UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

ŘÍZENÍ TEPLOTY V MÍSTNOSTI POMOCÍ ELETRONICKÉ REGULACE VENTILU TOPNÉHO TĚLESA

Marek Lochman

Bakalářská práce

2015

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Marek Lochman**
Osobní číslo:	**I11329**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Řízení procesů**
Název tématu:	**Řízení teploty v místnosti pomocí elektronické regulace ventilu topného tělesa**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Cílem práce je hardwarová a softwarová realizace řídicího systému na bázi mikrokontroléru Atmel AVR (řady ATmega8), který bude ovládat servopohon ventilu topného tělesa a řídit tak teplotu v místnosti. Řídicí systém bude prostřednictvím čidla průběžně měřit aktuální teplotu, přičemž její průběh v závislosti na čase bude možné graficky znázornit. Aplikace bude umožňovat uživatelskou volbu nastavení požadované teploty pro danou denní dobu (možnost více přednastavených teplot během dne).

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

[1] VÁŇA, V. Mikrokontroléry ATMEL AVR: popis procesoru a instrukční soubor. Praha: BEN - technická literatura, 2003. 336 s. ISBN 978-80-7300-083-0.
[2] VÁŇA, V. Mikrokontroléry ATMEL AVR: programování v jazyce C. Praha: BEN - technická literatura, 2003. 216 s. ISBN 978-80-7300-102-0.
[3] VLACH, J. Řízení a vizualizace technologických procesů. Praha: BEN - technická literatura, 2002. 160 s. ISBN 978-80-86056-66-X.
[4] VALTER, J. Regulace v praxi: aneb jak to dělám já. Praha: BEN - technická literatura, 2010. 176 s. ISBN 978-80-7300-256-5.

Vedoucí bakalářské práce: **Ing. Libor Kupka, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **18. listopadu 2013**
Termín odevzdání bakalářské práce: **9. května 2014**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2014

Prohlášení autora:

Prohlašuji, že tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 26.4. 2015 Marek Lochman

Poděkování

Chtěl bych poděkovat vedoucímu práce, Ing. Liboru Kupkovi, Ph.D., za konzultace při tvorbě této práce a poskytnutí odborných rad.

V Pardubicích dne 26. 4. 2015

Marek Lochman

ANOTACE

Cílem práce je hardwarová a softwarová realizace řídicího systému na bázi mikrokontroléruAtmel AVR (řady ATmega8), který bude ovládat servopohon ventilu topného tělesa a řídit tak teplotu v místnosti. Řídicí systém bude prostřednictvím čidla průběžně měřit aktuální teplotu, přičemž její průběh v závislosti na čase bude možné graficky znázornit. Aplikace bude umožňovat uživatelskou volbu nastavení požadované teploty pro danou denní dobu (možnost více přednastavených teplot během dne).

KLÍČOVÁ SLOVA

Měření teploty, regulace, mikroprocesor, h-můstek.

TITLE

ROOM TEMPETATURE CONTROL USING THE ELECTRONIC HEATER CONTROL VALVE

ABSTRACT

Theaimofthis thesis is hardware and software implementationof a control systém based on a microcontrollerAtmel AVR (ATmega8 series), whichwillcontroltheservo drive oftheheatervalve and thuscontrolthetemperature in theroom. Thecontrol systém willcontinuouslymeasurethecurrenttemperatureusing a sensor, wherebyitscoursedepending on thetimewillbepossible to depictgraphically. Theapplicationwillallowuser to set desiredtemperaturefor a givendaytime (the posibility of more presettemperaturesduringa day).

KEYWORDS

Thermostaticvalvecontrol, microprocessor, h-bridge.

Obsah

Seznam zkratek 9
Seznam značek 10
Seznam obrázků 11
Seznam tabulek 12
ÚVOD 13
1 TEORETICKÁ ČÁST 14
1.1 VENTILY TOPNÉHO TĚLESA 14
1.1.1 Termostatický ventil 14
1.2.2 Termostatická hlavice 15
1.2 MĚŘENÍ TEPLOTY 15
1.3 OPTICKÁ ZÁVORA 16
1.4 I2C 16
1.4.1 Princip přenosu 17
1.5 H MŮSTEK 17
1.6 RTC 18
1.7 RS232 19
1.7.1 Způsob komunikace RS232 19
1.7.2 Asynchronní přenos 19
1.8 MIKROPROCESOR 20
1.8.1 USART 20
1.8.2 Obecný popis čítače/časovače 20
1.8.3 Režimy čítače/časovače 0 21
2 REALIZACE 22
2.1 REGULACE 22
2.2 TMP275 23
2.3 FT232RL 23
2.3.1 Popis zapojení FT232RL 23
2.4 DS1307 24
3 HARDWAROVÁ REALIZACE 25
3.1 VOLBA MIKROPROCESORU 25
3.2 NAPAJECÍ ZDROJ 26
3.2.1 Popis schématu zapojení napájecího zdroje 27

3.3 REGULAČNÍ ZAŘÍZENÍ 27
3.3.1 Popis zapojení regulačního zařízení 28
3.4 ZAŘÍZENÍ TERMO HLAVICE 30
3.4.1 Popis zařízení termo hlavice 30
3.4.2 Popis schématu zapojení termo hlavice 31
4 GRAFICKÝ SOFTWARE 32
4.1 POPIS GRAFICKÉ ČÁSTI 32
ZÁVĚR 33
Literatura 35
Přílohy 36

Seznam zkratek

A/D	analogově digitální převodník
I^2C	multimasterová počítačová sériová sběrnice
LCD	displej z tekutých krystalů
LED	dioda emitující světlo
PWM	pulzně šířková modulace
SPI	sériové periferní rozhraní
USART	univerzální synchronní/asynchronní přijímač a vysílač

Seznam značek

I	elektrický proud, A
R	elektrický odpor, Ω
U	elektrické napětí, V

Seznam obrázků

Obr. 1.1 – Řez termostatickým ventilem (Matz, 2009) 14
Obr. 1.2 – Řez termostatickou hlavicí (TA Hydronics, 2006) 15
Obr. 1.3 – Optická závora 16
Obr. 1.4 – Příklad zapojení MASTER – SLAVE 16
Obr. 1.5 – Příklad přenosu I^2C na vodičích SDA a SCL (Olejár, 2000) 17
Obr. 1.6 – Proud při otáčení motoru po směru hodinových ručiček 18
Obr. 1.7 – Proud při otáčení motoru proti směru hodinových ručiček 18
Obr. 1.8 – Příklad asynchronního přenosu (Olmr, 2005) 19
Obr. 1.9 – Blokové schéma čítače/časovače 0 20
Obr. 1.10 – Režim fázově korigovaný PWM (Tůma, 2011) 21
Obr. 1.10 – Režim fázově korigovaný PWM (Tůma, 2011) 22
Obr. 2.1 – Blokové schéma TMP275 23
Obr. 2.2 – Schéma zapojení FT232RL 24
Obr. 2.3 – Schéma zapojení DS1307 25
Obr. 3.1 – Blokové schéma procesoru ATmega48 26
Obr. 3.2 – Schéma zapojení zdroje 27
Obr. 3.3 – Fotografie regulačního zařízení 28
Obr. 3.4 – Schéma zapojení regulačního zařízení 29
Obr. 3.5 – Fotografie zařízení termohlavice 30
Obr. 3.6 – Schéma zapojení termohlavice 31
Obr. 4.1 – Nastavení požadovaných hodnot 32
Obr. 4.2 – Definování sepnutí zásuvky 33
Obr. 4.3 – Grafické zobrazení 33

Seznam tabulek

Tab.1.1 – Tabulka pro nastavení čítače/časovače 0.....21

ÚVOD

Cílem bakalářské práce je vytvořit zařízení pro ovládání teploty v místnosti nahrazením hlavice termostatického ventilu topení. Pomocí tepelného čidla získává zařízení informace o aktuální teplotě v místnosti. Po porovnání s požadovanou hodnotou dle potřeby přivře nebo naopak otevře ventil. Zařízení je konstruováno tak, aby šlo aplikovat na dnes používané termostatické ventily. V softwaru zařízení je možné nastavit dvacet čtyři hodnot (pro každou jednotlivou hodinu dne) požadované teploty v místnosti. Další funkcí zařízení bude sepnutí zásuvky pro přídavné topení při překročení nejnižší požadované teploty na předdefinovaném intervalu dne.

1 TEORETICKÁ ČÁST

Tato část obsahuje teorii potřebou k sestavení regulátoru, je zde popsána teorie ventilů topného tělesa a jeho nejčastější provedení termostatický ventil. Poté je popsána funkce termostatické hlavice. Teorie měření teploty, optické závory, princip přenosu sběrnice I^2C, funkce h-můstku, RTC, rozhraní RS232 a hlavní částí zařízení mikroprocesor.

1.1 VENTILY TOPNÉHO TĚLESA

Dříve používané ventily neumožňovaly automatickou regulaci topného tělesa. Záleželo pouze na uživateli, jakým způsobem ventil manuálně nastavil. Tímto způsobem ovšem nebylo možné zajistit, aby vytápění bylo rovnoměrné a hospodárné. V případě, že uživatel opustil místnost a nechal plně otevřený ventil, docházelo k přetopení prostoru. V současnosti je již tento způsob ovládání vytápění zastaralý. Nyní většina domácností využívá termostatické ventily a termostatické hlavice umožňující plynulou regulaci (Matz, 2009).

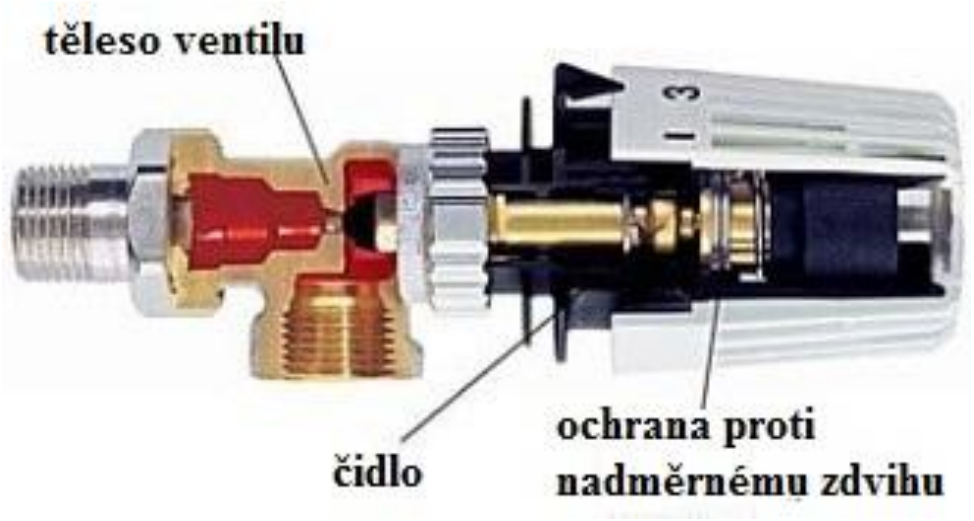
1.1.1 Termostatický ventil

Termostatický ventil se skládá z těla ventilu a vložky, která obsahuje kuželku.Tato kuželka při zatlačení omezuje přívod vody protékající ventilem. Vložka termostatického ventilu je zašroubována uvnitř ventilu. Jednotlivé vložky ventilu se liší dle tvaru kuželky, ale jejich princip zůstává stejný (Matz, 2009).

Obr. 1.1 – Řez termostatickým ventilem (Matz, 2009)

1.1.2 Termostatická hlavice

Termostatická hlavice slouží k teplotní regulaci vzduchu v místnosti. Na základě přednastavené teploty reguluje průtok teplé vody skrz termostatický ventil a následné proudění do otopného tělesa. Termostatická hlavice funguje na principu tepelné roztažnosti, kdy citlivá látka obsažená v řídicím snímači přímo ovlivňuje stlačení vlnovce. Ten tlačí na kuželku ventilu a reguluje tak průtok média topením (Matz, 2009).



Obr. 1.2 – Řez termostatickou hlavicí (TA Hydronics, 2006)

1.2 MĚŘENÍ TEPLOTY

Teplota je jednou ze základních jednotek fyzikální soustavy SI. Přesto její fyzikální definice je obtížná. Nejvíce se udává pojem, že látka při kontaktu s jinou látkou bude neustále v nerovnováze, tedy zda bude dávat nebo naopak přijímat teplo. Výše teploty se udává za pomocí jednotek tepla Kelvin (K), stupeň Celsia (°C), stupeň Fahrenheita (F) nebo Rankinův stupeň (°R). Teplotu lze dnes měřit velkým množstvím čidel a odvětví měření teploty se neustále rozvíjí. V elektronice se nejčastěji používají termočlánky, termistory, odporové teploměry a integrované obvody. Pro správný údaj naměřené teploty je důležité umístění čidla v místnosti. Například kdyby čidlo bylo umístěno u okna, vlivem slunečního záření by naměřená hodnota mohla být nepřesná. Důležitým faktorem při měření teploty je také rozlišení čidla, které udává množství informace na jednotku teploty.

