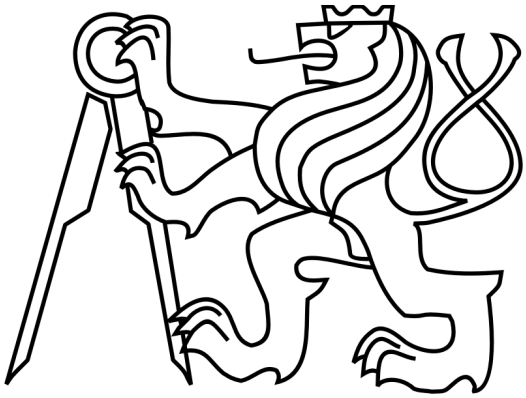
České vysoké učení technické v Praze
Fakulta elektrotechnická



BAKALÁŘSKÁ PRÁCE

Programové vybavení pro experimentální platformu pro distribuované řízení teploty

Praha, 2012 Autor: Petr Cincibus

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 2. 1. 2012

podpis

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Petr Cincibus**

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný
Obor: Kybernetika a měření

Název tématu: **Programové vybavení pro experimentální platformu pro distribuované řízení teploty**

Pokyny pro vypracování:

1. Realizujte programové vybavení pro experimentální platformu pro distribuované řízení teploty.
2. Pro tuto platformu navrhněte a odlaďte software v jazyce C, který bude realizovat nejen samotné distribuované řízení, ale i oboustrannou komunikaci s nadřízeným PC (pro účely zadávání referenčních teplotní profilů a v opačném směru pro účely sběru a vizualizace naměřených dat).
3. Pomocí dvou UART sériových rozhraní může komunikovat každý lokální procesor se svým sousedem.
4. Předveďte funkčnost systému pomocí několika základních úloh jako je udržení nastavené teploty i v přítomnosti rušení (kupříkladu fénem) či průchod definované teplotní vlny tyčí.

Seznam odborné literatury:

C. Rapson. Spatially distributed control: Heat conduction in a rod. Diploma thesis, Czech Technical University in Prague, 2008.

Vedoucí: Ing. Zdeněk Hurák, Ph.D.

Platnost zadání: do konce zimního semestru 2011/2012

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 8. 11. 2010

Poděkování

Mé velké poděkování patří především vedoucímu práce panu Ing. Zdeňku Hurákovi, Ph.D. za odborné vedení, ochotu a přátelský přístup.

Abstrakt

Cílem této bakalářské práce je navrhnout a odladit software pro experimentální platformu pro distribuované řízení teploty. Software by měl být napsán v programovacím jazyce C pro již existující zařízení. To se skládá z deseti samostatných mikroprocesorových jednotek, z nichž každá obsluhuje dva senzory teploty a dva akční členy, které ohřívají hliníkovou tyč. Každá mikroprocesorová jednotka komunikuje se sousedními jednotkami a jedna z krajních jednotek je spojena s počítačem. Ten regulaci řídí a monitoruje.

Abstract

The main objective of this Bachelor thesis is to design and debug software for experimental platform for distributed control of temperature. Software should be written in C programming language for already existing devices that consists of ten separated microprocessor units. Each unit controls two temperature sensors and two actuators, which heat the aluminum rod. Each unit also communicates with it’s two neighbor units. One of the outer units is connected to PC that controls and monitors the whole regulation process.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

1.1 Cíl vlastní práce

Cílem práce je odladit software laboratorního modelu, pro zkoumání prostorově distribuovaného řízení. Po dokončení práce by na tomto zařízení měly být odzkoušeny různé přístupy k řešení distribuovaného řízení. Tato práce by měla zároveň částečně zdokumentovat výsledky práce Jana Kříže, který dokončil hardwarovou stránku zařízení a pokusil se naprogramovat přípravek a částečně připravil základní funkce pro příjem dat v matlabu. Jeho práce ovšem není nikde zdokumentována.

1.2 Motivace pro celý projekt

1.2.1 Způsoby přístupu k řízení systémů s rozprostřenými parametry

Systém s rozprostřenými parametry je takový systém, který je složen z více identických částí. Tyto identické části si lze představit jako pole navzájem propojených systémů. Podmínkou pro systém s rozprostřenými parametry je, že každý z prvků svým chováním ovlivňuje sebe, ale zároveň i prvky ve svém nejbližším okolí. V případě že tato podmínka není splněna, jedná se pouze o velké množství nezávislých systémů umístěných vedle sebe [2]. Pro efektivní regulaci takovéhoto systému je zapotřebí uvažovat nejen vlastní chování subsystému, ale i chování subsystémů v nejbližším okolí.

Rozlišujeme tři základní typy přístupu k řízení systémů s rozprostřenými parametry. Jeden z nich je centralizované řízení, kde všechny naměřené hodnoty jsou vyhodnocovány pouze jediným regulátorem, který ovládá větší množství akčních členů. Nevýhodou tohoto řešení je, že nutnost svést všechny signály do jednoho bodu klade vysoké nároky na kabeláž. Velký počet vstupních hodnot a výpočet většího počtu akčních zásahů může zabrat relativně dost času. Navíc v případě že na regulátoru nastane porucha, je celý systém vyřazen z provozu. Tedy je nutná vysoká spolehlivost zařízení. Všechny tyto nároky se promítnou ve výsledné ceně zařízení [1].

Většinu nevýhod centralizovaného řízení odstraňuje koncepce řízení decentralizovaného. Tento přístup preferuje rozdělit řízení na větší počet částí, kde každý subsystém má svůj regulátor. Větší počet regulátorů s nižšími nároky na počet vstupů, výstupů a výpočetní kapacitu zajistí nižší cenu zařízení a zvýší spolehlivost. Dále se podstatně sníží nároky na kabeláž. Systém je také snadno rozšiřitelný. Stačí pouze přidat další regulátory. Oproti tomu u centralizovaného řízení je zapotřebí vyměnit nebo modifikovat hlavní regulátor, kvůli změně počtu vstupů a výstupů. Nevýhodou decentralizovaného přístupu k řízení je fakt, že regulace je omezena pouze v rámci subsystému.

Efektivnější přístup k řízení je řízení distribuované, které vychází z vlastností decentralizovaného řízení a navíc uvažuje při regulaci i hodnoty ze sousedních subsystémů. Při poruše na některém regulátoru, mohou okolní regulátory, tuto závadu kompenzovat.

1.2.2 Distribuované řízení teplotního profilu tyče

Některé systémy s rozprostřenými parametry, které vyžadují distribuované řízení mají v prostoru dva rozměry. V našem případě je hliníková tyč jednodušší systém s jedním rozměrem v prostoru. Díky tomu je snadnější reprezentovat naměřené veličiny například v 3D grafu. Na jednu z os bude vynesena naměřená hodnota. Na druhou osu budou vyneseny čísla regulačních buněk. Na třetí osu bude vynesen čas, což je další dimenze systému.

Při regulaci si budou mezi sebou sousední buňky vyměňovat informace o naměřených hodnotách teploty, které jsou potřeba pro výpočet akčního zásahu. Dále si buňky mohou vyměňovat informace o hodnotě akčního zásahu, které mohou být spolu s hodnotami naměřené teploty použity pro výpočet akčního zásahu.

1.3 Historie projektu řízení teplotního profilu hliníkové tyče

Historii práce na projektu bych rozdělil na tři části. Jako první započal práci na projektu Chris Rapson, který v rámci své diplomové práce popsal rovnice a vlastnosti šíření tepla hliníkovou tyčí. Jeho model zahrnuje i konečnou délku tyče. Na tomto modelu pak odzkoušel regulátory, které vycházely z centralizované, decentralizované a distribuované koncepce řízení systému s rozprostřenými parametry. V závěru své práce vyhodnotil jako nejlepší regulátory FIR – PID, LQR a PI.

Václav Klemš měl v rámci své bakalářské práce navrhnout a zrealizovat hardware zařízení. Rozhodl se teplotu hliníkové tyče řídit pomocí dvaceti regulátorů. Pro zjednodušení zařízení není každý regulátor umístěn na zvláštní desce plošných spojů, ale v rámci jedné desky s mikroprocesorem jsou realizovány dvě regulační buňky. Celkem tedy deset desek plošných spojů (modulů). Pro tyto moduly zhotovil schéma zapojení a navrhl desku plošných spojů. Na hliníkovou tyč připevnil snímače teploty a výkonové tranzistory. Pro napájení modulů vybral počítačový zdroj s výkonem 979 W. V rámci jeho práce byly implementovány a otestovány funkce pro měření teploty. Další práce už na projektu nestihl provést.

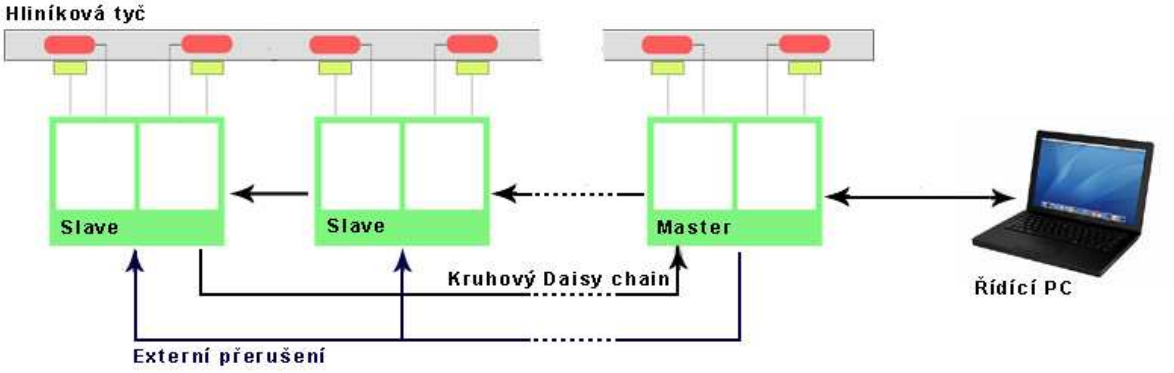
Následně na projektu pracoval Jan Kříž, který upravil schéma a zapojení modulů, které vyrobil a oživil. Pro komunikaci modulů byla zvolena lineární forma Daisy Chain. Komunikaci se podařilo zprovoznit jen z části, protože velký počet krátkých zpráv zahlcoval master modul. Ten nestíhal komunikaci obsluhovat. Dále implementoval funkce matlabu pro připojení k sériové lince a obsluhu přijmu zpráv v řídícím počítači.

.

2 Popis hardware

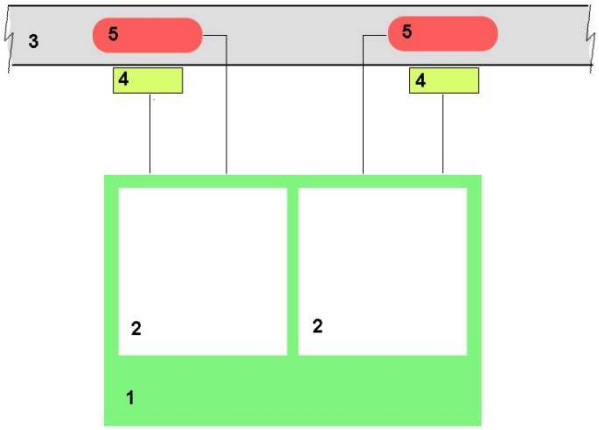
Úkolem zařízení je zahřívat kovovou tyč na požadovanou teplotu. Tato tyč má obdélníkový profil 2 cm na šířku a 1 cm na výšku. Délka tyče je 1 m. Na širší straně je s krokem 5 cm přilepeno 21 výkonových tranzistorů GM BD912, které mají funkci akčního členu a tedy tyč zahřívají. U každého tranzistoru je na užší straně přilepen snímač teploty SMT 160-30. Tyč je na obou koncích zavěšena na drátkách natažených mezi distančními sloupky. Drátky mají silnou vrstvu izolace, díky níž je přesun tepla mezi tyčí a závěsným zařízením minimální. V případě, že by tyč byla připevněna přímo k hliníkovým distančním sloupkům, byl by odvod tepla výrazně větší, což by mohlo podstatně narušit matematický model systému.

