České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídící techniky

BAKALÁŘSKÁ PRÁCE

Praha, 2007 **Michal Padyšák**

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídící techniky

BAKALÁŘSKÁ PRÁCE

Řídící systémy pro UAV prostředky

Praha, 2007

student: Michal Padyšák
vedoucí práce: Ing. Martin Hromčík, PhD.

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 _____ _____

podpis

České vysoké učení technické v Praze, fakulta elektrotechnická

Katedra řídicí techniky

Školní rok: 2006/2007

Zadání bakalářské práce

Student: Michal Padyšák

Obor: Kybernetika a měření

Název tématu: Řídicí systémy pro UAV prostředky

Zásady pro vypracování:

1. Prostudujte prezentační prostředí projektu malého UAV vrtulníku na adrese http://rtime.felk.cvut.cz/helicopter (vede doc. Hanzálek a Ing. Špinka).
2. Navrhněte změny a úpravy výše zmíněného webu, s důrazem na jeho srozumitelnost a funkčnost coby platformy pro sdílení letových dat a modelů pro návrh řídicích systémů.
3. Navrhněte na základě poskytnutých dat vhodný regulátor (např. stabilizátor stranových kmitů) a ověřte simulací, případně během letového testu.
4. Vyberte vhodný mikroprocesor pro centrální řídicí jednotku stabilizované základny pro UAV letoun z řady ARM procesorů firmy Atmel.
5. Zprovozněte příslušný vývojový kit a naprogramujte vybrané funkce (např. komunikace se senzory, seriový kanál apod.)

Poznámky:
Práce budou probíhat v rámci společného projektu K335, K333 a VTUL zaměřeného na vývoj stabilizované základny pro bezpilotní prostředky (Dr. Hurák, body 4 a 5) a projektu katedry směřujícího k vývoji lehkého UAV vrtulníku (doc. Hanzálek a Ing. Špinka, body 1, 2, 3).

Seznam odborné literatury: Dodá vedoucí práce

Vedoucí bakalářské práce: Ing. Martin Hromčík, Ph.D.

Datum zadání bakalářské práce: zimní semestr 2006/07

Termín odevzdání bakalářské práce: 15. 8. 2007

Prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

Prof. Ing. Zbyněk Škvor, CSc.
děkan

V Praze, dne 6. 3. 2007

Abstrakt

Táto bakalářská práce se skládá ze dvou částí. První část je zaměřena na projekt RAMA Katedry řídící techniky Fakulty elektrotechnické Českého vysokého učení technického v Praze, kterého účelem je vývoj a konstrukce univerzálního a kompaktního řídícího systému pro malá UAV. Součástí práce je hodnocení webových stránek projektu a návrh regulátorů pro nejnižší hladinu řídícího systému UAV helikoptéry. Druhá část této bakalářské práce je zaměřena na společný projekt Katedry řídící techniky FEL ČVUT a VTUL Praha zaměřeného na vývoj stabilizované základny pro malá UAV. Jsou srovnány vlastnosti vhodných mikrokontrolérů ATMEL pro řídící systém stabilizované základny. Na konec této práce byl napsán program pro testování doby výpočtu algoritmů pro mikrokontrolér AT91SAM7X256

Abstract

This bachelor thesis is divided into two parts. The first part is focused on the project RAMA of Department of Control Engineering, Faculty of Electrical Engineering at Czech Technical University in Prague, which purpose is to design and build a universal lightweight and compact control system for small UAV. Component of thesis is evaluation of web page of project and design controller for first layer control system of UAV helicopter. The second part of this bachelor thesis is focused on the concerted project of DCE FEE at CTU and VTUL Prague, which purpose is to development on stabilized platform for small UAV devices. There are compared properties of available microcontrollers units ATMEL for control system for stabilized platform. At the end of thesis a program project in C were designed for purposes of testing run time of algorithms for microcontroller AT91SAM7X256.

Obsah

Kapitola 1

Projekt RAMA

Úvod

Projekt RAMA (*Remotely operated Aerial Model Autopilot*) Katedry řídící techniky ČVUT v Praze si klade za cíl vývoj a konstrukci kompaktního řídícího systému pro lehká UAV. V první kapitole se zaměříme na hodnocení prezentačního prostředí webových stránek projektu a provedeme jejich hodnocení z hlediska srozumitelnosti a dostupnosti měřených dat. Je nutno poznamenat, že stránky projektu jsou teprve ve stavbě a množství informací zde ještě není uvedeno, zaměříme se proto hodnocení stávajícího prostředí. V další části se pomocí informací a dat sdílených na stránkach pokusíme navrhnout jednoduchý regulátor. Stránky jsou psány v jazyce anglickém, proto budeme v případě návrhu regulátoru uvádět nekteré parametry (roll rate, pitch rate...) v původním znění. Výsledky následně ověříme simulací.

1.1 Prezentační prostředí projetku RAMA

1.1.1 Webové stránky projektu RAMA

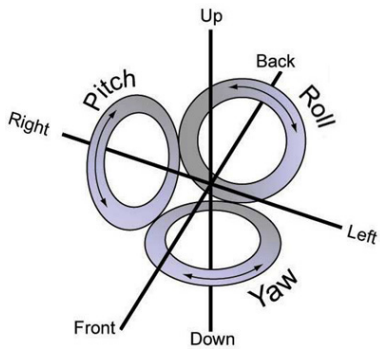
Účelem webových stránek projektu na adrese *www.rtime.felk.cvut.cz/helicopter* je poskytnout zájemcům o problematiku řízení UAV dostatečné informace o projektu, modelu helikoptéry a sdílená naměřená data. Po prozkoumání prezentačního prostředí nebyli nalezeny zásadní chyby nebo nedostatky v přístupu k naměřeným datům, metodám zpracování nebo návrhu regulátorů nízké úrovně. Informace týkající se pokročilých modelů a metod řízení nebyli hodnoceny. Nalezené nedostatky jsou spíše technického charakteru.

Doporučení:

- Doplnit stránky o seznam použitých zkratek
- V reportech pro zvětšení srozumitelnosti věnovat více prostoru popisu postupu řešení
- Grafy publikovat ve větším rozlíšení
- Na reportech uvádet informace pro dohledatelnost jejich původu

V další části uvedeme způsob stabilizace helikoptéry a navrhneme jednoduchý regulátor.

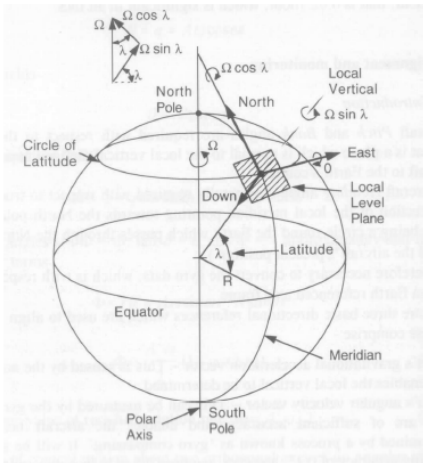
Na obr.1.1 je vidět možné způsoby náklonů a jejich pojmenování. Pro stabilizaci polohy helikoptéry jsou důležité *pitch* reprezentující předozadní náklon a *roll* reprezentující boční náklon. Zjednodušeně lze řict, že tyto náklony mají za následek pohyb helikoptéry v souřadnicovém systému WGS84 (vpřed-vzat, vpravo-vlevo). K dispozici máme velikosti ůhlových rychlostí daných náklonů. Stabilizace na místě pak spočívá v udržení jejich nulové hodnoty akčním zásahem do řízení helikoptéry (lat,lon). Příslušné přenosové funkce nalezneme na webových stránkách ve tvaru 1.3 a 1.4



Obrázek 1.1: Možnosti náklonu UAV

$$G_{lon\rightarrow q}(s) = \frac{1107}{s^2 + 18,42s + 287,5} \tag{1.1}$$

$$G_{lat\rightarrow p}(s) = \frac{327,4}{s^2 + 9,061s + 130,9} \tag{1.2}$$



Obrázek 1.2: Systém zápisu polohy WGS84

Na obr.1.2 můžeme vidět systém WGS84, využívající se pro zápis polohy v přirozeném tvaru jako šířka, délka a výška.

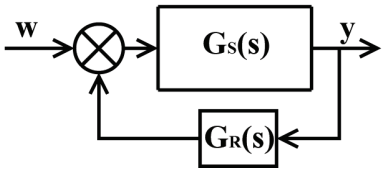
1.2 Metody návrhu regulátorů

1.2.1 Metoda Ziegler-Nichols

Metoda seřízení regulátorů Ziegler-Nichols (ZN) byla původně praktická a ryze empirická metoda pro nalezení parametrů svolených regulátorů. Nazývá se taky metodou seřízení podle kritického zesílení. Nosnou myšlenkou metody je přivést obvod na hranici stability změnou zesílení ve ZV a následně Postup pro seřizení regulátoru můžeme popsat postupem:

- Vyřadíme integrační a derivační složku regulátoru ($T_i \to \infty$ a $T_d \to 0$). Podle obr.1.3 bude $G_R(s) = r_0$
- Zvyšováním zesílení r_0 regulátoru dosáhneme stav, kdy systém netlumeně kmitá o konstantní apmlitude a konstantní periode, při které odečteme velikost zesílení r_{0k} a velikost doby kmitu T_k.
- Podle odečtených hodnot kritického zesílení r_{0k} a kritické doby kmitu T_k stanovíme podle tab.1.1 parametry požadovaného typu regulátoru.

Využití této metody je jednoduché a v praxi často používané, nemůžeme však říct, že jsme dosáhli nejleší možnou regulaci i když se k ní můžeme v některých případech přiblížit. Metodu Ziegler-Nichols nejde použít u strukturálně stabilních nebo nestabilních obvodů, které nelze přivést na hranici stability pro odečtení kritického zesílení r_{0k}.



Obrázek 1.3: Zpětnovazební řízení

typ	r_0	T_i	T_d
P	$0,5r_{0k}$	-	-
PI	$0,45r_{0k}$	$0.83T_k$	-
PD	$0,4r_{0k}$	-	$0,05T_k$
PID	$0,6r_{0k}$	$0,5T_k$	$0,12T_k$

Tabulka 1.1: Návrh podle ZN

1.2.2 Metoda geometrického místa kořenů

Metoda návrhu regulátorů pomocí geometrického místa kořenů (GMK) je založena na vykreslení všech možných kořenů uzavřené smyčky, vzhledem k měnícímu se parametru K, který představuje zesílení regulátoru.

Při praktickém návrhu pomocí metody GMK se v současnosti využívá řada softwarových produktů. Například MATLAB poskytuje nástroj sisotool, pomocí kterého lze jednoduše a interaktivně navrhovat a sledovat parametry regulátorů. Další informace můžeme nalézt v [1].

