UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

RYCHLOST IMPLEMENTACE ESTIMÁTORU STAVU V ZAŘÍZENÍ ARDUINO DUE

Jan Vojta

Diplomová práce

2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jan Vojta**
Osobní číslo:	**I16203**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Řízení procesů**
Název tématu:	**Rychlost implementace estimátoru stavu v zařízení Arduino Due**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Cíl: Zjistit možnosti generování zdrojového kódu jazyka C odpovídajícího uživatelské funkci MATLABu či modelu SIMULINku. Na příkladu diskrétního estimátoru stavu ověřit použitelnost získaného kódu v systému Arduino Due a experimentálně určit dobu provádění jednoho kroku estimace v závislosti na počtu stavů.
Teoretická část:
a) estimace stavu stavového modelu
b) diskrétní estimátor stavu stochastického systému
Praktická část:
a) koncept Arduino a vlastnosti zařízení Arduino Due
b) estimátor stavu jako uživatelská funkce v MATLABu v tvaru vhodném ke generování odpovídajícího zdrojového kódu funkce pro implementaci v programu Arduina
c) program pro Arduino Due realizující estimací stavu a umožňující měření doby provádění úseku kódu pomocí změn výstupních pinů
d) navrhnout a realizovat experiment, který umožní ověřit funkčnost estimátoru a změřit dobu provádění jednoho kroku výpočtu estimátoru

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

BALÁTĚ, Jaroslav. Automatické řízení. 2., přeprac. vyd. Praha: BEN, 2004, 663 s. ISBN 80-730-0148-9.
OGATA, Katsuhiko. Discrete-time control systems. 2nd ed. Englewood Cliffs, N.J.: Prentice Hall, c1995, xi, 745 p. ISBN 01-303-4281-5.
Arduino - Home [online]. (c)2015 [cit. 2015-10-06]. Dostupné z: https://www.arduino.cc/
Arduino - ArduinoDue. Arduino - Home [online]. (c)2015 [cit. 2015-10-06]. Dostupné z: https://www.arduino.cc/en/Guide/ArduinoDue

Vedoucí diplomové práce: **doc. Ing. František Dušek, CSc.**
Katedra řízení procesů

Datum zadání diplomové práce: **31. října 2017**
Termín odevzdání diplomové práce: **18. května 2018**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 8. listopadu 2017

Prohlášení

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 18. 5. 2018

Jan Vojta

Poděkování

Děkuji vedoucímu práce doc. Ing. Františku Duškovi, CSc., za příkladné vedení při tvorbě mé diplomové práce a spoustu hodin příjemně strávených nad její konzultací, mé trpělivé přítelkyni, která vždy stála za mnou s úsměvem a hrníčkem čaje, mým kamarádům a spolužákům, kteří mi připomínali, že vysoká škola není jen o studiu a hlavně mé báječné rodině, která mi tohle vše umožnila.

V Pardubicích dne 18. 5. 2018

Jan Vojta

ANOTACE

Práce se zabývá využitím zařízení Arduino Due a programu MATLAB pro řešení složitějších činností při realizaci řízení, konkrétně pro rozšíření úlohy měření veličin řízené soustavy o estimaci stavu s využitím matematického modelu. Byl proveden teoretický rozbor dané úlohy a rozbor možností implementace. Byl vypracován příslušný program v prostředích MATLAB a Arduino IDE, umožňující běh estimátoru v reálném čase s on-line zadáním parametrů v zařízení Arduino Due s využitím knihovních funkcí vygenerovaných programem MATLAB Coder. Funkčnost vypracovaného řešení byla ověřena experimentem na jednoduché reálné soustavě s měřitelnými stavy s následným vyhodnocením rychlosti implementace.

KLÍČOVÁ SLOVA

estimátor stavu, pozorovatel stavu, MATLAB, Arduino

TITLE

THE SPEED OF IMPLEMENTATION OF THE STATE ESTIMATOR IN THE DEVICE ARDUINO DUE

ANNOTATION

The thesis is focused on the issue of use of Arduino Due and MATLAB for solving more complicated tasks in the implementation of control, specifically for enhancing measurement of quantities of the controlled system with state estimation using mathematical model. Theoretical analysis was made of given assignment and implementation possibilities. An appropriate program was developed in MATLAB and Arduino IDE, which is used to real-time estimation with on-line parameters processing in Arduino Due using library functions generated by MATLAB Coder. Functionality of the developed solution is verified by experiment in simple real system with measurable states and with the speed of implementation subsequently\ evaluated.

KEYWORDS

State estimator, State observer, MATLAB, Arduino.

OBSAH

Seznam zkratek a značek 10
Seznam symbolů proměnných veličin a funkcí 12
Seznam ilustrací 14
Seznam tabulek 15
ÚVOD 16
1 CÍLE PRÁCE 17
2 MODELOVÁNÍ DYNAMICKÝCH PROCESŮ 19
2.1 Model 19
2.2 Matematický model 19
2.2.1 Lineární časově invariantní dynamický matematický model 20
2.3 Formy popisu matematického modelu 20
2.3.1 Stavový popis deterministického systému 21
2.3.2 Stavový popis stochastického systému 23
2.4 Tvorba modelu 24
2.4.1 Experimentální identifikace 24
2.4.2 Matematicko-fyzikální analýza 24
2.4.3 Kombinace uvedených metod 25
2.5 Diskretizace 25
2.5.1 Diskretizace stavového modelu 26
2.5.2 Diskrétní popis šumu 28
3 ESTIMACE STAVU 30
3.1 Pozorovatelnost 30
3.2 Stavový estimátor 31
3.2.1 Diskrétní estimátor úplného řádu 31
3.3 Určení zesílení estimátoru 33
3.3.1 Deterministický estimátor 33
3.3.2 Stochastický estimátor 35
3.4 Chyba estimace 37
4 MATLAB 38
4.1 Control System Toolbox 38
4.2 MATLAB Coder 38
4.2.1 Konverze datových typů 39

4.2.2 Omezení ... 40
4.2.3 Implementace knihovny do vlastního kódu ... 41
4.3 Funkce využitelné při návrhu estimátoru ... 42
4.4 Prostředky komunikace ... 43
4.4.1 Obsluha sériového portu ... 43
4.4.2 Přístup k souboru ... 44
4.4.3 GUI ... 45
5 ARDUINO ... 47
5.1 Filozofie a koncept ... 47
5.2 Hardware ... 47
5.2.1 Arduino Due ... 47
5.2.2 Srovnání desek Arduino ... 49
5.3 Software ... 50
5.3.1 Arduino Desktop IDE ... 50
5.3.2 Arduino Programming Language ... 51
5.3.3 Použité funkce ... 52
6 NÁVRH A PRŮBĚH EXPERIMENTU – HW ČÁST ... 53
6.1 Kalibrace vstupů a výstupů ... 53
6.2 RC soustava ... 55
6.2.1 Použité součástky ... 55
6.2.2 Zapojení ... 55
6.2.3 Odezva soustava ... 57
6.3 Tvorba stavového modelu ... 57
7 NÁVRH A PRŮBĚH EXPERIMENTU – SW ČÁST ... 60
7.1 Vygenerovaná knihovna ... 60
7.1.1 Funkce ardSoustavaRC ... 60
7.1.2 Funkce ardCon2Dis ... 61
7.1.3 Funkce ardPlace ... 61
7.1.4 Funkce ardPozorovatel ... 63
7.2 Program pro Arduino ... 63
7.3 MATLAB GUI ... 64
8 VÝSLEDKY MĚŘENÍ ... 66
8.1 Měření rychlosti ... 66
8.1.1 Měření doby výpočtu ... 66

8.1.2 Měření doby trvání I/O operací 67
8.2 Ověření správnosti estimace 68
8.3 Určení rychlosti implementace estimátoru stavu 69
9 ZÁVĚR 70
POUŽITÁ LITERATURA 71
PŘÍLOHY 73

SEZNAM ZKRATEK A ZNAČEK

A/D	analog/digitál
ADC	převodník analog/digitál
AI	analogový vstup
AO	analogový výstup
APL	Arduino Programming Language (programovací jazyk)
D/A	digitál/analog
DAC	převodník digitál/analog
DDR2	typ paměti
DI	digitální vstup
DO	digitální výstup
DPS	deska plošných spojů
EEPROM	typ paměti
Flash	typ paměti
FTTW	Rychlá Fourierova transformace
GPRS	systém mobilní komunikace
GSM	systém mobilní komunikace
GUI	grafické uživatelské rozhraní
HW	hardware
I/O	vstupně/výstupní
IDE	vývojové prostředí
IoT	Internet of Things (typ sítě)
LCD	displej z tekutých krystalů
LQ	lineárně kvadratický
LTI	lineární a časově invariantní
MEX	Matlab EXecutable (typ funkce)
PC	osobní počítač
PWM	pulzně šířková modulace
RAM	typ paměti
RC	článek tvořený odporem a kondenzátorem
RS-232	standard komunikace
RS-422	standard komunikace
RS-485	standard komunikace

SRAM	typ paměti
SW	software
USB	univerzální sériový port
ZOH	tvarovač nultého řádu

SEZNAM SYMBOLŮ PROMĚNNÝCH VELIČIN A FUNKCÍ

Proměnné modelu:

$\boldsymbol{A}$	matice dynamiky spojitého stavového modelu
$\boldsymbol{B}$	matice vstupu spojitého stavového modelu
$\boldsymbol{C}$	matice výstupu stavového modelu
$\boldsymbol{D}$	matice přímého přenosu stavového modelu
$\boldsymbol{H}$	matice zesílení estimátoru
$\boldsymbol{M}$	matice dynamiky diskrétního stavového modelu
$\boldsymbol{N}$	matice vstupu diskrétního stavového modelu
$\boldsymbol{N}_S$	vzájemná kovarianční matice šumu
$\boldsymbol{P}$	kovarianční matice chyby odhadu stavu
$\boldsymbol{P}_M$	matice pozorovatelnosti
$\boldsymbol{Q}$	volitelná matice váhy kvadratické plochy
$\boldsymbol{Q}_S$	kovarianční matice procesního šumu
$\boldsymbol{R}$	volitelná matice penalizace akčního zásahu
$\boldsymbol{R}_S$	kovarianční matice šumu měření
$\boldsymbol{T}_V$	perioda vzorkování
err_x	chyba estimace
n	řád systému
$\boldsymbol{p}$	vektor zvolených pólů estimátoru
$\boldsymbol{u}$	vektor vstupních veličin
$\boldsymbol{v}$	vektor bílého šumu měření
$\boldsymbol{w}$	vektor procesního bílého šumu
$\boldsymbol{x}$	vektor stavových veličin
$\boldsymbol{x}_E$	estimovaný vektor stavových veličin
$\boldsymbol{x}_S$	skutečný vektor stavových veličin
$\boldsymbol{y}$	vektor výstupních veličin
$\boldsymbol{y}_E$	estimovaný vektor výstupních veličin
$\boldsymbol{y}_S$	skutečný vektor výstupních veličin
$\Delta\boldsymbol{x}$	chyba odhadu stavového vektoru

Fyzikální proměnné:

C	elektrická kapacita, F

R	elektrický odpor, Ω
i	elektrický proud, A
k	čas v diskrétním vyjádření (číslo vzorku)
t	čas ve spojitém vyjádření, s
u	elektrické napětí, V

