SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta elektrotechniky a informatiky

Evidenčné číslo: FEI-104412-86356

Návrh ovládača pre riadenie teplotnej komory

Diplomová práca

Študijný program: elektronika a fotonika
Študijný odbor: elektrotechnika
Školiace pracovisko: Ústav elektroniky a fotoniky
Vedúci záverečnej práce: Ing. Daniel Arbet, PhD.
Konzultant: Ing. Adam Hudec

Bratislava 2021 **Bc. Rastislav Straka**

Slovenská technická univerzita v Bratislave
Ústav elektroniky a fotoniky

Fakulta elektrotechniky a informatiky
Akademický rok: 2020/2021
Evidenčné číslo: FEI-104412-86356

STU FEI

ZADANIE DIPLOMOVEJ PRÁCE

Študent:	**Bc. Rastislav Straka**
ID študenta:	86356
Študijný program:	elektronika a fotonika
Študijný odbor:	elektrotechnika
Vedúci práce:	Ing. Daniel Arbet, PhD.
Konzultant:	Ing. Adam Hudec
Miesto vypracovania:	Oddelenie návrhu a testovania IO

Názov práce: **Návrh ovládača pre riadenie teplotnej komory**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Charakterizácia vlastnosti a parametrov ASIC obvodov v danom teplotnom rozsahu prostredníctvom meraní predstavuje dôležitý krok vo fáze testovania IO. Pre charakterizáciu IO navrhnutých na Oddelení návrhu a testovania IO bol vyvinutý hardvér pre teplotnú komoru na báze Peltierovho článku. Táto práca je zameraná na implementáciu ovládača pre riadenie a reguláciu teploty v komore.

1. Oboznámte sa s princípom činnosti teplotnej komory využívajúcej Peltierov článok a jej hardvérom vyvinutým na Oddelení návrhu a testovania IO.

2. Navrhnite algoritmus v programovacom jazyku C/C++ na ovládanie a reguláciu teploty v teplotnej komore.

3. Implementujte navrhnutý algoritmus a overte jeho funkčnosť.

4. Overte funkčnosť teplotnej komory.

Zoznam odbornej literatúry:

1. KERNIGHAN, B W. – Ritchie, D M. The C Programming Language. 2nd Edition. New Jersey : Pearson, 1988. 274 s. ISBN 978-0-1311-0362-7
2. MAZIDI, M A. – CHEN, S. – GHAEMI, E. STM32 Arm Programming for Embedded Systems: Volume 6. MicroDigitalEd, 2018. 378 s. ISBN 978-0997925944

Riešenie zadania práce od: 15. 02. 2021

Dátum odovzdania práce: 14. 05. 2021

Bc. Rastislav Straka
študent

prof. Ing. Daniel Donoval, DrSc.
vedúci pracoviska

prof. Ing. Daniel Donoval, DrSc.
garant študijného programu

Čestné prehlásenie

Ja, dole podpísaný Bc. Rastislav Straka čestne vyhlasujem, že diplomovú prácu „Návrh ovládača pre riadenie teplotnej komory“ som vypracoval samostatne na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.

.....

Vlastnoručný podpis

V Zlatých Moravciach, dňa 21.5.2021

Poďakovanie

Týmto by som chcel poďakovať personálu Ústavu elektroniky a fotoniky na Fakulte elektrotechniky a informatiky Slovenskej technickej univerzity. Hlavné poďakovanie však patrí vedúcemu mojej záverečnej práce Ing. Danielovi Arbetovi, PhD. Osobné poďakovanie však patrí aj študentovi doktorandského štúdia elektroniky a fotoniky Ing. Adamovi Hudecovi.

ANOTÁCIA DIPLOMOVEJ PRÁCE

Slovenská technická univerzita v Bratislave
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný odbor:	Elektrotechnika
Študijný program:	Elektronika a Fotonika
Autor:	Bc. Rastislav Straka
Diplomová práca:	Návrh ovládača pre riadenie teplotnej komory
Vedúci práce:	Ing. Daniel Arbet, PhD.
Mesiac a rok odovzdania:	Máj 2021
Kľúčové slová:	Teplotná komora, Programovací jazyk C, PID regulátor, STM32, Termoelektrický efekt, Komunikačné rozhrania

Táto diplomová práca prezentuje postup práce a poznatky získané pri tvorbe ovládacieho programu pre teplotnú komoru. Pri práci boli použité nástroje STM32CubeMX, VSCode, GNU Embedded Toolchain, Make a OpenOCD na založenie projektu, editovanie kódu, kompiláciu/spracovanie a optimalizáciu navrhnutého algoritmu. Program napísaný v jazyku C pre mikro-kontrolér STM32F205VGT6 má za úlohu ovládať zariadenie teplotnej komory. Medzi jeho úlohy patrí ovládanie výstupov a vstupov mikro-kontroléra, obsluha jednotlivých periférií a výpočty riadiacich premenných v reálnom čase. Základom je riadiaca logika schopná prevziať parametre zadané užívateľom a správne ich interpretovať pri ovládaní komory. Pre reguláciu teploty vnútri komory slúži PID regulátor ktorého vstupom je vnútorná teplota a výstupom výkon dodávaný do termoelektrických článkov.

ABSTRACT OF THE DIPLOMA THESIS

Slovak University of Technology in Bratislava
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Branch:	Electrical and Electronics Engineerin
Study Programme:	Electronics and Photonics
Author:	Bc. Rastislav Straka
Bachelor Thesis:	Design of temperature chamber controler
Supervisor:	Ing. Daniel Arbet, PhD.
Submitted:	May 2021
Keywords:	Temperature Chamber, C Programing language, PID controller, STM32, Thermoelectric effect, Communication buses

This master thesis presents the work process and the knowledge gained in creating a control program for the temperature chamber. STM32CubeMX, VSCode, GNU Embedded Toolchain, Make and OpenOCD were used for project creation, code editing, compilation / processing and optimization of the proposed algorithm. The program written in C language for the STM32F205VGT6 microcontroller has the task of controlling the temperature chamber device. Its tasks include controlling the outputs and inputs of the micro-controller, operating individual peripherals and calculating control variables in real time. Base of the program is the control logic able to take over the parameters entered by the user and interpret them correctly when controlling the chamber. A PID controller is used to regulate the temperature inside the chamber, the input of which is the internal temperature and the output is the power supplied to the thermoelectric cells.

Obsah

Zoznam použitých skratiek a označení

m	mili, 10^{-3}
μ	micro, 10^{-6}
n	nano, 10^{-9}
k	kilo, 10^{3}
M	mega, 10^{6}
G	giga, 10^{9}
B	Bajt (Byte)
b	Bit (Bit)
DUT	Testovaný obvod (Device Under Test)
LDO	Regulátor napätia s nízkym úbytkom (Low-DropOut)
SI	Medzinárodná sústava jednotiek (Systéme International)
MCU	Jednočipový mikropočítač (Microcontroller Unit)
SPI	Synchrónne sériové periférne rozhranie (Serial Peripheral Interface)
I^2C	Synchrónna dvojvodičová zbernica (Inter-Integrated Circuit)
PWM	Impulzne šírková modulácia (Pulse Width Modulation)
UART	Asynchrónna dvojvodičová zbernica (Universal Asynchronous Receive and Transmit)
V	Volt
A	Ampér
Ω	Ohm
Float	Druh premennej v Jazyku C uchovávajúcej čísla s plávajúcou desatinnou čiarkou.
Int	Druh celočíselnej premennej v Jazyku C
DPS	Doska plošných spojov
SWD	Single Wire Debuging
RTC	Hodiny reálneho času (Real Time Clock)

Úvod

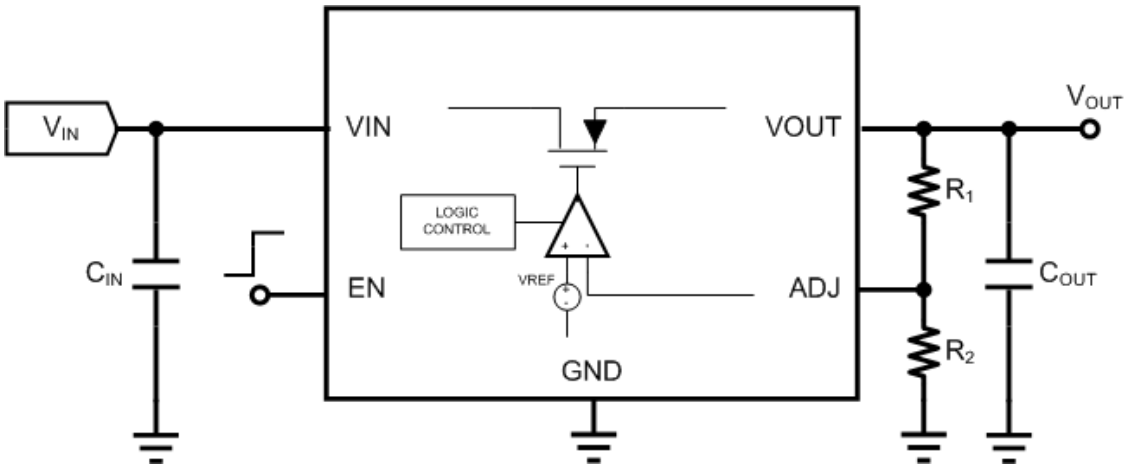
Pri návrhu, verifikácií a testovaní elektronických súčiastok, komponentov a obvodov je nevyhnutné realizovať rôzne merania ich parametrov. Taktiež takéto merania je potrebné realizovať v požadovanom teplotnom rozsahu. Na tento účel ako doplnok k meracím systémom slúžia zariadenia nazývané teplotné komory. Súčasťou teplotnej komory je uzavretý priestor, v ktorom je upravovaná teplota vzduchu. Úlohou teplotnej komory je spolupracovať s meracím systémom a upravovať teplotu v jej vnútornom priestore tak ako to merací systém požaduje. Základnou vlastnosťou je schopnosť neovplyvniť merací priestor inak ako zmenou teploty, pokiaľ nie je požadované inak. Takéto komory preto obsahujú zariadenia na odstraňovanie nahromadenej vlhkosti vznikajúcej pri kondenzácií pri znižovaní teploty. Teplotná komora by v ideálnom prípade mala byť riadená príkazmi priamo z riadiaceho systému, alternatívne vlastný ovládací program je dostačujúcou náhradou. Táto práca nadväzuje na tímový projekt, *Automatizované merania vybraných parametrov analógových IO s ultra-nízkym napájacím napätím*. Realizácia teplotnej komory na meranie vlastnosti IO s nízkym napájacím napätím, ktorá bola vytvorená na Oddelení návrhu a testovania IO FEI STU. Úlohou tejto práce bolo vytvoriť program na obsluhu a ovládanie zariadenia teplotnej komory skonštruovanej v tímovom projekte. Teplotná komora dosahuje zmenu vnútornej teploty pomocou termoelektrických Peltierových článkov. Program písaný v jazyku C pre mikro-kontrolér STM32F205 má za úlohu ovládať vstupno-výstupné porty a periférne rozhrania tohto mikro-kontroléra (MCU). Program využíva knižnice HAL, anglická skratka z Hardware Abstraction Layer, ktoré slúžia na priame ovládanie nízko úrovňových funkcií MCU. Teda slúžia na zjednodušené ovládanie registrov periférnych rozhraní, vstupno-výstupných portov a registrov vnútorného nastavanie MCU. Jednotlivé úlohy sú rozdelené do takzvaných modulov pozostávajúcich z .c a .h súborov C jazyka. Konkrétne tieto moduly slúžia na obsluhu displejov, tlačidiel dostupných užívateľovi, získavanie údajov o prostredí, ovládanie dodatočných vstupov a riadenie regulácií teploty. Na reguláciu teploty vnútri komory bol zvolený PID regulačný algoritmus. Tento bežne v praxi používaný algoritmus má za úlohu na základe spätnej väzby získanej interným meraním teploty komory, regulovať výstupnú veličinu ovládajúcu Peltierové články. Poslednou úlohou bola kalibrácia vnútorných senzorov teploty typu PT1000, pomocou kalibrovaných meracích prístrojov. Po správnom skalibrovaní merania teploty sa nastavia regulačné koeficienty PID regulátora tak aby bol priebeh regulácie dostatočne rýchly a presný. Nakoniec bola overená funkčnosť komory z hľadiska dosiahnutia maximálnej a minimálnej teploty ako aj dostatočné odstraňovanie vlhkosti.