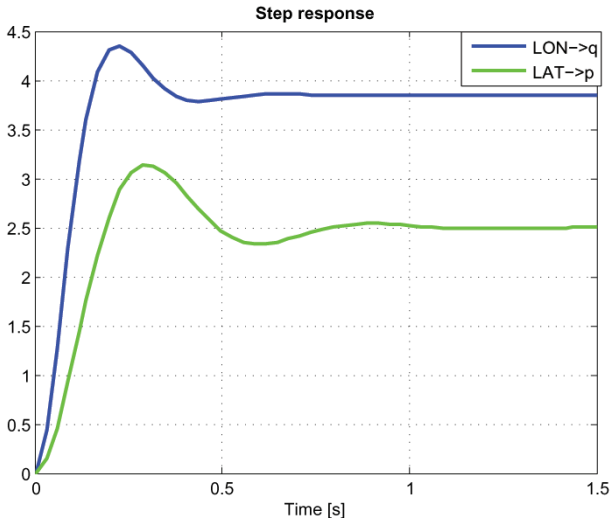
1.3 Návrh regulátoru pro stabilizaci polohy helikoptéry

Na začátku kapitoly jsme s využitím dostupných informací s webovských stránek zjistili přenosy

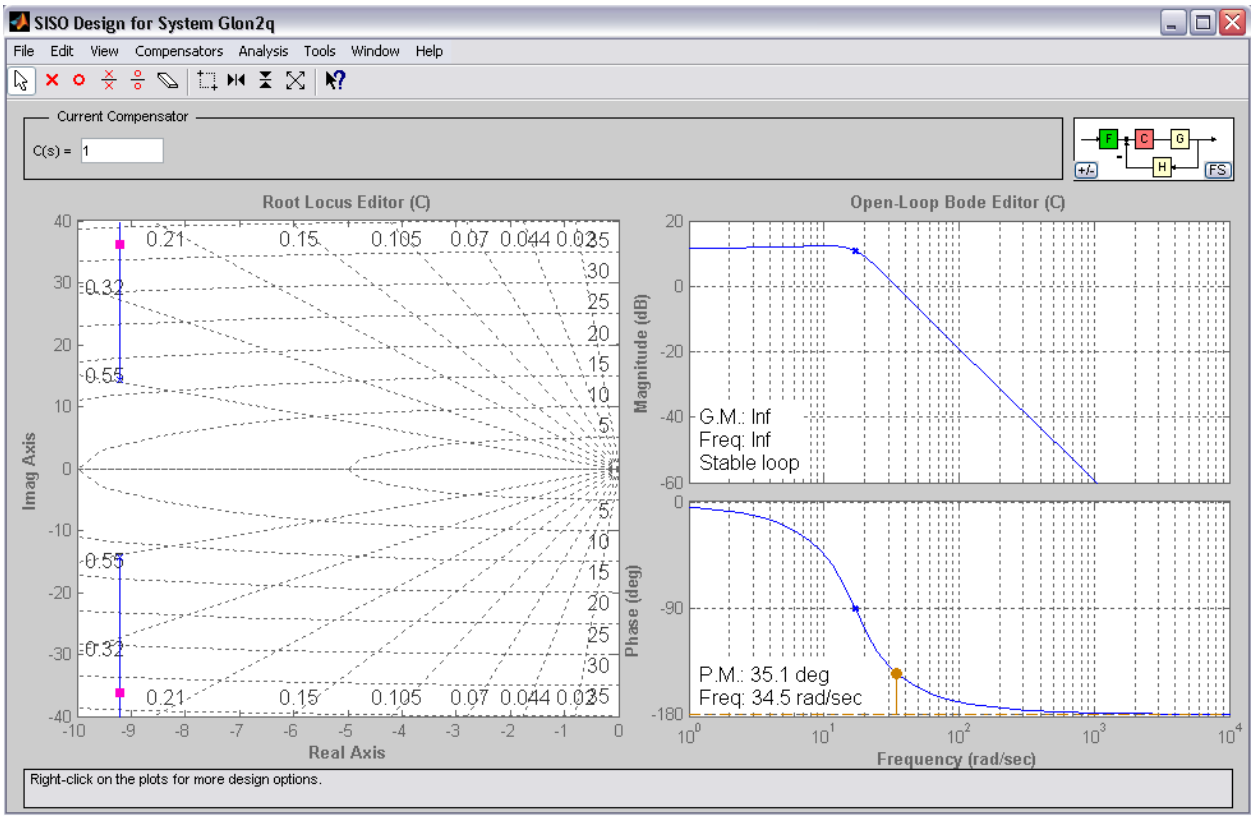
$$G_{lon\to q}(s) = \frac{1107}{s^2 + 18,42s + 287,5} \tag{1.3}$$

$$G_{lat\to p}(s) = \frac{327,4}{s^2 + 9,061s + 130,9} \tag{1.4}$$

Na obr 1.3 je odezva přenosů 1.3 a 1.4 na jednotkový skok. Z charakteristiky je vidět velká regulační odchylka, kterou se budeme snažit vhodným typem a volbou parametrů regulátorů odstranit při zachování nebo zlepšení dalších parametrů jako doba náběhu případně překmit. Regulační odchylku systému může odstránit integrační složka regulátoru, zaměříme se tedy na návrh PI regulátorů. Na obr. 1.5 je vidět vykreslení GMK pro přenos 1.3 (stejný charakter má i přenos 1.4). Můžeme pozorovat, že polohy pólů se nemůžou pro jakékoliv zesílení proporcionální složky dostat do nestabilní oblasti kladné reálné osy.



Obrázek 1.4: Přechodová charakteristika



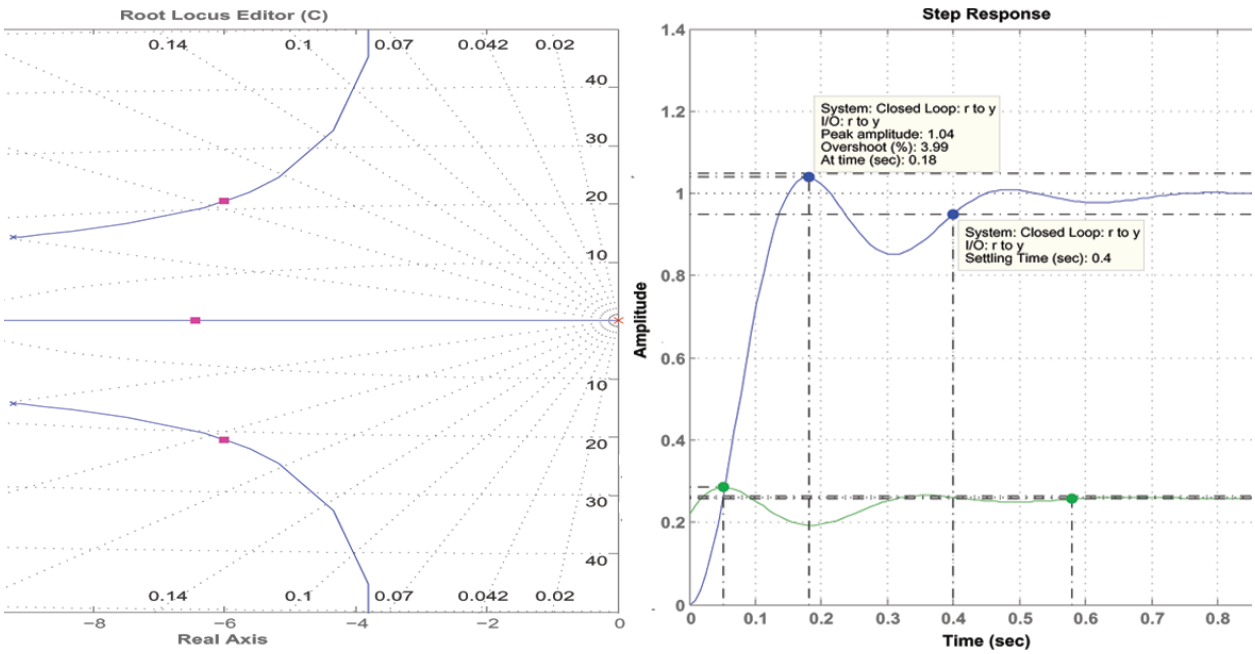
Obrázek 1.5: Nástroj MATLABu *sisotool* pro vykreslení GMK uzavřené smyčky pro přenos $G_{lon\to q}$podle1.3

1.3.1 Ziegler-Nichols

Z tvaru charakteristického polynomu uzavřené smyčky můžeme vidět, že se jedná o systém strukturálně stabilný, který nemůžeme přivést změnou zesílení podle koncepce metody ZN na mez stability, proto nelze uvedenou metodu využít pro návrh regulátorů. Můžeme vidět taky na GMK na obr.1.5

1.3.2 Metoda geometrického místa kořenů

Pro nulovou regulační odchylku odezvy na jednotkový skok přidáme na GMK pól do 0, reprezentující integrátor. Pro optimální seřízení se pokousíme nastavit změnou zesílení regulátoru trojici předominantních pólů.

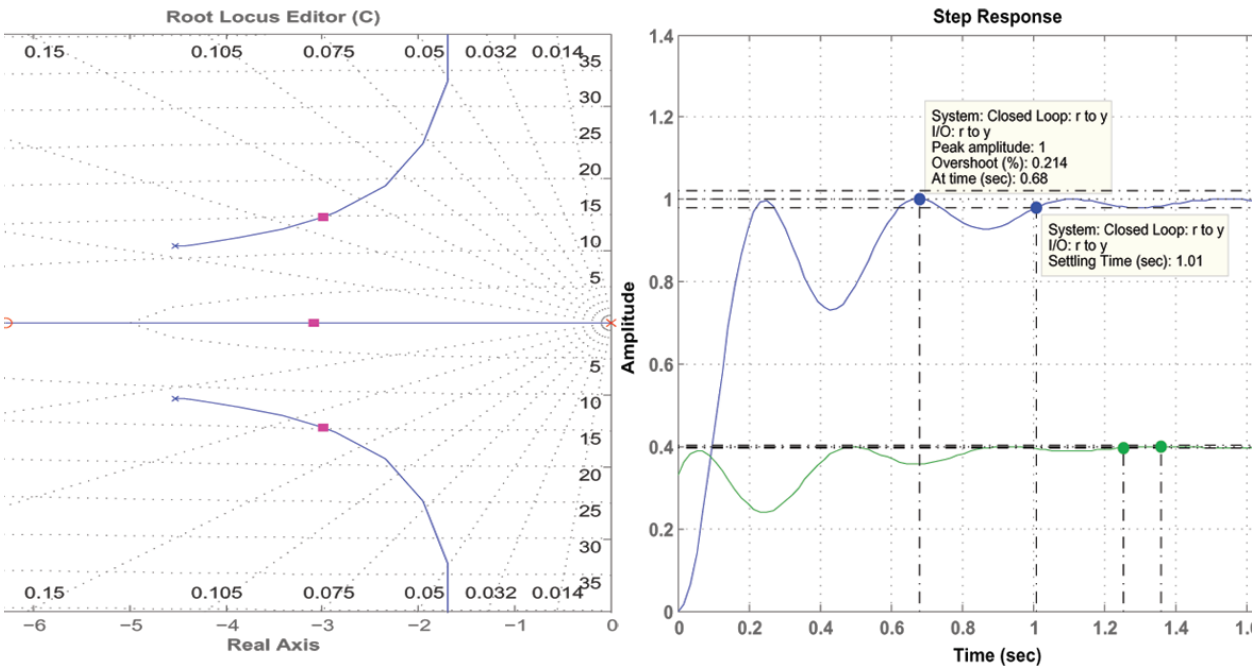


Obrázek 1.6: GMK vykresleno pro přenos 1.3

PI regulátor pro q *pitch* na obr.1.6 bude mít přenos 1.5

$$C_{lon\to q}(s) = 0,2191 + \frac{2.64}{s} \tag{1.5}$$

odkud pro konstanty PI regulátoru můžeme psát $k_p = 0,2191$ a $k_i = 2,64$.