SEZNAM ILUSTRACÍ

Obr. 2.1 – Vnější a vnitřní model 21
Obr. 2.2 – Stavový model 22
Obr. 2.3 – Stavový model se šumem 23
Obr. 2.4 – Diskrétní stavový model 28
Obr. 3.1 – Diskrétní estimátor stavu 31
Obr. 3.2 – Diskrétní estimátor úplného řádu 32
Obr. 4.1 – Rozhraní programu MATLAB Coder 39
Obr. 4.2 – Okno nástroje GUIDE 45
Obr. 5.1 – Arduino Due 48
Obr. 5.2 – Manažér knihoven 51
Obr. 6.1 – Kalibrace A/D převodníku 53
Obr. 6.2 – Kalibrace D/A převodníku 54
Obr. 6.3 – Elektrické schéma soustavy 55
Obr. 6.4 – Zapojení Arduina s RC článkem 56
Obr. 6.5 – Fotodokumentace soustavy 56
Obr. 6.6 – Odezva soustavy 57
Obr. 7.1 – Upozornění programu MATLAB Coder 62
Obr. 7.2 – Výstup programu MATLAB Coder 63
Obr. 7.3 – Program v zařízení Arduino 64
Obr. 7.4 – MATLAB GUI 65
Obr. 8.1 – Ověření správnosti estimace 68
Obr. 8.2 – Průběh chyby estimace 69

SEZNAM TABULEK

Tab. 2.1 – Rozměry veličin stavového modelu 22
Tab. 4.1 – Datové typy programu MATLAB Coder 40
Tab. 4.2 – Funkce MATLAB pro návrh estimátoru 42
Tab. 4.3 – Funkce obsluhy sériového portu 44
Tab. 4.4 – Funkce pro práci s textovým souborem 45
Tab. 5.1 – Přehled desek Arduino 49
Tab. 5.2 – Specifické APL funkce 52
Tab. 6.1 – Omezení D/A převodníku 54
Tab. 6.2 – Hodnoty součástek 55
Tab. 8.1 – Výsledky měření rychlosti 66
Tab. 8.2 – Rychlost ADC a DAC 68

ÚVOD

S rozvojem a prudkým rozšířením malých jednodeskových počítačů, jako je například zařízení Arduino Due, vyvstává téma, zda by bylo možné využít tato relativně laciná a otevřená zařízení v oboru automatizace a regulace. No poli aplikací jednoduchých domácích úkonů se podobná zařízení zásluhou amatérů a nadšenců vyskytují již běžně, a i když k jejich nasazení v průmyslových podmínkách v brzké době zřejmě nedojde – svým návrhem a provedením k tomu nejsou ani určeny – použití pro složitější domácí aplikace, školní a laboratorní úlohy a vývoj nestojí v cestě žádná objektivní překážka. Jednou z možností uplatnění je i funkce stavového estimátoru, který je takřka nedílnou součástí stavového regulačního obvodu.

Stavový regulátor potřebuje mít ke své funkci informaci o aktuální hodnotě stavových veličin regulovaného systému. Tyto veličiny však často nejsou měřitelné, a tak je nutné použít stavový estimátor, který z informací o hodnotě vstupních a výstupních veličin a znalosti matematického modelu systému dokáže stavové veličiny rekonstruovat.

Pro návrh stavového estimátoru bylo vyvinuto několik postupů vycházejících z různých předpokladů a definicí žádaných vlastností. Všechny rozšířené postupy jsou součástí vybavení programu MATLAB jako funkce a jejich využití návrh značně usnadňuje. Protože funkce v jazyce MATLAB typicky není možné použít v jednodeskovém počítači přímo, je nutná jejich konverze do jazyka, který tato zařízení nativně podporují. K tomu je možné využít nástroj MATLAB Coder, který je volitelnou součástí programu MATLAB.

Cílem této práce je ověřit možnosti implementace stavového estimátoru na zařízení Arduino Due. Samotný kód estimátoru a další pomocné funkce budou napsány v jazyce MATLAB, následně konvertovány do jazyka C a implementovány do zařízení. Funkčnost řešení bude ověřena experimentem.

1 CÍLE PRÁCE

Cílem práce je ověřit možnosti implementace stavového estimátoru na zařízení Arduino Due. Pozornost je věnována zjištění hardwarových omezení této platformy a možnosti generování zdrojového kódu v jazyce C/C++ dle originální funkce vytvořené v prostředí MATLAB.

Z hardwarových možností platformy Arduino Due bude zkoumána zejména rychlost provádění kódu a v návaznosti na to minimální možná délka periody (resp. maximální frekvence vzorkování) diskrétního stavového estimátoru. Do této periody bude zahrnut nejen vlastní výpočet stavových veličin, ale i generování a měření průběhů vstupního a výstupního napětí a zpracování získaných hodnot.

Z hlediska softwaru bude ověřena možnost generování zdrojového kódu programem MATLAB Coder. Budou zkoumána omezení, která jsou uplatňována na zdrojovou funkci v jazyce MATLAB, a možnosti začlenění vygenerovaného kódu do vlastního programu. Bude provedena analýza jeho funkčnosti a vyhodnocena rychlost jeho provádění vztažená k volitelným parametrům programu MATALB Coder.

Pro experimentální ověření funkčnosti implementace stavového estimátoru bude nutné navrhnout a realizovat jednoduchý reálný systém se stavovými veličinami měřitelnými zařízením Arduino Due a vytvořit jeho stavový popis ve vhodném tvaru, který má jako stavové veličiny právě ty měřené.

Praktickým výstupem této práce budou tři softwary, které umožní realizovat experiment na zvoleném zařízení a vyhodnotit získaná data.

Knihovna v jazyce C/C++ vygenerovaná softwarem MATLAB Coder podle vzorových funkcí v jazyce MATLAB. Funkce z této knihovny budou realizovat:

- vytvoření spojitého stavového modelu soustavy dle zadaných parametrů,
- diskretizaci spojitého modelu dle zadané periody vzorkování,
- výpočet zesílení estimátoru zvolenou metodou,
- estimaci stavových veličin.

Program v jazyce Arduino Programming Language spouštěný na zařízení Arduino Due, který bude (s využitím výše uvedené knihovny) realizovat:

- příjem parametrů soustavy z PC,
- výpočet modelu a zesílení estimátoru,
- měření a generování vstupních a výstupních hodnot,

- výpočet odhadu stavových veličin,
- měření rychlosti odhadu
- odesílání získaných dat do PC.

Program v jazyce MATLAB s grafickým rozhraním pro komunikaci mezi uživatelem a zařízením Arduino Due který bude realizovat:

- zpracování a kontrolu parametrů zadaných uživatelem,
- odeslání zadaných parametrů do zařízení Arduino,
- přijetí dat z experimentu od zařízení Arduino,
- archivaci a vizualizaci získaných dat.

2 MODELOVÁNÍ DYNAMICKÝCH PROCESŮ

Model a modelování jsou pojmy, které mají nezastupitelnou úlohu ve všech moderních přístupech k řízení a regulaci. Při snaze o kvalitní regulaci reálné soustavy je vhodné mít představu, jak bude takový systém reagovat na vnější zásahy. Dobře vypracovaný model popisuje systém s dostatečnou přesností a v dosti široké oblasti na to, aby o systému podal nezbytné množství informací potřebných k určení vhodného akčního zásahu. Ať už je model využit k odhadu aktuálního stavu systému nebo k modelování jeho reakce na změnu řídící veličiny, jeho přínos pro zkvalitnění regulace je nesporný.

2.1 MODEL

Model lze obecně chápat jako reprezentaci reálného objektu vytvořenou s určitým záměrem a jistým zjednodušením. Míra zjednodušení závisí na záměru, kdy by vlastnosti modelu měly odrážet realitu co nejpřesněji v oblastech, které jsou klíčové a mohou být zjednodušené nebo zcela rozdílné v oblastech, které jsou pro záměr nepodstatné. Základní rozdělení modelů dle reprezentace je na modely fyzické a modely abstraktní.

Fyzické modely jsou takové, které hmatatelně existují v reálném světě. Mohou se lišit velikostí, materiálem a množstvím vyvedených detailů, nebo být založené na zcela jiném fyzikálním principu. Fyzikální model nachází využití tam, kde by vytvoření abstraktního modelu bylo s ohledem na požadovanou přesnost neúměrně složité.

Abstraktní model je takový, který existuje pouze jako představa o chování reálného objektu. Tento model může být vyjádřen obrazem, slovem i písmem ve formě pravidel, zákonitostí, diagramů a funkčních vztahů. V teorii řízení procesů je nejčastější vyjádření matematickými rovnicemi, z čehož plyne označení matematický model.

2.2 MATEMATICKÝ MODEL

Matematický model popisuje reálný objekt pomocí matematických vztahů. Je to struktura tvořená veličinami, které jsou interpretací parametrů reálného objektu, a vazbami mezi nimi. Tyto vazby jsou vyjádřeny algebraickými, diferenciálními a diferenčními rovnicemi, které popisují statické a dynamické vlastnosti systému.

Tento model má řadu praktických výhod, především jednoznačnost, snadnou reprodukovatelnost a jednoduchost práce s ním. Vyhodnocení experimentů na matematickém modelu dnes probíhá téměř výhradně na číslicových počítačích. To umožňuje provádět velké

množství experimentů a simulací, a to i takových, které by reálný objekt nebo fyzický model nenávratně poškodily, s vynaložením minimálních nákladů. Nevýhodou je složitost vytvoření takového modelu, která obvykle vyžaduje odborné znalosti a často neúměrně vzrůstá s požadovanou přesností výsledků.

2.2.1 Lineární časově invariantní dynamický matematický model

Lineární matematický model je vyjádřený pomocí lineárních rovnic. Pokud je jediná rovnice nelineární, je nelineární celý model. Jedním z hlavních rysů takového modelu je, že v něm platí princip superpozice (Ogata, 1995), z čehož je možné (časté, ne však nezbytné) vyvodit požadavek na lineární statickou charakteristiku, tedy aby graf závislosti vstupní a výstupní hodnoty po ustálení byl přímkou procházející počátkem souřadnic (Vítečková, 2016).

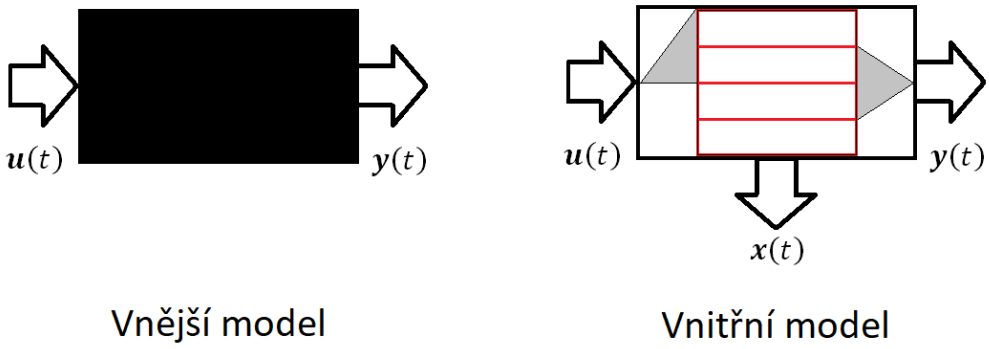
Časově invariantní (stacionární) matematický model má parametry, které jsou konstantní a časem se nemění. Tento model je popsán rovnicemi s konstantními koeficienty (Balátě, 2003).

Lineární časově invariantní matematický model je významný, protože pro něj existuje velký teoretický základ a spousta postupů v teorii řízení procesů počítá s linearitou a neměnností modelu. Proto je častá linearizace, tedy nahrazení modelu jeho lineární aproximací, a zanedbání malých změn parametrů v čase, ke kterým u reálné soustavy nutně postupně dochází. Tento model bývá označován jako LTI (Linear Time-Invariant).