1 Teoretický úvod

1.1 Teplotná komora

Teplotná alebo klimatická komora je zariadenie využívané pri testovaní vlastností rôznych zariadení alebo materiálov v širokom spektre teplôt. V elektrotechnike sa používa hlavne na meranie a ladenie parametrov elektronických súčiastok. Jedná sa najmä o polovodičové elektronické súčiastky ako sú napríklad tranzistory, diódy alebo integrované obvody. Vzhľadom k tomu že polovodičové súčiastky obsahujú P-N priechod, ktorého vlastnosti sú výrazne ovplyvňované teplotou je nutné pri ich návrhu zohľadniť rozsah teplôt a ich vplyv na vlastnosti a správanie celej súčiastky. Takáto testovaná súčiastka alebo zariadenie sa v anglickej terminológií označuje ako Device Under Test skrátene DUT. Ako jednoduchý príklad takéhoto testovaného zariadenia si môžeme uviesť napäťový regulátor s nízkym úbytkom napätia, označovaný ako Low-DropOut regulator skrátene LDO [1]. Pri takomto IO je nutné v rozsahu operačných teplôt zmerať napríklad tieto vlastnosti: úbytok napätia, odber prúdu v nezaťaženom stave, reguláciu záťaže (LDR parameter) a reguláciu výstupného napätia (LNR parameter). Parameter predstavujúci úbytok napätia je minimálny rozdiel medzi výstupným (regulovaným) a vstupným (neregulovaným) napätím pri ktorom je ešte daný obvod schopný spoľahlivo pracovať. V anglickej terminológií je práve tento parameter označovaný ako DropOut Voltage. Tento parameter je najviac ovplyvňovaný výkonovým prvkom regulátora. Saturačné napätie unipolárneho alebo bipolárneho tranzistora ktorý je využívaný ako výkonový prvok, je teplotne závislý. Saturačné napätie tohto tranzistora priamo udáva úbytok napätia regulatóra. Preto je tento test vyžadovaný. Parameter odberu prúdu v nezaťaženom stave vyjadruje vlastnú spotrebu daného obvodu. Pri LDO je tento parameter vzhľadom ku konštrukcií daného IO v porovnaní z jednoduchým lineárnym regulátorom veľmi dôležitý. Zatiaľ čo štandardný lineárny regulátor pri výkonovom prvku používa topológiu zapojenia typu „emitorového sledovača“ alebo inak povedané zapojenie so spoločným kolektorom, LDO regulátory využívajú zapojenie s takzvaným odpojeným kolektorom (open-collector alebo open-drain po anglicky). Pri tomto zapojení musí byť regulačný obvod postavený ako invertujúci zosilňovač, avšak spätná väzba z výstupu obvodu nemôže byť v tomto prípade privedená priamo na tento zosilňovač. Pretože napätia majú opačnú polaritu, spätná väzba musí byť doplnená o druhý inverujúci zosilňovač [2]. Týmto zložitosť celého integrovaného obvodu narastá a to negatívne ovplyvňuje parameter statickej spotreby. Regulácia záťaže predstavuje odozvu obvodu na zmenu výstupného prúdu. Pri zvýšení alebo znížení výstupného prúdu sa bez zásahu regulácie výstupné napätie v respektíve zníži alebo

zvýši. Vnútorný regulačný obvod LDO na tieto zmeny musí reagovať a udržať požadované výstupné napätie. Avšak vzhľadom k tomu, že rýchlosť odozvy obvodov operačných zosilňovačov sa mení na základe teploty je nutné aj tento parameter otestovať v rôznych teplotných medziach. Parameter LNR je vlastnosť LDO regulátora. ktorá udáva čas za aký sa po zmene vstupného napätia dostane výstupné regulované napätie na požadovanú hodnotu. Maximálny prúd ako parameter je naviazaný na minimálny úbytok napätia LDO regulátora. Tento parameter je okrem vlastností obvodu závislí aj na saturačnom napätí tranzistora. Pre rôzne typy LDO sa v závislosti na teplote tento parameter mení rôznymi spôsobmi. Na Obrázku č.1 Vnútorné zapojenie LDO je znázornená všeobecná vnútorná schéma takéhoto napäťového LDO regulátora.

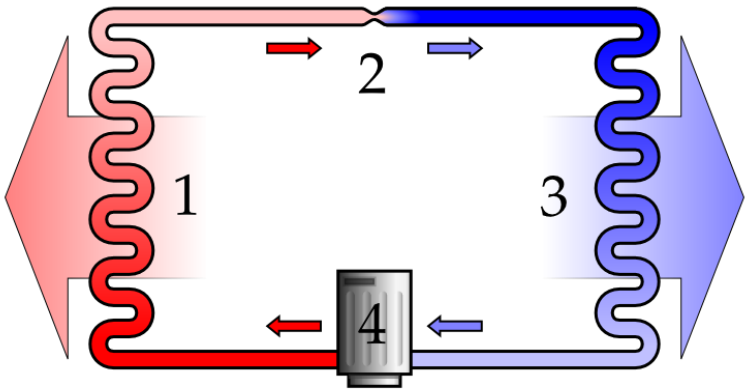


Obrázok 1 Vnútorné zapojenie LDO

Tieto parametre sú merané špeciálnymi prístrojmi umiestnenými mimo komoru. Prístroje sa starajú o správne napájanie LDO a meranie potrebných veličín, zatiaľ čo samotné LDO je umiestnené v komore, kde je menená teplota prostredia. Teplotná komora musí s takýmto meracím systémom spolupracovať a to tak, že bude udržiavať požadovanú teplotu týmto systémom počas celého procesu merania. Pokiaľ to nie je požadované, tak by v teplotnej komore, okrem zmeny teploty, nemalo dôjsť k zmene žiadnych iných fyzikálnych parametrov prostredia ako napríklad k zmene relatívnej vlhkosti alebo atmosférickému tlaku. Rozsah teplôt, ktorý musí vedieť komora dosiahnuť, je daný požiadavkami na verifikačné teploty DUT. Bežné rozsahy teplôt elektronických zariadení sa udávajú pre rôzne použitia podľa požiadaviek na očakávané prostredie. Aj keď sa jedná o bežne zaužívané rozsahy, výrobcovia ich môžu upraviť. V Tabuľka 1 sú uvedené najbežnejšie používané teplotné rozsahy. Pre automobilový sektor sa používa čiastočne aj norma AEC-Q100 alebo AEC-Q200 [3].

Tabuľka 1 - Rozsahy operačných teplôt elektronických súčiastok

Použitie	Minimálna operačná teplota	Maximálna operačná teplota
Komerčné (Commercial)	0 °C	70 °C
Priemyselné (Industrial)	-40 °C	85 °C
Vojenské (Military)	-55 °C	125 °C



Obrázok 2 Princíp fungovania tepelnej pumpy

Na dosiahnutie požadovaných teplôt sa používa princíp presunu tepelnej energie. Zariadenie ktoré na toto slúži sa nazýva tepelná pumpa. Môže pracovať na mechanickom, elektrickom alebo chemickom princípe. Chemický princíp, založený na exotermických a endotermických reakciách chemických látok, sa bežne nepoužíva vzhľadom na relatívne riziko týchto reakcií a možnosť narušenia atmosféry v meracom priestore. Mechanický princíp je založený na stláčaní a rozťahovaní chladiaceho média. Kompresor stlačí chladiace médium nad úroveň atmosférického tlaku, a týmto krokom sa médium zohreje ale stále ostáva v plynnej forme. Médium vchádza do kondenzátora kde sa schladí a skondenzuje do kvapalného skupenstva. Takto schladené médium prechádza cez zúžené miesto do výparníka kde sa zmení z kvapalnej formy na plynnú a pri tomto procese pohltí do seba tepelnú energiu a tak schladí okolie výparníka. Pri správnej mechanickej konštrukcií môže tento proces pracovať obojsmerne a teda vnútorný priestor komory chladiť alebo ohrievať [4]. Na Obrázku č. 2 znázornený funkčný princíp takejto tepelnej pumpy. Číslo 1 predstavuje kondenzátor, číslo 2 kapilárne zúženie, číslo 3 výparník a číslo 4 kompresor

Elektrický spôsob využíva termočlánky pracujúce väčšinou na termoelektrickom princípe. Termoelektrický princíp sa delí na Peltierov, Seebeckov a Thomsonov. Peltierov a Seebeckov princíp sú fyzikálne zhodné princípy, pomenované po dvoch objaviteľoch a ich nezávislom objavení tohto princípu, avšak popisujú rôzne prejavy Termoelektrického efektu. Thomsonov

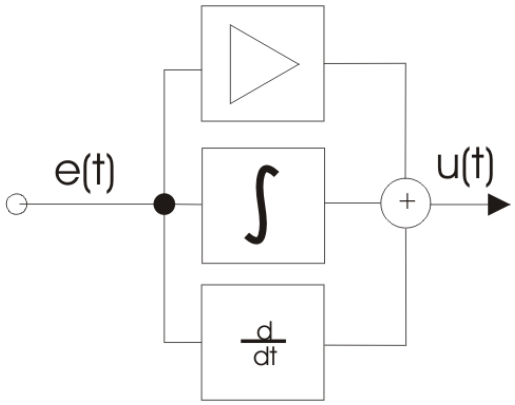
princíp je rozšírením predošlých princípov. V chladiacich systémoch sa používa Peltierov princíp ako tepelná pumpa. Pri prechode prúdu cez termočlánok jedna strana pohlcuje tepelnú energiu zatiaľ čo druhá tepelnú energiu vyžaruje. Pri odoberaní tepelnej energie z jednej strany a privádzaní tepelnej energie na stranu druhú môžeme docieliť schladenie jedného priestoru alebo zohriatie druhého [5].