Obrázek 1.7: GMK vykresleno pro přenos 1.3

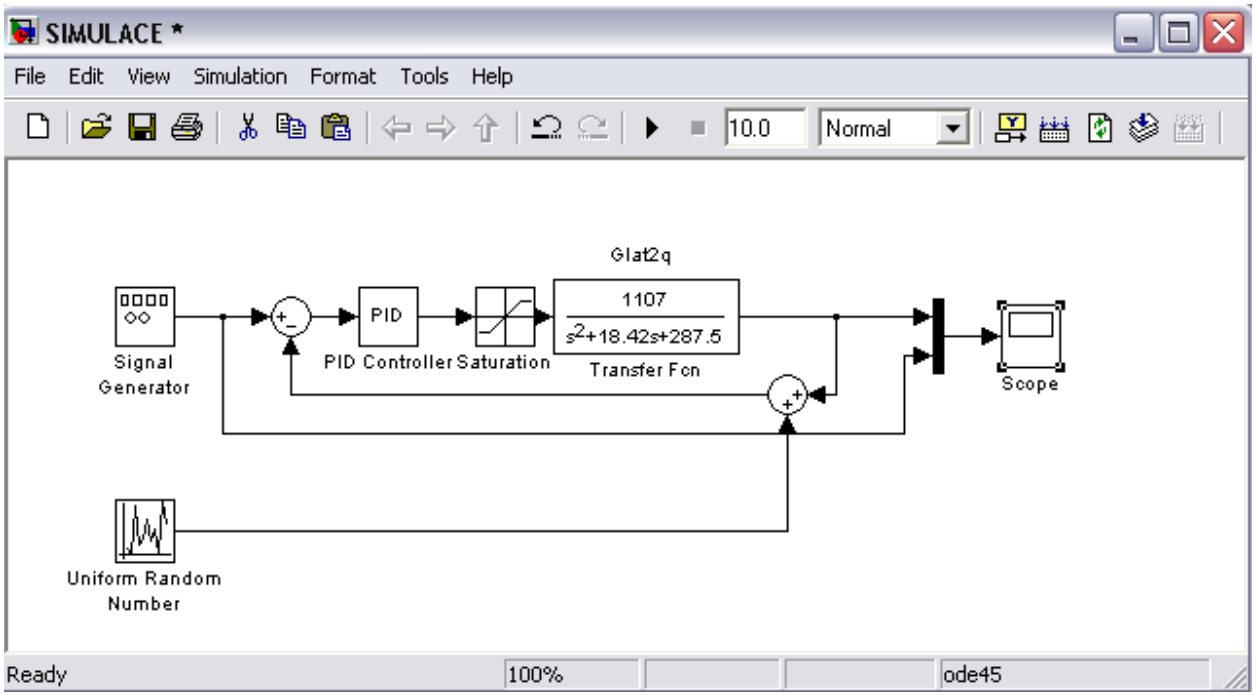
PI regulátor pro p *roll* podle obr.1.7 bude mít přenos regulátoru tvar 1.6

$$C_{lat\to p}(s) = 0,3328 + \frac{2,08}{s} \tag{1.6}$$

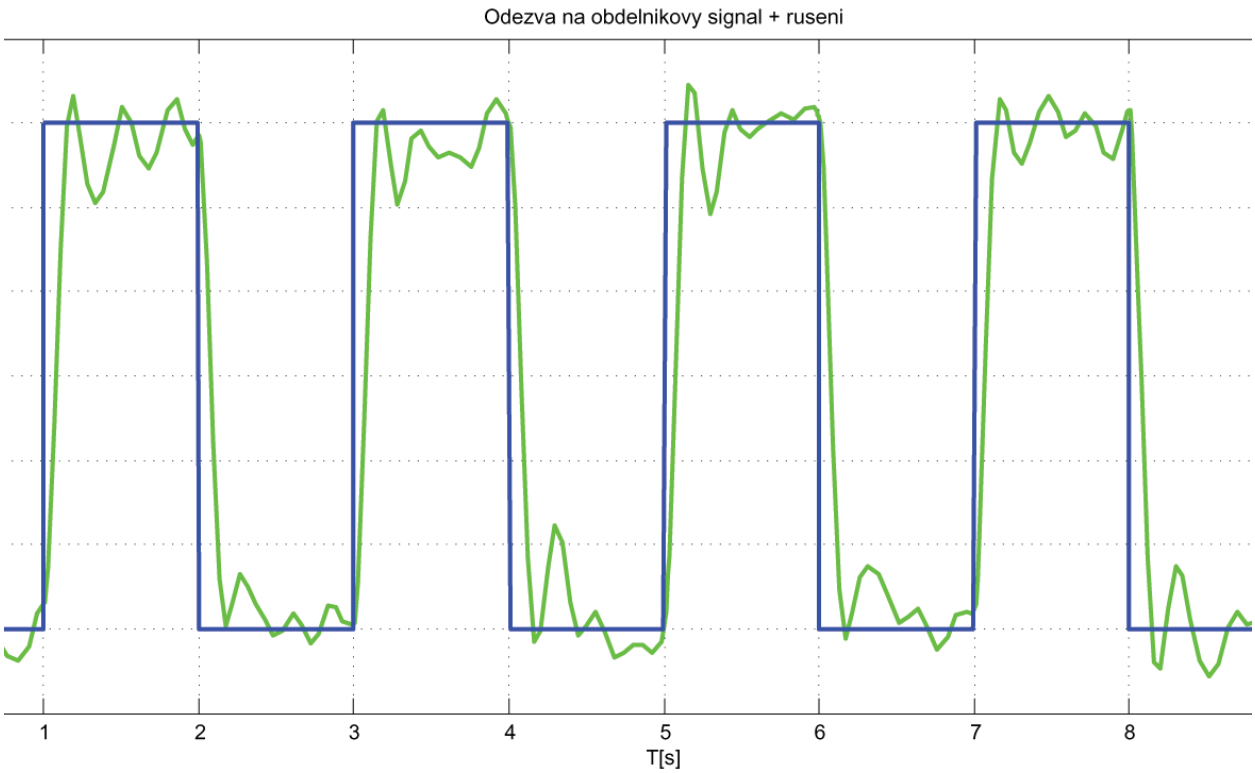
odkud pro konstanty PI regulátoru můžeme psát $k_p = 0,3328$ a $k_i = 2,08$.

1.3.3 Simulace navržených regulátorů

Na obr.1.8 je zobrazené schéma simulace v prostředí simulink v MATLABu a následujícím obrázku obr.1.9 můžeme vidět průběh regulace simulován pomocí simulinku.



Obrázek 1.8: Schéma v simulinku pro simulaci regulátoru



Obrázek 1.9: Simulace odezvy přenosu $G_{lat \to q}$na obdélnikový signál s rušením

Závěr

Webové stránky projektu RAMA jsou vedené s velkou pečlivostí a i po nekolikanásobném prohlédnutí nebyli nalezeny zásadní chyby nebo nedostatky, které by znemožnily sdílení letových dat nebo návrh regulátorů. Většina informací je poskytnuta v přehledné formě a nalezený chyby jsou technického charakteru. V další části byl navržen regulátor PI pro stabilizaci polohy pomocí metody GMK z poskytnutých dat prezentačního prostředí. Návrh regulátorů pro stabilizaci a řízení helikoptéry je obecne nesmírně složitý ze samotné povahy dynamiky helikoptéry. Proto jsme se omezili jen na demonstraci návrhu jednoduchého regulátoru.

Kapitola 2

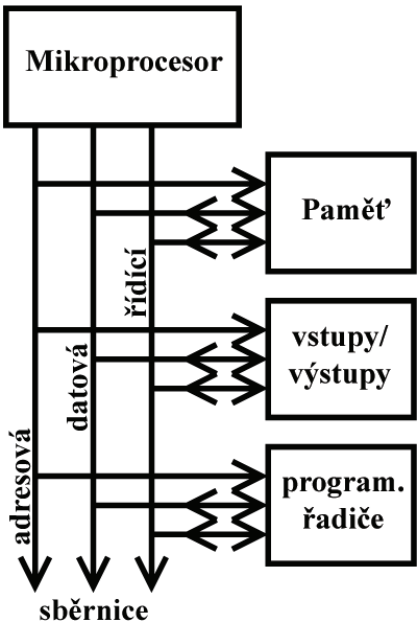
Projek stabilizované základny

Úvod

Projekt vývoje stabilizováné základny pro malá UAV je společným projektem K335, K333 FEL ČVUT a VTUL Praha. Ve druhé kapitole se zaměříme na možnost řízení plošiny mikrokontroléry ATMEL s jádrem AVR, uvedeme několik specifikací řízení a následně provedeme testy časové náročnosti vybraných algoritmů běžících na vybraném mikrokontroléru AT91SAM7X256. V závěru vyhodnotíme výsledky. Vybraný mikrokontrolér bude provádět sběr dat ze senzorů, realizovat algoritmy řízení, řídit akční členy a vyhodnocovat informace ze systému řízení vyšší úrovně. Předpokládá se, že nebude nasazen operační systéme reálného času, spravující všechyny úkoly mikrokontroléru.

2.1 Použití mikroprocesorů v řídících systémech účelových aplikací

2.1.1 Mikrokontroléry



Obrázek 2.1: Mikropočítač

Na obr. 2.1 můžeme vidět strukturu mikropočítače. Jednočipový mikropočítač je kompletní malý integrovaný mikropočítač, který je schopen samostatně snímat informaci na svých vstupech, vyhodnotit ji podle uloženého programu a na základě toho ovládat své chování a výstupy.

2.1.2 Systémy vestavěných aplikací

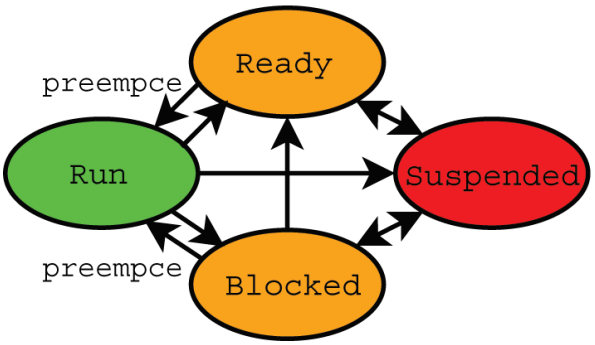
Z hlediska dalších úvah je účelné se pozastavit u definice vestavěných (embedded) systémů. Mohou být definovány jako kombinace hardwarových a softwarových prostředků, které jsou určeny pro řízení systému, procesu nebo zařízení. Jsou přímo fyzickou součástí zařízení, které řídí, a dávají zařízení ve kterém jsou integrovány vyšší stupeň inteligence. Většina embedded systémů vykonává svou činnost v reálném čase s omezenými systémovými zdroji. Jako příklady můžeme brát aplikace v letectví, medicíně, dopravě a poslední dobou pronikají také do zařízení běžného života.

2.1.3 Systémy běžící v reálném čase

Systémy bežící v reálném čase mají následující vlastnosti:

- Mají zaručenou a definovanou dobu odezvy jednotlivých úkolů (tasků).
- Každý přicházející signál je zpracován dostatečně rychle v souladu s požadavky, které jsou na systém kladeny.
- Systémy reálného času musí zaručovat absolutní spolehlivost
- Umožňují simultánní způsob bráce, t.j. několik událostí může běžet současně

Pro zabezpečení požadovaných vlastností je mikropočítač realizující řídící operace většinou vybaven operačním systémem reálného času (RTOS), který spravuje výpočetní prostředky. Požadavky na řežii RTOS jsou značně rozdílné a začínají se u 3kB ROM a 20B/task RAM u malých operačných systémů jako jsou Tiny52Kernel, až po 50 nebo 250 kB ROM a 10 až 30 kB RAM u největčích 32bitových operačních systémů.



