní schéma zapojení a návrh plošného spoje se nachází v přílohách.

5 SOFTWAROVÉ VYBAVENÍ PŘEVODNÍKU

Řídící program převodníku teploty byl vytvořen v prostředí Dynamic C. Program byl strukturován do několika souborů dle jejich konkrétní funkčnosti.

- **main.c** - hlavní soubor aplikace
- **ethernet_com.c** - funkce související s komunikací po Ethernetu
- **Http_server.c** - web server pro zpřístupnění webového rozhraní
- **Modbus.c** - funkce související s komunikací po protokolu Modbus
- **sd_flash.c** - funkce pro zápis na paměťovou kartu, funkčnost dataloggeru
- **SensorData.c** - správa údajů o používaných snímačích
- **smtp_klient.c** - funkce pro odeslání emailu při překročení povolených mezních teplot.
- **SPI_com_with_AD.c** - funkce pro komunikaci s A/D převodníkem

Kromě vlastního vyhodnocení teploty měřené připojeným snímačem spočívá činnost řídícího programu v komunikaci s klientskými zařízeními po rozhraní Ethernet, zajištění běhu Http serveru zpřístupňujícího webové rozhraní a pravidelném zaznamenávání naměřených teplot na paměťovou kartu typu miniSD, tedy realizaci zadáním požadovaného dataloggeru.

5.1 Vkládání souborů v prostředí Dynamic C

Jednou ze změn v jazyku C používaném vývojovým prostředím Dynamic C oproti specifikaci ANSI C, je skutečnost, že knihovní soubory nejsou vkládány pomocí direktivy #include, nýbrž pomocí direktivy #use. Prostředí Dynamic C rovněž nepodporuje hlavičkové soubory s příponou .h a všechny soubory vkládané pomocí direktivy #use jsou považovány za knihovní soubory, navzdory skutečnosti, že mají příponu .c. Tomu musí být uzpůsoben způsob deklarace funkcí v jednotlivých souborech vkládaných do

hlavního souboru aplikace main.c. Jednotlivé funkce ve vkládaných souborech musí být nejprve deklarovány a tato deklarace obalena komentáři */*** *BeginHeader* */ a */*** *EndHeader* */. Až za tímto blokem se může nacházet vlastní definice funkce.

*/*** BeginHeader NazevFunkce */*
navratový_typ NazevFunkce(Parametry);
*/*** EndHeader */*
navratový_typ NazevFunkce(Parametry)
{*Tělo funkce*}

Tímto způsobem jsou zapsány všechny funkce s výjimkou těch obsažených v souboru main.c, do něhož jsou ostatní soubory vkládány jako knihovny [25].

5.2 Hlavní soubor aplikace

Pro řízení chodu aplikace bylo využito operačního systému reálného času μC/OS-II dodávaného spolu s vývojovým prostředím Dynamic C. Program tak byl rozdělen do dvou nezávislých, paralelně běžících úloh, z nichž první úloha, pojmenovaná MeasureTask, zajišťuje samotné měření teploty, zatímco druhá úloha PeriphTask, vykonává operace související s komunikací po síti a funkčností dataloggeru. Jediný sdílený zdroj mezi oběma úlohami, k němuž je potřeba řídit přístup, představuje proměnná sloužící pro sdílení informace o měřené teplotě. Přístup k této proměnné je v aplikaci řízen pomocí semaforu.

Funkce main() se nachází v souboru main.c. Na počátku funkce dojde k provedení inicializace převodníku teploty, v níž jsou nastaveny proměnné a provedeny inicializační operace nezbytné pro činnost převodníku. Dále v této funkci již dojde pouze k vytvoření obou úloh řídící aplikace, vytvoření semaforu pro synchronizaci mezi úlohami a následně ke spuštění vlastního operačního systému.

5.2.1 Průběh inicializace

Při inicializaci převodníku dojde nejprve k volání funkce *OSinit* inicializující operační systém. Následně dochází k volání funkce *Inicialization()* nastavující modul mikro-

kontroléru Rabbit. To zahrnuje nastavení některých registrů mikrokontroléru Rabbit. Kontrétně se jedná o registry PxDR a PxDDR(x zde zastupuje označení portu, pro port C tedy daný registr nese označení PCDR, případně PCDDR). Každý z těchto registrů obsahuje osm bitů, z nichž každý zastupuje jeden ze vstupně/výstupních pinů mikrokontroléru. Registry PxDDR určují, zda daný pin bude sloužit jako vstup, či jako výstup. V případě, že je daný pin nastaven jako výstup, hodnoty v registru PxDR určují, zda bude ve stavu log.1 nebo log.0.

V převodníku je potřeba nakonfigurovat registry příslušející k paralelním portům B, E a G, jimiž jsou ovládány LED diody a tranzistory spínající relé, konfigurující měřící odporový můstek. Následně jsou volány inicializační funkce pro nastavení tcp/ip stacku, http serveru, zápisu na paměťovou kartu nebo načtení konstant nutných pro výpočet teploty. Obsahem těchto inicializačních operací je rovněž načtení hodnot uložených v uživatelském bloku flash paměti, jemuž bude věnována samostatná kapitola.

5.2.2 Úloha MeasureTask

V této úloze je zapouzdřeno periodické měření teploty. Úloha se skládá z nekonečné smyčky realizované pomocí cyklu *while*, jež je spuštěna po provedení nezbytné inicializace několika v této úloze používaných proměnných. Časování je zde zajištěno pomocí funkce *OSTimeDlyHMSM*, která vždy úlohu, na parametrem předanou časovou jednotku, uspí. Dobu po jakou bude úloha uspána lze měnit z webového rozhraní. Tímto způsobem je realizována prodleva mezi jednotlivými měřeními.

5.2.3 Úloha PeriphTask

Nekonečný cyklus *while* obsažený v této úloze se skládá, kromě funkcí zajišťujících chod Tcp/Ip stacku a http serveru, z několika bloků *costate*. Bloky *costate* představují rozšíření standardního programovacího jazyka C o bloky chovající se jako na sobě nezávislé stavové automaty [25]. Pokud by jednotlivé *costate* bloky obsahovaly pouze běžný C kód, postup vykonávání by byl stejný jako v případě běžného sekvenčně napsaného programu, kdy by se při každém průchodu programu postupně vykonali všechny operace zapsané v jednotlivých blocích tak, jak následují za sebou. Odlišnosti se projeví při použití dalšího

rozšíření jazyka Dynamic C, příkazu *waitfor(podminka)*, který umožňuje pozastavení vykonávání operací v daném bloku na dobu dokud není podmínka v jeho parametru vyhodnocena jako log.1. V případě, že podmínka je vyhodnocena jako 0, vykonávání bloku bude pozastaveno a program bude pokračovat vykonáváním operací následujících bezprostředně za daným *costate* blokem. V dalším cyklu programové smyčky bude daný *costate* blok vykonáván až od bodu, kde byl pozastaven, tedy opětovným vyhodnocením *waitfor* příkazu, pokud bude podmínka uvedená v parametru příkazu *waitfor* již vyhodnocena jako log.1, program bude pokračovat příkazy nacházejícími se v bloku bezprostředně za tímto příkazem. V opačném případě bude vykonávání daného bloku opět ukončeno. Ve chvíli, kdy program dosáhne konce bloku, pokračuje ve vykonávání operací nacházejících se za tímto blokem a v dalším cyklu bude opět vykonáván od začátku. V této aplikaci je *waitfor* příkaz použit v kombinaci s funkcemi *DelayMs* a *DelaySec*, jejichž návratová hodnota bude rovna log.1, až uplyne časový interval předaný těmto funkcím jako parametr.

První *costate* blok slouží k periodickému zápisu na připojenou paměťovou kartu. Periodu zápisu může uživatel nastavit přes webové rozhraní.

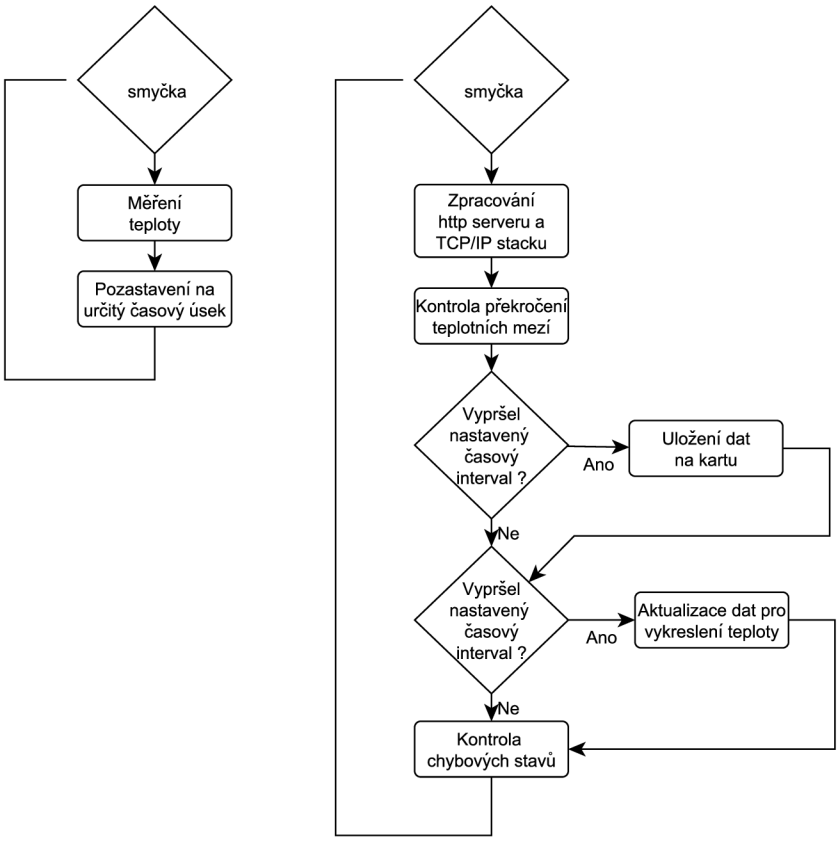
Následující blok periodicky aktualizuje datovou strukturu použitou pro vykreslování průběhu měřené teploty na webových stránkách.

Další z těchto bloků v úloze obsažených pak při každém průchodu smyčkou periodicky kontroluje, zda nedošlo k překročení uživatelem nastavených teplotních mezí.

Poslední *costate* blok obsažený v úloze, slouží k periodické kontrole zda při vykonávání programu nedošlo k chybě a ovládání signalizačních LED diod. Chyby při vykonávání programu mohou zahrnovat nemožnost zápisu do nebo čtení dat z uživatelského bloku nebo provádět operace s paměťovou kartou. Zjednodušený diagram úlohy PeriphTask zobrazuje obrázek 5.1

5.2.4 Uživatelský blok paměti

Pro potřeby uchování dat i po odpojení napájení od převodníku, je na modulu vyhrazeno 8kB paměti v primární flash paměti modulu. K této paměti lze přistupovat pomocí standardních funkcí prostředí Dynamic C *writeUserBlock* a *readUserBlock*. Převodník teploty



Obrázek 5.1: Zobrazení průběhu jednotlivých úloh

tuto paměť využívá pro uložení dat, u nichž je nezbytné, aby byla zachována i po vypnutí a opětovném zapnutí převodníku.

Do trvalé paměti převodník teploty ukládá IP adresu a ostatní parametry sítě modifikované uživatelem z původního nastavení. Dále jsou ukládána data o aktuálně nahraných snímačích, uživatelské jméno a heslo pro přístup k webovým stránkám a rovněž je sem průběžně zaznamenáváno, v určitém časovém intervalu (v programu pevně nastaven na dvě hodiny), číslo stránky paměti miniSD karty, do níž bylo naposledy zapisováno. Tento údaj při opětovné inicializaci, po případném výpadku napájení, výrazně urychlí vyhledávání poslední zapsané stránky, od níž bude zápis navázán. S výjimkou čísla naposledy zapsané stránky SD karty, jsou všechny hodnoty zapisovány vždy po modifikaci,

jejich hodnoty. K načtení hodnot z této tabulky dochází vždy při inicializaci aplikace. Umístění jednotlivých údajů v pamětí zobrazuje tabulka 5.1. Relativní adresa jednotlivých údajů je vztažena k počátku uživatelského bloku ve flash paměti modulu RCM3900.

Hodnota	Adresa	Délka	Datový typ
Počet do převodníku nahraných snímačů	0	2 byty	int
Počet stránek zapsaných do paměťové karty	2	4 byty	long
Data snímačů	10	1310 bytů	pět struktur sensors_data
Ip adresa	1500	4 byty	long
Maska podsítě	1504	4 byty	long
Výchozí brána	1508	4 byty	long
Adresa DNS serveru	1512	4 byty	long
Uživatelské jméno	1550	20 bytů	pole char
Uživatelské heslo	1570	20 bytů	pole char

Tabulka 5.1: Uložení dat v uživatelském bloku

5.2.5 Měření teploty

Perioda mezi jednotlivými měřeními činí při počátečním nastavení 100ms, přičemž tuto hodnotu lze měnit v intervalu od 1 s do 50 ms. Hodnota získaná A/D převodníkem představuje číselné vyjádření hodnoty rozdílového napětí měřícího odporového můstku. Tuto hodnotu napětí je následně nutné přepočítat na teplotu.

5.2.6 Výpočet teploty

Původním záměrem prezentovaným v semestrálním projektu MM2K, bylo získání hodnoty měřené teploty, pomocí předpočítané look-up tabulky obsahující 4096 hodnot, z nichž každá by představovala jednu hodnotu teploty. Tato tabulka by byla realizována jako pole hodnot typu float. Hodnotu získanou z A/D převodníku bychom poté mohli přímo použít jako index pro přístup k prvku pole, v němž by byla uložena příslušná teplota.

Výhodou tohoto přístupu by byla především jeho rychlost, mikrokontrolér by nemusel vykonávat žádné operace s plovoucí řádovou čárkou (nejsou mikrokontrolérem hardwarově podporovány). Nevýhodu tohoto přístupu pak tvoří skutečnost, že klade značné nároky na paměťový prostor. Při velikosti jedné hodnoty typu float čtyři byty a při počtu 4096 prvků pole, by tak tato tabulka pro jeden snímač zabrala 16 kB paměti. V případě, kdy v paměti budou uloženy tabulky pro pět snímačů, by tyto tabulky zabrali 80 kB paměti.

Výpočet měřené teploty nakonec probíhá způsobem, kdy look-up tabulka pro každý snímač obsahuje 64 hodnot představujících jednotlivé body závislosti mezi výstupní hodnotou A/D převodníku a teplotou, které rovnoměrně pokrývají celý měřící rozsah. Tento vztah budeme považovat za po částech lineární. Teplota uložená v prvním prvku tabulky tedy představuje teplotu odpovídající stavu, kdy výstup A/D převodníku nabývá hodnoty 64(prvních 64 hodnot A/D převodníku nebude využito z důvodu nelinearity operačních zesilovačů obvodu mcp602 pro nízká napětí), druhá hodnota tabulky představuje teplotu odpovídající stavu, kdy výstup A/D převodníku nabývá hodnoty 128 atd. Výsledná teplota se vypočte s pomocí konstant uložených v této tabulce a hodnoty A/D převodníku interpolací dle vzorců:

$$i = \frac{hodnota\_AD\_prevodniku}{64} \tag{5.1}$$

kde i představuje index look-up tabulky

$$k = \frac{tabulka\_koeficientu[i+1] - tabulka\_koeficientu[i]}{64} \tag{5.2}$$

kde k představuje směrnici přímky na níž leží výsledná, z výstupní hodnoty A/D převodníku vypočtená, teplota.

$$teplota = (hodnota\_AD\_prevodniku - (i*64))*k + tabulka\_koeficientu[i] \tag{5.3}$$

Pro přesnější odhad hodnoty teploty ještě dojde k číslicové filtraci měřených hodnot. Jelikož předpokládáme, že měřená teplota se bude měnit pomalu, tak pro zpřesnění využijeme plovoucí průměr počítaný z posledních deseti naměřených hodnot.