Dynamický model je takový, jehož výstupy jsou závislé na stavu vstupních a výstupních veličin. Stavem veličiny jsou kromě aktuální hodnoty i její derivace u veličiny spojité, u veličiny diskrétní jsou to i její minulé hodnoty. Všechny modely uvažované v této práci jsou dynamické.

2.3 FORMY POPISU MATEMATICKÉHO MODELU

Během vývoje se ustálilo několik praktických a používaných forem popisu vazeb modelu, které lze rozdělit do dvou skupin na modely vnitřní a modely vnější. Obě tyto skupiny jsou určitým pohledem na modelovaný objekt. Lze mezi nimi převádět. Převod z vnitřního popisu na vnější je jednoznačný, ztrácí se však informace o vnitřních stavech. Převod vnějšího popisu na vnitřní jednoznačný není (vnitřních popisů lze touto cestou vytvořit nekonečně mnoho), vnitřní stavy tohoto modelu jsou zcela fiktivní a nemají fyzikální význam.



Obr. 2.1 – Vnější a vnitřní model

Vnější model se dívá na reálný objekt jako na závislost výstupních veličin na veličinách vstupních. Tento model je nejčastěji popsán pomocí jedné diferenciální rovnice vyššího řádu nebo obrazového přenosu. V praxi je nejběžnější a je široce používán při navrhování obvodů automatické regulace (Burý, 2011). Znázorněno na obr. 2.1 vlevo.

Vnitřní model popisuje vnitřní (stavové) veličiny modelovaného objektu, jejich vzájemné vazby a jejich vazby na veličiny vstupní a výstupní. Tento model bývá popsán pomocí soustavy obyčejných diferenciálních rovnic 1. řádu nebo ve formě stavového popisu. Znázorněno na obr. 2.1 vpravo.

2.3.1 Stavový popis deterministického systému

Stavový popis je formou popisu matematického modelu sestávající ze soustavy obyčejných diferenciálních rovnic prvního řádu a soustavy algebraických rovnic výstupu.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t), t], \tag{2.1}$$

$$\boldsymbol{y}(t) = \boldsymbol{g}[\boldsymbol{x}(t), \boldsymbol{u}(t), t], \tag{2.2}$$

kde $\boldsymbol{x}(t)$ je vektor stavových veličin,
$\boldsymbol{u}(t)$ je vektor vstupních veličin,
$\boldsymbol{y}(t)$ je vektor výstupních veličin.

Pro lineární časově invariantní model, jenž se používá nejčastěji, je obvyklý zápis pomocí maticového vyjádření

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A} \times \boldsymbol{x}(t) + \boldsymbol{B} \times \boldsymbol{u}(t), \tag{2.3}$$

$$\boldsymbol{y}(t) = \boldsymbol{C} \times \boldsymbol{x}(t) + \boldsymbol{D} \times \boldsymbol{u}(t), \tag{2.4}$$

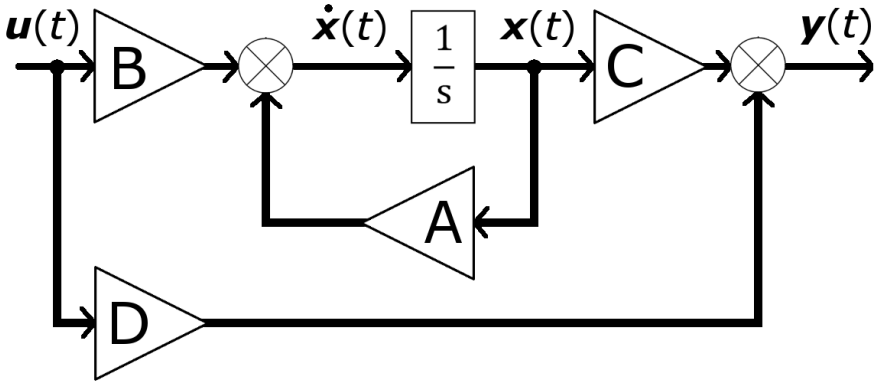
kde $\boldsymbol{A}$ je matice dynamiky,

$\boldsymbol{B}$ je matice vstupu,

$\boldsymbol{C}$ je matice výstupu,

$\boldsymbol{D}$ je matice přímého přenosu.

Spojitý stavový model lze zobrazit blokovým schématem na obr. 2.2.



Obr. 2.2 – Stavový model

Rozměry matic $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ a $\boldsymbol{D}$ a vektorů $\boldsymbol{u}$, $\boldsymbol{x}$ a $\boldsymbol{y}$ jsou určeny počtem vstupních a výstupních veličin a řádem systému. Je-li n řád systému, k je počet vstupních veličin a l je počet výstupních veličin, budou platit rozměry matic a vektorů uvedené v tab. 2.1.

Tab. 2.1 – Rozměry veličin stavového modelu

veličina	rozměry (*počet řádků* × *počet sloupců*)
$\boldsymbol{u}$	$k \times 1$
$\boldsymbol{x}$	$n \times 1$
$\boldsymbol{y}$	$l \times 1$
$\boldsymbol{A}$	$n \times n$
$\boldsymbol{B}$	$n \times k$
$\boldsymbol{C}$	$l \times n$
$\boldsymbol{D}$	$l \times k$

Maticemi $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ a $\boldsymbol{D}$ je model plně určen. Matice $\boldsymbol{A}$ určuje dynamiku modelu a její vlastní čísla odpovídají pólům soustavy. Matice $\boldsymbol{B}$ a $\boldsymbol{C}$ určují vztahy mezi vektorem vstupu, stavovým vektorem a vektorem výstupu. Matice $\boldsymbol{D}$ popisuje přímou vazbu mezi vstupem a výstupem, a protože taková vazba u většiny reálných systémů není nebo je zanedbatelná, bývá matice $\boldsymbol{D}$ zpravidla nulová a často se vynechává.

Stavový vektor $\boldsymbol{x}$ jednoznačně udává aktuální stav systému. Znalost stavu v čase $t = t_0$ a průběhu vstupu v čase $t \geq t_0$ úplně popisuje chování systému v čase $t \geq t_0$ (Ogata, 1995). Skládá se z n stavových proměnných, kde n je řád systému. Stavové proměnné obecně můžou, ale nemusejí mít fyzikální význam, vektory vstupu a výstupu ho však mají vždy (Balátě, 2003).

2.3.2 Stavový popis stochastického systému

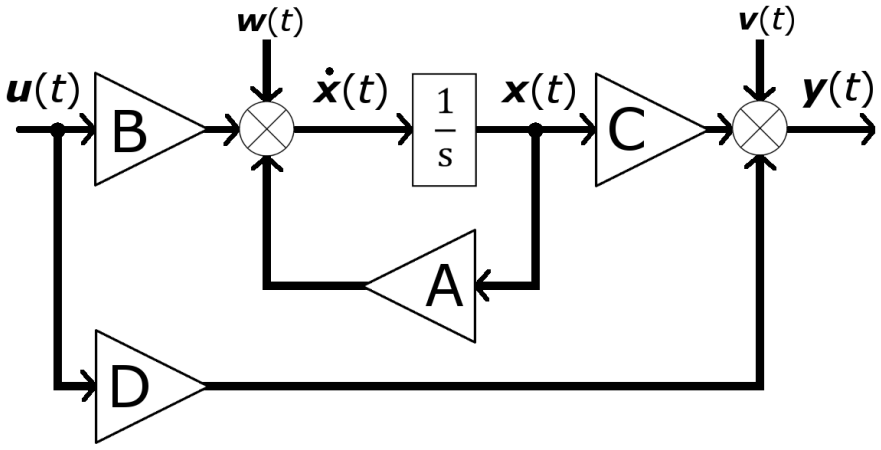
Pokud je naměřen aditivní šum na měřené veličině $\boldsymbol{y}(t)$, předpokládá se šum i na stavových veličinách $\boldsymbol{x}(t)$. K popisu stochastického LTI systému (LTI systému zatíženého šumem) se poté využívá upravený tvar popisu

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A} \times \boldsymbol{x}(t) + \boldsymbol{B} \times \boldsymbol{u}(t) + \boldsymbol{w}(t), \tag{2.5}$$

$$\boldsymbol{y}(t) = \boldsymbol{C} \times \boldsymbol{x}(t) + \boldsymbol{D} \times \boldsymbol{u}(t) + \boldsymbol{v}(t), \tag{2.6}$$

kde $\boldsymbol{w}$ je vektor procesního bílého šumu,
$\boldsymbol{v}$ je vektor bílého šumu měření.

Spojitý stavový model zatížený šumem lze zobrazit blokovým schématem na obr. 2.3.



Obr. 2.3 – Stavový model se šumem

Tyto šumy se zpravidla uvažují nekorelované, s nulovou střední hodnotou a konstantním rozptylem (Dušek, 2012).

2.4 TVORBA MODELU

Model lze vytvářet na základě analýzy nebo experimentu, což jsou dva krajní přístupy. V praxi se často využívá jejich kombinace, kdy je analýza upřesněna parametry zjištěnými experimentálním měřením (Základní postupy při modelování a identifikaci, 2006).

2.4.1 Experimentální identifikace

V průběhu vývoje teorie řízení procesů bylo navrženo mnoho způsobů experimentální identifikace a další jsou stále publikovány, všechny jsou však založeny na současném měření veličin, které jsou zvoleny jako vstupní a výstupní. Struktura modelu je založena na empirických zkušenostech.

Výhodou modelu sestaveného na základě experimentálních dat je relativní jednoduchost jeho vytvoření, která obecně nevyžaduje znalosti fyzikálního a funkčního pozadí identifikované soustavy. Nevýhodou je reálná platnost pouze v měřené oblasti a nezbytnost přístupu a provádění měření na modelovaném objektu.

Výsledný model postihuje závislost mezi vstupními a výstupními veličinami a případné stavové veličiny nemají konkrétní fyzikální význam, jedná se tedy „pouze“ o model chování. Přirozeným produktem experimentální identifikace je model vnější.

2.4.2 Matematicko-fyzikální analýza

Při matematiko-fyzikální analýze modelované soustavy se nejčastěji vychází z bilančních rovnic hmoty nebo energie. Veličiny mají konkrétní fyzikální význam a vztahy mezi nimi jsou často složité a nelineární. Je nutné nalézt kompromis mezi přesností a složitostí zanedbáním nepodstatných nebo málo výrazných vlastností (Vrožina, 2012).

Výhodou matematicko-fyzikálního modelu je možnost jeho vytvoření, i když skutečný objekt neexistuje, například během jeho projektování, a snadné rozšíření modelu o další sledované parametry. Oproti modelu vzniklému experimentální analýzou má takto vzniklý model platnost v celém rozsahu. Nevýhodou je celková náročnost vytvoření modelu, plynoucí z nutné hluboké znalosti příslušného oboru a potřebného matematického aparátu. Je také

nutné pamatovat, že matematický popis vyjádřený fyzikálními vztahy je pouze aproximací reality a nemusí tedy být vždy zcela přesný.