Obrázek 2.2: Stavy tasku

Na obr. 2.2 můžeme vidět nekterý z možných stavů ve kterém se právě může task nacházet. Ve stavu *RUN* se může v daném okamžiku nacházet jen jediný task.

- **Run** - Aktuálně běžící task
- **Blocked** - Task čeká na zprávy, události (*events*) nebo zpoždění*(delay)*.
- **Ready** - Tasky, které jsou připravené a čekají na zpracování
- **Suspended** - Task zablokován pro další zpracování

V oblasti RT systémů definujeme
MULTITASKING - Několik tasků běží z hlediska uživatele nebo nadřazeného systému zdánlivě paralelně jako samostatné programy na různých prostředcích.

PREEMPTING - Preempcí se nazývá přerušení vykonávání tasku plánovačem (*scheduler*) ve stavě *RUN* do kterého přechází task s vyšší prioritou. Přerušený task přechází do stavu *READY*

INVERZE PRIORIT - Slouží na zamezení vzniku stavu, kdy task s vyšší prioritou čeká na zdroj, který bude uvolněn teprve taskem s nižší prioritou. Bez inverze priorit by byl systém zastaven a proto task s nižší pritou dostává k dispozici výpočetní kapacitu systému na uvolnění blokovaných zdrojů.

Požadavky na RT systémy můžeme shrnout do následujících bodů:

- Včasná a deterministická reakce na definované i nedefinované události
- Dodržování požadavků na časování sběrnice (reakční doby)

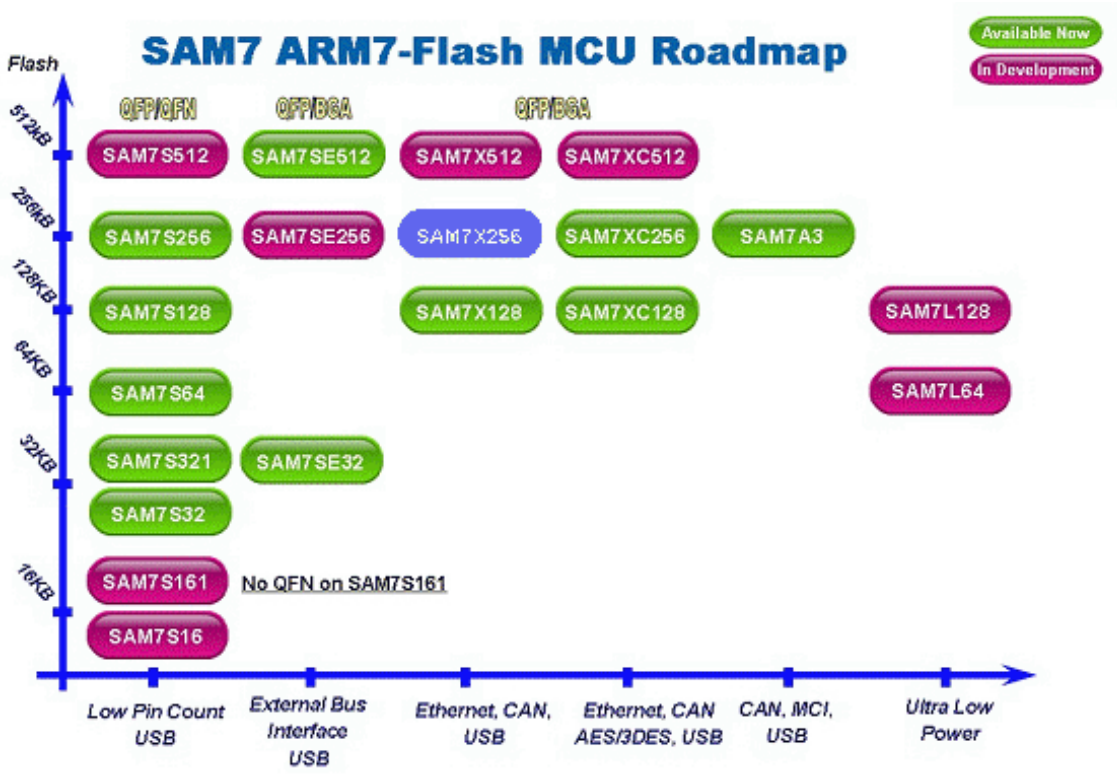
V aplikaci systému stabilizované základny si můžeme task představit jako nejmenší samostatnou programovou jednotku určenou pro samostatné zpracování uzavřené formulace úlohy. Každý task je v jazyce C založen jako nekonečná smyčka, v níž probíhá obsluha, řešení dané úlohy. Bez operačního systému reálného času může na mikrokontroléry běžet jen jediný task založen funkcí `main()`. (To však neznamená, že ve funkci main nemůže být implementován kód vykonávající obsluhu, řešení více úkolu současně). V naší aplikaci můžeme dekompozicí problému nalézt nekolik úkolů řešení, třeba: sběr dat ze senzorů, realizace algoritmu řízení, komunikace s nadřazeným systémem, obsluha akčních členů,... Na to, aby byl mikrokontrolér schopen zvládnout všechny požadavky, musí disponovat dostatečnou pamětí a zejména výpočetním výkonem. Testy výpočetního výkonu budou provedeny v části 2.5

2.2 32-bitové mikrokontroléry ATMEL řady ARM

Základní charakteristiky 32-bitových mikrokontroléru ATMEL řady ARM mohou být shrnuta do následujicích bodů:

- Deterministické chování - předvídatelná odezva na událost reálného času se specifikovaným počtem hodinových cyklů
- Vestavěná pamět FLASH
- Množství rozhrání
- Nízká cena vývojových nástrojů
- Spolehlivá, dlouholetým vývojem a zlepšováním ověřená architektura
- Příznivý poměr výpočetního výkonu a spotřeby.

Kompletní srovnávací tabulku všech typů 32-bitových mikrokotroléru ATMEL AVR můžeme nalézt na přiloženém CD. V užším výběru se nacházeli procesory řady SAM7 podle obr.2.3.



Obrázek 2.3: Přehled mikrokontrolérů řady SAM7

2.2.1 Výběr vhodného procesoru

Pravidla pro výběr procesorů obecně mohou být shrnuta do následujících bodů:

1. dostateční výpočetní výkon pro realizaci algoritmů řízení
2. obsluha více funkcí současně
3. počet a typ rozhrání (**SPI**, UART, CAN, ...)
4. cena a dostupnost procesoru na trhu
5. cena a dostupnost vývojových prostředků
6. perspektivnost, kompatibilita a informační servis

Na základě katalogových listů a po zvážení předpokládaných nároků algoritmů řízení[1] s přihlédnutím k ostatním vlastnostem jako počet periferií a velikost paměti byl vybrán mikrokontrolér firmy ATMEL rodiny AVR typ AT91SAM7X256, který patří mezi výkonnější představitele kategorie 32bit ARM procesorů ATMEL.

2.3 Mikrokontrolér AT91SAM7X256

2.3.1 Charakteristika

Mikrokontrolér AT91SAM7X256 je určen pro obecné použití, zejména v oblasti embedded systémů pracujících v reálném čase. To je vhodná charakteristika pro možné nasazení

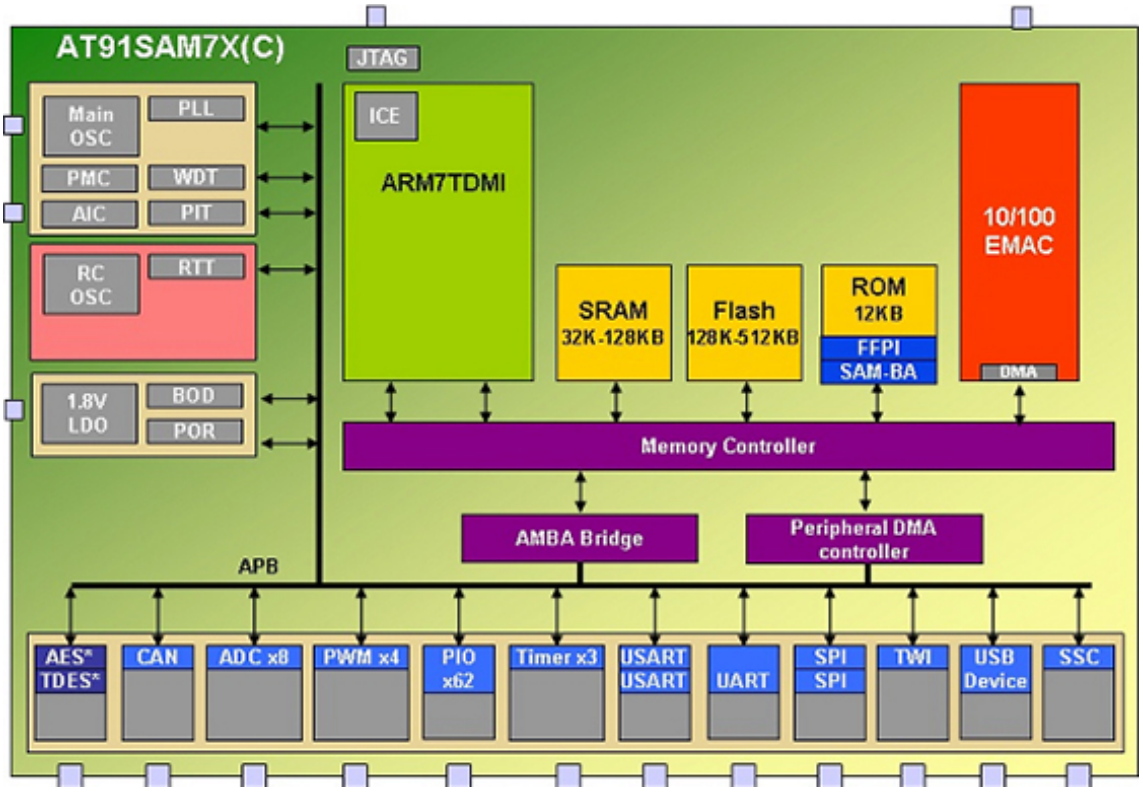
[1] algoritmy řízení jsou ve vývoji

jako řídícího mikrokontroléru pro řízení stabilizované základny.
Má 32bit RISC architektúru AVR s jádrem 7TDMI. Jádro 7TDMI provádí instrukce jako 3 stupňový pipelining což umožňuje mimo jiné v optimálním případě provedení jedné instrukce na jeden hodinový puls systémového časovače. (Pak $1MHz$ odpovídá 1MIPS Blokovou strukturu mikrokontroléru AT91SAM7X můžeme vidět na obr. 2.5. Instrukční soubor využívá sadu instrukcí RISC o šířce 16 bitů.