1.2 Teória PID regulátora

Skratka PID znamená Proporčný-Integračný-Derivačný. Jedná sa o jednotlivé zložky regulátora, Ktoré sú reprezentované koeficientami K_p, K_i a K_d. Tieto koeficienty predstavujú zosilňovací činiteľ jednotlivých zložiek regulátora a teda aj ich podiel na konečnom výsledku. Každý takýto regulátor má na vstupe dve premenné, a to riadiacu premennú a poruchovú premennú. Riadiaca premenná predstavuje z vonku nastavený požadovaný stav systém, v angličtine býva označená ako Set Point, teda požadovaná hodnota. Poruchová premenná vyjadruje aktuálny stav v regulovanom systéme a je teda spätnou väzbou zo systému. Systém obsahuje aj jednu vnútornú premennú, nazývanú regulačná odchýlka, označovanú *e*, skratka z anglického error. Výstupom zo systému je premenná označovaná ako regulačný zásah alebo jednoducho výstupná premenná, ktorá priamo alebo nepriamo riadi akčný člen regulovaného systému A označovaná je ako u(t). Jednotlivé zložky sú reprezentované nasledovne. Proporčná zložka je rozdiel nastaveného bodu a poruchovej premennej, a teda je regulačnou odchýlkou. Integračná zložka integruje nazbierané odchýlky v čase. Derivačná zložka je rozdiel aktuálnej regulačnej odchýlky a poslednej regulačnej odchýlky a teda predstavuje celkovú zmenu regulačnej odchýlky v čase. Všetky tri zložky sú vynásobené príslušnými koeficientmi a sčítané dokopy. Celý proces je možné vyjadriť rovnicou.

$$u(t) = K_p * e(t) + K_i * \int_0^t e(t)dt + K_d * \frac{de(t)}{dt} \tag{1.1}$$

Výstupná premenná môže byť ďalej upravená tak aby jej jednotka zodpovedala jednotkám prijímaným akčným členom. Graficky znázornená schéma je na obrázku č. 3.



Obrázok 3 PID regulátor princíp [6]

Pri reálnom použití regulátora je nutné správne nastaviť jednotlivé koeficienty. V praxi sa používajú hlavne dve metódy. Pri prvej metóde sa nastavia koeficienty K_i a K_d na hodnotu nula a následne sa zvyšuje koeficient K_p až do doby, pokiaľ nezačne výstup regulátora oscilovať a zároveň sa nedosiahne požadovaná rýchlosť odozvy systému. Následne sa začne nastavovať parameter K_i tak, aby sa odstránili oscilácie a zároveň dosiahla čo najmenšia oscilačná odchýlka. Toto spôsobí vysoký presah regulácie pri snahe dosiahnuť žiadaný stav, takzvaný overshoot. Tento presah sa odstráni správnym nastavením koeficientu K_d. Druhá metóda sa nazýva Ziegler-Nicholsonova, kde sa znovu koeficienty K_i a K_d nastavia na hodnotu nula a hodnota koeficientu K_p sa zvyšuje dovtedy, pokiaľ systém nezačne oscilovať. Následne sa zaznamená perióda oscilácií označená P_c a kritická úroveň koeficientu K_p označená ako K_c a podľa tabuľky 2 sa následne nastavia koeficienty.

Tabuľka 2 Zieger-Nicholsova tabuľka nastavení [7]

Regulátor	**K_p**	**K_i**	**K_d**
P	0,5 x K_c	-	-
PI	0,45 x K_c	$P_c/1,2$	-
PID	0,60 x K_c	0,5 x P_c	$P_c/8$

2 Popis zariadenia

2.1 Popis tepelnej komory

Teplotná komora je skonštruovaná z hliníkových profilov a ohýbaného plechu. Vrchná časť zariadenia je skonštruovaná tak aby mohla poňať samotnú teplotnú komoru, táto komora je skonštruovaná v tvare vaničky zasadenej v izolácií. Po stranách sú umiestnené Peltierové článkyna ktorých sú upevnené chladiče z ventilátormi. V spodnej časti zariadenia je vytvorené miesto pre uloženie elektroniky ovládajúcej zariadenie. Na dosiahnutie potrebných teplôt vo vnútri komory sú používané termoelektrické Peltierové články. Tieto články pracujú na termoelektrickom princípe. Prechod prúdu takýmto článkami spôsobuje prenos tepelnej energie z jednej strany článku na druhý, ako dôsledok teplota jednej strany klesá a teplota druhej strany stúpa. Moduly článkov použité v zariadení sú typu MCPK2-19808AC-S [8]. Maximálny rozdiel teplôt medzi jednotlivými stranami je 85 °C. Ich maximálny výkon je 52 W na modul, a v zariadení je použitých celkovo 6 modulov. Pre dosiahnutie najlepšieho chladiaceho/výhrevného výkonu je potrebné zabezpečiť prietok vzduchu okolo chladičov peltierových článkov. Toto zabezpečujú priemyselné ventilátory z výkonom 20 W a rozmermi

Obrázok 4 Teplotná komora

92 x 92 mm. Počas procesu chladenia vzniká v komore vlhkosť, ktorá kondenzuje zo vzduchu. O odstránenie vlhkosti sa stará systém so vzduchovým čerpadlom ktoré vzduch cirkuluje cez nádobu s absorpčnou látkou. Vlhkosť a teplota vo vnútri komory je meraná pomocou senzorov. Zozbierané informácie sú zhromažďované v ovládacej doske zariadenia.

2.2 Elektronické časti zariadenia

Elektronická časť zariadenia sa skladá zo štyroch funkčných blokov. Každý z týchto blokov je realizovaný ako samostatná doska plošných spojov. Dosky sú elektricky prepojené pomocou konektorov a vodičov. Napájanie termoelektrických článkov je realizované pomocou zdrojov Mean Well, typu RSP-750-48 [9] a ich výstupné jednosmerné napätie je 48 V a disponujú maximálnym výkonom 750 W. Hlavná riadiaca doska obsahuje mikro-kontrolér (ďalej len MCU) a je teda základným funkčným blokom. Ďalším blokom je tranzistorový H-mostík, ktorý zabezpečuje prepínanie polarity peltierových modulov a tým pádom aj zmenu medzi režimom chladenia a vyhrievania. Výkon dodávaný do peltierových modulov reguluje lineárny zdroj realizovaný ako ďalší samostatný blok a DPS. Je osadený výkonovým tranzistorom a obvodom s operačným zosilňovačom. Zdroj je realizovaný ako prúdom riadený lineárny zdroj napätia. Posledným článkom je zásuvný modul pre sieťové zdroje RSP-750. Modul slúži na dodatočnú reguláciu výstupného napätia zdrojov. V súčasnej verzií však tento modul nie je osadený, a ani súčasná verzia hlavnej riadiacej dosky nepodporuje jeho obsluhu.

2.2.1 Hlavná riadiaca doska

Na hlavnej riadiacej doske sa nachádzajú súčasti používateľského rozhrania a periférií. Základom ovládacej dosky MCU STM32F205VGT6 [10]. Vzhľadom k tomu že tento druh MCU nemá jednoducho prístupnú energeticky nezávislú (non-volatile) pamäť na dlhodobé ukladanie dát, je k danému MCU pripojená EEPROM pamäť. Veľkosť pamäte je 16kb (2kB) a komunikuje s MCU cez rozhranie I^2C. Na zobrazovanie údajov slúžia tri 5-číslicové 7-segmentové displeje. Ovládacie rozhranie je znázornené na obrázku č.4 Ovládacie rozhranie zariadenia. Veľký displej je ovládaný priamo výstupmi MCU v zapojení multiplex. Dva menšie sú ovládané pomocou obvodu MAX6954ATL [11]. Tento obvod komunikuje s MCU pomocou rozhrania SPI. Pre vstup užívateľa sú na doske osadené štyri tlačidlá a jeden optický enkodér. Poslednou časťou užívateľského rozhrania je akustický menič schopný vytvárať zvuk jedného tónu. Spätná väzba o stave komory je získavaná pomocou teplotných snímačov a senzorov vlhkosti. Teplota v komore je snímaná pomocou dvojice platinových odporových snímačov. Pre jednoduchšiu obsluhu sú snímače pripojené k obvodom MAX31865ATP [12]. Tieto

obvody zabezpečujú konverziu z analógových údajov o odpore na digitálne údaje ktoré je MCU schopné spracovať, zároveň zabezpečujú aj správne napájanie odporového mostíka snímačov. S MCU komunikujú cez rozhranie SPI. Na získanie teploty a vlhkosti sú použité obvody SHT31-DIS-B [13]. Tieto obvody obsahujú integrovaný senzor vlhkosti a teplotný snímač. Ich súčasťou je ADC a s MCU komunikujú po zbernici I^2C. Vďaka ich konštrukcií nie je potrebná kalibrácia údajov vysielaných zo senzora, nakoľko sú vstavané senzory vnútorne kalibrované na presnosť 1%. Posledným senzorom dostupným na doske je senzor osvetlenia VEML7700 [14] komunikujúci s MCU taktiež po zbernici I^2C. Doska ďalej obsahuje obvody na úpravu IO signálov z MCU pre priame ovládanie podporných blokov na samostatných DPS a ostatných periférií. Jedná sa o ovládanie ventilátorov, pumpy a DPS obsluhujúcich Peltierové moduly a napájacie zdroje.

Obrázok 5 Ovládacie rozhranie zariadenia

2.2.2 Zásuvný modul na riadenie zdrojov RSP-750

Výstupné napätie zdrojov Mean-Well RSP-750 je možné riadiť pomocou malého jednosmerného napätia v rozsahu 40 - 100 %. Na vytvorenie tohto napätia slúži tento zásuvný modul. Jeho úlohou je pomocou operačných zosilňovačov upraviť výstupné riadiace napätie

z MCU na napätie vhodné ovládať tieto zdroje. Modul má slúžiť ako primárny regulačný prvok výkonu dodávaného do Peltierových článkov.

2.2.3 Lineárny zdroj napätia

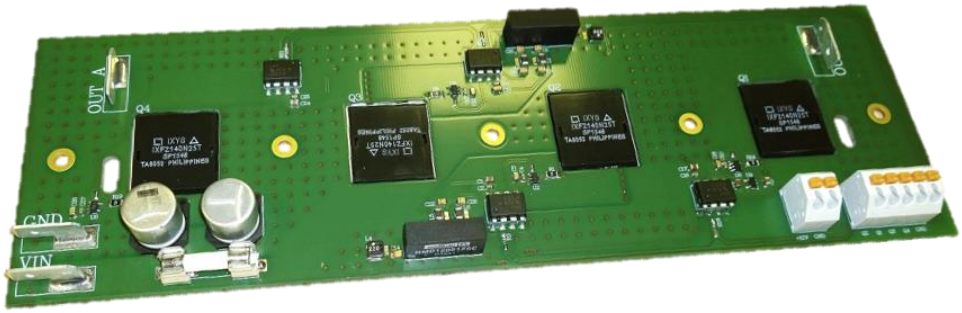
Úlohou tohto modulu je dodatočná a presnejšia regulácia napätia a teda aj výkonu dodávaného do Peltierových článkov. Jedná sa o lineárny zdroj napätia riadený prúdom ktorý je ovládaný DAC výstupmi MCU. Výstupné napätie je možné ovládať v rozsahu 0 - 48,3 V. Tento modul umožňuje presne kontrolovať výkon dodávaný do termoelektrických článkov a tak riadiť chladiaci/výhrevný výkon. Dvojitý operačný zosilňovač ADA4522-2ARZ[15] je zapojený ako snímač prúdu so sekundárnym stupňom napájajúcim NPN tranzistor v zapojení so spoločným kolektorom. Na výstupe tohto stupňa sa nachádza výkonový PMOS tranzistor s maximálnou výkonovou stratou 890 W. Táto výkonová rezerva by mala byť dostatočná pre regulovanie výkonu zdrojov RSP-750 ktorých maximálny výstupný výkon je 750 W. V zariadení sa nachádzajú dva takéto moduly, každý pre jeden zdroj RSP-750 a trojicu Peltierových článkov. Regulátor sa nachádza na obrázku č.6 Lineárny regulátor.

