BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2016
http://excel.fit.vutbr.cz

Výpočetní cluster složený z Mikrokontrolérů

Boleslav Šídlo

Abstrakt
Cílem této práce je prozkoumat možnosti využití miniaturního výpočetního clusteru, složeného z jednoduchých mikrokontrolérů, pro paralelní výpočty. Práce zkoumá chování tohoto výpočetního clusteru při řešení různých typů úloh, popisuje jeho možnosti a omezení. Pokusy byly prováděny na výpočetním clusteru tvořeného 4 vyvojovými deskami, které byly osazeny 8 bitovými čipy a komunikovaly přes I2C rozhraní. Výsledkem pokusných měření je srovnání rychlosti výpočtu při použití jednoho mikrokotroléru a při použití clusteru. Experimentálně bylo zjištěno, že v případě aplikací nevyžadující velký objem přenášených dat lze dosáhnout několikanásobného urychlení. Dále se potvrdil předpoklad, že takto jednoduché mikrokotroléry nejsou vhodné pro výpočty s desetinnými čísly vyžadující velkou přesnost.

Klíčová slova: Mikrokotrolér — Cluster — Paralelní výpočet

Přiložené materiály: žádné

*xsidlo02@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Pojmem MCU(microcontroller), nebo také jednočipový počítač, v sobě zahrnuje značné množství typů výpočetní techniky s velkými možnostmi využití a proto není snadné tento pojem jednoduše obsáhnout. Jako obvyklá definice se uvádí, že mikrokotrolér je složen z CPU a periferií, které jsou všechny uloženy na jednom čipu. Mikrokontrolérem může být velice jednoduchý počítačový čip disponující pouze tou nejzákladnější funkcionalitou a používaný na ty nejjednodušší ulohy. Ale také to může být velice výkonný a komplexní prostředek umožňující řízení mimořádně složitých aplikací. Z hlediska dostupných výpočetních zdrojů lze najít velice „skromné“ čipy s operační pamětí v řádu jednotek bytů a minimální funkcionalitou, jejichž cena nepřasahuje $0,5\,\$$ [1], ale také mimořádně výkonné čipy, které disponují značnou výpočetní silou a mnoha rozšiřujícími moduly [2].

Ačkoliv jsou mikrokontroléry značně různorodé, mají společnou jednu vlastnost a tou je efektivita využití zdrojů. Nejedná se naprosto univerzální zařízení, které potenciálně zvládne úplně všechny typu úloh, ale naopak je každý mikrokontrolér specializovaný výpočetní stroj, který je často určen k vykonávání jediné činnosti po celou dobu své životnosti. Tato specializace umožňuje, aby byly mikrokontroléry vysoce efektivní nejen z hlediska rychlosti výpočtu dané úlohy, ale také rozměrů, příkonu a ceny. Vybrání toho nejvhodnějšího mikrokontroléru pro danou aplikaci pak představuje netriviální úlohu a k jejímu řešení je třeba znát důkladně nejen tuto aplikaci, ale také možnosti různých mikrokontrolérů [3][4].

Cílem této práce je prouzkoumat možnosti velice jednoduchých a levných 8 bitových mikrokotrolérů pro použití při paralelních výpočtech, stanovit podmínky, za kterých má takováto paralelizace smysl, upozornit na omezení a nedostatky tohoto řešení a nastínit možnosti dalšího rozvoje tohoto tématu.

2. Výpočetní prostředky

2.1 Popis zvolené platformy

Při hledání vhodné platformy pro tuto práci bylo potřeba vzít v úvahu několik hledisek. Mezi hlavní kritéria patřila cena zařizení, jeho jednoduchost a možnosti jeho komunikačních rozhraní. Velice důležitým hlediskem byla také podpora ze strany výrobce, tedy dostupná dokumentace, vývojový software, implementované knihovny. Účelem této práce nebylo vzít samotný mikrokontrolér a následně k němu konstruovat vývojovou desku a implementovat řadu ovladačů. Účelem bylo vzít jednoduchou, levnou, dostupnou platformu osazenou vhodným čipem a následně prozkoumat její možnosti. Na základě provedeného průzkumu možných kandidátů byla vybrána vývojová deska Arduino Leonardo [5] osazená čipem Atmel ATmega32U4 [6]. Cena jedné desky je cca 250 Kč, jedná se tedy o jednu z nejlevnějších desek na trhu. Ceny výkonějších desek se poté pohybují v relaci 800 Kč a více.