HLAVNÍ PARAMETRY PROCESORU:

- maximální taktovací frekvence $f_{MCK} = 55MHz$
- paměť FLASH 256 kB
- paměť SRAM 64 kB

Obrázek 2.4: AT91SAM7X256



Obrázek 2.5: Blokové schema mikrokontroléru řady AT91SAM7. Zdroj

2.3.2 Čítače/časovače

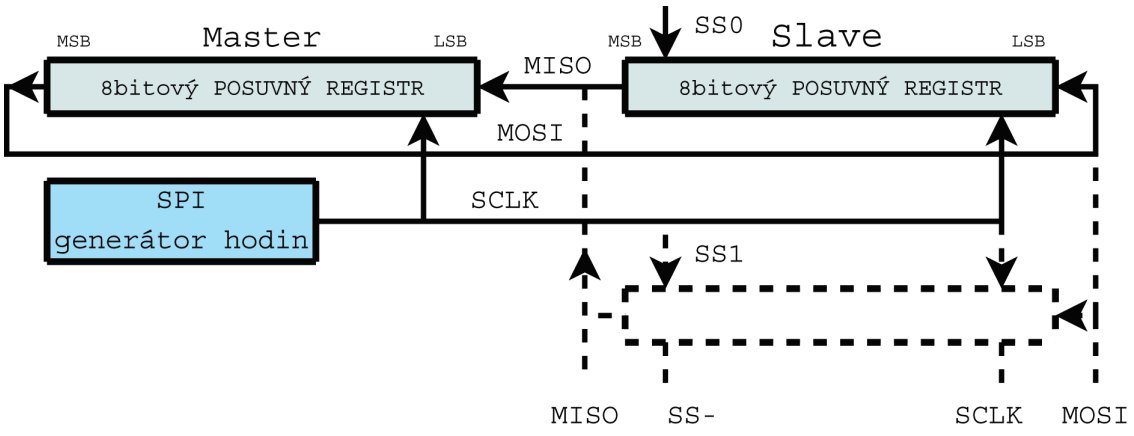
Mikrokontrolér obsahuje tři identické na sobě nezávislé 16bitové čítače/časovače využitelné pro měření frekvence, počítání událostí, měření času, generování pulsů nebo časového zpoždění. Jako zdroj hodin lze použít výběr ze 3 externích vstupů nebo z 5 interních vstupů, kterých frekvence je přimo dělícím poměrem odvozena od hlavní systémové frekvence MCK. Výběr zdroje pro čítač se realizuje zapsáním příslušných hodnot do registru TCCLKS podle tab.2.1

Zdroj	frekvence čítání	hodnota v TCCLKS
TIMER1	$MCK/2$	0x0
TIMER2	$MCK/8$	0x1
TIMER3	$MCK/32$	0x2
TIMER4	$MCK/128$	0x3
TIMER5	$MCK/1024$	0x4

Tabulka 2.1: Interní zdroje hodin pro čítač

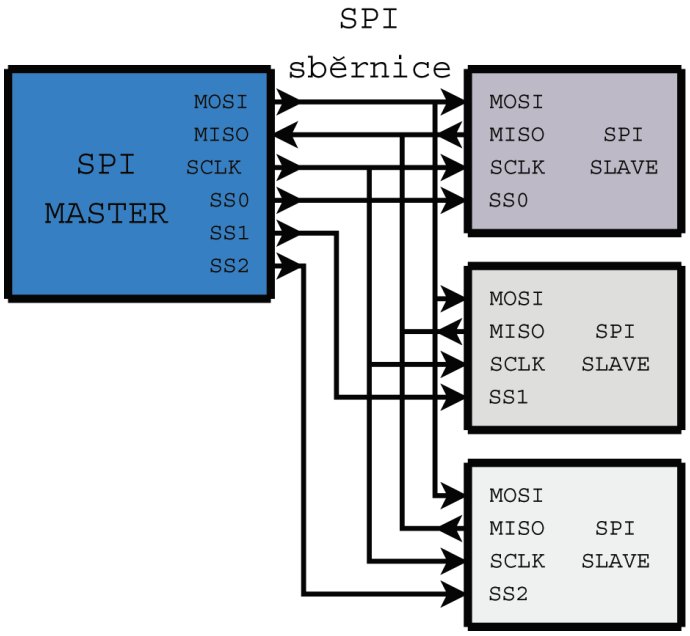
2.3.3 Rozhrání SPI

Rozhrání SPI (*Serial Peripheral Interface*) je rozšířené komunikační rozhrání, umožňující jednosměrnou nebo obousměrnou komunikaci mezi procesorem a jeho periferiemi. (sériové paměti, generátory PWM, zobrazovače, senzory, ...). Používá se většinou pro komunikaci uvnitř vestavěných (empedded) zařízení na relativně malé vzdálenosti, což umožňuje na rozdíl od jiných sériových rozhrání využití jednoduchého komunikačního protokolu. Dochází jen k přenosu užitečné informace a odbourává se potřeba použití speciálních protokolů pro zabezpečení komunikace, které vyžadují přenos řídících a zabezpečujících bitů (start bit, stop bit, parita, řízení). Tak může být dosaženo vyšších přenosových rychlostí.



Obrázek 2.6: Realizace SPI pomocí posuvných registrů

Pro sériový přenos datových bitů využívá v zásadě SPI rozhrání posuvní registr způsobem zobrazeným na obr. 2.6. V průběhu transferu dat jedno a jen jedno zařízení s SPI rozhráním funguje jako řídící (MASTER), ostatní zařízení připojené na sběrnici "poslouchají" v podřízeném režimu (SLAVE). V úloze Master se může střídat více zařízení, ale jen po dokončení aktuálního přenosu skupiny bitů o zvolené šířce slova.



Obrázek 2.7: Způsob propojení zařízení s SPI

Na obr. 2.7 můžeme vidět způsob propojení mezi řídícím zařízením MASTER a zařízeními SLAVE. Řídící obvod na rozdíl od podřízeného generováním taktovacího signálu určuje kdy a jakou rychlostí se boudou přenášet data. Svým výstupem SS také rozhoduje, se kterým podřízeným zařízením bude probíhat komunikace.

Význam jednotlivých vodičů sběrnice SPI je následovný:

- **MISO** (*Master In Slave Out*) - datový vodič obsahuje v určitých okamžicích platná výstupní data (jeden bit) z vybraného zařízení SLAVE, která jsou přesouvána do zařízení MASTER
- **MOSI** (*Master Out Slave In*) - datový vodič přivádí data z jednotky MASTER do vybrané jednotky SLAVE (jeden bit)
- **SCLK** (*Clock*) - potvrzovací vodič buzený zařízením MASTER, který svou náběžnou nebo sestupní hranou (podle režimu SPI) definuje platnost přenášených dat (bitů) na datových vodičích. Přijímací stranu vyzývá ke přijetí a odeslání dat a tak řídí rychlost přenosu dat.
- **SS** (*System Select*) - vodič výběru periferního zařízení (pro každé zařízení zvlášť), slouží k výběru zařízení SLAVE, se kterým bude probíhat komunikace

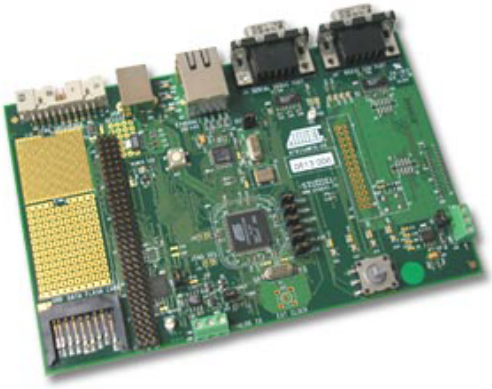
2.4 Vývojový Kit AT91SAM7X-EK

2.4.1 Základní popis

Vývojový kit AT91SAM7X-EK je určen pro vývoj a testování aplikací běžících na procesoru AT91SAM7X256/128. Sada vývojových prostředků obsahuje: Vývojovou desku AT91SAM7X-EK, komunikační rozhrání Jlink , CD s vývojovým prostředím Embedded Workbench firmy IAR.

2.4.2 Hardwarová část

Na obr.2.8 je vývojová deska obsahující mikrokontrolér AT91SAM7X256, umožňující testování a ladění všech funkcí MCU. V procesu vývoje programu umožňuje jeho nahrání do RAM nebo FLASH paměti prostřednícvím rozhrání JTAG a následně jeho krokování a ladění. Obsahuje následující rozhrání USB, DBGU, RS232, JTAG/ICE, CAN, MII Ethernet, PWM, konektor DataFlash. Bližší popis včetne blokového schematu je možno nalézt na stránkách výrobce IAR nebo na přiloženém CD.



Obrázek 2.8: Vývojová deska

Adapter USB⇔JTAG, který slouží pro komunikaci vývojový desky s PC je na obr. 2.9. Bližší informace nalezneme na přiloženém CD.

Obrázek 2.9: USB adaptér

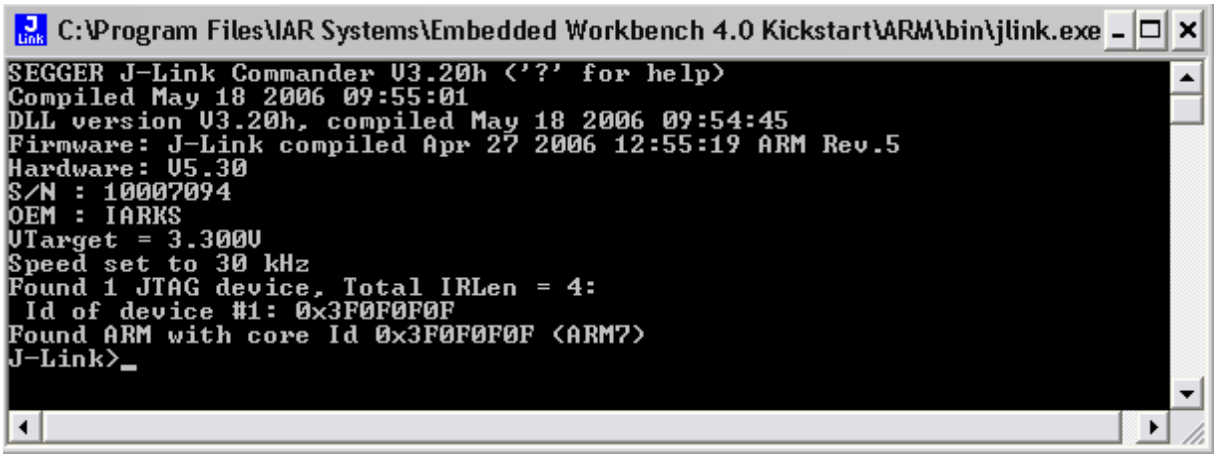
2.4.3 Zprovozneni vyvojoveho kitu

Práce s vývojový kitem probíhala na přenosném počítači s parametry : Intel C2D 2x2GHz, 2GB RAM, 120GB HDD, kde byl instalován příslušný vývový software a ovládače ke komunikačnímu rozhrání JTAG. Průběh zprovoznění vývojového kitu lze popsat v následujících bodech:

- **Instalace softwarových nástrojů do PC** - Omezenou verzi vývojového prostředí IAR Embedded Workbench můžeme získat po zaregitrování získání sériového čísla. Naše verze použitá při vývoji má omezení velkosti kódu kompilátoru na 32kB.
- Instalace ovládačů J-Link

- **Verifikace nastavení propojek desky vývojového kitu** - spočívá v nastavení správné konfigurace propojovacích pinů na vývojové desce, které slouží k nastavení napájecích napětí, periferií, připadně mazání interní FLASH paměti. Správné umístnění nalezneme v katalogu ke vývojový desce na stránkách výrobce nebo taky na přiloženém CD.