Obrázok 6 Lineárny regulátor

2.2.4 Tranzistorový mostík

Tento modul slúži na zmenu polarity termoelektrických článkov. Zmena polarity zabezpečuje zmenu medzi módmi chladenia a vyhrievania. Modul predstavuje tranzistorový H-mostík zložený z NMOS tranzistorov. Riadiace vstupy od MCU sú opticky izolované od budiacich IO tranzistorov a tak aj od celého H-mostíka pomocou obvodov HCPL-2300-300E[16]. Tieto obvody sú rýchle opto-členy s izolačnou schopnosťou do 600 V. Mostík

využíva len NMOS tranzistory s nízkym odporom kanálu v zopnutom stave, teda parameter $R_{DS(ON)}$, ktorý má hodnotu menej ako 17 mΩ [17]. Na to aby mohli byť takéto tranzistory použité aj na vrchnej (high-side) strane mostíka je nutné použiť špecializované obvody na napájanie ako opto-členov tak aj hradiel tranzistorov. Na tejto doske sú pre tento účel použité obvody NMD120512SC[18]. Výkonové tranzistory sú dimenzované tak aby dokázali prepínať maximálny výkon dodávaný jedným zdrojom RSP-750. Podobne ako lineárne regulovateľné zdroje sú v zariadení použité dva tranzistorové mostíky, každý pre jeden zdroj RSP-750 a trojicu Peltierových článkov.



Obrázok 7 H-Mostík

3 Tvorba ovládacieho programu

3.1 Nástroje použité na programovanie

Základná štruktúra kódu je vytvorená pomocou programu STM32CubeMX. V tomto programe je možné nastaviť základnú inicializačnú konfiguráciu vstupno-výstupných (ďalej len I/O) portov a periférnych rozhraní daného MCU, v našom prípade STM32F205VGTx. Podľa schémy hlavnej riadiacej dosky boli nastavené parametre portov a zároveň boli vhodne symbolicky označené pre lepšiu možnosť orientácie v kóde. Pri bežných I/O bol nastavený ich smer, a teda či sa jedná o vstup alebo výstup. Zároveň boli nastavené hodnoty registrov I/O portov nastavujúcich pripojenie pinu cez rezistor k napájaciemu napätiu alebo k zemi. Pre piny umožňujúce pulze-šírkovej modulácie (ďalej len PWM) výstupného signálu, boli priradené a správne nastavené vnútorné časovače MCU. Týmto krokom je možné ušetriť výpočtový čas MCU a zjednodušiť riadenie takýchto pinov. Pri vstupných pinoch, pri ktorých je nutná rýchla odozva na zmenu ich stavu, boli k takýmto pinom priradené externé prerušenia. Ďalej boli v programe nakonfigurované a priradené na zodpovedajúce piny zbernice a rozhrania UART, I^2C a SPI. Zároveň bolo nakonfigurované programovacie a ladiace rozhranie SWD a pripojenie rezonančných kryštálov. V tomto programe boli nastavené aj konfigurácie časovania MCU, a teda hodnoty taktu hodinového signálu pre výpočtovú časť MCU, hodiny reálneho času (ďalej len RTC) a periférií. Pri vytváraní projektu týmto programom sme ako výstup zvolili možnosť tzv. makefile. Pri zvolení tejto možnosti môžeme na úpravu, kompilovanie a ladenie kódu použiť ľubovoľné nástroje. Ako editačné prostredie bol zvolený editačný program VSCode. Tento program umožňuje doinštalovanie mnohých užitočných modulov, ktoré pomáhajú pri písaní kódu. Pre skompilovanie kódu bol zvolený kompilátor GNU Arm Embedded Toolchain. Nahrávanie programu do MCU v podobe binárneho súboru nám umožnil nástroj OpenOCD. Oba nástroje je možné používať priamo z prostredia VSCode po správnom nakonfigurovaní projektu. Na vytváranie súborov pre kompilovanie zo zdrojového kódu je použitý nástroj Make.

3.2 Popis programu

Hlavnou úlohou programu je obsluha súčastí zariadenia a užívateľského rozhrania. Program musí zabezpečiť obsluhu senzorov a automaticky nastavovať parametre výkonu pre ventilátory, Peltierové články a automaticky zabezpečovať odstránenie vlhkosti, podľa parametrov zadaných užívateľom. Program je napísaný v jazyku C. Užívateľské rozhranie je tvorené trojicou 7-segmentových displejov, štyrmi tlačidlami a rotačným enkodérom. Riadenie

samotnej teplotnej komory má na starosti PID regulačná slučka. Regulátor prepočítava údaje získané z platinových odporových senzorov teploty. Výstupom sú riadiace premenné pre DAC výstupy ktoré priamo ovládajú výkon dodávaný do Peltierových článkov. PID regulátor zároveň ovláda smer prúdu cez Peltierové články a tak prepína medzi módmi chladenia a vyhrievania. Na základe výkonu dodávaného do Peltierových článkov a vonkajšej teploty je nastavovaná úroveň rýchlosti ventilátorov. Program je rozdelený na dve časti. Synchrónnu a asynchrónnu časť. Synchrónna časť programu je spúšťaná z prerušenia od systémového časovača. V tejto časti programu sú vykonávané rutiny PID regulátora, ovládania H-mostíkov a ventilátorov, obnovovanie a multiplexovanie displejov. Jedná sa o úlohy ktoré musia byť vykonané v daných časových intervaloch. V asynchrónnej časti programu sú vykonávané časovo náročnejšie funkcie pri ktorých však nezáleží na tom v ktorom čase budú vykonané. Obe slučky môžu prerušiť krátke externé prerušenia obsluhujúce vstupy pri ktorých je nutné presne zaznamenávať čas ich zmeny. Jedná sa hlavne o vstupy tachometrov jednotlivých ventilátorov a vstup od optického enkodéra. Pri týchto vstupoch je potrebné presne odmerať čas zmeny vstupu. A na základe tohto času vypočítať otáčky ventilátora alebo zistiť smer otáčania enkodéra. Program je rozdelený na moduly. Modul je predstavovaný jedným .c a .h súborom. Predstavuje logickú jednotku ktorá obsluhuje určitú časť zariadenia.

3.2.1 Inicializačná časť programu

O inicializáciu programu sa z väčšej časti stará kód vytvorený programom CubeMX. Podľa parametrov nastavených v tomto programe sa s použitím funkcií knižníc HAL nastavia funkcie jednotlivých GPIO (General Purpose Input Output). Potrebné IO sa priradia k funkciám periférnych komunikačných zariadení ako SPI, I^2C a UART. K ostatným IO sa priradia funkcie vstupov a výstupov, nastavia sa požadované parametre výstupných portov ako stav logickej jednotky alebo logickej nuly pomocou tzv. pull-up a pull-down rezistorov. Pre potrebné vstupné porty sa aktivujú módy externých prerušení a nakonfigurujú sa potrebné funkcie spätného volania tzv. Callback funkcie. U výstupných portov sa pre žiadané IO nastavia dodatočné funkcie ako napríklad funkcia PWM (pulzná šírková modulácia). Ďalej sa nastavia interné registre hradlovej schémy a vyberú sa zdroje pre hodinový signál ako procesora samotného tak aj jednotlivých častí. Nastavia sa parametre vnútorných časovačov a zdroja reálneho času tzv. RTC (Real-Time Clock). Nasledujúca funkcia nahráva potrebné údaje o nastaveniach vnútorného rozdelenia hodinového signálu. O nastavení zdrojov hodinového taktu pre hlavný oscilátor tak ako aj pre oscilátor obvodu hodín reálneho času.

```
1.    /**
2.    * @brief System Clock Configuration
3.    * @retval None
4.    */
5.  void SystemClock_Config(void)
6.  {
7.    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
8.    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
9.    RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
10.
11.   /** Initializes the RCC Oscillators according to the specified parameters
12.   * in the RCC_OscInitTypeDef structure.
13.   */
14.   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE|RCC_OSCILLATORTYPE_LSE;
15.   RCC_OscInitStruct.HSEState = RCC_HSE_ON;
16.   RCC_OscInitStruct.LSEState = RCC_LSE_ON;
17.   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
18.   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
19.   RCC_OscInitStruct.PLL.PLLM = 16;
20.   RCC_OscInitStruct.PLL.PLLN = 192;
21.   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
22.   RCC_OscInitStruct.PLL.PLLQ = 4;
23.   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
24.   {
25.     Error_Handler();
26.   }
27.   /** Initializes the CPU, AHB and APB buses clocks
28.   */
29.   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
30.                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
31.   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
32.   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
33.   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
34.   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV4;
35.
36.   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
37.   {
38.     Error_Handler();
39.   }
40.   PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_RTC;
41.   PeriphClkInitStruct.RTCClockSelection = RCC_RTCCLKSOURCE_LSE;
42.   if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
43.   {
44.     Error_Handler();
45.   }
46. }
```