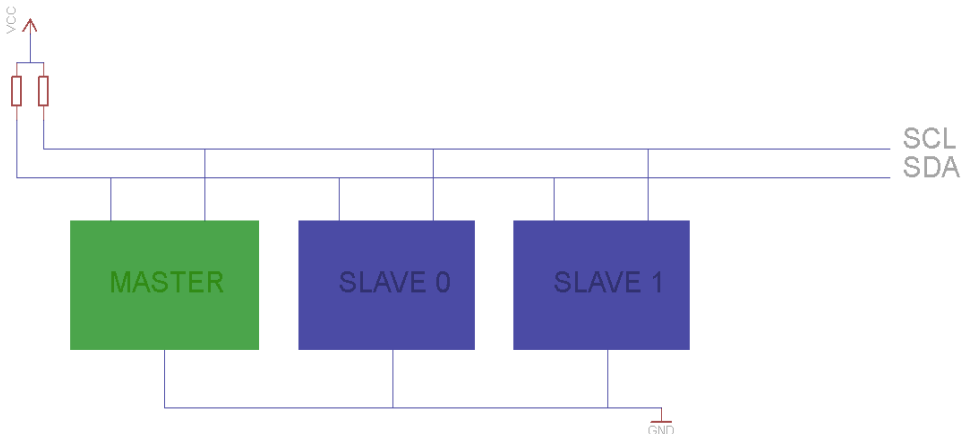
1.3 OPTICKÁ ZÁVORA

Optická závora je elektronicko-optické zařízení, skládající se z LED diody nebo laserové diody a fototranzistoru. Paprsek diody dopadá na přechod PN fototranzistorů, který se otevře a začne jím procházet proud. Pokud je přenos paprsku přerušen například listem papíru mezi diodou a fototranzistorem, dojde k uzavření PN přechodu. Tímto způsobem lze vyrábět pulzy (například pulzně šířkové modulace), které lze načítat nebo odečítat, a zjišťovat polohu otevření nebo uzavření ventilu.

Obr. 1.3 – Optická závora

1.4 SBĚRNICEI^{2}C

Sběrnice I^{2}C byla vyvinuta firmou Philips přibližně před 20 lety. Jedná se o vnitřní datovou sběrnici sloužící k dorozumívání integrovaných obvodů s následným přenosem dat. Největší výhodou této technologie je, že přenos probíhá obousměrně pouze po dvou vodičích, sériové datové lince SDA a lince hodinového signálu SCL, což výrazně ulehčuje zapojení. V dnešní době tento přenos využívá celá řada integrovaných obvodů – například jednočipové počítače, sériové paměti či A/D nebo D/A převodníky a další specializovaná zařízení (Olejár, 2000).

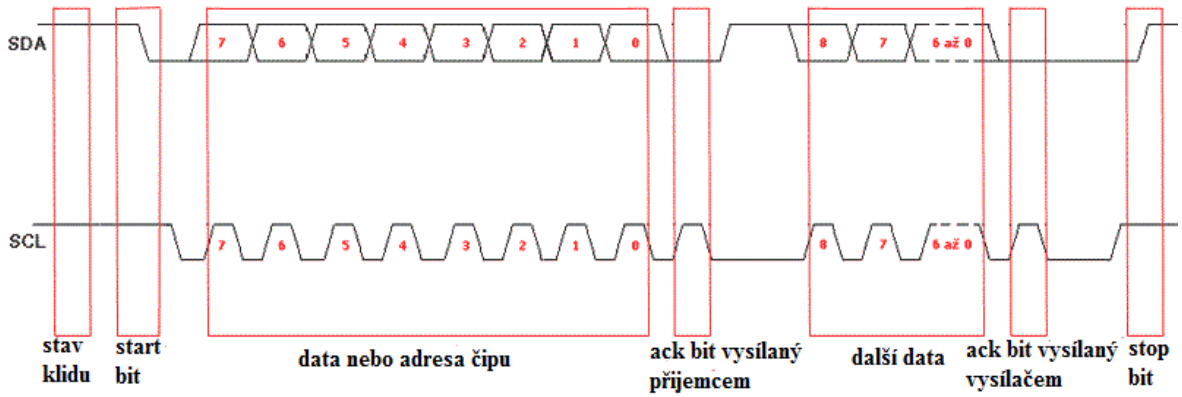


Obr. 1.4 – Příklad zapojení MASTER – SLAVE

1.4.1 Princip přenosu

Jeden nebo více integrovaných obvodů zastává roli nadřazenou (Master), ostatní jsou zapojeny jako podřízené (Slave). Master generuje hodinový signál, který slouží jako nositel informace. Při přenosu informace je nejprve nutné definovat adresu, která určuje, pro který čip jsou data určena. Poté je definováno, zda jde o čtení nebo zápis dat. Data se přenášejí po 1 bajtu od nejvíce významného bitu po nejméně významný bit. Přenos probíhá v následujících stavech (Olejár, 2000):

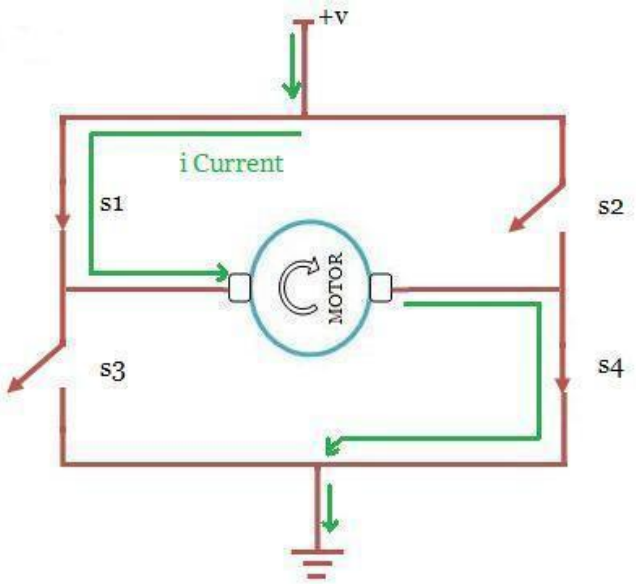
- stav klidu – na obou vodičích SDA a SCL je logická jednička; tento stav je způsoben „pull up“ rezistory
- start bit – na SDA se změní logická jednička na logickou nulu; na SCL je logická jednička; start bit zahajuje přenos
- stop bit – na SDA se změní logická nula na logickou jedničku; na SCL je logická jednička; tento stav ukončuje přenos
- bit Ack – slouží ke kontrole správného přijmutí dat



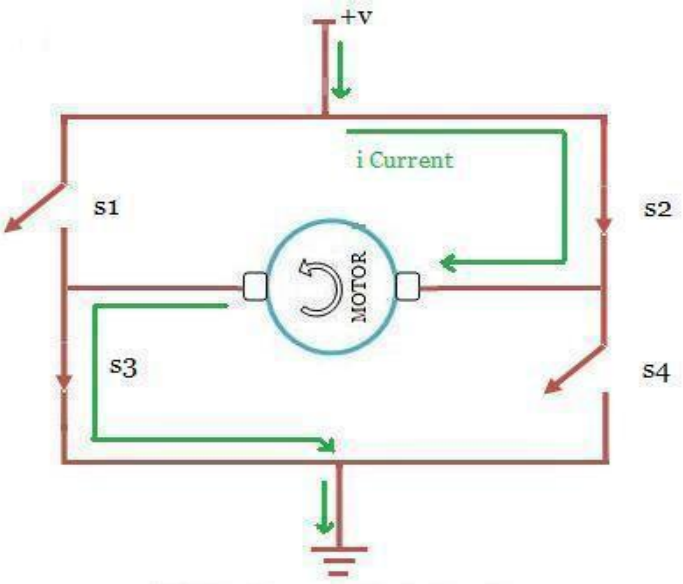
Obr. 1.5 – Příklad přenosu I^2C na vodičích SDA a SCL (Olejár, 2000)

1.5 H-MŮSTEK

H-můstek je obvod se čtyřmi spínacími prvky (jedná se obvykle o bipolární nebo FET tranzistory) se zátěží uprostřed. Touto zátěží bývá obvykle stejnosměrný DC motor. Pokud chceme, aby se motor točil po směru hodinových ručiček, sepneme spínače S1 a S4. Jestliže požadujeme, aby se motor točil proti směru hodinových ručiček, sepneme spínače S2 a S3. Zároveň je nutné zajistit, aby nedošlo současně k sepnutí spínačů S1 a S3,případně S2 a S4, v takovém případě totiž hrozí zničení obvodu.



Obr. 1.6 – Proud při otáčení motoru po směru hodinových ručiček



Obr. 1.7 – Proud při otáčení motoru proti směru hodinových ručiček

1.6 RTC

Pokud je potřeba v jakémkoli zařízení načítat, ukládat nebo jinak zpracovávat aktuální čas, využívá se obvodu reálného času. Tyto obvody využívají ke své činnosti oscilátoru, většinou krystalu, který je zdrojem hodinového signálu. Tento signál se vydělí tak, aby bylo možné počítat vteřiny, od kterých je možné oddělovat čas. Hodiny reálného času využívají pro komunikaci s mikroprocesorem rozhraní I^2C.V sestavovaném zařízení byl použit obvod DS1307.

1.7 ROZHRANÍRS232

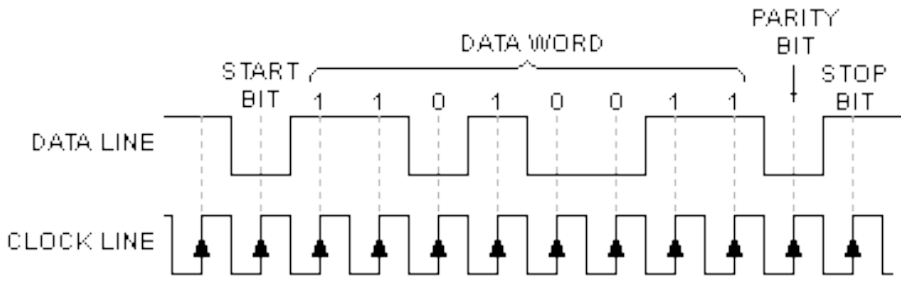
Rozhraní RS232 bylo vyvinuto pro komunikaci dvou nebo více zařízení (například počítače a tiskárny). V dnešní době se však tento způsob technologie v komunikaci s počítačem již téměř nevyskytuje a ustupuje rychlejšímu a praktičtějšímu USB. Nicméně v průmyslových aplikacích se tohoto způsobu přenosu pořád hojně využívá. Přenos probíhá synchronně nebo asynchronně (to určuje hodinový signál) od nejméně významného po nejvíce významný bit. Úrovně napětí používané rozhraním RS232 jsou pro logickou jedničku −3 až −15 V a pro logickou nulu 3 až 15 V(Olmr, 2005).

1.7.1 Způsob komunikace rozhraníRS232

RozhraníRS232 může komunikovat jak asynchronně, tak synchronně. Zpravidla se ale používá asynchronní přenos z důvodu nižšího počtu vodičů (Olmr, 2005).

1.7.2 Asynchronní přenos

Pokud neprobíhá přenos, je datová linka na vysoké úrovni napětí. Přenos využívá dvou datových linek – Rx a Tx – ale princip si vysvětlíme na jedné. Přenos začíná start bitem, kdy na sestupné hraně hodinového signálu se změní hodnota datové linky z logické jedničky na logickou nulu a nastaví se synchronizace. Tato synchronizace trvá po celou dobu stavu startu bitu a udává odchylku mezi datovým a hodinovým signálem. Poté následuje n-bitové slovo o velikosti 5 až 9 bitů. Nejčastěji se používá8bitů (=1 byte). Přenos probíhá od nejméně významného bitu (LSB) po nejvíce významný bit (MSB). Po n-bitovém slově je umístěn paritní bit, který slouží pro kontrolu správného přenosu signálu. Například sudá parita kontroluje, zda počet jedniček přenosu plus paritní bit je sudé číslo.Přenos je ukončen stop bitem, u pomalejších zařízení můžou být i dva stop bity (Olmr, 2005).



Obr. 1.8 – Příklad asynchronního přenosu (Olmr, 2005)

1.8 MIKROPROCESOR

Mikroprocesor je řídicí jednotkou celého přípravku. Řídí regulaci kovového ventilu. Při sestavování jsem vybral typ AVR ATmega48. Je to 8bitový mikroprocesor typu RISC. Disponuje dvěma sériovými rozhraními UART. Hodí se pro komunikaci přes rozhraní RS232.