Při vytváření matematického modelu pomocí matematicko-fyzikální analýzy je velmi důležité určit, které vlastnosti systému jsou podstatné pro vytvoření takového modelu, jehož chování a parametry bude s dostatečnou přesností kopírovat chování reálného systému ve sledovaných oblastech z hlediska statických i dynamických vlastností. Zanedbáním vlastností, které toto chování ovlivňují jen minimálně nebo vůbec, se matematický model může velmi zjednodušit. Zjednodušení lze provést několika způsoby:

- redukcí, tedy zanedbáním některých proměnných,
- agregací, tedy nahrazením více podobných proměnných proměnnou jednou,
- změnou charakteru proměnné, například nahrazením proměnné konstantou,
- změnou charakteru závislostí, například lineární nebo kvadratickou aproximací,
- změnou omezujících podmínek, například omezením zkoumané oblasti (Burý, 2011).

Výsledkem je popis, který vystihuje reálnou strukturu zkoumaného objektu, včetně jeho vnitřních stavů. Přirozeným produktem matematicko-fyzikální analýzy je model vnitřní.

2.4.3 Kombinace uvedených metod

Pro nalezení velmi přesného modelu s širokou oblastí platnosti se využívá kombinace obou výše uvedených metod. Matematicko-fyzikální analýza je vhodná pro určení struktury modelu, jehož parametry jsou následně identifikovány pomocí několika opakovaných experimentů (Vrožina, 2012).

2.5 DISKRETIZACE

Pouze relativně malý počet reálných objektů je přirozeně popsán pomocí rovnic v diskrétním tvaru (např. modely z oblasti informatiky a logistiky). Naprostá většina reálných vztahů je však spojitých a proto je pro použití s číslicovou technikou nutné tyto vztahy (resp. modely) diskretizovat (Hlava, 2012), neboli převést rovnice diferenciální na rovnice diferenční.

Pro diskretizaci je nutné zvolit periodu vzorkování. Musí být dostatečně malá, aby vystihla chování systému, není však žádoucí, aby byla malá příliš, kvůli omezením reálných převodníků a nadměrnému zatěžování číslicového přístroje, který navzorkované hodnoty zpracovává.

Nejčastěji se používá diskretizace typu ZOH (zero-order hold), která předpokládá na vstupu soustavy tvarovač nultého řádu, který po celou dobu délky periody udržuje hodnotu nastalou na začátku této periody, tj. že hodnota vstupu je mezi intervaly vzorkování konstantní, tedy

$$\boldsymbol{u}(t) = \boldsymbol{u}(k \cdot T_{\mathrm{V}}) = \mathbf{konst} \tag{2.7}$$

pro

$$k \cdot T_{\mathrm{V}} \leq t \leq (k+1) \cdot T_{\mathrm{V}} \tag{2.8}$$

kde $\boldsymbol{u}$ je vektor výstupních veličin tvarovače nultého řádu,
t je spojitá časová veličina,
T_{V} je perioda vzorkování,
k je číslo vzorku.

Z uvedeného je patrné, že diskrétní popis reálného spojitého systému je vždy pouze aproximací, neboť se předpokládá důsledné dodržení vztahu (2.7) (Ogata, 1995). Při zavedení diskrétního popisu se často používá značení zjednodušující zápis, kdy se u časového údaje časově závislých proměnných vynechává perioda vzorkování, tedy

$$\boldsymbol{u}(k \cdot T_{\mathrm{V}}) \equiv \boldsymbol{u}(k) \text{ a} \tag{2.9}$$

$$\boldsymbol{u}((k+1) \cdot T_{\mathrm{V}}) \equiv \boldsymbol{u}(k+1). \tag{2.10}$$

2.5.1 Diskretizace stavového modelu

Spojitý stavový model popsaný rovnicemi (2.3) a (2.4) bude diskretizován s periodou T_{V}. Koeficienty výstupní rovnice (2.4) se zavedením vzorkování nezmění, neboť je tvořena pouze algebraickými rovnicemi, které nepopisují dynamickou část systému. Diskretizace se tedy týká pouze rovnice (2.3) popisující dynamiku stavového vektoru. Dochází při ní k náhradě matic $\boldsymbol{A}$ a $\boldsymbol{B}$ spojitého modelu za matice $\boldsymbol{M}$ a $\boldsymbol{N}$ modelu diskrétního. Rovnice popisující lineární časově invariantní diskrétní model budou mít tvar

$$\boldsymbol{x}(k+1) = \boldsymbol{M} \times \boldsymbol{x}(k) + \boldsymbol{N} \times \boldsymbol{u}(k), \tag{2.11}$$

$$\boldsymbol{y}(k) = \boldsymbol{C} \times \boldsymbol{x}(k) + \boldsymbol{D} \times \boldsymbol{u}(k), \tag{2.12}$$

kde $\boldsymbol{x}(k+1)$ je predikovaný vektor stavových veličin,
$\boldsymbol{x}(k)$ je aktuální vektor stavových veličin,

$\boldsymbol{u}(k)$ je vektor vstupu,

$\boldsymbol{y}(k)$ je vektor výstupu,

$\boldsymbol{M}$ je diskrétní matice dynamiky,

$\boldsymbol{N}$ je diskrétní matice vstupu,

$\boldsymbol{C}$ je matice výstupu,

$\boldsymbol{D}$ je matice přímého přenosu.

Samotný výpočet matic $\boldsymbol{M}$ a $\boldsymbol{N}$ vyžaduje nalezení řešení spojité stavové rovnice (2.3). Jeden z možných postupů (Hlava, 2012) spočívá v nalezení obecného řešení homogenní diferenciální rovnice

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A} \times \boldsymbol{x}(t), \tag{2.13}$$

jako

$$\boldsymbol{x}(t) = \mathrm{e}^{(\boldsymbol{A}\cdot t)} \times \boldsymbol{c}(t). \tag{2.14}$$

Úplné řešení se získá metodou variance konstanty, kde

$$\boldsymbol{c}(t) = \boldsymbol{x}(0) + \int_0^t [\mathrm{e}^{-(\boldsymbol{A}\cdot\tau)} \times \boldsymbol{B} \times \boldsymbol{u}(\tau)] \cdot \mathrm{d}\tau \tag{2.15}$$

a tedy

$$\boldsymbol{x}(t) = \mathrm{e}^{(\boldsymbol{A}\cdot t)} \times \boldsymbol{x}(0) + \mathrm{e}^{(\boldsymbol{A}\cdot \mathrm{t})} \times \int_0^t [\mathrm{e}^{-(\boldsymbol{A}\cdot\tau)} \times \boldsymbol{B} \times \boldsymbol{u}(\tau)] \cdot \mathrm{d}\tau, \tag{2.16}$$

kde po vyjádření pro daný interval t_1 až t_2

$$\boldsymbol{x}(t_2) = \mathrm{e}^{[\boldsymbol{A}\cdot(t_2 - t_1)]} \times \boldsymbol{x}(t_1) + \mathrm{e}^{[\boldsymbol{A}\cdot(t_2 - t_1)]} \times \int_{t_1}^{t_2} [\mathrm{e}^{-[\boldsymbol{A}\cdot(\tau - t_1)]} \times \boldsymbol{B} \times \boldsymbol{u}(\tau)] \cdot \mathrm{d}\tau, \tag{2.17}$$

lze při použití zjednodušeného zápisu dle (2.9) a (2.10) vyjádřit pro $T_\mathrm{V} = t_1 - t_2$ jako

$$\boldsymbol{x}(k) = \mathrm{e}^{(\boldsymbol{A}\cdot k\cdot T_\mathrm{V})} \times \boldsymbol{x}(0) + \mathrm{e}^{(\boldsymbol{A}\cdot k\cdot T_\mathrm{V})} \times \int_0^{(k\cdot T_\mathrm{V})} [\mathrm{e}^{-(\boldsymbol{A}\cdot\tau)} \times \boldsymbol{B} \times \boldsymbol{u}(\tau)] \cdot \mathrm{d}\tau, \tag{2.18}$$

z čehož lze vyjádřit přírůstek jako

$$\boldsymbol{x}(k+1) = \mathrm{e}^{(\boldsymbol{A}\cdot T_\mathrm{V})} \times \boldsymbol{x}(k) + \mathrm{e}^{(\boldsymbol{A}\cdot T_\mathrm{V})} \times \int_{k\cdot T_\mathrm{V}}^{(k+1)\cdot T_\mathrm{V}} [\mathrm{e}^{-[\boldsymbol{A}\cdot(\tau - k\cdot T_\mathrm{V})]} \times \boldsymbol{B} \times \boldsymbol{u}(\tau)] \cdot \mathrm{d}\tau. \tag{2.19}$$

Platí-li (2.7), integrální výraz lze upravit, provést substituci $\lambda = \tau - k \cdot T_\mathrm{V}$ a psát ve zjednodušené formě

$$\boldsymbol{x}(k+1) = \mathrm{e}^{(\boldsymbol{A}\cdot T_\mathrm{V})} \times \boldsymbol{x}(k) + \left\{ \int_0^{T_\mathrm{V}} [\mathrm{e}^{(\boldsymbol{A}\cdot\lambda)}] \cdot \mathrm{d}\lambda \times \boldsymbol{B} \right\} \times \boldsymbol{u}(k), \tag{2.20}$$

a tedy

$$\boldsymbol{M} = \mathrm{e}^{(\boldsymbol{A}\cdot T_\mathrm{V})}, \tag{2.21}$$

$$\boldsymbol{N} = \int_0^{T_\mathrm{V}} [\mathrm{e}^{(\boldsymbol{A}\cdot\lambda)}] \cdot \mathrm{d}\lambda \times \boldsymbol{B}, \tag{2.22}$$

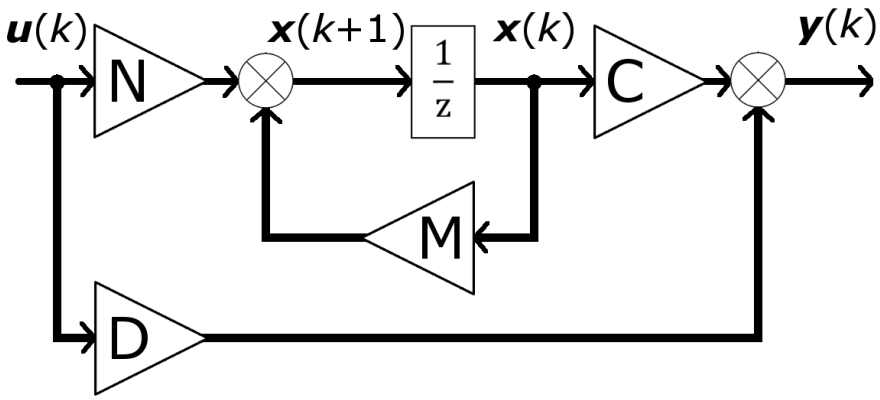
kdy integrál v (2.22) je možné (za splnění podmínek (2.7) a regulární matice $\boldsymbol{A}$) lehce vypočítat a vyjádřit matici $\boldsymbol{N}$ dle (Ogata, 1995) jako

$$\boldsymbol{N} = \boldsymbol{A}^{-1} \times \left(\mathrm{e}^{(\boldsymbol{A}\cdot T_\mathrm{V})} - \mathbf{I}\right) \times \boldsymbol{B} = \boldsymbol{A}^{-1} \times (\boldsymbol{M} - \mathbf{I}) \times \boldsymbol{B}. \tag{2.23}$$

Matice $\boldsymbol{M}$ a $\boldsymbol{N}$ musí být rozměrově shodné s maticemi $\boldsymbol{A}$ a $\boldsymbol{B}$.

Existuje více metod pro výpočet maticové exponenciály, např. rozklad exponenciály v nekonečnou Taylorovu řadu, který je obecný a lze použít i pro singulární matici $\boldsymbol{A}$. Výhodné je použití funkce prostředí MATLAB `expm()`.

