TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Realizace tepelné regulace v uzavřeném prostoru

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

Autor práce: **Bc. Vít Bílek**
Vedoucí práce: Ing. Pavel Herajn

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Vít Bílek**
Osobní číslo:	**M14000186**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Automatické řízení a inženýrská informatika**
Název tématu:	**Realizace tepelné regulace v uzavřeném prostoru**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Navrhněte hardware, kterým bude možno udržovat konstantní nastavitelnou teplotu v uzavřeném prostoru (líheň, kvasné procesy, apod.).
2. Realizujte software (webové rozhraní v minipočítači), kterým bude možné měřit potřebné charakteristiky pro identifikaci, zadávat parametry PID regulátoru a následně spouštět samotný proces regulace.
3. Proveďte měření a identifikujte soustavu v prostředí MATLAB.
4. Navrhněte optimální PID regulátor.
5. Proveďte porovnání simulačního modelu s reálnou soustavou.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 40–50 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **BALÁTĚ, Jaroslav. Automatické řízení. 1. vyd. Praha: BEN - technická literatura, 2003, 663 s. ISBN 80-730-0020-2.**

[2] **KUPKA, Libor a Josef JANEČEK. Matlab: řešené příklady. 1. vyd. Lanškroun: SOŠ a SOU Lanškroun, 2007, 224 s. ISBN 978-80-239-9532-9.**

Vedoucí diplomové práce:	**Ing. Pavel Herajn** Ústav mechatroniky a technické informatiky
Konzultant diplomové práce:	**Ing. Tomáš Martinec, Ph.D.** Ústav mechatroniky a technické informatiky
Datum zadání diplomové práce:	**10. října 2015**
Termín odevzdání diplomové práce:	**16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -
L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 - školní dílo.

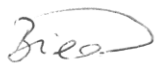
Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 13. 5. 2016

Podpis: 

Poděkování

Rád bych touto cestou poděkoval všem lidem, kteří mi byli zdrojem informací pro vytvoření této diplomové práce. Zejména děkuji svému vedoucímu práce panu Ing. Pavlu Herajnovi za odborné rady při návrhu HW, návrhu regulátoru a matematické analýze systému.

Rád bych také poděkoval své rodině za podmínky, které mi pro studium vytvořila.

Abstrakt

Tato diplomová práce se zabývá návrhem a realizací tepelné soustavy a regulací její vnitřní teploty. Cílem diplomové práce je návrh a realizace hardware a software, který bude schopen udržovat konstantní nastavitelnou teplotu v uzavřeném prostoru reálného modelu. Pro realizaci by měl být použit minipočítač, embedded systém, na platformě Linux.

K nastavování parametrů systému by mělo sloužit grafické uživatelské rozhraní, které by mělo pracovat přes webový prohlížeč. Pomocí rozhraní lze nastavit výkon akčních členů, konstanty regulátoru, žádanou hodnotu a pracovní režim systému. Zařízení by mělo umět pracovat jako zdroj konstantního výkonu akčních členů, v režimu PID regulátoru a v režimu automatického měření.

Dalším požadavkem pro uživatelské rozhraní je zaznamenávání měřených a nastavovaných hodnot pro jejich další zpracování. Výstupní soubor dat by měl být kompatibilní s prostředím Matlab.

Součástí práce je identifikační měření na tepelné soustavě, určení matematického modelu, jeho simulace a návrh konstant PID regulátoru.

Klíčová slova:

regulace teploty, tepelná soustava, identifikace, PID regulátor, webové rozhraní, Linux, Matlab, embedded systém, hardware, software

Abstract

This thesis is about temperature control on real temperature system. The thesis aims are design and create hardware and software that will be able to keep the temperature on set-point in closed space real model. For design will used embedded system on Linux platform.

For systems parameters settings is used graphical user interface that will worked in internet browser. By this interface user can setting actuator power, PID controllers constants, set-point and working mode. This device can worked as constant actuator power as PID controller and contain system for automatic measurement. Next request for user interface is saving process data for offline processing. Output files must be compatible with Matlab.

Thesis contain identification process for dynamic system, discover transfer functions model for this temperature system and discover PID controllers constants for this model.

Keywords:

temperature control, temperature system, system identification, PID controller, web interface, Linux, Matlab, embedded system, hardware, software

Obsah

Seznam ilustrací

Seznam grafů

Seznam zdrojových kódů

Seznam použitých zkratek

A/D	Analogově Digitální převodník
AJAX	Asynchronus JavaScript and XML, asynchronní zpracování webových stránek
C	Programovací jazyk
CD	Compact Disc, datové médium
CSV	Comma-Separated Values, souborový formát pro tabulková data
CRC	Cyclic Redundancy Check, cyklický redundandní součet
DP	Diplomová práce
DPS	Deska Plošných Spojů
FM	Fakulta mechatroniky, informatiky a mezioborových studií
GCC	GNU Compiler Collection, sada překladačů pro jazyk C
GPIO	General-Purpose Input/Output, vstupně výstupní piny embedded systému
HTML	HyperText Markup Language, značkovací jazyk pro tvorbu hypertextových dokumentů
HW	HardWare
I2C	Inter-Integrated Circuit, multimasterová počítačová sběrnice
ISBN	International Standard Book Number, mezinárodní standardní číslo knihy
JavaScript	Objektově orientovaný skriptovací jazyk
LAN	Local Area Network, místní síť
LED	Light Emitting Diode, dioda emitující světlo
MATLAB	MATrix LABoratory, výpočetní SW se skriptovacím jazykem
NTC	Termistor s negativní charakteristikou
PDF	Portable Document Format, Přenosný formát dokumentů
PHP	Personal Home Page, Hypertextový procesor, skriptovací programovací jazyk
PID	Proporcionálně Integračně Derivační regulátor
PNG	Portable Network Graphics, bezztrátový grafický formát rastrové grafiky
PWM	Pulse Width Modulation, pulsně šířková modulace
SPI	Serial Pheriperial Interface, sériové periferní rozhraní
SW	SoftWare
UART	Universal Synchronus/Asynchronus Receiver and Transmitter, Synchronní/Asynchronní sériové rozhranní
W1	One Wire, datová sběrnice

Úvod

Tato diplomová práce se zabývá návrhem a realizací tepelné soustavy a regulací její vnitřní teploty. Cílem této práce je realizace hardwaru a softwaru, který bude zajišťovat konstantní žádanou teplotu v uzavřeném prostoru, identifikace realizovaného systému a návrh optimálního PID regulátoru.

Soustava se skládá z termokomory, která slouží jako reálný model tepelné soustavy. V ní jsou umístěny členy pro měření teploty a členy pro její vytápění, které jsou ovládány embedded systémem, minipočítačem, na platformě Linuxu. V minipočítači je implementována aplikace pro řízení a ovládání soustavy, která může pracovat v několika pracovních režimech. Lze si zvolit mezi režimy manuálního nastavení výkonu akčních členů, regulací teploty, identifikačního měření, režimem OFF a dalších.

Dále je v minipočítači nainstalován webový server, na kterém běží webová stránka, která slouží jako uživatelské rozhraní mezi uživatelem a řídicí aplikací. Pomocí webové stránky lze zadávat požadavky na změnu pracovního režimu, konstant PID regulátoru, žádané hodnoty, hodnoty výkonu akčních členů nebo nastavit časový harmonogram pro identifikační měření. Součástí stránky je panel pro zobrazování aktuálních hodnot a interaktivní grafy pro zobrazování on-line průběhů měřených a nastavovaných veličin. Všechny tyto veličiny jsou zaznamenávány do souboru, který může být použit pro další off-line zpracování. Tento soubor je typu *csv*.

V první části práce je stručně popsána skupina počítačů nazývaná embedded systémy. Jsou zde popsány jejich výhody, nevýhody fyzické parametry a základní vybavení. Druhá část popisuje funkci a strukturu PID regulátoru, princip numerické integrace a z ní vycházející diskretizaci PID regulátoru. Třetí část je praktická a řeší realizaci hardwarové části systému, řídicí aplikace, webového rozhraní a komunikačního rozhraní pro přenos dat mezi aplikacemi. V poslední části je popsán postup identifikace diskutovaného systému, postup návrhu regulátoru a porovnání simulovaného a reálného regulačního pochodu. Dosažené výsledky jsou shrnuty v závěru.

Cíle diplomové práce

- Návrh a realizace reálného modelu tepelného systému
- Výběr minipočítače (embedded systému)
- Výběr akčních a měřicích členů
- Návrh a realizace elektronické části
- Identifikace realizovaného systému
- Návrh PID regulátoru pro řízení realizovaného systému
- Návrh a realizace řídicí aplikace, pomocí které bude možné regulovat teplotu dynamického systému PID regulátorem a provádět na něm identifikační měření
- Návrh a realizace webové stránky, pomocí které bude možné nastavovat parametry PID regulátoru, provádět měření na dynamickém systému a zobrazovat průběhy systémových veličin
- Návrh komunikačního rozhraní mezi webovou stránkou a řídicí aplikací

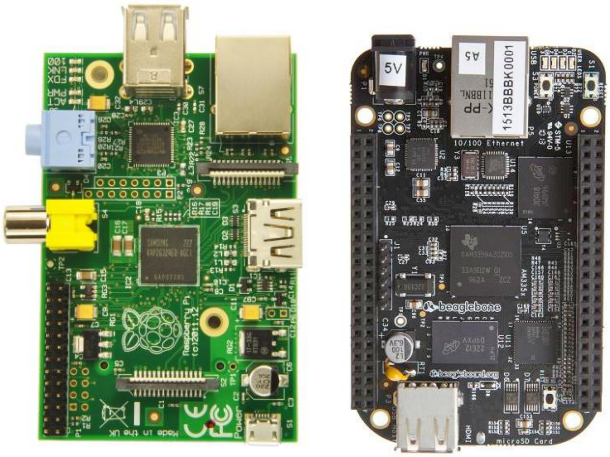
1. Minipočítače – embedded systémy

Minipočítač, neboli embedded systém, je jedno nebo vícejádrový procesor, ve kterém je nahrán operační systém, zpravidla Linux. Výhodou těchto zařízení je, že lze ovládat piny procesoru přímo z OS, což běžné počítače neumožňují.

S myšlenkou minipočítačů, embedded systémů, přišla na trh v roce 2001 britská firma Raspberry Pi Foundation. Ty však původně nebyly určeny k řízení robotů či regulaci teploty, ale především pro účely vzdělávání. Jejich cílem bylo, že minipočítač musí být tak levný, aby si ho mohl pořídit každý, kdo by se ho chtěl naučit obsluhovat a vytvářet na něm různé aplikace. Pořizovací cena byla 25 USD.

Dnes již existuje více firem, které tato zařízení vyrábějí. Jejich největší výhodou je nízká pořizovací cena, která se pohybuje, dle výrobce, mezi 25 a 50 USD. Prodávají se jako osazené DPS o velikosti kreditní karty. Spotřeba energie se pohybuje řádově v jednotkách wattů. Zpravidla jsou podporovány operačním systémem Linux, nejčastěji založeném na základě distribuce Debian.

Minipočítače se vyznačují tím, že obsahují svorkovnici, na kterou jsou vyvedeny piny procesoru tzv. GPIO. Pomocí těchto pinů může uživatel z operačního systému přímo ovládat hardwarové prvky. Operační systémy obsahují připravené knihovny pro práci s GPIO pro sběrnice, jako je *SPI*, *UART*, *I2C*, *CAN*, zpravidla mívají podporu PWM výstupů, případně A/D převodníků, atd. Jsou osazovány procesory ARM Cortex o jednom až čtyřech jádrech s frekvencí 1 GHz. Většina modelů obsahuje i modul pro připojení zařízení do LAN sítě.



Obr. 1 - Ukázka minipočítačů (Raspberry Pi typ B, BeagleBone Black rev c)

Informace pro tuto kapitolu byly čerpány ze zdroje [2].

2. PID Regulátor

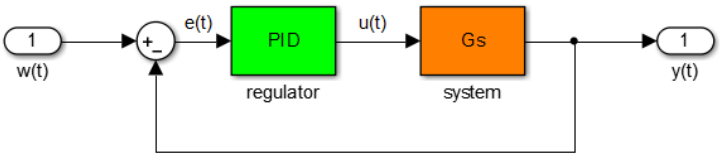
Regulátor je zařízení pro ovlivnění regulovaného systému, které zajistí dosažení a udržení požadovaného stavu.

PID regulátor patří mezi spojité regulátory. Používá se ve zpětnovazebním zapojení. Veličina vstupující do regulátoru se nazývá regulační odchylka $e(t)$. Regulátor na jejím základě určuje velikost akčního zásahu, podle kterého poté nastaví výstupní hodnotu. Regulátor reguluje systém s cílem eliminovat regulační odchylku nebo ji udržet v předepsaných mezích. Výstupní veličina regulátoru se značí $u(t)$ a nazývá se akční veličina. Je přivedena na vstup regulované soustavy, která je touto veličinou řízena.

Výstupní veličina řízené soustavy se nazývá regulovaná veličina $y(t)$. Zvolená hodnota, které má regulovaná veličina dosáhnout, se nazývá hodnota veličina $w(t)$. Ta může být buď po částech spojitá ve formě skoků, nebo spojitá ve smyslu harmonického signálu.

Zavedením zpětné vazby mezi výstupní regulovanou veličinou $y(t)$ a žádanou hodnotou $w(t)$ se získá regulační odchylka $e(t)$.

$$e(t) = w(t) - y(t) \tag{1}$$



Obr. 2 - Zpětnovazební regulační obvod

Rutinní povinností PID regulátoru je vypočítávat akční veličinu. Výpočet je určen součtem tří složek, ze kterých se regulátor skládá. V časové oblasti ho lze popsat následující funkcí.

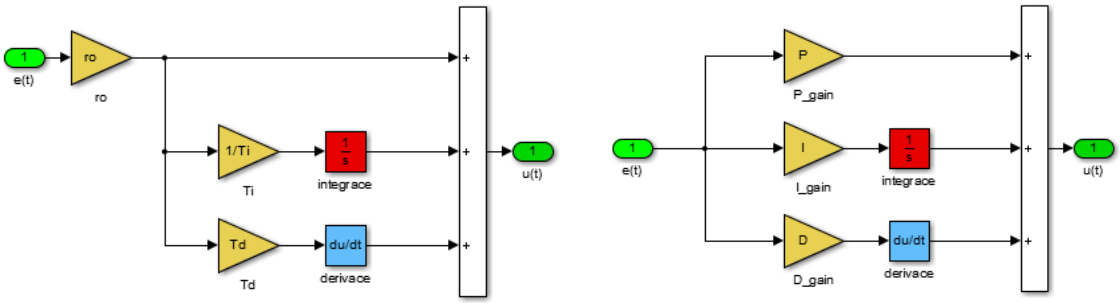
$$u(t) = r_0 \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right) \tag{2}$$

První složkou regulátoru je složka proporcionální. Jejím úkolem je zesilovat regulační odchylku daným zesílením r_0. Tato složka funguje tak, že čím větší je regulační odchylka, tím větší je akční zásah. Čím více se blíží regulovaná hodnota žádané, tím více se zmenšuje regulační odchylka, což způsobí postupné zmenšování akční veličiny. Proporcionální složka se vypočítá podle vztahu $P(t) = r_0 e(t)$ (3).