Pod tyčí je umístěno deset modulů desek plošných spojů. Každý modul je realizován jedním mikrokontrolerem Atmel ATmega162, který obsluhuje dvě sériová rozhraní USART. Každý USART je vyveden na čtyrpinový konektor, který kromě pinů pro přenos dat (TXD0 pro USART0 a TXD1 pro USART1) a příjem dat (RXD0 pro USART0 a RXD1 pro USART1) má vyveden ještě pin pro vnější přerušení (INT0 společný pro oba konektory) a pin napěťové země (GND).



Obrázek 2.1 Schéma přípravku

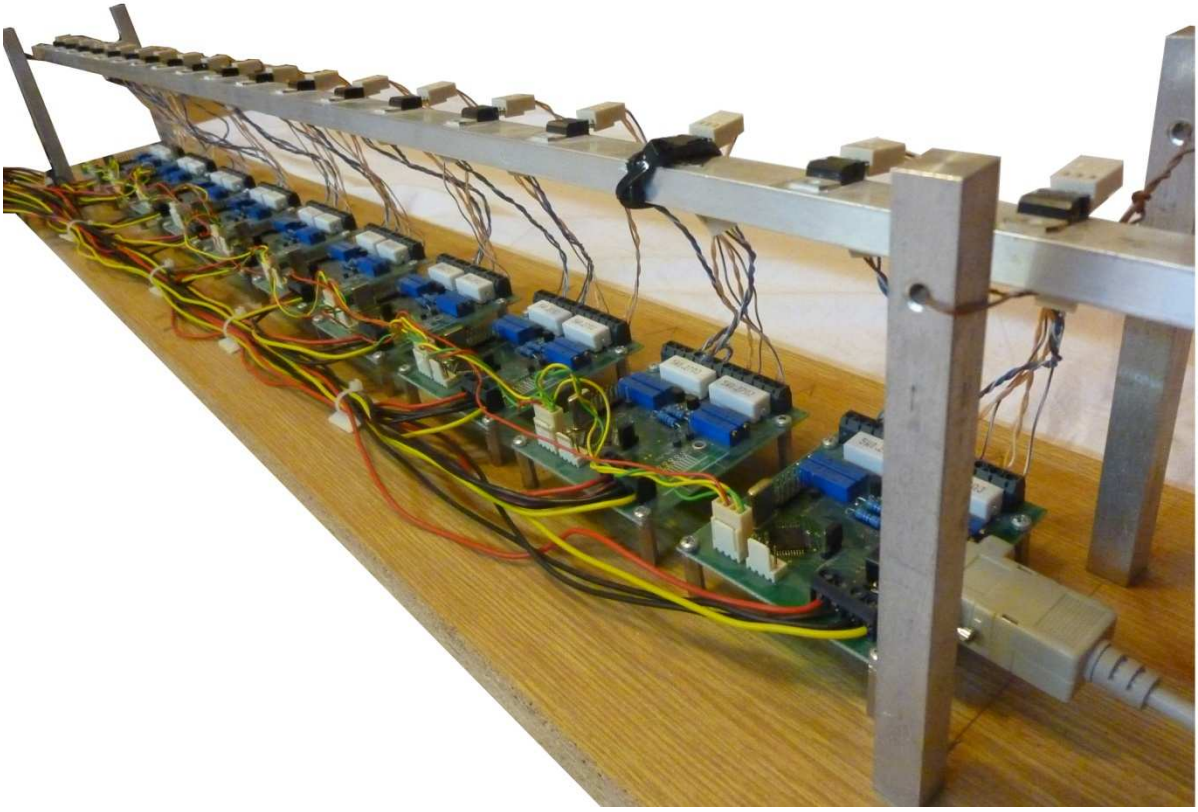
Řešení jednoho z modulů je na obrázku 2.1. Každý modul obsluhuje dva akční členy a dva měřící členy. Tedy můžeme říci, že obsahuje dvě buňky, z nichž každá zastává funkci regulátoru s jedním akčním členem a jedním měřícím členem. Na každém modulu je přepínač, kterým se dá zvolit, zda je daný modul ve funkci master nebo slave. Jeden z krajních modulů je navíc vybaven převodníkem z USART na RS232 a konektorem pro připojení k řídícímu počítači. Tento převodník je připojen na USART0. Moduly jsou napájeny z počítačového zdroje, který má dva napěťové okruhy. První okruh má napětí 12 V s výkonem 900 W a je využit na napájení topných tranzistorů. Druhý okruh má napětí 5 V s výkonem 150 W a je využíván pro napájení integrovaných obvodů. Celé zařízení je připevněno k dřevotřískové desce.



Obrázek 2.2 Modul mikroprocesoru

1	Modul mikroprocesoru
2	Regulační buňka
3	Hliníková tyč
4	Snímač teploty
5	Výkonový tranzistor

Tabulka 2.1 Popis obrázku 2.2



Obrázek 2.3 Fotografie zařízení

3 Popis Softwaru

3.1 Komunikace na nízké úrovni

3.1.1 Koncepce

Pro spojení modulů rozhraním USART byla původně vybrána koncepce Daisy chain v lineární formě. To v našem případě znamená, že na začátku řetězce je master modul, který je přes USART1 spojen s USART0 souseda. Tohoto souseda nazveme slave 1. Stejně tak je i slave 1 spojen přes USART1 s USART0 modulu slave 2 a tak dále s výjimkou posledního modulu v řadě, který má USART1 nezapojen. K master modulu je na USART0 připojen řídící počítač přes převodník max232. Komunikace mezi moduly je duplexní.

Pro každou informaci, byla vytvořena zpráva, která má kromě dané informace ještě jeden bajt pro identifikaci typu zprávy, bajt s číslem buňky a jeden bajt pro ukončení zprávy. Tedy zpráva nesoucí jednobajtovou informaci má ve výsledku 4 bajty. Zprávy, které byly určeny pro řídící počítač, byly posílány pouze na USART0, kde je přijal nejbližší soused s nižším identifikačním číslem a předal dalšímu. Nakonec se přes master modul zpráva dostala do řídícího počítače. Naopak zprávy, které byly určeny pro všechny moduly, byly posílány jak na USART0. Zde se zpráva zpracovávala tak, jak jsem popsal u zpráv určených jen pro řídící počítač, tak na USART1 kde ji přebral sousedící modul s vyšším identifikačním číslem. Ten zprávu zpracoval a poslal dále, až informace došla k poslednímu modulu v řetězci, který ji zpracoval a dále již neposílal.

Nevýhoda tohoto řešení je, že zprávy jsou příliš dlouhé vzhledem k informaci, kterou nesou. Tato koncepce navíc přináší problém, že počet zpráv, které jsou modulem zpracovány je vyšší pro moduly s nižším identifikačním číslem a dá se říct že se zprávy hromadí směrem k master modulu. Ten je přetížen a nezvládá obsluhu všech zpráv. Řešení tohoto problému je spojit všechny zprávy jednoho typu do jedné zprávy, jejíž délka bude odvozena z typu zprávy a počtu buněk v systému, jenž musí být předem zjištěn. Tyto dlouhé zprávy budou vyslány master modulem a každý modul do nich doplní svoji informaci na předem určená místa, popřípadě si některé informace uloží a pošle zprávu dál. Navíc informaci z které buňky hodnota pochází nebo naopak pro kterou buňku je hodnota určena, nese hodnota svým umístěním ve zprávě.

Když hromadná zpráva doputuje na konec řetězce, měla by být odeslána zpátky a putovat přes všechny moduly až k řídícímu počítači. Tuto cestu nazpět, můžeme zjednodušit tím, že místo lineární formy Daisy chain, zvolím kruhovou formu Daisy chain. Toho docílíme spojením posledního modulu s prvním master modulem, který může zprávy rovnou odesílat do řídícího počítače, aniž by putovaly znovu přes všechny moduly systému. V tomto případě jsou moduly spojeny mezi sebou přes USART1 a to tak, že TX pin modulu je spojen s RX pinem modulu sousedního. TX pin posledního modulu je spojen s RX pinem master modulu. Všechny slave moduly tedy využívají pouze USART1 a USART0 je volný. Komunikace mezi moduly je tedy simplexní. Master modul jako jediný používá i USART0, kterým komunikuje s řídícím počítačem. Tato komunikace je plně duplexní.

3.1.2 Buffery

Každý modul má čtyři kruhové buffery. Každé dva jsou vyhrazeny pro jeden USART, z nichž jeden pro příjem dat a druhý pro odesílání dat. Pro zjednodušení práce s bufferem musí být jeho velikost dána vztahem (3.1). Díky tomu je zjednodušeno indexování. Při změně indexu je hodnota ukazatele bitově vynásobena velikostí bufferu. Díky tomu se nemůže stát, že by index ukazoval mimo rozsah bufferu.

$$s = 2^n, pro\ n \in \mathbb{N}^+ \quad (3.1)$$

Pro každý buffer jsou čtyři indexi. Dva pro zápis, kde jeden tzv. hlava ukazuje na začátek právě zapisované zprávy a nebo na první volné místo v případě, že aktuálně není zapisována žádná zpráva. Druhý index ukazuje vždy na první volné místo pro zápis. V případě, že právě není zapisována žádná zpráva, se hodnoty těchto indexů shodují. Další dva jsou pro čtení z bufferu. Jeden tzv. hlava ukazuje na začátek právě čtené zprávy nebo na začátek první nepřečtené zprávy. Druhý index ukazuje na právě čtený bajt. V případě, že jsou všechny zprávy přečteny, ukazují čtecí indexi na první volné místo v bufferu.

3.1.3 Příjem a odesílání zpráv

Příjem a odesílání zpráv je na lince USART obsluhováno pomocí přerušení. Přerušení pro odesílání je vyvoláno softwarově v každé programové smyčce. Přerušení pro příjem je vyvoláno hardwarově při přijmu dat.

Při příjmu dat program nejprve ověří, zda se jedná o začátek zprávy, jejíž první bajt identifikuje typ zprávy. Podle něj zjistí velikost zprávy a přijímá další bajty zprávy a ukládá do bufferu pro příjem dat. V případě že poslední bajt zprávy není ukončovací znak (hodnota 10), tak se přijatá data zneplatní. Pokuď funkce pro odesílání dat zjistí, že v bufferu je připravena zpráva pro odeslání, začne zapisovat bajty zprávy na příslušný pin, dokuď zprávu neodešle celou.

3.2 Inicializace

Při startu systému je třeba nejprve všechny buňky očíslovat identifikačním číslem. Číslování začne od jedničky a číslo se inkrementuje pro každou další buňku.

Master modul nejprve očísluje vlastní buňky a potom vyšle zprávu IDENT s číslem vyšší buňky. Tuto zprávu přijme sousední slave modul, který očísluje vlastní dvě buňky a pošle zprávu IDENT s identifikačním číslem jeho vyšší buňky na další modul. Takto se očíslují všechny moduly, až zpráva dorazí zpět k master modulu, kde již číslo poslední buňky z přijaté zprávy IDENT odpovídá celkovému počtu buněk v systému. Informace o počtu buněk je odeslána master modulem všem modulům ve zprávě NODECOUNT. Následně je možné uvést systém do chodu pomocí z počítače odeslané zprávy RUN s parametrem RUN_AUTO.

3.3 Reset

Při vývoji programu nastávaly situace, kdy jsem potřeboval přehrát program pouze na slave modulech. Při spuštění aplikace se systém zinicializoval (viz. Kapitola 3.2) a následně byl na některém ze slave modulů přehrán nebo zresetován program. V tu chvíli byl daný modul bez identifikačních čísel buněk a neznal celkový počet buněk systému, proto ani nemohl přijímat zprávy, jejichž délka se odvíjí od celkového počtu buněk v systému. Z tohoto důvodu byl do systému zaveden požadavek o novou inicializaci prostřednictvím zprávy IDENTRESET (viz. Kapitola 3.5.6).

V obsluze příjmu zprávy TEMP je vložena kontrola inicializace. V případě že modul není zinicializován, vyšle modul zprávu IDENTRESET, která po přijetí master modulem spustí novou inicializaci systému. V řídícím počítači novou inicializaci poznáme přijetím zprávy NODECOUNT. Zpráva IDENTRESET se do řídícího počítače neposílá.

3.4 Regulační cyklus

Pro zjednodušení matematického popisu regulace je zapotřebí, aby některé kroky regulace proběhly v přibližně stejných okamžicích. Začátek regulačního cyklu určuje časovač v master modulu. Ten generuje napěťový impulz na portu INT0, který je spojen s porty ostatních slave modulů. Na rozdíl od master modulu, kde je port INT0 nastaven jako pouhý výstup, je na slave modulech nastaven jako zdroj externího přerušení. V obsluze přerušení se zajistí měření teploty. Protože na master modulu se spustí měření hned po vygenerování signálu pro přerušení, můžeme říci, že měření teploty proběhne na všech modulech v přibližně stejný okamžik.