Blokové schéma diskrétního stavového modelu je zobrazeno na obr. 2.4. Schéma diskrétního modelu je konceptuálně totožné se schématem modelu spojitého, pouze matice $\boldsymbol{A}$ a $\boldsymbol{B}$ jsou nahrazeny svými diskrétními ekvivalenty $\boldsymbol{M}$ a $\boldsymbol{N}$ a blok integrátoru je nahrazen blokem jednotkového zpoždění (zpoždění o T_V).



Obr. 2.4 – Diskrétní stavový model

2.5.2 Diskrétní popis šumu

Popis vlastností šumu v diskrétní oblasti je téměř totožný s popisem v oblasti spojité. Vlastnosti šumu jsou určeny příslušnými kovariančními maticemi, které jsou získány jako (při použitém značení (2.9))

$$\boldsymbol{Q}_\mathrm{S}(k) = \mathrm{E}\{\boldsymbol{w}(k) \times \boldsymbol{w}^\mathrm{T}(k)\}, \tag{2.24}$$

$$\boldsymbol{R}_{\mathrm{S}}(k) = \mathrm{E}\{\boldsymbol{v}(k) \times \boldsymbol{v}^{\mathrm{T}}(k)\}, \tag{2.25}$$

$$\boldsymbol{N}_{\mathrm{S}}(k) = \mathrm{E}\{\boldsymbol{w}(k) \times \boldsymbol{v}^{\mathrm{T}}(k)\}, \tag{2.26}$$

kde E je střední hodnota,

$\boldsymbol{Q}_{\mathrm{S}}$ je kovarianční matice procesního šumu,

$\boldsymbol{R}_{\mathrm{S}}$ je kovarianční matice šumu měření,

$\boldsymbol{N}_{\boldsymbol{S}}$ je vzájemná kovarianční matice šumu.

Opět zde platí předpoklad, že jsou šumy nekorelované a s nulovou střední hodnotou.

Odhad vlastností šumu (jeho kovariančních matic) je důležitý pro návrh stochastického estimátoru.

3 ESTIMACE STAVU

Pro stavové řízení je nutná znalost všech stavů. Ty jsou ale ne vždy měřitelné (je to nemožné, náročné, nepřesné, nákladné nebo nemá stavová veličina žádný fyzikální význam) a je tedy nutné stavy odhadovat. K odhadu (estimaci) aktuálního stavu slouží stavový estimátor, který na základě znalosti matematického modelu soustavy, vstupních a výstupních veličin, případně i některých měřených stavů (poté je označován jako redukovaný estimátor), dokáže tento odhad určit.

V ideálním případě by k rekonstrukci stavu stačil model s informací o vstupních hodnotách. V reálném případě na systém působí neměřitelné vnější vlivy (poruchy), nelze jednoznačně odhadnout počáteční stav systému, a ani model není úplně přesný, použití samotného stavového modelu by tedy nezaručilo správnost odhadu (Dušek, 2012). Kompenzace těchto vlivů se provádí přičtením vážené hodnoty rozdílu měřeného výstupu systému a vypočteného výstupu estimátoru. Matice vah se nazývá matice pozorovatele (Turek, 2007).

3.1 POZOROVATELNOST

Aby bylo možné stav odhadnout, musí být systém pozorovatelný. Výstup soustavy musí být (alespoň nepřímou vazbou mezi nimi samotnými) závislý na všech stavových veličinách a tyto veličiny musí být závislé na veličinách vstupních.

K určení pozorovatelnosti stavu lineárního časově invariantního stavového modelu se sestavuje matice pozorovatelnosti dle vztahu

$$\boldsymbol{P}_{\mathrm{M}} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C} \times \boldsymbol{M} \\ \boldsymbol{C} \times \boldsymbol{M}^2 \\ \vdots \\ \boldsymbol{C} \times \boldsymbol{M}^{n-1} \end{bmatrix} \tag{3.1}$$

kde $\boldsymbol{P}_{\mathrm{M}}$ je matice pozorovatelnosti,

$\boldsymbol{M}$ je matice dynamiky diskrétního stavového modelu,

$\boldsymbol{C}$ je matice výstupu stavového modelu,

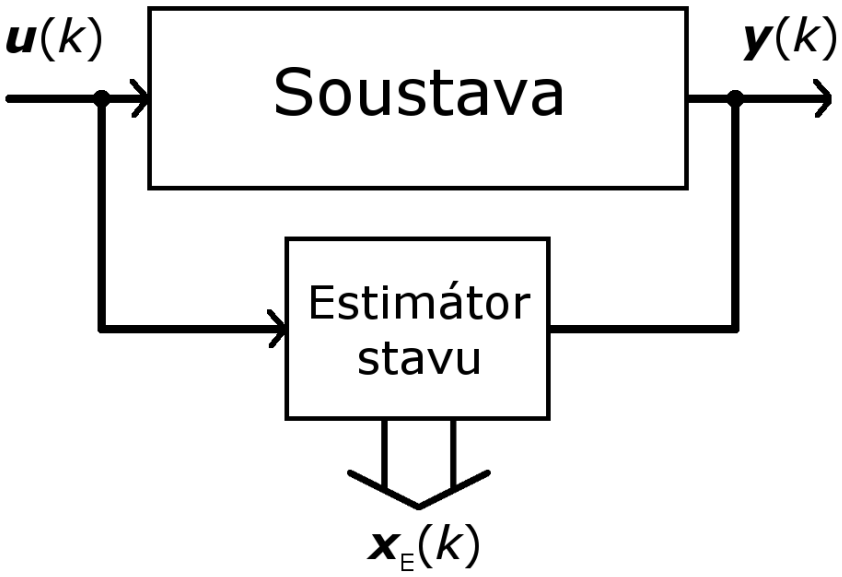
n je řád systému.

Má-li matice pozorovatelnosti stejnou hodnost, jako je řád systému (rozměr čtvercové matice $\boldsymbol{M}$), je stavový vektor pozorovatelný a je možné jej estimovat. Tato podmínka je nutná a zároveň postačující (Ogata, 1995).

Stavový model vzniklý experimentální identifikací nebo převodem z modelu vstupně výstupního je pozorovatelný vždy. V případě modelu vzniklého matematicko-fyzikální analýzou nebo neznámého původu je ověření pozorovatelnosti vhodné.

3.2 STAVOVÝ ESTIMÁTOR

Také stavový pozorovatel a rekonstruktor stavu. Žádanou funkcí estimátoru stavu je odhadovat stavový vektor $\boldsymbol{x}_E$, který bude v ideálním případě identický s vektorem stavů reálné soustavy $\boldsymbol{x}_S$ a kteréžto shody dosáhne v konečném čase nezávisle na počátečním odhadu, pokud je systém pozorovatelný (Dušek, 2012). Princip estimátoru je znázorněný na obr. 3.1.



Obr. 3.1 – Diskrétní estimátor stavu

Podle počtu odhadovaných stavů lze rozlišovat estimátory úplného a redukovaného řádu. Estimátor úplného řádu rekonstruuje všechny vnitřní stavy pozorované soustavy na základě stavového modelu soustavy, a má tedy stejný řád jako soustava. Estimátor redukovaného řádu rekonstruuje jen ty stavy, které nejsou měřitelné, využívá tedy upravený model nižšího řádu.

3.2.1 Diskrétní estimátor úplného řádu

Principiálně je estimátor stavový model soustavy doplněný o korekci na základě rozdílu měřených a odhadovaných výstupních veličin. Funkci diskrétního estimátoru je možné popsat rovnicemi

$$\boldsymbol{x}_{\mathrm{E}}(k+1) = \boldsymbol{M} \times \boldsymbol{x}_{\mathrm{E}}(k) + \boldsymbol{N} \times \boldsymbol{u}(k) + \boldsymbol{H} \times [\boldsymbol{y}_{\mathrm{S}}(k) - \boldsymbol{y}_{\mathrm{E}}(k)], \tag{3.2}$$

$$\boldsymbol{y}_{\mathrm{E}}(k) = \boldsymbol{C} \times \boldsymbol{x}_{\mathrm{E}}(k) + \boldsymbol{D} \times \boldsymbol{u}(k), \tag{3.3}$$

kdy po dosazení rovnice (3.3) do (3.2) lze dojít do tvaru

$$\boldsymbol{x}_{\mathrm{E}}(k+1) = (\boldsymbol{M} - \boldsymbol{H} \times \boldsymbol{C}) \times \boldsymbol{x}_{\mathrm{E}}(k) + (\boldsymbol{N} - \boldsymbol{H} \times \boldsymbol{D}) \times \boldsymbol{u}(k) + \boldsymbol{H} \times \boldsymbol{y}_{\mathrm{S}}(k), \tag{3.4}$$

kde $\boldsymbol{x}_{\mathrm{E}}(k+1)$ je odhadovaná predikce stavového vektoru,

$\boldsymbol{x}_{\mathrm{E}}(k)$ je odhadovaný aktuální stavový vektor,

$\boldsymbol{u}(k)$ je měřený vektor vstupu,

$\boldsymbol{y}_{\mathrm{S}}(k)$ je měřený vektor výstupu pozorované soustavy,

$\boldsymbol{y}_{\mathrm{E}}(k)$ je odhadovaný vektor výstupu estimátoru,

$\boldsymbol{M}$ je matice dynamiky,

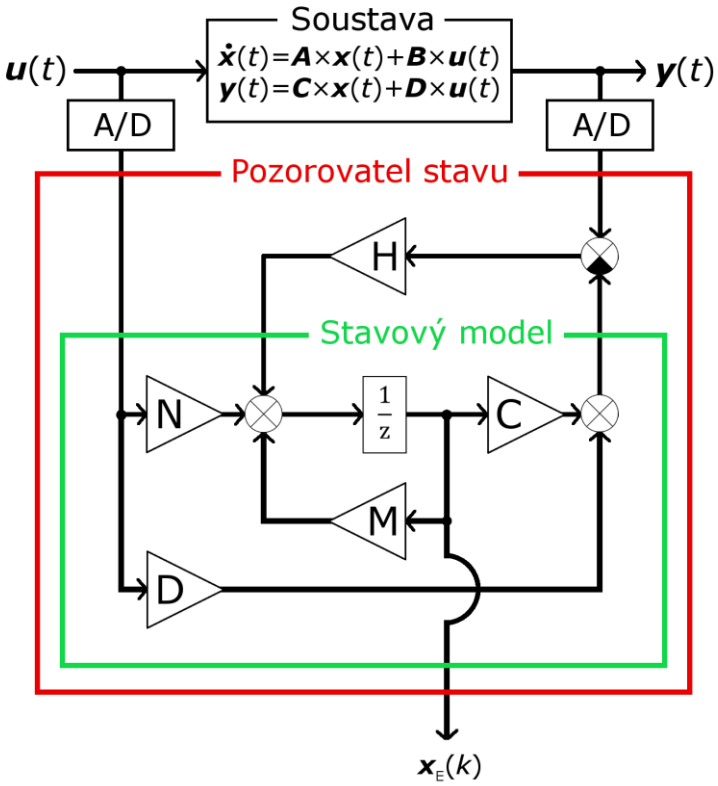
$\boldsymbol{N}$ je matice vstupu,

$\boldsymbol{C}$ je matice výstupu,

$\boldsymbol{D}$ je matice přímého přenosu,

$\boldsymbol{H}$ je matice zesílení estimátoru.

Blokové schéma zapojení diskrétního estimátoru úplného řádu je na obr. 3.2.