Koeficienty uložené v look-up tabulce jsou do převodníku nahrány PC aplikací sloužící pro kalibraci. Tyto koeficienty jsou spočítány ze známých hodnot rezistorů zapojených v měřící části obvodu. Nevýhoda uvedeného postupu výpočtu teploty spočívá v

nutnosti vykonávat operace násobení a dělení v plovoucí řádové čárce, jež jsou pro mikrokontrolér časově náročné. Výhodu pak představují nižší nároky na paměť, jelikož tabulka pro jeden snímač zabere pouze 256 bytů.

5.2.7 Funkce pro výpočet teploty

Všechny funkce související s přepočtem výstupní analogové hodnoty A/D převodníku se nacházejí v souboru main.c. Základní funkci pro výpočet teploty představuje

- **float** *CountTemperature*(**int** *AD_val*)

Funkce *CountTemperature* slouží přímo k výpočtu měřené teploty z hodnoty, získané z A/D převodníku, jež je jí předána parametrem. Pro samotný výpočet teploty funkce využívá výše uvedené vzorce.

5.3 Možnost připojení více snímačů

Software pro převodník teploty byl vytvořen s ohledem na to, aby k převodníku bylo možné připojit i jiné snímače než trojici snímačů, která byla při testování k dispozici. Funkce pro správu snímačů, jejichž údaje byli do převodníku nahrány obsahuje soubor SensorData.c. Údaje o každém snímači jsou uloženy ve struktuře *sensors_data*, která má podobu

```
struct {
    char sensor_name[8];
    int configuration;
    float koef_array[64];
}sensor_data;
```

Struktura nesoucí informace o snímači tak obsahuje řetězec osmi bytů, v němž může být obsažen název nebo jiný údaj popisující dotyčný snímač. Hodnota uložena v tomto poli bude zobrazena v PC aplikaci a na webových stránkách jako identifikátor snímače. Položka configuration obsahuje číselnou konstantu od jedné do tří, popisující v jaké, ze tří možných konfigurací, se má nacházet odporový měřící můstek. A pole hodnot typu

float obsahuje křivku závislosti mezi hodnotami A/D převodníku a teplotou, přičemž tyto konstanty jsou použity k výpočtu měřené teploty.

Jelikož v převodníku může být v jeden okamžik nahráno jeden až pět snímačů (tedy jedna až pět uvedených struktur obsahujících informace o snímači), bylo by vhodné, aby těmto strukturám byla přidělována paměť dynamicky, až ve chvíli, kdy bude z PC aplikace do převodníku nahráván nový snímač. Ve výchozí podobě neobsahuje vývojové prostředí Dynamic C, žádný prostředek pro dynamické přidělení paměti (funkce *malloc* není podporována). Jediným způsobem by bylo staticky alokovat určitý úsek paměti a z něj za chodu programu přidělovat paměť jednotlivým strukturám v okamžiku, kdy to bude potřeba. Tento postup však oproti přímé statické alokaci paměti pro jednotlivé struktury neposkytuje žádnou výhodu, pouze by komplikoval program. Proto jsou tyto struktury uloženy ve staticky definovaném poli (paměť je jim přidělena ještě před spuštěním programu). Ve výchozím stavu jsou v převodníku nahrány údaje pro snímače Pt100, Ni1000 a Ni10000, popsané v kapitole věnující se hardwarovému zapojení s tím, že zbývající dvě struktury nejsou v danou chvíli využity. V případě nahrání nového snímače do převodníku program nejprve ověří, zda již není nahrán maximální možný počet pěti snímačů a pokud ne, data jsou nahrána do první volné struktury v poli.

V převodníku tak uživatel nepřepíná přímo konfiguraci měřícího můstku, nýbrž přepíná mezi jednotlivými v převodníku nahranými snímači, přičemž konfigurace můstku je jedním z údajů, který je každému snímači přiřazen.

Soubor SensorsData.c obsahuje funkce

- **int** *SensorArrayInit()*()
- **int** *AddSensor*()
- **int** *DeleteSensor*(**int** *sensor_num*)
- **void** *SwitchConfiguration*(**int** *conf*)

5.3.1 Přidání snímače

K přidání senzoru dojde ve chvíli kdy do uchovávacího registru, datové tabulky Modbus protokolu, 136 bude zapsána nenulová hodnota. V tom okamžiku dojde k zavolání funkce

AddSensor. Údaje o snímači jež budou uloženy, musí být předem nahrány do prvních 133 registrů tabulky uchovávacích registrů. Tato data budou vyplněna do první volné struktury v poli struktur určeném pro ukládání informací o snímačích. Pokud je v převodníku nahrán již plný počet pěti snímačů, funkce pouze vrátí příznak informující o nemožnosti nahrání snímače.

5.3.2 Smazání snímače z převodníku

Pro vymazání snímače slouží funkce *DeleteSensor*, jíž je v parametru předáno číselné označení snímače, který má být vymazán. Předaná hodnota reprezentuje index struktury, jež má být uvolněna, v poli těchto struktur. Funkce je volána ve chvíli kdy do uchovávacích registrů na adresu 138 je zapsána hodnota v rozmezí od nuly do čtyř. Rozsah hodnot je omezen z důvodu, že do registru uložená hodnota zároveň reprezentuje pořadové číslo snímače, jež má být vymazán. Funkce je vytvořena tak, aby v převodníku zůstal alespoň jeden snímač. Žádost o smazání posledního snímače obsaženého v převodníku tedy bude ignorována.

Funkce *SwitchConfiguration* přepíná měřící odporový můstek do konfigurace dané předaným parametrem, platné hodnoty jsou od jedné do tří.

5.4 Získání dat z A/D převodníku

Funkce potřebné pro komunikaci s A/D převodníkem MCP3202 jsou obsaženy v souboru SPI_com_with_AD.c. Ten sestává z následujících základních funkcí nezbytných pro komunikaci s A/D převodníkem.

- **void** *SPI_Init*()
- **int** *SPI_GetADval*(**int** *channel_num*)
- **void** *SPI_csDown*()
- **void** *SPI_csUp*()
- **int** *SwapBytes*(**int** *num*)

Pro vyčtení hodnoty z A/D převodníku slouží funkce *SPI_GetADval*. Při tvorbě této funkce se vycházelo z příkladu dodávaného spolu s vývojovým prostředím Dynamic C, pojmenovaném SPI-A2D-1.c. Tento příklad slouží pro komunikaci s rovněž 12-bitovým A/D převodníkem LTC1294 a pro správnou funkci bylo nutné pouze upravit sekvenci bitů odesílanou A/D převodníku pro zahájení převodu. Parametr num ve funkci *SPI_GetADval* určuje, napětí kterého ze dvou vstupních kanálů A/D převodníku má být převedeno. Funkce *SPI_csDown()* a *SPI_csUp()*, použité ve funkci *SPI_GetADval* slouží k přivedení pinu A/D převodníku na hodnotu log.1 a hodnotu log.0. Pojmenování funkcí může být mírně zavádějící, jelikož funkce *SPI_csDown()* nastaví pin PF1 modulu mikrokontroléru na úroveň log.1, zatímco funkce *SPI_csUp()* na log.0. Toto pojmenování bylo zvoleno z důvodu, že úroveň log.0 na pinu "chipselect"slouží k přivedení převodníku do aktivního stavu, jakmile je připraven zahájit převod napětí a komunikovat.

Frekvenci hodin, jež bude pro komunikaci s A/D převodníkem použita, lze nastavit pomocí definice parametru SPI_CLK_DIVISOR před vložením knihovny SPI.lib. Tento parametr byl v programu nastaven na hodnotu 5. Frekvenci hodinového signálu lze následně spočítat dle vzorce

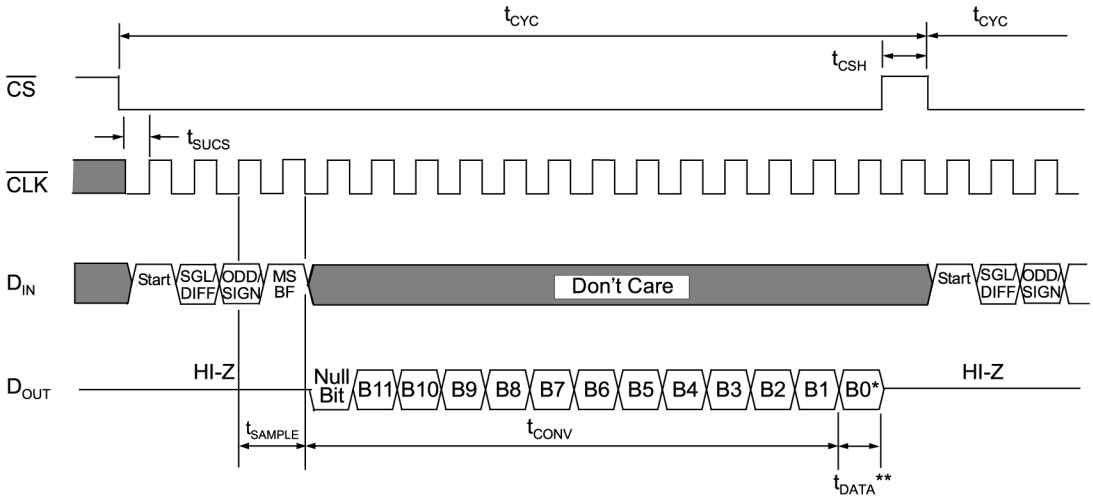
$$f_{CLK} = \frac{Fcpu}{2 * (SPI\_CLK\_DIVISOR + 1)} \tag{5.4}$$

. Při frekvenci procesoru Rabbit 44.2 MHz to odpovídá frekvenci 3.68 MHz.

5.4.1 Komunikace s A/D převodníkem

Komunikace s A/D převodníkem probíhá následujícím způsobem.

Komunikace s A/D převodníkem je zahájena přivedením úrovně log.0 na pin CS. Tím převodník přejde do aktivního stavu a čeká na zaslání převod zahajující sekvence bitů na pin Din. Tato sekvence se skládá ze start bitu a trojice konfiguračních bitů, které definují způsob konverze napětí. První dva konfigurační bity jsou v datasheetu k převodníku označeny jako SGL/DIF a ODD/SIGN. Tyto bity určují, napětí kterého kanálu chceme převést. Těmito bity lze rovněž nastavit variantu, kdy se převádí rozdílové napětí mezi kanály A/D převodníku. Význam nastavení těchto dvou bitů znázorňuje tabulka 5.2. Třetí konfigurační bit nastavuje pořadí, v jakém budou data z převodníku odesílána. Pokud je tento bit nastaven na log.0 datové bity budou odeslány ve formátu MSB, tedy



Obrázek 5.2: Komunikace s A/D převodníkem MPC3202 [21]

	Config. Bits		Channel selection	
	Sql/Diff	Odd/sign	0	1
Single ended mode	1	0	+	-
	1	1	-	+
Pseudo differential mode	0	0	IN+	IN-
	0	1	IN-	IN+

Tabulka 5.2: Konfigurační bity převodníku [21]

nejvýznamnější bit je odeslán první. Pokud tento bit nastavíme na hodnotu log. 1, budou data odeslána nejprve v pořadí MSB a na ně budou neprodleně navazovat data odeslaná v pořadí LSB, tedy nejméně významný bit bude odeslaný jako první.

Na přijetí trojice konfiguračních bitů reaguje A/D převodník nastavením pinu Dout na úroveň log.0, na dobu jednoho taktu hodinového signálu. Po tomto prázdném bitu již následuje dvanáct datových bitů, představujících číslicovou reprezentaci převáděného napětí. K začátku převodu napětí dojde při příchodu druhé náběžné hrany hodinového signálu od příchodu start bitu a převod je ukončen příchodem třetí sestupné hrany hodinového signálu od příchodu start bitu.

5.5 Ukládání dat do paměťové karty

Funkce pro čtení a zápis na paměťovou SD kartu jsou obsaženy v souboru sd_flash.c. Základní funkce obsažené v tomto souboru tvoří :

- **int** *SdFlashInit*()
- **int** *InsertToSdCard*(**float** *data*)
- **int** *ClearSdCard*()
- **int** *GetDataFromSdCard*(**char*** *buff*, **long** *address*, **int** *length_to_read*)

5.5.1 Zápis na paměťovou kartu

Na kartu jsou data ukládána pravidelně, v uživatelem nastaveném intervalu. Data jsou na kartu zaznamenaná ve struktuře obsahující jednak zaznamenávanou teplotu, časovou značku udávající kdy byla zaznamenávaná teplota naměřena a jeden příznakový byte udávající, zda v danou chvíli byl aktivní alarm.

```
struct{
    float temperature;
    tm time;
    char flag;
}SD_dataStruct;
```

Tato struktura na paměťové kartě zabere 12 bytů. Data musí být do paměťové karty zapisována po stránkách velkých 512 bytů. Na jednu tuto stránku se tedy vejde 42 záznamů a zbylých osm bytů stránky slouží jako hlavička, jíž každá zapsaná stránka začíná. Hlavičku představuje textový řetězec OCCUPIED. Tato hlavička je využita při inicializaci převodníku po odpojení napájení, kdy musíme určit, do které stránky bylo naposledy zapisováno, aby bylo možné od této stránky na zápis navázat. Vzhledem k nutnosti zapisovat na kartu po celých stránkách se jednotlivé záznamy ukládají do bufferu, z něhož jsou vždy po uložení 42 záznamů uloženy do karty. Je třeba poznamenat, že k archivaci tohoto bufferu nedochází. Pokud tedy dojde k výpadku napájení, záznamy zapsané do tohoto bufferu před zapsáním na kartu budou ztraceny.

Funkci *InsertToSdCard* je v parametru předána pouze teplota, čas ve kterém byla hodnota změřena se zjistí přímo ve funkci, voláním funkcí *read_rtc* a *mktm*, přičemž první funkce vrací 4-bytovou hodnotu představující počet sekund, které uběhly od data 1.1.1980, druhá vyplní v knihovně RTCLOCK.lib definovanou strukturu tm, jež obsahuje údaj o aktuální vteřině, minutě, hodině, dnu, měsíci a roce.

5.5.2 Vymazání dat z paměťové karty

Vymazání veškerých dat na paměťové kartě zajišťuje funkce *ClearSdCard*, která všechny byty na kartě vyplní hodnotou 0xff. Tuto funkci lze volat pouze z PC aplikace Data_reader, která na adresu 143 v uchovávacích registrech protokolu Modbus zapíše příznak, že data na kartě mají být smazána. V závislosti na množství obsazené paměti může tato operace zabrat delší časový interval.

5.5.3 Vyčítání dat z paměťové karty

Funkce *GetDataFromSCard* slouží k vyčítání hodnot z paměťové karty a jejich následnému zápisu do znakového pole předaného v parametru. Funkce je volána ze souboru modbus.c v případě požadavku na čtení dat z paměťové karty. Která data budou čtena se určí z parametru rovněž předané adresy. Z ní se vypočte číslo stránky a od jaké pozice na této stránce mají být data čtena. Parametr length_to_read určuje počet záznamů, jež mají být přečteny (jeden záznam odpovídá dvěma bytům).

Kromě uvedeného soubor sd_flash.c dále obsahuje funkci *FindLastRecord* volanou v inicializační části aplikace sloužící pro dohledání naposledy zapsané stránky.