Po dokončení měření vyšle master modul zprávu TEMP, která zajistí roznesení informací o teplotách v systému (viz. Kapitola 3.5.5). Následně generuje master modul signál přerušení, pro synchronizaci výpočtu akčních zásahů. Poté vyšle master modul zprávu ACTION, která informuje řídící počítač o akčních zásazích (viz. Kapitola 3.5.7). Po zbytek doby regulačního cyklu se pouze udržuje hodnota akčního zásahu a po signalizaci časovače se cyklus opakuje.

3.5 Typy zpráv

3.5.1 Ident

Zpráva slouží k identifikaci a očíslování identifikačními čísly jednotlivých regulačních buněk systému. Délka zprívy je 3 bajty. Následující tabulka určuje formát zprávy.

Bajt zprávy	Význam	Popis
1	Identifikační znak	Char ‘i’
2	Identifikační číslo souseda	Uint8
3	Ukončovací znak	LF

Tabulka 3.1 Popis částí Ident zprávy

3.5.2 Nodecount

Zpráva nese informaci o celkovém počtu regulačních buněk v systému. Délka zprávy je 3 bajty.

Bajt zprávy	Význam	Popis
1	Identifikační znak	Char ‘n’
2	Počet buněk systému	Uint8
3	Ukončovací znak	LF

Tabulka 3.2 Popis částí Nodecount zprávy

3.5.3 Run

Zpráva přepíná moduly do jednoho z pracovních módů systému (viz. Kapitola 3.9). Délka zprávy je 3 bajty.

Bajt zprávy	Význam	Popis
1	Identifikační znak	Char ‘r’
2	Pracovní mód	Uint8
3	Ukončovací znak	LF

Tabulka 3.3 Popis částí Run Zprávy

3.5.4 SetTemp

Zpráva nese požadované hodnoty teplotního profilu tyče. Tyto hodnoty se ukládají do Rm paměti mikrprocesrů a zároveň i do EEPROM paměti. Díky tomu i po vypnutí napajení zůstanou požadované teploty uloženy.

Bajt zprávy	Význam	Popis
1	Identifikační znak	Char ‘s’
2 až 1+2*nodecount	Požadovanéteploty	Uint16
2 + 2*nodecount	Ukončovací znak	LF

Tabulka 3.4 Popis částí SetTemp zprávy

3.5.5 Temp

Zpráva nese informace o aktuální teplotě úseků tyče naměřených regulačními buňkami. Zpráva dále obsahuje status bajt, který nese informaci o platnosti teplot ve zprávě.

Hodnota	Význam
0	Data neplatná
1	Data platná

Tabulka 3.5 Hodnoty statusu Temp zprávy

V první fázi se sbírají informace o naměřených teplotách. Zprávu vyšle master modul který doplní teploty naměřené vlastními regulačními buňkami. Teploty ostatních buněk, které mu nenáleží, nastaví na hodnotu 0. Status bajt je nastaven na hodnotu data neplatná. Slave moduly si při příjmu zprávy nejprve ověří hodnotu status bajtu. Po zjištění stavu Data neplatná, doplní do zprávy teploty svých buněk a pošle dále. K master modulu dorazí zpráva s již všemi hodnotami teplot.

V druhé fázi se informace o naměřených teplotách posílají do systému. Master si hodnoty z přijaté zprávy zapíše do tabulky teplot a přijaté zprávě změní status byte na hodnotu data platná. Následně zprávu odešle do systému a stejně tak i řídícímu počítači. Slave moduly si ze zprávy ukládají informace o naměřených teplotách do tabulky teplot. Zpráva zanikne po přijetí master modulem.

Bajt zprávy	Význam	Popis
1	Identifikační znak	Char ‘t’
2	Status	Uint8
3 až 2+2*nodecount	Naměřené teploty	Uint16
3+2*nodecount	Ukončovací znak	LF

Tabulka 3.6 Popis částí Temp zprávy

Teploty jsou seřazeny podle identifikačního čísla regulační buňky, která teplotu naměřila od nejnižšího po nejvyšší.

3.5.6 IdentReset

Zpráva je žádostí o reset inicializace systému. To znamená že master modul po přijmutí této zprávy vyšle zprávu IDENT a následně NODECOUNT. Délka zprávy je 2 bajty.

Bajt zprávy	Význam	Popis
1	Identifikační znak	Char ‘d’
2	Ukončovací znak	LF

Tabulka 3.7 Popis částí IdentReset zprávy

3.5.7 Action

Zpráva nese informaci o akčních zásazích regulačních buněk. Její délka je odvozena z počtu buněk.

Bajt zprávy	Význam	Popis
1	Identifikační znak	Char ‘a’
2 až 1+nodecount	Akční zásah	Uint8
2+nodecount	Ukončovací znak	LF

Tabulka 3.8 Popis částí Action zprávy

Akční zásahy jsou seřazeny podle identifikačního čísla regulační buňky, která teplotu naměřila od nejnižšího po nejvyšší.

3.5.8 Coef

Zpráva nese nové hodnoty koeficientů pro jeden řádek tabulky koeficientů. Číslo řádku je určeno druhým bajtem zprávy. Ideální by bylo přenést celou tabulku koeficientů najednou, ovšem zpráva by byla tak dlouhá, že by se nevešla do bufferů pro příjem a odesílání dat. Na rozdíl od zprávy SETTEMP se koeficienty neukládají do EEPROM paměti, takže po resetu napájení sou koeficienty nastaveny na původní hodnoty.

Bajt zprávy	Význam	Popis
1	Identifikační znak	Char ‘c’
2 až 1+2*COL_NUM	Koeficienty	Uint16
2+ 2*COL_NUM	Ukončovací znak	LF

Tabulka 3.9 Popis částí Coef zprávy

Kde

- COL_NUM je počet sloupců tabulky koeficientů

3.6 Měření

Způsob měření teploty popsal ve své bakalářské práci Laboratorní model pro výzkum prostorově distribuovaného řízení Václav Klemš. V jeho implementaci měření jsem změnil způsob zpracování výsledků, abych dosáhl přesnějších hodnot. Původní algoritmus změřil více hodnot, z kterých následně vypočetl průměrnou hodnotu. To ve výsledku znamená, že každé špatné měření ovlivní výslednou hodnotu. Já jsem se rozhodl udělat větší počet měření, z kterých nakonec budu považovat za správnou hodnotu jejich medián. Přesnost měření je pak závislá na počtu měření, z kterých se medián počítá.

Pro výpočet mediánu je pole hodnot nejprve setříděno a následně je vybrán prostřední prvek. Abych snížil velikost kódu funkce, omezil jsem počet měření pouze na lichý počet. V případě sudého počtu měření bych musel z prostředních hodnot počítat průměr. Hlavní požadavek na algoritmus pro setřídění pole je velikost kódu, protože počet prvků bude relativně malý, tak časová složitost algoritmu není nejdůležitější. Z tohoto důvodu jsem se rozhodl pro třídění pole implementovat třídící algoritmus Bubble Sort.

3.7 Regulátor

3.7.1 Struktura regulátoru

Implementovaný regulátor je typu FIR s integrační složkou. Výpočet je popsán vztahem (3.2). První část výpočtu výstupní veličiny vynásobí hodnotu výstupní veličiny v minulém kroku

s koeficientem ‘a’. Druhá část je suma regulačních odchylek všech sousedních buněk až do hloubky ‘n’ včetně regulačních odchylek vlastní buňky. Druhá suma rozšiřuje tento výpočet i do historie až do hloubky ‘m‘. Regulační odchylky jsou před součtem vynásobeny příslušným koeficientem ‘b’.

Přenos v Laplaceově transformaci je určen vztahem (3.3). Kromě časového operátoru je do přenosu třeba zavést operátor, který určuje polohu buňky v prostoru.

$$u_i(k) = a \cdot u_i(k-1) + \sum_{x=0}^{m} \sum_{y=-n}^{n} b_{xy} \cdot e_{i+y}(k-x) \tag{3.2}$$

$$K(z,w) = \frac{U(z,w)}{E(z,w)} = \frac{\sum_{x=0}^{m} \sum_{y=-n}^{n} b_{xy} \cdot z^{-x} \cdot w^{y}}{1 - a \cdot z^{-1}} \tag{3.3}$$

Kde

- i index buňky pro kterou je akční zásah vypočítáván
- y index sousedních buněk
- x index historie měření
- m hloubka historie
- n počet sousedních buněk uvažovaných pro regulaci
- z časový operátor
- w prostorový operátor
- a,b koeficienty regulátoru

3.7.2 Výpočet akčního zásahu

Funkce pro výpočet akčního zásahu je v souboru action.c . Název funkce je **calculateAction()**. V rámci výpočtu akčního zásahu je navíc posunuta tabulka s teplotami o jeden řádek dolů při čtení jejích hodnot. Tím ušetříme jeden celý průchod tabulkou, který by jsme museli udělat, kdyby bylo posunutí implementováno ve zvláštní funkci. Při případné implementaci jiných regulátorů je nutno dodržet tento postup.

Pro výpočet akčního zásahu je třeba načíst koeficienty akčního zásahu, které jsou uloženy v EEPROM paměti v dvourozměrném poli typu uint16. Velikost tabulky koeficientů je o jeden sloupec menší než velikost tabulky teplot. Počet sloupců je tedy lichý. Při výpočtu zásahu se tabulka teplot maskuje tabulkou koeficientů a to tak, že prostřední sloupec tabulky koeficientů je nad sloupcem tabulky teplot, v kterém jsou uloženy teploty buňky, pro kterou je právě vypočítáván akční zásah. Koeficienty se vynásobí s teplotami a následně sečtou. Dále je přičtena hodnota akčního zásahu v minulém kroku vynásobena příslušným koeficientem. Tento koeficient nelze nastavovat z řídícího počítače, ale pouze při programování modulu. Po vydělení konstantou 10000 se ověří, zda výsledek není větší než limitní hodnota, která je uložena v EEPROM paměti. V případě že je vypočítaná hodnota větší, je nahrazena limitní hodnotou. Při výpočtu akčního zásahu pro druhou buňku modulu se tabulka koeficientů posune o jeden sloupec doprava, tak aby byl opět prostřední sloupec tabulky koeficientů nad sloupcem tabulky teplot, v němž jsou uloženy teploty buňky, pro kterou je právě vypočítáván akční zásah. Matice (3.4) naznačuje organizaci tabulky koeficientů.

$$\begin{pmatrix} b_{0,-n} & \cdots & b_{0,n} \\ \vdots & \ddots & \vdots \\ b_{m,-n} & \cdots & b_{m,n} \end{pmatrix} \tag{3.4}$$

3.8 Tabulka teplot

Tabulka slouží pro ukládání teplot naměřených sousedními buňkami, které můžou ovlivňovat vlastní teplotu v regulovaném bodě. Počet sousedních buněk, které budeme brát pro regulaci v potaz byl zvolen na tři regulační buňky z každé strany. Dále je třeba ukládat i teploty vlastních regulačních buněk. Tabulka má 8 řádku a 8 sloupců. Každý řádek reprezentuje teplotní profil v jednom regulačním cyklu. První hodnota v řádku reprezentuje 3. sousední buňku z horní strany. Teploty sousedních regulačních buněk jsou seřazeny sestupně podle vzdálenosti od uvažovaného modulu. Tedy třetí hodnota je teplota nejbližší sousední buňky z dolní strany. Čtvrtá a pátá hodnota jsou teploty vlastních buněk. Šestá, sedmá a osmá hodnota jsou teploty prvního, druhého a třetího nejbližšího souseda z horní strany. Při výpočtu akčního zásahu se řádky posunou o jeden dolů, přičemž informace uložené v posledním řádku tabulky zaniknou. Na první řádek se dosadí hodnoty nově naměřené. Do tabulky se ukládají teploty typu uint16.

3.9 Pracovní módy modulu

3.9.1 Run stop

V tomto módu je modul omezen pouze na komunikaci se sousedními buňkami. Není prováděno měření teplot ani ohřívání tyče.

3.9.2 Run auto

V tomto módu se provádí jak ohřívání tyče, tak měření teploty.