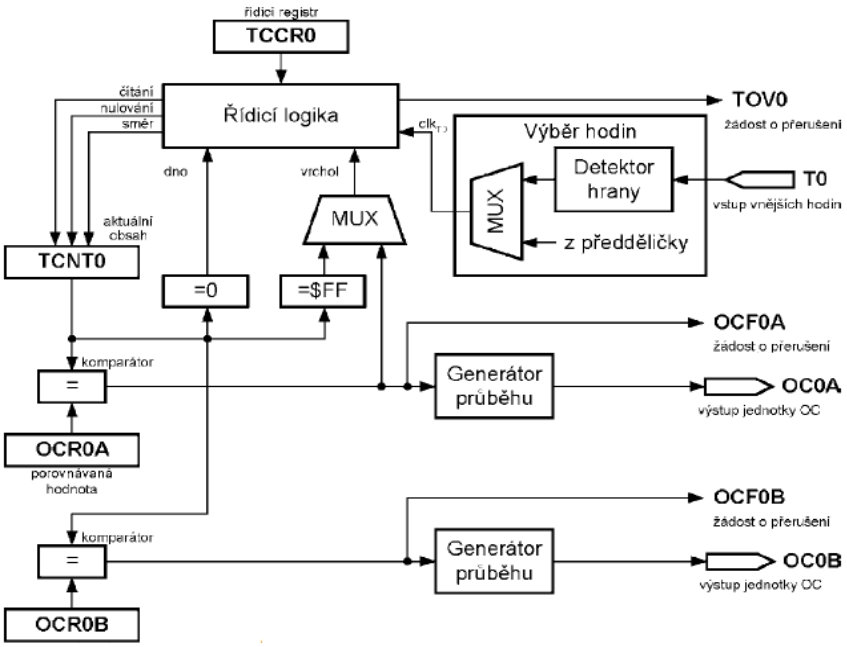
1.8.1 USART

Jedná se o univerzální komunikátor pro sériovou komunikaci podporující asynchronní a synchronní přenos. Nastavení parametru přenosu se nastaví registrem.

1.8.2 Obecný popis čítače/časovače

Jednou ze základních funkcí procesoru je funkce čítače/časovače. ATmega48 obsahuje jeden 16bitový a dva 8bitové čítače/časovače schopné pracovat v několika režimech.

Každý čítač/časovač má registr TCNTx. Tento registr obsahuje aktuální hodnotu čítače odvozeného od hodinového (taktovacího) signálu mikrořadiče. Mezi taktovacím signálem a tímto registrem je možné nastavit předděličku frekvence. Registry OCR představují takzvaný komparační registr, jehož hodnota je neustále porovnávána s obsahem TCNT. Nastane-li shoda, může dojít buď k přerušení, nebok události dle zadaných kritérií(Matoušek, 2006).



Obr. 1.9 – Blokové schéma čítače/časovače 0

1.8.3 Režimy čítače/časovače 0

Mikroprocesor obsahuje celou řadu jednolitých režimů čítače/časovače. Volba jednotlivých režimů se provádí pomocí hodnoty WGM (Wave Form Generation Mode), která je umístěna v registru TCCR (Matoušek, 2006).

Tab. 1.1 – Tabulka pro nastavení čítače/časovače 0

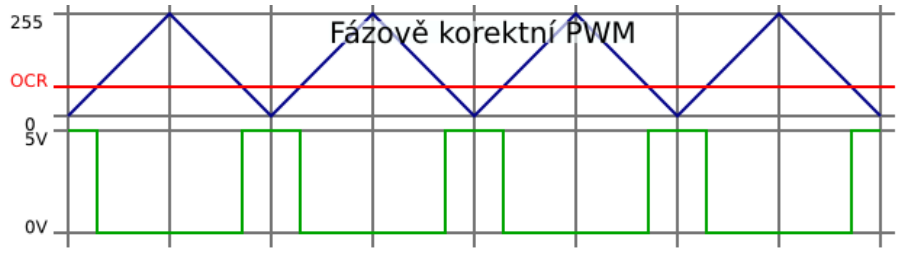
WGM	Režim	Vrchol	OCR0x	příznaku TOV0
0	normální	0xFF	okamžitě	při maximu
1	fázově korigovaný PWM	0xFF	na vrcholu	na dnu
2	CTC	OCR0A	okamžitě	při maximu
3	rychlý PWM	0xFF	na vrcholu	při maximu
4	*vyhrazeno (nepoužívat)*	–	–	–
5	fázově korigovaný PWM	OCR0A	na vrcholu	na dně
6	*vyhrazeno (nepoužívat)*	–	–	–
7	rychlý PWM	OCR0A	na vrcholu	na vrcholu

Normální režim (WGM=0)

V tomto režimu se vždy čítá nahoru do 255 kroků. Jedná se o nejjednodušší a nejrychlejší režim – pouze načítá náběžné nebo sestupné hrany od taktování procesoru. Když dosáhne svého maxima, přeteče a nastane přerušení, nebo se nastaví příznak přetečení a čítá se znovu od nuly (Matoušek, 2006).

Fázově korigovaný (WGM=1 nebo WGM = 5)

Čítá do hodnoty 255 v režimu jedna nebo do nastavené hodnoty OCR v režimu pět. Od normálního režimu se liší ve způsobu čítaní, které probíhá nahoru i dolů. Pokud se dočítá do hodnoty OCR výstupní pin OCXX se změní z logické jedničky na logickou nulu (Matoušek, 2006). Příklad průběhu je na obr. 1.10.



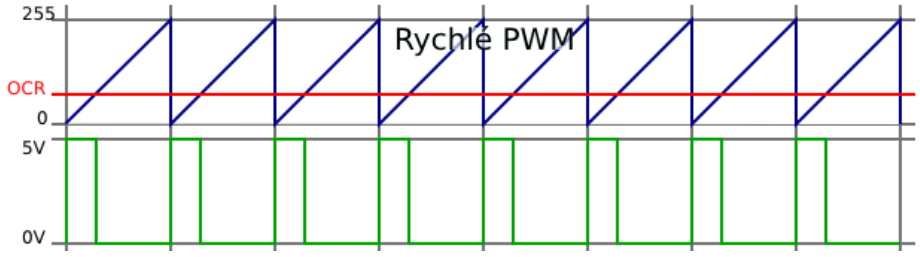
Obr. 1.10 – Režim fázově korigovaný PWM (Tůma, 2011)

CTC režim (WGM=2)

V tomto režimu se při shodě registru TCNT registrem OCR vynuluje čítač. OCR udává rozlišení čítače. Tento režim je vhodný pro měření frekvence (Matoušek, 2006).

Rychlý PWM režim (WGM = 3 nebo WGM =7)

Nejrychlejší z režimů PWM. Čítání probíhá od 0 do 255 nebo do hodnoty nastavené v OCR. Po přetečení se režim vynuluje a čítání probíhá znovu. Podobně jako u fázově korigovaného PWM nastane změna při shodě OCR a výstupního pinu OCXX. Při přetečení čítače dojde k nastavení výstupního pinu do logické jedničky. Příklad průběhu je na obr. 1.11 (Tůma, 2011).



Obr. 1.11 – Režim fázově korigovaný PWM (Tůma, 2011)

2 REALIZACE

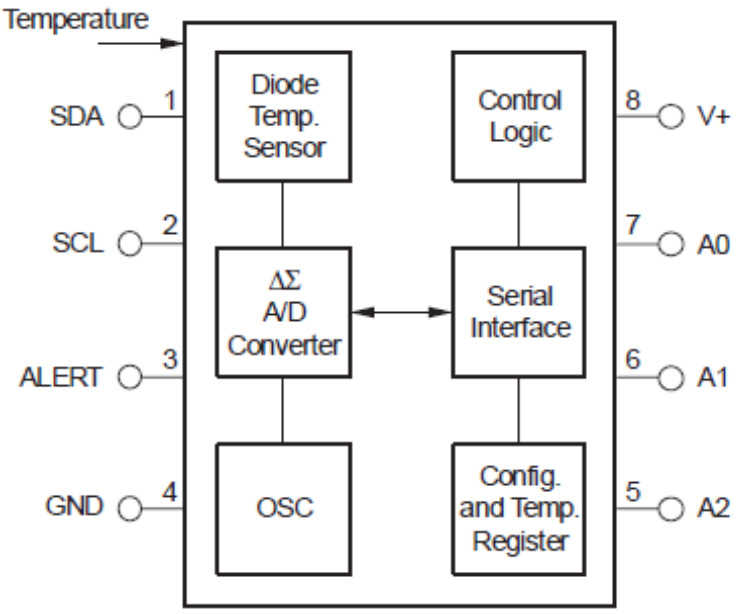
V této kapitole je popsán způsob regulace. Teplotní čidlo TMP275, které slouží pro měření teploty v místnosti. Rozhraní FT232RL přes které je zajištěna komunikace mezi počítačem a regulačním zařízením a hodiny reálného času DS1307.

2.1 REGULACE

Otočné zařízení disponuje 32 kroky, od 0, kdy je ventil zcela otevřen, až do31, kdy je ventil naopak uzavřen a průtok teplé vody zcela zamezen. Jeden krok reprezentuje průchod paprsku fotodiody na fototranzistor. 32 kroků je rozděleno po 0,1 °C, tedy od −1,6 °C (0) až po 1,5 °C (31). Když je tedy požadovaná teplota 25 °C a teplota v místnosti 25 °C, ventil je otevřen na 15. stupni. Když požadovaná teplota v místnosti bude 26 °C a teplota v místnosti bude 25 °C, ventil se otevře na15 − 10, tedy na 5. stupni. Když je požadovaná teplota 25 °C a okolní teplota 24 °C, tak se ventil otevře na 15 + 10, tedy na 25. stupni.

2.2 TMP275

TMP275 je integrovaný obvod pro měření teploty s přesností 0,5 °C. Podporuje komunikaci I^2C.Je schopný obsluhovat až 8 zařízení po jedné sběrnici. TMP275 čte teplotu s rozlišením 0,0625 °C. Klidový odebíraný proud je 50 µA, v pohotovostním režimu je množství odebíraného proudu 0,1 µA. Napájecí napětí obvodu je 2,7 V až 5,5 V.



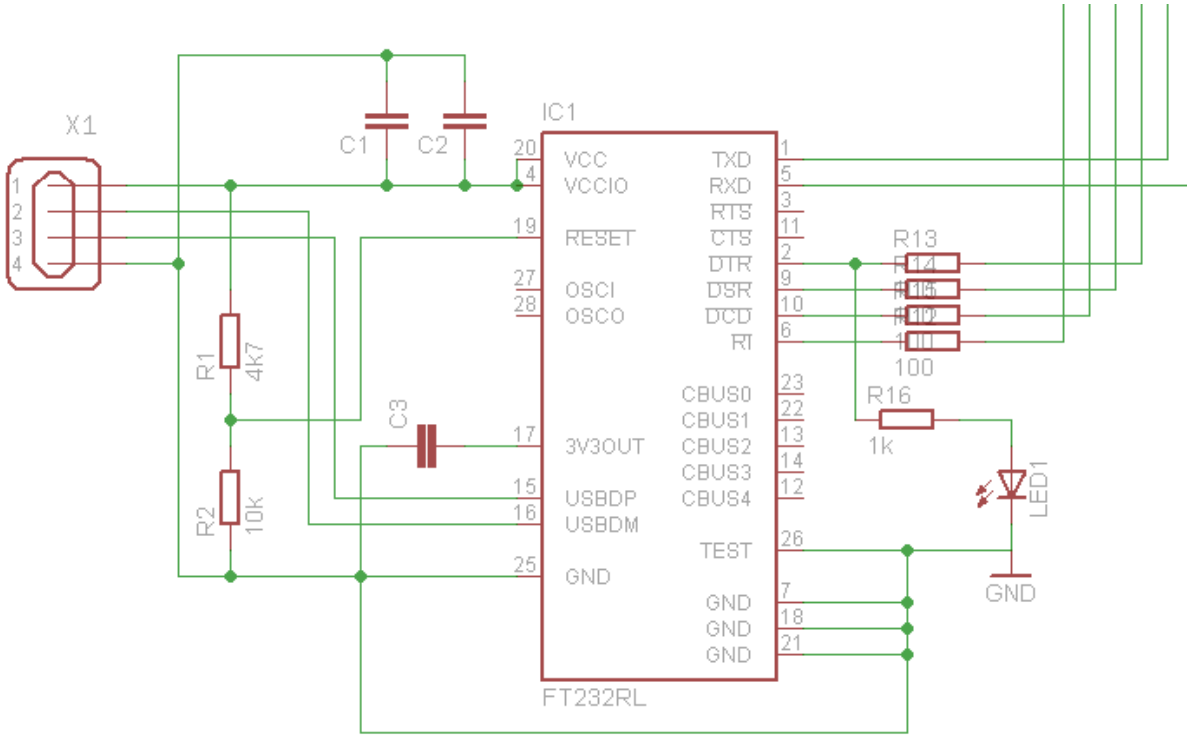
Obr. 2.1 – Blokové schéma TMP275

2.3 ROZHRANÍFT232RL

Rozhraní FT232RL slouží jako prostředník mezi USB počítače a mikroprocesorovou sběrnicí UART mikroprocesoru. Pro komunikaci je využito pinu RXD, který zprostředkovává data z PC a pin RXT přijímá data pro PC.