Obrázok 8 - Inicializácia oscilátorov MCU

3.2.2 Hlavná obslužná slučka

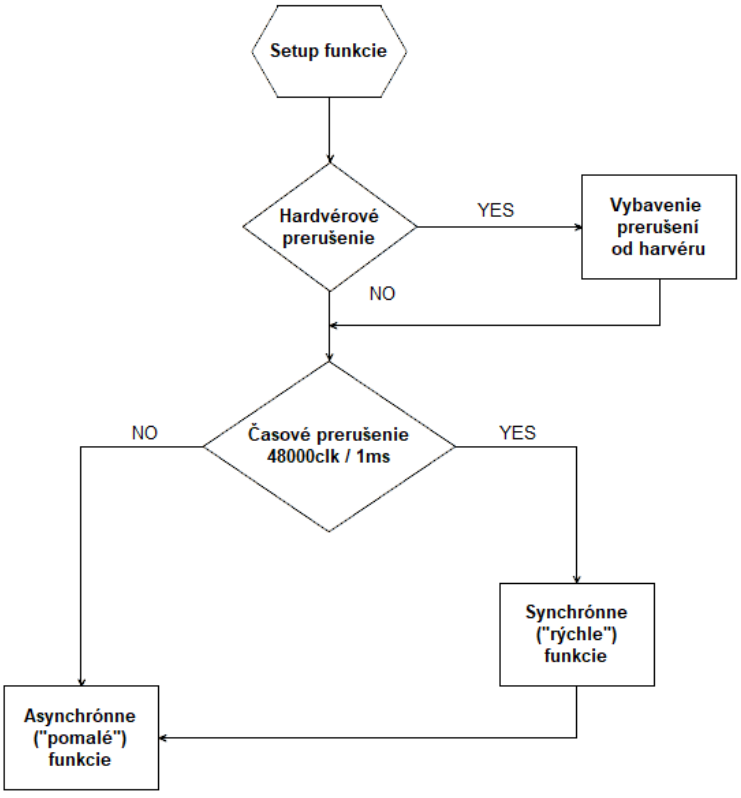
Program je rozdelený do troch hlavných častí podľa toho akým spôsobom je volané vykonávanie rôznych funkcií. Po štarte programu pred začatím behu hlavnej slučky sú vykonané vyššie popísané základné nastavenia. Po dokončení inicializácií prejde program do hlavnej slučky z ktorej sú volané takzvané asynchrónne funkcie. Tieto funkcie nie je potrebné vykonávať v daných časových intervaloch a ich vykonávanie môže byť prerušené. Interný časovač číslo 7 má nastavenú pred-deličku na hodnotu 47, toto znamená že hodinový takt

daného časovača je 1MHz. Časovač je nastavený tak aby počítal od 0 do 1000. Pri dosiahnutí hodnoty 0 v registri časovača dochádza k internému prerušeniu.

```
1.  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim7) {
2.    MD_INT_FSync();
3.  }
```

Obrázok 9 - Funkcia časového prerušenia

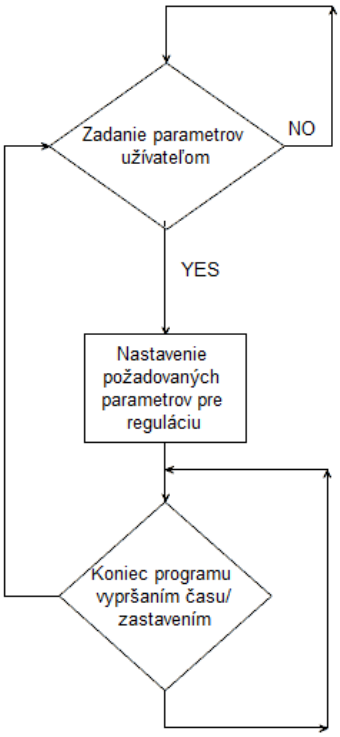
Interné prerušenie spustí funkciu *HAL_TIM_PeriodElapsedCallback* ktorá zavolá funkciu modulu *md_int.c TcEcode MD_INT_FSync(void).* V rámci tejto funkcie sú volané ďalšie funkcie obsluhujúce procesy ktoré je potrebné vykonávať pravidelne a v istých časových intervaloch. Do oboch behov programu môže zasiahnuť externé hardvérové prerušenie. Funkcie hardvérového prerušenia sú volané z callback funkcií daných hardvérových prerušení. Ich priorita je nastavená tak aby mohla prerušiť všetky ostatné prerušenia. Z tohto dôvodu sú všetky tieto funkcie malé a optimalizované tak aby neoberali hlavné slučky o výpočtový výkon. Na obrázku č.10 Diagram priebehu programu je tento postup graficky znázornený.



Obrázok 10 Diagram priebehu programu

Riadiaca logika

Obsluha riadiacej logiky je rozdelená medzi asynchrónnu a synchrónnu slučku. Jej úlohou je riadiť ostatné funkcie ktoré riadia jednotlivé funkcie teplotnej komory. Po zadaní dĺžky programu a požadovanej teploty cez jednotlivé tlačidlá a rotačný enkodér sa zaznamená aktuálny čas v sekundách z RTC. Volaním funkcií modulu md_pid.c sa nastaví požadovaná teplota a následne sa začne proces regulácie. Podľa požiadaviek programu zadaného užívateľom sa čas pre skončenie programu počíta buď od začiatku programu alebo od dosiahnutia požadovanej teploty. Na veľkom displeji sa začne zobrazovať aktuálna teplota. Slučka kontroluje stav zariadenia až pokiaľ nevyprší nastavený čas alebo užívateľ cez tlačidlo neskončí program. Celý proces je znázornený na obrázku č.11 Diagram ovládacej slučky.



Obrázok 11 Diagram ovládacej slučky

3.2.3 Riadenie displejov

Zariadenie obsahuje tri päťčíslicové sedem segmentové displeje. Dva z týchto displejov sú zapojené na špecializovaný obvod MAX6954ATL ktorému sú posielané príkazy na zobrazenie cez zbernicu SPI. Tretí displej je zapojený priamo na IO porty MCU, v zapojení multiplex. To znamená že segmenty rozsvietené na jednej číslici pomocou pripojenia anód sa zobrazia aj na ostatných čísliciach. Riešením je rýchle prepínanie medzi číslicami takzvané multiplexovanie.

Jednotlivé katódy sa postupne aktivujú a pri každej zvolenej číslici sa nastavia anódy tak aby výsledný obrazec zodpovedal zvolenej číslici. Keďže prepínacia frekvencia IO MCU dosahuje v maxime až 2MHz je možné postupne aktivovať jednotlivé číslice z ich vlastnými obrazcami takou rýchlosťou že pre ľudské oko bude celý displej rozsvietený. V programe sa o toto stará funkcia *TcECode MD_D7_Mul_Refresh()* periodicky volaná z časového prerušenia. Hodnota ktorá má byť zobrazená na displeji je uložená v štruktúre typu *St_D7_Mul_Config* definovanej v súbore md_d7seg.h. Pri každom volaní sa obslúži len jedna číslica, pričom sa upraví vnútorná premenná funkcie typu „static“ tak aby bola v ďalšom behu obslúžená nasledujúca číslica. Ako prvá sa vypne katóda číslice obslúženej v predchádzajúcom behu. Následne sa prejde na ďalšiu číslicu. Podľa definície v štruktúre sa s poľa CharDefMltpx[] vyberie definícia čísla, ktoré má byť zobrazené na displeji. Následne sa podľa definície nastavia výstupy pre jednotlivé anódy. Funkcia v rýchlom slede nastavuje pri každom volaní jednotlivé číslice podľa definícií.

```
1.  TcECode MD_D7_Mul_Refresh(void) {
2.    // Variables
3.    uint8_t Mask = 0b10000000;
4.    uint8_t CurrentAnode = 0;
5.    uint8_t CurrentDigit = 0;
6.    static int8_t RefreshedDigit = 0;
7.    // Turn of last refreshed cathode
8.    HAL_GPIO_WritePin(WhiteDisplay.Cathodes_Port[RefreshedDigit],
    WhiteDisplay.Cathodes[RefreshedDigit], SET);
9.    // If whole display is refreshed start from start
10.   RefreshedDigit++;
11.   if(RefreshedDigit >= 5) {
12.     RefreshedDigit = 0;
13.   }
14.   // Pick value of current digit
15.   CurrentDigit = WhiteDisplay.Digit[RefreshedDigit];
16.   // Turn on current cathode
17.   HAL_GPIO_WritePin(WhiteDisplay.Cathodes_Port[RefreshedDigit],
    WhiteDisplay.Cathodes[RefreshedDigit], RESET);
18.   // Find the status of decimal point
19.   if(WhiteDisplay.DigitDPStatus[RefreshedDigit] == DP_ON)
20.     HAL_GPIO_WritePin(WhiteDisplay.Anodes_Port[CurrentAnode],
    WhiteDisplay.Anodes[CurrentAnode], SET);
21.   else
22.     HAL_GPIO_WritePin(WhiteDisplay.Anodes_Port[CurrentAnode],
    WhiteDisplay.Anodes[CurrentAnode], RESET);
23.   // Advance anode and mask
24.   CurrentAnode++;
25.   Mask = Mask >> 1;
26.   // Set all anodes acording to given character selection
27.   for(;CurrentAnode <= 7; CurrentAnode++) {
28.     if(CharDefMltpx[CurrentDigit] & Mask)
29.       HAL_GPIO_WritePin(WhiteDisplay.Anodes_Port[CurrentAnode],
    WhiteDisplay.Anodes[CurrentAnode], SET);
30.     else
31.       HAL_GPIO_WritePin(WhiteDisplay.Anodes_Port[CurrentAnode],
    WhiteDisplay.Anodes[CurrentAnode], RESET);
32.     Mask = Mask >> 1;
33.   }
34.   return TC_OK;
35. }
```

Obrázok 12 Obsluha displeja

Ostatné dva päťčíslicové displeje pripojené na obvod MAX6954 je nutné riadiť pomocou príkazov vysielaných na SPI zbernicu. Aj keď obvod MAX6954 obsahuje vnútorne zadefinované znaky, ktoré je možné zobrazovať na displejoch, jedná sa len o 16 znakov bežných pre tieto displeje. Avšak tento obvod umožňuje aj priame riadenie jednotlivých segmentov. Keďže pre predchádzajúce riadenie displeja máme zadefinovanú mapu segmentov pre rôzne znaky, tak ju použijeme pri riadení. Pre riadenie týchto displejov je vytvorená funkcia *MD_D7_SPI_Write()*. Vstupom do funkcie je smerník na štruktúru *St_D7_SPI_Config*. V štruktúre sú uložené údaje o požadovanom zobrazení číslic a adresy jednotlivých registrov pre dané číslice. Funkcia v slučke naplní odosielací zásobník podľa záznamov v priloženej štruktúre. Pokiaľ je CS (Chip select) výstup nastavený na úroveň logická 1 tak sa v neblokujúcom režime odošle zásobník na zbernicu SPI. Funkcia je volaná z funkcie *MD_D7_SPI_Refresh0* pre každý päť číslicový displej zvlášť. Táto funkcia je volaná z hlavného časového prerušenia každú sekundu.

```
1.  TcECode MD_D7_SPI_Write(St_D7_SPI_Config *pDisplay) {
2.
3.    uint8_t Count = 0;
4.    uint8_t *pDigit = pDisplay->Digit;
5.    uint8_t *pAddresss = pDisplay->DigitAdress;
6.    uint8_t *pDPStatus = pDisplay->DigitDPStatus;
7.    uint8_t OutBuff[20];
8.    uint8_t *pOutBuff = OutBuff;
9.    // If last send digits are same as digits to be displayed skip!
10.   if(!(memcmp(pDisplay->Digit, pDisplay->LastWDigit, 5)))
11.     return TC_OK;
12.   // Fill the buffer with segment adresses and data
13.   for(Count=0; Count <= 4; Count++) {
14.     *pOutBuff = *pAddresss;
15.     pOutBuff++;
16.     *pOutBuff = CharDefMltpx[*pDigit];
17.     if(*pDPStatus == DP_ON)
18.       *pOutBuff |= 0b10000000;
19.     else
20.       *pOutBuff &= 0b01111111;
21.     pOutBuff++;
22.     pDigit++;
23.     pAddresss++;
24.     pDPStatus++;
25.   }
26.   // Access SPI only when previous writes were done, since we are using non blocking
    mode.
27.   if((HAL_GPIO_ReadPin(DDCS_GPIO_Port, DDCS_Pin)) == GPIO_PIN_SET) {
28.     memcpy(pDisplay->LastWDigit, pDisplay->Digit, 5);
29.     HAL_GPIO_WritePin(DDCS_GPIO_Port, DDCS_Pin, GPIO_PIN_RESET);
30.     HAL_SPI_Transmit(&hspi2, OutBuff, 20,100);
31.   }
32.   else {
33.     return TC_BUSY;
34.   }
35.   return TC_OK;
36. }
```

Obrázok 13 Obsluha displejov pripojených na SPI

Podporná funkcia na obsluhu modulu ovládania displejov *TcECode MD_D7_FloatToDis(float Value)* slúži na nastavenie štruktúr a naplnenie požadovaných hodnôt pre jednotlivé číslice. Funkcia teda prevádza číslo s plávajúcou desatinnou čiarkou na požadovanú konfiguráciu displeja a aktiváciu zobrazenia desatinnej čiarky na požadovanom mieste. Zároveň zisťuje či je potrebné zapnúť alebo vypnúť znamienko pred číslom, vzhľadom k tomu že je možné zobraziť len znamienko mínus.