3.10 Reprezentace teploty

Pro reprezentaci teploty by jsme mohli využít čtyřbajtový datový typ float, který pracuje v rozsahu 1E-37 až 1E+37 (6 desetinných míst). V celém systému měříme a pracujeme s teplotními hodnotami se dvěmi desetinnými místy. Abychom ušetřili místo v paměti, byl zvolen datový typ uint16, který zabírá oproti typu float poloviční místo v paměti, tedy 2 bajty. Ukládanou teplotu vynásobíme 100x, čímž zajistíme celočíselnou hodnotu.

$$t_{ukládaná} = 100 \cdot t_{opravdová} \tag{3.5}$$

Pro teploty používáme Celsiovu stupnici. Tedy hodnota 0 odpovídá $0°C$.

3.11 Popis softwaru Řídícího počítače

Pro řídící počítač byl již mými předchůdci vybrán program Matlab. Hlavní úkoly softwaru řídícího počítače jsou příjem zpráv, řízení regulace a zpracování naměřených dat.

3.11.1 Ukládání dat

Proměnné, které nesou informace o nastavení zařízení a do kterých se ukládají informace o regulaci jsou inicializovány při prvním spuštění funkce **heat_config()** . Tato funkce navíc proměnné zviditelní pro oblast v které byla spuštěna. Proto je **heat_config()** volána na začátku každé funkce, která pracuje s těmito proměnnými. Pokuď chceme proměnné zviditelnit v hlavním okně matlabu nebo chceme uložit workspace, musíme funkci **heat_config()** spustit z příkazové řádky matlabu. Pokud chceme uložené workspace opětovně použít, je třeba pře načtením spustit funkci **heat_config()**, aby se proměnné inicializovaly.

Naměřená teplota a hodnota akčního zásahu se ukládají do proměnných **node_temp** a **node_action**, což jsou matice kde řádky reprezentují vývoj systému v čase a sloupce reprezentují regulační buňky. Čas příchodu zprávy s naměřenými teplotami, který odpovídá začátku regulačního cyklu se ukládá do matice **node_tempTime** ve formátu, který vrací funkce **clock().**

Defaultně je požadovaná teplota nastavena na hodnotu $0\ °C$. Při odesílání TEMP zprávy s nově požadovanými teplotami se požadované teploty uloží do vektoru **node_desiredTemperatureActual**. Ten reprezentuje aktuální požadovaný teplotní profil. Pro uchování historie požadovaných teplotních profilů, se hodnoty z vektoru aktuální požadovaně teploty **node_desiredTemperatureActual** ukládají do matice **node_desiredTemperature** při každém příchodu zprávy s naměřenou teplotou, abychom věděli, k jakému teplotnímu profilu se systém v daném regulačním cyklu snažil přiblížit.

Všechny údaje o teplotě jsou ukládány v jednotkách Celsiovy stupnice.

3.11.2 Inicializace

Nejprve je třeba připojit se k sériové lince počítače. To se provede spuštěním funkce **heat_connect().** Parametry nastavení sériové linky, jako třeba port, přenosová rychlost, parita atd. se dají nastavit ve funkci **heat_config().** Dále je třeba zjistit počet regulačních buněk (nodecount). Zařízení se po spuštění zinicializuje a pošle do řídícího počítače zprávu s počtem regulačních buněk. Pokud tedy přípravek připojíme k napájení až po spuštění funkce **heat_connect()** zpráva s počtem regulačních buněk přijde automaticky. Pokud je přípravek zapnut dříve, musíme se na počet regulačních buněk dotázat vysláním zprávy pomocí funkce **heat_queryNodecount().** Pro odpojení od sériové linky a její uvolnění slouží funkce **heat_disconnect()**. Pokuď chceme připojení resetovat například kvůli vynulování proměnných s naměřenými daty, můžeme použít funkci **heat_restart()**, která jednoduše zavolá funkci **heat_disconnect()** a následně funkci **heat_connect().**

3.11.3 Reprezentace dat

Pro reprezentaci naměřených hodnot lze použít funkce pro zobrazení dat do 3D grafů. Pro vykreslení grafu s naměřenými teplotami slouží funkce **heat_plotTemperature()**, pro zobrazení akčních zásahu je určena funkce **heat_plotAction()** a pro zobrazení grafu s požadovanými teplotami je určena funkce **heat_plotDesiredTemperature()**. Na ose „x“ je vynesen čas měření počínaje hodnotou 0 sekund. Na ose „y“ jsou vyneseny regulační buňky a na ose „z“ jsou naměřené hodnoty. Dále lze uložit data do workspace (viz. Kapitola 3.11.1).

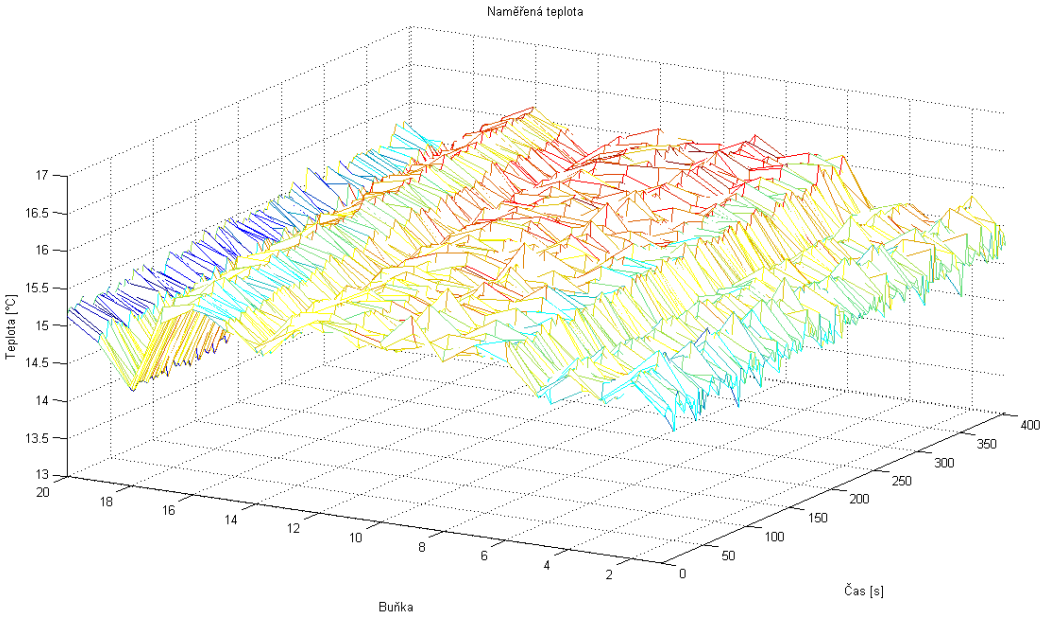
3.11.4 Funkce řídícího PC

- **heat_config –** inicializuje proměnné
- **heat_connect –** připojení k sériové lince
- **heat_disconnect –** odpojení od sériové linky
- **heat_restart –** restart připojení sériové linky
- **heat_queryNodecount –** dotaz na počet inicializovaných regulačních buněk
- **heat_ident –** požadavek na novou inicializaci systému
- **heat_setRunAuto –** uvedení systému do stavu RUN_AUTO
- **heat_setRunStop** - uvedení systému do stavu RUN_STOP
- **heat_setCoefRow** - nastaví jeden řádek koeficientů regulátoru
- **heat_setCoef** – nastaví celou tabulku koeficientů
- **heat_step** – provede skok systému z teploty t_1 na teplotu t_2
- **heat_wave –** provede průchod teplotní vlny
- **heat_cascade –** provede několik teplotních skoků za sebou
- **heat_plotTemperature –** zobrazí 3D graf naměřených teplot
- **heat_plotDesiredTemperature –** zobrazí 3D graf požadovaných teplot
- **heat_plotAction –** zobrazí 3D graf hodnot akčních zásahů

3.12 Kompenzace chyb

V případech kdy lze přepokládat, že tyč má ve všech bodech přibližně stejnou teplotu (například ve chvíli kdy už dlouho nebyla zahřívána by měla mýt ve všech bodech pokojovou teplotu) jsem zjistil, že hodnoty ze senzorů se dosti liší. Proto jsem se rozhodl zavést do systému kompenzaci chyby offsetu.

Pro zjištění této chyby je třeba změřit větší množství vzorků při konstantní teplotě, tedy bez zahřívání tyče. Toho lze jednoduše dosáhnout tím, že požadovanou teplotu nastavíme na 0 °C.



Obrázek 3.1 Naměřené hodnoty pokojové teploty

Následně spočteme průměrnou teplotu t_i pro každý senzor. Dále spočteme opravdovou teplotu t a kompenzační konstantu c_i pro každý senzor podle vztahu (3.8).

$$t_i = \frac{1}{N}\sum_{j=1}^{N} t_{ij} \tag{3.6}$$

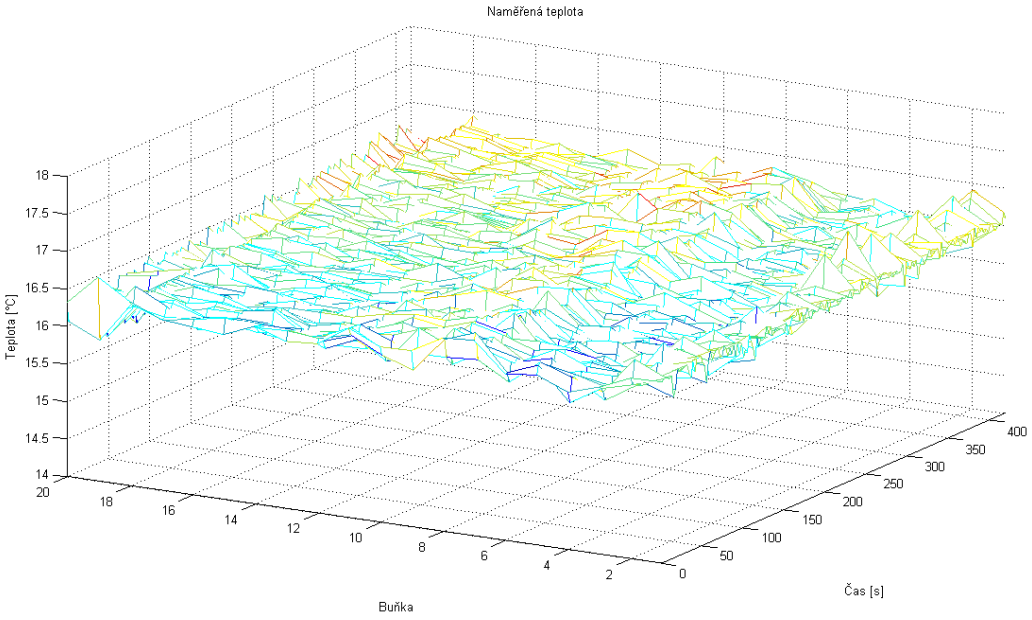
$$t = \frac{1}{I}\sum_{i=1}^{I} t_i \tag{3.7}$$

$$c_i = t - t_i \tag{3.8}$$

Kde

- N je počet vzorků
- j je číslo měření
- i je index senzoru
- I je počet senzorů
- t_i je průměrná naměřená hodnota i-tým senzorem
- t je průměrná teplota tyče
- c_i je kompenzace i-tého senzoru

Na obrázku 3.2 je vidět jak kompenzace chyb zlepšila měření teploty. Oproti měření teploty bez kompenzace je charakteristika vyrovnanější.