Druhou složkou je složka integrační. Jejím úkolem je urychlení regulačního pochodu. Čím bude regulační odchylka větší, tím rychleji se bude integrační složka měnit. Hlavní výhodou této složky je to, že v ideálním případě, po ustálení regulované veličiny na žádané hodnotě, působí na systém pouze tato složka. Ta pak soustavě dodává přesně tolik energie, kolik je potřeba k udržení konstantní hodnoty regulované veličiny. Dodaná energie pak odpovídá ztrátám regulované soustavy. Při nevhodně zvolené konstantě může integrační složka způsobit kmitavý průběh regulované veličiny. Integrační složku lze v časové oblasti popsat dle vztahu $I(t) = r_0 \frac{1}{T_i} \int_0^t e(\tau)\, d\tau$ (4).

Poslední složkou je derivační složka. Jejím úkolem je tlumení kmitů regulované veličiny a urychlení regulačního pochodu. Jelikož tato složka, jak už její název napovídá, vypočítává akční veličinu ze změny (derivace) regulační odchylky, je citlivá na šum. Proto se ve spoustě případů používá tzv. filtrace derivační složky. Čím je derivace průběhu regulační odchylky větší, tím větší je akční zásah. Obecně lze derivační složku v časové oblasti vypočítat podle vztahu $D(t) = r_0 T_d \frac{de(t)}{dt}$ (5).

Výsledná akční veličina se potom vypočítá jako součet všech tří složek regulátoru.



Obr. 3 - Vnitřní zapojení PID regulátoru - sériové a paralelní zapojení

Vnitřní zapojení PID regulátoru lze zakreslit dvěma základními způsoby. Na *obrázku 3* na levé straně je vnitřní zapojení sériové varianty PID regulátoru odpovídající rovnici regulátoru (2). Na pravé straně je zobrazeno paralelní zapojení PID

regulátoru. Každá složka je vynásobena příslušným zesílením odpovídající dané složce. Zapojení lze mezi sebou převádět dle vztahů $P = r_0 (6)$, $I = r_0 \frac{1}{T_i} (7)$, $D = r_0 T_d (8)$.

Ve spojité oblasti, v *Laplaceově* prostoru, lze regulátor vyjádřit pomocí přenosové funkce.

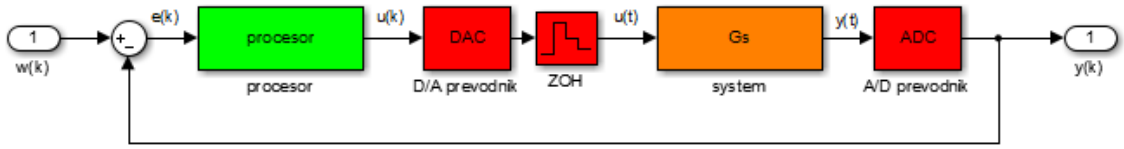
$$R(s) = r_0 \left(1 + \frac{1}{T_i}\frac{1}{s} + T_d s\right) = P + I\frac{1}{s} + Ds = \frac{Ps + I + Ds^2}{s} \qquad (9)$$

Informace pro tuto kapitolu a následující podkapitoly byly čerpány ze zdrojů [1],[6] a [9].

2.1 Diskretizace PID regulátoru

Při řízení dynamických systémů pomocí mikroprocesoru dochází k měření a výpočtu systémových veličin v určitých časových okamžicích. Procesor zpravidla nemá za úkol pouze vypočítávat regulační proces, ale také může například ovládat rozhraní mezi uživatelem a systémem jako je displej, klávesnice, potenciometr, tlačítka a podobně. Pokud by procesor v každém časovém okamžiku prováděl výpočty pro regulaci, nezbyl by mu pak žádný výpočetní výkon na ostatní operace.

Z tohoto důvodu se provádí výpočet akční veličiny v diskrétních krocích k s tzv. periodou vzorkování T_v. V každém kroku se provede měření regulované veličiny (vzorkování), výpočet regulační odchylky, výpočet akční veličiny a zapsání akční veličiny na příslušný výstup, respektive na vstup regulované soustavy.



Obr. 4 - Regulace v diskrétním čase

Zavedením vzorkování se z regulace ve spojitém čase stane regulace v čase diskrétním. Proto, aby se regulační pochod v diskrétním čase co nejvíc podobal regulačnímu pochodu ve spojité časové oblasti, je třeba najít odpovídající diskrétní podobu tohoto PID regulátoru. Tu lze získat diskretizací přenosu spojitého PID regulátoru (9).

Diskretizace regulátoru spočívá v tom, že se do přenosové funkce regulátoru ve spojité oblasti (9) za operátor s dosadí určitý diskrétní výraz, který vyplývá z numerické integrace. Tato myšlenka vychází z aproximace spojité integrační funkce

konkrétní diskrétní integrační metodou. V praxi to vypadá tak, že přenosová funkce spojitého integrátoru $\frac{1}{s}$ (10) se položí rovna přenosové funkci diskrétního integrátoru, například přenosové funkci integrující lichoběžníkovou metodou $\frac{Tv}{2}\frac{z+1}{z-1}$ (11).

$$\frac{1}{s} = \frac{Tv}{2}\frac{z+1}{z-1} \tag{12}$$

$$s = \frac{2}{Tv}\frac{z-1}{z+1} \tag{13}$$

Tento vztah (13) pro převod mezi s-rovinou a z-rovinou se nazývá *bilinerární* transformace. Tato transformace se používá u diskretizace nazývané *Tustinova aproximace* spojitého systému.

2.2 Numerická integrace

Integrace je matematická operace pro výpočet integrálu. Určitý integrál z funkce je roven obsahu plochy pod touto funkcí na daném intervalu. U spojité integrace se obsah vypočítává po nekonečně malých krocích, takže je výpočet úplně přesný.

U numerické integrace se výpočet obsahu provádí po krocích s určitou délkou, což do výpočtu vnáší jistou chybu. Existuje několik různých integračních metod. Rozdíl mezi nimi se projevuje především v jejich přesnosti.

Jednou z nejjednodušších metod je obdélníková metoda. Ta spočívá v tom, že se daný signál vzorkuje s periodou T_v. V době mezi dvěma vzorky je uvažován průběh vzorkované veličiny za konstantní, čímž vznikne schodová funkce, kde každý schod má délku Tv. Výsledkem integrace je pak součet obsahů všech vzniklých obdélníků.

$$y(k) = Tv\sum_{i=1}^{k} f(i) \tag{14}$$

Metodu lze rozdělit na levou a pravou obdélníkovou metodu. Název metody určuje směr integrace. Pokud byly získány dva vzorky $y(i-1)$ a $y(i)$, metoda se nazývá pravou obdélníkovou metodou, pokud je obsah dané plochy na daném intervalu určen jako $f(i) = y(i-1).\big(x(i) - x(i-1)\big)$. Pokud je obsah intervalu určen jako $f(i) = y(i).\big(x(i) - x(i-1)\big)$, pak se jedná o levou obdélníkovou metodu.

Přesnější integrační metodou je lichoběžníková metoda. Ta spočívá v tom, že se daný signál vzorkuje s periodou T_v a vždy po sobě jdoucí vzorky se pomyslně propojí interpolačním polynomem prvního řádu, čímž vznikne lichoběžník, ze kterého je určen jeho obsah. Výsledkem integrace je pak suma všech obsahů.

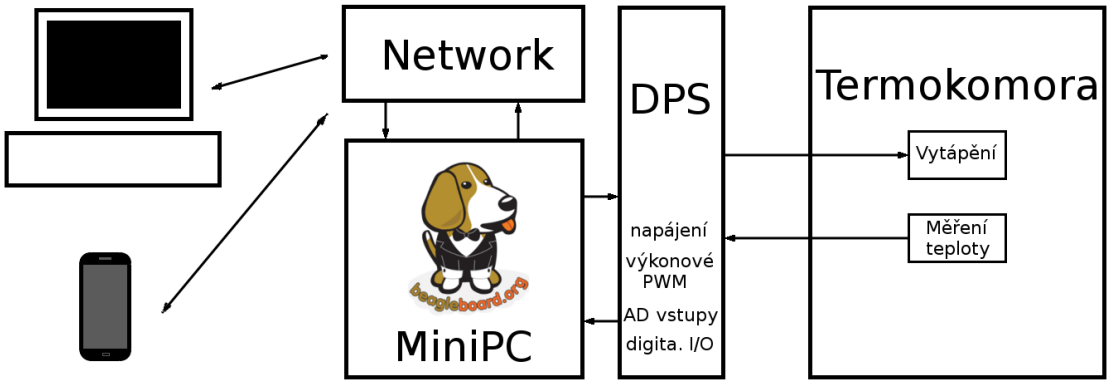
$$y(k) = \frac{T_v}{2}\sum_{i=1}^{k}(f(i) + f(i-1)) \tag{15}$$

Další možností diskrétní integrace je *Simpsnova* metoda. Tato metoda se vyznačuje tím, že dva po sobě jdoucí vzorky prokládá interpolačním polynomem druhého řádu, z tohoto obrazce je následně vypočten obsah. Výsledná hodnota integrace je suma všech obsahů.

$$y(k) = \frac{Tv}{6}\sum_{i=1}^{k}\left(f(i) + 4f\left(\frac{2i-1}{2}\right) + f(i-1)\right) \tag{16}$$

3. Praktická část 1 – Realizace tepelného systému

Tato kapitola se zabývá návrhem a zhotovením tepelné soustavy. Úkolem soustavy je ohřívat vzduch v jejím vnitřním prostoru na určitou teplotu, která je zvolena uživatelem. Principiální funkce systému je naznačena na *obrázku 5*.



Obr. 5 - Funkce systému, blokové schéma

Uživatel do vyhledávače v počítači, telefonu, tabletu apod. zadá IP adresu minipočítače, embedded systému, který je připojen do lokální sítě pomocí síťového kabelu. Do vyhledávače se načte stránka, která běží na minipočítači. Slouží jako panel pro nastavování a odečítání parametrů na měřicím přístroji nebo laboratorním zdroji. Skládá se ze tří záložek, z nichž každá plní konkrétní funkci.

Součástí minipočítače je i *řídicí aplikace*, jejímž úkolem je měření teploty v termokomoře a řízení výkonu akčních členů. Výkon akčních členů je řízen na základě zvoleného pracovního režimu. Informace o zvoleném pracovním režimu a parametry potřebné k jeho provedení jsou získávány od *webové aplikace* přes souborové rozhraní. Stejným způsobem probíhá i přenos dat opačným směrem. Výkon akčních členů je řízen pomocí PWM modulace.

Blok nazvaný DPS zastává dvě funkce. Hlavním významem tohoto bloku je zajišťování napájení pro minipočítač a akční členy. Slouží tedy jako zdrojová část. Druhým úkolem bloku je převod úrovně napětí PWM výstupu a realizace sběrnice One-Wire. Pro propojení DPS a minipočítače slouží vstupně výstupní piny tzv. GPIO.

Posledním blokem je termokomora, která představuje reálný model tepelné soustavy. Jsou v ní umístěna čidla pro měření teploty a akční členy pro vytápění vnitřního prostoru. Komora slouží také k uložení veškerého HW. Ten je umístěn

v oddělené části pod prostorem, ve kterém se provádí regulace teploty. Díky integraci všech komponent do této komory se systém jeví jako celek.

Zdrojem informací pro následující podkapitoly byly [3], [4], [5], [6], [7], [8] a [11].

3.1 Termokomora

Pro realizaci regulace na reálném tepelném systému byla navržena a zkonstruována termokomora, která je vyrobena z lamino desek a skla. Lamino je deska z dřevotřísky s plastovým povrchem. Tyto desky se často používají v truhlářství a dřevozpracujícím průmyslu. Výhodou tohoto materiálu je nízká pořizovací cena. Jeho nevýhodou je jistý akumulační jev, který se projevuje při vytápění komory a vnáší tak do systému určitou dynamiku.

Termokomora má tvar hranolu o rozměrech 450 x 250 x 270 mm. Boky, dno, zadní stěna a víko jsou z dřevotřísky. Přední stěna je z transparentního skla. Víko je připevněno dvěma panty. Do komory tak lze kdykoliv cokoli vložit nebo z ní vyjmout. Ve víku jsou vyříznuty tři otvory, ve kterých jsou upevněny keramické objímky. V každé objímce je nainstalována klasická žárovka o výkonu 100 W. Na bočních stranách komory jsou umístěny digitální snímače teploty.

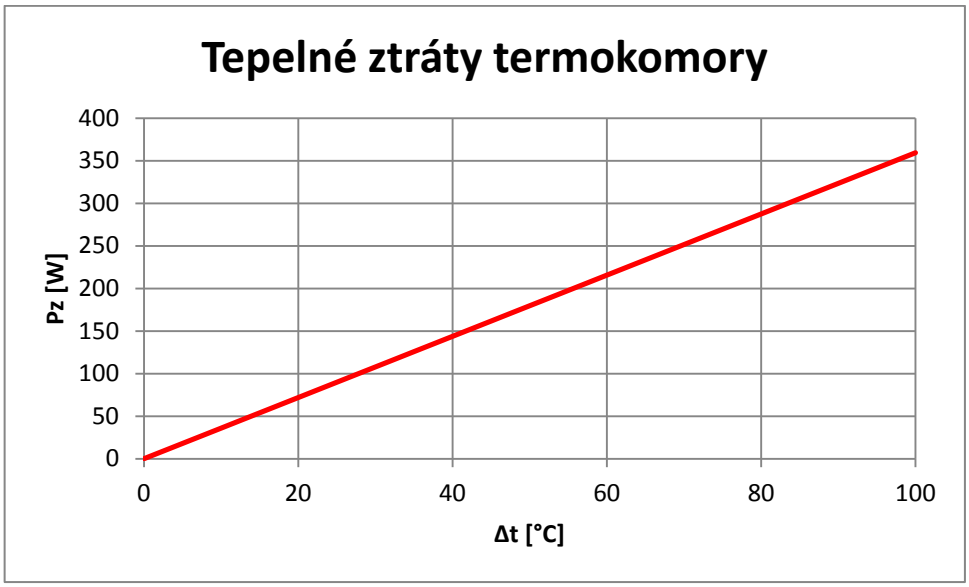
Komora má dvojité dno. Pod prostorem, který je určený k regulaci teploty, je ještě jeden prostor o rozměrech 450 x 250 x 50 mm. Tento prostor slouží k uložení veškerých elektronických částí. Oba prostory od sebe odděluje dno, skrze něj jsou vedeny potřebné vodiče z jednoho prostoru do druhého. Zadní stěnou je vyveden napájecí kabel s vidlicí a síťový kabel s konektorem RJ45 pro připojení do lokální sítě.