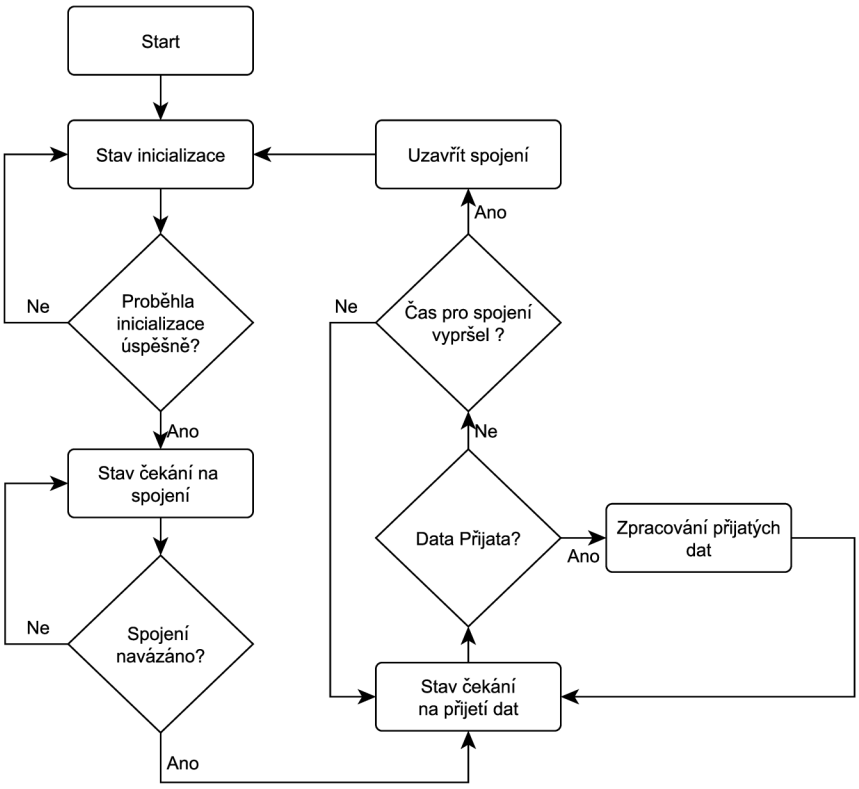
5.6 Komunikace po Ethernetu

Funkce pro komunikaci po ethernetu jsou obsaženy v souboru ethernet.c. V tomto souboru se vkládá s vývojovým prostředím Dynamic C dodávaná knihovna dcrtcp.lib obsahující funkce pro komunikaci po tomto rozhraní.

Základní funkci obsaženou v souboru ethernet.c představuje funkce *ethernet_process()*, volaná při každém cyklu hlavní programové smyčky převodníku. Funkce

zajišťuje navázání spojení s klientskými zařízeními, přijímání převodníku zaslaných zpráv a zajištění chodu Tcp/Ip stacku. Funkce je realizována jako stavový automat, kdy při každém volání jsou vykonány operace příslušné stavu, ve kterém se funkce v danou chvíli nachází. Pokud dojde ke splnění podmínky pro přechod do nového stavu, bude příslušná proměnná udržující informaci o aktuálním stavu funkce nastavena na novou hodnotu.

V prvotním stavu inicializace, v němž se bude funkce *ethernet_process()* nacházet při jejím prvním volání, dochází k volání funkce *tcp_listen*, která nastaví knihovnu dcrtcp.lib tak, aby akceptovala příchozí spojení na portu 502(standardní port protokolu Modbus). Tímto je převodník nakonfigurován jako server, tedy vyčkává na požadavek navázání spojení od klientského zařízení [26]. Po volání funkce *tcp_listen* se nastaví, aby se funkce při příštím volání nacházela ve stavu čekání na příchozí spojení, ve kterém funkce kontroluje zda nedošlo k navázání spojení od klienta. Pokud ano, pak se při příštím volání bude funkce nacházet ve stavu, kdy čeká na klientem zaslaná data. Přijatá data jsou předána funkci DataReceived pro zpracování Modbus zprávy. Každé přijaté spojení ser-



Obrázek 5.3: Znázornění průběhu funkce *ethernet_process*

ver udržuje pouze po dobu 2 sec, od posledního přijetí dat od klienta. Jelikož aplikace umožňuje komunikaci pouze s jedním zařízením, byla tato doba zvolena s ohledem na to, aby nedocházelo k přílišným prodlevám ve chvíli, kdy se k aplikaci pokouší připojit více klientských zařízení a zároveň aby nedocházelo ke zbytečnému zpomalení cyklické komunikace (vyčítání dat z paměťové karty) neustálým obnovováním přerušeného spojení. Po vypršení časového intervalu funkce přejde do stavu čas vypršel, kde dojde k uzavření spojení a jelikož funkci *tcp_listen* je nutné volat při každém přerušení spojení, přejde stavový automat opět do stavu inicializace. Uvedený postup znázorňuje diagram na obrázku 5.3

5.7 Soubor smtp_klient.c

Podpora SMTP klienta je obsažena ve standardní knihovně dodávané s prostředím Dynamic C smtp.lib, jež je do projektu vložena. V souboru stmp_klient.c se nacházejí pouze funkce *SendMailHighAlarm* a *SendMailLowAlarm* volané z hlavní programové smyčky v případě, že dojde k překročení uživatelem nastavené minimální nebo maximální hodnoty teploty. Tyto funkce pomocí funkce *smtp_sendmail* ze standardní knihovny smtp.lib odešlou zprávu o vyvolání alarmu na e-mailovou adresu, zadanou uživatelem přes webové rozhraní. Pauza mezi dvěma po sobě následujícími odesláními je v programu pevně nastavena na půl hodiny. Tato pauza byla zvolena tak, aby nedošlo k zahlcení e-mailové schránky příchozími zprávami.

5.8 Implementace Modbus serveru

Na aplikační vrstvě využíváme v zadání požadovaný protokol Modbus. Jelikož jako síťové rozhraní využíváme Ethernet, byla použita varianta Modbus TCP/IP. V převodníku jsou implementovány datové tabulky, proto vystupuje v rámci komunikace po protokolu Modbus jako server. Je tedy pasivním zařízením, nezahajuje komunikaci, nýbrž pouze reaguje na požadavky klienta. Návrh byl proveden s ohledem na požadavek, aby převodník byl schopen komunikovat s libovolným klientem, který podporuje v převodníku implementované Modbus funkce. Komunikace se odehrává přes pro Modbus vyhrazený port číslo

502.

Z tabulek definovaných datovým modelem protokolu Modbus převodník implementuje obě tabulky 16-bitových registrů. Tabulky diskrétních hodnot nebyli implementovány, protože v dané aplikaci pro ně nebylo nalezeno vhodné využití. Převodník teplotních snímačů podporuje pětici funkcí definovaných specifikací. Tyto funkce jsou vypsány v tabulce 5.3.

Název funkce	Kód funkce
Čtení více uchovávacích registrů	0x03
Čtení více vstupních registrů	0x04
Zápis jednoho uchovávacího registru	0x06
Zápis více uchovávacíh registrů	0x10
Čtení ze souboru	0x14

Tabulka 5.3: Převodníkem implementované funkce

5.8.1 Implementace uchovávacích registrů

V programu jsou uchovávací registry představovány polem hodnot typu integer (integer má v mikrokontroléru Rabbit velikost 16-bitů, tím tedy přesně odpovídá velikosti jednoho uchovávacího registru) o velikosti stopadesáti registrů pojmenovaným holding_registers.

Význam prvních 133 registrů (číslovaných 0 až 132) odpovídá struktuře, v níž jsou uchovávány informace o snímačích. Těchto prvních 133 registrů je využíváno pro přenos informací o v převodníku nahraných snímačích do vytvořené PC aplikace a rovněž pro nahrávání nových snímačů do převodníku. V prvních čtyřech registrech se tedy nachází 8-bytový identifikátor snímače. Pátý registr je vyhrazen pro číslo konfigurace odporového můstku. Následujících 128 registrů představuje konstanty používané pro výpočet teploty, které jsou uloženy ve formátu čísla s plovoucí řádovou čárkou float standardu IEEE754. Tyto hodnoty jsou 32-bitové a každá z nich tedy zabírá dva registry.

Následující dva registry jsou volné. Do registru číslo 135 se ukládá hodnota udávající, který z v převodníku nahraných snímačů je právě používán. Zápisem do tohoto registru lze převodník nakonfigurovat pro používání jiného z nahraných snímačů. Program pak přepne

měřící můstek do požadované konfigurace a pro výpočet teploty budou použity konstanty požadovaného snímače. Platné hodnoty tohoto registru se pohybují v rozmezí nula až čtyři a jedná se o indexy pole, v němž jsou struktury obsahující údaje pro jednotlivé snímače uloženy.

Následující registry slouží k vyvolání určité funkce převodníku, pokud je do nich zapsána odpovídající hodnota. Běžně tyto registry obsahují hodnotu -1. Pokud dojde k zápisu, program vyhodnotí, zda zapsaná hodnota leží v pro daný registr povoleném rozsahu a pokud ano, zavolá odpovídající funkci. V případě, že zapsaná hodnota neleží v povoleném intervalu, k vyvolání příslušné operace nedojde. V obou případech bude po vyhodnocení do registru opět zapsána výchozí hodnota -1. Zápis do registru číslo 136 bude vyhodnocen jako příkaz k nahrání nového snímače do převodníku, přičemž pro přidání budou využita data, jež jsou obsažena v prvních 133 uchovávacích registrech. Registr 137 je použit pro změnu konstant používaných pro výpočet teploty. Hodnota zapsaná do tohoto registru určuje konstanty, kterého snímače mají být změněny. Rozdíl oproti předcházející funkci spočívá v tom, že z prvních 133 uchovávacích registrů budou použity pouze registry obsahující tyto konstanty. Hodnotami v těchto registrech zapsaných budou přepsány konstanty požadovaného snímače. Tato funkce je využita při kalibrování snímače. Zápisem do registru číslo 138 smažeme z paměti převodníku snímač. Který snímač má být vymazán určuje, podobně jako v předchozím případě, hodnota do tohoto registru zapsaná. Zapsáním čísla do registru 139 budou do prvních 133 uchovávacích registrů nahrány informace o snímači. Toho se využívá v případě, že chceme z převodníku vyčíst informace o uložených snímačích. Zápis libovolné hodnoty (jiné než přednastavená -1) do registru číslo 140 způsobí, že obvod reálných hodin převodníku bude nastaven na hodnotu uloženou v registrech 141 a 142. Hodnota v těchto dvou registrech uložená je typu long a představuje počet sekund, jež uběhl od data 1.1.1980. Zápis do registru číslo 140 musí následovat bezprostředně po zapsání hodnoty do těchto registrů. Nastavení hodin reálného času je nezbytné vždy po zapnutí převodníku. Dokud nedojde k nastavení přesného času, převodník nebude ukládat hodnoty do paměťové karty. Zápis do registru číslo 143 vyvolá funkci, jež vymaže veškerá zapsaná data z paměťové karty převodníku.

Číslo registru	Význam	Rozsah hodnot
0 až 3	Označení snímače	cokoliv
4	konfigurace odporového můstku	1-3
5 - 133	Hodnoty teplotních koeficientů	cokoliv
135	číslo používaného snímače	0-4
136	Přidání nového snímače	cokoliv
137	Změna výpočetních konstant konkrétního snímače	0-4
138	Smazání konkrétního snímače z paměti převodníku	0-4
139	Aktualizace údajů o snímači v uchovávacích registrech	cokoliv
140	Nastavení hodin reálného času	cokoliv
141-142	Aktuální čas	cokoliv
143	Vymazání paměťové karty	cokoliv

Tabulka 5.4: Uchovávací registry

5.8.2 Implementace vstupních registrů

Vstupní registry jsou v řídícím programu mikrokonroléru Rabbit implementovány jako pole deseti hodnot typu integer.

Číslo registru	Význam
0-1	aktuální měřená teplota
2	hodnota výstupu A/D převodníku
3	počet snímačů aktuálně nahraných v převodníku
4-5	počet stránek zapsaných do paměťové karty

Tabulka 5.5: Vstupní registry

Registry 0 a 1 obsahují údaj o teplotě ve stupních Celsia, uložený jako typ float. Tento údaj se aktualizuje při každém měření teploty. Registr číslo 2 obsahuje hodnotu, kterou mikrokontrolér získává z A/D převodníku, ta se stejně jako informace o teplotě aktualizuje při každém měření převodníku. V registru číslo 3 se nachází informace o počtu snímačů, jež jsou v převodníku v danou chvíli nahrány. Registry 4 a 5 obsahují počet

stránek aktuálně zapsaných do paměťové karty připojené k převodníku. Tento údaj je využíván pro vyčtení všech hodnot uložených na kartu PC aplikací.

5.8.3 Chybové kódy

V případě, že vykonání požadované operace z nějakého důvodu selže, zařízení odešle chybovou zprávu s jedním z chybových kódů uvedených v 5.6.

Číslo	**Význam**
0x01	Označuje, že dané zařízení (server) nepodporuje požadovanou operaci
0x02	Adresa, případně rozsah adres, z nichž chceme data číst nebo na ně zapisovat je neplatná
0x03	Neplatná hodnota dat u operací, kdy pracujeme s určitým rozsahem adres, tento kód může znamenat, že se snažíme zapisovat nebo číst větší množství registrů než je udáno specifikací.
0x04	Vykonání operace selhalo

Tabulka 5.6: Implementované chybové kódy a jejich význam [8]

5.8.4 Programová implementace

Modbus server je realizován v souboru modbus.c, v němž se nachází všechny funkce a definice potřebné pro jeho funkčnost.

V inicializační části programu je z tohoto souboru volána funkce *InitHoldingRegisters()*, v níž dojde k prvotnímu nahrání údajů do pole holding_registers(důležité je zajistit, aby funkce *InitHoldingRegisters()* byla volána až po funkci *SensorArrayInit()* ze souboru SensorData.c). Základní funkce je pojmenována **void** *ModbusDataReceived*() a je volána ze souboru http_server.c vždy, když jsou po Ethernetu přijata data. Ve funkci se nejprve určí, zda přijatá data jsou zprávou protokolu Modbus, čili zda prvek MBAP hlavičky nazvaný identifikátor protokolu nabývá hodnoty 0x0000. Dále se určí, zda zpráva byla přijata celá, to lze určit z prvku MBAP hlavičky nazvaném délka, který obsahuje počet bytů zprávy v PDU části zprávy + 1 byte pro identifikátor zařízení. Pokud počet bytů celé zprávy nesouhlasí s počtem bytů, jež byli přijaty, dosud přijatá část zprávy se uchová a

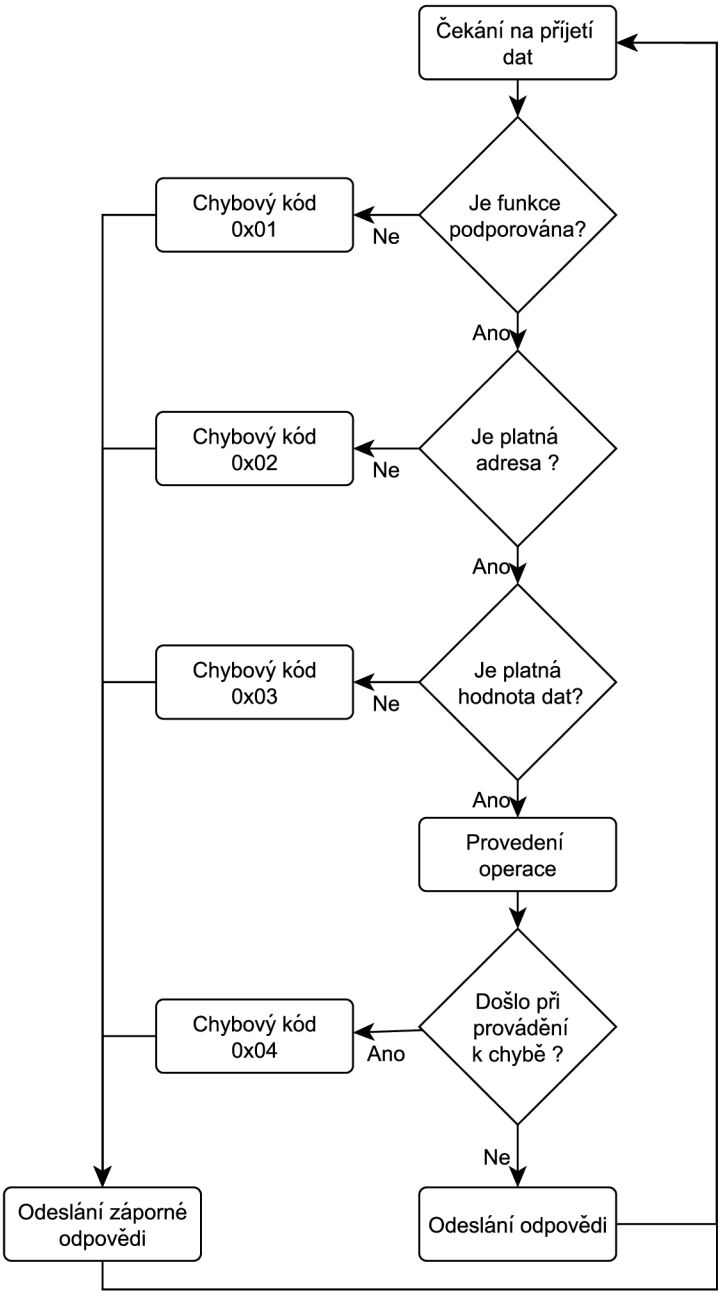
funkce je ukončena a znovu volána až v době, kdy jsou přijata další data. Po přijetí celé Modbus zprávy funkce otestuje kód funkce uvedený ve zprávě, zda je požadovaná funkce podporována. Pokud podporován není, převodník odešle zpátky zápornou odpověď s chybovým kódem 0x01 pomocí funkce *SendExceptionResponse*, pokud podporována je, z přijatého řetězce dat jsou extrahovány potřebné parametry (adresy kam zapisovat, či číst apod.) a zavolá se příslušná funkce vykonávající požadovanou operaci.