3.2.4 Obsluha užívateľských vstupov

Obsluha všetkých vstupov je realizovaná v module *md_per.c*. Užívateľské vstupy predstavujú štyri tlačidlá na prednom paneli a rotačný optický enkodér s vlastným tlačidlom. Stav tlačidiel je zisťovaná priamo čítaním digitálnych vstupov. Pre smer otáčania enkodéra je nutné presne zaznamenať čas zmeny jednotlivých vstupov. Keďže vstupy enkodéra sú pripojené na externé prerušenie pri dobežnej hrane jedného zo vstupov je aktivovaná funkcia v súbore *main.c*.

```
1.  void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin) {
2.      if((GPIO_Pin == ENCA_Pin) || (GPIO_Pin == ENCB_Pin))
3.          MD_PER_EncoderCapture(GPIO_Pin);
```

Obrázok 14 Externé prerušenie od IO

Najprv táto funkcia porovná názov vstupu ktorý aktivoval prerušenie so vstupmi enkodéra. Ak sa zhodujú je zavolaná funkcia modulu *md_per.c MD_PER_EncoderCapture()*.

```
1.  TcECode MD_PER_EncoderCapture(uint16_t GPIO_Pin) {
2.
3.    static uint16_t PrevGPIO_Pin;
4.
5.    if(EncoderStatus == ENC_NOMOVE) {
6.      PrevGPIO_Pin = GPIO_Pin;
7.      EncoderStatus = ENC_CALC;
8.    }
9.    if(EncoderStatus == ENC_CALC) {
10.     if((PrevGPIO_Pin == ENCA_Pin) && (GPIO_Pin == ENCB_Pin))
11.       EncoderStatus = ENC_RIGTH;
12.     else if((PrevGPIO_Pin == ENCB_Pin) && (GPIO_Pin == ENCA_Pin))
13.       EncoderStatus = ENC_LEFT;
14.     else
15.       EncoderStatus = ENC_NOMOVE;
16.   }
17.   return TC_OK;
18. }
```

Obrázok 15 Zaznamenávanie stavu enkodéra

Funkcia najprv zistí či nebolo zistené nevybavené otočenie enkodéra. Ak nie zaznamená si hodnotu vstupu ktorý aktivoval prerušenie. Pri ďalšom prerušení sa porovná poradie aktivačných pinov prerušenia a zistí sa smer otáčania enkodéra. Ďalšie zaznamenanie nie je možné pokiaľ programové slučky nepreberú zaznamenaný pohyb. Zaznamenávanie hodnôt zvyšných piatich tlačidiel je vykonávané vo funkcií *MD_PER_BUT_Refresh0*. Funkcia periodicky volaná z časového prerušenia zisťuje hodnoty vstupov. Pri zmene na hodnotu logická 1 je zaznamenané prvé stlačenie a funkcia pri danom tlačidle čaká 10 períód volania. Potom znovu overí či je hodnota vstupu logická 1 a následne funkcia tlačidlo považuje za stlačené. Pri uvoľňovaní tlačidla funkcia postupuje rovnako ale so stavom logická 0. Takto sa docieli takzvaný softvérový „debouncing“ alebo inak eliminácia oscilácie kontaktov.

3.2.5 Získavanie údajov o prostredí

Údaje z RTD-to-Digital prevodníkov na meranie teploty pomocou senzorov PT1000 sú získavané z obvodu MAX31865ATP cez zbernicu SPI. O túto funkcionalitu sa stará funkcia modulu *md_per.c MD_PER_PTD_Get()*. Funkcia v prvom behu začne komunikáciu so zvoleným obvodom. Pre daný obvod sa najprv aktivuje CS výstup pripojený na MCU. Následne sa zapíše adresa registra, z ktorého sa má prečítať hodnota teploty. Šírka adresy registra, obsahujúca hodnotu teploty, je 16-bitová s bajtovým prístupom. Z tohto dôvodu je najprv posielaný horný bajt s hexadecimálnou adresou 0x01, a potom dolný bajt s adresou 0x02. Pri čítaní sa takisto hodnota vyčítavá ako horný a dolný bajt, a následne sa spoja v programe dohromady, pričom najmenej významný bit sa zahodí. Takáto hodnota je potom 15 bitov široká a môže nadobúdať hodnoty od 0 do 32 767. Zistená hodnota je len číslo z prevodníka, ktoré je potrebné zakomponovať do vzťahu na výpočet reálnej teploty, ktorá sa uloží do premennej v príslušnej štruktúre *St_Temp*.

3.2.6 PID regulátor

V samostatnom module zloženom s súborov md_pid.c a md_pid.h je naprogramovaná celá logika PID. Tieto súbory predstavujú zásuvný modul priradený zvyšku programu. V súbore typu C sú naprogramované jednotlivé funkcie a v súbore H sa nachádzajú potrebné definície. Hlavná funkcia *TcECode MD_PID_CALCULATE()* vykonáva všetky výpočty potrebné pre správnu činnosť regulátora. Celá funkcia je uvedená nižšie.

```
1.  TcECode MD_PID_CALCULATE(St_PID *PID) {
2.
3.      float Error = 0;
4.      float Derivative = 0;
5.      float Output = 0;
6.      int16_t CalculatedOutput = 0;
7.
8.      // Calculate new error from target and current temp
9.      Error = PID->TargetTemp - PID->CurrentTemp;
10.     // Recalculate integral variable
11.     PID->Integral += Error;
12.     // Calculate new derivative variable
13.     Derivative = Error - PID->LastError;
14.     // Calculate output
15.     Output = (PID->KP * Error) + (PID->KI * PID->Integral) + (PID->KD * Derivative);
16.     // Map Calculated variable to DAC range
17.     Output = mc_map(Output, PID->OutsideTemp - PeltierMaxDif, PID->OutsideTemp +
    PeltierMaxDif, -4095, 4095);
18.     // Change type to integer
19.     CalculatedOutput = (int16_t)(Output);
20.     // Decide on the heating or cooling
21.     if(CalculatedOutput > 0) {
22.         PID->HC_Status = Heating;
23.     }
24.     else if(CalculatedOutput < 0) {
25.         PID->HC_Status = Cooling;
26.         CalculatedOutput = CalculatedOutput * (-1);
27.     }
28.     else {
29.         PID->HC_Status = Off;
30.     }
31.     // Write to the memory
32.     PID->CalculatedOutput = (uint16_t)(CalculatedOutput);
33.     // Refresh Last members
34.     PID->LastError = Error;
35.     PID->LastCurrentTemp = PID->CurrentTemp;
36.
37.     return TC_OK;
38. }
```

Obrázok 16 Funkcia PID regulátora

Táto funkcia je volaná periodicky s pomocou prerušenia časovača. Jediným vstupom do funkcie je smerník na štruktúru typu *St_PID* zadefinovanú v súbore *md_pid.h*. Premenné obsiahnuté v tejto štruktúre uchovávajú všetky potrebné dáta pre regulátor. PID regulátor pracuje s reálnymi teplotami ako premennými. To znamená že požadovaná teplota aj skutočná nameraná teplota, ktoré predstavujú cieľový bod a spätnú väzbu, sú zaznamenávané v stupňoch Celzia. Najprv sa vypočíta odchýlka medzi skutočnou hodnotou teploty zmeranej vo vnútri zariadenia a nastavenou (požadovanou) teplotou. Táto odchýlka často označovaná ako chyba, predstavuje základnú výpočtovú premennú. Hodnota integrálnej časti sa vypočíta pripočítaním chyby k predchádzajúcej hodnote tejto časti. Hodnota je uchovávaná v premennej typu float v štruktúre *St_PID*. Derivácia je počítaná ako rozdiel aktuálnej odchýlky a odchýlky zaznamenanej pri predchádzajúcom behu funkcie. Následne sú všetky zložky vynásobené ich koeficientami a sčítané dokopy. Výstup je potom na-mapovaný na rozsah 12-bitového DAC

a teda rozsah hodnôt medzi -4095 až 4095, pretože h-mostík umožňuje využiť celý rozsah DAC ako pri chladení tak aj pri ohrievaní. Vstupný rozsah mapovacieho makra je určený z hodnoty vonkajšej teploty a maximálneho možného rozdielu teplôt Peltierového článku. Následne sa v logickom vetvení zistí či je požadované chladenie alebo zohrievanie a pre DAC sa hodnota výstupu upraví na absolútnu hodnotu. Výstupy výpočtu sú zapísané do *St_PID* štruktúry. V poslednom kroku sa zaznamenajú potrebné premenné pre ďalší beh. Pre ostatné moduly sú dostupné cez prototypové deklarácie v súbore *md_pid.h* funkcie s preponou *MD_PID_* a to funkcie: *GetValue, GetHCStatus, SetFeedBack, SetRefTemp* a *Refresh*. Prvé dve slúžia na získanie výsledku vypočítaného PID regulátorom, druhé dve slúžia na zadanie novo nameraných vstupných hodnôt pre regulátor a posledná je periodicky volaná z časového prerušenia a vykonáva výpočet hodnôt PID regulátora.