Obr. 3.2 – Diskrétní estimátor úplného řádu

Matice $\boldsymbol{M}$, $\boldsymbol{N}$, $\boldsymbol{C}$ a $\boldsymbol{D}$ jsou matice stavového modelu, úloha návrhu estimátoru tedy spočívá v určení váhové matice zesílení $\boldsymbol{H}$. Je více možností jak tuto matici určit, struktura estimátoru však zůstává stejná.

3.3 URČENÍ ZESÍLENÍ ESTIMÁTORU

Fundamentálním požadavkem na estimátor je, aby byl stabilní a, pokud je to možné, reagoval rychleji než pozorovaný systém. Po definování chyby estimace jako

$$\Delta\boldsymbol{x}(k) = \boldsymbol{x}_{\mathrm{S}}(k) - \boldsymbol{x}_{\mathrm{E}}(k), \tag{3.5}$$

lze po dosazení rovnic (2.11) a (3.2) vyjádřit vývoj chyby

$$\Delta\boldsymbol{x}(k+1) = (\boldsymbol{M} - \boldsymbol{H} \times \boldsymbol{C}) \times \Delta\boldsymbol{x}(k), \tag{3.6}$$

kde $\Delta\boldsymbol{x}(k+1)$ je predikce chyby odhadu stavového vektoru,
$\Delta\boldsymbol{x}(k)$ je aktuální chyba odhadu stavového vektoru,
$\boldsymbol{x}_{\mathrm{S}}(k)$ je stavový vektor pozorované soustavy,
$\boldsymbol{x}_{\mathrm{E}}(k)$ je odhad stavového vektoru,

z čehož vyplývá, že vývoj chyby odhadu závisí pouze na maticích $\boldsymbol{M}$, $\boldsymbol{C}$ a $\boldsymbol{H}$.

Matice $\boldsymbol{M}$ a $\boldsymbol{C}$ jsou matice stavového modelu, a jsou tedy dané, tudíž jediným volitelným parametrem je matice $\boldsymbol{H}$. Volbou matice $\boldsymbol{H}$ lze ovlivnit vlastní čísla matice $(\boldsymbol{M} - \boldsymbol{H} \times \boldsymbol{C})$, která jsou zároveň póly estimátoru, a tím i dynamiku vývoje chyby. Poloha pólů tedy určuje rychlost konvergence odhadu ke správné hodnotě. Aby chyba odhadu konvergovala k nule (aby byla estimace stabilní) pro libovolnou počáteční podmínku, póly diskrétního estimátoru musí ležet uvnitř jednotkové kružnice (v levé části komplexní roviny pro spojitý estimátor) (Turek, 2007).

Existuje více metod určení matice $\boldsymbol{H}$, které zaručují splnění podmínky stability estimace, liší se však ve formulaci dalších požadavků, které jsou na estimátor kladeny a vlastní metodě výpočtu. Matice $\boldsymbol{H}$ může být konstantní během celé doby funkce estimátoru (poté se označuje za ustálené zesílení) nebo se může průběžně měnit (adaptivní zesílení), což je možné využít při návrhu estimátoru pro soustavu obsahující proměnlivý aditivní šum.

3.3.1 Deterministický estimátor

Pokud není měřený výstupní vektor pozorované soustavy $\boldsymbol{y}_{\mathrm{S}}(k)$ zatížený aditivním šumem vůbec nebo je šum zanedbatelný, nepředpokládá se šum ani na vektoru stavových

veličin $\boldsymbol{x}_S(k)$ a k návrhu estimátoru lze využít postupy vycházející z deterministického stavového modelu.

Úloha určení zesílení deterministického estimátoru může být také formulována jako duální úloha k návrhu zesílení stavového regulátoru, což znamená, že pro návrh estimátoru lze využít všechny metody určené k výpočtu zesílení zpětné vazby regulátoru po substituci dvojice matic ($\boldsymbol{M}$,$\boldsymbol{N}$) dvojicí ($\boldsymbol{M}^T$,$\boldsymbol{C}^T$) (Dušek, 2012).

Metoda umístění pólů

Tato metoda (umístění pólů, pole placement, pole assigment) umožňuje určit matici zesílení estimátoru tak, aby póly estimátoru ležely ve zvolených bodech komplexní roviny. Obecně u metod tohoto typu platí, že

$$\boldsymbol{H} = f(\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{p}) \tag{3.7}$$

kde $\boldsymbol{p}$ je sloupcový vektor zvolených pólů estimátoru délky řádu systému.

Póly estimátoru se volí vždy uvnitř jednotkové kružnice a obvykle několikanásobně „rychlejší“ (tedy blíže k nule), než póly regulátoru (estimace se nejčastěji provádí z důvodu regulace), aby póly regulátoru byly dominantní a estimátor nezpomaloval dynamiku (Šebek, 2016). Jedním z možných přístupů k volbě umístění pólů diskrétního pozorovatele je požadavek, aby všechny póly pozorovatele ležely v komplexní rovině blíže ke středu než všechny póly soustavy (více vlevo v komplexní rovině pro spojitého pozorovatele (Vítečková, 2016)). Jsou-li zvoleny všechny póly nulové, estimovaný stavový vektor konverguje ke správným hodnotám velmi rychle (estimátor je poté tzv. konečný a minimální), je však velice citlivý i na malý šum, který výrazně zesiluje.

Funkce f musí zajistit polohu pólů estimátoru přímo nebo co nejblíže k $\boldsymbol{p}$. Jednou z nejčastěji využívaných funkcí je např. Ackermannova formule. Pro úlohu umístění pólů lze s výhodou využít implementované funkce prostředí MATLAB, které tento problém řeší, např. funkce `acker()` a `place()`.

LQ návrh

Tato metoda určí zesílení estimátoru tak, aby došlo k přechodu z počátečního do správného stavu způsobem, který minimalizuje kritérium

$$K(\boldsymbol{H}) = \sum_{k=1}^{\infty}[\boldsymbol{x}^T(k) \times \boldsymbol{Q} \times \boldsymbol{x}(k) + \boldsymbol{y}^T(k) \times \boldsymbol{R} \times \boldsymbol{y}(k)], \text{ kde} \tag{3.8}$$

$$\boldsymbol{y}(k) = -\boldsymbol{H} \times \boldsymbol{x}(k), \tag{3.9}$$

kde *K* je hodnota kritéria,

$\boldsymbol{Q}$ je volitelná matice váhy kvadratické plochy,

$\boldsymbol{R}$ je volitelná matice penalizace akčního zásahu.

Obecně lze vyjádřit, že

$$\boldsymbol{H} = f(\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{Q}, \boldsymbol{R}). \tag{3.10}$$

Matice $\boldsymbol{Q}$ a $\boldsymbol{R}$ jsou volené a jsou chápány jako relativní váhy k určení důležitosti vlivu $\boldsymbol{x}(k)$ nebo $\boldsymbol{y}(k)$ na hodnotu kritéria. $\boldsymbol{Q}$ i $\boldsymbol{R}$ musí být pozitivně semidefinitní (Štecha, 1999).

Funkce f hledá matici zesílení $\boldsymbol{H}$ tak, aby hodnota kritéria byla minimální. Jednou z možností určení konstantní hodnoty $\boldsymbol{H}$ je nalezení ustáleného řešení Riccatiho rovnice. Pro úlohu LQ návrhu lze využít implementované funkce prostředí MATLAB, např. funkci `dlqr()`.

3.3.2 Stochastický estimátor

Pokud je měřený výstupní vektor pozorované soustavy $\boldsymbol{y}_S(k)$ zatížen aditivním šumem, lze šum přepokládat i na vektoru stavových veličin $\boldsymbol{x}_S(k)$ a je vhodné toto zohlednit a uvažovat statistické vlastnosti šumu jako další parametr při návrhu estimátoru.

Kalmanův filtr

Je-li systém popsán stochastickým stavovým modelem a statistické vlastnosti šumu se v čase nemění (kovarianční matice šumu $\boldsymbol{Q}_S$, $\boldsymbol{R}_S$ a $\boldsymbol{N}_S$ jsou konstantní), je vhodné použít estimátor, který se označuje jako Kalmanův filtr. Návrh tohoto estimátoru vede k minimalizaci střední kvadratické chyby odhadu stavu v nekonečném horizontu. Tuto chybu lze vyjádřit jako

$$\boldsymbol{P}(\boldsymbol{H}) = \lim_{k\to\infty} \mathrm{E}\{[\boldsymbol{x}_S(k) - \boldsymbol{x}_E(k)] \times [\boldsymbol{x}_S(k) - \boldsymbol{x}_E(k)]^T\}, \tag{3.11}$$

kde $\boldsymbol{P}$ je kovarianční matice chyby odhadu stavu v nekonečném horizontu.

Snahou je najít takovou matici zesílení $\boldsymbol{H}$, která povede k minimalizaci střední hodnoty stopy (součtu diagonálních prvků) matice $\boldsymbol{P}$ (Dušek, 2012), tedy

$$\boldsymbol{H} = \arg\min_{\boldsymbol{H}} \; tr\{\boldsymbol{P}(\boldsymbol{H})\} \tag{3.12}$$

Obecně lze úlohu formulovat jako

$$\boldsymbol{H} = f(\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{Q}_{\mathrm{S}}, \boldsymbol{R}_{\mathrm{S}}, \boldsymbol{N}_{\mathrm{S}}) \tag{3.13}$$

kde $\boldsymbol{Q}_{\mathrm{S}}$ je kovarianční matice procesního šumu dle (2.24),
$\boldsymbol{R}_{\mathrm{S}}$ je kovarianční matice šumu měření dle (2.25),
$\boldsymbol{N}_{\mathrm{S}}$ je vzájemná kovarianční matice šumu dle (2.26).

Pro nalezení řešení úlohy lze využít implementované funkce prostředí MATLAB, např. funkci `kalman()`.

Adaptivní Kalmanův filtr.

Je-li systém popsán stochastickým stavovým modelem a kovarianční matice šumu $\boldsymbol{Q}_{\mathrm{S}}$, $\boldsymbol{R}_{\mathrm{S}}$ a $\boldsymbol{N}_{\mathrm{S}}$ se v čase mění, je vhodné použít estimátor, který se označuje jako Adaptivní Kalmanův filtr. Tento estimátor se od všech výše uvedených liší v tom, že matice zesílení estimátoru $\boldsymbol{H}$ není konstantní, ale upravuje se v každém kroku na základě nového odhadu vlastností šumu.

Návrh tohoto estimátoru vede k minimalizaci aktuální střední kvadratické chyby odhadu stavu. Tuto chybu lze vyjádřit jako

$$\boldsymbol{P}_{\mathrm{k}}(\boldsymbol{H}_{\mathrm{k}}) = \mathrm{E}\{[\boldsymbol{x}_{\mathrm{S}}(k) - \boldsymbol{x}_{\mathrm{E}}(k)] \times [\boldsymbol{x}_{\mathrm{S}}(k) - \boldsymbol{x}_{\mathrm{E}}(k)]^{\mathrm{T}}\}, \tag{3.14}$$

kde $\boldsymbol{P}_{\mathrm{k}}$ je aktuální kovarianční matice chyby odhadu stavu,
$\boldsymbol{H}_{\mathrm{k}}$ je aktuální matice zesílení pozorovatele,

kdy snahou je opět minimalizovat střední hodnotu stopy (součtu diagonálních prvků) matice $\boldsymbol{P}_{\mathrm{k}}$ (Dušek, 2012). V popisu algoritmu adaptivního Kalmanova filtru je pro přehlednost použito značení

$$\boldsymbol{y}(a|b) = f[\boldsymbol{x}(a|b)], \tag{3.15}$$

kde a je číslo vzorku proměnné,
b je poslední číslo vzorku, který ovlivnil hodnotu proměnné.