2.2 Komunikační rozhraní zvolené platformy

Vývojová deska Arduino Leonardo poskytuje následující komunikační rozhraní [5]:

- **USB**
  USB rozhraní umožňuje propojení vývojové desky s PC, prostřednictvím USB lze čip programovat a také umožňuje posílat data z vývojové desky a následně je zobrazovat na počítačovém monitoru. USB slouží též k napájení platformy.
- **TWI**
  TWI (Two wire interface) je jednoduchá sběrnice, která umožňuje sériovou komunikaci $Master \leftrightarrow Slave(s)$ v režimu poloviční duplex. Zařizení musí být uspořádány do topologie typu hvězda, se zařízením *Master* jako centrálním prvkem. 7–bitová adresace umožňuje propojení až 128 uzlů [7]. Sběrnice TWI je v podstatě totožná se sběrnicí I2C [8].
- **SPI**
  SPI je sériové plně duplexní komunkační rozhraní, které je primárně určeno pro komunikaci mezi mikrokontrolérem ovládanými periferiemi. Podobně jako u TWI musí být zařízení uspořádána do topologie typu hvězda s jedním zařízením *Master* a několika zařízeními typu *Slave*. Adresace je realizována tak, že pro každé zařízení typu *Slave* vede jeden vodič ze zařízení typu *Master*. Počet adresovatelných zařízení je tak shora omezen počtem programovatelných výstupních digitálních pinů, kterými disponuje zařízení *Master* [9].
- **Programovatelné I/O digitální piny**
  Deska Ardunino Leonardo má k dispozici celkem 14 vstup–výstupních digitálních pinů. Tyto piny lze využít jak ke komunikaci mezi mikrokontroléry, tak ke komunikaci s jinými zařízeními. Piny pracují s hodnotami napětí 0 V a +5 V.
- **Analogové vstupní piny**
  Deska Ardunino Leonardo má k dispozici celkem 6 analogových vstupních pinů. Tyto piny jsou primárně určeny pro sběr dat z analogových senzorů a následné zpracování těchto dat pomocí A–D převodníků. Pro vzájemnou komunikaci mezi mikrokontroléty nejsou příliš vhodné.

3. Výpočetní mini cluster

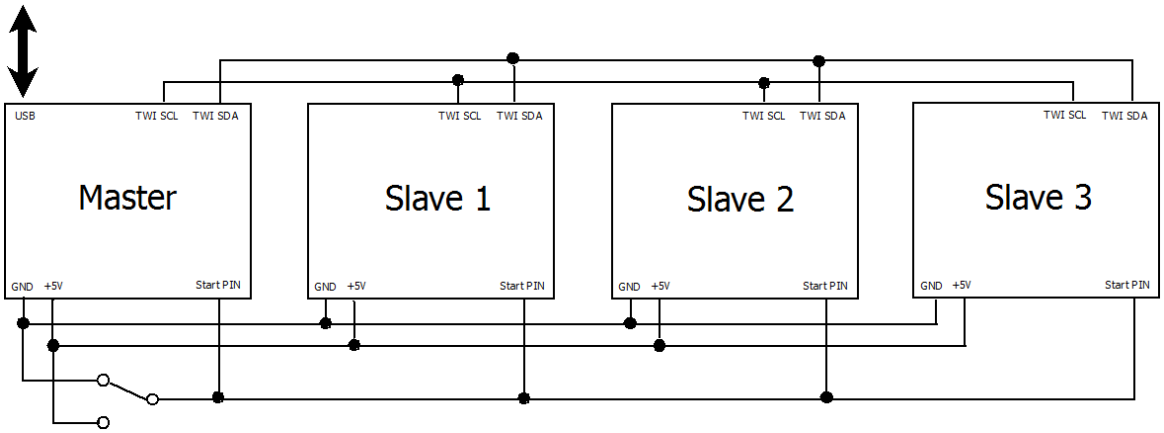
3.1 Možnosti vytvoření výpočetního clusteru