Obrázek 3.2 Naměřené hodnoty pokojové teploty s kompenzací chyby offsetu

4 Měření

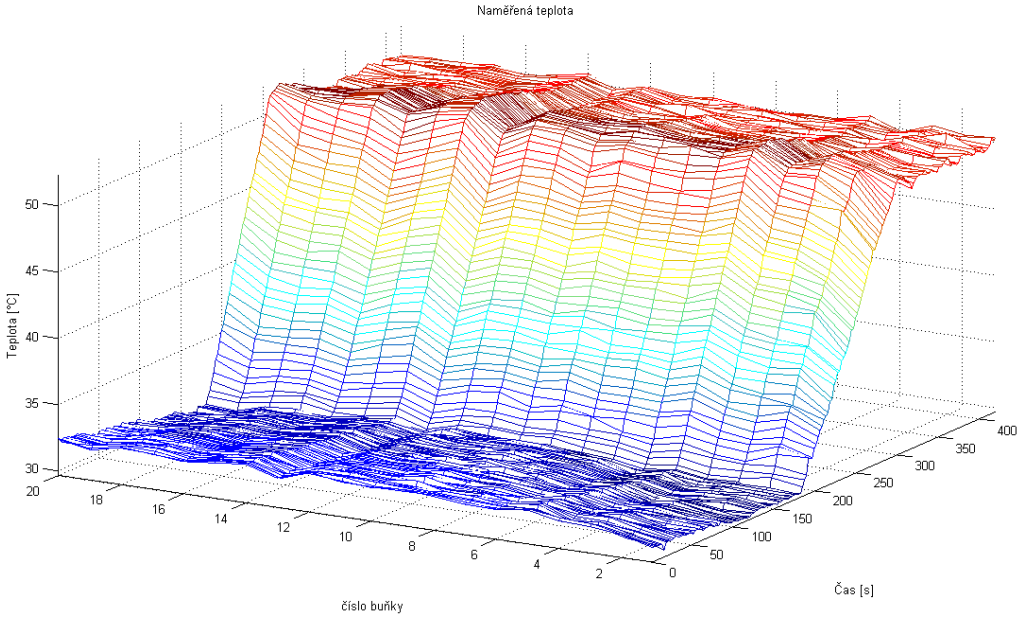
4.1 Měření charakteristik

4.1.1 Měření 1

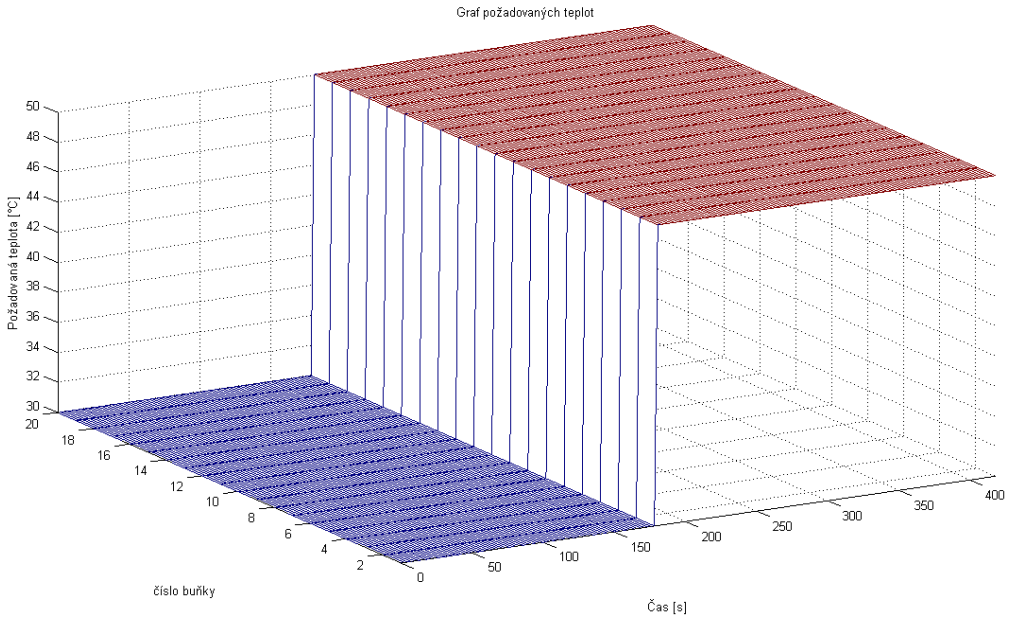
Při měření číslo 1 jsem koeficienty regulátoru nastavil tak, aby výpočet regulace byl prováděn pouze z vlastních hodnot regulační buňky. Hodnoty koeficientů jsou popsány výrazem (4.1) a (4.2). Na obrázku číslo 4.1 je změřena odezva systému na skok z 30 ℃ na 50 ℃ v čase 180 s.

$$b_{xy} = \begin{pmatrix} 0 & 0 & 0 & 60 & 0 & 0 & 0 \\ 0 & 0 & 0 & 52 & 0 & 0 & 0 \\ 0 & 0 & 0 & 36 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.5 & 0 & 0 & 0 \end{pmatrix} \tag{4.1}$$

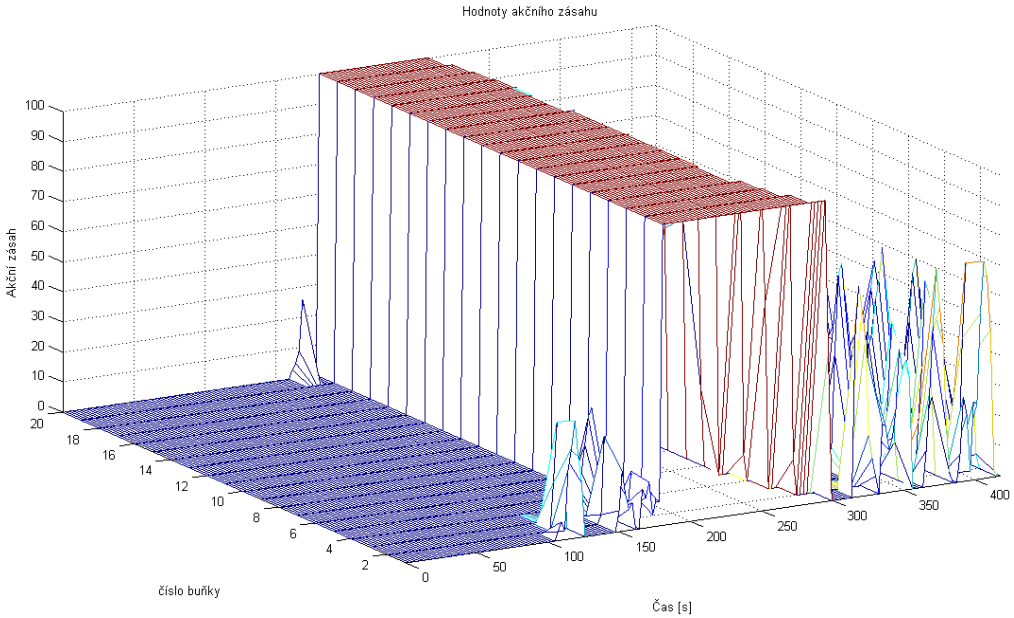
$$a = 15 \tag{4.2}$$



Obrázek 4.1 Naměřené hodnoty teploty při skoku z 30 ℃ na 50 ℃ v měření č.1

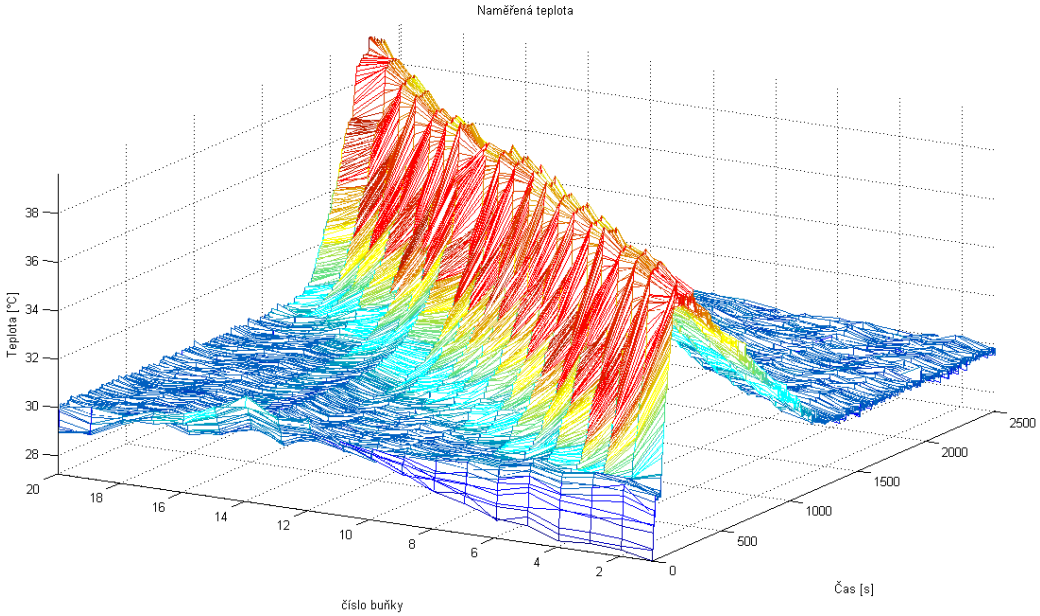


Obrázek 4.2 Požadované hodnoty teploty při skoku z 30 ℃ *na* 50 ℃ v měření č.1

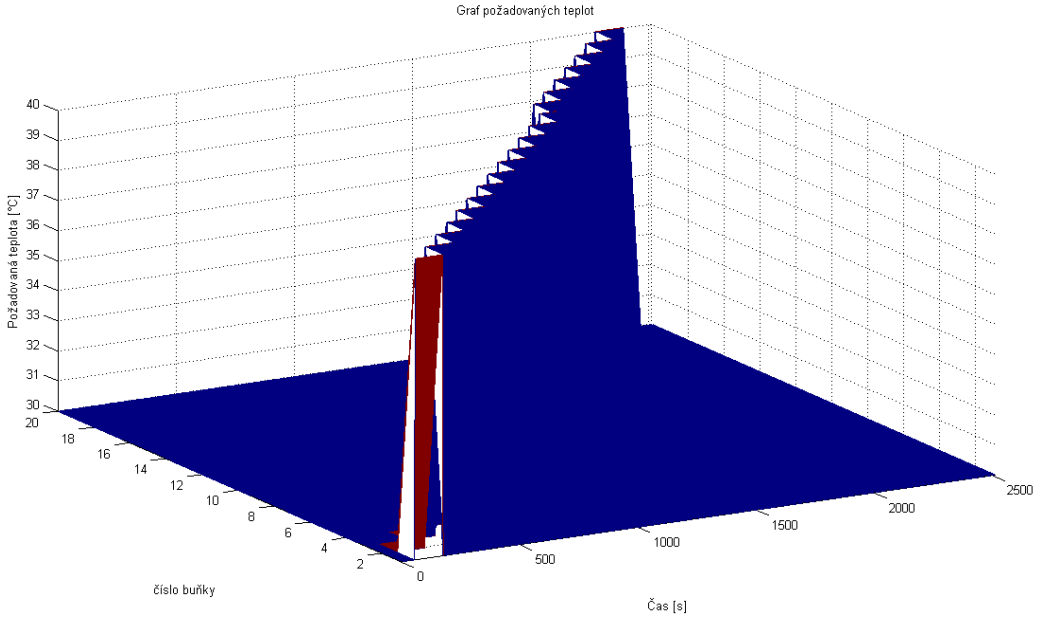


Obrázek 4.3 Hodnoty akčního zásahu při skoku z 30 ℃ *na* 50 ℃ v měření č.1

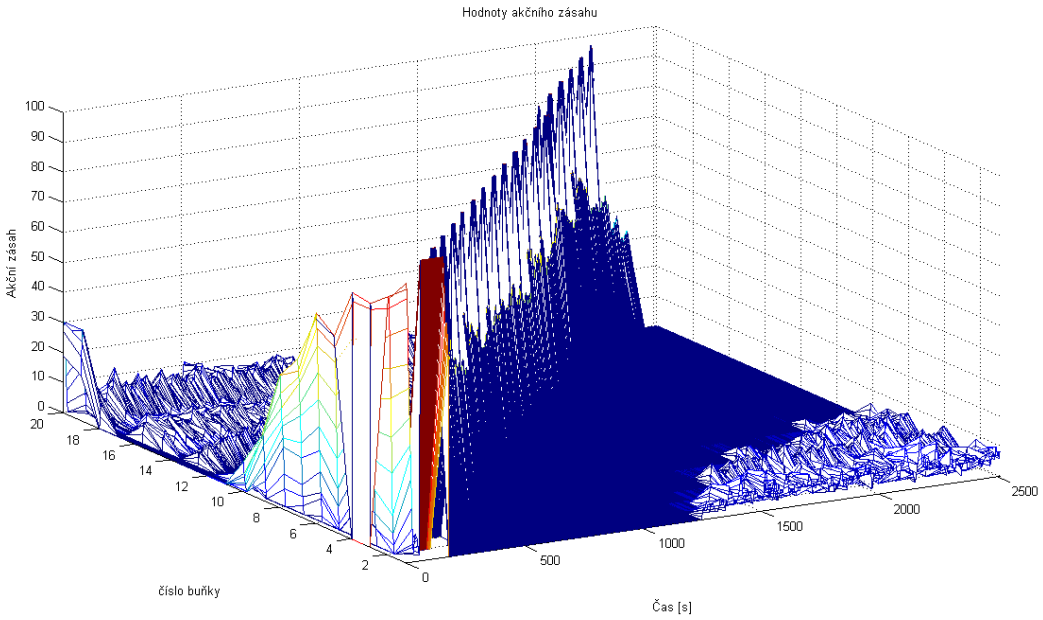
Na obrázku 4.4 je odezva systému na průchod vlnou. Systém je nastaven na teplotu 30 ℃ a amplituda vlny je nastavena 45 ℃. Amplituda je na každé buňce nastavena na 180 sekund, pak se nastaví na další buňce v pořadí.