Postup zpracování přijaté Modbus zprávy byl implementován dle diagramu 5.4 uvedeném ve specifikaci. Kontrola toho, zda je daná funkce zařízením podporována, se odehrává ve funkci *void ModbusDataReceived()*. Úlohy v diagramu následující po ověření kódu funkce spočívající v ověření platnosti adresy apod., jsou již implementovány ve funkcích zastupujících konkrétní modbus funkce, jelikož jejich podmínky se mezi sebou mohou vzájemně lišit. Implementované Modbus funkce jsou v převodníku zastoupeny funkcemi:

void *ReadMultipleRegisters*(**int** *address_begin,* **int** *address_end*) - funkce realizuje čtení z více uchovávacích registrů. Parametry funkce specifikují rozsah adres, z nichž mají být data čtena. Maximální počet registrů, jež lze přečíst najednou činí dle specifikace 125.

void *ReadMultipleInputRegisters*(**int** *address_begin*, **int** *address_end*) - funkce realizuje čtení z více vstupních registrů. Význam parametrů a omezení týkající se množství registrů, jež můžeme najednou přečíst, je stejné jako u předchozí funkce.

void *ReadFileRecord*(**int** *file_num*, **int** *record_num*, **int** *record_length*, **int** *byte_count*)- funkce realizuje čtení ze souboru. V převodníku je využita pro vyčítání dat z paměťové karty. Pareametr file_num představuje dle specifikace identifikátor souboru, z nějž chceme data číst a parametr record_num představuje adresu záznamu v souboru. Záznam, dle definice, v podstatě odpovídá vstupnímu nebo uchovávacímu registru, tedy má velikost 16-bitů a je jednoznačně identifikovatelný pomocí čísla. Parametr record_length představuje počet záznamů, jež mají být přečteny. Tento počet musí být takový, aby v odpovědi nebyla překročena maximální povolená délka PDU části zprávy.



Obrázek 5.4: Diagram zpracování modbus požadavku [7]

void *WriteMultipleRegisters*(**int** *address_begin*, **int** *length_to_write*, **char** *byte_count*) - funkce realizuje zápis více uchovávacích registrů. Význam prvních dvou parametrů je stejný jako u obou výše zmíněných funkcí. Poslední parametr představuje počet bytů, jež mají být zapsány(jelikož máme 16-bitové registry, toto číslo musí být vždy sudé). Maximálně počet registrů, jež můžeme najednou zapsat činí dle specifikace 122.

void *WriteSingleRegister*(**int** *address*, **int** *value*) - Zápis jednoho uchovávacího registru.

Na počátku každé z těchto čtyř funkcí dojde nejprve k ověření platnosti adres, nad nimiž je provedení dané operace požadováno. Následně se z rozsahu adres vypočte počet registrů, které mají být v operaci zahrnuty(v případě funkce zápisu do jednoho registru, vyhodnocení této podmínky odpadá). Tento počet registrů je u všech Modbus funkcí dán specifikací a jednotlivé hodnoty se mohou vzájemně lišit. Pokud některá z těchto podmínek není splněna, server odešle zápornou odpověď, v opačném případě následuje vykonání požadované operace.

MBAP hlavička zůstává u odpovědi stejná jako u přijatého požadavku, s výjimkou prvku délky, jež je třeba vyplnit podle skutečného počtu bytů odpovědi. Následně jsou do bufferu vyplněny datové byty zprávy a zpráva je odeslána pomocí standardní funkce *Ethernet_write*.

5.8.5 Čtení dat z paměťové SD karty pomocí protokolu Modbus

Pro vyčítání dat z paměťové SD karty není možné použít žádnou z funkcí pro čtení vstupních nebo uchovávacích registrů. Teoreticky by bylo možné využít vstupních registrů, do nichž klientům není umožněno zapisovat a slouží pouze pro čtení s tím, že odpovídající datová tabulka by se namapovala na data uložená v paměťové kartě. Standardní funkce pro práci se vstupními registry umožňují však adresovat pouze 65536 16-bitových hodnot. Tím by jednak zůstala nevyužita většina kapacity použité paměťové karty a navíc při délce jednoho záznamu v paměťové kartě rovného 12 bytům a při periodě zápisu 1 s, by na kartu bylo možné zapisovat pouze po dobu tří hodin, než by došlo k překročení adresovatelné kapacity. Proto byla pro vyčítání z karty zvolena Modbus funkce Read File Record. Tato funkce je určena pro čtení souborů a není vázána na žádnou z ve specifikaci

definovaných datových tabulek. Data, která chceme číst, jsou v této funkci určena pomocí dvou parametrů, čísla souboru a čísla záznamu v tomto souboru. Každý soubor může dle specifikace obsahovat až 10000 záznamů, číslovaných 0 až 9999. Přičemž funkce umožňuje čtení z až 65536 souborů. Při délce záznamů 16-bitů, tak pomocí této funkce můžeme přistupovat k více než jednomu gigabytu dat.

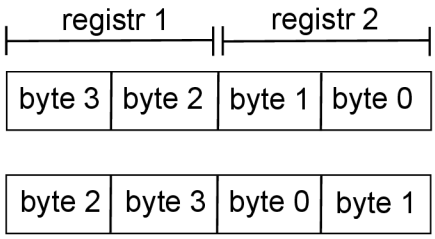
Čtení z karty by bylo možné provádět tím způsobem, že číslo souboru by představovalo číslo stránky paměťové karty, z níž chceme data číst a číslo záznamu by pak určovalo adresu dat v této stránce. Tímto způsobem by bylo možné přistupovat k 32 MB paměti na kartě s tím, že doba, po kterou by bylo možné data do paměti zapisovat, než by došlo k překročení této velikosti, by činila 31 dní (perioda zápisu 1 s).

Pro maximální možné využití kapacity připojené paměťové karty bylo zvoleno adresování, kdy součin obou položek (čísla souboru a adresy záznamu) tvoří sekvenční adresu dat v paměťové kartě. Mapování této adresy na adresu v paměťové kartě probíhá tím způsobem, že počet bytů jedné stránky paměťové karty dělíme dvěmi, abychom velikost stránky přizpůsobily 16-bitovým záznamům. Od této hodnoty následně odečteme velikost hlavičky každé stránky, jež neobsahuje žádná užitečná data a tudíž ji není nutné adresovat. Výsledek použijeme pro dělení, protokolem Modbus, zaslané adresy (součin čísla stránky a čísla záznamu). Výsledek dělení představuje číslo stránky, jíž chceme číst. Pokud provedeme stejnou operaci s tím rozdílem, že operaci dělení nahradíme operací modulo, získáme pozici, od které má být z této stránky čteno. Pokud například chceme číst třetí stránku paměťové karty od bytu 20, parametry funkce Read file record budou mít následující hodnoty, adresa souboru = 1, číslo záznamu = 510 (hodnota 510 byla vypočtena jako 2*((512-8)/2) + (20-8)/2).

5.8.6 Formát přenášených dat

Dle specifikace jsou data po protokolu Modbus přenášena ve formátu Big-endian. To znamená, že u všech datových položek delších než jeden byte, je jako první po rozhraní Ethernet přenášen nejvyšší byte (MSB) a jako poslední byte nejnižší(LSB). To se tyká všech 16-bitových položek přenášených mezi převodníkem a klienty. 32-bitové položky reprezentující teplotu, které zabírají vždy po dvou registrech, jsou odesílány po jednotlivých registrech, tím je myšleno, že není odeslán napřed čtvrtý byte a jako poslední byte

první, ale data jsou odeslána ve formátu viz 5.5. Horní část obrázku zobrazuje jednot-



Obrázek 5.5: Uspořádání odesílané hodnoty typu float

livé byty hodnot typu float, jak jsou uloženy v paměti, dolní část obrázku pak zobrazuje hodnoty typu float, poté co dojde ke změně pořadí bytů.

5.9 Soubor Http_server.c

Soubor obsahuje inicializační funkci *Http_init* a funkci *http_process* periodicky volanou z hlavní smyčky aplikace. Dále tento soubor obsahuje definice nezbytné pro funkci webového rozhraní související s importem souborů obsahujících html zdrojový kód webových stránek, styly a zajištění interakce mezi webovým rozhraním a programem mikrokontroléru. Podrobněji bude implementace webového rozhraní popsána v následujících kapitolách.

5.10 Webové rozhraní

Webové rozhraní převodníku teploty sestává ze čtyř webových stránek sloužících k vizualizaci měřených dat a základní konfiguraci převodníku. Informace nezbytné pro správnou implementaci tohoto rozhraní byli čerpány z [27]. Webové stránky celkově sestávají z následujících souborů.

- main.html - hlavní stránka pro vizualizaci naměřených údajů
- net_param.html - nastavení parametrů síťové komunikace
- prev_param.html - nastavení obecných parametrů převodníku teploty

- password.html - nastavení přihlašovacích údajů pro přístup k webovým stránkám
- style.css - styly pro správné vykreslení stránek
- jquery.flot.js - javascriptová knihovna pro vykreslení grafu
- jquery.min.js - javascriptová knihovna jquery

Při vývoji samotných webových stránek byli čerpány informace z [30]

5.10.1 Vložení webových stránek do mikrokontroléru

Http server lze do aplikace zahrnout vložením knihovny http.lib. Pro vlastní implementaci webových stránek do mikrokontroléru Rabbit je nejprve nutné tyto ve zdrojovém kódu naimportovat pomocí direktivy
#ximport cesta_k_souboru symbolický_název Tato direktiva zkopíruje do paměti mikrokontroléru soubor specifikovaný prvkem cesta_k_souboru a zkopírovaným datům přiřadí symbolický název, pomocí něhož se na tyto soubory lze později odkazovat.

Následně je potřeba vyplnit MIME tabulku, jež registruje jednotlivé typy použitých souborů a zajišťuje jejich pozdější správnou interpretaci. MIME tabulka je uvozena makrem *SSPEC_MIMETABLE_START* a zakončena makrem *SSPEC_MIMETABLE_END*. Mezi tato dvě návěští se vkládají jednotlivé řádky MIME tabulky makrem *SSPEC_MIME*, čímž zaregistrujeme jednotlivé přípony používaných souborů a zajistíme jejich správnou interpretaci. Příponu .html jako jedinou importujeme pomocí makra *SSPEC_MIME_FUNC*. Makro definuje, že pokud importovaný soubor má příponu .html, daný soubor může obsahovat zhtml skript a je nutné zavolat interní knihovní funkci *zhtml_handler*, která dané skripty vykoná a jejich výsledek převede na běžný html kód, jemuž rozumí prohlížeč [27]. V případě této aplikace musíme zaregistrovat přípony .html, .css a .js, soubory těchto typů budou ve webovém rozhraní použity.

Závěrem je nutné vytvořit tabulku statických zdrojů, která umožní Http serveru přístup k souborům, jež byli naimportovány pomocí direktivy #ximport. Tabulka zdrojů se uvozuje pomocí makra *SSPEC_RESOURCETABLE_START* a ukončuje makrem *SSPEC_RESOURCETABLE_END*. Jednotlivé zdroje jsou zapisovány pomocí *SSPEC_RESOURCE_XMEMFILE*. Implementace webových stránek do mikrokontroléru

Rabbit má v tomto případě následující podobu.

```
#ximport "Lib/tcpip/main.html" web_main
#ximport "Lib/tcpip/net_param.html" web_second
#ximport "Lib/tcpip/prev_param.html" web_third
#ximport "Lib/tcpip/password.html" web_fourth
#ximport "Lib/tcpip/style.css" style
#ximport "Lib/tcpip/jquery.min.js" jquery
#ximport "Lib/tcpip/jquery.flot.js" flot

SSPEC_MIMETABLE_START
    SSPEC_MIME_FUNC(".html","text/html",zhtml_handler),
   SSPEC_MIME(".js","text/javascript"),
    SSPEC_MIME(".css","text/css")
SSPEC_MIMETABLE_END

SSPEC_RESOURCETABLE_START
    SSPEC_RESOURCE_XMEMFILE("/index.html",web_main),
    SSPEC_RESOURCE_XMEMFILE("/podsl/net.html",web_second),
    SSPEC_RESOURCE_XMEMFILE("/podsl/prev.html",web_third),
    SSPEC_RESOURCE_XMEMFILE("/podsl/pass.html",web_fourth),
    SSPEC_RESOURCE_XMEMFILE("/jquery.js",jquery),
    SSPEC_RESOURCE_XMEMFILE("/flot.js",flot),
    SSPEC_RESOURCE_XMEMFILE("/style.css",style)
SSPEC_RESOURCETABLE_END
```

5.10.2 Skriptovací jazyk zhtml

Skriptovací jazyk zhtml slouží k dynamickému vytváření webových stránek, jejichž výsledná podoba tak může být odvozena od aktuálních hodnot interních proměnných obsažených v mikrokontroléru. Převodník teploty tento skriptovací jazyk používá

například k zobrazení výstrahy (alarmu), že teplota překročila nastavené meze nebo k naplnění grafu změřenými teplotami.

Jednotlivé příkazy jsou uzavírány dvojicí znaků *<?z příkaz ?>*. Skriptovací jazyk podporuje podmínkové příkazy typu *if* i smyčky typu *for*. Syntaxe obou těchto příkazů je prakticky totožná s jazykem C.

5.10.3 Zadávání hodnot do webového rozhraní

Aby bylo možné hodnotu proměnné obsažené v mikrokontroléru prezentovat nebo změnit pomocí webového rozhraní, musí tato proměnná být zaregistrována pomocí direktivy *#web*. Pro zobrazení hodnoty proměnné na webové stránce je v html kódu použito zhtml skriptu *<?z echo($nazev_promenne) ?>*, jenž funkci zhtml_handler říká, aby tento text v html kódu nahradila hodnotou interní proměnné. Název samotné proměnné je vždy nezbytné uvozovat znakem $.

Změnu hodnot proměnných mikrokontroléru uživatelem skrze webové rozhraní umožňují formuláře jazyka Html. Pro předání dat http serveru je použito metody POST. Parametr formuláře ACTION obsahující informaci o tom, jaký skript se má na straně serveru pro zpracování předaných dat použít, stačí nastavit na název konkrétního Html souboru, v němž je formulář obsažen. Http server běžící v mikrokontroléru se pak postará o správné zpracování dat. Pro zádání hodnot jsou použity především formulářové prvky typu input mající podobu prostého textového pole pro zadání dat. Aby na straně mikrokontroléru bylo rozlišeno, který prvek formuláře přísluší ke které proměnné, musí parametr NAME daného formulářového prvku obsahovat hodnotu dotyčné proměnné. Definice jednoho formulářového prvku pak vypadá takto

<input TYPE="text"NAME="g_alarmHigh"VALUE="<?z print($g_alarmHigh) ?>">

Text zapsaný v hodnotě VALUE zajistí, že po načtení stránky bude v daném formulářovém prvku zobrazena aktuální hodnota proměnné.