Možností jak konstruovat výpočetní mini cluster z mikrokontrolérů je více. Vzhledem k zaměření celé práce je potřeba si zoodpovědět následující otázky:

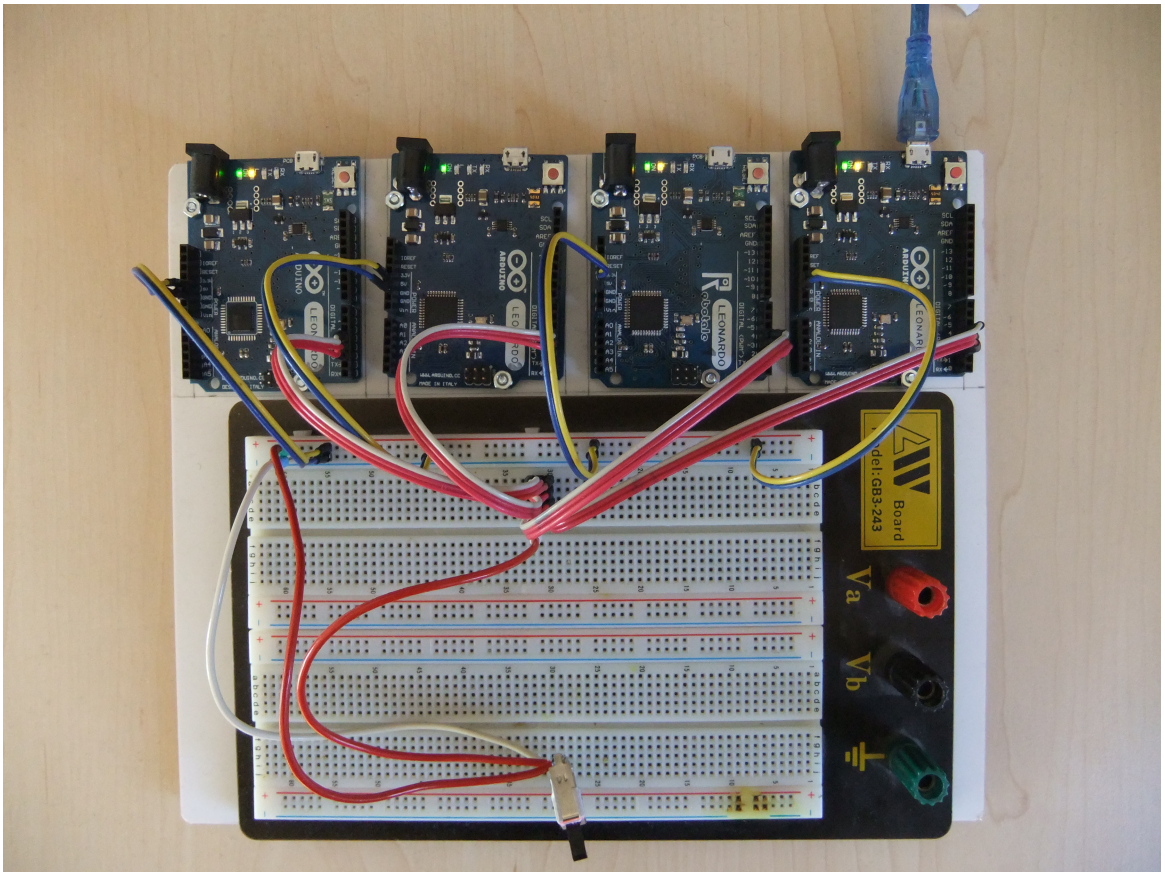
- Z kolika uzlů se bude cluster skládat?
- Jak bude vypadat topologie propojení?
- Jak bude vyřešeno napájení jednotlivých uzlů
- Jaké bude použito komunikační rozhraní mezi uzly?
- Jak bude vypadat vstup–výstupní rozhraní pro práci s daty