Položka	Součinitel tepelné vodivosti $\lambda\ [W/m.K]$	Tloušťka stěny $d\ [m]$	Součinitel prostupu tepla konstrukce $U\ [W/m^2.K]$	Povrch kontaktní plochy $S\ [m^2]$
Dřevotříska	0,11	0,018	6,11	0,45
Sklo	0,023	0,003	7,67	0,11

Tabulka 1 - Vlastnosti použitých materiálů

Na základě tepelné vodivosti a tloušťky stěn byl vypočten tepelný odpor konstrukce termokomory, podle vztahu $R_T = \frac{d}{\lambda}$ (17), na základě kterého byl určen součinitel prostupu tepla konstrukce jako $U_T = \frac{1}{R_T}$ (18). Pomocí této veličiny

pak lze vyjádřit, jaký vliv na tepelné ztráty komory bude mít rozdíl vnější a vnitřní teploty $\Delta t = t_{in} - t_{out}$ (19). Ztráty pro použité materiály se určí pomocí vztahu $P_z = U.S.\Delta t$ (20). Celková ztráta se rovná součtu ztrát všech daných ploch. Jejich průběh je naznačen v *grafu 1*.



Graf 1 - Tepelné ztráty termokomory

Z tohoto grafu je patrné, že pomocí použitých akčních členů lze dosáhnout vnitřní teploty maximálně o 82 °C vyšší, než jaká je teplota okolí. Reálné chování systému odpovídá tomuto tvrzení.

Obr. 6 - Termokomora

3.2 Minipočítač – Embedded systém

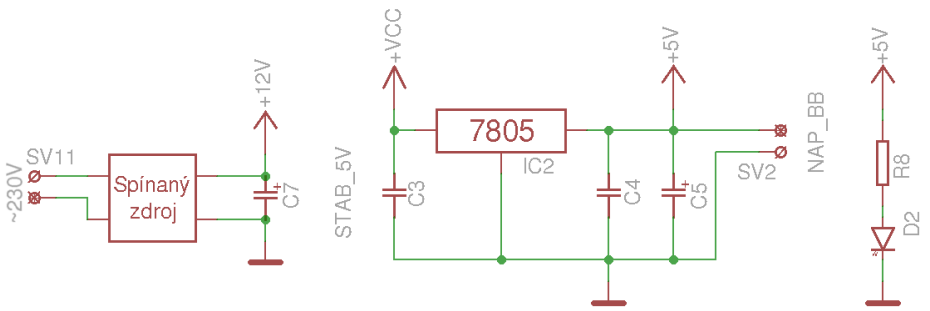
Úkolem minipočítače je řídit soustavu a ovládat hardwarové části, jako je vyčítání teplotních čidel, řízení výkonu PWM výstupu, atd. K těmto úkonům by spolehlivě postačil mikroprocesor, ovšem k získání dat z mikroprocesoru by bylo nutné využít nějakou sběrnici. V embedded systému se tento problém dá vyřešit například zobrazením dat pomocí webové stránky.

Pro řešení této aplikace byl zvolen minipočítač *BeagleBone Black Rev C* od firmy Texas Instruments. Má jedno-jádrový procesor *ARM Cortex-A8* o frekvenci 1 GHz, což je pro tuto aplikaci plně dostačující. Velkou výhodou zvoleného minipočítače je podpora sběrnice *One-Wire*, *PWM* výstupů, dvanáctibitové *A/D* převodníky a integrovaná *síťová karta*.

V minipočítači jsou implementovány dvě aplikace, které dohromady tvoří softwarovou část soustavy. První aplikace běží v reálném čase v nekonečné smyčce. Slouží k měření teploty, nastavování akční veličiny a zastává funkci regulátoru. Druhá aplikace vytváří grafické rozhraní pro obsluhu první aplikace Podrobný popis funkce obou aplikací je popsán v kapitole *3.6 Řídicí aplikace* a *3.8 Webová stránka, webová aplikace*.

3.3 Elektronická část

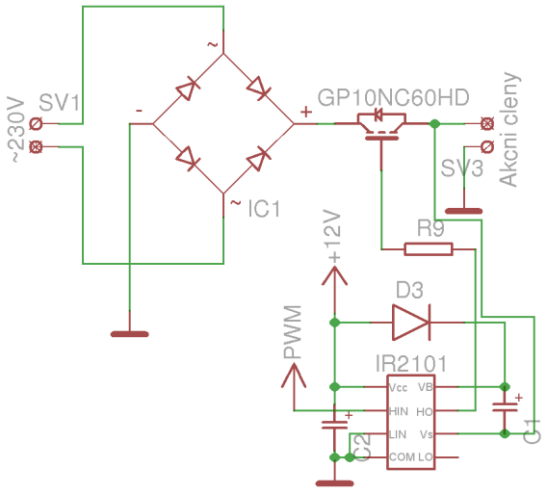
Úkolem této části je zajistit napájení pro minipočítač a ostatní elektroniku a také vytvořit rozhraní mezi minipočítačem a hardwarovými komponentami, jako jsou akční členy a teplotní čidla.



Obr. 7 - Zdrojová část

Elektroniku lze rozdělit do třech částí. První částí je *zdrojová část*, která je naznačena na *obrázku 7*. V této části schématu je nakreslen blok spínaného zdroje s výstupním napětím 12 V a stabilizátor, který vytváří úroveň napětí pro napájení minipočítače. Spínaný zdroj byl zakoupen jako komerční zařízení.

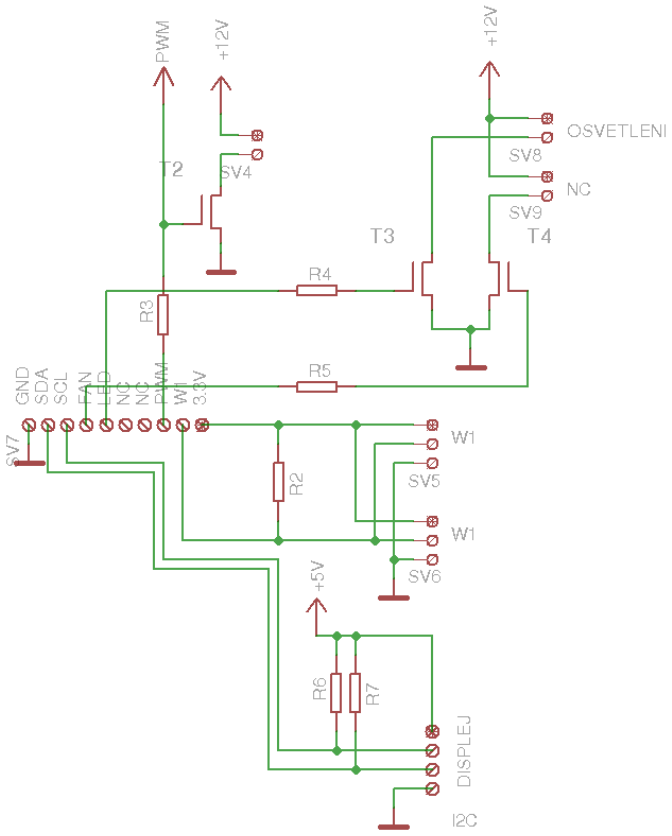
Druhou částí elektroniky je zdroj napájení akčních členů a obvod pro řízení jejich výkonu. K napájení je použito usměrněné síťové napětí 230V. Toto napětí je následně spínáno pulsně šířkovou modulací, k čemuž slouží IGBT tranzistor GP10NC60HD, který je řízen tranzistorovým driverem IR2101. Tento obvod zajišťuje rychlý přechod z úplného otevření do úplného zavření a naopak, což zajišťuje minimální ohřívání tranzistoru. Vstup tohoto obvodu je buzen PWM výstupem minipočítače.



Obr. 8 - Napájení akčních členů, výkonové PWM

Třetí část pracuje jako rozhraní mezi minipočítačem a elektronikou. Přes svorkovnici SV7 se připojí DPS k minipočítači, čímž je zajištěno napájení minipočítače a propojení zemí. Od svorky W1 je veden datový vodič na svorkovnici SV5 a SV6. K nim jsou připojeny teploměry Dallas 18b20. Mezi vodičem W1 a napájecím vodičem 3,3 V je zapojen pull-up rezistor. Pro případné připojení displeje jsou z minipočítače vyvedeny vodiče sběrnice I2C, SDA a SCL, které jsou přes pull-up rezistory připojeny k napětí 5 V. Jako poslední jsou vyvedeny tři I/O výstupy. První výstup řídí tranzistorový driver. Minipočítač generuje PWM signál, kterým je spínán tranzistor T2, který pouze zvyšuje úroveň napětí, z 3,3 na 12 V. Druhý výstup slouží

k ovládání doplňkového osvětlení termokomory. Poslední výstup je nevyužit, ale mohl by sloužit například jako PWM výstup pro řízení ventilátoru.



Obr. 9 - Rozhraní mezi minipočítačem a elektronikou

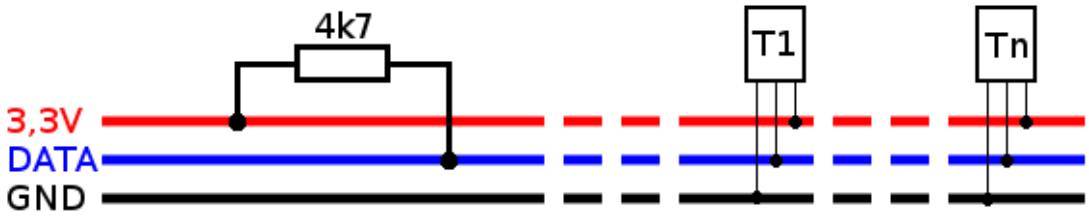
3.4 Snímače pro měření teploty

Pro měření teploty byla uvažována teplotní čidla odporová typu NTC a digitální Dallas 18B20. Výhodou NTC oproti teploměrům Dallas je mnohonásobně větší opakovatelnost měření za určitou časovou jednotku, což je způsobeno vlastnostmi sběrnice, na které jsou umístěny digitální teploměry. Nevýhodou NTC je jejich nelineární charakter, což zvětšuje nepřesnost měření při vyšších teplotách. Teploměry NTC se musí kalibrovat, teploměry Dallas provádí autokalibraci. Pro měření teploty v dané soustavě byly zvoleny teplotní čidla firmy Dallas typu 18B20.

Tyto teploměry komunikují přes sběrnici One-Wire (W1), která se vyznačuje tím, že pro přenos dat používá pouze jeden vodič. Existují dvě varianty zapojení těchto zařízení ke sběrnici a to dvou nebo třívodičová varianta. Přínosem dvouvodičové varianty je fakt, že pro přenos dat používá pouze dva vodiče a to datový a GND.

K napájení teploměru se využívá energie akumulovaná z datového vodiče. Nevýhodou je, že teploměr potřebuje při měření teploty větší proud, než jaký může téct přes pin procesoru a po dobu měření tak musí být datový vodič připnut přímo k napájecímu zdroji, čímž se komunikace komplikuje. Třívodičová varianta obsahuje navíc napájecí vodič, kterým zajistí dostatečný proud po celou dobu provozu.

V této soustavě se používají teploměry typu THT, které jsou uloženy v pouzdře *TO92*. Pro komunikaci je použita třívodičová sběrnice. Jejich zapojení je naznačeno na *obrázku 10*.



Obr. 10 - Zapojení teploměrů Dallas 18B20

Všechny teploměry jsou zařízení typu *slave* s unikátním sériovým číslem daným výrobcem. Podle sériového čísla je lze jednoznačně identifikovat.

Pro embedded systémy BeagleBone existují již vytvořené knihovny, pomocí kterých lze snadno sběrnici obsluhovat. V *Příloze A - Skript pro inicializaci sběrnice One-Wire* je kód, který po přeložení vytváří knihovnu pro obsluhu sběrnice W1. Ve zdrojovém kódu knihovny byl změněn pin, na který má být připojen datový vodič. Poté byl kód knihovny přeložen jako soubor *BB-W1.00A0* a umístěn do adresáře /lib/firmware. V tomto adresáři jsou soustředěny všechny systémové ovladače. Pro aktivaci knihovny musí být do inicializačního skriptu, který je spouštěn při startu operačního systému, vložen příkaz

echoBB-W1:00A0 > /sys/devices/bone_mng.9/slots,

pomocí kterého se vytvoří adresář */sys/bus/w1*. V tomto adresáři je podadresář *devices*, jehož obsahem jsou další podadresáře se jmény, která odpovídají sériovým číslům aktuálně připojených teploměrů ke sběrnici. Každý ze zmíněných podadresářů obsahuje soubor *w1_slave*. Přečtením tohoto souboru se vyšle požadavek danému teploměru na vrácení teploty. Ten pošle zpět několik bytů, které jsou zobrazovány jako obsah souboru. Příkaz pro čtení souboru vrací řetězec znaků, který obsahuje přijaté datové byty v HEXA formátu, výsledek kontrolního součtu CRC=YES/NO a naměřenou hodnotu, např. 21000 ~ 21.0 °C.

```
cat /sys/bus/w1/devices/10-0000001239/w1_slave
07 01 4b 46 7f 09 10 da : crc=da YES
07 01 4b 46 7f 09 10 da t=16437
```

Ukázka kódu 1 - Vyčtení teploměru Dallas 18b20

3.5 Akční členy, nastavení výkonu akčních členů

Pro vyhřívání termokomory bylo testováno několik akčních členů, jako například topné fólie, halogenové žárovky, topná spirála, atd. U zmíněných členů se projevoval buď nedostatečný, nebo naopak příliš velký topný výkon. Akční členy s malým výkonem, jako jsou halogenové žárovky a topné fólie, nedokázaly zvýšit teplotu komory, po ustálení odezvy na skok maximálního nastavení akční veličiny, ani o 20 °C. Akční členy s velkým výkonem, jako například topná spirála, naopak způsobovaly po několika sekundách doutnání termokomory a roztékání jejího povrchu. Jako nejoptimálnější akční členy pro tuto aplikaci se jeví klasické žárovky, jejichž výhodou je velice nízká pořizovací cena a jednoduché řízení výkonu.

Pro vytápění termokomory byly zvoleny tři klasické žárovky o jmenovitém výkonu 100 W, které jsou zapojeny paralelně a jejich maximální výkon tak dosahuje 300 W. Jako zdroj napájení se využívá usměrněné síťové napětí pomocí *Gretzova* diodového můstku *BR3510W*. Výkon žárovek je řízen pulsně šířkovou modulací, generovanou minipočítačem, pomocí IGBT tranzistoru *GP10NC60HD*. *Gate* tranzistoru je řízený spouštěcím obvodem *IR2101S*, tranzistorovým driverem, který zajišťuje velice rychlý přechod z otevřeného do uzavřeného stavu, což snižuje ohřívání spínacího tranzistoru. Vstup tranzistorového driveru je přiveden na PWM výstup minipočítače.

Embedded systémy *BeagleBone* obsahují knihovnu pro nastavení určitých GPIO, na PWM výstupy. Inicializace této knihovny probíhá při spouštění OS. Provádí ji inicializační skript. Knihovna pro ovládání PWM výstupů je v adresáři /lib/firmware Inicializace PWM provede příkaz *echo BB_PWM_P27 > /sys/devices/bone.mng9/slots*, který v adresáři /sys/devices/omp3/ vytvoří podadresář *pwm_test_P27*, ve kterém vznikne několik konfiguračních souborů. Pomocí konfiguračního souboru *period* lze nastavit periodu, s jakou má PWM výstup pracovat. Pomocí souboru *duty* se nastavuje střída, respektive doba, po kterou je výstup v logické „0“. Hodnoty těchto parametrů jsou udávány v nanosekundách. Změnou konfiguračního souboru *run*

z hodnoty 0 na hodnotu 1 se aktivuje výstup, jehož signál bude odpovídat nadefinovaným parametrům.