- **Propojení všech součástí a ověření komunikace mezi PC a vývojovou deskou** - pro ověření správné instalace ovladačů a ověření vzájemné komunikace zadáme v příkazovém řádku v programovom adresáři příkaz `Jlink`. Následně proběhne ověření komunikace a vypsání hlášení na monitor. V dalším kroku můžeme přistoupit k otevření výrobcem přiložených projektů.

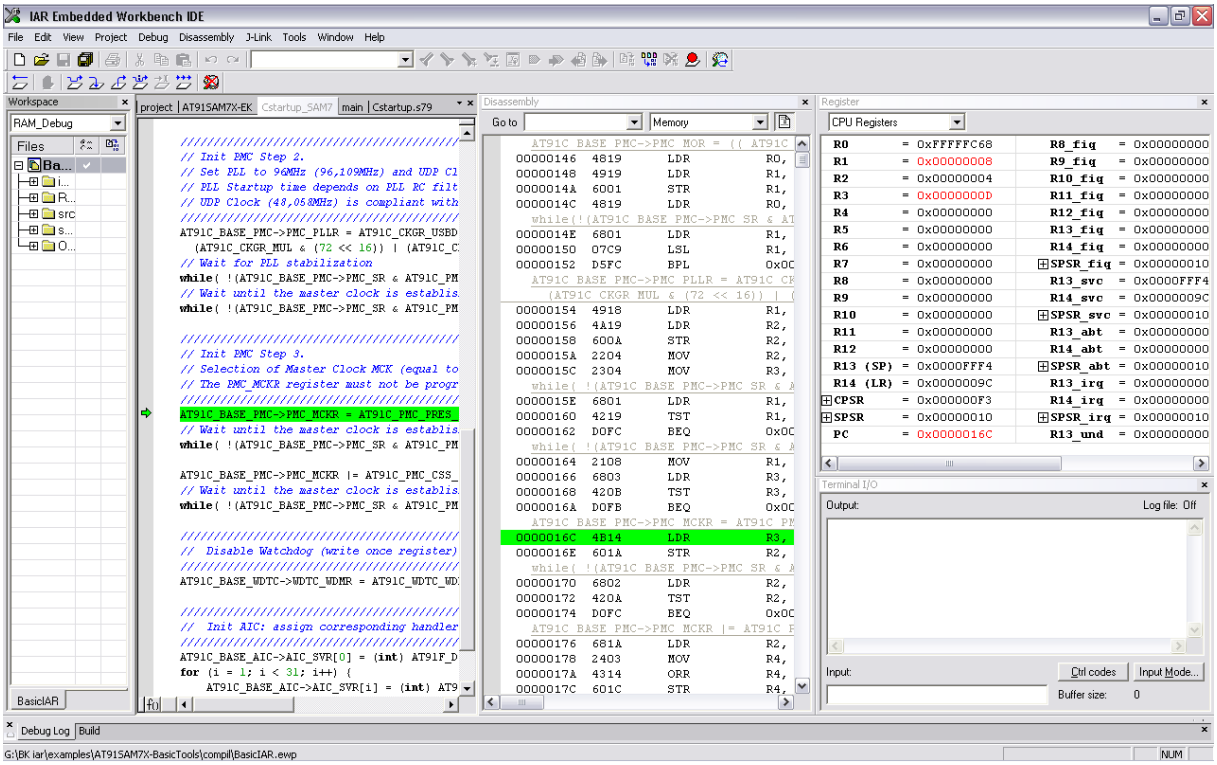


Obrázek 2.10: Ověření správné instalace rozhraní Jlink

- **Test dodaných programů a ověření funkce** - V instalaci programu EW jsou k dispozici taky demonstrační programy pro ověření základních funkcí a zprovoznění kitu (blikání LED, přerušení, atd.), které ulehčují počáteční práci ve vývojovém prostředí.

2.4.4 Vývojové prostředí IAR Embedded Workbench

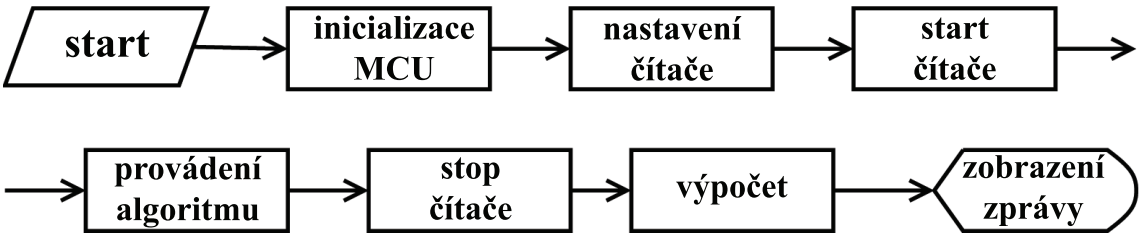
Embedded Workbench je vývojové prostředí Švédské firmy IAR Systems o které lze říct, že je v současné době vedoucím výrobcem nástrojů pro vývoj softwaru v oblasti MCU. Dlouholeté zkušenosti IAR s tvorbou vývojových prostředků pro aplikace MCU vedly k tvorbě integrovaných vývojových prostředí pro programování MCU. (IDE - Integral Development Environment) Reprezentantem takového prostředí je i produkt EW, který do svého pracovního prostředí integruje správu projektů, editor, kompilátor, spojovací program, assembler, ladící program a další části, umožňující rychlou tvorbu efektivního zdrojového kódu a pokročilé možnosti simulace a ladění programu. Velkou výhodou je také plná podpora programování MCU v jazyce C a široká podpora MCU různých typů a výrobců. Všechny důležité parametry pro kompilátor, assembler a linker (typ MCU, formát výstupu, model paměti atd.) jsou přednastaveny výchozími hodnotami a pro aktuální projekt se mohou měnit v odpovídajících zadávacích maskách pro používaný MCU. Ve KickStart edition je EW plne funkční s omezením na 32 kB.



Obrázek 2.11: Vývojové prostředí Embedded Workbench v režimu ladění

2.4.5 Způsob měření časové náročnosti algoritmů

Pro účely měření časových výpočetních nároků jsme navrhli a implementovali program podle obr. 2.12. Koncepce využívá jedného ze trí nezávislých 16bitových časovačů MCU, který čítá pulsy přicházející od systémového časovače přes vhodně nastavený dělící poměr (režim čitače tab.2.2). Použití a běh časovače nemá žádné nároky na výpočetní čas. Po startu a inicializaci MCU dojde k nastavení čitače/časovače, nulování a jeho startu. Před začátkem provádění testovaného algoritmu je zachycena aktuální hodnota čitače. Následně dochází k běhu testovaného algoritmu na pozadí kterého je čítač inkrementován. Po skončení práce dochází opět k zachycení hodnoty a výsledek je předán na spracování a následný výpis na terminálový výstup.



Obrázek 2.12: Vývojový diagram navrhnutého programu

- **Inicializace MCU** - Před spuštěním hlavního programu funkcí `main()` je proveden tzv. startup program `Cstartup.s79`. Tento assemblerovský program je výrobcem kompilátoru dodán do zdrojového a objektového kódu. Hlavním úkolem tohoto programu

je start MCU. Vykonává se inicializace registrů, paměti RAM, zásobníků, vstupně-výstupních částí, dátových oblastí a pod. Tento program je závislý na použitém mikrokontroléru. Následně dochází k volání hlavní funkce `main()`.

- **Nastavení čitače** - Nastavení čitače probíha v hlavní funkci `main()` voláním funkce `AT91F_TC_Open( AT91PS_TC TC_pt, unsigned int Mode, unsigned int TimerId)` při kterém dochází také k předání parametrů pro výběr čitače/časovače(TC) a výběr režimu čítání podle tab.2.2 . Časové hodnoty v této tabulce jsou platné pro $MCK = 48MHz$.

režim čitače	rozlišení [ns]	maximální čas [ms]
2	42	2,2775
8	166	10,9100
32	667	43,6401
128	2660	174,54
1024	21300	1396,4

Tabulka 2.2: Režim čítače

- **Start čitače** - Start a nulování čitače je provedeno pomocí zápisu do řidícího registru čítače TC0. (`AT91C_BASE_TC0->TC_CCR = AT91C_TC_SWTRG ;`) Následně před startem testovaného programu je zaznamenána aktuální hodnota stavu čitače TC0. (`valueCounter0 = AT91C_BASE_TC0->TC_CV;`)

- **Provádění algoritmu** - V této částí dochází k běhu programu, provádění programu pro získání správných výsledků nesmí být přerušen žádostí o přerušení. To je možné realizovat zákazem přerušení. Testovací algoritmy jsou popsány v následující části 2.4.6.

- **Stop čitače** - Po skončení výpočtu dochází k zastavení čítání formou zachycení aktuálního stavu TC0. (`valueCounter1 = AT91C_BASE_TC0->TC_CV;`)

- **Výpočet**- Výpočet resp. přepočet počtu zachycených puslů během výpočtu je realizován podle 2.1

$$T_{alg} = \frac{1}{MCK} R_{tc} N_{tc} \quad [\mu s] \tag{2.1}$$

kde:
MCK $[MHz]$ je taktovací frekvence systémového časovače
R_{tc} $[-]$ je řežim čitače podle 2.2
N_{tc} $[-]$ je počet pulsů zaznamenaných během provádění algoritmu

Počet zachycených pulsů (`int result = valueCounter1-valueCounter0;`) během algoritmu je dál předán funkci (`convert()`), která na základě zjištění hodnoty MCK a režimu časovače R_{tc} vypočte délku trvání algoritmu v μs.

- **Zobrazení správy** - V poslední části jsou na terminálový výstup vypsány výsledky výpočtu a volána funkce `report()`, která vytiskne souhrní informace o průběhu měření. Obr.ref:Zprava.

2.4.6 Testovací algoritmy

Na tomto místě stručně definujeme a popíšeme testovací algoritmy, použité při měření výpočetního výkonu MCU podle kterých byly implementovány v jazyce C. Výpis algoritmů je součásti přílohy a kompletní program včetně komentářů lze nalézt na přiloženém CD.

Faktoriál čísla

Výpočet faktoriálu reprezentuje operaci součinu. Pro

Definice 2.4.1 *Faktoriálem čísla n nazveme číslo rovné součinu všech kladných celých čísel menších nebo rovných n.*

$$n! = 1.2...n = \prod_{k=1}^{n} k \quad \text{pro} n \geq 0 \tag{2.2}$$

IMPLEMENTACE V C: Algoritmus výpočtu faktoriálu byl implementován podle 2.2 nerekurzivní funkcí

```
faktorial()
```

Fibonacciho posloupnost

Fibonacciho posloupnost ve výpočetních testech reprezentuje operaci sčítání.

Definice 2.4.2 *Fibonacciho posloupnost můžeme definovat jako nekonečnou posloupnost přirozených čísel začínnající* $0, 1$*, kde každý další člen je součtem členů předchozích.*

$$F(n) = \begin{cases} 0 & \textit{pro } n = 0; \\ 1 & \textit{pro } n = 1; \\ F(n-1) + F(n-2) & \textit{pro } n > 1; \end{cases} \tag{2.3}$$

IMPLEMENTACE V C: Pro výpočet existuje nekolik algoritmů, my se omezíme na nerekurzivní výpočet pomocí pamatování posledních dvou členů. Implementující funkce má název

```
fibonacci()
```

Násobení matic

Algoritmus maticového násobení je častým algoritmem, který se mimo jiné využívá v celé řade řídících aplikací MCU.