2.3.1 Popis zapojení FT232RL

Rozhraní FT232RL je zapojeno s drobnými změnami dle doporučeného zapojení z datasheetu. RXD A RXT slouží pro komunikaci s mikroprocesorem. Odpory R12 až R15 mají ochrannou funkci, dále vedou na SPI (programovací rozhraní mikroprocesoru). Toto zapojení umožňuje programování mikroprocesoru přes rozhraní SPI pomocí BITBANG modu obvodu FT232RL. Odpor R16 společně s LED diodou signalizují, že do mikroprocesoru jsou posílána data. Kondenzátorem C3 se filtruje napětí 3,3 V, které slouží jako napěťová úroveň pro datový přenos. Kondenzátory C1 a C2 filtrují napětí z USB. Piny USBDP a USBDM jsou datové piny USB komunikace, X1 je USB B konektor.



Obr. 2.2 – Schéma zapojení FT232RL

2.4 DS1307

Napájecí napětí obvodu DS1307 je 5 V. Pokud napětí klesne pod 3 V, je čip napájen ze záložní baterie. Tento obvod má velmi nízký odběr, při připojení na záložní baterii čerpá pouze 500 nA. Činnost obvodu je řízena externím krystalem pracujícím na kmitočtu 32,768 kHz.

Popis jednotlivých pinů

GND –uzemnění

X1,X2 –piny pro připojení oscilátoru

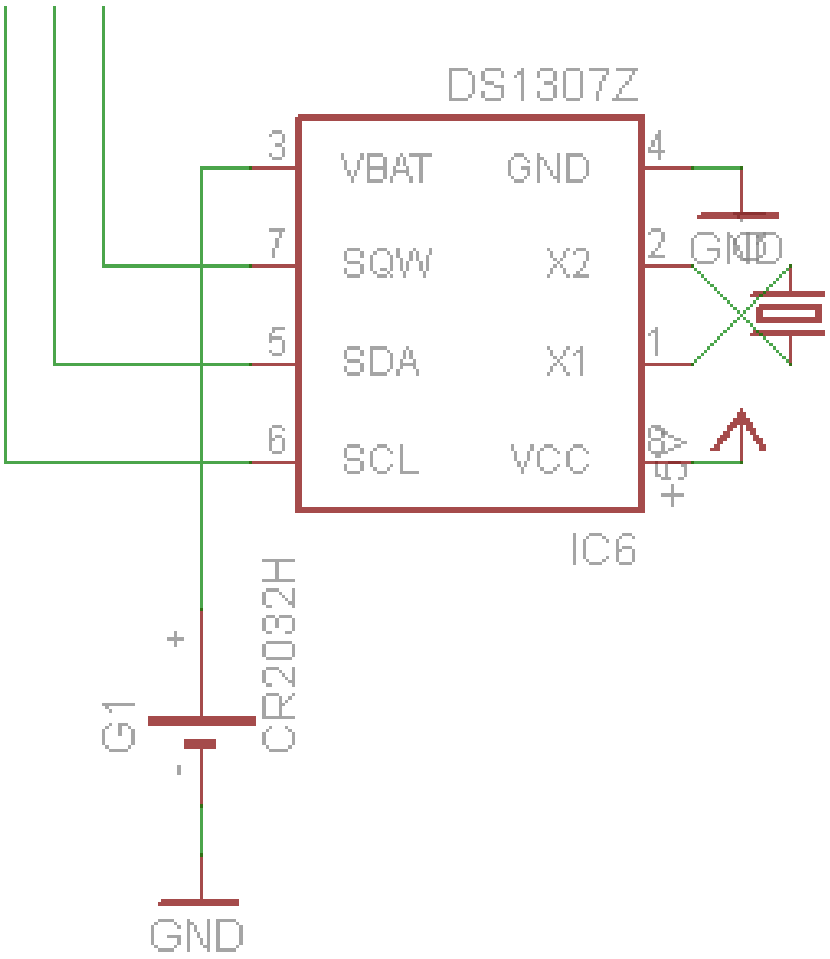
VCC –primární napájení

VBAT –pin pro připojení záložní baterie

SQW / OUT – Square Wave/Output Driver

SDA –sériová data

SCL –serialclock



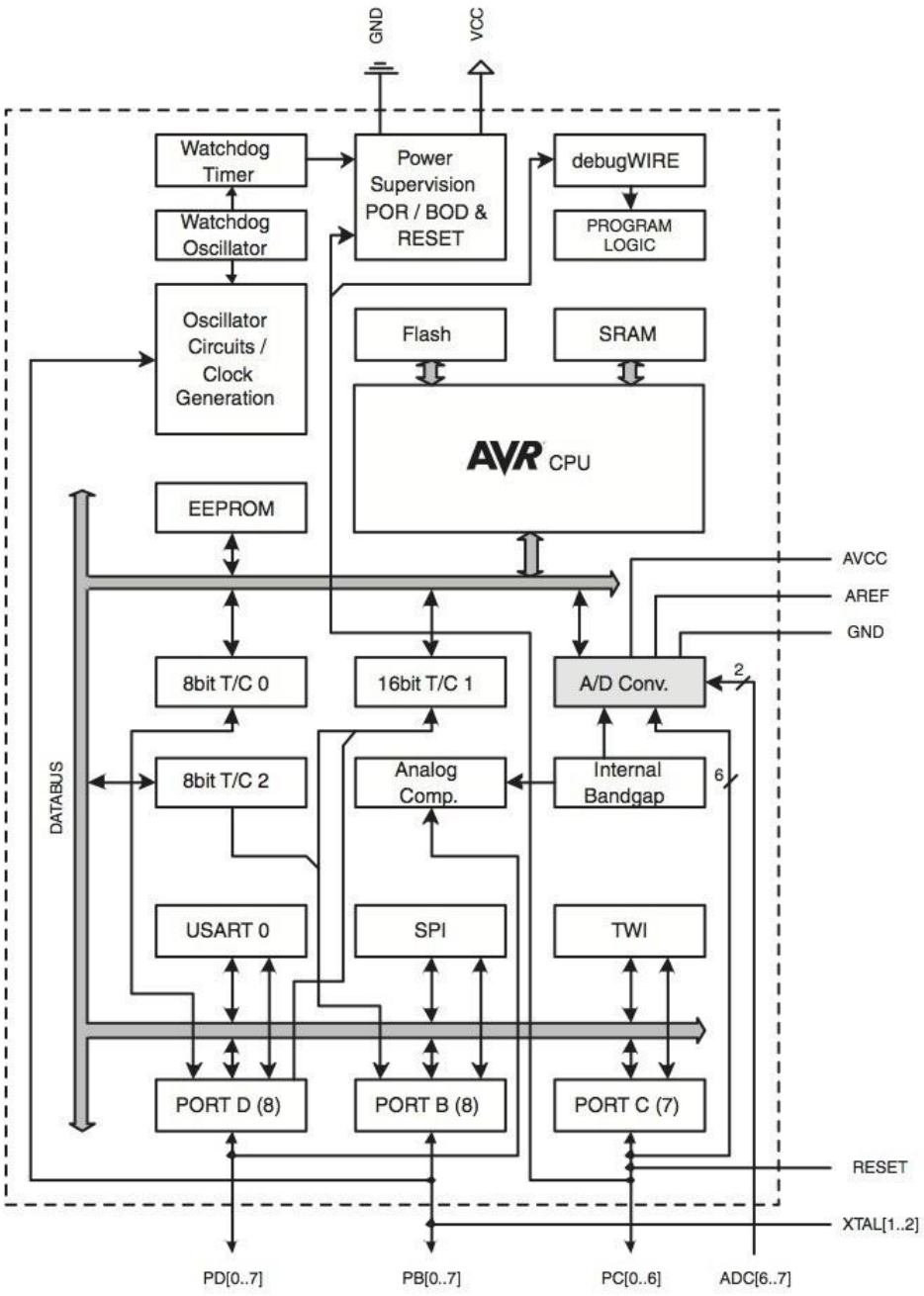
Obr. 2.3 – Schéma zapojení DS1307

3 HARDWAROVÁ REALIZACE

Tato kapitola zahrnuje výběr procesoru, návrh a popis zdroje, schémata zapojení popis regulačního zařízení, které obstarává regulaci a nově navržené termostatické hlavice, která funguje jako akční člen, tím že reguluje množství teplé vody procházející otopným zařízením, tedy utlumení nebo naopak otevření termostatického ventilu.

3.1 VOLBA MIKROPROCESORU

Mikroprocesor je řídicí jednotkou celého přípravku. Řídí regulaci kovového ventilu. Při sestavování jsem vybral typ AVR ATmega48. Je to 8bitový mikroprocesor typu RISC. Tento mikroprocesor disponuje dvěma sériovými rozhraními UART. Je vhodný pro komunikaci přes rozhraní RS232.



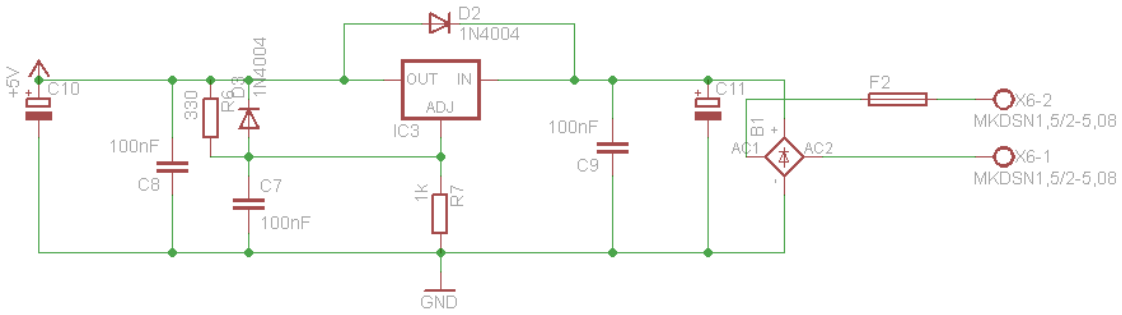
Obr. 3.1 – Blokové schéma procesoru ATmega48

3.2 NAPAJECÍ ZDROJ

Protože napájení z USB počítače je pouze 5 V a odebíraný proud je maximálně 500 mA, je v přípravku využito napájecího zdroje s lineárním stabilizátorem LM317, který je schopen dodávat až 1,5 A.

3.2.1 Popis schématu zapojení napájecího zdroje

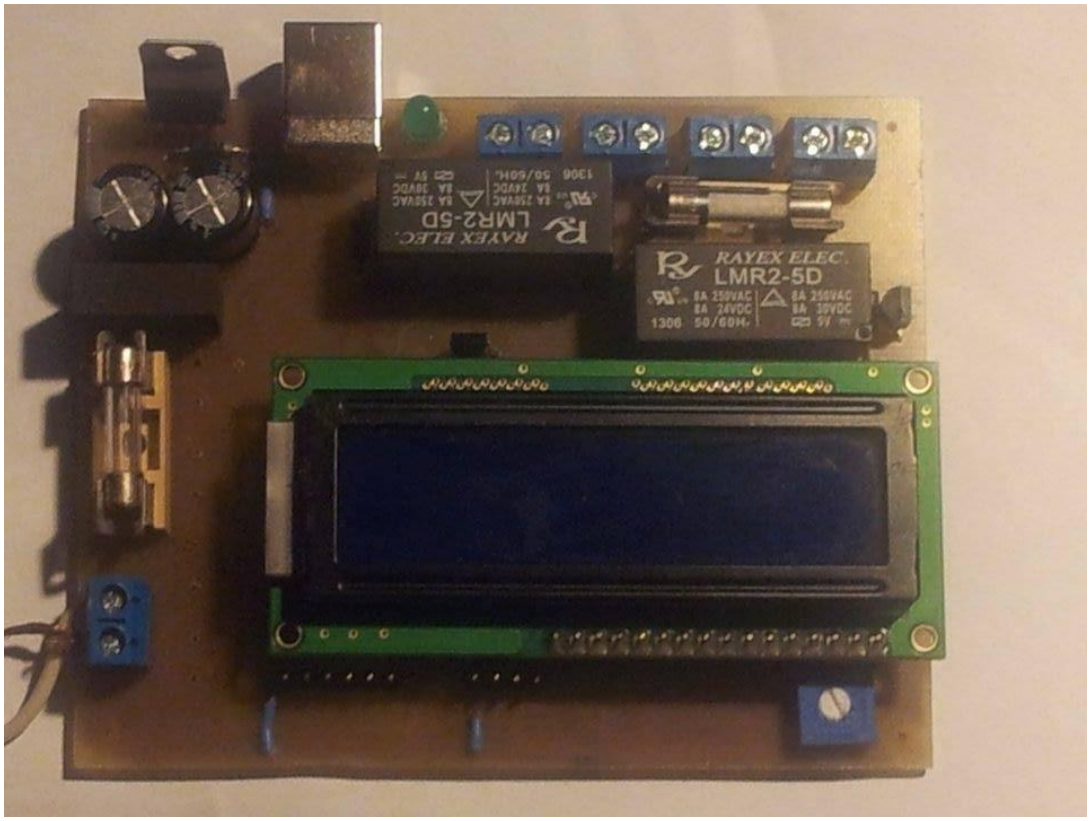
Pojistka F2 chrání zdroj před přetížením. Na něj je napojen usměrňovací můstek, který usměrňuje střídavé napětí na stejnosměrné. Následují kondenzátory C11 a C9, které filtrují napětí. Dioda D2 chrání LM317 před větší hodnotou napětí na výstupu než na vstupu. LM317 zajišťuje stabilizaci napětí. Odpory R6 a R7 nastavují výstupní napětí. D3 zabraňuje, aby napětí na pinu ADJ nebylo vyšší než na výstupu. Kondenzátor C7 blokuje referenční napětí stabilizátoru. C8 a C10 filtrují výstupní napětí.