3.2.7 Obsluha dodatočných výstupov

Medzi dodatočné výstupy patrí ovládanie H-mostíkov, pumpy čerpadla odvlhčovača, riadenie ventilátorov a ovládanie výstupov DAC. Táto časť programu je tiež vykonávaná v module *md_per.c*. Pre ovládanie H-mostíkov je vytvorená ovládacia slučka ktorej úlohou je pri prepínaní MOSFET-ov zabezpečiť dostatočný čas na prepnutie tranzistorov. Aby sa predišlo skratu na vertikálnych vetvách mostíka pri prepínaní smeru toku prúdu cez horizontálnu vetvu, je najprv každý tranzistor v mostíku prepnutý do nevodivého stavu. Tento stav trvá 100 períód hlavného časového prerušenia. Následne je zvolený požadovaný mód H-mostíka a stav výstupov ovládajúcich tranzistory je nastavený podľa definície v *md_per.h*. O obsluhu H-mostíka sa primárne stará funkcia *MD_PER_HB_Refresh()* uvedená nižšie. Keďže funkcia je volaná každé časové prerušenie najprv sa skontroluje či je aktuálny stav mostíka rozdielny od požadovaného stavu. Ak áno mostík je uvedený do stavu pri ktorom sú všetky tranzistory deaktivované, zároveň sa nahrá do premennej hodnota oneskorovacieho časovača. Pokiaľ sa hodnota premennej nerovná 0 je pri každom volaní jej hodnota dekrementovaná o jedna. Keď tento časovač dosiahne hodnotu 0 požadovaný stav H-Mostíka je nastavený volaním funkcie *MD_PER_HB_Set()*. Po nastavení je vynulovaný časovač a požadovaný mód je zapísaný do premennej uchovávajúcej aktuálny mód.

```
1.  TcECode MD_PER_HB_Refresh(St_HBridgeConfig *HBridge) {
2.
3.    if(HBridge->DesiredMode != HBridge->ActualMode) {
4.      if(HBridge->ActualMode != HB_AllOff) {
5.        MD_PER_HB_Set(HBridge, HB_AllOff);
6.        HBridge->ActualMode = HB_AllOff;
7.        HBridge->DelayTimer = 100;
8.        return TC_BUSY;
9.      }
10.     else {
11.       if(HBridge->DelayTimer > 0) {
12.         HBridge->DelayTimer--;
13.         return TC_BUSY;
14.       }
15.       else {
16.         MD_PER_HB_Set(HBridge, HBridge->DesiredMode);
17.         HBridge->ActualMode = HBridge->DesiredMode;
18.         HBridge->DelayTimer = 0;
19.       }
20.     }
21.   }
22.   return TC_OK;
23.   }
```

Obrázok 17 Ovládanie H-Mostíkov

Na obsluhu štyroch panelových LED je vytvorená nasledujúca funkcia. Vstupom do funkcie je číslo danej LED od 0 po 3. Funkcia prepína výstup danej LED a aktuálny stav zaznamenáva do premennej *Leds*. Táto premenná je typu štruktúra zadefinovaná ako *St_LEDConfig*. V štruktúre sú obsiahnuté premenné odkazujúce na čísla a porty jednotlivých výstupov a jedna premenná uchovávajúca stav všetkých LED.

```
1.  TcECode MD_PER_LED_Toogle(uint8_t LED) {
2.
3.    uint8_t LED_MASK = 1;
4.    uint16_t pLED = (Leds.LED[LED]);
5.    GPIO_TypeDef *pLED_Port = (Leds.LED_Port[LED]);
6.
7.    LED_MASK = LED_MASK << LED;
8.    if (HAL_GPIO_ReadPin(pLED_Port, pLED)) {
9.      LED_MASK = ~LED_MASK;
10.     Leds.LEDStatus = Leds.LEDStatus & LED_MASK;
11.   }
12.   else {
13.     Leds.LEDStatus = Leds.LEDStatus | LED_MASK;
14.   }
15.   HAL_GPIO_TogglePin(pLED_Port, pLED);
16.   return TC_OK;
17. }
18.
```

Obrázok 18 Ovládanie LED

Riadenie rýchlosti ventilátorov je kvôli nemožnosti snímať teplotu na vonkajšej strane Peltierových článkov naviazané na výkon dodávaný do týchto článkov. Pre zamedzenie

poškodenia je minimálna hodnota rýchlosti stanovená na 30 percent. Funkcia preberie jednotlivé hodnoty výkonu z PID regulátora, vypočíta ich aritmetický priemer a na-mapuje ho na rozsah PWM regulácie a to 0-100 percent. Ak sa hodnota dostane pod hodnotu 30 % tak funkcia nastaví rýchlosť ventilátorov na túto hodnotu. Následne sa pre každý ventilátor zavolá funkcia *MD_PER_PWM_Set()*.

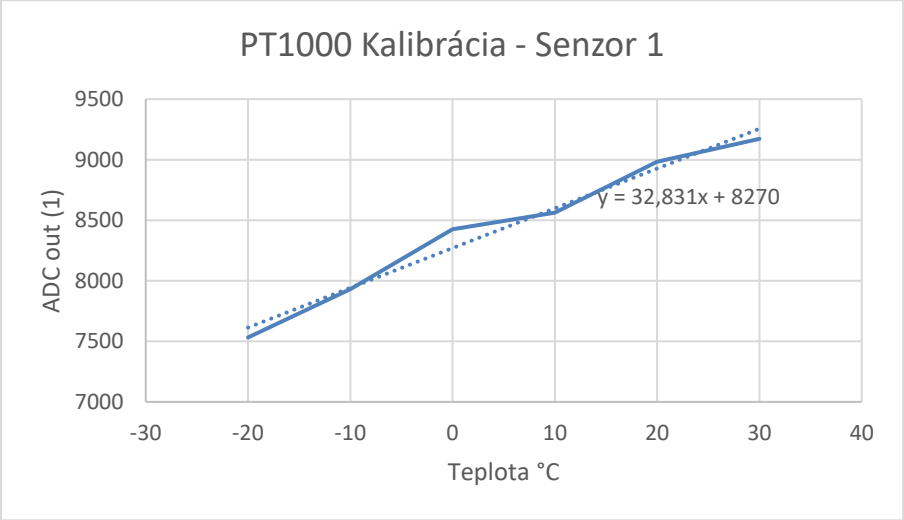
```
TcECode MD_PER_Fan_Control(void) {


1.
2.    uint16_t CV_PID1;
3.    uint16_t CV_PID2;
4.    uint16_t Control;
5.    float DutyCycle;
6.    // Get PID values
7.    CV_PID1 = MD_PID_GetValue(&PID1);
8.    CV_PID2 = MD_PID_GetValue(&PID2);
9.    // Get average
10.   Control = (CV_PID1 + CV_PID2) / 2;
11.   // Map control to PWM
12.   DutyCycle = mc_map((float)(Control), 0, 4095, 0.0, 100.0);
13.   // Limit min to 30 percent
14.   if(DutyCycle < 30.0)
15.     DutyCycle = 30.0;
16.   // Set duty cycle
17.   MD_PER_PWM_Set(FC0_TIM, FC0_CHAN, DutyCycle);
18.   MD_PER_PWM_Set(FC1_TIM, FC1_CHAN, DutyCycle);
19.   MD_PER_PWM_Set(FC2_TIM, FC2_CHAN, DutyCycle);
20.   MD_PER_PWM_Set(FC3_TIM, FC3_CHAN, DutyCycle);
21.   MD_PER_PWM_Set(FC4_TIM, FC4_CHAN, DutyCycle);
22.   MD_PER_PWM_Set(FC5_TIM, FC5_CHAN, DutyCycle);
23.   return TC_OK;
24. }
```

Obrázok 19 Ovládanie ventilátorov

4 Uvedenie do prevádzky

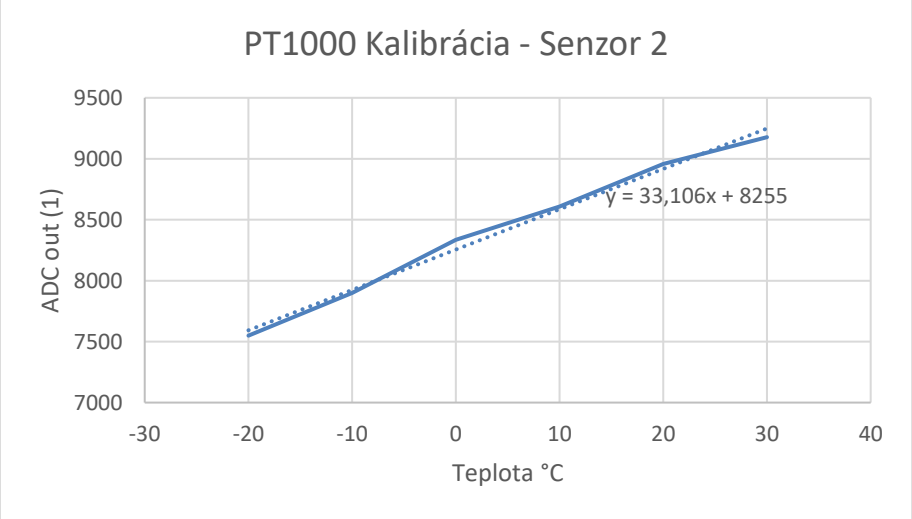
4.1 Kalibrácia PT senzorov

Pri kalibrácií senzorov PT1000 bol výkon dodávaný do jednotlivých Peltierových článkov riadený cez ladiace (debug) prístroje pre STM32. Postupne boli nastavované teploty od - 20 °C do 30 °C a cez ladiace nástroje bola odčítavaná aktuálna hodnota z ADC prevodníka obvodov MAX31865. Hodnoty boli zapisované do tabuľky a následne vnesené do grafu závislosti hodnoty ADC od teploty. V grafe boli jednotlivé hodnoty preložené lineárnou priamkou. Rovnica priamky tvaru $y = kx + q$ slúži na prepočítavanie kalibrácie. Na obrázku č.20 Kalibrácia senzora č.1 sa nachádzajú údaje namerané pre prvý senzor.



Obrázok 20 Kalibrácia senzora č.1

Na obrázku č.21 Kalibrácia senzora č.2 sa nachádzajú v grafe znázornené údaje namerané pre druhý senzor, kde vidíme podobnú závislosť od teploty



Obrázok 21 Kalibrácia senzora č.2

Na meranie teploty bol použitý merací prístroj UNI-T UT204R, s meracím termočlánkom typu K. Presnosť tohto meracieho prístroja v celom meracom rozsahu – 40 °C až 1000 °C je ±1.5 % + 5 číslic. Pri zobrazovacej presnosti na celé stupne to znamená že posledná meraná číslica má neistotu ± 5 °C. Na dodatočné overenie našej presnosti môžeme prepočítať hodnoty ADC na hodnotu odporu v Ω. Prepočet je možné spraviť podľa nasledujúceho vzorca.

$$R = 4 * \left(\frac{1}{32767}\right) * (ADC) \quad (1.2)$$

Hodnoty ktoré sme týmto prepočtom dostali sú v tabuľke 3 Prepočítané hodnoty ADC na odpor.