Definice 2.4.3 *Nechť* $\boldsymbol{A} = (a_{i,j})$ *je matice typu* (m, n) *a* $\boldsymbol{B} = (b_{j,k})$ *je matice typu* (n, p)*. Pak je definován součin matic* $\boldsymbol{A}.\boldsymbol{B}$ *(nekomutující) jako matice typu* (m, p) *takto: každý prvek* $c_{i,k}$ *matice* $\boldsymbol{A}.\boldsymbol{B}$ *je dán vzorcem*

$$c_{i,k} = a_{i,1}b_{1,k} + a_{i,2}b_{2,k} + ... + a_{i,n}b_{n,k} = \sum_{j=1}^{n} a_{i,j}b_{j,k}, \qquad i \in \{1, \ldots, m\}, k \in \{1, \ldots, p\} \tag{2.4}$$

IMPLEMENTACE V C: Algoritmus maticového násobení byl implementován pomocí vztahu 2.4 funkcí

```
soucinMatic()
```

Transpozice matic

Transpozice je další maticovou operací využívanou při řídících aplikacích MCU.

Definice 2.4.4 *Nechť* $\boldsymbol{A} = (a_{i,j})$ *je matice typu* (m, n). *Pak matici*$\boldsymbol{A}^T = (a_{j,i})$, *která je typu* (m, n) *nazýváme transponovanou maticí k matici* $\boldsymbol{A}$. *Matice* $\boldsymbol{A}^T$ *vznikne z matice* $\boldsymbol{A}$ *přepsáním řádků matice* $\boldsymbol{A}$ *do sloupců matice* $\boldsymbol{A}^T$, *resp. přepsáním sloupců matice* $\boldsymbol{A}$ *do řádků matice* $\boldsymbol{A}^T$

IMPLEMENTACE V C: Operace transpozice matice byla implementováná pomocí definice 2.4.4 funkcí

```
tranMatic()
```

Výpočet determinantu

Realizace výpočtu determinantu matice

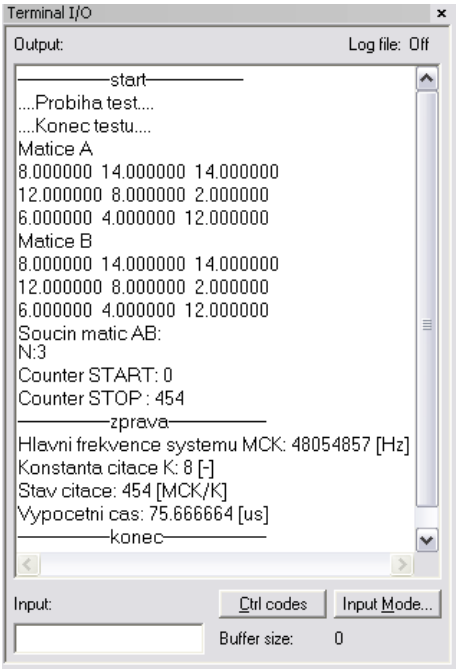
Věta 2.4.1 *Nechť* $\boldsymbol{A} = (a_{i,j})$ *čtvercová je matice typu* (n, n) *s hodností* n. *Pak determinant matice* $\boldsymbol{A}$, *můžeme spočítat převedením matice* $\boldsymbol{A}$ *pomocí Gaussovy eliminační metody na horní trojuhelníkovou, kde determinant tvoří součin prvků na hlavní diagonále upravené matice.*

IMPLEMENTACE V C: Algoritmus výpočtu determinantu matice byl implementován funkcí

```
vypocetDet()
```

2.5 Naměřené hodnoty

Tab.2.3 obsahuje naměřené hodnoty časových nároků jednotlivých algoritmů v závislosti na stupni N. Měření bylo prováděno v režimu ladění RAM debug spuštěním běhu programu, při frekvenci hlavního časovače $48054857Hz$. Výstup programu můžeme vidět na obr.2.13.



Obrázek 2.13: Závěrečná zpráva programu

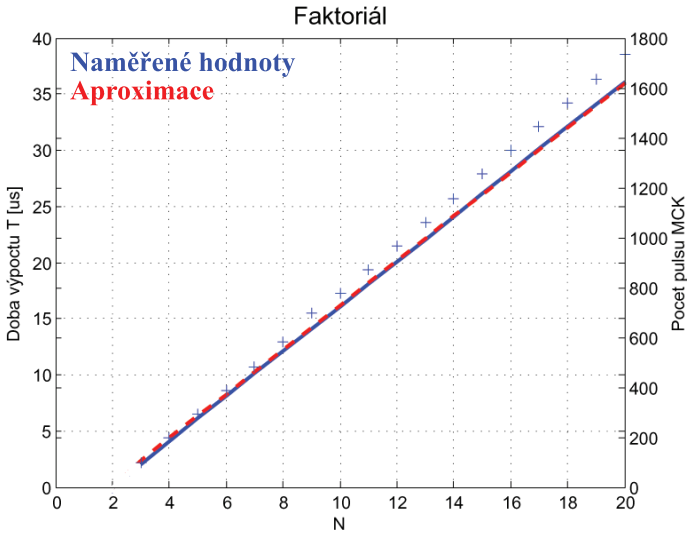
N	faktoriál	fibonacci	násobení M	transpozice M	determinant M
1	-	-	-	-	-
2	-	-	26,042	2,458	12.041
3	0,083	1,417	81,667	4,167	35.292
4	4,083	1,541	187,833	6,000	77.625
5	6,083	1,708	361,042	8,083	143.208
6	8,083	1,833	617,792	10,417	237.217
7	10,083	2,000	974,583	13,125	367.125
8	12,125	2,125	1447,917	15,750	529.500
9	14,083	2,292	2054,292	19,167	753.667
10	16,125	2,417	2810,167	22,500	1007.000
11	18,125	2,583	3732,167	26,583	1321,417
12	20,083	2,750	4836,167	30,750	1716,542
13	22,083	2,875	6140,167	34,792	2131,667
14	24,083	3,000	7658,833	39,208	2662,917
15	26,125	3,167	9410,500	44,000	3268,333
16	28,083	3,292	11409,330	49,042	3949,667
17	30,083	3,458	13674,000	55,375	4692,834
18	32,083	3,583	16220,000	61,000	5559,334
19	34,083	3,750	19065,334	67,000	6574,835
20	36,125	3,875	22223,334	73,333	7571,834

Tabulka 2.3: Tabulka naměřených hodnot

Dálka výpočtu je závislá zejména na:

- Architektuře mikrokontroléru (AVR,51, ...)

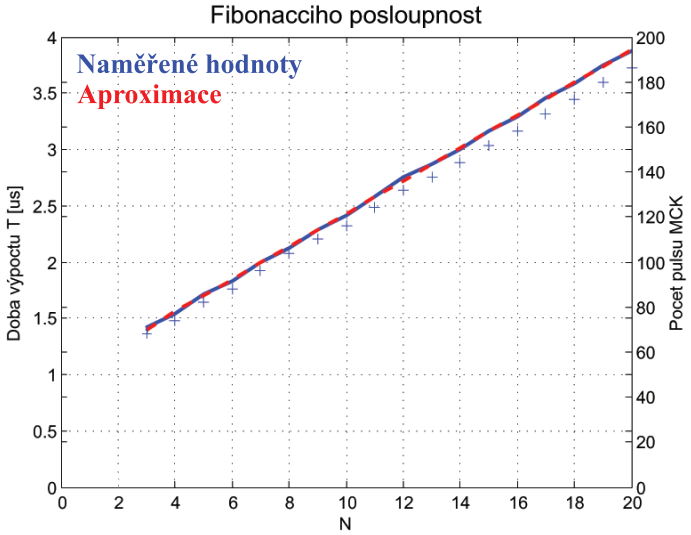
- Kmitočtu MCK systémového časovače
- Stupně optimalizace zdrojového kódu kompilátorem (speed, size)
- Typu proměnných(lokální, globální, INT,FLOAT)
- Konstrukci programu (smyčky, makra, cykly, přístup do paměti)



Obrázek 2.14: Výpočet faktoriálu

Na obr. 2.5 můžeme vidět lineární nárůst výpočetního času v závislosti na N. Zobrazenou závislost můžeme aproximovat pro hodnoty $N > 2$ vztahem 2.5.

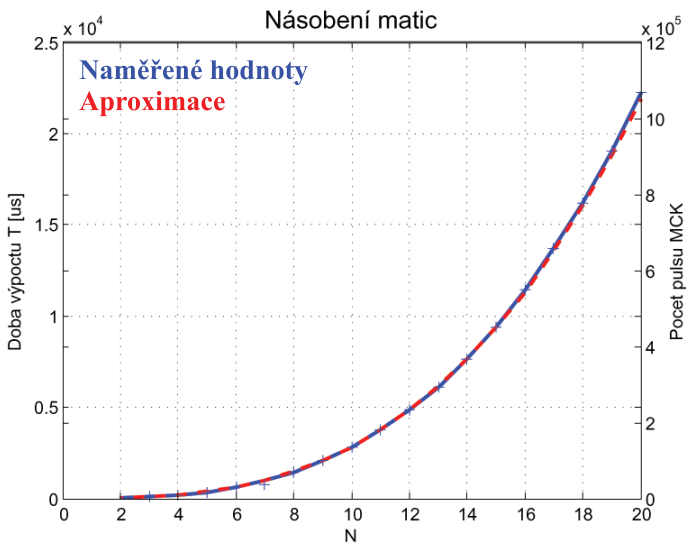
$$T_{alg} = 1,9744N - 1,5715 \quad [\mu s] \qquad (2.5)$$



Obrázek 2.15: Výpočet Fibonacciho řady

Lineární nárust výpočetního času v závislosti od N můžeme vidět taky u Fibonacciho posloupnosti obr. 2.15. Pro $N > 2$ můžeme charakteristiku aproximovat vztahem 2.6

$$T_{alg} = 0,1457N + 0,9728 \quad [\mu s] \qquad (2.6)$$

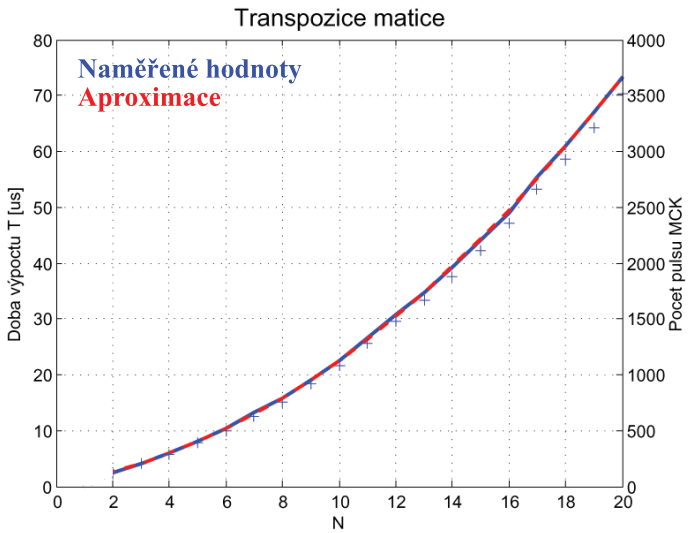


Obrázek 2.16: Výpočet součinu matic

Na obr. 2.16 je zachycen průběh charakteristiky pro součin matic. Z tvaru aproximačního vztahu 2.7 můžeme vidět mocninnou závislost nárůstu výpočetního času s rostoucím N.

$$T_{alg} = 3,2073N^{2,9466} \quad [\mu s] \qquad (2.7)$$

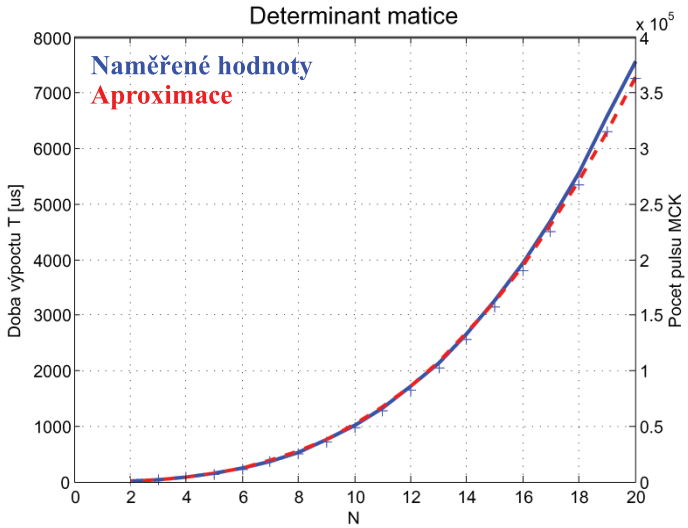
Srovnáním s ostatnými charakteristikami můžeme říct, že součin matic zejména vyšších řádů patří mezi časově náročné operace.