5.10.4 Stránka main.html

První stránka webového rozhraní slouží pouze k zobrazení naměřených hodnot. Stránka obsahuje pravidelně aktualizovanou informaci o aktuální teplotě. Zobrazení maximální a minimální teploty naměřené převodníkem. Zároveň se zde zobrazují alarmy, pokud dojde k překročení uživatelem nastavených mezních teplot. Na této stránce se rovněž zobrazují interní chyby, ke kterým při vykonávání programu může dojít (selhání zápisu do uživatelského bloku, selhání operace při práci s paměťovou kartou). Přístup k této stránce není na rozdíl od ostatních stránek nijak omezen, naměřená data tak mohou zobrazit všichni uživatelé, jež se k webovému rozhraní převodníku připojí.

Pro vykreslení grafu zobrazujícího naměřené teploty v čase byla použita volně dostupná knihovna flot, již lze získat ze zdroje [28]. Jedná se o v javascriptu napsanou knihovnu, rozšiřující knihovnu jquery o možnost vykreslování grafů. Z množství souborů, které jsou spolu s touto knihovnou dodávány, bylo v projetku nutné využít pouze soubory jquery.min.js a jquery.flot.js. Graf vykresluje průběh teplot z posledních 24 hodnot uložených do struktury nazvané graph_values. Časový interval mezi jednotlivými hodnotami lze nastavit pomocí webového rozhraní, přičemž nejnižší povolená hodnota tohoto intervalu činí jednu sekundu, zatímco nejvyšší povolená hodnota je jedna hodina. V grafu tedy může být vykreslen průběh teplot v posledních 24 sekundách až 24 hodinách. Struktura graph_values obsahuje kromě pole obsahujícího hodnoty teploty, rovněž pole, v němž jsou uloženy časové značky udávající, kdy byli jednotlivé teploty naměřeny. Časové značky jsou uloženy v datovém typu long a mají podobu počtu sekund, jež uběhli od data 1.1.1980.

Jelikož javascript používá časové značky odvozené od 1.1.1970 musíme před vykreslením grafu k časové značce uložené v převodníku připočíst hodnotu 315525600, reprezentující počet sekund uběhnuvších během chybějících deseti let. Tuto hodnotu musíme následně vynásobit tisíci, jelikož je rovněž potřeba, aby grafu předaná hodnota byla v milisekundách. Uvedený výpočet se nachází ve skriptu zapsaném ve webové stránce main.html, tento výpočet tedy nijak nezatěžuje převodník teploty, jelikož ke zpracování javascriptu dochází na straně klienta, tedy na straně webového prohlížeče.

Struktura graph_values obsahuje stejné hodnoty jako jsou ty uložené na miniSD kartě připojené k převodníku. Vzhledem k nutnosti načítání celých stránek, by však získání

těchto dat z karty bylo značně časově náročné, proto byla zvolena možnost vytvoření struktury, jejíž data budou průběžně aktualizována. Webová stránka pro vizualizaci dat převodníku je na obrázku 5.6

Obrázek 5.6: Webová stránka main.html pro vizualizaci dat

5.10.5 Stránka prev_param.html

Stránka prev_param.html obsahuje základní možnosti konfigurace převodníku teploty. Na této stránce lze nastavit teplotní meze při jejichž překročení dojde k vyvolání alarmu. Rovněž lze nastavit emailovou adresu na niž bude zasláno upozornění v případě, že k překročení nastavených mezí došlo. Dále stránka obsahuje nastavení periody mezi jednotlivými měřeními teploty, jak často má docházet k zápisu na paměťovou kartu a za jak dlouhý časový úsek mají být vykresleny naměřené hodnoty v grafu na titulní stránce. Kromě uvedeného, pak na této stránce, lze již pouze změnit konfiguraci převodníku z hlediska používaného snímače (přepnutí je realizováno zapsáním čísla snímače do k tomu určeného pole a potvrzení formuláře. Seznam v převodníku nahraných snímačů s jim příslušejícími pořadovými čísly, je uveden pod tímto polem).

Parametr	Hodnota
IP adresa	192.168.0.2
Maska podsítě	255.255.255.0
Výchozí brána	192.168.0.1

Tabulka 5.7: Počáteční konfigurace síťových parametrů

5.10.6 Stránky net_param.html a password.html

Zbývající dvě stránky obsahují nastavení parametrů sítě a přihlašovacích údajů pro přístup k webovým stránkám souvisejícím s konfigurací. Na stránce pojmenované net_param lze nastavit Ip adresu převodníku, masku podsítě, výchozí bránu a adresu DNS serveru.

Stránka password.html pak umožňuje nastavení přihlašovacího jména a hesla, jimiž se musí uživatel prokázat, aby mu byl umožněn přístup ke stránkám souvisejícím s konfigurací převodníku.

5.10.7 Autentizace uživatele

Pro přístup k webovým stránkám souvisejícím s konfigurací převodníku se uživatel musí prokázat svým uživatelským jménem a znalostí hesla (ve výchozím stavu jsou heslo i jméno admin). V současném stavu jsou uživatelské jméno i heslo posílány v nezašifrované podobě. Knihovny prostředí Dynamic C nabízí možnost využití tzv. ”digest”autentizace, kdy po sítí není přenášeno samotné heslo a jméno, nýbrž jejich hash vypočtený pomocí algoritmu MD5. Tuto funkci se však do současné doby nepodařilo zprovoznit.

5.11 Nastavení parametrů připojení

Počáteční konfiguraci převodníku, která je nastavena po inicializaci, uvádí tabulka 5.7. IP adresa ani další parametry nejsou v převodníku nastaveny pevně, ale lze je měnit skrze implementované webové rozhraní.

Konfigurace výchozí brány by nebyla nutná, pokud by komunikace převodníku s klienty a přístup k webovým stránkám byl vyžadován pouze v rámci vnitřní sítě. Samotný převodník přístup k internetu vyžaduje pouze k odeslání emailu se zprávou o překročení

alarmu. Původním záměrem bylo, aby hodiny reálného času převodníku byli synchronizovány podle jednoho z na internetu dostupných serverů poskytujících informaci o aktuálním čase. Tato funkce však z časových důvodů již nebyla implementována a čas je tak nutné synchronizovat pomocí PC aplikace pro kalibraci.

Je vhodné zmínit, že v rámci ověřování funkčnosti jako klient vystupovalo PC s programovým vybavením popisovaným v kapitole ”Programové vybavení PC”a veškerá komunikace probíhala pouze mezi těmito dvěma zařízeními. V síti s více komunikujícími zařízeními, která se nacházejí v jedné kolizní doméně, nebyl převodník testován. V současné době je však zcela běžné, že zařízení jsou do sítě připojena přes switch, v daném segmentu tudíž nedochází ke kolizím a chování převodníku by se tedy nemělo lišit od chování, jež převodník vykazoval při testování.

6 SOFTWAROVÉ VYBAVENÍ PC

Jako programové vybavení pro PC byli naprogramovány dvě aplikace. Temperature converter pro konfiguraci parametrů převodníku a Data_reader pro vyčítání naměřených teplot z paměťové karty převodníku a jejich následné uložení do CSV souboru. Obě PC aplikace pro konfiguraci převodníku byla napsána pomocí multiplatformní knihovny pro vývoj programů s grafickým uživatelským rozhraním Qt. Konkrétně bylo využito verze 4.6.2. Knihovna využívá programovacího jazyku C++ a její nekomerční varianta je zdarma šířena pod GPL licencí. Jako vývojového prostředí bylo využito programu Visual studio 2008.

6.1 Aplikace pro kalibraci převodníku

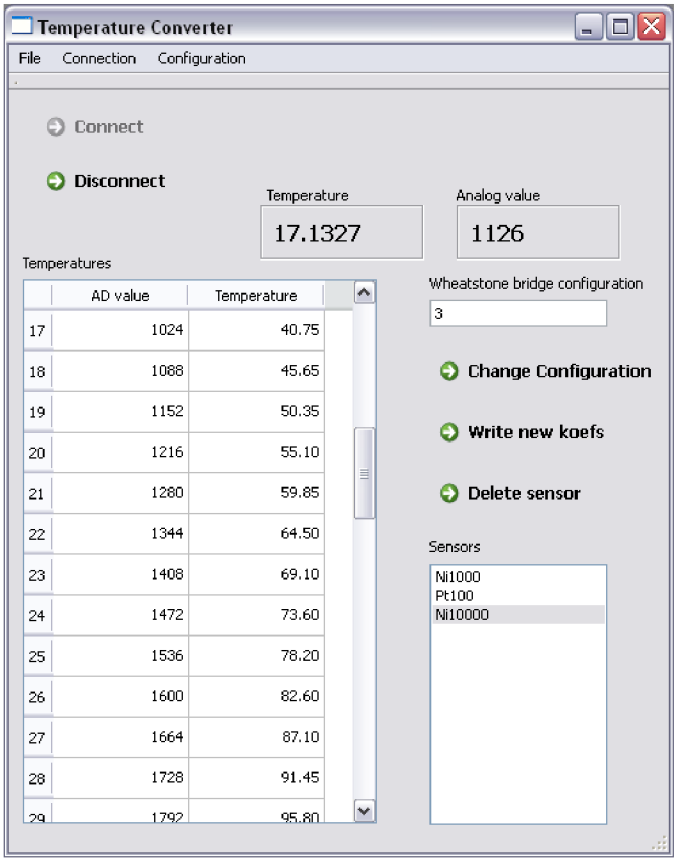
Aplikace Temperature converter poskytuje kompletní možnosti správy snímačů převodníku, což zahrnuje nahrávání nových snímačů až do maximálního počtu pěti. Mazání snímačů a upravování koeficientů pomocí nichž je v převodníku počítána teplota (kalibraci). Aplikace rovněž zobrazuje každou vteřinu aktualizované hodnoty výstupu A/D převodníku a z něho vypočtené měřené teploty. Aplikace sestává z následujících souborů

- **calibration.cpp** - třída dialogového okna pro kalibraci snímače
- **ComputeEngine.cpp** - třída pro výpočet teplotních konstant pro zadaný snímač
- **ConversionOperations.cpp** - statická třída obsahující operace pro převod mezi formátem big endian a little endian
- **creatordialog.cpp** - obsahuje třídu dialogového okna pro přidání snímače
- **IpSet.cpp** - třída dialogového okna pro nastavení parametrů síťové komunikace
- **Modbus.cpp** - obsahuje třídu realizující Modbus klienta
- **RtdSensors.cpp** - obsahuje třídu s rovnicemi snímačů pro výpočet odporu při dané hodnotě teploty

- **TcpIp.cpp** - třída pro vlastní komunikaci přes rozhraní Ethernet(poskytuje rozhraní pro přístup k tcp/ip stacku)
- **TempConverterInterface.cpp** - třída poskytující rozhraní mezi třídou Modbus klienta a vlastní uživatelskou aplikací
- **temperatureconverter.cpp** - třída hlavního uživatelského okna aplikace

(Ke všem uvedeným souborům v projektu samozřejmě existuje párový hlavičkový soubor s příponou .h)

6.1.1 Hlavní dialogové okno aplikace Temperature converter



Obrázek 6.1: Hlavní okno aplikace Temperature converter

Jednotlivé metody třídy CTemperatureConverter sestávají z obsluh událostí vyvolaných interakcí uživatele s ovládacími prvky dialogového okna aplikace. Kromě tlačítek pro připojení a odpojení od převodníku, s nímž chceme komunikovat, hlavní dialogové

okno obsahuje list, v němž se nachází seznam snímačů nahraných v aktuálně připojeném převodníku teploty. Dále se zde nachází textové pole, v němž je zapsána konfigurace měřícího můstku pro aktuálně vybraný snímač a tabulka obsahující dvojici hodnot, z nichž jedna představuje výstupní hodnotu A/D převodníku a druhá teplotu, která je v takovém případě měřena. K načtení v převodníku uložených snímačů dojde bezprostředně po navázání komunikace s převodníkem, tedy po stisku tlačítka connect. Tlačítky delete sensor a change koefs je možné z převodníku vymazat snímač nebo zapsat změněné hodnoty koeficientů pro výpočet teploty označeného snímače. Další funkce se skrývají v menu, odkud lze vyvolat dialogová okna pro nastavení Ip adresy zařízení, k němuž se připojujeme pro jednoduchou kalibraci koeficientů a pro nahrání nového snímače do převodníku. Hlavní okno aplikace je zobrazeno na obrázku 6.1

6.1.2 Třídy dialogových oken kalibrace a nastavení Ip adresy zařízení

Třídy CCalibration a CIpSet představují třídy dialogových oken. V dialogovém okně CIpSet může uživatel zadat Ip adresu zařízení, k němuž se má aplikace připojit a dialogové okno třídy CCalibration umožní nastavit konstantu, o níž se posunou koeficienty pro výpočet teploty aktuálně zapsané v tabulce. Třídy obsahují pouze veřejné metody:

QString *GetIpAddress*() - metoda třídy CIpSet, vrací hodnotu Ip adresy uživatelem zadanou v dialogovém okně.

int *GetKoefsShift*() - metoda třídy CCalibration, vrací hodnotu, o níž se mají posunout koeficienty aktuálně zapsané v tabulce.

6.1.3 Třída CCreatorDialog

Tato třída realizuje dialogové okno, jež umožňuje přidání nového snímače do převodníku teploty. Uživatel zde zadá identifikátor snímače, číslo konfigurace převodníku teploty a následně vyplní údaje nutné k vypočtení teplotních konstant používaných v převodníku k výpočtu teploty. Pomocí ComboBoxu umístěném nahoře, přibližně uprostřed dialogového okna, lze zvolit jednu z předdefinovaných rovnic popisující závislost elektrického odporu

snímače na teplotě. Textová pole v levé části aplikace slouží k zadání konstant nezbytných k výpočtu této rovnice. Veřejné metody této třídy představuje pouze metoda:

SSensorData& *GetNewSensorData*() - metoda vrací strukturu obsahující údaje o přídávaném snímači, data této struktury jsou bezprostředně po jejich získání nahrána do převodníku.

K vlastnímu výpočtu teplotních koeficientů slouží Třída CComputeEngine. Tato třída obsahuje pouze jedinou veřejnou metodu

void *NewDataSend*(SUserData* *userDataStruct*, QList<**double**>* *dataList*) - první parametr představuje strukturu sloužící jako kontejner pro údaje o snímači zadané uživatelem, jež budou následně použity pro výpočet teplotních koeficientů. Druhý parametr představuje ukazatel na kontejner, do nějž budou třídou vypočtené koeficienty uloženy.