3.2 Výsledná architektura clusteru

Po zodpovězení výše zmíněných otázek byla vytvořena tato architektura. Cluster se skládá ze 4 výpočetních uzlů, jeden typu *master* a 3 typu *slave*. Topologie zapojení je typu hvězda. Uzel *master* je nápajen prostřednictvím USB konektoru z PC, uzly *slave* jsou napájeny z uzlu *master*. Celý cluster komunikuje prostřednictví rozhraní TWI, které umožňuje nejjednodušší implmentaci. Dále je na digitalní vstupní pin každého uzlu přiveden signál START, který slouží ke kontrole běhu aplikace. Tento signál je regulován dvoupolohovým přepínačem. Výsledky jsou posílány z uzlu *master* prostřednictvím USB rozhraní do PC, kde jsou zobrazovány prostřednictvím sériové konzole, která je součástí standardní softwarové podpory dodávané výrobcem desky Leonardo [10]. Výsledné schéma zapojení je vidět na obrázku 1. Pro fyzickou realizaci propojení uzlů clusteru bylo použito nepájivé pole, obrázek 2.



Obrázek 1. Schéma mini clusteru tvořeného 4 výpočetními uzly



Obrázek 2. Fotografie prototypu výpočetního mini clusteru

3.3 Způsob komunikace uzlů v clusteru

Jak již bylo zmíněno výše, pro přenos dat mezi uzly bylo zvoleno rozhraní TWI. Hlavní výhodou tohoto tozhraní je jeho jednoduchost, z každého uzlu vedou pouhé dva vodiče. Toto rozhraní ma ovšem i svá omezení. Všechna data vždy musí procházet přes hlavní uzel. Není tedy možné jeho prostřednictvím realizovat komunikaci typu $peer-to-peer$. V jednu chvíli mohou komunikovat právě dvě zařízení, jeden *master* a jeden *slave*. Tato komunikace je polovičně duplexní, v jednu chvíli mohou data téct pouze jedním směrem. Dále není možné, aby komunikaci zahájilo nějaké jiné zařízení než *master*. Zařízení *slave* vždy pouze pasivně čeká až mu *master* pošle data, nebo až ho *master* vyzve, aby data odeslal.

Komunikace TWI má také to omezení, že není primárně určena k přenosu většího objemu dat. U desky Arduino Leonardo má datový buffer pro TWI modul kapacitu 64 bytů, navíc samotná implementace standardní knihovny od Arduina *Wire.h* [11], která slouží pro používání TWI modulu je implementována tak, že v rámci jedné transakce je možné přenést maximálně 32 bytů.

Pro některé aplikace je toto omezení značně limitující, proto byl v rámci této práce implementován komunikační protokol, který umožňuje poslat z jednoho zařízení na druhé až 7650 bytů v rámci jednoho spojení. Vzhledem k tomu, že celková kapacita operační paměti čipu ATmega32U4 je 2,5 KB, je tento objem více než dostatečný [2].

Podstata tohoto komunikačního protokolu spočívá v rozdělení dat do paketů o maximální délce 32 bytů, které jsou posílány ve sledu na sebe navazujících TWI transakcí. Pakety jsou dvojího druhu:

Inicializační pakety Inicializační paket je v rámci přenosu vždy jeden a má 5 bytů, přičemž první byte má hodnotu 0 a zbylé 4 byty reprezentují 32–bitové celé číslo. Toto číslo představuje délku celého objemu dat (v bytech), který bude v rámci této komunikace přeposlán. **Datové pakety** Datových paketů může být v rámci přenosu 1-255. Každý z nich obsahuje 2 bytovou hlavičku a množství přenášených dat je 1-30 bytů. První byte hlavičky představuje číslo paketu, druhý byte hlavičky představuje délku dat (v bytech) tohoto paketu.

Tento jednoduchý komunikační protokol neobsahuje žádné navazování spojení, potvrzování přijetí ani kontrolní součty. Avšak je velice efektivní z hlediska naprosto minimální režie.