Logicky zde platí $a \geq b$. Algoritmus dle (Dušek, 2012) a (Burešová, 2017) popsaný rovnicemi (3.16) až (3.20) je možné rozdělit do dvou kroků, kdy prvním krokem je predikce a

druhým krokem je korekce. Uvažuje se model s nulovou maticí $\boldsymbol{D}$, která tím může být zcela vypuštěna.

Predikce:

$$\boldsymbol{x}_{\mathrm{E}}(k+1|k) = \boldsymbol{M} \times \boldsymbol{x}_{\mathrm{E}}(k|k) + \boldsymbol{N} \times \boldsymbol{u}(k), \tag{3.16}$$

$$\boldsymbol{P}(k+1|k) = \boldsymbol{M} \times \boldsymbol{P}(k|k) \times \boldsymbol{M}^{\mathrm{T}} + \boldsymbol{Q}_{\mathrm{S}}(k), \tag{3.17}$$

Korekce:

$$\boldsymbol{H}(k+1) = \boldsymbol{P}(k+1|k) \times \boldsymbol{C}^{\mathrm{T}} \times [\boldsymbol{C} \times \boldsymbol{P}(k+1|k) \times \boldsymbol{C}^{\mathrm{T}} + \boldsymbol{R}_{\mathrm{S}}(k+1)]^{-1}, \tag{3.18}$$

$$\boldsymbol{P}(k+1|k+1) = \boldsymbol{P}(k+1|k) - \boldsymbol{H}(k+1) \times \boldsymbol{C} \times \boldsymbol{P}(k+1|k). \tag{3.19}$$

$$\boldsymbol{x}_{\mathrm{E}}(k+1|k+1) = \boldsymbol{x}_{\mathrm{E}}(k+1|k) + \boldsymbol{H}(k+1) \times [\boldsymbol{y}_{\mathrm{S}}(k) - \boldsymbol{C} \times \boldsymbol{x}_{\mathrm{E}}(k+1|k)], \tag{3.20}$$

Hodnotu $\boldsymbol{x}_{\mathrm{E}}(k+1|k+1)$ lze již považovat za správnou.

3.4 CHYBA ESTIMACE

Jednou z objektivních možností určení chyby odhadu stavových veličin je výpočet průběžné kvadratické chyby (Dušek, 2012) dle vztahu

$$err\_x(k) = [\boldsymbol{x}_{\mathrm{S}}(k) - \boldsymbol{x}_{\mathrm{E}}(k)]^{\mathrm{T}} \times [\boldsymbol{x}_{\mathrm{S}}(k) - \boldsymbol{x}_{\mathrm{E}}(k)], \tag{3.21}$$

kde $err\_x(k)$ je aktuální kvadratická chyba odhadu,

$\boldsymbol{x}_{\mathrm{S}}(k)$ je měřený stavový vektor pozorované soustavy,

$\boldsymbol{x}_{\mathrm{E}}(k)$ je vypočtený odhad stavového vektoru.

Pro správně navržený estimátor bude tato chyba klesat od počátečních vysokých hodnot (kde se projeví rozdíl skutečného stavu soustavy s počátečním odhadem stavového vektoru v prvních krocích estimace) k nule. Chyba se může v průběhu i zvýšit, dojde-li k rychlé změně hodnot vektoru vstupních veličin a hodnoty estimovaného stavového vektoru budou na tuto změnu reagovat s lehkým zpožděním proti skutečným.

Předpokladem pro možnost tohoto výpočtu je znalost hodnot skutečných stavů.

4 MATLAB

MATLAB je interaktivní programové prostředí a skriptovací programovací jazyk určený pro analýzu dat, vývoj algoritmů, modelovaní a vizualizaci. Jeho hlavním specifikem jsou matice jako základní datový typ a přirozený přístup k práci s nimi, díky čemuž je určen primárně pro vědecké a technické výpočty. Všechny proměnné vytvořené během provádění skriptů jsou uloženy v prostoru MATLAB Workspace, jehož obsah je udržován aktuální a přístupný k zobrazení. Skripty a funkce je možné zapsat a uložit do souborů s příponou .m, což jsou textové soubory obsahující zadané posloupnosti příkazů.

Už v základní verzi obsahuje MATLAB velké množství funkcí profesionálně a efektivně realizujících často používané algoritmy a tuto výbavu lze dále znatelně rozšířit zakoupením a doinstalováním některé z četných oborově zaměřených aplikačních knihoven (toolboxů), které jsou momentálně k dispozici.

4.1 CONTROL SYSTEM TOOLBOX

Control System Toolbox je jedním z toolboxů prostředí MATLAB, který doplňuje jeho vybavení o nástroje z oblastí teorie systémů a řídicí techniky (Humusoft, 2018).

Součástí jsou funkce a vizualizační nástroje, které umožňují popsat systém více způsoby (přenosové funkce, stavové modely, pomocí pólů a nul atd.). Vlastnosti těchto systémů (přechodové a frekvenční charakteristiky, polohu pólu a nul v komplexní rovině apod.) je možné lehce vyhodnotit a zobrazit.

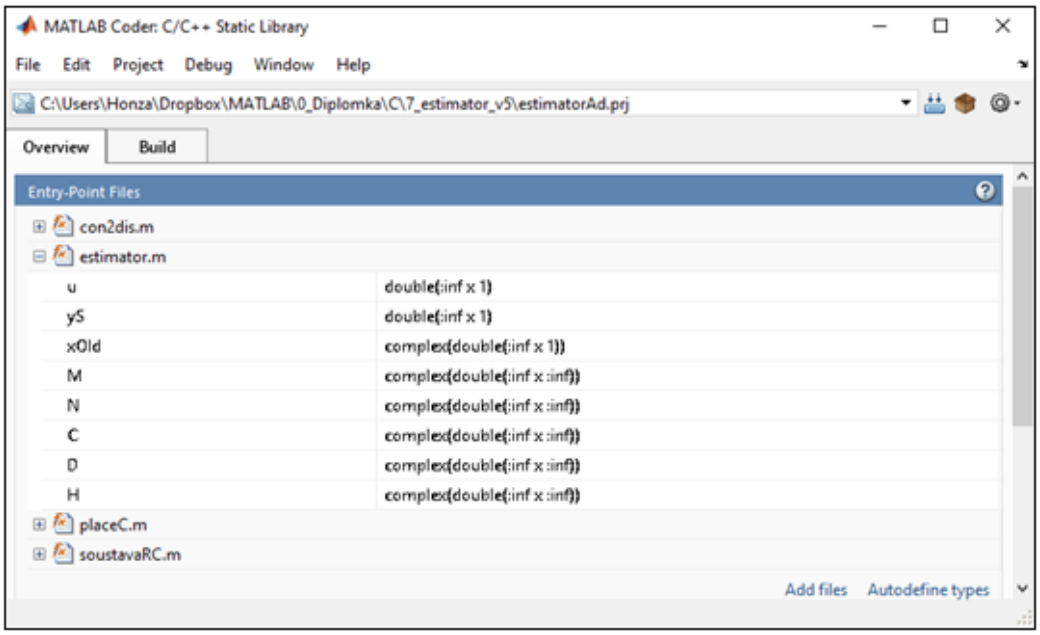
Obsažené algoritmy lze využít i k přehlednému grafickému určení parametrů regulátorů a k analýze a určení bezpečnosti regulačního obvodu.

Control System Toolbox nabízí i několik algoritmů použitelných při návrhu stavového estimátoru. Tyto algoritmy jsou kvalitně zpracované a implementují i kontrolu vstupních parametrů a dosažených výsledků.

4.2 MATLAB CODER

MATLAB Coder je volitelným toolboxem prostředí MATLAB, který umožňuje konverzi funkcí napsaných v jazyce MATLAB do samostatného kódu jazyka C a C++. Výstupem Coderu může být spustitelná aplikace, knihovna C/C++ nebo funkce MEX (Matlab EXecutable), která se spouští z prostředí MATLAB a umožňuje zrychlení výpočetně náročných algoritmů. V této práci je využita možnost generování statické knihovny, která je

začleněna do výsledného řešení realizace estimátoru. Embedded Coder je dalším toolboxem rozšiřujícím možnosti programu MATLAB Coder o generování optimalizovaného kódu určeného pro embedded procesory (Humusoft, 2018).



Obr. 4.1 – Rozhraní programu MATLAB Coder

Na obr. 4.1 je okno projektu MATLAB Coder, které slouží pro přidání funkcí a definování jejich vstupních proměnných. Zatímco typ proměnné (single/double, real/complex, atd.) musí být zadán jednoznačně, rozměry (v prostředí MATLAB je základní datový typ matice) mohou být zadány jako „až nekonečné“, nejsou-li předem známy. Volbu typu a rozměrů výstupních proměnných následně provádí Coder automaticky.

Výsledkem úspěšné konverze je (v závislosti na nastavení) několik vygenerovaných souborů, pro statickou knihovnu, kterou využívá tato práce, jsou to soubory .c a .h. Soubor jmeno_projektu.h obsahuje deklarace všech vygenerovaných funkcí, včetně funkcí pro vytvoření složených datových typů.

4.2.1 Konverze datových typů

Rozdílem, který uživatel při generování kódu pocítí patrně nejvíce, je konverze datových typů použitých v prostředí MATLAB do nativních typů jazyka C.

V MATLAB je základní datový typ matice. Uživatel nemusí definovat její datový typ, její rozměry, ani zda jsou čísla reálná či komplexní. Použití těchto proměnných je velmi intuitivní a jednoduché. Prostředí zajišťuje kontrolu rozměrů, orientace a podobně.

V jazyce C je implementace těchto vlastností úkolem uživatele a může být realizována mnoha způsoby. MATLAB Coder definuje vlastní datové typy, z nichž některé jsou uvedené v tab. 4.1.

Tab. 4.1 – Datové typy programu MATLAB Coder

Jméno struktury	Obsah struktury
creal_T	dvě položky velikosti double: reálná a imaginární část čísla
creal32_T	dvě položky velikosti float: reálná a imaginární část čísla
emxArray_real_T	tři položky: pole typu double, počet řádků a počet sloupců
emxArray_creal_T	tři položky: pole typu creal_T, počet řádků a počet sloupců
emxArray_real32_T	tři položky: pole typu float, počet řádků a počet sloupců
emxArray_creal32_T	tři položky: pole typu creal32_T, počet řádků a počet sloupců s

Je nutné dbát na jednotu obsahu proměnné v průběhu životního cyklu algoritmu. V MATLAB není problémem uložit do proměnné číslo s plovoucí desetinnou čárkou a o řádek dál do té samé proměnné uložit textový řetězec. V jazyce C něco takového možné není a tak je třeba v původním kódu zajistit, aby se to nestalo. Už při konverzi musí být známo, jakého typu proměnná bude, tento typ se nesmí změnit a ani není možné, aby typ proměnné mohl mít více alternativ, například kvůli větvení programu.

4.2.2 Omezení

Z pochopitelných důvodů není možná konverze vstupně-výstupních operací. Některé z nich (např. `disp()` – jednoduchá funkce k výpisu do terminálu) jsou ignorovány a v generovaném kódu chybí, jiné (např. `ctrlMsgUtils.error()` – funkce k zobrazení chybového hlášení) nesmí být v originálním kódu vůbec.