Obrázek 4.4 Naměřené hodnoty teploty při průchodu teplotní vlnou v měření č.1



Obrázek 4.5 Požadované hodnoty teploty při průchodu teplotní vlnou v měření č.1



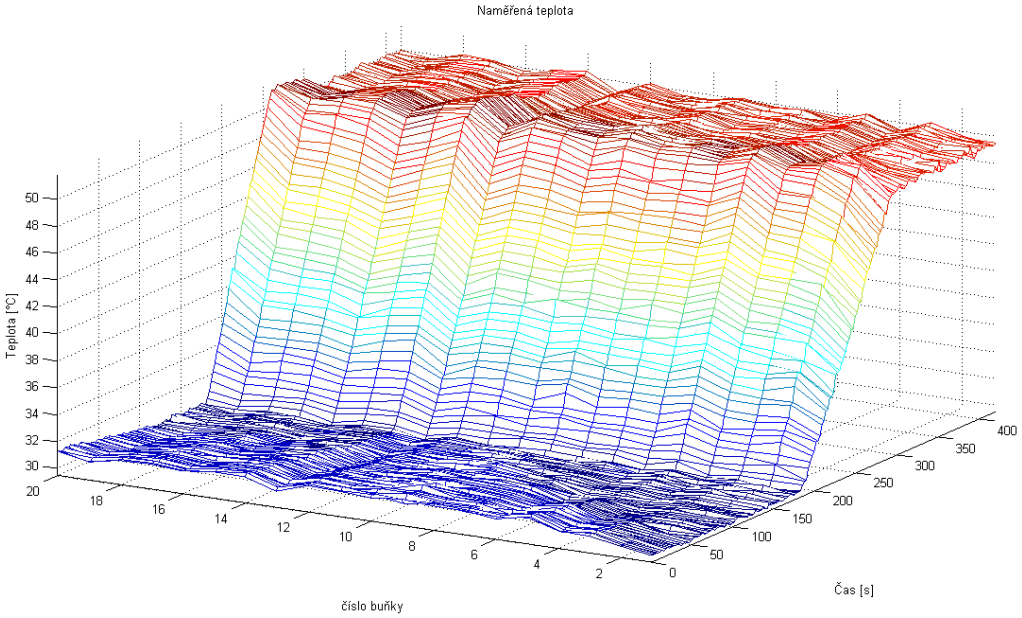
Obrázek 4.6 Hodnoty akčního zásahu při průchodu teplotní vlnou v měření č.1

4.1.2 Měření 2

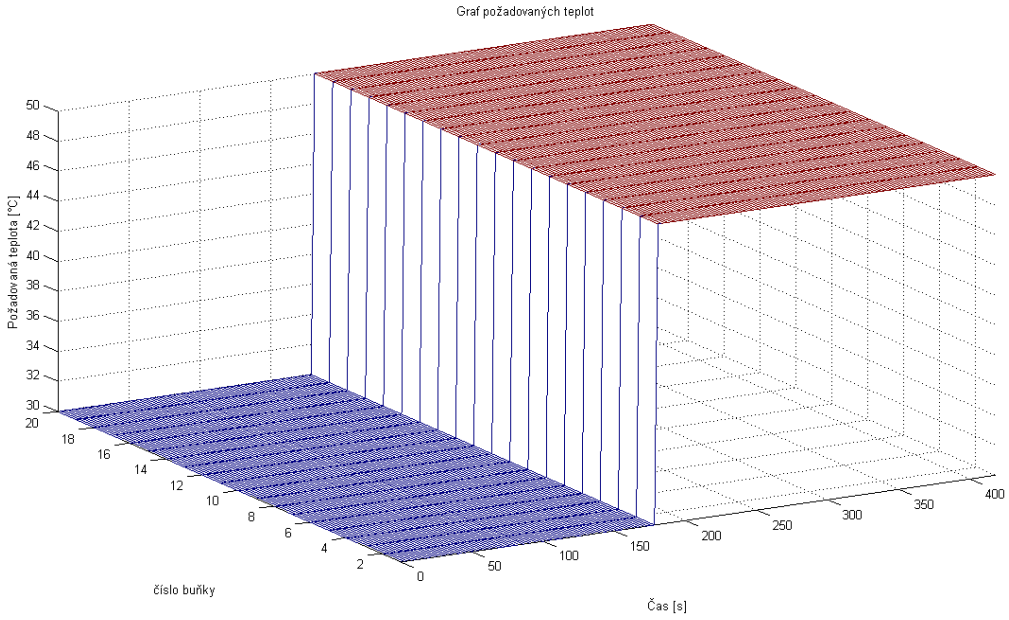
Při měření číslo 2 jsem koeficienty regulátorů nastavil tak, aby regulátor uvažoval i změřené teploty ostatních regulačních buněk. Hodnoty koeficientů jsou popsány výrazem (4.3) a (4.4). Na obrázku číslo 4.7 je změřena odezva systému na skok z 30 ℃ na 50 ℃ v čase 180 s.

$$b_{xy} = \begin{pmatrix} 0 & 10 & 24 & 60 & 24 & 10 & 0 \\ 0 & 0{,}5 & 9 & 52 & 9 & 0{,}5 & 0 \\ 0 & 0{,}1 & 5 & 36 & 5 & 0{,}1 & 0 \\ 0 & 0 & 0{,}5 & 20 & 0{,}5 & 0 & 0 \\ 0 & 0 & 0{,}1 & 10 & 0{,}1 & 0 & 0 \\ 0 & 0 & 0{,}05 & 5 & 0{,}05 & 0 & 0 \\ 0 & 0 & 0{,}01 & 2.5 & 0{,}01 & 0 & 0 \end{pmatrix} \tag{4.3}$$