4. Testovací aplikace

4.1 Kategorie testovacích aplikací

Při volbě testovacích aplikací bylo potřeba vzít v úvahu možnosti jejich paralelizace a dále tyto aplikace rozdělit do vhodných kategorií. Pro zjednodušení byly uvažovány dva parametry aplikací. Prvním parametrem je nejčastější typ zpracovávaných dat, tedy jestli se jedná o celá nebo desetinná čísla. Druhým parametrem je množství přenášených dat, tedy jestli je objem přenášených dat srolvnatelně velký s celkovým objemem dat zpracovávaných.

4.2 Zvolené testovací aplikace

Součet prvků v poli (data uložena v každém uzlu)
Tato aplikace spadá do kategorie celá čísla–malý objem přenášených dat. Algoritmus spočívá v tom, že každý uzel má před během programu ve své paměti uloženo pole hodnot typu $uint8\_t$. Každý uzel *slave* hodnoty tohoto pole sečte a výsledek pošle uzlu *master*, který všechny čtyři dílčí součty (včetně toho svého) zpracuje. Měří se doba od začátku sčítání na uzlu *master* do chvíle, kdy je k dispozici konečný součet.

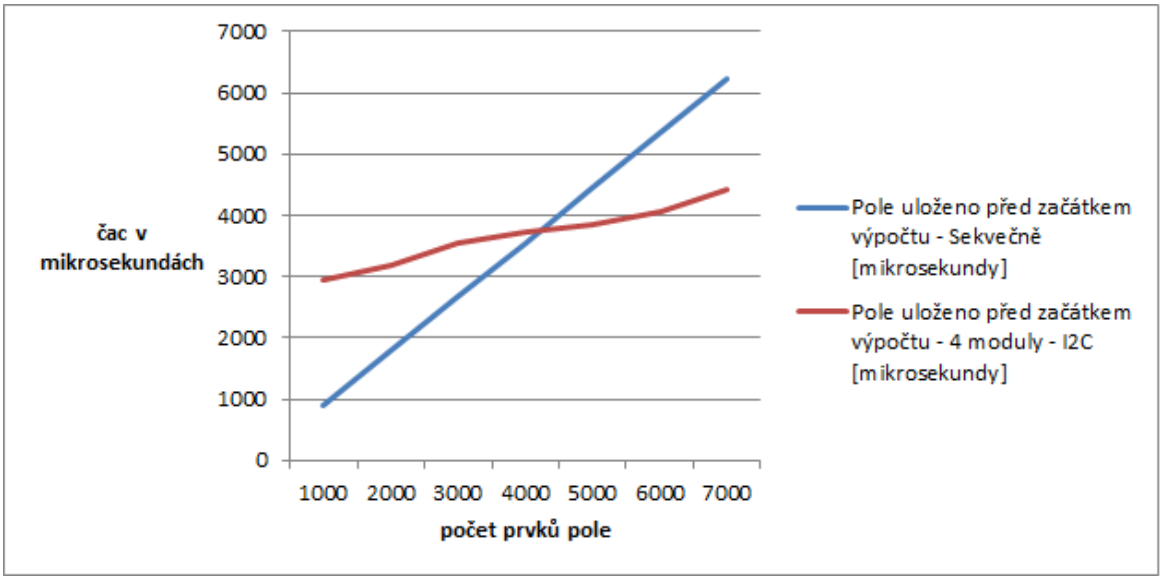
Součet prvků v poli(data distribuována z hlavního uzlu)
Tato aplikace spadá do kategorie celá čísla–velký objem přenášených dat. Výpočet je obdobný jako u aplikace předešlé s tím rozdílem, že pole je posíláno z hlavního uzlu každému zařízení *slave*. Dochází tak ke třem přenosům pole směrem od hlavního uzlu a následně ke třem přenosům dílčího součtu směrem do hlavního uzlu. Měří se doba výpočtu včetně času potřebného pro distribuce pole z hlavního uzlu.

Paralelní výpočet čísla π
Tato aplikace spadá do kategorie desetinná čísla–malý objem přenášených dat. Aplikace počítá číslo π za použití Bailey–Borwein–Plouffe formule [12].