Obr. 3.2 – Schéma zapojení zdroje

3.3 REGULAČNÍ ZAŘÍZENÍ

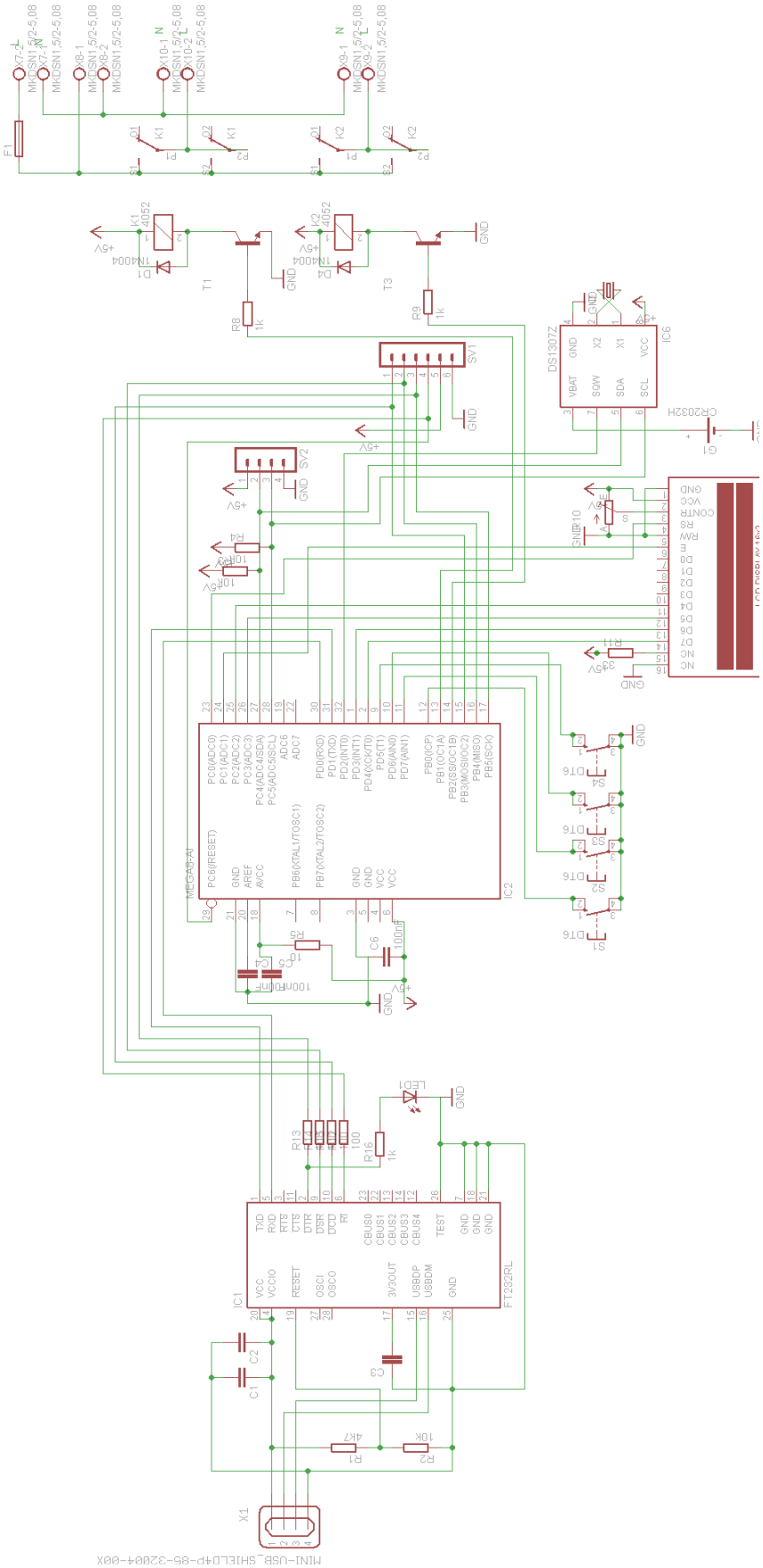
Regulační zařízení je nejdůležitější částí celého regulačního systému. Při jeho vytvoření jsem měl nejvíce problémů od malých závad až po zkrat Graetzova můstku, zničil se mi i mikroprocesor ATmega48, ale i přes tyto problémy se mi podařilo sestavit funkční zařízení, které reguluje dle mých počátečních představ. Toto zařízení obstarává veškerou komunikaci s PCi novou termostatickou hlavicí. Je vybavena displejem pro zobrazení času a aktuální teploty v místnosti. Zařízení také obstarává veškerou regulaci podle nastavených předem definovaných požadovaných hodnot. Obsahuje USB konektor pro komunikaci s PC, displej pro zobrazení aktuální teploty v místnosti, času, polohy ventilu a rozsah otáčení motorku, trimer sloužící pro nastavení podsvícení displeje, diodu pro signalizaci zapnutí a jako kontrolku komunikace při nahrávání dat z PC do zařízení. Zařízení je také vybaveno dvěma pojistkami, které chrání důležité častí zařízení před zničením. Dále relé, které při překročení předem stanovené hodnoty teploty v místnosti sepne zásuvku, do které může být zapojené například externí topení a po něj lze zvýšit tepotu v prostoru na požadovanou.



Obr. 3.3 – Fotografie regulačního zařízení

3.3.1 Popis zapojení regulačního zařízení

Relé K1 a K2 spínají zásuvku dle zadaných kritérii v programu. Pojistka F1 zajišťuje, aby relé K1 a K2 nespínali příliš velký proud.R8 a R9 jsou omezovací odpory.Tranzistory T1 aT3 spínají relé K1 a K2. Diody D1 a D4 chrání tranzistory T1 a T3 před zápornou napěťovou špičkou, kterou vytváří cívka při rozepnutí tranzistoru. SV1 je SPI programovací rozhraní pro programování zařízení. SV2 se používá pro I^2C rozraní pro komunikaci. R3 a R4 jsou „pullup“ rezistory komunikace přes rozhraní I^2C. Na tuto sběrnici je připojen i obvod IC6, který obstarává funkci zálohovaných hodin reálného času. Displej DIS1 je připojen k mikroprocesoru. Pin R/W je uzemněn, proto lze do displeje pouze zapisovat. Trimer R10 nastavuje kontrast displeje, R11 zase omezovací odpor pro podsvícení displeje. Tlačítka S1 až S4 jsou zapojena pro budoucí možné uplatnění. C6 je blokovací kondenzátor, jeho zapojení je doporučeno dle datasheetu. Umísťuje se co nejblíže mikroprocesoru. Kombinace R5 a C5 tvoří dolní propusť a filtrují napájení AD převodníku mikroprocesoru. C4 je filtrační kondenzátor referenčního napětí AD převodníku. Za ním je připojeno rozhraní FT232RL, které je popsáno výše.



Obr. 3.4 – Schéma zapojení regulačního zařízení

3.4 ZAŘÍZENÍ TERMOHLAVICE

Zařízení termohlavice slouží k regulaci množství teplé vody topení, kdy za pomocí motorku a h-můstku hlavice ovládá termostatický ventil podle požadovaných hodnot, uzavírá nebo naopak otevírá ventil topného tělesa.

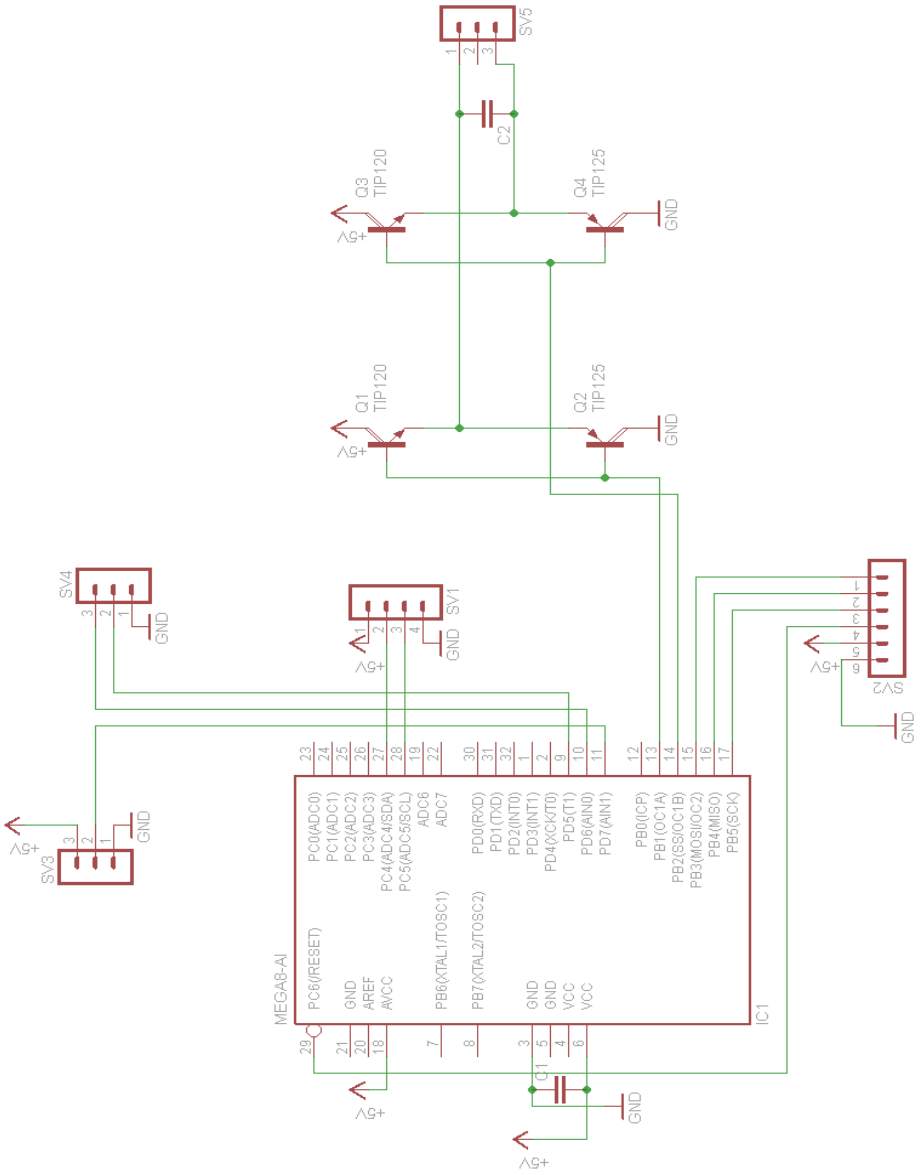
3.4.1 Popis zařízení termohlavice

Zařízení bylo vytvořeno z klasické termostatické hlavice, kdy po odříznutí většiny částí zbyla pouze matice pro našroubování na klasický termostatický ventil a malá plastová část. Do této plastové časti, byl vyvrtán závit velikosti M8. Na tento závit je přišroubován šroub stejné velikosti (M8), který obsahuje kolečko s dvanácti dírkami, jež zkoumá polohu otevření ventilu pomocí optické závory. Šroubek je napevno připojen k motorku. Motorek je pak pomocí h-můstku otáčen na obě strany. Do plastové části byly také připevněny dvě tyčky, po kterých se motorek pohybuje. K motorku byly přidány dva hliníkové pláty, které s plastovými trubičkami upevňují pohyblivou část zařízení.

Obr. 3.5 – Fotografie zařízení termohlavice

3.4.2 Popis schématu zapojení termohlavice

Na pinové lišty SV5 je připojen stejnosměrný DC motor pro řízení ventilu. Kondenzátor C2 omezuje špičky vznikající při běhu motorku. Tranzistory Q1 až Q4 zajišťují funkci h-můstku, který mění a otáčí motor. K SV1 je připojeno zařízení obstarávající regulaci. Komunikace je zajištěna sběrnicí I^2C. Na SV4 jsou připojeny koncové spínače, které chrání ventil před přetáčením. LED dioda a fototranzistor snímající polohu uzavření ventilu jsou připojeny na pinové lišty SV3. C1 je blokovací kondenzátor napájení. Procesor ATmega48 řídí činnost ventilu a komunikuje mikroprocesorem v regulačním zařízení.