Jelikož má síťové napětí průběh $230.\sin(2\pi\ 50)$ (21), usměrněním z něj vznikne průběh $\sqrt{2}\ 230|\sin(2\pi\ 50)| - u_p$ (22). Usměrněním signálu se ze záporných period stanou kladné, což způsobí, že se signál začne opakovat už po polovině periody a perioda signálu se tedy zdvojnásobí. Lze tedy tvrdit, že akční členy jsou napájeny stejnosměrným napětím, které se mění s periodou *100 Hz*. Perioda pulsně šířkové modulace je zvolena *10 kHz*. To znamená, že za jednu periodu napájecího napětí akčních členů se provede sto period PWM modulace. Z toho vyplývá, že výkon akčních členů lze poměrně spojitě měnit.

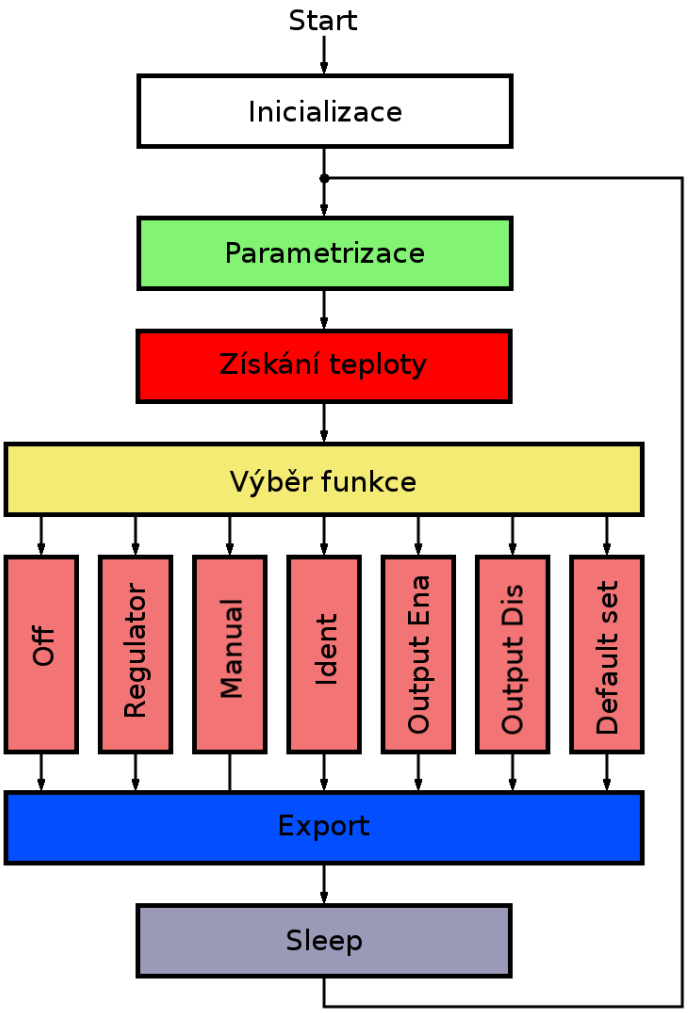
3.6 Řídicí aplikace

K ovládání akčních členů, měření teploty digitálními teploměry, výpočtu regulační odchylky a k výpočtu akční veličiny pomocí PID regulátoru byla navržena a naprogramována aplikace, která řeší všechny zmíněné problémy. K vývoji aplikace bylo použito prostředí Eclipse Mars, které umožňuje vyvíjení aplikace na vzdáleném počítači. Vzdálený přístup probíhá přes *SSH*. Aplikace je napsána v jazyce C. Výhodou tohoto jazyka je jeho jednoduchost, elegance a podpora Linuxu. Pro překlad programu byl použit překladač GCC, pomocí kterého lze překládat program přímo z terminálu pomocí příkazu *gcc aplikace.c –o aplikace.exe -lm*. Tato aplikace je dále označována pojmem *řídicí aplikace*.

Funkce *řídicí aplikace* je naznačena na *obrázku 11*. *Řídicí aplikace* je spuštěna současně se spuštěním operačního systému. Spuštění zajišťuje příkaz vložený v inicializačním skriptu */home/debian/PID/pid.exe*. Inicializační skript se jmenuje *rc.local* a je umístěn v adresáři */etc/init.d*.

Po spuštění aplikace dojde k inicializačním záležitostem jako je deklarace proměnných, definice konstant, inicializace W1, inicializace PWM, aktivace PWM , k defaultnímu nastavení konfiguračního souboru, k vymazání historie naměřených hodnot a k zahájení měření času. Po vykonání inicializačních úkonů přechází program do nekonečné smyčky, ve které se nachází až do odpojení počítače od napájecího napětí.

Program pracuje v cyklech s určitou časovou periodou T_S. Pro práci s časem je použita knihovna *time.h*. Na začátku každého cyklu je pomocí funkce *get_time()* získán aktuální čas (relativní systémový čas), který je zaznamenán do proměnné *time_register*. Po získání časového razítka se provede parametrizace.



Obr. 11 - Blokové schéma řídicí aplikace

Úkolem parametrizace je nastavit aplikaci tak, aby pracovala podle uživatelem zvolených parametrů. Tyto parametry jsou uloženy v konfiguračním souboru */home/debian/interface*, který slouží jako rozhraní mezi touto a webovou aplikací. K získání parametrů se využívá funkce *get_interface_params()*. V této funkci se provede načtení obsahu konfiguračního souboru jako řetězce. Ze získaného řetězce je selektován podřetězec, který leží před příslušným ukončovacím znakem, zbytek řetězce je eliminován. Vybraný podřetězec je rozdělen podle definovaných separátorů, čímž se získá pole konfiguračních parametrů. Prvky pole jsou poté přetypovány

na příslušné datové typy odpovídající konkrétní proměnné, do které je daný parametr zapsán. Těmito parametry je pracovní režim, konstanty PID regulátoru, žádaná hodnota teploty a požadovaný výkon akčního členu. Princip souborového rozhraní je podrobně popsán v kapitole *3.7 Rozhraní mezí webovou a řídicí aplikací*. Pro práci se soubory je použit hlavičkový soubor *fcntl.h*.

Po provedení parametrizace programu je změřena teplota uvnitř termokomory. Je měřena digitálními teploměry Dallas 18B20. Měření teploty se provádí přímo v teploměru, který její hodnotu pošle po sběrnici nadřazenému systému. K získání teploty se používá funkce *w1_get_temp(W1_T1)*, kde *W1_T1* je konstanta, která určuje teploměr, kterým se bude měřit. Vyčtení teploty z teploměru se provede přečtením souboru *w1_slave* v adresáři daného teploměru. Cestu k tomuto souboru definuje konstanta W1_Tx, kde x je číslo teploměru. Přečtením tohoto souboru je získán řetězec o dvou řádcích. Na obou řádcích jsou zobrazeny přijaté byty nesoucí informaci o teplotě ve formátu HEXA. Jsou odděleny mezerou. Poslední atribut prvního řádku uvádí shodu kontrolního součtu přijatých dat. CRC=YES pro korektní přenos a CRC=NO pro nekorektní přenos. V případě, že informace o teplotě byla přijata bez chyby, vrací funkce w1_get_temp(W1_T1) aktuální teplotu, v opačném případě vrací hodnotu minulou.

```
/* select mode */
switch(mode)
{
	case ROUTINE_OFF: routine_off();
		break;
	case ROUTINE_REGULATOR: routine_regulator(w,y);
		break;
	case ROUTINE_MANUAL: routine_manual(u);
		break;
	case ROUTINE_IDENTIFICATION: routine_identification(time,y,u);
		break;
	case SET_DEFAULT: set_default();
		break;
	case OUTPUT_ENABLE: pwm_enable();
		break;
	case OUTPUT_DISABLE: pwm_disable();
		break;
	case SHUT_DOWN_PRG: return 0;
		break;
	default: printf("ERR - mode %i !!!\n", mode);
		break;
}
```

Ukázka kódu 2 - Výběr pracovního režimu

Po změření teploty je vybrán pracovní režim. Informace o aktuálním pracovním režimu reprezentuje proměnná *mode*.

Na základě proměnné *mode* se provede příslušná akce. Tou mohou být buď pracovní režimy, jako je režim *OFF*, *regulátor*, *manuál*, *identifikace* nebo jednorázové akce, jako je *defaultní nastavení konstant* a *aktivace/deaktivace PWM výstupu*. Pomocí příkazu *switch* je vybrána odpovídající funkce. Pracovní režimy jsou podrobně popsány v podkapitolách *3.5.1 - 3.5.15*.

Po provedení příslušné akce v daném režimu jsou určitá data exportována do souboru */home/debian/data_logger.csv*. Pomocí souborového rozhraní jsou důležité hodnoty jako $y(t)$, $u(t)$, $w(t)$ a $e(t)$, ale i aktuálně nastavené parametry regulátoru, poskytnuty webové aplikaci, která je používá k online vizualizaci. Každý záznam, *log*, obsahuje časovou značku, která jednoznačně určuje čas vytvoření záznamu. Souborové rozhraní je popsáno v kapitole *3.7 Rozhraní mezi webovou a řídicí aplikací*

Na konci každého cyklu se aplikace uspí až do času dalšího vzorkovacího okamžiku. K uspání slouží funkce *time_wait_for_sampletime(time_register)*, kde vstupní parametr *time_register* je časová značka získaná na počátku cyklu.

```
/* wait for next sample */
int time_wait_for_sampletime(clock_t time_reg)
{
	int ms;
	clock_t time_now;

	time_now = get_time();

	while((ms = difftime_ms(time_now,time_reg)) < SAMPLING_T)
	{
		sleep_ms(SLEEP_FOR_SAMPLE);
		time_now = get_time();
	}

	return ms;
}
```

Ukázka kódu 3 - Uspání aplikace do doby dalšího vzorku

Ve funkci *time_wait_for_sampletime()* je implementována smyčka, která určuje časovou diferenci mezi současným časem a časem zaznamenaným na začátku cyklu. Pokud je časová diference menší než perioda vzorkování, *SAMPLING_T*, aplikace se uspí na dobu *SLEEP_FOR_SAMPLE*. Po probuzení je opět získán aktuální čas a znovu je otestována podmínka časové diference. Pokud je větší nebo rovna konstantě *SAMPLING_T*, smyčka se ukončí a provede se další cyklus. Tato funkce vrací dobu trvání posledního vzorkovacího kroku.

3.6.1 Pracovní režim OFF

Do tohoto režimu se zařízení dostane vždy po spuštění. Tento režim slouží jako *standby* mód, ve kterém se provádí pouze měření teploty. Výkon akčních členů je nulový, akční členy jsou tedy neaktivní. Využívá se zde pouze jediného konfiguračního parametru, *mode*. Tento režim zajišťuje funkce *routune_off()*.

3.6.2 Pracovní režim regulátoru

V tomto pracovním režimu je výkon akčních členů nastavován automaticky. Je zde implementován PID regulátor, který vypočítává akční veličinu. Konstanty regulátoru *P*, *I*, *D* a žádanou hodnotu teploty *W* si volí uživatel pomocí webové aplikace. Aplikace je získává z konfiguračního souboru. Pro výpočet akční veličiny se v tomto pracovním režimu využívá funkce *routine_regulator(w,y)* kde je vstupní parametr *w*, který udává žádanou hodnotu a *y* parametr, který měřenou teplotu ve vnitřním prostoru termokomory.

PID regulátor vypočítává akční veličinu z regulační odchylky $e(t)$. Regulační odchylka se vytvoří zavedením záporné zpětné vazby z výstupu na vstup, obecně tedy platí: $e(t) = w(t) - y(t)$ (1). Reálně však probíhá regulační proces v diskrétním čase, to znamená po určitých diskrétních krocích. Regulační odchylka nabývá pouze hodnot $e(k.Ts)$, kde k je číslo kroku, respektive počet cyklů řídicí aplikace, a Ts je délka kroku, tedy perioda vzorkování. Regulační odchylka se tedy reálně určuje jako: $e(k) = w(k) - y(k)$ (23). Z tohoto důvodu musí být PID regulátor diskretizován, viz. kapitola *2.1 Diskretizace PID regulátoru*.

Protože se při výpočtu proporcionální složky spojitého regulátoru využívá pouze současné regulační odchylky, je metoda výpočtu diskrétní verze regulátoru stejná $p_{term} = P * e$ (24).

Pro diskretizaci integrační složky byla použita *Tustinova* aproximace. Ta spočívá v tom, že se za operátor *s*, u integrátoru, dosadí výraz $s = \frac{2}{Tv}\frac{z-1}{z+1}$ (13). Integrační složka se pak vypočte jako $i_{term} = i_{past} + I * \left(\frac{Ts}{2}\right) * \left(e + e_{past}\right)$ (25), kde *i_term* je současná hodnota integrační složky *i_past* je minulá hodnota integrační složky, *e* je současná hodnota regulační odchylky a *e_past* je minulá hodnota regulační odchylky.

Výpočet derivační složky vychází z přenosové funkce filtrované derivace $D(s) = \frac{r_0 T_v s}{(1+s\alpha T_v)}$ (26), která je diskretizována *Tustinovou* aproximací. Parametr α slouží k ladění citlivosti filtrace. Volí se od 0,05 do 0,2. Výpočet derivační složky se provádí následovně, pro α = 0,05: $d_{term} = \frac{2D}{T_s+2\alpha\frac{D}{P}}\left(e - e_{past}\right) + \frac{2\alpha\frac{D}{P}-T_s}{T_s+2\alpha\frac{D}{P}} d_{past}$ (27).

```
/* regulator */
int routine_regulator(float w, float y)
{
	int output_val;
	float p_term,i_term,d_term;
	static float i_past, d_past, e_past;

	if(mode_changed_detect() > 0)
	{
		i_past = 0;
		d_past = 0;
	}

	if(I <= 0)
		i_past = 0;
	if(D <= 0)
		d_past = 0;

	if(P == 0)
	{
		pwm_set_output(PWM_MIN);
		return 0;
	}

	e = w - y;

	p_term = P*e;
	i_term = i_past + I*((float)Ts/2)*(e + e_past);
	d_term = (e-e_past) * (2*D)/(Ts+2*D*ALPHA/P);
	d_term += d_past * (2*D*(float)ALPHA/P - (float)Ts)/(2*D*(float)ALPHA/P + (float)Ts);

	output_val = (int)((p_term + i_term + d_term)*PID_PRESCALE);

	e_past = e;
	d_past = d_term;

	/*wind-up cutting*/
	if(output_val > PWM_MAX)		// pokud je splnena podminka neintegruje se
	{
		i_term = i_past;
		output_val = PWM_MAX;
	}
	if(output_val < PWM_MIN)
	{
		i_term = i_past;
		output_val = PWM_MIN;
	}
	i_past = i_term;

	/* set output */
	u = output_val;
	pwm_set_output(output_val);
	return 0;
}
```

Ukázka kódu 4 - Algoritmus PID regulátoru s ošetřením wind-up efektu a saturace

Výsledná hodnota akční veličiny $u(k)$ se vypočte jako součet všech tří složek. Pro zlepšení regulačního pochodu bylo do regulátoru implementováno dynamické omezení akční veličiny proti *wind-up* efektu. Tento efekt se může projevovat u systémů s omezeným rozsahem akční veličiny. Projevuje se tehdy, pokud akční veličina překročí jednu z mezních hodnot vstupního rozsahu. Po překročení meze se stále provádí integrace regulační odchylky. Integrační složka se tak neustále zvětšuje, ovšem akční veličina je omezena saturací. Při dosažení žádané hodnoty bude velikost samotné integrační složky přesahovat saturační mez a regulovaná veličina bude růst. Po překročení žádané hodnoty začne regulační odchylka nabývat záporných hodnot, což způsobí zmenšování integrační složky.