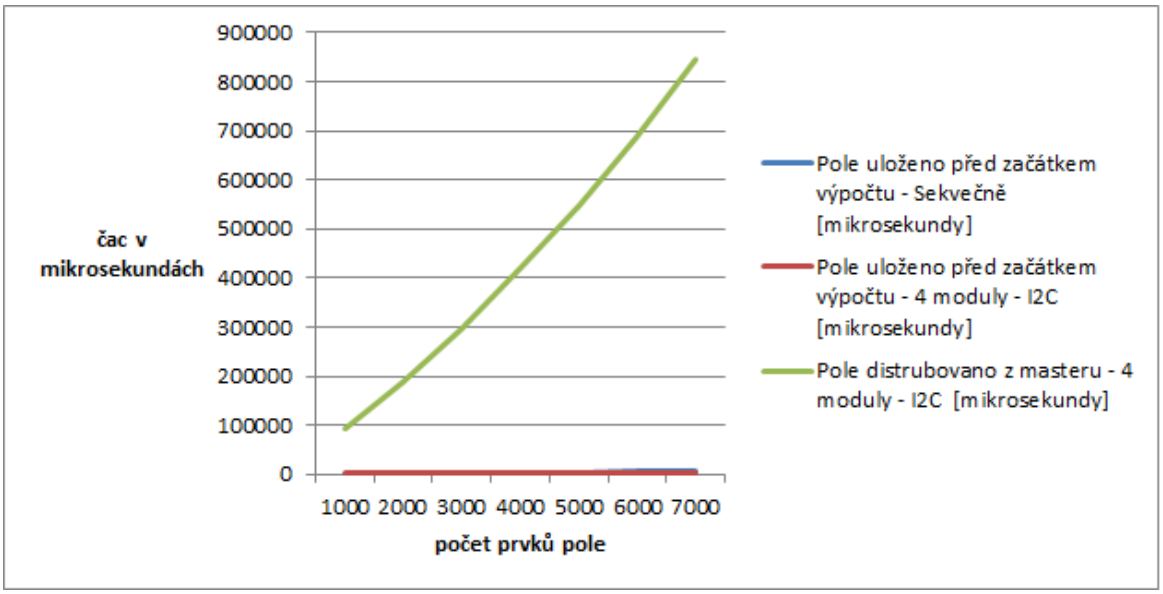
$$\pi = \sum_{k=0}^{\infty}\left[\frac{1}{16^k}\left(\frac{120k^2+151k+47}{512k^4+1024k^3+712k^2+194k+15}\right)\right]$$

Tabulka 1. Naměřené hodnoty delky výpočtu v závislosti na objemu zpracovávaných dat

Medián délky výpočtu v mikrosekundách							
Počet prvků pole	1000	2000	3000	4000	5000	6000	7000
Čas sekv. řešení	896	1788	2678	3560	4452	5350	6240
Čas paralel. řešení (pole uloženo)	2952	3178	3554	3730	3842	4062	4408
Čas paralel. řešení (pole distribuováno)	91608	188402	296298	418462	547992	687996	844200



Obrázek 3. Porovnání délky sekvenčního a paralelního výpočtu (pole uloženo v každém uzlu) v závislosti na množství zpracovávaných dat



Obrázek 4. Porovnání délky sekvenčního a paralelního výpočtu (pole distribuováno z hlavního uzlu) v závislosti na množství zpracovávaných dat

5. Výsledky experimentů

Metoda testovacích experimentů

Při testovacích experimentech se měřila doba výpočtu sekvenčního případně paralelního řešení nad různě velkými objemy dat. Pro každou kombinaci bylo provedeno 10 měření, z nichž byl vybrán medián k dalšímu zpracování. naměřené hodnoty jsou uspořádány v tabulce 1 a zobrazeny v grafech 3 a 4 na předcházející stránce.

5.1 Součet prvků v poli

Při sekvečním provádění výpočtu má součet prvků v poli podle očekávání lineární časovou složitost.

V případě paralelizace, kdy je část pole uložena v každém uzlu již před začátkem výpočtu, je časový nárůst pomalejší a od určitého objemu dat dochází, v porovnání se sekvečním řešením, k významnému urychlení.