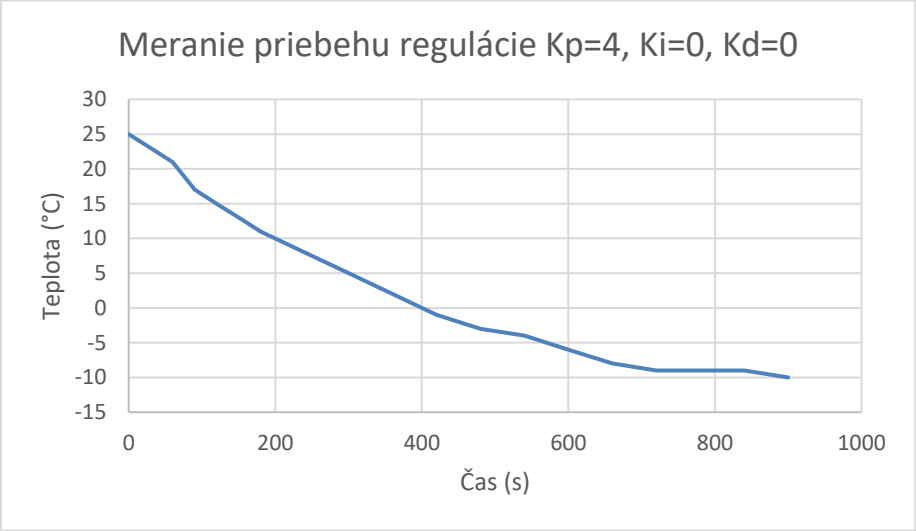
Tabuľka 3 Prepočítané hodnoty ADC na odpor

Teplota (°C)	Senzor 1 (Ω)	Senzor 2 (Ω)
-20	919,5	921,6
-10	968,4	964,3
0	1028,4	1017,4
10	1045,2	1050,7
20	1069,6	1093,5
30	1119,7	1120,3

Ak porovnáme tieto hodnoty s dostupnými hodnotami pre senzory PT1000, tak zistíme že nami namerané hodnoty sa približne zhodujú s hodnotami očakávanými pre tieto senzory pri daných teplotách.

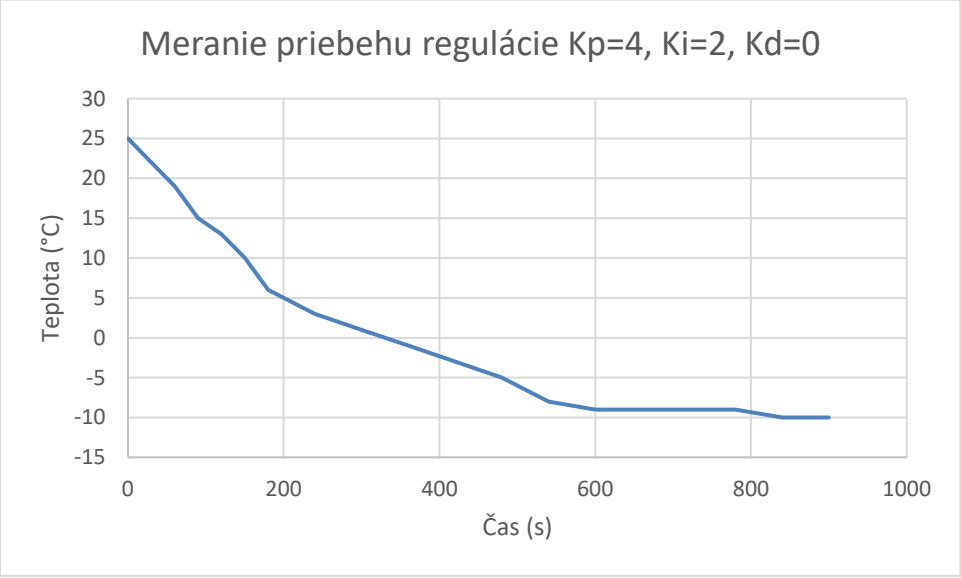
4.2 Nastavovanie koeficientov PID regulátora

Pri nastavovaní PID regulátora je nutné nájsť správne hodnoty pre jednotlivé regulačné koeficienty K_p, K_i a K_d. Metódy na toto určené sú popísané vyššie v kapitole 1.2. Pre obe metódy je nutné dostať regulátor do stavu oscilácií zvyšovaním hodnoty koeficientu K_p. Avšak kvôli vysokej rýchlosti implementovaného PID regulátora a pomalej odozve systému nie je možné regulátor uviesť do takéhoto stavu ani pri vysokých hodnotách koeficientu K_p, teda $K_p > 10$. Na obrázku 12 sa nachádza meranie pri hodnote $K_p = 4$.



Obrázok 22 Odozva regulátora $K_P = 4$

Pri nastavenej cieľovej teplote -10 °C sa na túto teplotu postupne regulátor dostane. Aj keď sa úplne neodstráni regulačná chyba. Dané chýbajúcou K_i zložkou. Regulátor však aj napriek tomu splní cieľ dosiahnutia požadovanej teploty. Ďalšími pokusmi sa dosiahlo optimálnejšie nastavenie avšak na toto nastavenie nebolo možné použiť Ziegler-Nicholsovú metódu. Pre optimálnejšie nastavenie bol zavedený aj koeficient K_i. Koeficient K_d nemá pri takto pomalej odozve systému veľký vplyv. Na obrázku č.23 sa nachádza Odozva regulátora Kp=4, Ki=2.



Obrázok 23 Odozva regulátora K_p=4, K_i=2

Pri tejto úprave parametrov sa zlepšil čas ustálenia a čiastočne odstránila regulačná chyba. Avšak pre dosiahnutie rýchlejšej odozvy regulátora by bolo potrebné ďalej optimalizovať jeho koeficienty.

Záver

Cieľom tejto práce bolo vytvoriť obslužný softvér schopný ovládať teplotnú komoru. Požiadavkou bolo napísať program v programovacom jazyku, ktorý je možné implementovať pre STM32 mikro-kontroléry. Pre nás najvhodnejšou voľbou bol jazyk C. Jeho veľkou výhodou je, že sa jedná o jazyk nízkej úrovne, a preto je možné dosiahnuť lepšej optimalizácie softvéru pre daný hardvér. Teda je možné napísať veľmi efektívny kód, ktorý bude dostatočne rýchly pre dané použitie bez zbytočného spomaľovania spôsobeného dodatočnou réžiou nadstavbových knižníc. Jediné knižnice použité pri písaní tohto projektu sú knižnice HAL. Jedná sa o knižnicu s nízkou úrovňou abstrakcie, ktorá uľahčuje ovládanie periférií. Pomocou týchto knižníc bol nakoniec v jazyku C vytvorený program, ktorý predstavuje most medzi užívateľom a zariadením. S využitím jazyka C bola implementovaná komunikácia medzi MCU a perifémymi obvodmi. Celý program riadia dve slučky, a to asynchrónna a synchrónna. Tieto slučky volajú jednotlivé obslužné funkcie a obsluhujú zariadenie v potrebnom momente. O reguláciu teploty vo vnútri komory sa stará PID regulátor implementovaný v programe ako samostatný blok. Tento regulátor preberá údaje o teplote namerané pomocou odporových PT-1000 senzorov. Tieto údaje spätnej väzby sú potom spracované tak, aby bol dosiahnutý stav požadovaný užívateľom. Pri kalibrácii a nastavovaní regulátora bol objavený drobný nedostatok, v podobe príliš rýchlej funkcie tohto regulátora. Toto znemožňuje dostatočne dobré nastavenie jeho parametrov. Do budúcna by bolo možné túto chybu odstrániť optimalizáciou odozvy PID regulátora, hlavne jeho spomalením vzhľadom k tomu, že v súčasnej verzii prebieha obnovovanie hodnôt PID regulátora každú 1 sekundu, čo je oveľa menej ako rýchlosť, za ktorú odozva veličiny nastavenej na akčnom člene spropaguje na meracie senzory. Ďalej by bolo užitočné zlepšiť užívateľské rozhranie a vymyslieť viac možností ovládania komory. Na tento účel by sa mohlo využiť integrované rozhranie RS-232 na priame ovládanie komory z osobného počítača vhodným programom. V neposlednej rade by bolo dobré komoru doplniť o senzory teploty snímajúce teplotu Peltierových článkov z vonkajšej strany. Týmto krokom by mohlo byť priatelenejšie vyriešená regulácia rýchlosti ventilátorov a tým by sa znížila hlučnosť zariadenia pri prevádzke. Celkovo sa však dá povedať že pri vypracovávaní tejto diplomovej práce sme nadobudli dostatok poznatkov ako takýto problém riešiť do budúcna.

Literatúra

[1] Low-Dropout regulator [online]
Dostupné na internete:
https://en.wikipedia.org/wiki/Low-dropout_regulator

[2] Josef Punčochár. BEN – technická literatura, 2002. *Operační zesilovače v elektronice* ISBN:8073000598

[3] AEC Council component technical committee defined qualification requorements [online]
Dostupné na internete:
http://www.aecouncil.com/AECDocuments.html

[4] Heat pump [online]
Dostupné na internete:
https://en.wikipedia.org/wiki/Heat_pump

[5] Thermoelectric Effect - an overview | ScienceDirect Topics [online]
Dostupné na internete:
https://www.sciencedirect.com/topics/chemistry/thermoelectric-effect

[6] PID regulátor [online]
Autor: Róbert Blažek
Dostupné na internete:
https://sk.wikipedia.org/wiki/PID_regul%C3%A1tor#/media/S%C3%BAbor:PID.png

[7] PID Theory Explained – National Instruments white paper[online]
Dostupné na internete:
https://www.ni.com/cs-cz/innovations/white-papers/06/pid-theory-explained.html

[8] Multicomp PRO MCPK2-19808AC-S Peltier cooler datasheet [online]
Dostupné na internete:
http://www.farnell.com/datasheets/3179065.pdf

[9] RSP-750 series datasheet [online]
Dostupné na internete:
http://www.meanwell.co/assets/data/RSP-750-spec.pdf

[10] STM32F205VG Product page [online]
Dostupné na internete:
https://www.st.com/en/microcontrollers-microprocessors/stm32f205vg.html

[11] Maxim Integrated MAX6954 LED Display driver datasheet [online]
Dostupné na internete:
https://datasheets.maximintegrated.com/en/ds/MAX6954.pdf

[12] Maxim Integrated MAX31865 RTD-to-Digital Converter datasheet [online]
Dostupné na internete:
https://datasheets.maximintegrated.com/en/ds/MAX31865.pdf

[13] Sensirion SHT3x-DIS datasheet [online]
Dostupné na internete:
https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/2_Humidity_Sensors/Datasheets/Sensirion_Humidity_Sensors_SHT3x_Datasheet_digital.pdf

[14] Vishay High Accuracy Amblient Light Sensor datasheet[online]
Dostupné na internete:
https://www.vishay.com/docs/84286/veml7700.pdf

[15] Analog Devices Rail-to-Rail OP Amp datasheet[online]
Dostupné na internete:
https://www.analog.com/media/en/technical-documentation/data-sheets/ADA4522-1_4522-2_4522-4.pdf

[16] Avago 8 MBd Low Input Current Optocoupler datasheet[online]
Dostupné na internete:
https://docs.broadcom.com/doc/AV02-0919EN

[17] IXYS GigaMOS HiperFET Power MOSFET datasheet[online]
Dostupné na internete:
https://m.littelfuse.com/~/media/electronics/datasheets/discrete_mosfets/littelfuse_discrete_mosfets_n-channel_trench_gate_ixfz140n25t_datasheet.pdf.pdf

[18] Murata NMD Series DC-DC Converters datasheet[online]
Dostupné na internete:
https://www.murata.com/products/productdata/8807031046174/kdc-nmd.pdf?1619494216000

Zoznam príloh

Príloha A: CD Médium, bakalárska práca v elektronickej podobe a zdrojové súbory.