Pro odstranění tohoto jevu je v každém kroku porovnávána velikost akční veličiny s mezními hodnotami vstupního rozsahu PWM. Pokud akční veličina překročí jednu z mezí, neprovede se integrace regulační odchylky a integrační složka bude mít hodnotu předchozího kroku. Pokud tento jev nastane, akční veličina bude omezena na příslušnou mezní hodnotu rozsahu.

3.6.3 Manuální pracovní režim

Tento režim slouží k ručnímu nastavení výkonu akčních členů, kde si uživatel pomocí webové aplikace zvolí hodnotu od 0 do 100 %. V tomto režimu se využívá pouze konfigurační parametr U (viz kapitola *3.6 Rozhraní mezi webovou a řídicí aplikací*), který udává hodnotu výkonu akčních členů.

```
/* manual */
void routine_manual(int intenzity)
{
        mode_changed_detect();

        if(intenzity < PWM_MIN)
                intenzity = PWM_MIN;

        if(intenzity > PWM_MAX)
                intenzity = PWM_MAX;

        pwm_set_output(intenzity);
}
```

Ukázka kódu 5 - Funkce pro manuální nastavení výkonu akčních členů

Pro ošetření vstupního rozsahu akčních členů je parametr ošetřen saturací. Ošetřená hodnota je poté nastavena na vstup akčních členů.

3.6.4 Pracovní režim identifikace

Režim identifikačního měření slouží k automatickému měření na tepelném systému. V principu funguje stejně jako režim manuální. Změny akční veličiny řídí webová aplikace. Oproti manuálnímu režimu se průběhy akční veličiny a měřené veličiny ukládají do speciálního souboru dat, který je určen k offline zpracování.

Uživatel na webové stránce vyplní harmonogram pro měření (timetable). Webová aplikace podle něho pak nastavuje skoky žádané veličiny v řídicí aplikaci.

Při aktivaci tohoto režimu se vždy vytvoří nový soubor pro zaznamenávání těchto dat. Ukládá se do něho čas záznamu, velikost akční veličiny a změřená teplota v termokomoře. Čas, který je zapisován do souboru, je měřen od spuštění příslušného režimu, takže každý soubor tohoto typu začíná od času nula.

3.6.5 Nastavovací funkce set default, enable output, disable output

Tyto funkce byly implementovány pro doplnění základních vlastností systému k implementovaným režimům.

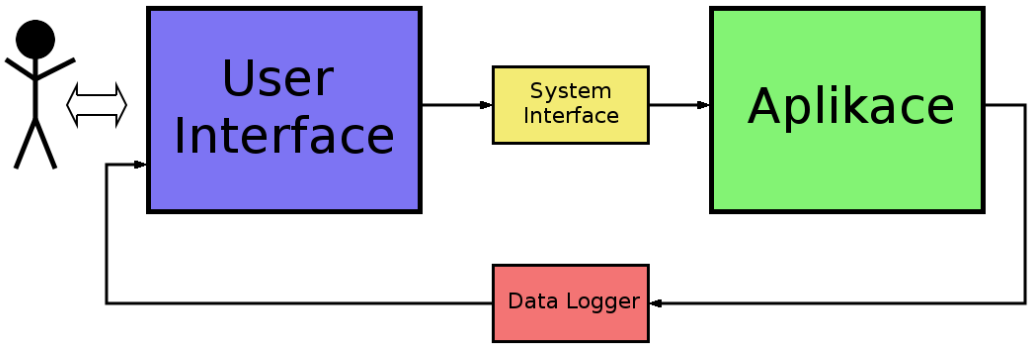
Funkce *set_default()* slouží k nastavení systému do továrního nastavení. Tato funkce zajistí, že se uživatel, po jakémkoli rozladění systému, může vrátit vždy do výchozího nastavení.

Funkce *enable / disable* output slouží k aktivaci a deaktivaci PWM výstupu, respektive k aktivaci a deaktivaci akčních členů. Pokud byla vykonána funkce *disable_output()*, pak budou v každém režimu akční členy neaktivní. Pro jejich aktivaci je nutno vykonat funkci *enable_output()*.

3.7 Rozhraní mezi webovou a řídicí aplikací

Webová stránka slouží jako grafické rozhraní, pomocí kterého uživatel volí parametry, kterými definuje, jak se má ovládaný systém chovat. Na základě zvolených parametrů vykonává řídicí aplikace konkrétní výpočty a případnou konfiguraci hardwarových prvků. Jelikož obě aplikace pracují jako samostatné systémy, bylo potřeba definovat rozhraní, pomocí kterého spolu budou komunikovat.

Jako nejjednodušší a nejefektivnější řešení bylo navrženo rozhraní pomocí konfiguračních (komunikačních) souborů. Pro přehlednost byly zvoleny dva soubory. Jeden slouží pro předávání dat z webové do řídicí aplikace a druhý pro opačný směr. Jelikož je přenos dat přes daný soubor vždy jednosměrný, data mohou být přenášena v obou směrech současně, tedy plný duplex.



Obr. 12 - Přenos dat mezi aplikacemi

Data distribuovaná z webové aplikace obsahují pouze konfigurační parametry pro řídicí aplikaci. Tato data jsou zapisována do konfiguračního souboru s názvem */home/debian/interface*. Z tohoto souboru je čte řídicí aplikace při parametrizaci. Do konfiguračního souboru se ukládají tyto parametry: pracovní režim (mode), konstanty PID regulátoru (P,I,D), žádaná teplota (W) a velikost akční veličiny (U). V souboru interface jsou data zapsána ve formátu řetězce s oddělovačem *';'* a ukončovacím znakem *'~'*. Tento soubor obsahuje pouze jeden řádek s aktuálním nastavením.

Obr. 13 - Rozhraní pro konfiguraci řídicí aplikace

V řídicí aplikaci probíhá měření teploty, výpočet akční veličiny, měření času, atd. Všechna tato data je nutné poskytovat webové stránce pro jejich vizualizaci. Stejně jako průběhy akčních veličin je potřeba poskytovat zpětnou informaci o aktuálně nastavených parametrech.

Tato data distribuuje řídicí aplikace v každém cyklu. Výstupní data jsou ukládána do souboru */home/debian/data_logger.csv*. Oproti souboru *interface* se zde aktuální data připisují vždy na konec souboru. Jako oddělovač atributů slouží znak ‘;’. Pro ukončení rámce slouží znak ‘\n’. Poslední řádek souboru tak vždy obsahuje aktuální data. V jazyce BASH lze aktuální data získat například pomocí příkazu: *tail -1 /home/debian/data_logger.csv*.

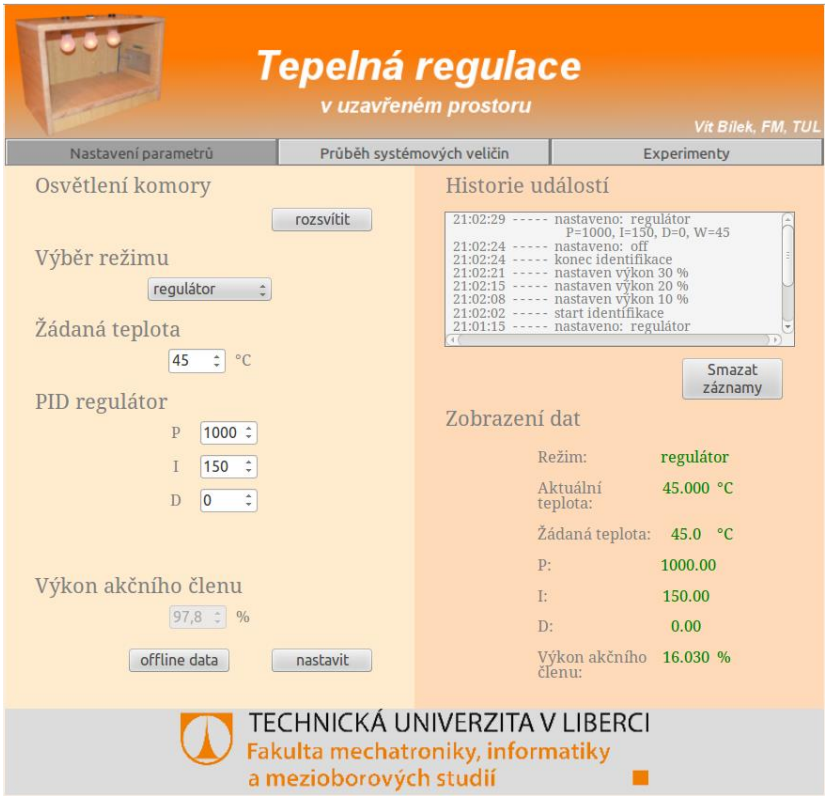
Obr. 14 - Rozhraní pro přenos aktuálních dat mezi řídicí aplikací a webovou stránkou

3.8 Webová stránka, webová aplikace

Úkolem této stránky je vytvořit pro uživatele intuitivní grafické rozhraní, pomocí kterého bude moci uživatel nastavit parametry systému podle své představy a současně bude moci sledovat vývoj teploty v regulované soustavě a průběh regulačního pochodu. Stránka současně poskytuje možnost automatického měření podle časového harmonogramu sestaveného uživatelem. Skládá se ze tří záložek, na kterých jsou rozmístěny komponenty sloužící k vykonávání zmíněných funkcí.

3.8.1 Nastavení parametrů

Tato záložka se skládá ze dvou sekcí. První sekce má funkci nastavovacího panelu, pomocí kterého lze nastavit parametry, podle kterých má systém pracovat. Druhá sekce slouží k zobrazení aktuálně nastavených a naměřených hodnot.



Obr. 15 - Webová stránka, nastavování konfiguračních parametrů

Pomocí tlačítka pod nápisem *Osvětlení komory* a s textem *„rozsvítit“*, lze rozsvítit nebo zhasnout doplňkové osvětlení v termokomoře. Po stisknutí se změní jeho text na *„zhasnout“*. Podle aktuálního textu tlačítka se po stisku vyvolá příslušná

akce. Osvětlení termokomory je vyrobeno z LED diod, takže nijak neovlivňuje teplotu v komoře.

Pod nápisem *„Výběr režimu“* je umístěn prvek *select*, pomocí kterého lze zvolit pracovní režim (funkci), ve kterém má systém pracovat. Nabízenými režimy jsou *OFF*, *regulátor*, *identifikace*. Nabízenými funkcemi pak *output enable*, *output disable* a *set to default*. Činnost režimů a funkcí je podrobně popsána v kapitole *3.6 Řídicí aplikace*.

Na základě volby režimu lze aktivovat příslušné prvky pro volbu žádané teploty, konstant PID regulátoru a výkonu akčních členů. Pro nastavení zvoleného režimu a příslušných parametrů slouží tlačítko *nastavit*. Po jeho stisku se nastavené hodnoty zapíší do konfiguračního souboru */home/debian/interface*, čímž se zadá požadavek pro řídicí aplikaci.

Pro zápis do souborů se využívá funkce *write_to_file(fcn,data)*. Vstupním parametrem *fcn* je funkce, která má být vykonána. V parametru *data* jsou data potřebná pro vykonání zvolené funkce. Tato funkce využívá metodu *ajax*, pomocí které je volán skript *set_values.php*, kterému jsou předány vstupní parametry *fcn* a *data*, pomocí kterých vykoná požadovanou akci a pošle hlášení o úspěchu či neúspěchu. Takovou funkcí může být například zápis konfiguračních dat reprezentovaných vstupní proměnnou *data* do souboru *interface*.

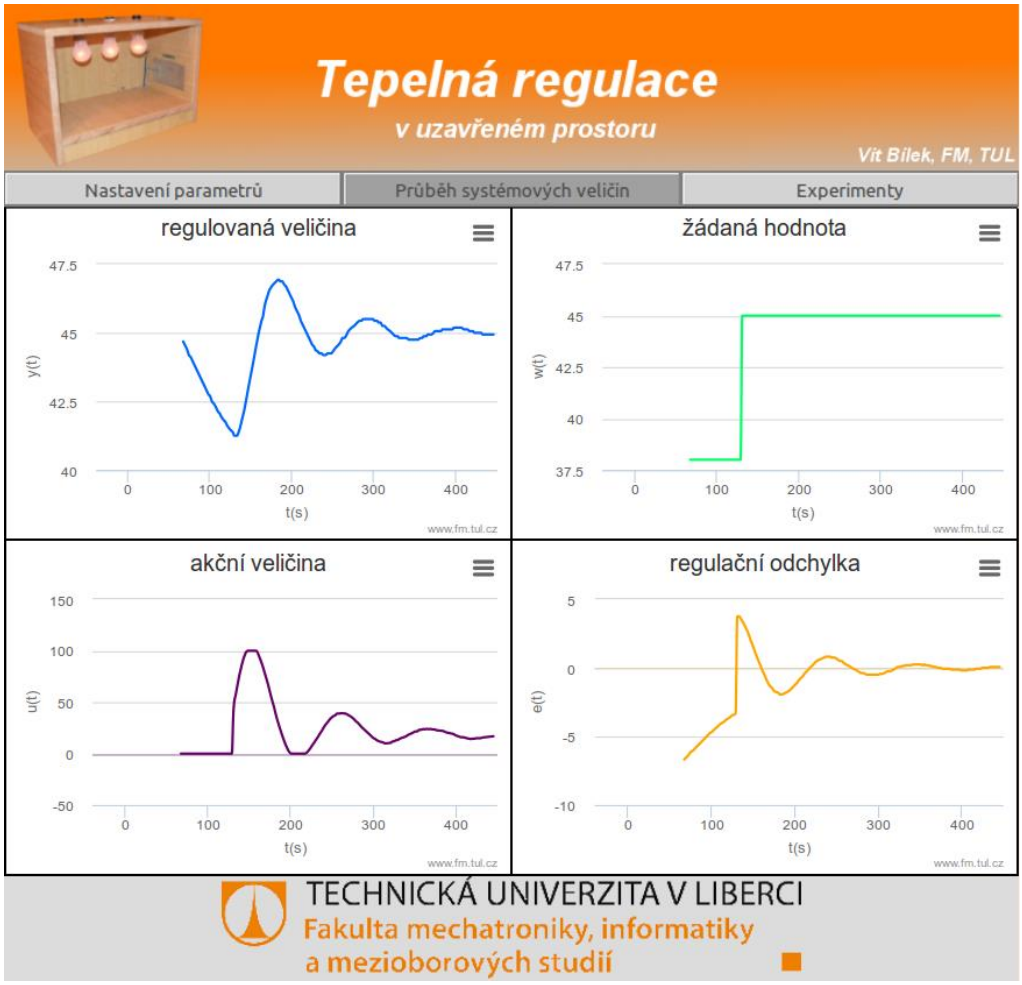
Každá událost, která byla provedena se zapíše do logovacího žurnálu *„Historie událostí“*. Každý záznam začíná časovou značkou, která jednoznačně určuje čas provedení události. Pro nenávratné smazání záznamů slouží tlačítko *„Smazat záznamy“*.