V případě paralelizace, kdy je pole před začátkem výpočtu distribuváno z hlavního uzlu, nejenže dochází k několikanásobnému zpomalení oproti sekvenčnímu řešení, ale časová složitost zde vykazuje polynomální charakter. Pakliže bychom naměřené hodnoty času výpočtu aproximovali polynomem 2. stupně, dospěli bychom (po zaokrouhlení) ke vztahu:

$$y = 0,006x^2 + 80x + 5517$$

kde y je doba výpočtu v mikrosekundách a x je počet prvků v poli. Uvedený vztah byl odvozen pomocí nástroje textitRegresion Tool [13].

5.2 Paralelní výpočet čísla π

V případě této aplikace bylo provedeno testování pouze sekvenčního výpočtu. Povrdilo se, že 8–bitový mikrokontrolér skutečně není vhodným nástrojem pro velice přesné počítání s desetinnými čísly. Mikrokontrolér byl schopen provést pouze 8 iterací, poté již jeho přesnost reprezentace dat nestačila. Číslo π bylo spočítáno s přesností pouze na 6 desetinných míst, což není příliš přesný výsledek. V této oblasti probíhají další experimenty s možnostmi reprezentace dat, který by umožnily přesnější výpočet.

6. Závěr

Na základě provedených experimentů bylo zjištěno, že v případě paralelizace výpočtu prostřednictvím clustru, složeného z jednoduchých mikrokontrolérů, lze dosáhnou určitého urychlení, ale za cenu značných omezení. Aplikace musí primárně pracovat s celými čísly a objem dat přenášený mezi jednotlivými uzly musí být naprosto minimální. Ukázalo se že komunikační rozhraní TWI není vhodné pro přenos velkého objemu dat a představuje zásadní omezení celého výpočtu.

Tuto problematiku je možno dále rozvinout dalšími experimenty, například testování clusterů s více výpočetními uzly, nebo použití jiných komunikačních rozhraní. Také by bylo možné různá komunikační rozhraní navzájem kombinovat, případně využít složitějšího mikrokontroléru s více rozhraními jednoho typu a vytvořit tak mnohem komplexněší topologii výpočeního clusteru.

Dále by bylo vhodné prověřit vlastnosti tohoto clusteru v reálné aplikaci. Jako vhodný kadidát se nabízí sběr a zpracování dat v senzorových sítích.

Poděkování

Rád bych velice poděkoval vedoucímu své bakalářské práce, kterým byl Ing. Michal Bidlo Ph.D., za věnovaný čas, cenné rady a inspirující podněty.

Literatura

[1] Microchip pic10f200 introduction. http://www.microchip.com/wwwproducts/en/PIC10F200. Accessed: 2016-04-02.

[2] Atmel at32uc3c0512c. http://www.atmel.com/devices/AT32UC3C0512C.aspx?tab=parameters. Accessed: 2016-04-02.

[3] Tammy Noergaard. *Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers (Embedded Technology)*. Newnes, 2005.

[4] John Catsoulis. *Designing Embedded Hardware*. O'Reilly Media, 2005.

[5] Arduino leonardo. https://www.arduino.cc/en/Main/ArduinoBoardLeonardo. Accessed: 2016-04-02.

[6] Atmel atmega32u4. http://www.atmel.com/devices/ATMEGA32U4.aspx. Accessed: 2016-04-02.

[7] Atmel avr libc reference manual. http://www.atmel.com/webdoc/AVRLibcReferenceManual/group__twi__demo_1twi_demo_intro.html. Accessed: 2016-04-02.

[8] Philips Semiconductors. *I2C MANUAL*, 3 2003.

[9] Atmel atmel-ice physical interfaces. http://www.atmel.com/webdoc/atmelice/atmelice.using_ocd_physical_spi.html. Accessed: 2016-04-02.

[10] Arduino ide software. https://www.arduino.cc/en/Main/Software. Accessed: 2016-04-02.

[11] Arduino wire library. https://www.arduino.cc/en/Reference/Wire. Accessed: 2016-04-02.

[12] MathWorld bbp formula. http://mathworld.wolfram.com/BBPFormula.html. Accessed: 2016-04-02.

[13] Xuru online polynomial regression. http://www.xuru.org/rt/pr.asp. Accessed: 2016-04-02.