6.1.4 Třída CTempConverterInterface

Poskytuje rozhraní mezi třídou CModbus a třídou vlastní uživatelské aplikace CTemperatureConverter. Třída obsahuje metody:

void *ReadSensorData*(**short** *sensor_num*) - vyčte údaje o jednotlivých v převodníku nahraných snímačích (identifikátor snímače, konfigurace měřícího můstku, koeficienty pro výpočet teploty)

void *WriteSensorData*(SSensorData& *sensData*) - nahraje do převodníku nový snímač

void *ReadNumberOfSensors*() - přečte počet v převodníku aktuálně nahraných snímačů

void *ReadTemperature*() - přečte aktuální hodnotu teploty

void *ReadAnalogValue*() - přečte akutální výstupní hodnotu A/D převodníku

void *ReadDataFromConverter*() - metoda pro periodické vyčítání z paměťové karty převodníku

void *ChangeConverterConfiguration*(**short** conf) - přepíná mezi v převodníku uloženými snímači

void *DeleteSensor*(**short** *sensor_num*) - vymaže z převodníku údaje pro daný snímač

void *ChangeSensorKoefs*(QVector< **float**>& *tempKoefs*, **int** *sens_num*) - změní koeficienty pro výpočet teploty převodníkem aktuálně používaného snímače (používá se při kalibraci)

void *ChangeConverterActualTime*() - funkce nastaví převodník na aktuální čas

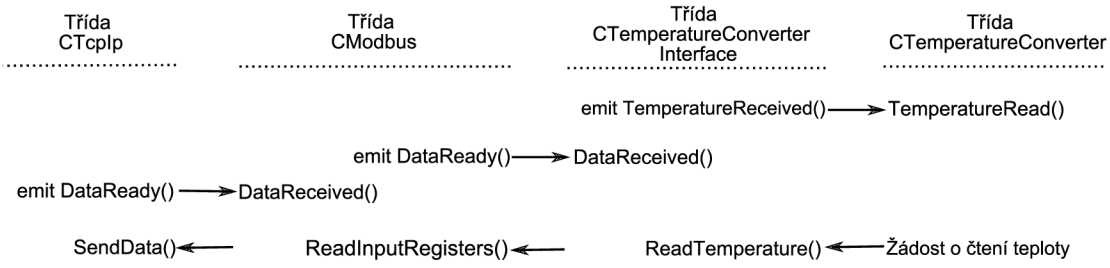
void *SetDeviceAddress*(QString *address*) - slouží k nastavení ip adresy zařízení, s nimž budeme z aplikace Temperature converter komunikovat

6.1.5 Postup zpracování požadavků

Odeslaní Modbus požadavku převodníku sestává ze zřetězeného volání metod několika tříd, přičemž platí, že třída nadřízená volá vždy, v hirearchii odesílání požadavku, metodu třídy podřízené. Například v případě žádosti vyčtení teploty z převodníku, třída CTemperatureConverter volá metodu ReadTemperature třídy CTempConverterInterface, z níž je volána metoda ReadHoldingRegisters třídy CModbus a až z této třídy je volána metoda *SendData* třídy CTcpIp, zajišťující vlastní odeslání požadavku po rozhraní Ethernet.

Pro opačný postup, kdy je mezi uvedenými třídami nutné v opačném pořadí předávat přijatou odpověď, bylo využito rozšíření Qt frameworku jazyka C++, skládající se z tzv. signálů a slotů. Signály v podstatě představují události, jež jsou vyvolávány klíčovým slovem emit a sloty pak představují metody třídy volané v případě, že dojde k výskytu události. K napojení slotu na určitý signál (událost) slouží funkce
connect(uzatel_na_třídu_emitující_událost, název_událost,ukazatel_na_třídu_obsahující_slot, název_slotu)
Následující řádek kódu
connect(&m_tcpConnection,SIGNAL(DataReady(char,int)),this,SLOT(DataReceived(char*,int)));*
tedy způsobí, že pokud dojde v některé z metod třídy CTcpIp k vyvolání události pomoci *emit DataReady(data, length)*; dojde k zavolání metody *DataReceived* třídy CModbus. V aplikaci pak třída CTcpIp ve chvíli, kdy přijme zprávu, emituje událost DataReady, jež způsobí vyvolání metody *DataReceived*() třídy CModbus. Tímto způsobem jsou přijatá

data předána až na nejvyšší úroveň třídě CTemperatureConverter. Příklad volání metod mezi jednotlivými třídami při požadavku čtení teploty z převodníku zobrazuje obrázek 6.2



Obrázek 6.2: Vztah mezi jednotlivými třídami při komunikaci s převodníkem

6.1.6 Třída CModbus

Třída CModbus slouží ke skládání vlastní zprávy protokolu Modbus. Kromě třídy CModbus soubor Modbus.c dále obsahuje pomocnou třídu CModbusRequest, sloužící jako kontejner pro vytvářenou Modbus zprávu. Třída CModbus obsahuje následující veřejné metody:

void ReadHoldingRegisters(**int** *beginAddress*, **int** *length*) - metoda vytvoří Modbus zprávu pro funkci čtení více registrů z datové tabulky uchovávacích registrů.

void *ReadInputRegisters*(**int** *beginAddress*, **int** *length*) - metoda vytvoří Modbus zprávu pro čtení vstupních registrů.

void *WriteMultipleRegisters*(**int** *beginAddress*, **int** *length*, QVector<**short**> *values*) - metoda vytvoří Modbus zprávu pro funkci zápisu do více uchovávacích registrů.

void *WriteSingleRegister*(**int** *address*, **int** *value*) - metoda vytvoří Modbus zprávu pro

void *ReadFileRecord*(**int** *fileNumber*, **int** *recordNumber*, **int** *recordLength*) - metoda vytvoří Modbus zprávu pro funkci čtení záznamů ze souboru, pomocí níž je realizováno vyčítání dat z paměťové karty převodníku.

void *SetDeviceAddress*(QString& *address*) - nastavení Ip adresy zařízení, s nímž chceme komunikovat

Aby nadřazené třídy nemuseli při odesílání požadavků čekat, až dojde k přijetí odpovědi pro požadavek aktuálně vyslaný, obsahuje třída CModbus frontu, kde jsou nevyřízené požadavky hromaděny. Požadavky jsou zde ukládány zapouzdřené v pomocné třídě CModbusRequest. Fronta je typu FIFO, realizováná pomocí třídy QList. A vždy po úspěšném přijetí odpovědi na předchozí odeslaný požadavek je z této fronty vyjmut a odeslán nejstarší uložený požadavek.

Aby nedošlo k tomu, že v případě poruchy na zařízení bude program na některý ze zaslaných požadavků čekat donekonečna, je v programu implementován časovač nastavený na 1 s, po jeho uplynutí dojde k znovuvyslání požadavku. Maximální počet opakovaných vyslání činí pro každý požadavek tři. Pokud i po třetím vyslání požadavku zůstane tento bez odpovědi, program vyprázdní frontu požadavků a oznámí uživateli poruchu na síti.

6.1.7 Třída CTcpIp

Třída CTcpIp obaluje standardní třídu prostředí Qt QTcpSocket, představující rozhraní pro přístup k TCP/IP stacku a tedy komunikaci po Ethernetu. Třída obsahuje veřejné metody:

bool *SendData*(**char*** *buffer*, **int** *numberOfBytes*) - odešle data po Ethernetu, první parametr buffer obsahuje řetězec bytů, jež má být odeslán a proměnná numberOfBytes, pak udává délku tohoto řetězce.

void *SetDeviceAddress*(QString& *Address*) - metoda udávající Ip adresu zařízení, s nímž chceme po Ethernetu komunikovat (pokud tato metoda není volána použije se defaultně adresa 192.168.0.2)

6.1.8 Kalibrace snímače

Nepříliš účinnou variantu kalibrace převodníku teploty představuje snížení všech konstant pro výpočet teploty o konstantní hodnotu, jíž poskytuje třída CCalibration. Takto lze snímač zkalibrovat v podstatě pouze pro měření v aktuálním pracovním bodě.

Lepší variantou je kalibrace převodníku pomocí úpravy Calendar-Van Dusenovi křivky [29], kdy pro několik různých teplot z měřícího rozsahu snímače odečteme aktuální hodnotu odporu snímače. Křivku tvořenou těmito hodnotami pak nahrajeme do převodníku teploty, kde bude použita pro výpočet teploty. Vytvořený převodník teploty využívá obdobný princip, zaznamenávány jsou dvojice hodnot, výstupní hodnota z A/D převodníku a příslušná teplota. Hodnoty A/D převodníku, pro něž musí být přesná hodnota teploty udána, není volitelná, nýbrž je v programu pevně dána s krokem 64. Teplota tedy bude zaznamenána pro hodnoty výstupu A/D převodníku 64,128,192 atd. Kalibrace pak může probíhat způsobem, že pomocí nastavitelného odporového normálu připojeného místo snímače, budeme postupně nastavovat výstup A/D převodníku do hodnot 64, 128, 192 atd. a pomocí známé velikosti odporu při těchto hodnotách můžeme vypočíst teplotu, která této hodnotě odporu pro daný snímač odpovídá. Postupně upravíme všechny hodnoty teploty v tabulce aplikace Temperature converter. Kalibrace pomocí úpravy křivek Calendar-Van Dusenovi křivky představuje účinný způsob jak minimalizovat nejistoty měření. Další výhodu této metody představuje snadná možnost automatizace, takto prováděné kalibrace [29].

6.2 Aplikace pro vyčítání z paměťové karty

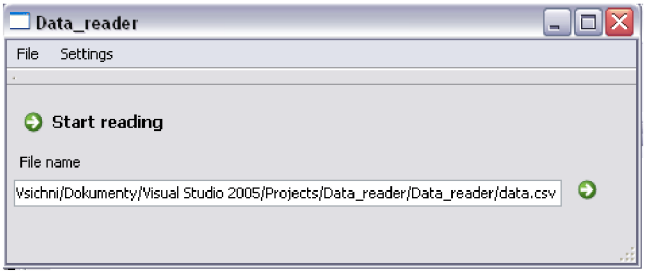
Účelem této aplikace je vyčtení veškerých dat na paměťové miniSD kartě připojené k převodníku teploty. V této aplikaci lze také zadat název a umístění .csv souboru, do nějž budou vyčtená data uložena. Tato aplikace komunikuje s převodníkem teploty stejně jako aplikace Temperature converter po Ethernetu pomocí aplikačního protokolu Modbus. Soubory ConversionOperations.c, TcpIp.c a Modbus.c obsahující třídy CTcpIp a CModbus jsou stejné jako v případě aplikace Temperature_converter. Dále tato aplikace sestává ze souborů:

- **data_reader.c** - obsahuje třídu hlavního dialogového okna aplikace
- **FileSaver.c** - obsahuje třídu pro uložení načtených dat do souboru
- **Interface.c** - obsahuje třídu rozhraní mezi třídou Modbus klienta a uživatelskou aplikací

- **Progress.c** - obsahuje třídu dialogového okna s ukazatelem kolik procent dat, již bylo vyčteno z paměťové karty převodníku.

6.2.1 Třída CDataReader

Třída hlavního dialogového okna aplikace obsahuje metody pro interakci ovládacích prvků dialogového okna a uživatele. Tlačítko slouží k vyvolání dialogu pro výběr souboru, do nějž mají být data uložena(pokud není žádný soubor vybrán, aplikace neumožní data z převodníku vyčíst). Hlavní okno aplikace Data reader zobrazuje obrázek 6.3



Obrázek 6.3: Hlavní okno aplikace Data reader

6.2.2 Třída CInterface

Třída poskytuje rozhraní mezi třídou CModbus podporující komunikaci po protokolu Modbus a hlavní třídou uživatelské aplikace CDataReader. Jelikož smyslem aplikace Data_reader je pouze vyčítání z karty, omezují se metody implementované v této třídě na operace s tím související. Třída obsahuje metody:

void *ReadDataFromConverter*() - metoda volaná za účelem vyčtení dat z paměťové karty převodníku

void *DeleteDataFromMemoryCard*() - metoda volaná za účelem vymazání dat na paměťové kartě

void *SetFileName*(QString *FileName*) - metoda nastaví cestu k souboru, do nějž mají být data uložena.

Pro vizualizaci stavu vyčítání tato třída využívá třídu CProgress představující dialogové okno zobrazující, z kolika procent je již operace vyčítání hotova. Toto okno je zobrazeno po celou dobu vyčítání dat z paměťové karty převodníku. Třída obsahuje veřejné metody:

void *SetMax*(**int** *max*) - metoda nastaví hodnotu při níž bude ukazatel průběhu zobrazovat, že operace již byla vykonána ze 100 %

void *SetProgress*(**int** *progress*) - metoda nastaví kolik dat již bylo vyčteno z paměťové karty, třída následně z této hodnoty a metodou *SetMax* nastavené maximální hodnoty vypočte, z kolika procent již byla operace vyčítání provedena.

Ukládání dat do paměťové karty zajišťuje třída CFileSaver. Tato třída obsahuje veřejné metody:

bool *SaveDataToFile*(QList<SSensorData> *dataList*, QString& *fileName*) - uloží do souboru s příponou .csv data předaná v prvním parametru. Struktura SSensorData má stejnou podobu jako struktura SD_dataStruct používaná v programu mikrokontroléru Rabbit (QList představuje standardní třídu prostředí Qt a realizuje v podstatě dynamicky vázané pole).

bool *WriteHeader*(QString& *fileName*, QString& *header*) - zapíše do souboru hlavičku jejíž podoba je předána druhým parametrem.

Se zápisem do souboru aplikace nečeká na chvíli, až budou z převodníku vyčtena všechna data, ale data jsou do souboru zapisován průběžně vždy, když počet záznamů přesáhne určitou v programu pevně nastavenou hodnotu.

6.2.3 Formát ukládaného souboru

Soubor s příponou .csv je prostým textovým souborem, v němž každý řádek odpovídá jednomu měření, přičemž jednotlivé údaje pro každé měření jsou odděleny středníkem. V prvním sloupci tabulky se nachází měřená teplota, další sloupce pak obsahují vteřinu, minutu, hodinu, den, měsíc a rok, tedy časovou značku, kdy byla daná teplota změřena. Poslední sloupec udává, zda byl v danou chvíli vyvolán alarm.

7 ZMĚNY V ZAPOJENÍ

Zapojení převodníku tak jak bylo navrženo nekompenzuje vliv přívodních vodičů snímače na měření. Kalibraci převodníku pro daný snímač je tedy nutné provést s k převodníku již připojenými přívodními vodiči a v případě jejich změny musí dojít k nové kalibraci snímače. Možností, jak odpor přívodních vodičů eliminovat, by bylo měření odporu přívodního vodiče, resp. úbytku napětí na něm. K tomuto účelu by bylo možné využít druhý kanál A/D převodníku. Hodnotu odpovídající úbytku napětí na tomto vodiči by poté bylo možné odečíst od úbytku napětí snímače. Pro tuto kompenzaci by však bylo zapotřebí, aby měřící můstek byl napájen z proudového zdroje. Proudový zdroj je obecně pro napájení odporových snímačů teploty vhodnější než zdroj napěťový. V přílohách se proto nachází návrh zapojení a desky plošných spojů, které uvedené nedostatky opravují.

Místo třívodičového připojení snímače teploty toto zapojení využívá zapojení čtyřvodičové, kde samotný snímač je napájen z proudového zdroje realizovaného pomocí jednoho z operačních zesilovačů obvodu MCP602. Relé, která původně sloužila pro konfiguraci měřícího můstku, v tomto zapojení nastavují velikost proudu proudového zdroje. Pro zesílení napětí snímače teploty zapojení opět využívá přístrojového zesilovače AD620. Zbývající zesilovač obvodu MCP602 je použit jako aktivní filtr druhého řádu, na nějž je signál snímače přiveden před převodem do digitální podoby. Jako A/D převodník byl tentokrát zvolen obvod AD7790. Jedná se o 16-bitový převodník převádějící rozdílové napětí přivedené na jeho vstupy. Na jeden ze vstupů je pak přiváděno výstupní napětí filtru a druhý je připojen k zemi.

Kromě měřící části došlo ke změnám i v napájení, kde spínaný regulátor TSR1-2450 je již použit pouze k napájení digitální části zapojení, zatímco analogová část je napájena obvodem LF50CV. Spínaný regulátor byl původně použit i pro napájení některých citlivých součástek analogové části zapojení převodníku.

V softwarové části práce by bylo zapotřebí především dopracovat, aby k synchronizaci hodin reálného času modulu RCM3900 bylo využito některého z na internetu dostupných serverů poskytujících informaci o aktuálním čase. Rovněž by bylo vhodné dopracovat zabezpečení komunikace převodníku, aby data byla přenášena v šifrované podobě.

8 ZÁVĚR

Na úvod této práce byli popsány základní principy měření teploty. Následně byl představen teoretický úvod do komunikačního standardu Ethernet a aplikačního protokolu Modbus. Další kapitoly se věnovali realizaci samotného převodníku teploty. Bylo představeno vlastní zapojení převodníku teploty a některé v zapojení použité obvody. Zbytek práce se pak již věnoval vytvořenému softwarovému vybavení. Na konci práce byl představen upravený návrh převodníku, který potlačuje nedostatky hardwarového návrhu.