Poslední položkou záložky je zobrazovací panel „*Zobrazení dat*“, na kterém jsou zobrazeny aktuální hodnoty. Jejich aktualizace probíhá každé dvě sekundy. Tuto aktualizaci zajišťuje časovač, který po vypršení timeoutu načte aktuální data ze souboru */home/debian/data_logger.csv*, zpracuje je a zobrazí v tomto panelu. Tato data se používají i pro interaktivní grafy viz. *3.8.2 Interaktivní grafy*. Pro čtení dat ze souboru se používá metoda *read_from_file(fcd, data)*, která funguje podobně jako funkce *write_to_file()*, ale vrací vyžádaná data. Například funkce *read_from_file(„get_data_lines“,5)* vrací posledních pět záznamů ze souboru *data_logger.csv*.

3.8.2 Interaktivní grafy

Druhá záložka s názvem *„Průběh systémových veličin“* slouží uživateli ke grafickému zobrazení průběhů veličin $y(t)$, $w(t)$, $e(t)$ a $u(t)$. Obsahuje čtyři interaktivní grafy.

Po načtení stránky je volána funkce *init_charts()*, která vytvoří pole grafů. Pro každý graf je definováno jméno a nadpis grafu, barva čáry a jednotka zobrazovaných dat například °C.



Obr. 16 - Webová stránka, zobrazení systémových veličin

Pro zobrazení průběhu systémových veličin je použita demoverze knihovny *highcharts*, pomocí níž lze vytvářet grafy v jazyce *JavaScript*. Demoverze omezuje počet zobrazených bodů v jednom grafu na 1000 bodů pro jednu proměnnou.

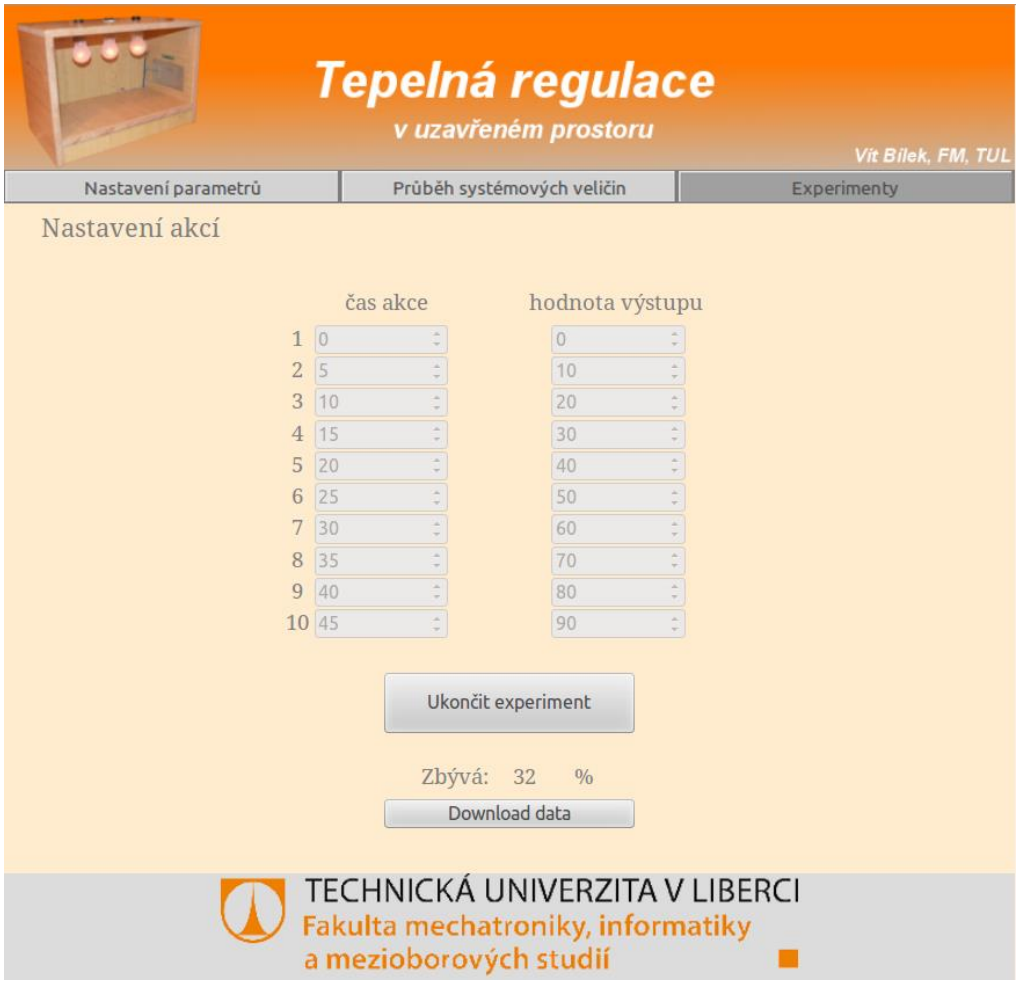
Pro vytvoření grafu je potřeba provést jeho parametrizaci, která spočívá v nadefinování parametrů, jako je typ grafu, název, nadpis, popis os, vlastnosti os, typy

čáry, atd. Ukázka parametrizace grafů je v příloze *E - Ukázka implementace grafu highcharts*. K vytvoření grafů slouží funkce *vykresli_graf(graf)*. Vstupním parametrem této funkce je jeden prvek z pole grafů.

Při parametrizaci grafu je v parametru *chart - event* inicializován časovač, který s periodou 2 s přidává do pole bodů nový bod odpovídající aktuální hodnotě příslušné veličiny. V grafu je zobrazován stále stejný počet bodů. Přidáním nového bodu dojde k vymazání historicky nejstaršího a přidání nového bodu, což způsobí posunutí časové osy x o jeden krok, tedy o 2 s. Graf tak funguje jako grafický posuvný registr.

3.8.3 Identifikační měření

Poslední záložka webové stránky slouží pro automatické měření teploty na tepelném systému.



Obr. 17 - Webová stránka, identifikační měření

Záložka obsahuje tabulku o dvou sloupcích, která slouží jako časový

harmonogram. První sloupec tabulky nese informaci o času, kdy má být provedena změna akční veličiny. Hodnota druhého sloupce udává hodnotu výkonu akčního členu, která má být v daný čas nastavena. Čas je uváděn v sekundách od započetí experimentu, velikost akčního zásahu je v procentech.

Stisknutím tlačítka *„Spustit experiment“* se spustí proces a text tlačítka se změní na *„Ukončit experiment“*. Provede se deaktivace tabulky, aby nemohlo dojít ke změně hodnot během experimentu a k deaktivaci záložky „Nastavení parametrů“, aby nemohlo během experimentu dojít ke změně režimu.

Před spuštěním samotného experimentu se provede kontrola prvního sloupce tabulky. Při kontrole se postupně prochází řádky. Pokud je hodnota aktuálního řádku větší než hodnota následujícího řádku, jsou všechny následující další řádky, včetně aktuálního, nastaveny na nulovou hodnotu a čas tohoto řádku je považován za čas poslední změny akční veličiny.

Po provedení kontroly se spustí časovač, který kontroluje první sloupec tabulky. Pokud nastal čas určité akce, dojde k nastavení výkonu akčního členu na příslušnou hodnotu dle tabulky. Současně se také provádí výpočet pro zjištění doby průběhu experimentu. Tato hodnota se zobrazuje pod tabulkou časového harmonogramu v procentech. Po dokončení experimentu se časovač vypne.

Stiskem tlačítka „Download data“ může uživatel získat naměřená data, jak v průběhu měření, tak i po jeho ukončení. Poskytnutá data jsou v souboru ident_data.csv.

Po stisku tlačítka *„Ukončit experiment“* se deaktivované prvky opět aktivují pro možnost dalšího nastavení. Pracovní režim se nastaví do režimu *OFF*. Při dalším zahájení experimentu dojde k přepsání souboru naměřených dat.

4. Praktická část 2 – Identifikace systému, simulace

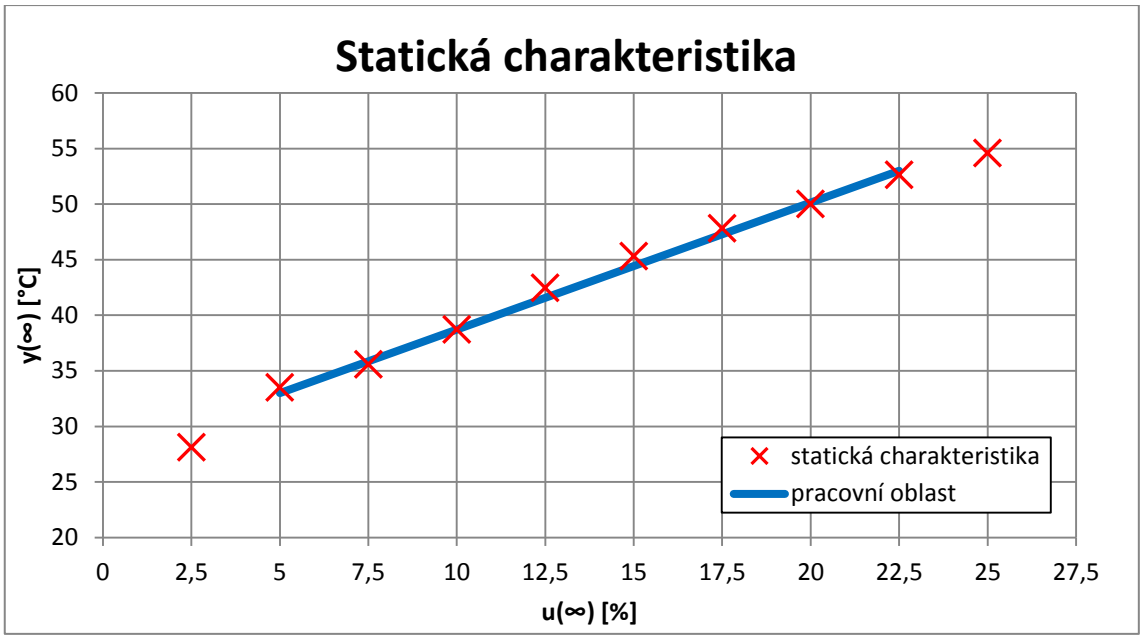
Tato část se zabývá určením matematického modelu realizovaného systému a simulací jeho vlastností. Pro určení modelu dynamického systému je nutné provést tzv. identifikaci, která se skládá z měření různých charakteristik na identifikovaném systému. Pomocí charakteristik lze získat přenosovou funkci v *Laplaceově* obrazu. Díky získané přenosové funkci pak lze simulovat chování systému.

Informace pro následující podkapitoly byly čerpány ze zdrojů [9], [13] a [14].

4.1 Statická charakteristika

Statická charakteristika charakterizuje chování dynamického systému v ustálených stavech. Pomocí ní lze určit lineární či nelineární charakter zesílení dynamického systému $y(\infty) = f(u(\infty))$.

Pro získání průběhu statické charakteristiky bylo naměřeno několik bodů v rozmezí 27 – 55 °C. Každý bod vyjadřuje ustálenou hodnotu odezvy na budicí funkci typu jednotkový skok o různé velikosti akční veličiny. Velikost akční veličiny byla volena od 2,5 do 25 % maximálního výkonu akčních členů s krokem 2,5 %.



Graf 2 - Statická charakteristika

Při měření statické charakteristiky byl systém umístěn v místnosti s konstantní teplotou 18 °C. Doba pro dosažení ustáleného stavu akční veličiny se pohybovala mezi 10 000 a 15 000 s. V průběhu prvních 350 s, po změně akční veličiny, se projevovala jistá dynamika systému. Po jejím odeznění však nedošlo k ustálení a teplota se dále zvyšovala.

Z naměřené statické charakteristiky je patrné, že se systém v rozmezí 5 - 22,5 % maximálního výkonu akční veličiny chová téměř lineárně. V tomto rozmezí akční veličiny byla tedy stanovena pracovní oblast, pro kterou bylo provedeno identifikační měření.

Použité akční členy (žárovky) distribuují teplo do okolí sáláním a prouděním, což zřejmě vytváří jistý přechodový jev v prvních 350 s. Díky tomu, že dochází k ohřevu objímek, dochází tak i k přenosu tepla vedením do víka, ve kterém jsou objímky uchyceny a víko se tak samo stává zdrojem proudění tepla.

Na systém má vliv prostup tepla z okolí a do okolí. Termokomora má jistou akumulační vlastnost, která je zřejmě způsobena použitým materiálem.

4.2 Identifikace systému

Po určení statické charakteristiky byla na základě chování systému stanovena pracovní oblast, která byla linearizována. Pro určení dynamiky systému bylo provedeno identifikační měření, které se skládalo z několika skoků akční veličiny v pracovní oblasti. Měření probíhalo v místnosti s konstantní teplotou 18 - 18,5 °C.