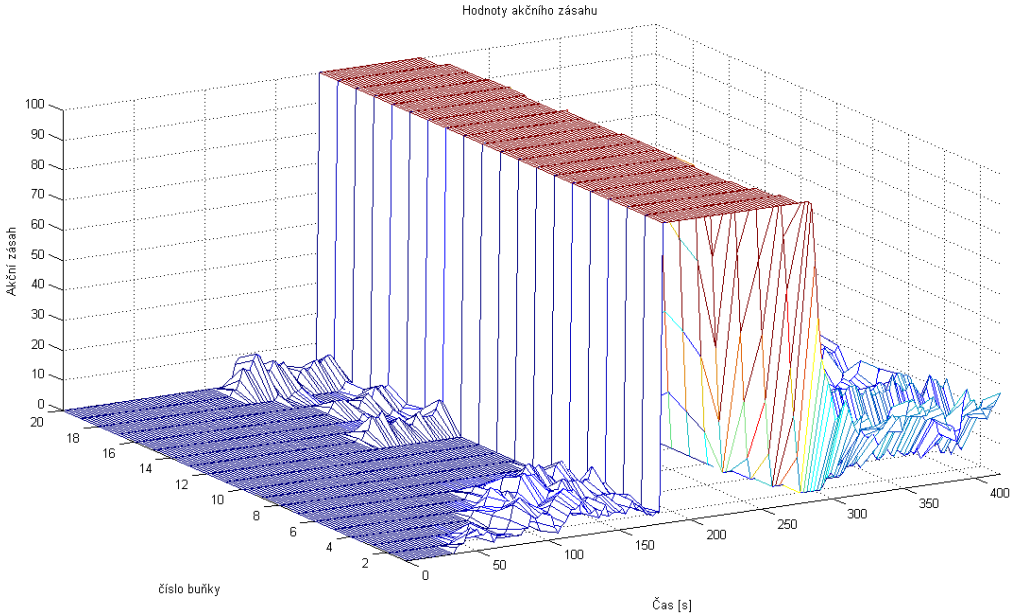
$$a = 15 \tag{4.4}$$



Obrázek 4.7 Naměřené hodnoty teploty při skoku z 30 ℃ *na* 50 ℃ v měření č.2

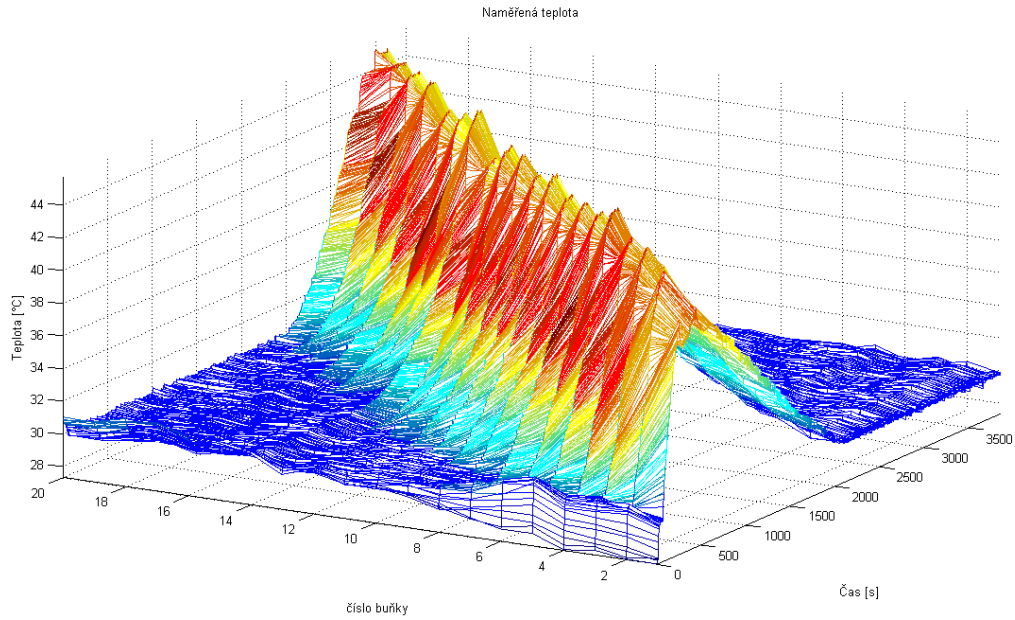


Obrázek 4.8 Požadované hodnoty teploty při skoku z 30 ℃ *na* 50 ℃ v měření č.2

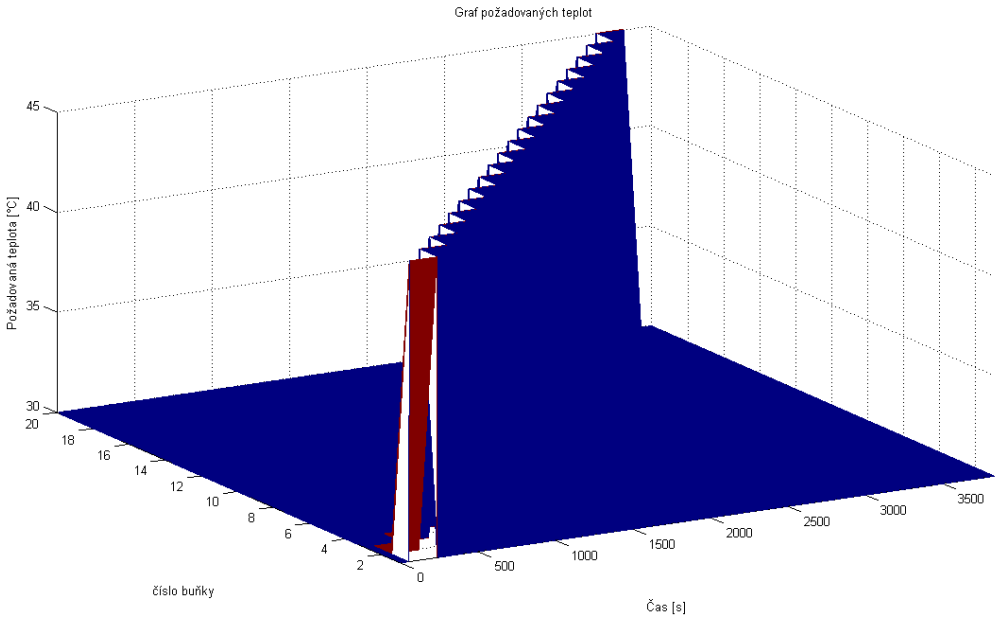


Obrázek 4.9 Hodnoty akčního zásahu při skoku z 30 ℃ *na* 50 ℃ v měření č.2

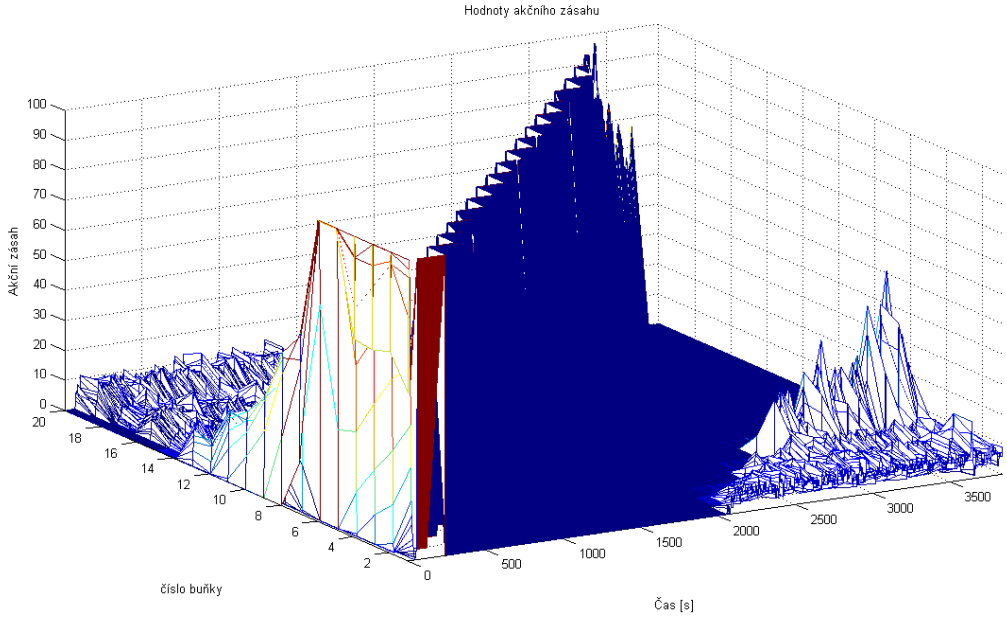
Na obrázku 4.10 je odezva systému na průchod vlnou. Systém je nastaven na teplotu 30 ℃ a amplituda vlny je nastavena na 40 ℃. Amplituda je na každé buňce nastavena na 120 sekund, pak se nastaví na další buňce v pořadí.



Obrázek 4.10 Naměřené hodnoty teploty při průchodu teplotní vlnou v měření č.2



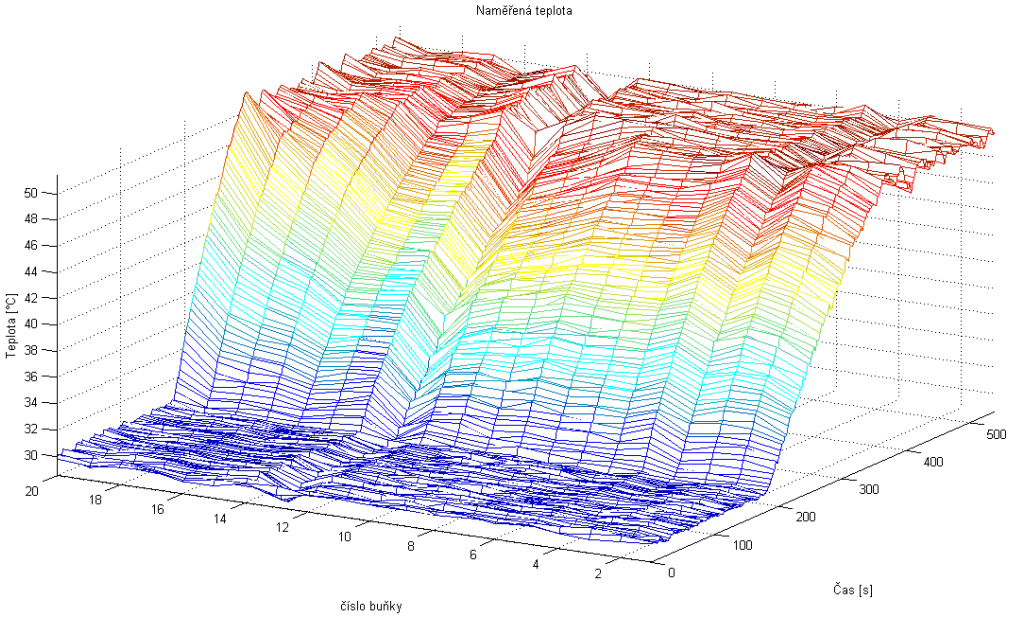
Obrázek 4.11 Požadované hodnoty teploty při průchodu teplotní vlnou v měření č.2



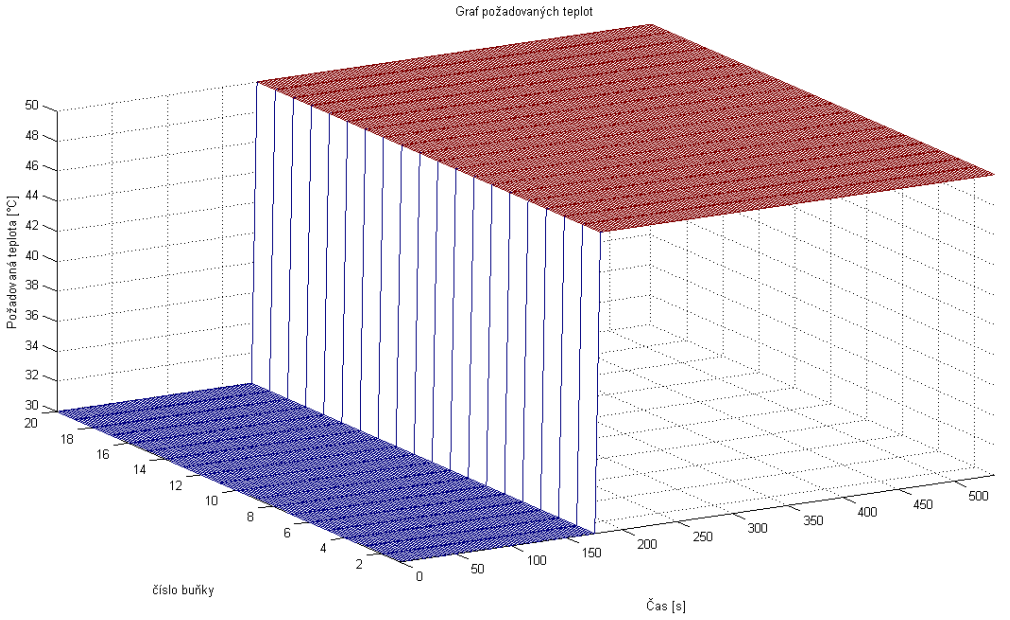
Obrázek 4.12 Hodnoty akčního zásahu při průchodu teplotní vlnou v měření č.2

4.1.3 Měření 3

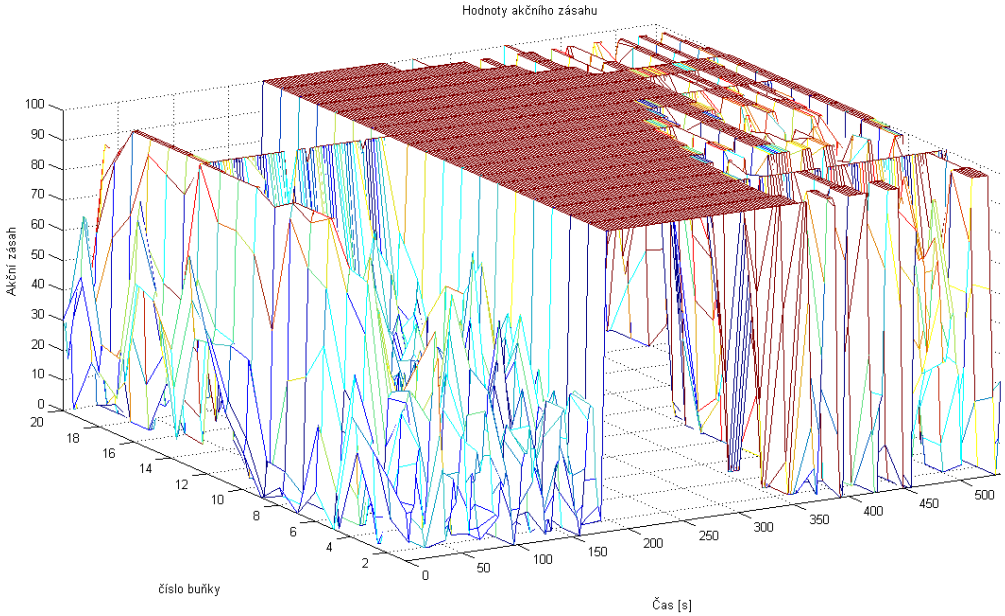
Při tomto měření byl systém zatížen rušením. Toto rušení bylo způsobeno ventilátorem, který byl ve vzdálenosti přibližně 1 m od tyče. Hodnoty koeficientů jsou popsány výrazem (4.3) a (4.4). Na obrázku číslo 4.13 je změřena odezva systému na skok z 30 ℃ na 50 ℃ v čase 180 s.



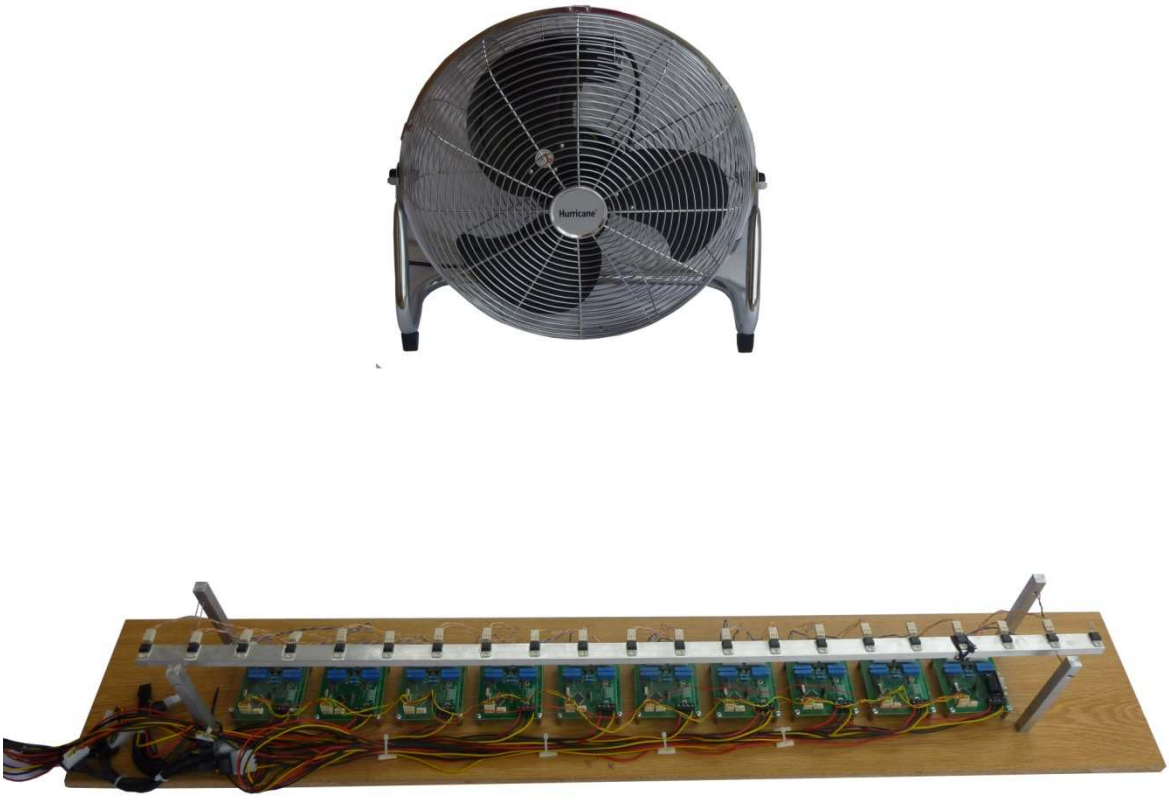
Obrázek 4.13 Naměřené hodnoty teploty při skoku z 30 ℃ *na* 50 ℃ v měření č. 3



Obrázek 4.14 Požadované hodnoty teploty při skoku z 30 ℃ *na* 50 ℃ v měření č. 3



Obrázek 4.15 Naměřené hodnoty teploty při skoku z 30 °C *na* 50 °C v měření č. 3



Obrázek 4.16 Snímek měření zatíženého rušením

5 Závěr

Na začátku práce se bylo třeba rozhodnout zda navázat na předchozí práci Jana Kříže, který se snažil implementovat komunikaci jako lineární Daisy Chain a nebo začít od začátku. Nevýhodou navázání na práci Jana Kříže je nutnost nastudovat cizí zdrojový kód, ovšem na druhou stranu můžu využít jeho zkušeností a nedopustím se chyb, které už byly mým předchůdcem vyřešeny. Z tohoto důvodu jsem se rozhodl vyjít z jeho zdrojového kódu.