Obrázek 2.17: Operace transpozice

Výsledek měření pro operaci transpozice matice M zobrazuje obr. 2.8. Aproximační vztah má pro $N \geq 2$ tvar polynomu 2.řádu podle 2.8

$$T_{alg} = 0,143N^2 + 0,771N + 0.557 \qquad (2.8)$$



Obrázek 2.18: Výpočet determinantu

Mocninnou závislost podobně jako u operace součinu matic 2.18 můžeme vidět taky u výpočtu determinantu matice. Rovnice aproximačního vztahu má tvar 2.9

$$T_{alg} = 1,5786N^{2,8166} \quad [\mu s] \qquad (2.9)$$

Závěr

V úvodu druhé kapitoly jsme provedli základní přehled informací o embedded zařízeních pracujících v reálném čase a poukázali na několik aspektů návrhu takových zařízení. Pro realizaci řízení stabilizované základny byl v rámci projektu stabilizované základny vybrán mikrokontrolér AT91SAM7X256. Navrhli jsme všeobecnou koncepci měření výpočetních nároků algoritmů z hlediska času na svoleném mikrokontroléru. Koncepci jsme pomocí projektu ve vývojovém prostředí Embedded Workbench realizovali a pomocí naprogramovaných funkcí implementujících zvolené testovací algoritmy jsme zjistili časovou náročnost pro výpočet jednotlivých algoritmů, která poskytuje orientaci v nárocích řídících algoritmů vyšší úrovně. Konzultací s Tomášem Hanišem navrhujícím algoritmy řízení pro stabilizovanou plošinu jsme srovnáním s naměřenými výsledky předběžně zjistili vysoké výpočetní nároky algoritmů a vhodnost dalších testů pro ověření možnosti použití daného mikrokonroléru k řízení. Jak již bylo poznamenáno, rychlost výpočtu je závislá na množsví faktorů, které můžeme více či méne ovlivnit, jsme však limitování taktovací frekvencí mikrokontroléru. Řešením může být použití výkonnejšího mikrokontroléru nebo pridání další jednotky pro realizaci zejména maticových operací. Jako další směr v pokračování této práce je možné vylepšit testovací program (rozšíření přípustných rozsahů,automatické zpracování dat), provést portaci a následně testy časových nároků speciálních matematických knihoven komerčních produktů, které používají zejména pro rešení matic vysokých řádů sofistikovaných matematických metod.

Poděkování

Děkuji vedoucímu své bakalářské práce Ing. Martinovi Hromčíkovi, PhD. za pochopení a věnovaný čas během práce.

Literatura

[1] FRANKLIN G.F., POWELL J.D., EMAMI-NAEINI A.: *Feedback Control of Dynamic Systems.* 5th edn, Prentice-Hall. 2006, ISBN 0-13-149930-0.

[2] JOHN,J.: *Systémy a řízení.* Praha, Vydavatelství ČVUT 2003

[3] HRBÁČEK, J.: *Komunikace mikrokontroléru s okolím - 1.díl.* 1.vyd.Praha : BEN - technická literatura 1999, ISBN 80-86056-42-2

[4] ŠVARC,Ivan: *Teorie automatického řízení* 1.vyd.Brno: VUT 2003

[5] VÁŇA, Vladimír: *Mikrokontroléry ATMEL AVR* 1.vyd.Praha:BEN - technická literatura 2003, ISBN 80-7300-083-0

[6] MANN,Burkhard: *C pro mikrokontroléry* 1.vyd.Praha:BEN - technická literatura 2003, ISBN 3-7723-4154-3

[7] BARR,Michael: *Programming Embedded Systems* first edn, Sebastopol : O'Reilly&Associates 1999, ISBN 1-u56592-354-5

[8] ujc) HEROUT,Pavel: *Učebnice jazyka C - 1. díl* 4.vyd.České Budejovice:KOPP 2005, ISBN 80-7232-220-6

[9] BAYER, Jiří; ŠEBEK, Zdeněk; PÍŠA, Pavel; *Počítače pro řízení* 1.vyd.Praha:Vydavatelstvý ČVUT 2002

[10] HRBÁČEK, J.: *Komunikace mikrokontroléru s okolím - 1.díl.* 1.vyd.Praha : BEN - technická literatura 1999, ISBN 80-86056-42-2

[11] LIU, Jane W.S.: *Real-time systems* 1th edn, Prentice-Hall. 2000, ISBN 0-13-099651-3

[12] ftp://math.feld.cvut.cz/pub/olsak/linal

[13] www.dce.felk.cvut.cz

[14] www.control.aau.dk

[15] www.iar.com

[16] www.atmel.com

Seznam tabulek

Příloha A

Výpis zdrojových kódů

V této části se nachází výběr několika částí zdrojového kódu, zejména funkce realizující požadované operace. Kompletní a komentovaný zdrojový kód je k dispozici na přiloženém CD jako součást projektu v EW.

Funkce pro výpočet faktoriálu N

```
#if(TEST==1)
static float faktorial(float i) {
  register int j;
  if (i <= 1) return 1;
  for (j = i - 1; j >= 2; j--) i *= j;
  return i;
}
#endif
```

Funkce pro výpočet N-tého členu Fibonacciho posloupnosti

```
#if(TEST==2)
int fibonacci(int cislo){
int posledni;
register int i, pom, a=0, b=1;
for (i=1; i<(cislo-1); i++){
pom=a+b;
a=b;
b=pom;
}
posledni=b;
return posledni;
}
#endif
```

Funkce pro násobení matic

```
#if(TEST==3)
void soucinMatic(matrixx pA, matrixx pB, matrixx pC,int rad){
register int i,j,k;
for (i=0; i<=(rad-1);i++)
      for (j=0; j<=(rad-1);j++)
         for (k=0; k<=(rad-1); k++) pC[i][j]=pC[i][j]+(pA[i][k]*pB[k][j]);
```

```
}
#endif
```

Funkce pro výpočet determinantu matice

```
#if(TEST==4)
float vypocetDet(matrixx matice, int rad){
register int radek, sloupec,k;
float p=1;
  for (radek=1; radek<=(rad-1);radek++){
    for (sloupec=(radek+1); sloupec<=(rad);sloupec++){
        matice[(sloupec-1)][(radek-1)]=matice[(sloupec-1)]\
        \[(radek-1)]/matice[(radek-1)][(radek-1)];
        for (k=(radek+1); k<=(rad);k++){
        matice[(sloupec-1)][(k-1)]=matice[(sloupec-1)][(k-1)]\
        \-matice[(sloupec-1)][(radek-1)]*matice[(radek-1)][(k-1)];
        }
    p=p*matice[(radek-1)][(radek-1)];
    }
  p=p*matice[(rad-1)][(rad-1)];
  }
return p;
}
#endif
```

Funkce pro transpozici matice

```
#if(TEST==5)
static void tranMatic(matrixx matice, int rad){
float pom; register int i, j;
for (i=0; i<rad; i++)
  {
    for (j=i+1; j<rad; j++)
    {
      pom=matice[i][j];
      matice[i][j]=matice[j][i];
      matice[j][i]=pom;
    }
  }
}
#endif
```

Funkce pro přepočet hodnoty čítače na us

```
float convert(int count){
float us;
float konstanta_citace;
#if(KONSTANTA==0x0)
konstanta_citace=2;
#endif
```

```
#if(KONSTANTA==0x1)
konstanta_citace=8;
#endif
#if(KONSTANTA==0x2)
konstanta_citace=32;
#endif
#if(KONSTANTA==0x3)
konstanta_citace=128;
#endif
#if(KONSTANTA==0x4)
konstanta_citace=1024;
#endif
float uOSC = ((18432000*73/14)/2)/1000000;
us=((count*konstanta_citace)/uOSC);
return us;
}
```

Funkce pro zobrazeni zpravy

```
void report(int tik){
int konstanta_citace;

#if(KONSTANTA==0x0)
konstanta_citace=2;
#endif
#if(KONSTANTA==0x1)
konstanta_citace=8;
#endif
#if(KONSTANTA==0x2)
konstanta_citace=32;
#endif
#if(KONSTANTA==0x3)
konstanta_citace=128;
#endif
#if(KONSTANTA==0x4)
konstanta_citace=1024;
#endif

  printf("----------------zprava-----------------\n");
  printf("Hlavni frekvence systemu MCK: %d [Hz]\n",AT91B_MCK);
  printf("Konstanta citace K: %d [-]\n",konstanta_citace);
  printf("Stav citace: %d [MCK/K]\n",tik);
  printf("Vypocetni cas: %4f [us]\n",convert(tik));
  printf("----------------konec------------------\n");
}
```

Funkce pro výpis matice

```
static void vypisMatici(matrixx matice, int rad){
 register int i,j;
  for(i=0; i<rad; i++){
```

```
    for(j=0; j<rad; j++){
      printf("%4f  ",matice[i][j]);
    }
  printf(" \n");
  }
}
```

Funkce pro inicializaci časovače

```
void AT91F_TC_Open ( AT91PS_TC TC_pt, unsigned int Mode, unsigned int TimerId)
{
unsigned int dummy;
AT91F_PMC_EnablePeriphClock ( AT91C_BASE_PMC, 1<< TimerId ) ;
  TC_pt->TC_CCR = AT91C_TC_CLKDIS ;
TC_pt->TC_IDR = 0xFFFFFFFF ;
  dummy = TC_pt->TC_SR;
  dummy = dummy;
  TC_pt->TC_CMR = Mode ;
TC_pt->TC_CCR = AT91C_TC_CLKEN ;
}
```

Příloha B

Obsah přiloženého CD

K této práci je přiloženo CD, na kterém jsou uloženy následující data:

- BK_PDF - táto práce ve formátu pdf
- BK_TEX - zdrojové kódy pro TEX
- HELICOPTER - Sdílená data projektu RAMA
- MATLAB - soubory programu MATLAB
- ATMEL - dokumentace firmy ATMEL
- IAR - projekty v Embedded Workbench