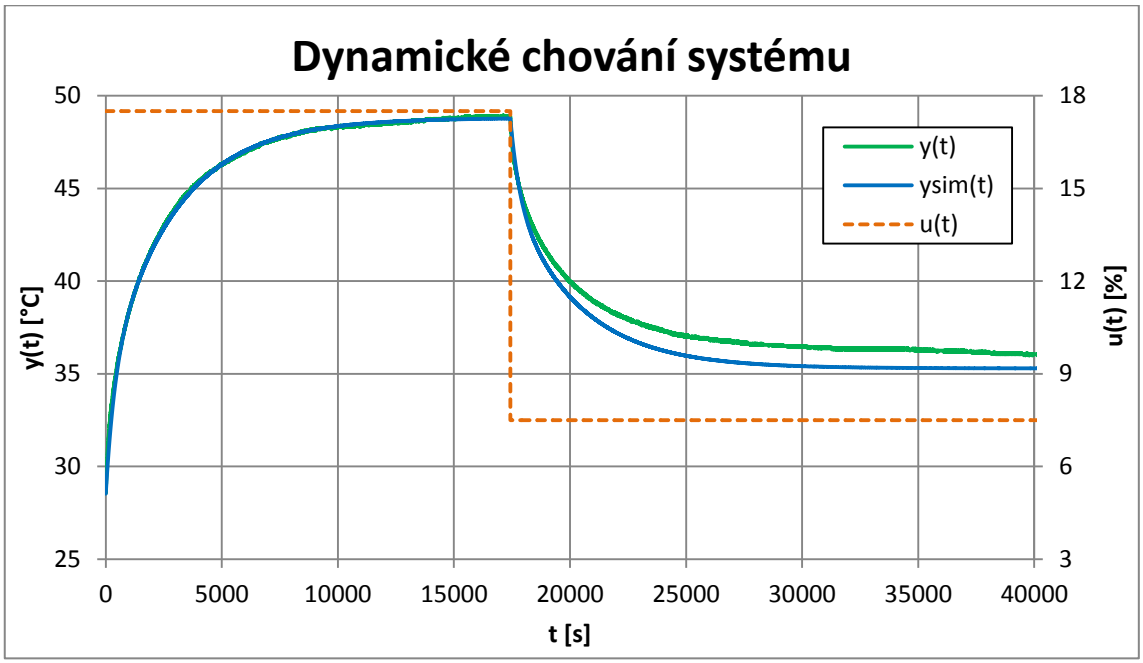
Při identifikačním měření dynamických vlastností byl nejprve proveden skok akční veličiny na hodnotu 2,5 % maximálního výkonu. Po ustálení regulované veličiny byl proveden skok na hodnotu u = 17,5 %. Po ustálení regulované veličiny byl proveden skok na u = 7,5 %. Pomocí těchto tří skoků se projevilo dynamické chování systému. Průběhy $u(t)$ a $y(t)$ jsou zobrazeny v *grafu 3*.

Před samotnou identifikací musela být naměřená data upravena do určitého formátu. Pro identifikaci byla použita odezva na skok z u = 2,5 na 17,5 % a z u = 17,5 na 7,5 %. Ze zaznamenaných dat byla vybrána data od času, kdy byl proveden skok na 17,5 %. Od průběhu akční veličiny byla odečtena konstanta 2,5 % a od průběhu

měřené veličiny byl odečten průměr ustálené hodnoty při odezvě na u = 2,5 %. Vektor času byl upraven tak, aby začínal od času nula.

Pro identifikaci dat bylo použito prostředí *Matlab*. Samotná identifikace systému pak byla provedena pomocí toolboxu *Ident*. Byla provedena identifikace pro celý soubor dat, tedy pro skok „nahoru“ i „dolů“. Při simulaci se ukázalo, že se systém chová jinak při ohřevu a jinak při chladnutí, což je způsobeno charakterem systému. Při návrhu regulátoru neodpovídá identifikovaný model dostatečně přesně reálnému chování soustavy. Pro lepší shodu reálného chování s chováním modelu byla provedena identifikace pouze pro skok „nahoru“, tedy pro skok z u = 2,5 na 17,5 %, na jejím základě byla získána tato přenosová funkce:

$$G(s) = \frac{0{,}001394\ (s\ + 0{,}0008752)}{(s + \ 0{,}006261)\ (s + \ 0{,}0003461)} \tag{28}$$



Graf 3 - Porovnání modelu s reálným chováním systému

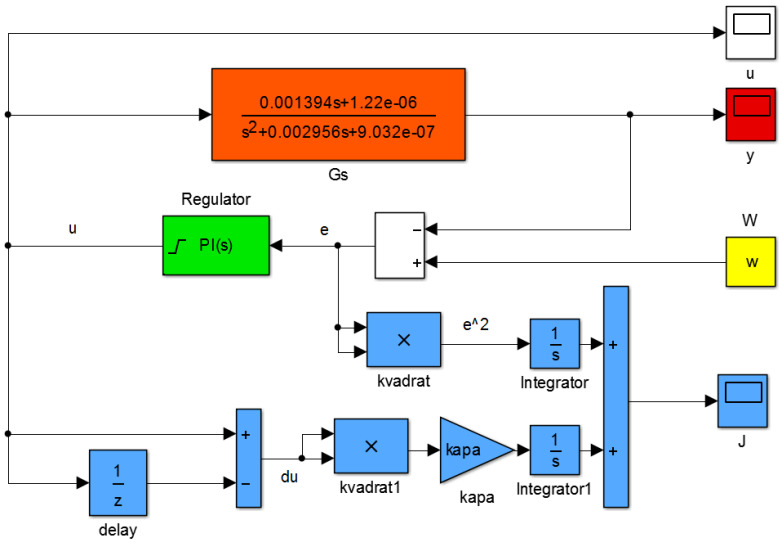
Z *grafu 3* je patrné, že pro skok „nahoru“ se model s reálným systémem téměř shodují, pro skok dolů se liší. Pokud při regulaci nastane situace, že bude žádaná hodnota nižší než regulovaná, bude akční veličina na spodní saturační mezi. Na rychlost ochlazování prostoru komory bude mít vliv pouze prostup tepla do okolí a měrná tepelná kapacita komory. Jelikož je v regulátoru implementováno omezení proti *wind-up efektu*, regulátor začne mít vliv na regulační pochod až při dostatečně malé absolutní

hodnotě regulační odchylky. V praxi to znamená, že pro návrh regulátoru, má tento rozdíl systému a modelu zanedbatelný vliv. Pro návrh regulátoru tedy lze tento model použít.

4.3 Návrh konstant PID regulátoru pro regulaci systému

Pro návrh konstant PID regulátoru bylo použito prostředí *Matlab*. Při použití metod, které neuvažují omezení akční veličiny, bylo dosaženo nejlepších výsledků pomocí experimentálně-iterační metody, která spočívala v ručním nastavení složek a jejich postupném dolaďování. Touto metodou se jako optimální jevil PI regulátor s konstantami P = 45,2 a I = 1,56.

Pro nalezení optimálních konstant regulátoru byla zvolena metoda, která uvažuje omezení akční veličiny a omezení integrační složky proti *wind-up* efektu. Tato metoda využívá prostředí *Matlab Simulink*, ve kterém bylo sestrojeno simulační schéma zobrazené na *obrázku 18*.



Obr. 18 - Simulační schéma pro nalezení kvadratického kritéria

Simulační schéma se skládá z regulačního obvodu tvořeného modelem systému a PID regulátorem. Výstup regulátoru má omezený rozsah akční veličiny na 0 až 100, což odpovídá 0 – 100 % výkonu akčních členů. Dále má aktivováno omezení proti

wind-up efektu. Cílem vnesení všech omezení do regulačního obvodu je co největší přiblížení simulace reálnému procesu.

Z regulačního obvodu jsou vyvedeny signály $e(t)$ a $u(t)$. Z regulační odchylky je v každém kroku vypočten kvadrát, jehož hodnota je integrována. Akční veličina je nejprve derivována, respektive je vypočten rozdíl mezi aktuálním a minulým vzorkem. Z této hodnoty je poté určen kvadrát, který je následně integrován. Součtem těchto dvou signálů je získáno kvadratické kritérium *J*.

$$J = \int_0^{Tsim} \bar{e}(t)^2 + \kappa \int_0^{Tsim} \Delta u(t)^2 \qquad (29)$$

Výsledkem této rovnice je tzv. kvadratická regulační plocha, pro její určení se využívá integrál kvadrátu regulační odchylky a integrál kvadrátu derivace akční veličiny, kde $\bar{e}(t) = e(t) - e(\infty)$ (30) a $\Delta u = \left(\frac{du(t)}{dt}\right)^2$ (31). $e(t)$ je regulační odchylka, $e(\infty)$ je ustálená hodnota regulační odchylky, která je v ideálním případě rovna nule. κ je váhový koeficient, jehož volbou se dosáhne potlačení akční veličiny. Čím větší hodnoty dosahuje, tím více je akční veličina tlumena.

Pro vyjádření kritéria *J* byla vytvořena funkce *kriterium.m*, která má tři vstupní parametry odpovídající konstantám P, I, D. Tato funkce na základě vstupních parametrů provede simulaci. Její návratová hodnota je rovna hodnotě kritéria *J*.

```
function [ output ] = kriterium(x)

    global p i d j

    p = abs(x(1));
    i = abs(x(2));
    d = abs(x(2));

    sim('simm');

    output = j(2);
end
```

Ukázka kódu 6 - Funkce pro určení kvadratického kritéria

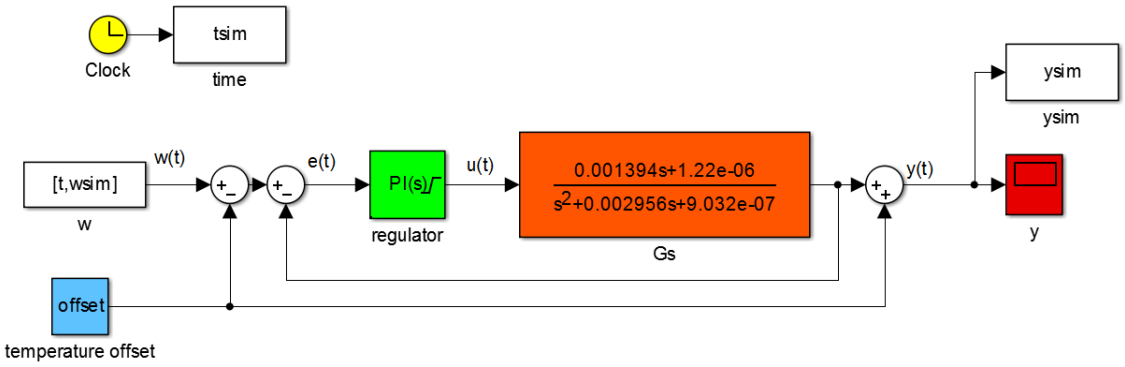
K nalezení optimálních konstant regulátoru byl vytvořen skript *find_pid.m*, jehož úkolem je nalézt takové konstanty regulátoru, pro které dosahuje *J* nejmenší hodnoty. K nalezení optimálních konstant byla použita funkce *fminsearch*. Pomocí této funkce je iteračně volána funkce *kriterium.m*, s tím, že v každém iteračním kroku

jsou změněny konstanty regulátoru, na základě kterých je provedena simulace a určena hodnota kritéria J. Po nalezení nejmenší hodnoty, které nabývá J, vrátí funkce *fminsearch* příslušné hodnoty konstant regulátoru.

Pomocí této metody byl nalezen optimální regulátor typu PI s konstantami P = 53,525 a I = 2,088, pro κ = 0,1.

4.4 Simulace regulačního pochodu a porovnání s reálnem

Po získání optimálních konstant PID regulátoru byla provedena simulace regulačního pochodu v prostředí *Matlab Simulink*. Simulační schéma je naznačeno na *obrázku 19*. Pro regulátor byly uvažovány konstanty P = 53,525, I = 2,088 a D = 0. Výstup regulátoru omezuje akční veličinu na rozsah hodnot 0 - 100, což odpovídá 0 až 100 % výkonu akčních členů. Dále omezuje integrační složku proti vzniku *wind-up* efektu.

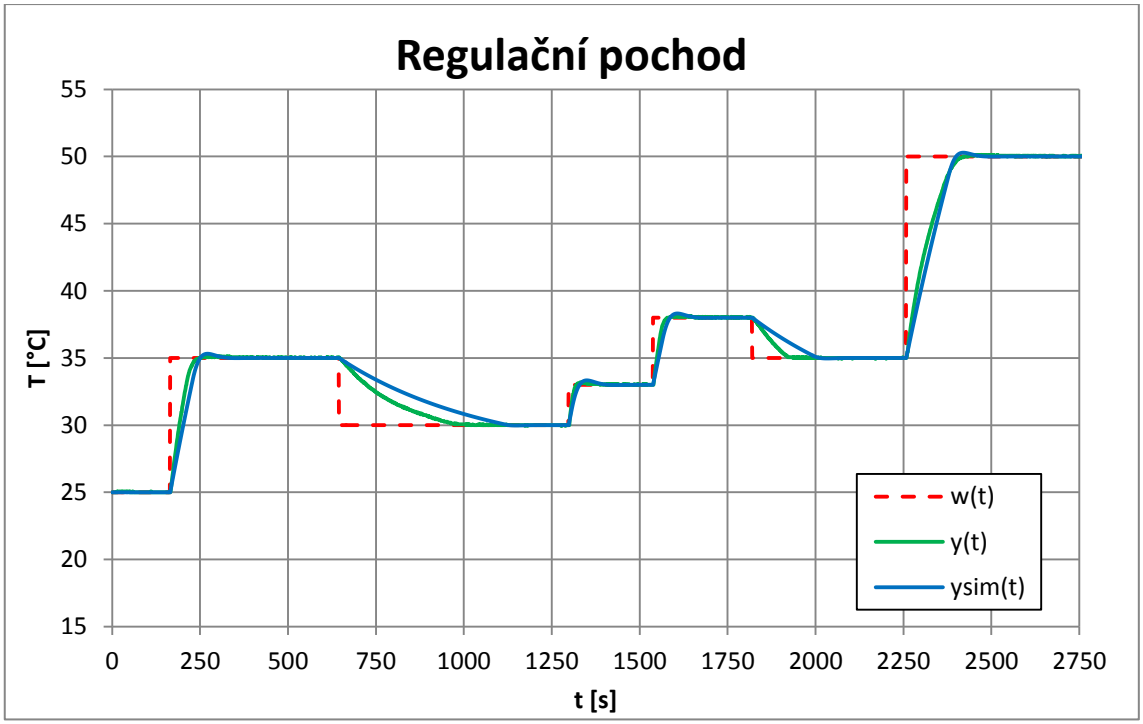


Obr. 19 - Simulační schéma regulovaného systému

Pro porovnání regulačního pochodu simulovaného a reálného systému byl pro oba systémy použit stejný průběh žádané veličiny $w(t)$. U každého systému jsou však jiné počáteční podmínky. Zatímco v simulaci je $y(0) = 0$, u reálného systému odpovídá $y(0)$ ustálené okolní teplotě místnosti, ve které je reálný model umístěn. Pokud je tedy u reálného systému žádaná hodnota 35 °C a počáteční teplota $y(0)$ 18 °C, skok akční veličiny pak není 35, ale pouze 17 °C. Pro zajištění stejných podmínek v simulaci byla od vektoru žádaných hodnot odečtena konstanta odpovídající ustálené teplotě okolí. O tuto hodnotu byla následně zvýšena regulovaná veličina $y(t)$. Tato konstanta je reprezentována blokem *temperature offset*.

K nastavení žádané veličiny byl použit blok *from workspace*, jehož vstupem je vektor času a vektor průběhu žádané veličiny. Pro zaznamenání průběhu regulované veličiny $ysim(t)$ a vektoru času simulace $tsim$ byly použity bloky *to workspace*.

Pro ověření kvality regulace bylo provedeno několik skoků akční veličiny, která byla volena tak, aby se regulovaná veličina pohybovala po stanovené pracovní oblasti. Průběh žádané veličiny a průběhy regulované veličiny reálného systému a simulace jsou zobrazeny v *grafu 4*.