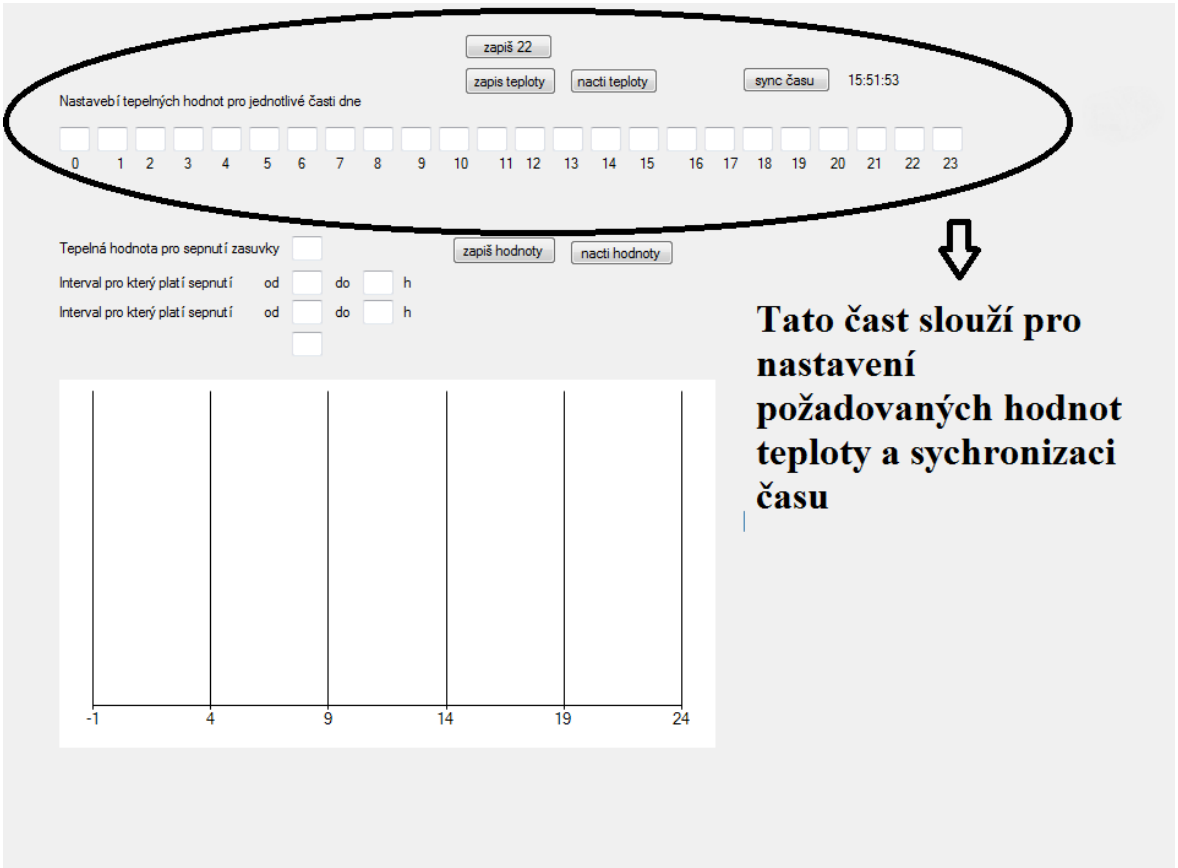
Obr. 3.6 – Schéma zapojení termohlavice

4 GRAFICKÝ SOFTWARE

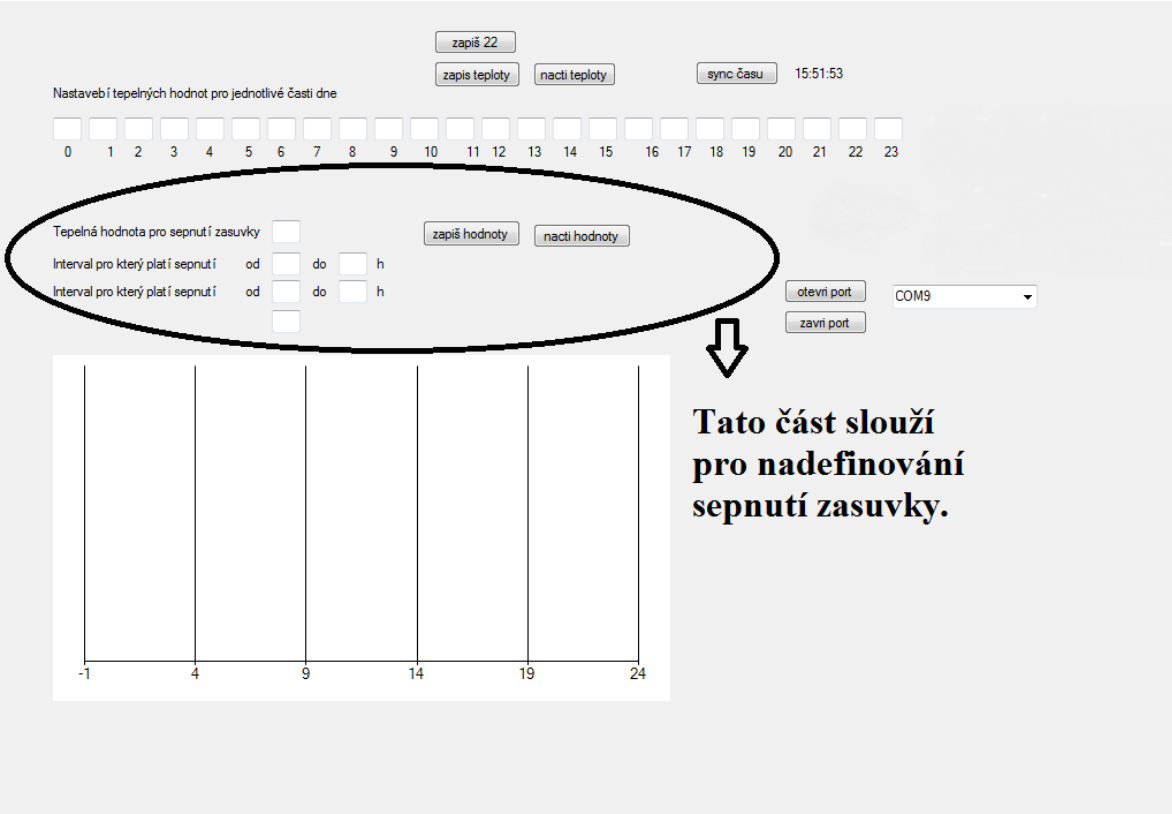
Grafický software slouží pro uživatelské nastavení zařízení. Umožňuje zadat požadované teploty v místnosti, zobrazit je v grafu a vypsat a jejich grafické zobrazení. Umožňuje také vypsání požadované teploty zapsané v paměti EEPROM.

4.1 POPIS GRAFICKÉ ČÁSTI

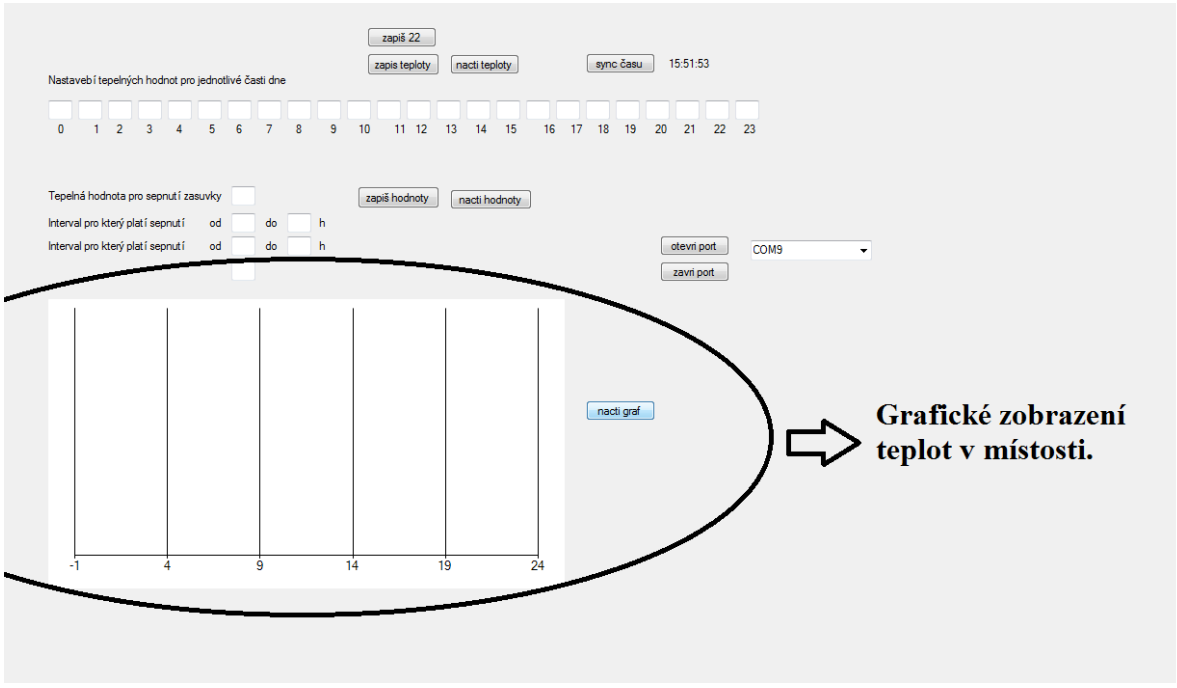
Grafické rozhraní bylo vytvořeno pomocí programu JavaScript. V pravém rohu je 24 textových polí sloužících nastavení požadovaných hodnot teploty pro každou hodinu dne. Tlačítko**zapiš teploty**zapisuje hodnoty do pamětí EEPROM. Tlačítkem **načti hodnoty** se načtou hodnoty uložené v paměti procesoru. Tlačítko **sync času** synchronizuje čas s aktuální hodnotou na ploše počítače. Dále se v rozhraní nacházejí textová pole pro nastavení spínaní zásuvky. Tlačítka **otevři port** a **zavři port** slouží k výběru portu pro komunikaci regulačního zařízení s PC. V dolní části uživatel vidí graf zobrazující průměrné teploty v průběhu 24 hodin.



Obr. 4.1 – Nastavení požadovaných hodnot



Obr. 4.2 – Definování sepnutí zásuvky



Obr. 4.3 – Grafické zobrazení

ZÁVĚR

Bylo vytvořeno regulační zařízení, které reguluje teplotu dle zadaných kritérií. Při vhodných příležitostech spíná relé zásuvku, na které může být připojeno externí topení. Nová termostatická hlavice pak pomoci h-můstku a motorku pracuje s nastavením termostatického ventilu. Při stavbě se vyskytlo několik problémů (např. se zapojením sběrnice I^2C mezi obě zařízení nebo při snímání polohy ventilu optickou závorou). Také zkratoval Graetzův můstek a následně se zničil obvod ATmega48. I přes všechny tyto problémy se podařilo vytvořit obě zařízení funkční dle výchozích představ. Už teď je jasné, že zařízení bude potřeba ještě upravovat a vylepšovat (například přidáním tlačítek pro ovládání displeje).

Literatura

8-bitAtmelMicrocontrollerwith 4/8/16K Bytes In-SystemProgrammableFlash. *Atmel* [online]. Atmel, 2011 [cit. 2015-08-28]. Dostupné z: http://www.atmel.com/images/doc2545.pdf

BABČANÍK, J.*Začínáme s mikroprocesory Atmel AVR* [online].2006 [cit. 2014-05-10]. Dostupné z: http://www.hw.cz/teorie-a-praxe/zaciname-s-mikroprocesory-atmel-avr.html

FT232R USB UART IC. *Future Technology Devices International* [online]. FTDI, 2010 [cit. 2015-08-28]. Dostupné z: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

MATOUŠEK, D.Programování a aplikace mikrořadiče ATimy2314.*Amatérské rádio*.Praha: Magnet-Press, VI/2007, s. 3–40. ISSN 0322-9572.

MATOUŠEK, D. Programování a aplikace mikrořadičeATimy2314 (2.díl).*Amatérské rádio*.Praha: Magnet-Press, VII/2007, s. 3–40. ISSN 0322-9572.