Výsledný převodník umožnuje měření teploty pomocí standardních odporových teplotních snímačů. Údaje o jednotlivých snímačích teploty je možné do převodníku nahrávat a následně z něj mazat. Uživatel tak není omezen pouze na několik pevně v převodníku definovaných teplotních snímačů, nýbrž pro měření teploty může použít snímač pro nějž převodník nakonfiguruje. Převodník teploty komunikuje s nadřazeným systémem po Ethernetu pomocí aplikačního protokolu Modbus TCP. Převodník rovněž průběžně zaznamenává naměřené hodnoty na paměťovou kartu, odkud mohou být následně vyčteny.

Implementované webové rozhraní pak umožňuje zobrazení převodníkem měřené teploty, základní konfiguraci některých parametrů převodníku a nastavení teplotních mezí, při jejichž překročení je uživateli zaslána varovná emailová zpráva.

V rámci práce byli rovněž vytvořeny dvě aplikace pro PC. První sloužící pro kalibraci a základní konfiguraci převodníku teploty. Druhá pak pro vyčítání dat z paměťové karty připojené k převodníku a následnému zapsání těchto dat do uživatelem zvoleného souboru.

REFERENCE

[1] Teplota. *Wikipedia: otevřená encyklopedie*. [online]. [cit. 2009-11-18] Dostupné z WWW: <http://cs.wikipedia.org/wiki/Teplota#M.C4.9B.C5.99en.C3.AD_teploty>.

[2] KREIDL, Marcel. *Měření teploty : Senzory a měřící obvody*. 1. vydání. Praha : BEN-technická literatura, 2005. [cit. 2010-04-09]. 230 s.

[3] RIPKA, Pavel; ĎAĎO, Stanislav; KREIDL, Marcel; NOVÁK, Jiří. *Senzory a převodníky*. Praha : Česká technika, 2005. [cit. 2010-05-02]. 135 s.

[4] ZEZULKA, František; HYNČICA, Ondřej. Průmyslový Ethernet I : Historický úvod. *Automa* [online]. 2007, 01, [cit. 2010-05-15]. Dostupný z WWW: <http://www.uamt.feec.vutbr.cz/žezulka/download/KPPA/A010741-I.pdf>

[5] ZEZULKA, František; HYNČICA, Ondřej. Průmyslový Ethernet II : Referenční model ISO/OSI. *Automa* [online]. 2007, 03, [cit. 2010-05-15]. Dostupný z WWW: <http://www.uamt.feec.vutbr.cz/žezulka/download/KPPA/A030786_II.pdf>

[6] DOSTÁLEK Libor; KABELOVÁ Alena. *Velký průvodce protokoly TCP/IP a systémem DNS*. Brno : Computer Press, 2002. [cit. 2010-05-02]. 542 s.

[7] RONEŠOVÁ, Andrea. *Přehled protokolu MODBUS.* [online]. květen 2005 [cit. 2010-04-09]. Dostupné z WWW: <http://home.zcu.cz/ ronesova/bastl/files/modbus.pdf>.

[8] Modbus-IDA [online]. December 28, 2006 [cit. 2010-04-09]. Modbus application protocol specification v1.1b. Dostupné z WWW: <http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf>.

[9] Modbus-IDA [online]. October 24, 2006 [cit. 2010-04-09]. Modbus messaging on TCP/IP implementation guide v1.0b. Dostupné z WWW: <http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf>.

[10] NATIONAL INSTRUMENTS *Measuring Temperature with an RDT or Thermistor* [onlione]. April 2003 [cit. 2010-03-25]. Dostupné z WWW: <labview360.com/document/an/pdf/an046.pdf>

[11] MICROCHIP TECHNOLOGY INC. *Temperature Measurement Circuits for Embedded Applications* [online]. 2004 [cit. 2010-03-25]. Dostupné z WWW: <ww1.microchip.com/downloads/en/AppNotes/00929a.pdf>

[12] PANJIT *1N4001 THRU 1N4007* [online]. 2002 [cit. 2010-04-12]. Dostupné z WWW: <http://www.datasheetcatalog.com/datasheets_pdf/1/N/4/0/1N4004.shtml>

[13] TRACOPOWER*TSR-1 Series* [online]. [cit. 2010-04-12]. Dostupné z WWW: <http://www.soselectronic.com/a_info/resource/pdf/ine/TSR1.pdf>

[14] FAIRCHILD SEMICONDUCTOR *TL431/TL431a Programmable Shunt Regulator* [online]. 2003 [cit. 2010-04-12]. Dostupné z WWW: <www.fairchildsemi.com/ds/TL% 2FTL431.pdf>

[15] MAXIM *Switched Capacitor Voltage Conversion* [online]. 1994 [cit. 2010-04-10]. Dostupné z WWW: <datasheets.maxim-ic.com/en/ds/ICL7660-MAX1044.pdf>

[16] MIROSLAV, Patočka; VOREL, Pavel. *Řídící elektronika - aktivní obvody : 2. díl*. Brno, 2004. [cit. 2010-05-09]. 154 s.

[17] ANALOG DEVICES. *Low Cost, Low Power Instrumentation Amplifier AD620.*[online]. 1999 [cit. 2009-03-16]. Dostupný z WWW: <www.mit.tut.fi/MIT-1016/AD620.pdf>

[18] BRZEZINA, Petr. *Charakteristika čidla teploty Pt100* [online]. 2008 [cit. 2010-05-10] Dostupné z WWW: <http://www.sensit.cz/images/stories/download/charakteristiky_cidel/Pt100_3850.pdf>

[19] BRZEZINA, Petr. *Charakteristika čidla teploty Ni1000* [online]. 2008 [cit. 2010-05-10] Dostupné z WWW: <http://www.sensit.cz/images/stories/download/charakteristiky_cidel/Ni1000_5000.pdf>

[20] BRZEZINA, Petr. *Charakteristika čidla teploty Ni10000* [online]. 2008 [cit. 2010-05-10] Dostupné z WWW: <http://www.sensit.cz/images/stories/download/charakteristiky_cidel/Ni10000_5000.pdf>

[21] MICROCHIP TECHNOLOGY INC. *MCP3202 - 2.7V dual channel 12-bit A/D converter with SPI serial Interface* [online]. 2006 [cit. 2010-04-28]. Dostupné z WWW: <ww1.microchip.com/downloads/en/DeviceDoc/21034D.pdf>.

[22] RABBIT SEMICONDUCTOR*RabbitCore RCM3900: User's manual* [online]. 2008 [cit. 2010-05-01]. Dostupné z WWW: <http://www.rabbit.com/hottag/index.php?ht=/documentation/docs/manuals/RCM3900/RC3900UM.pdf>.

[23] *RABBIT SEMICONDUCTOR* [online]. 2010 [cit. 2010-05-12]. RCM3900 RabbitCore®. Dostupné z WWW: <http://www.rabbit.com/products/rcm3900/>

[24] COSMO ELECTRONICS CORPORATION *Product specification* [online]. 2010 [cit. 2010-05-12]. RCM3900 RabbitCore®. Dostupné z WWW: <http://www.gme.cz/_dokumentace/dokumenty/634/634-033/dsh.634-033.1.pdf>

[25] RABBIT SEMICONDUCTOR*Dynamic C: User Manual* [online]. 2007-2008 [cit. 2010-04-09]. Dostupné z WWW: <http://www.rabbit.com/hottag/index.php?ht=/documentation/docs/manuals/DC/DCUserManual9/DCPUM.pdf>.

[26] RABBIT SEMICONDUCTOR *TCP/IP User´s Manual: volume 1* [online]. 2007-2009 [cit. 2010-05-02]. Modbus Dostupné z WWW: <http://www.rabbit.com/documentation/docs/manuals/TCPIP/UsersManualV1/index.html>.

[27] RABBIT SEMICONDUCTOR *TCP/IP User´s Manual: volume 2* [online]. 2006-2008 [cit. 2010-04-09]. Modbus Dostupné z WWW: <http://www.rabbit.com/documentation/docs/manuals/TCPIP/UsersManualV2/index.html>.

[28] *Google code : flot*[online]. 2010 [cit. 2010-04-20]. Dostupné z WWW: <http://code.google.com/p/flot/>

[29] Uchopte správně měření teploty. *Automa* [online]. 2009, č.06, [cit. 2010-05-12]. Dostupný z WWW: <http://www.odbornecasopisy.cz/index.php?id_document=39155>.

[30] JANOVSKÝ, Dušan. *Jak psát web* [online]. [cit. 2010-05-17]. Dostupné z WWW: <http://www.jakpsatweb.cz/>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

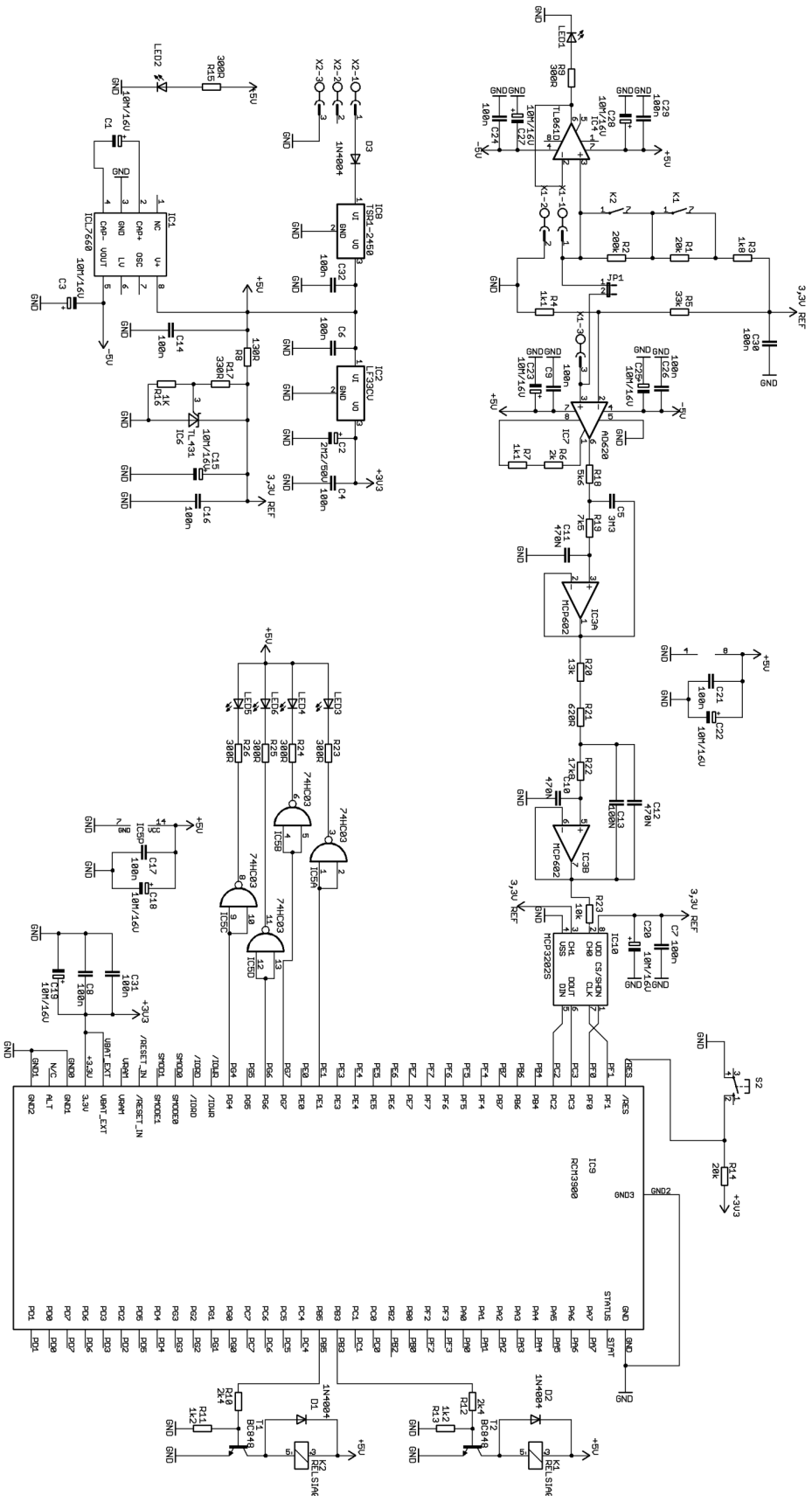
ADU	Application Data Unit
CPU	Central Processing Unit
CSV	Comma Separated Value
EEPROM	Electrically Erasable Programmable Read-Only Memory
GPL	General Public License
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
LSB	Least Significant Byte
MBAP	Modbus Application Header
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
MSB	Most Significant Byte
PC	Personal computer
PDU	Protocol Data Unit
SMTP	Simple Mail Transfer Protocol
STP	Shielded Twisted Pair
TCP	Transmission Control Protocol
UTP	Unshielded Twisted Pair

SEZNAM PŘÍLOH

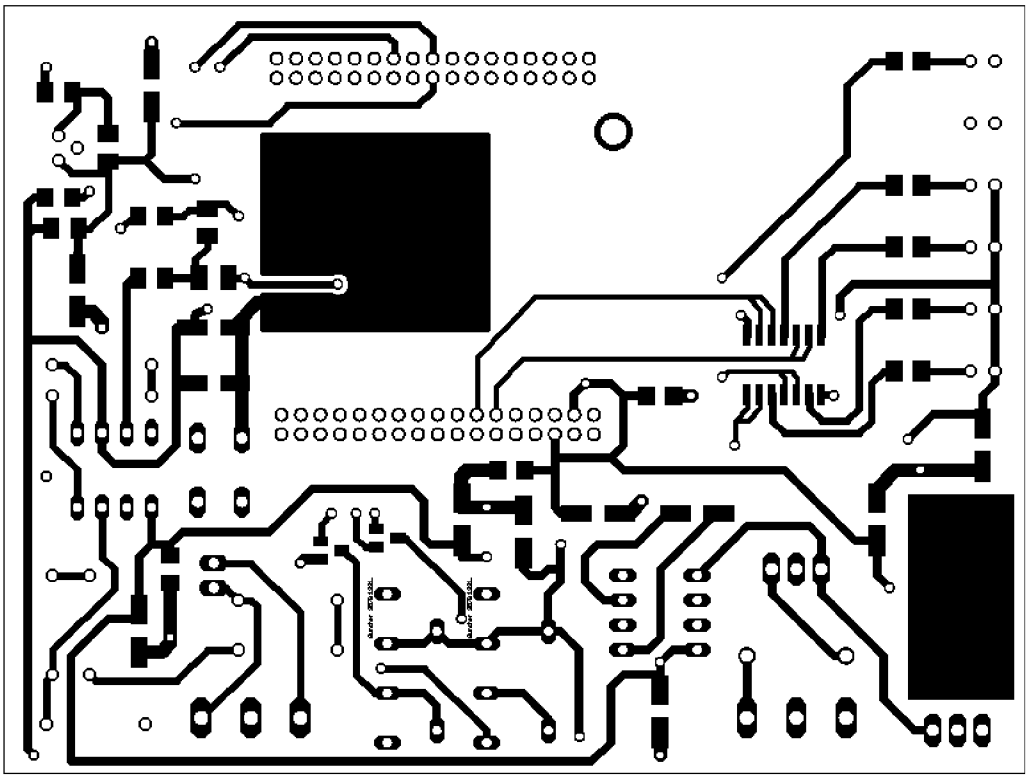
A SEZNAM SOUČÁSTEK

Rezistory			
R1	20k	R2	200k
R3	1k8	R4,R7	1k1
R5	33k	R6	2k
R8	130R	R9,R15,R23,R24,R25,R26	300R
R10,R12	2k4	R11,R13	1k2
R14	20k	R16	1k
R17	330R	R18	5k6
R19	7k5	R20	13k
R21	620R	R22	17k8
Kondenzátory			
C1,C3,C15,C18,C19	10M/16V SMD	C2	2M2/50V
C4,C6,C7,C8,C9	100n	C5	3M3
C10,C11,C12	470n	C13,C14,C16,C17,C32	100n
C19,C20,C22,C23,C27,C28	10M/16V	C21,C24,C26,C29,C30,C31	100n
Diody			
D1,D2,D3	1N4004	LED1,LED4	L-934HD
LED2,LED5	L-934GD	LED3,LED6	L-HLMP-1401
Tranzistory			
T1,T2	BC848 SMD		
IO			
IC1	ICL7660	IC2	LF33CV
IC3	MCP602-I/SN	IC4	TL061D
IC5	74HC03 SMD	IC6	TL431
IC7	AD620	IC8	TSR1-2450
IC9	RCM3900	IC10	MCP3202-CI/SN
Relé			
K1,K2	Relsia05-500		