Graf 4 - Regulační pochod simulované a reálné soustavy

Z průběhů regulovaných veličin je patrné, že při skoku žádané veličiny na vyšší hodnotu, než jaká je aktuální teplota systému, jsou průběhy regulované veličiny modelu a reálné soustavy velice podobné. Při změně žádané hodnoty na nižší teplotu, než je aktuální teplota systému, se průběhy regulovaných veličin liší. Tento problém byl diskutován už v kapitole *4.2 Identifikace systému*. Při změně žádané hodnoty na nižší teplotu dochází k chladnutí termokomory. Jelikož použité akční členy umí teplo pouze dodávat, při chladnutí soustavy se tak téměř neprojeví. Hlavní vliv na chladnutí má rozdíl vnitřní a vnější teploty. Tento jev je dán *součinitelem tepelné vodivosti* a *měrnou tepelnou kapacitou* termokomory, respektive materiálů, ze kterých je vyrobena.

Závěr

Během této diplomové práce byla navržena a zhotovena termokomora, která slouží jako reálný model tepelné soustavy. Byla doplněna o akční a měřicí členy.

K regulaci teploty v termokomoře byl vybrán embedded systém BeagleBone, který byl rozšířen o další elektronické prvky. K samotnému měření teploty, regulaci a zaznamenávání měřených dat byla vyvinuta *řídicí aplikace*. Pro uživatelskou obsluhu systému, tj. pro nastavování parametrů regulátoru, výběru pracovního režimu, změny žádané hodnoty atd., byla vytvořena *webová aplikace*, kterou lze ovládat přes internetový prohlížeč. Součástí této aplikace jsou interaktivní grafy zobrazující online průběhy veličiny $u(t)$, $w(t)$, $y(t)$ a $e(t)$. Ve *webové aplikaci* je implementován panel pro manuální a automatické experimentální měření na systému. Výstupem je soubor naměřených dat ve formátu *csv*.

Po zkonstruování a validaci systému na něm byla provedena série měření, na základě kterých vznikla statická charakteristika a matematický model reprezentovaný přenosovou funkcí. Pro zpracování dat a nalezení modelu bylo použito prostředí Matlab.

Na základě modelu systému byly, pomocí kvadratického kritéria, nalezeny optimální konstanty PI regulátoru. Pro tento regulátor byl simulován průběh regulačního pochodu, který byl následně porovnán s regulačním pochodem reálného systému. Při porovnání se oba systémy chovaly velice podobně. Jediný problém regulace se projevil při chladnutí systému, kdy regulátor není schopen ovlivnit dynamiku tohoto průběhu. Ta záleží pouze na rozdílu teploty systému a okolí.

Součástí diplomové práce je tento dokument, který vznikal současně se zařízením, které je touto prací prezentováno. Součástí dokumentu jsou přílohy s ukázkami použitých kódů a skriptů. Kompletní přílohy včetně elektronické podoby tohoto dokumentu jsou umístěny na přiloženém CD.

Možné uplatnění tohoto zařízení by mohlo být například pro líhně, kvasné procesy, sauny a podobně.

Během vytváření této práce mi byly velkým přínosem znalosti získané během studia na FM TUL. V práci jsem uplatnil praktické zkušenosti získané zejména z předmětů ZSR, ARI, CRI, RPS atd.

Použitá literatura

[1] BALÁTĚ, Jaroslav. *Automatické řízení*. Praha: BEN – technická literatura, 2003. ISBN 80-730-0020-2.

[2] BOŘÁNEK, Roman. *Srovnání - Raspberry Pi a jeho konkurenti*. In: ROOT.cz [online]. Praha 2015 [cit. 2016-05-09]. Dostupné z: http://www.root.cz/clanky/srovnani-raspberry-pi-a-jeho-nejvetsi-konkurenti/

[3] *DS18B20, technická specifikace* [online]. [cit. 2016-05-12]. Dostupné z: http://pdf1.alldatasheet.com/datasheetpdf/view/58557/DALLAS/DS18B20.html.

[4] GROSMAN, Josef. *Řídicí počítačové systémy* [online]. Liberec, 2015 [cit. 2016-03-03]. Dostupné z: https://elearning.tul.cz/course/view. php?id-=2223. Kurz. Technická univerzita v Liberci

[5] HEROUT, Pavel. *Učebnice jazyka C 1. Díl*. 6. vyd. České Budějovice: Kopp, 2010. ISBN 978-80-7232-383-8.

[6] HEROUT, Pavel. *Učebnice jazyka C 2. Díl*. 6. vyd. České Budějovice: Kopp, 2010. ISBN 978-80-7232-367-8.

[7] HLAVA, Jaroslav. *Číslicové řízení* [online]. Liberce, 2015. [cit. 2016-01-20]. Dostupné z: https://elearning.fm.tul.cz/course/view.php?id=2185. Kurz. Technická univerzita v Liberci.

[8] *Highcharts documentation* [online]. [cit. 30. 3. 2016]. Dostupné z: http://www.highcharts.com/docs

[9] JANEČEK, Josef. *Základy spojitého řízení* [online]. Liberec, 2011 [cit. 2016-05-09]. Dostupné z: https://elearning.tul.cz/course/view.php?-id=212. Kurz. Technická univerzita v Liberci

[10] KUPKA, Libor a Josef JANEČEK. *Matlab: Řešené příklady*. Lanškroun: SOŠ a SOU Lanškroun, 2007. ISBN 978-80-239-9532-9

[11] MACH, Jakub. *PHP pro úplné začátečníky*. 2. vyd. Brno: CP Books, a.s., 2005. ISBN 80-7226-834-1.

[12] MAŤÁTKO, Jan. *Elektronika*. 6. vyd. Praha: IDEA SERVIS, 2005. ISBN 1866-067-05.

[13] MODRLÁK, Osvald. *Teorie automatizačního řízení I. - Syntéza regulačních obvodů*. Liberec, 2004.

[14] MODRLÁK, Osvald a Lukáš HUBKA. *Automatické řízení*, Liberec, 2011.

[15] PLÍVA, Zdeněk, Jindra DRÁBKOVÁ, Jan KOPRNICKÝ a Leoš PETRŽÍLKA. *Metodika zpracování bakalářských a diplomových prací*. 2. upr. vyd. Liberec: Technická univerzita v Liberci, 2014. ISBN 987-80-7494-049-1.

Seznam příloh

Příloha A - Obsah přiloženého CD

Přiložené CD obsahuje adresáře s názvy *Diplomová práce*, *Webová aplikace*, *Řídicí aplikace*, *Matlab* a *Data*. Obsahem adresáře *Diplomová práce* je soubor *Diplomova_prace_Vit_Bilek.pdf*. Tento soubor obsahuje tuto diplomovou práci v elektronické podobě. Adresář *Webová aplikace* obsahuje kompletní projekt aktuální verze Webové aplikace implementované v minipočítači. V adresáři *Řídicí aplikace* je uložen zdrojový kód této aplikace. Adresář *Matlab* obsahuje veškeré skripty typu *.m* a simulační schémata použitá v této diplomové práci. Adresář *Data* obsahuje naměřená data charakteristiky použité v této práci.

Příloha B - Skript pro inicializaci sběrnice One-Wire

```
/dts-v1/;
/plugin/;

/ {
        compatible = "ti,beaglebone", "ti,beaglebone-black";

        part-number = "BB-W1";
        version = "00A0";

        /* state the resources this cape uses */
        exclusive-use =
                /* the pin header uses */
                "P9.11",
                /* the hardware IP uses */
                "gpio0_30";

        fragment@0 {
        target = <&am33xx_pinmux>;
        __overlay__ {
                                        dallas_w1_pins: pinmux_dallas_w1_pins {
                                                pinctrl-single,pins = <0x70 0x37>;
                        };
        };
        };

        fragment@1 {
        target = <&ocp>;
        __overlay__ {
                        onewire@0 {
                                compatible      = "w1-gpio";
                                pinctrl-names   = "default";
                                pinctrl-0       = <&dallas_w1_pins>;
                                status          = "okay";

                                gpios = <&gpio1 30 0>;
                        };
        };
        };
};
```

Příloha C - Ukázka kódu řídicí aplikace, jazyk C

```
/* set output intensity*/
void pwm_set_output(int set_val)		// set pwm_output set_val must be in range 0 to PWM_T
{
	if(set_val < 0)				// set pwm space
		set_val = 0;			// 0  -> no power in output
	if(set_val > PWM_RANGE)
		set_val   = PWM_T*100;		// 10000 -> maximum power in output

	set_val = PWM_T*1000 - set_val*10;
	pwm_duty_ns(set_val);
}

/* identification  0x02 */
void routine_identification(int time, float y_term, int intenzity)
{

	if(intenzity < PWM_MIN)			// check minimum range
		intenzity = PWM_MIN;

	if(intenzity > PWM_MAX)			// chceck maximum range
		intenzity = PWM_MAX;

	pwm_set_output(intenzity);		// set output
	u = intenzity;

	if(mode_changed_detect() > 0)		// if start identificatiom -> init ident routine
	{
		init_ident();			// clear data in init_data file
		ident_time_offset = time;	// ident time is 0
	}

	ident_logging(time - ident_time_offset,y_term);		// write to identification file

	printf("identification intezity %i, time offset %i\n\n", intenzity, ident_time_offset);  // -> std out
}

/* get output intenzity */
int pwm_get_value(void)
{
	int duty,period;

	duty = get_value_from_file(PWM_DUTY);		// get pwm duty
	period = get_value_from_file(PWM_PERIOD);	// get pwm period

	return (period - duty)/100;			// count output power in %x100
}

/* enable pwm output */
void pwm_enable()
{
	set_value_to_file(PWM_ENABLE, 1);		// switch on pwm
}
```

Příloha D - Ukázka kódu webové stránky, jazyk HTML

```
<!--
    prava sekce
  -->

    <div id="sekce_2">

    <!-- Historie logu -->



     <div class="nadpis1">Historie událostí</div>
    <div id="logs">.. Žádné záznamy ..</div>
    <button class="but_set" id="smaz_udalosti">Smazat záznamy</button>
    <div class="odsazeni"></div>

      <!--
        Vizualizace dat
      -->

      <div class="visual_area" >

        <div class="odsazeni"></div>
        <div class="nadpis1">Zobrazení dat</div>

        <div id="display_area">
          <div class="nadpis2">Režim:</div>
          <div class="set_val" id="set_mode"></div><br>
          <div class="nadpis2">Aktuální teplota:</div>
          <div class="set_val" id="set_y"></div>
          <div class="value" >°C</div><br>
          <div class="nadpis2">Žádaná teplota:</div>
          <div class="set_val" id="set_w"></div>
          <div class="value">°C</div><br>
          <div class="nadpis2">P:</div>
          <div class="set_val" id="set_p"></div><br>
          <div class="nadpis2">I:</div>
          <div class="set_val" id="set_i"></div><br>
          <div class="nadpis2">D:</div>
          <div class="set_val" id="set_d"></div><br>
          <div class="nadpis2">Výkon akčního členu:</div>
          <div class="set_val" id="set_u"></div>
          <div class="value">%</div><br>
        </div>
      </div>
    </div>
```

Příloha E - Ukázka aplikace webové stránky, jazyk Java Script

```
$('#download_idt').click(function ()
 {
    logs_add("Pokus o stahování");
 });

 //zalozky
 $('#but_params').click(function ()
{
    go_to_params();
 });
 $('#but_charts').click(function ()
 {
    go_to_charts();
 });
 $('#but_exp').click(function ()
 {
    go_to_experiment();
 });

 // log button
 $('#smaz_udalosti').click(function (){
    logs = "";
    logs_add("clear window");
 });

 // download offline data
 $("#download_offline_data").click(function ()
 {
    get_actual_vals('#graf1');
 });


 function disable_all()
 {
    $('#in_P').prop('disabled', true);
    $('#in_I').prop('disabled', true);
    $('#in_D').prop('disabled', true);
    $('#in_teplota').prop('disabled', true);
    $('#in_power').prop('disabled',true);
 }
```

Příloha F - Ukázka implementace grafu highcharts

```
function vykresli_graf(graf)
{

    $(document).ready(function () {
     Highcharts.setOptions({
        global: {
           useUTC: false
        }
     });});


   $(graf.name).highcharts({
      chart:{
         type: 'spline',
         events: {
            load: function()
                {
                   var series = this.series[0];
                   var g_name = graf.name;

                   setInterval(function()
                           {
                              var data = get_actual_vals(g_name);
                              console.log(data[0].x.toString()+"   "+data[0].y);
                              series.addPoint([data[0].x*1,data[0].y*1],true,true);
                           }, t_sampling);
                }
         }
      },
      title: {
         text: graf.title,
         x: 0
      },
      xAxis: {
         tickPixelInterval: 70,
         title: {
            text: 't(s)'
         }
      },
      yAxis: {
         title: {
            text: graf.value
         },
         plotLines: [{
            value: 0,
            width: 1,
            color: graf.color
         }]
      },
      legend: {
         enabled: false
      },
      tooltip: {
         valueSuffix: 'value'
      },
      credits: {
         text: 'www.fm.tul.cz',
         href: 'http://www.fm.tul.cz'
      },
      series: [{
         name: graf.value,
```

```
            data: (function () {
                // generate an array of random data
                var data = [],
                    i;

                for (i = 1-velikost_grafu; i < 0; i += 1) {
                    data.push({
                        x: i,
                        y: null
                    });
                }
                return data;
            }()),
            color: graf.color
        }]
    });
}
```

Příloha G - Identifikační skript, Matlab

```
%% load data

        data = load('dynamika.csv');

        t = data(:,1);
        y = data(:,6);
        u = data(:,8);

        %figure
        %plot(t,u,t,y)

    %% identifikace 1

        ti = t(130000:end);
        yi = y(130000:end);
        ui = u(130000:end);

        u = ui(1945:end);
        ui = ui-ui(1940);
        ui = ui(1945:end);
        ti = ti(1945:end);

        ti = ti-ti(1);
        offset = (sum(yi(1:1940))/length(yi(1:1940)));
        yi = yi(1945:end);
        t = 0:1:(length(ti)-1);

        y = yi;
        yi = yi- offset;

        %ident

        Gs = tf([0.000873 1.413e-07],[1 0.000943 9.836e-08]);

        [ysim tsim] = lsim(Gs,ui,t);

        zpk(Gs)

    %% identifikace 2

        %ident

        Gs2 = tf([ 0.001394 1.22e-06],[1 0.002956 9.032e-07])
        Gs2 = zpk(Gs2)
        [ysim2, tsim2] = lsim(Gs2,ui,t);

        ysim2 = ysim2+offset;

        figure;
        %plot(t,y,tsim2,ysim2+offset,tsim,ysim+offset,'red');
        plot(t,y,tsim2,ysim2);

    %%  export data
        export1 = [t',u,y];
        export2 = [tsim2,ysim2];
```

Příloha H - Návrh regulátoru pomocí kvadratické plochy - skript

```
%% Find PID

    % zadana hodnota
    w = 20;

    % omezeni
    kapa = 0.1;

    % pocatecni konstanty
    P0 = 10;
    I0 = 1;
    D0 = 0.1;

    ropt = fminsearch(@(x) kriterium(x),[P0,I0,D0]);

    P = p
    I = i
    D = d
    kriteruim = J(2)
```

```
%% simulation and return J
    function [ output ] = kriterium(x)

        global p i d j

        p = abs(x(1));
        i = abs(x(2));
        d = abs(x(2));

        sim('simm');

        output = j(2);
    end
```