MATZ, V. Využití termostatických ventilů a termostatických hlavic pro regulaci vytápění. TZB-info [online]. 2009 [cit. 2015-08-28]. ISSN 1801-4399. Dostupné z: http://vytapeni.tzb-info.cz/mereni-a-regulace/5917-vyuziti-termostatickych-ventilu-a-termostatickych-hlavic-pro-regulaci-vytapeni

DS1307 64 x 8, Serial, I2C Real-TimeClock. *Maxim Integrated* [online]. Maxim Integrated, 2015 [cit. 2015-08-28]. Dostupné z: http://datasheets.maximintegrated.com/en/ds/DS1307.pdf

Motor controller: H-Bridge. *Botskool* [online]. 2009 [cit. 2015-08-28]. Dostupné z: http://www.botskool.com/tutorials/electronics/general-electronics/motor-controller-h-bridge

OLEJÁR, M. Stručný popis sběrnice I^2C a její praktické využití k připojení externí eeprom 24LC256 k mikrokontroléru PIC16F877. *Hw.cz* [online].2000 [cit. 2014-05-05]. Dostupné z: http://www.hw.cz/navrh-obvodu/strucny-popis-sbernice-i2c-a-jeji-prakticke-vyuziti-k-pripojeni-externi-eeprom-24lc256

OLMR, V. HW server představuje - Sériová linka RS-232.*Hw.cz* [online].2005 [cit. 2014-05-05]. Dostupné z: http://www.hw.cz/rozhrani/hw-server-predstavuje-seriova-linka-rs-232.html

TA HYDRONICS. Regulace podlahového, stropního a stěnového vytápění a chlazení (I). *TZB-info* [online]. 2006, 13.12.2006 [cit. 2015-08-28]. ISSN 1801-4399. Dostupné z: http://www.tzb-info.cz/3758-regulace-podlahoveho-stropniho-a-stenoveho-vytapeni-a-chlazeni-i

0.5°C Digital OutTemperature Sensor: TMP275. *Texas Instruments* [online]. Texas Instruments, 2006 [cit. 2015-08-28]. Dostupné z: http://www.ti.com/lit/ds/sbos363d/sbos363d.pdf

TŮMA, O. Arduino - pulsně šířková modulace (PWM) v C(++). Z*eropage.cz*[online]. 2011[cit. 201405-19]. Dostupné z: http://zeropage.cz/article/arduino_-_pulsne_sirkova_modulace_pwm_v_c

Přílohy

A – Program ventilu topného tělesa

B – Program regulačního zařízení

C – CD s programem

Příloha A

Příloha k bakalářské práci

Řízení teploty v místnosti pomocí elektrické regulace ventilu topného tělesa

Marek Lochman

PROGRAM VENTILU TOPNÉHO TĚLESA

```
#include <stdint.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/eeprom.h>
#include <avr/pgmspace.h>

#define XTAL 8000000                    // frekvence procesoru je 8MHz

        // globální proměnné
volatile static uint8_t aktualniPozice = 10;        // proměnná s aktuální pozicí
volatile static uint8_t pozice = 14;                // proměnná s požadovanou pozicí
volatile static uint8_t pravyDoraz = 0;    // koncový spínač
volatile static uint8_t levyDoraz = 0;              // druhý koncový spínač
volatile static uint8_t tocitDoprava = 0; // požadovaný směr doprava
volatile static uint8_t tocitDoleva = 0;    // požadovaný směr doleva
volatile static uint8_t stavVstupu Pozice = 0; // předchozí hodnota na fototranzistoru
volatile static uint8_t kalib = 0;          // kalibrace

                                            //prototypové funkce
void EEPromRead(void);
void EEPromWrite(void);
void init(void);
void init_I2C(void);
void zjistiDorazy(void);
void otocNaPozici(uint8_t);
void otoc(void);
void zjistiPozici(void);
void nastavKrajniPoziciPriDorazu(void);
void vypisPozici(uint8_t);
void kalibrace(void);
```

```
ISR(TIMER0_OVF_vect)                    // přerušení z přetečení
{

        zjistiDorazy();
        otocNaPozici(pozice);
        otoc();
        zjistiPozici();
        nastavKrajniPoziciPriDorazu();
}
```

Main cyklus

Nekonečný cyklus main obsahuje funkci init(), pro nastavení portu, čítače a předěličky. Funkci init_I2C(), která obsahuje adresu ventilu. Funkci kalibrace () pro nastavení dorazů. Zapnutí přerušení sei()a instrukce pro komunikaci I2C.

```
int main(void)
{
        init();
        init_I2C();
        kalibrace();
        sei();
        while(1)
        {
                while(((TWSR & 0xF8)!= 0x60))
                {

                        TWCR=(1<<TWEA)|(1<<TWEN)|(1<<TWINT);
                        while ((!(TWCR & (1<<TWINT))));
                }

                {
                        TWCR= (1<<TWINT)|(1<<TWEA)|(1<<TWEN);
                        while ((!(TWCR & (1<<TWINT)))){
                        }
                        while(((TWSR & 0xF8)!=0x80)){
```

```
                    }
                    pozice=TWDR;
                }
                While (((TWSR & 0xF8)!= 0xA8))
        {
                TWCR=(1<<TWEA)|(1<<TWEN)|(1<<TWINT);
                while ((!(TWCR & (1<<TWINT))))
                    {
       }
        }
                {
                    TWDR= aktualniPozice;
                    TWCR= (1<<TWEN)|(1<<TWINT);
                    while(((TWSR & 0xF8) != 0xC0)){
                    } /
    }
                        _delay_ms(50);
        }
}
```

Tato část slouží k inicializaci jednotlivých portů, nastavení čítače, předěličky a čtení pozice ventilu z eepromky.

```
void init()
{
    DDRD =  0b00000000;          // port D je vstupní
    PORTD = 0b11100000;          // zapíná pull up rezistory na  PD5, PD6,PD7
    TIMSK0 = 0b00000001;         // zapnutí přerušení při přetečení čítače TC0
    TCCR0B = 0b00000100;         // předělička 256
    EEPromRead();                // načte pozici ventilu z eepromky
    DDRB = 0b00000110;           // nastavení pinu Pb1 a Pb2 jako výstupní
    PORTB = 0b00000000;          // nastavení Pb1 a Pb2 na logickou nulu
```

Tato část slouží pro nastavení adresy ventilu pro i2c komunikaci.

```
void init_I2C()
{
        TWAR=0x20;                  // i2c adresa ventilu
        TWCR = 1;                   // podřízený slave
}
```

Tento kód je asi nejdůležitějším z programu ventilu. Obsahuje podmínku kalibrace, funkce na zjištění dorazu, otáčení na pozici, otoč pro zapnutí h-můstku, zjištění pozice pro ovládaní optické závory.

```
void kalibrace()
{
        if(kalib == 1)              // podmínka pro kalibraci
        {
                while(levyDoraz == 0)           // dokud levý doraz je nula
                {
                zjistiDorazy();   // funkce zjišťující dorazy
                otocNaPozici(0);      // nastaví točit doleva, nebo točit doprava
                                                podle požadované pozice
                otoc();           // funkce pro zapnutí h-můstku
                zjistiPozici();     // funkce pro zjištění pozice natočení
                nastavKrajniPoziciPriDorazu();  // funkce pro reset proměnné
                                        aktuální pozice při dorazu
                _delay_ms(50); // funkce čekaní 50ms
                }
        }
}


void EEPromRead(void)               // přečtení konstant z eepromky
{
aktualniPozice = eeprom_read_byte((uint8_t*)0); //eeprom přečte aktuální pozice
        if(aktualniPozice > 40)     // pokud je větší jak 40 spusť kalibraci
        {
                kalib = 1;          // kalibruj
```

```
        }
    }

    void EEPromWrite(void)
    //Přečtení konstant z EEPROMky
    {
        eeprom_write_byte((uint8_t*)0,aktualniPozice);
    }
    Část kodu pro
    void zjistiDorazy()
    {
        if((PIND & 0b10000000) ==0)      // pokud Pd7 rovná se nule (zapne se koncový spínač)
        {
            pravyDoraz = 1;
        }
        else
        {
            pravyDoraz = 0;
        }

        if((PIND & 0b01000000) == 0)
        {
            levyDoraz = 1;
        }
        else
        {
            levyDoraz = 0;
        }
    }

    void otocNaPozici(uint8_t p)
    {
        if(p > aktualniPozice) // pokud vstupní hodnota je větší než aktuální
        {
            tocitDoprava = 1;     // toč doprava
            tocitDoleva = 0;      // netočit doleva
        }
        else if(p < aktualniPozice)    // pokud je vstupní hodnota menší než aktuální
        {
            tocitDoleva = 1;      // toč doleva
            tocitDoprava = 0;     // netoč doprava
        }
        else if(p == aktualniPozice)// vstupní hodnota stejná s aktuální netoč
        {
            tocitDoleva = 0;      // netočit doleva
            tocitDoprava = 0;     // netočit doprava
        }
    }
```

```
void otoc()
{
	if((pravyDoraz == 0)&&(tocitDoprava == 1))
	{
		DDRB |= 0b00000110;	// nastav jako výstupní port Pb1 a Pb2
		PORTB &= ~(0b00000110); // nastav na pozici Pb1 a Pb2 nuly
		PORTB |= 0b00000100;		// nastav na pozici Pb2 jedničku
	}
	else if((levyDoraz == 0)&&(tocitDoleva == 1))
	{

		DDRB |= 0b00000110;		// nastav jako výstupní port Pb1
a Pb2
		PORTB &= ~(0b00000110); // nastaví na pozici Pb1 a Pb2 nuly
		PORTB |= 0b00000010;		// nastav na pozici Pb1 jedničku
	}

	else
	{
		DDRB &= ~(0b00000110);
		PORTB &= ~(0b00000110);
	}
}

void zjistiPozici()
{
	if(tocitDoprava == 1)		// pokud se točí doprava
	{
if(((PIND & 0b00100000) != stavVstupuPozice) && (PIND & 0b00100000)==0)

// pokud stav Pd5 nerovná se stavVstupuPozice a zaroveň staav Pd5 rouná se nula.
		{

			aktualniPozice++;	// Přičti jedničku do aktuální Pozice
			EEPromWrite();	// zapiš aktuální pozici do eepromky
		}
	}
	if(tocitDoleva == 1)		// pokud se točí doleva
	{
if(((PIND & 0b00100000) != stavVstupuPozice) && (PIND & 0b00100000)==0)
		{
			if(aktualniPozice>0)  //
			aktualniPozice--;	// odečti jedničku od aktuální pozice
			EEPromWrite();	// zapiš aktuální pozici do eepromky

		}
	}
	stavVstupuPozice = ((PIND & 0b00100000));
}
```

```
void nastavKrajniPoziciPriDorazu() // funkce pro zjištění krajních dorazů
{

	if((levyDoraz == 1))		//pokud je na dorazu nastaví pozici na 0
	{
		aktualniPozice = 0;
	}
	if((pravyDoraz == 1))		//pokud je na druhém dorazu nastav na 40
	{
		aktualniPozice = 40;
	}
}
```