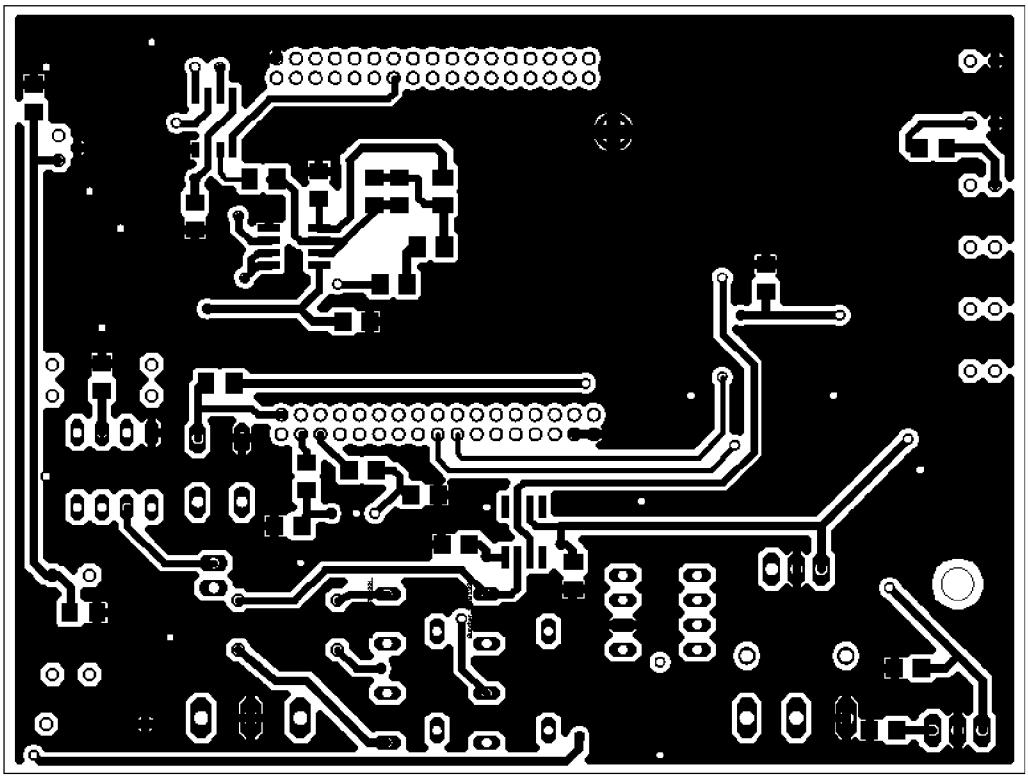
B SCHÉMA ZAPOJENÍ



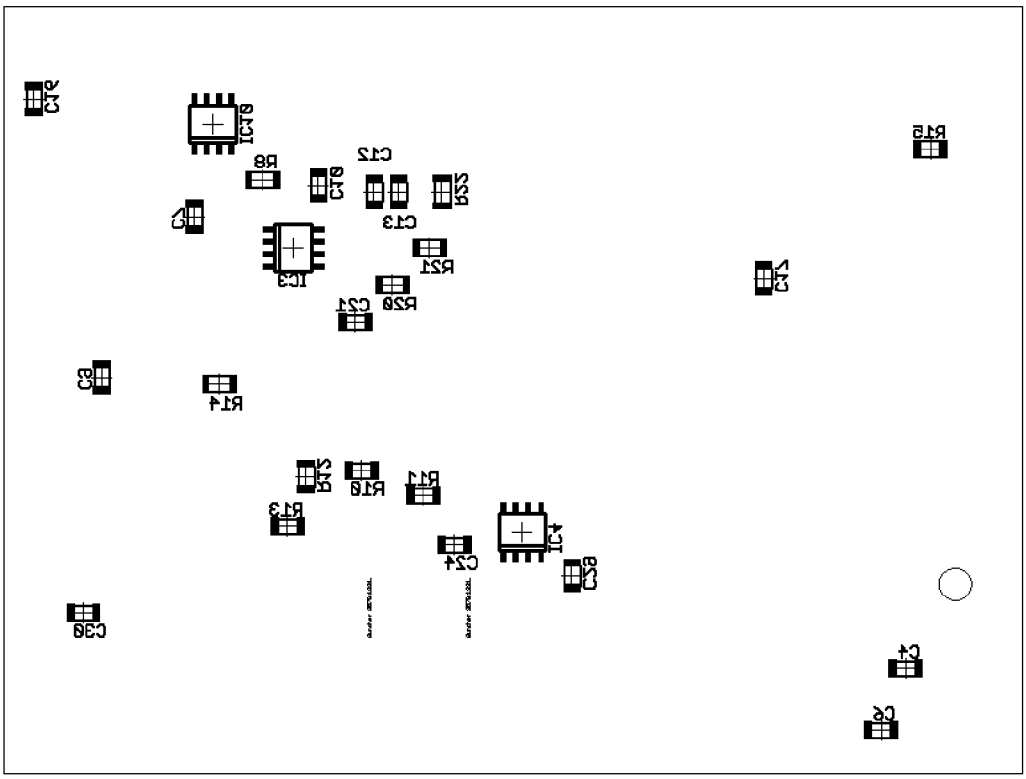
C DESKA PLOŠNÝCH SPOJŮ - SPODNÍ STRANA



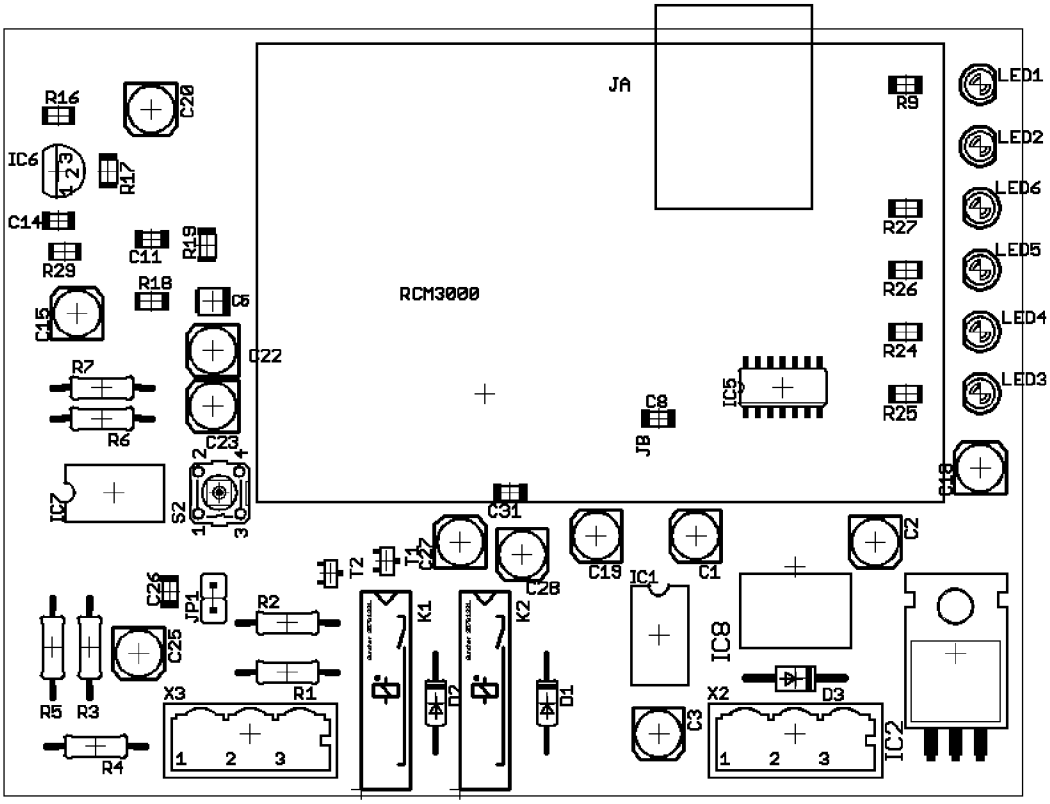
D DESKA PLOŠNÝCH SPOJŮ - VRCHNÍ STRANA



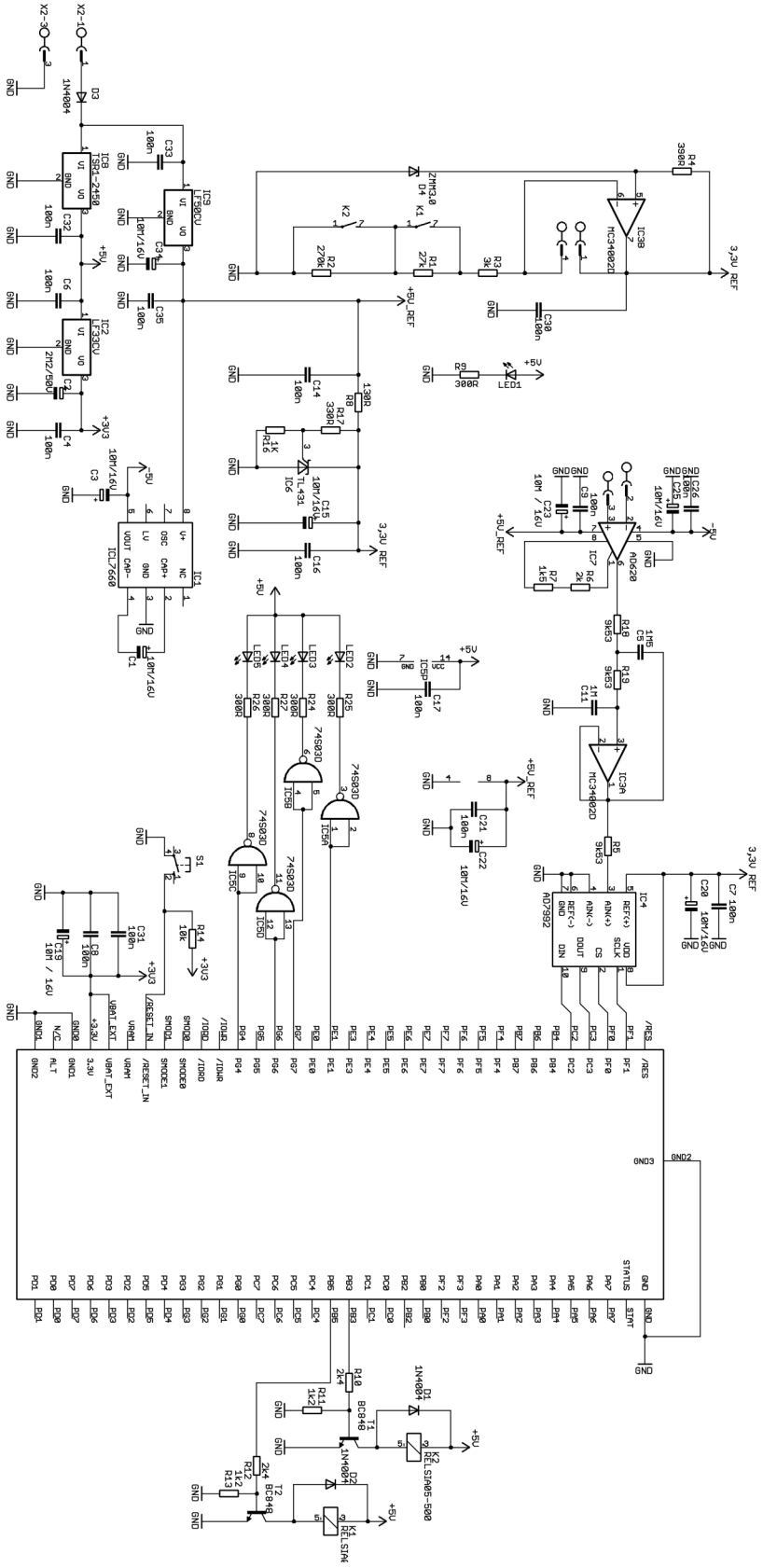
E OSAZOVACÍ SCHÉMA - SPODNÍ STRANA



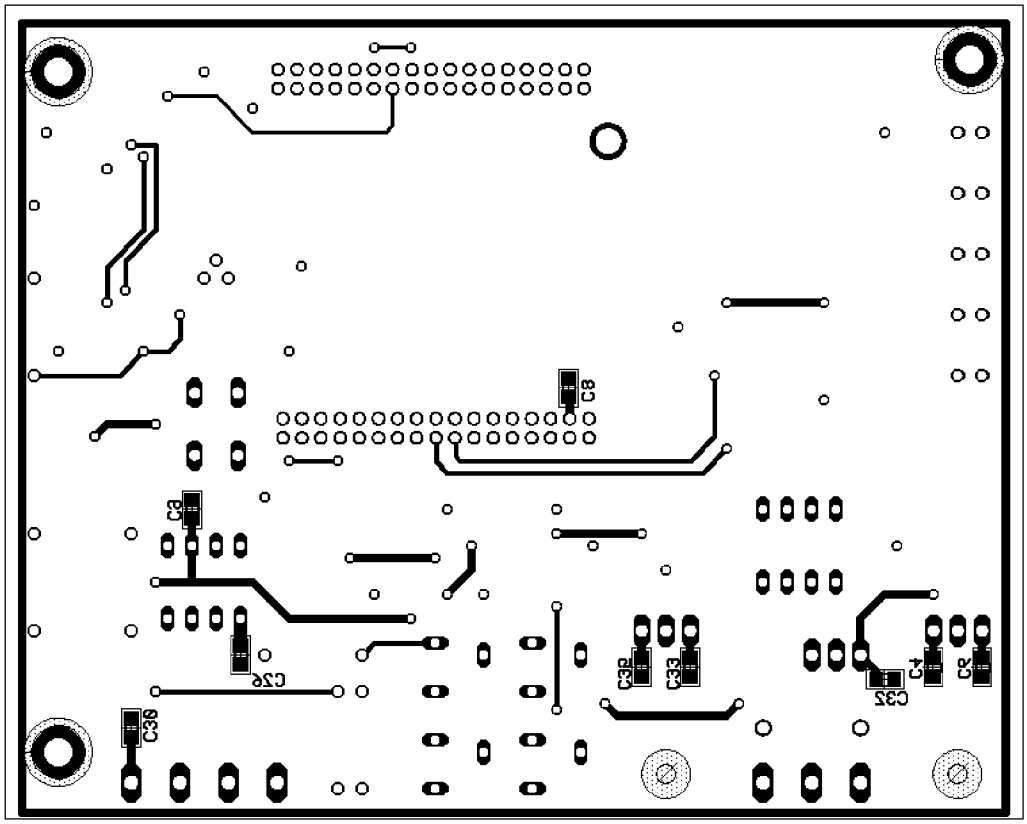
F OSAZOVACÍ SCHÉMA - VRCHNÍ STRANA



G UPRAVENÉ SCHÉMA ZAPOJENÍ



H DESKA PLOŠNÝCH SPOJŮ(UPRAVENÉ ZAPOJENÍ) - SPODNÍ STRANA



I DESKA PLOŠNÝCH SPOJŮ(UPRAVENÉ ZAPOJENÍ) - VRCHNÍ STRANA