Realizace práce byla dosti zpomalena chybami hardwarové části. Při těchto chybách se systém choval nestandartně, a protože jsem systém považoval po hardwarové stránce za funkční, hledal jsem chyby ve vlastním softwaru. Po delší době kdy jsem udělal více testů jsem zjistil, že chyby jsou v hardwaru. Celkem jsem našel nějakou hardwarovou chybu v šesti z jedenácti modulů. Pomocí krokování softwaru, spouštěním jednotlivých částí programu apod. jsem téměř všechny chyby lokalizoval. Nejčastější chybou bylo, že program uváznul, při čekání na přerušení od měření teploty. Příčinou byl většinou špatně vyrobený konektor, v jednom z případů byla dokonce jedna z nožiček snímače přelomena uvnitř konektoru, což zvenčí nebylo znatelné. Na jednom z modulů nefungovalo externí přerušení. Další modul měl zas nefunkční tranzistor. Master modul přijímal data, která na něj z počítače nebyla poslána a to i bez připojeného kabelu sériové linky. U náhradního master modulu nebylo možno odeslat data do počítače. Až na náhradní modul, byly všechny chyby opraveny. Ve většině případů se jednalo o špatně zapájené součástky (studené spoje a nebo cínem spojené sousední piny).

Zařízení je dosti omezeno velikostí programové paměti a datové paměti mikroprocesoru. Kdyby byla programová paměť větší, bylo by možné nastavovat větší množství parametrů, jako například limitní hodnotu akčního zásahu, periodu výpočtu akčního zásahu nebo například uvažovat pro výpočet akčního zásahu i hodnoty akčních zásahů od sousedních regulačních buněk. S větší datovou pamětí by bylo možné zvětšit kruhové buffery pro příjem a odesílání zpráv, což by zvýšilo spolehlivost doručení zpráv. Velké nepřesnosti měření teploty jsem vyřešil větším počtem měření a nalezením mediánu (viz. Kapitola 3.6). Ovšem lepší řešení by bylo při návrhu použít přešnější snímače teploty.

Software pro zařízení je v době odevzdání práce dokončen a vzhledem k zaplnění programové paměti mikroprocesoru nejsou větší rozšíření možné. Software pro řídící počítač by bylo možné rozšířit například o grafické uživatelské rozhraní.

6 Seznam použité literatury

[1] Pavel Doleček. Kouzlo decentralizace řízení technologických procesů. Časopis Automatizace, září 2006

[2] Chris Rapson. Spatially distributed control: Heat conduction in a rod. Master's thesis, České vysoké učení technické v Praze, 2008.

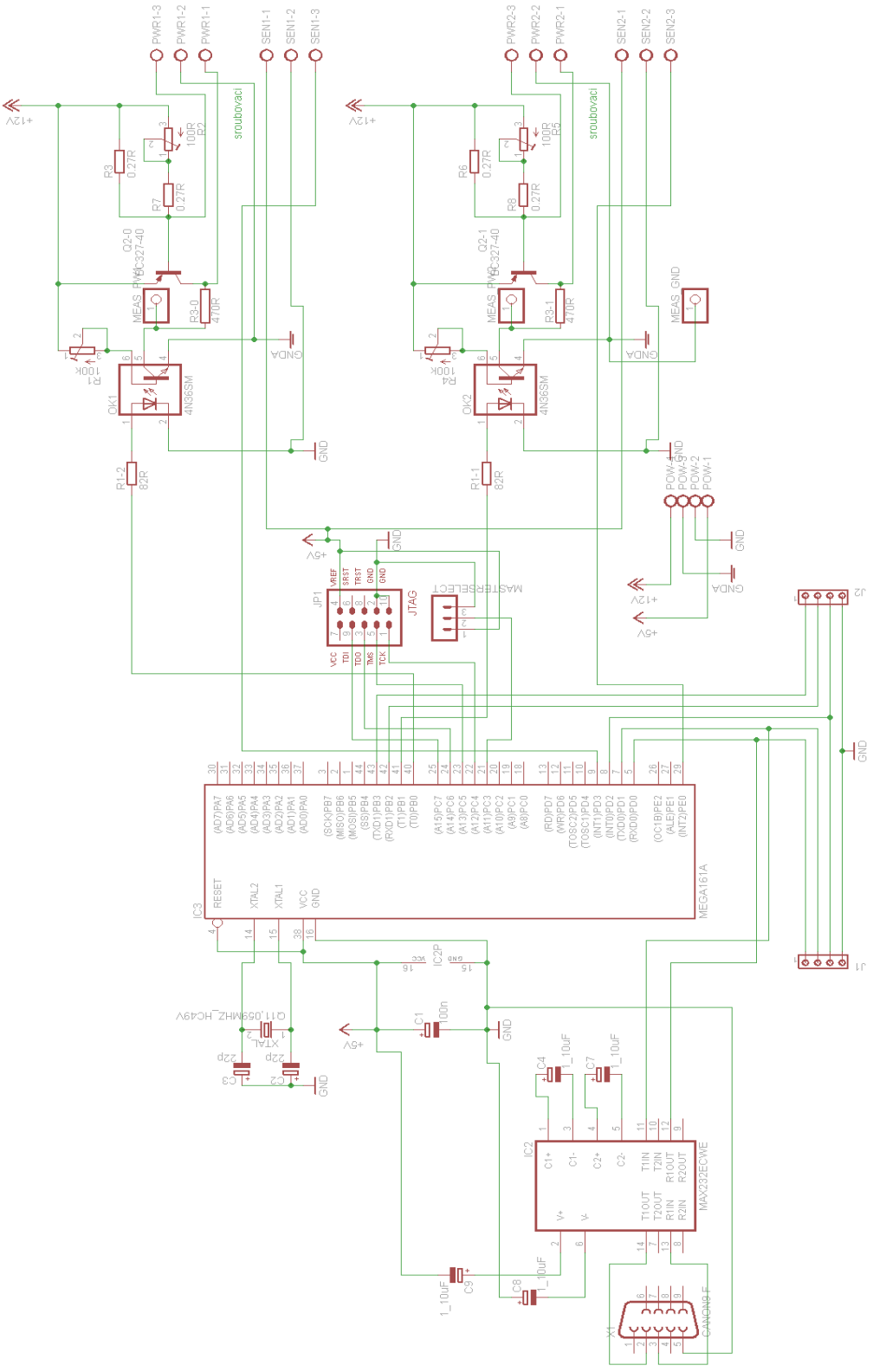
[3] Václav Klemš. Laboratorní model pro výzkum prostorově distribuovaného řízení. Bakalářská práce, České vysoké učení technické v Praze, 2008.

[4] AVR Libc – Online manual
http://www.nongnu.org/avr-libc/user-manual

A Seznam použitých zkratek

USART	Universal Synchronous Asynchronous Receiver Transmitter
LF	Line Feed
EEPROM	Electrically Erasable Programmable Read-Only Memory
ČVUT	České vysoké učení technické
DPS	Deska plošných spojů

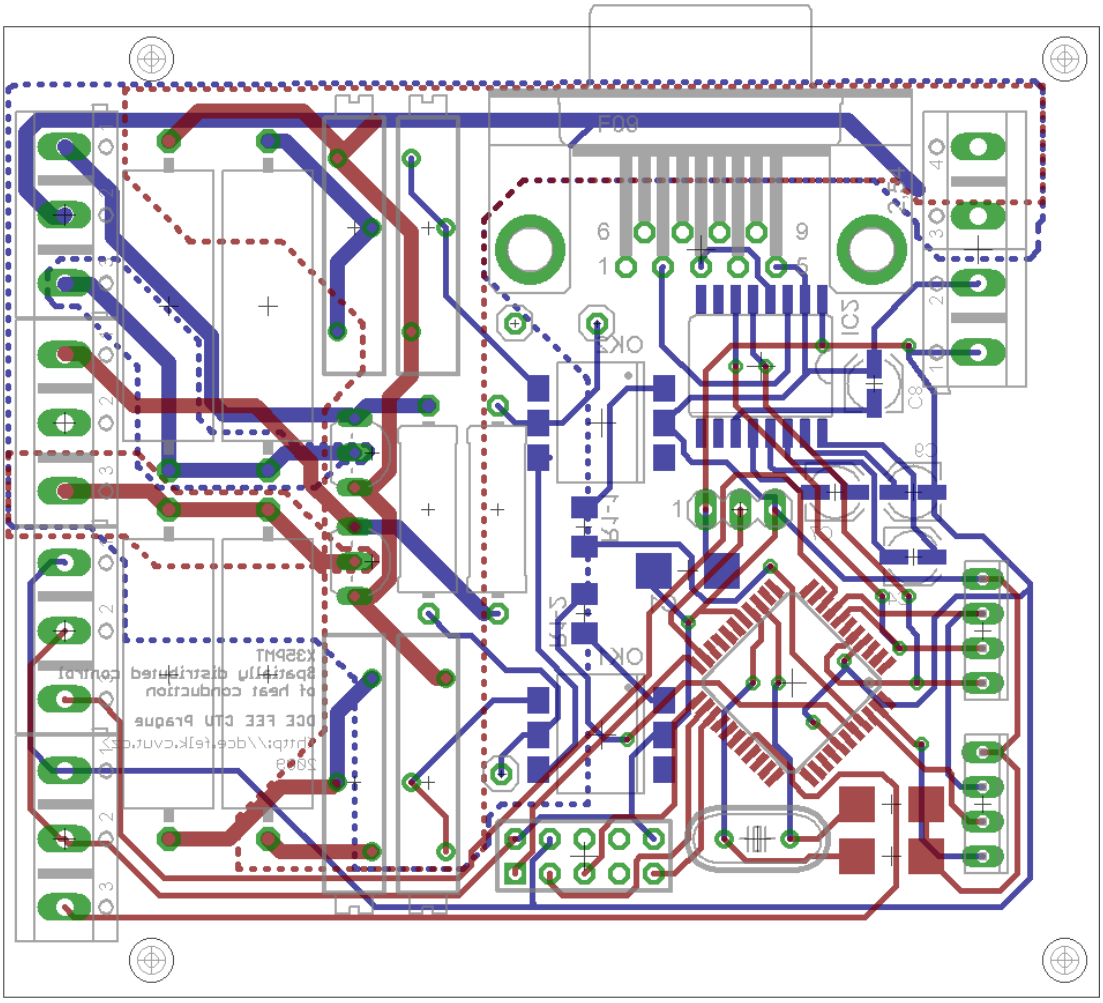
B Schéma modulu s mikrokontrolerem



Obrázek 6.1 Schéma master modulu[1]

[1] Návrh schématu byl vytvořen Janem Křížem

C DPS modulu s mikrokontrolerem



Obrázek 6.2 DPS master modulu[2]

[2] Návrh DPS byl vytvořen Janem Křížem

D Obsah přiloženého CD

Přiložené CD obsahuje následující soubory:

- BP_Petr_Cincibus.pdf
- Adresář: Avr – Soubory se zdrojovými kódy pro mikroprocesor
- Adresář: Matlab – Soubory se zdrojovými kódy pro Řídící počítač
- Adresář: Mereni – Naměřená data