Příloha B

Příloha k bakalářské práci

Řízení teploty v místnosti pomocí elektrické regulace ventilu topného tělesa

Marek Lochman

PROGRAM REGULAČNÍHO ZAŘÍZENÍ

```
#include <stdint.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
#include "twi_lib.c"
#include "lcd.c"

#define Ventil1 0x20       //adresa ventilu je 20
#define Teplomer1 0b10011110  // je zvolen 12 bitový rámec
#define RTC 0b11010000     // adresa hodin
#define F_CPU 8000000ul    // adresa procesoru
#define USART_BAUDRATE 9600
#define UBRR_VALUE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)

 // definování statických proměnných
static uint8_t hod, min, sec, hodp;
static char UARTbuffer[10];
static int8_t teploty[24];
static int8_t teplrele1 = 0;
static uint8_t casrele1od = 0;
static uint8_t casrele1do = 0;
static int8_t teplrele2 = 0;
static uint8_t casrele2od = 0;
static uint8_t casrele2do = 0;
static uint16_t aktualniteplota;
static uint8_t ventil = 0;
static uint8_t ventila = 0;

 ISR(USART_RX_vect)
{
```

```
    posunbuffer();
   UARTbuffer[0]=UDR0;
    vetveni();
}
void eepromReadData()    // funkce pro čtení dat z eeprom
{
    for(uint8_t i = 0; i<24 ; i++)    // cyklus pro 24 hodnot telopty
    {
        teploty[i] = eeprom_read_byte((uint8_t*)i);
    }
    teplrele1 = eeprom_read_byte((uint8_t*)24);
    casrele1od = eeprom_read_byte((uint8_t*)25);
    casrele1do = eeprom_read_byte((uint8_t*)26);
    teplrele2 = eeprom_read_byte((uint8_t*)27);
    casrele2od = eeprom_read_byte((uint8_t*)28);
    casrele2do = eeprom_read_byte((uint8_t*)29);
}
uint8_t zaznTepl(uint8_t hour)
{
    return eeprom_read_byte((uint8_t*)hour+30);
}
void eepromWriteData()  funkce pro zápis teplot
{
    for(uint8_t i = 0; i<24; i++)
    {
        eeprom_update_byte (( uint8_t *) i, teploty[i] );
    }
    eeprom_update_byte (( uint8_t *) 24, teplrele1 );
    eeprom_update_byte (( uint8_t *) 25, casrele1od );
    eeprom_update_byte (( uint8_t *) 26, casrele1do );
    eeprom_update_byte (( uint8_t *) 27, teplrele2 );

    eeprom_update_byte (( uint8_t *) 28, casrele2od );
    eeprom_update_byte (( uint8_t *) 29, casrele2do );
```

```
}
void vetveni()      // funkce větvení
{
    if((UARTbuffer[2]=='g')&&(UARTbuffer[1]=='t')&&(UARTbuffer[0]=='1'))
    {
        poslibyteUSART('a');
        poslibyteUSART('h');
        poslibyteUSART('o');
        poslibyteUSART('j');
    }
    if((UARTbuffer[8]=='s')&&(UARTbuffer[7]=='t'))
    {
        if((UARTbuffer[6]=='m'))
        {
            nastavcas(((UARTbuffer[5]-48)*10+(UARTbuffer[4]-48)),((UARTbuffer[3]-
48)*10+(UARTbuffer[2]-48)),((UARTbuffer[1]-48)*10+(UARTbuffer[0]-48)));   //   nástavní
času
        }
        for(uint8_t i = 0; i<24; i++)
        {
            if((UARTbuffer[6]==(i/10+48))&&(UARTbuffer[5]==(i%10+48)))
            {
                teploty[i] = ((UARTbuffer[4]-48)*10+(UARTbuffer[3]-48));
                    poslibyteUSART(teploty[i]/10+48);
                    poslibyteUSART(teploty[i]%10+48);
            }
        }
        eepromWriteData(); // funkce zápisu dat
    }



    if((UARTbuffer[5]=='s')&&(UARTbuffer[4]=='r'))
    {
```

```
    if(UARTbuffer[3]=='t')
    {
        if((UARTbuffer[2]-48) == 1)            {
            teplrele1 = (UARTbuffer[1]-48)*10+(UARTbuffer[0]-48);
             }
        if((UARTbuffer[2]-48) == 2)
        {
            teplrele2 = (UARTbuffer[1]-48)*10+(UARTbuffer[0]-48);
                }
    }
    if(UARTbuffer[3]=='o')
    {
        if((UARTbuffer[2]-48) == 1)
        {
                    casrele1od = (UARTbuffer[1]-48)*10+(UARTbuffer[0]-48);
       }
        if((UARTbuffer[2]-48) == 2)
        {
            casrele2od = (UARTbuffer[1]-48)*10+(UARTbuffer[0]-48);
        }
    }
    if(UARTbuffer[3]=='d')
    {
        if((UARTbuffer[2]-48) == 1)
        {
            casrele1do = (UARTbuffer[1]-48)*10+(UARTbuffer[0]-48)
        }
        if((UARTbuffer[2]-48) == 2)
        {
            casrele2do = (UARTbuffer[1]-48)*10+(UARTbuffer[0]-48);
        }
    }
    eepromWriteData();
}
```

```
    if((UARTbuffer[4]=='a')&&(UARTbuffer[3]=='f')&&(UARTbuffer[2]=='t'))
          {
        for(uint8_t i = 0; i<24; i++)
        {
            if((UARTbuffer[1]==(i/10+48))&&(UARTbuffer[0]==(i%10+48)))
           {
                poslibyteUSART(teploty[i]/10+48);
                poslibyteUSART(teploty[i]%10+48);
            }
        }
    }
    if((UARTbuffer[3]=='a')&&(UARTbuffer[2]=='f')&&(UARTbuffer[1]=='r'))      {
        if((UARTbuffer[0]-48) == 1)
        {
            poslibyteUSART(teplrele1/10+48);
            poslibyteUSART(teplrele1%10+48);
        }
        if((UARTbuffer[0]-48) == 2) // nevím
        {
            poslibyteUSART(teplrele2/10+48);
            poslibyteUSART(teplrele2%10+48);
}
    }
    if((UARTbuffer[3]=='a')&&(UARTbuffer[2]=='f')&&(UARTbuffer[1]=='o'))
    {
        if((UARTbuffer[0]-48) == 1)
        {
            poslibyteUSART(casrele1od/10+48);
            poslibyteUSART(casrele1od%10+48);
        }

        if((UARTbuffer[0]-48) == 2)

        {
```

```
                poslibyteUSART(casrele2od/10+48);
                poslibyteUSART(casrele2od%10+48);
            }
        }
        if((UARTbuffer[3]=='a')&&(UARTbuffer[2]=='f')&&(UARTbuffer[1]=='d'))
        {
            if((UARTbuffer[0]-48) == 1)
            {
                poslibyteUSART(casrele1do/10+48);
                poslibyteUSART(casrele1do%10+48);
            }
            if((UARTbuffer[0]-48) == 2)
            {
                poslibyteUSART(casrele2do/10+48);
                poslibyteUSART(casrele2do%10+48); // nevm
            }
        }
        if((UARTbuffer[4]=='a')&&(UARTbuffer[3]=='f')&&(UARTbuffer[2]=='l'))
        {
            uint8_t d = zaznTepl((UARTbuffer[1]-48)*10+(UARTbuffer[0]-48));
           poslibyteUSART(d/10+48);
            poslibyteUSART(d%10+48);
                                }
    }
void posunbuffer()// nevím
{
    for(uint8_t i = 9; i>=1 ; i--) // nevím
    {
        UARTbuffer[i] = UARTbuffer[i-1]; // nevím
    }
}
Void inicializace() // Inicializace
{
    eepromReadData(); // čtení dat s eepromky
```

```
    DDRB = 0b00000110;
    TWSR = 0; /* no prescaler */
    TWBR = 32; /* must be > 10 for stable operation */
    TWCR=(0<<TWINT|1<<TWEA|1<<TWEN);
        i2c_init(); // zapnutí i2c

    lcd_init(LCD_DISP_ON);    //inicializace LCD, připojen na PortC
    lcd_puts_p(PSTR("ahoj"));  //uvodni text
    sei();       // Zapnutí přerušení
}
void nastavteplomer()
{
    i2c_start_wait(Teplomer1+I2C_WRITE);
    i2c_write(0b00000001);     // nevím
    i2c_write(0b01100000);     // přenos informaci
    i2c_stop();                 // konec přenosu
}
uint16_t ctiteplotu()             //funkce pro čtení teploty
{
    uint16_t temp;
    i2c_start_wait(Teplomer1+I2C_WRITE);
    i2c_write(0);
    i2c_rep_start(Teplomer1+I2C_READ);
    temp = i2c_readAck()*16;
    temp += i2c_readAck()/16;
    temp *= 10;
    temp /= 16;
    i2c_stop();                      // konec i2c přenosu
    return temp;                      // Opakování smyčky
}
uint8_t nastavventil(uint8_t hodnota)         // funkce pro nastavení pozice
{

    uint8_t temp;
```

```
    i2c_start_wait(Ventil1+I2C_WRITE);
    i2c_write(hodnota);
    i2c_rep_start(Ventil1+I2C_READ);
    temp = i2c_readNak();
    i2c_stop();
    return temp;
}
void nastavcas(uint8_t hodiny, uint8_t minuty, uint8_t sekundy)
{
    i2c_start_wait(RTC+I2C_WRITE);
    i2c_write(0x00);
    i2c_write(((sekundy/10)*16+(sekundy%10))&0b01111111);
    i2c_write(((minuty/10)*16+(minuty%10)));
    i2c_write(((hodiny/10)*16+(hodiny%10)));
    i2c_stop(); // konec i2c komunikace
}
void nastavdatum(uint8_t roky, uint8_t mesice, uint8_t dny)
{
    i2c_start_wait(RTC+I2C_WRITE);
    i2c_write(0x04);
    i2c_write(((dny/10)*16+(dny%10)));
    i2c_write(((mesice/10)*16+(mesice%10)));
    i2c_write(((roky/10)*16+(roky%10)));
    i2c_stop();
}
void cticas() // metoda pro čtení času
{
    uint8_t temp;
    i2c_start_wait(RTC+I2C_WRITE);
    i2c_write(0x00);
    i2c_rep_start(RTC+I2C_READ);
    temp = i2c_readAck();

    sec = (temp&0b00001111) + (((temp&0b01110000)/16)*10);
```

```
    temp = i2c_readAck();
    min = (temp&0b00001111) + (((temp&0b01110000)/16)*10);
    temp = i2c_readNak();
    hod = (temp&0b00001111) + (((temp&0b01110000)/16)*10);
    i2c_stop();
     if(hodp!= hod)
    {
        eeprom_update_byte (( uint8_t *) (30+hod), aktualniteplota/10 );
        hodp = hod;
    }
}
void sepnirele1() // sepni rele
{
    PORTB |= 0b00000010;
}
void sepnirele2() // sepni rele2
{
    PORTB |= 0b00000100;
}
void vypnirele1() //vypnutí rele1
{
    PORTB &= ~(0b00000010);
}
void vypnirele2() vypnutí rele 2
{
    PORTB &= ~(0b00000100);
}
void poslibyteUSART(uint8_t u8Data)
{
    while (!(UCSR0A & (1<<UDRE0)));   // Wait if a byte is being transmitted
    UDR0 = u8Data;      // Transmit data
}
```

```
void nastavUSART()
{
    UBRR0= UBRR_VALUE;
    UCSR0B=(1<<RXCIE0)|(1<<RXEN0)|(1<<TXEN0);
    UCSR0C=(3<<UCSZ00);       }
void regulace()  // funkce pro regulaci
{
    int8_t rozdil = aktualniteplota-teploty[hod]*10;
    if(rozdil > 10)
    {
        ventil = 0;
    }
    else if(rozdil < -10)
    {
        ventil = 40;
    }
    Else
    {
        ventil = (-rozdil+10)*2;
    }
    if(casrele1od<casrele1do)
    {
        if((hod>=casrele1od)&&(hod<=casrele1do))
        {
            if((teplrele1*10) > aktualniteplota)
                sepnirele1();
            else
                vypnirele1();
        }
        else
                vypnirele1();
    }
```

```
else
{
    if((hod>=casrele1od)||(hod<=casrele1do))
    {
        if((teplrele1*10) > aktualniteplota)
            sepnirele1();
        else
            vypnirele1();
    }
    else
            vypnirele1();
}
if(casrele2od<casrele2do)
{
    if((hod>=casrele2od)&&(hod<=casrele2do))
    {
        if((teplrele2*10) > aktualniteplota)
            sepnirele2(); // volání funkce sepni rele2
        else // nevim
            vypnirele2(); // volání funkce vypni rele2
    }
    else
            vypnirele2();// volání funkce vypni rele2
}
else
{
    if((hod>=casrele2od)||(hod<=casrele2do))
    {
        if((teplrele2*10) > aktualniteplota)
            sepnirele2();// volání funkce sepni rele2
        else
            vypnirele2();// volání funkce vypni rele2

    }
```

```
            else

                        vypnirele2();// volání funkce vypni rele2

        }

//      poslibyteUSART(ventil/10+48);

//      poslibyteUSART(ventil%10+48);

}




int main()

{
        inicializace();
        nastavUSART();
        nastavteplomer();
//      nastavcas(0,48,0);
//      nastavdatum(14,4,21);
        uint8_t buffer[5];
        uint8_t i;
        for(;;)
        {
           cticas();
        aktualniteplota = ctiteplotu();
            regulace();
             ventila = nastavventil(ventil);
```

```
lcd_gotoxy(0,0);
itoa(aktualniteplota,buffer,10);
lcd_putc(buffer[0]);
lcd_putc(buffer[1]);
lcd_putc(buffer[2]);
lcd_putc(buffer[3]);
lcd_putc(' ');
itoa(hod,buffer,10);
if(hod>10)
{
    lcd_putc(buffer[0]);
    lcd_putc(buffer[1]);
}
else
{
    lcd_putc('0');
    lcd_putc(buffer[0]);
}
lcd_putc(':');
itoa(min,buffer,10);
if(min>10)
{
    lcd_putc(buffer[0]);
    lcd_putc(buffer[1]);
}
else
{
    lcd_putc('0');
    lcd_putc(buffer[0]);
}
lcd_putc(':');;
itoa(sec,buffer,10);
if(sec>10)
```

```
        {
            lcd_putc(buffer[0]);
            lcd_putc(buffer[1]);
        }
        else
        {
            lcd_putc('0');
            lcd_putc(buffer[0]);
        }
        lcd_putc(' ');
        lcd_gotoxy(0,1);
        itoa(ventil,buffer,10);
        lcd_putc(buffer[0]);
        lcd_putc(buffer[1]);
        lcd_putc(buffer[2]);
        lcd_putc(buffer[3]);
        itoa(ventila,buffer,10);
        lcd_putc(buffer[0]);
        lcd_putc(buffer[1]);
        lcd_putc(buffer[2]);
        lcd_putc(buffer[3]);
    /*    itoa(temp,buffer,10);
        while(buffer[i]!=0)
        {
            poslibyteUSART(buffer[i]);
            buffer[i]=0;
            i++;
        }
        poslibyteUSART(' ');*/
        i=0;
    }
}
```

Příloha C

Příloha k bakalářské práci

Řízení teploty v místnosti pomocí elektrické regulace ventilu topného tělesa

Marek Lochman

CD S PROGRAMEM

Obsah

LochmanM_ŘízeníTeploty_LK_2015.pdf
Program regulačního zařízení
Program regulačního přípravku
Grafické prostředí