Dalším omezením je nemožnost rozměrových nejistot. MATLAB Coder před zahájením vlastní konverze provede kontrolu originální funkce v jazyce MATLAB, projde všechny její větve a určí typ a rozměry vnitřních a výstupních proměnných. Tyto parametry musí být možné určit jednoznačně a musí být pro všechny větve stejné.

Z těchto omezení vyplývá i možnost použití funkcí, které jsou součástí vybavení prostředí MATLAB. Pokud jsou Coderem nativně podporovány (např. funkce `expm()`), je jejich použití uvnitř uživatelské funkce bezproblémové. Nepodporované funkce (např. funkce `place()`) musí být před konverzí upraveny (vytvořeny jejich upravené kopie).

Velký vliv na omezení má verze programu MATLAB Coder. Z některých význačných změn jsou to např. podpora Cell Array (pole buněk) od verze R2016a, podpora rekurzivních funkcí a Machine Learning Toolboxu od verze R2016b, podpora textových řetězců a FFTW library od verze R2017b a podpora řídkých matic od verze R2018a (MATLAB, 2018a).

Při přípravě vzorové funkce ke generování je vhodné umístit na úplný začátek souboru, ve kterém je funkce zapsaná, frázi `%#codegen`, která zajistí průběžnou kontrolu omezení a graficky zvýrazní nepodporované operace.

Demonstrace obejití některých omezení je spolu s průběhem přípravy kódu uvedena v popisu praktické části této práce.

4.2.3 Implementace knihovny do vlastního kódu

Následující příklad kódu ukazuje použití funkce v jazyce MATLAB ve srovnání s použitím téže funkce (vygenerované) v jazyce C při operaci s komplexními maticemi.

```
%kód v jazyce MATLAB
complexMatIn=[11.2+1i, 3.8-5.2i; 5, -0.3i];
%vytvoření a naplnění proměnné
complexMatOut=mojeFce(complexMatIn);
%volání funkce
```

```
//kód v jazyce C využívající generovanou knihovnu
#include <stdio.h>
#include "mojeFce\mojeFce.c"
#include " mojeFce\rt_nonfinite.c"
#include " mojeFce\rtGetInf.c"
#include " mojeFce\rtGetNaN.c"
//připojení vygenerované knihovny

emxArray_creal_T *complexMatIn, *complexMatOut;
//vytvoření ukazatelů na proměnné

int main(){
   //extern emxArray_creal_T *emxCreate_creal_T
   //(int rows, int cols);
   //hlavička vygenerované funkce pro vytvoření proměnných
   complexMatIn = emxCreate_creal_T(2,1);
   complexMatOut = emxCreate_creal_T(2,1);
   //alokace paměti a určení rozměrů proměnných

   complexMatIn->data[0].re=11.2;
   complexMatIn->data[0].im=1.0;
   complexMatIn->data[1].re=3.8;
```

```
    complexMatIn->data[1].im=-5.2;
    complexMatIn->data[2].re=5.0;
    complexMatIn->data[2].im=0;
    complexMatIn->data[3].re=0;
    complexMatIn->data[3].im=-0.3;
    //naplnění proměnné hodnotami

    //extern void mojeFceNa2
    //(const emxArray_creal_T *compMat,
    // emxArray_creal_T *compMatNa2);
    //hlavička vygenerované funkce mojeFce
    mojeFce(complexMatIn, complexMatOut);
    //volání vygenerované funkce

    return 0;
}
```

4.3 FUNKCE VYUŽITELNÉ PŘI NÁVRHU ESTIMÁTORU

V tab. 4.2 jsou uvedeny některé funkce prostředí MATLAB využitelné pro návrh diskrétního estimátoru stavu (MATLAB, 2018b).

Tab. 4.2 – Funkce MATLAB pro návrh estimátoru

zdroj	funkce	popis	použití
MATLAB	`expm()`	výpočet maticové exponenciály	výpočet matic diskrétního stavového modelu
	`rank()`	výpočet hodnosti matice	určení pozorovatelnosti systému popsaného stavovým modelem (v kombinaci s `obsv`)
	`eig()`	výpočet vlastních čísel matice	určení pólů systému popsaného stavovým modelem
Control System Toolbox	`obsv()`	sestavení matice pozorovatelnosti	určení pozorovatelnosti systému popsaného stavovým modelem (v kombinaci s `rank`)
	`acker()`	výpočet zesílení estimátoru	výpočet zesílení deterministického estimátoru metodou umístění pólů, starší metoda

Tab. 4.2 – Funkce MATLAB pro návrh estimátoru - pokračování

zdroj	funkce	popis	použití
Control System Toolbox	`place()`	výpočet zesílení estimátoru	výpočet zesílení deterministického estimátoru metodou umístění pólů, náhrada `acker`
	`dlqr()`	výpočet zesílení estimátoru	výpočet zesílení deterministického estimátoru LQ návrhem
	`kalman()`	výpočet zesílení estimátoru	výpočet zesílení stochastického estimátoru s předpokladem konstantních vlastností šumu

4.4 PROSTŘEDKY KOMUNIKACE

MATLAB nabízí několik možností reprezentace dat a komunikace s uživatelem i s dalšími zařízeními.

4.4.1 Obsluha sériového portu

Sériový port (neboli sériové rozhraní, sériová linka nebo standard RS-232) je komunikační rozhraní pro spojování počítačů a jiných elektronických zařízení. Ačkoli se od něj u osobních počítačů pomalu opouští a často nebývá vůbec vyveden ven ze skříně PC, pro svou jednoduchost a další typické vlastnosti se pro některé účely používá stále (případně jeho modifikace standardy RS-422 a RS-485, které jsou široce využívány v průmyslových podmínkách). Zařízení, která komunikují přes sériovou linku, se dnes k PC zpravidla připojují přes USB port pomocí převodníků USB-to-Serial (neboli USB/RS-232), což je i případ platformy Arduino, kde je převodník součástí DPS.

Obsluha sériového portu je v prostředí MATLAB zajištěna několika implementovanými funkcemi, které umožňují snadnou cestou najít a otevřít libovolný dostupný port, nastavit požadované vlastnosti přenosu a realizovat komunikaci s připojeným zařízením. Funkce a příkazy použité v této práci a některé další jsou uvedené v tab. 4.3.

Tab. 4.3 – Funkce obsluhy sériového portu

funkce	popis
`seznam_portu = instrhwinfo('serial');`	vyhledání všech dostupných sériových portů a uložení výpisu do struktury seznam_portu
`port = seznam_portu.SerialPorts(n);`	výběr jednoho portu ze seznamu (na n-té pozici) a uložení jeho identifikátoru do proměnné port
`scom = serial(port, 'BaudRate', rychlostB);`	vytvoření struktury scom typu serial s vybraným portem a zadanými parametry
`set(scom, 'TimeOut', 1, 'InputBufferSize', 65536);`	změna parametrů struktury scom typu serial
`flushinput(scom);`	vyprázdnění vstupního bufferu sériového portu scom
`fopen(scom);`	otevření sériového portu scom
`fclose(scom);`	uzavření sériového portu scom
`inputline = fscanf(scom);`	načtení textového řetězce z portu scom a uložení do proměnné inputline
`fprintf(scom, '%s\n', outputline);`	zápis textového řetězce v proměnné outputline do portu scom

4.4.2 Přístup k souboru

Zápis a čtení dat ze souboru je nezbytné pro práci s informacemi, které mají být dostupné delší dobu a to i po vypnutí a opětovném zapnutí programu. MATLAB umožňuje pracovat s textovými i binárními soubory a definuje vlastní datový typ .mat pro rychlé uložení a načtení dat, která jsou aktuálně v jeho workspace.

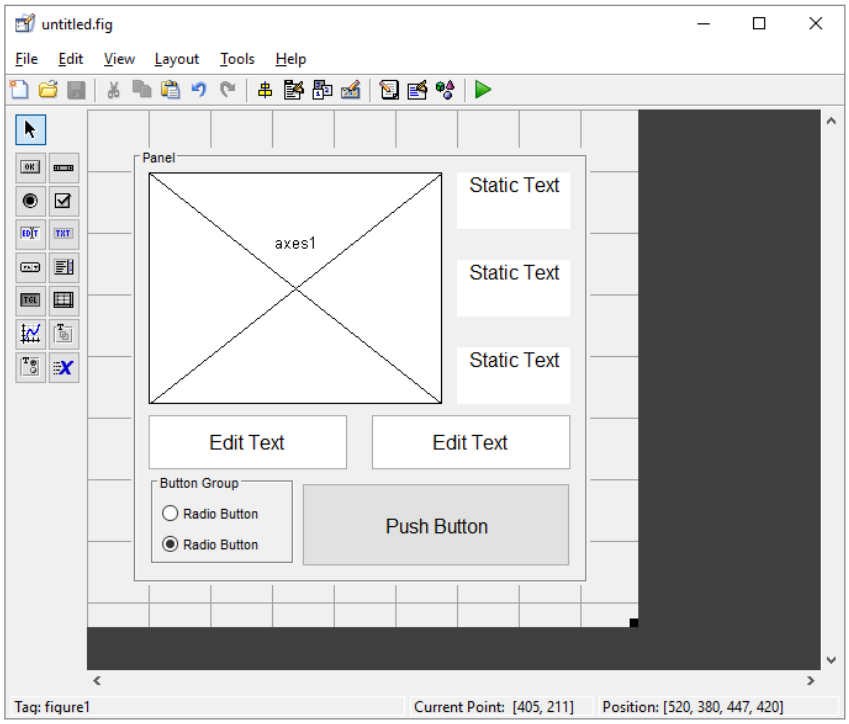
V této práci je pro univerzálnost a přenositelnost využito práce s textovým souborem. Funkce, které jsou určené pro práci s ním, jsou formálně totožné s funkcemi pro obsluhu sériového portu. Seznam použitých funkcí a jejich popis je uveden v tab. 4.4.

Tab. 4.4 – Funkce pro práci s textovým souborem

funkce	popis
`fid = fopen('log.txt', 'w');`	otevření/vytvoření souboru log.txt a uložení jeho identifikátoru do proměnné fid
`fclose(fid);`	uzavření souboru s identifikátorem fid
`fgetl(fid);`	načtení řádku z fid
`A=(fscanf(fid, '%f %f %f %f', [4 inf]));`	načtení všech čísel z fid a uložení do matice o 4 řádcích a předem neurčeném počtu sloupců
`fprintf(fid, '\n%s', inputline);`	zápis proměnné inputline do fid

4.4.3 GUI

MATLAB nabízí jednoduchou cestu k vytvoření grafického uživatelského rozhraní pomocí nástroje GUIDE. Ten umožňuje použití metody drag&drop (táhni a pusť) k určení rozmístění prvků na hlavním panelu a automaticky generuje příslušný kód obsluhující tyto prvky. Nástroj GUIDE je možné spustit potvrzením příkazu `guide()` v prostředí MATLAB.



Obr. 4.2 – Okno nástroje GUIDE

Nástroj GUIDE (na obr. 4.2) obsahuje všechny základní prvky pro tvorbu jednoduchého GUI. Z často používaných jsou to například tlačítka (PushButton, ToggleButton), textová pole (EditText, StaticText), nabídky (ListBox, Pop-upMenu) a grafy (Axes).

Grafické rozhraní je v MATLAB možné vytvořit i textově pomocí funkcí k tomu určených. Tento způsob je vhodný pro případy, kdy přesná podoba není předem známa a rozhraní je nutné generovat dynamicky podle zadaných parametrů.

5 ARDUINO

Arduino je souhrnný název pro různé verze jednodeskového počítače a k nim příslušné vývojové prostředí. Od svého vzniku v roce 2005 dosáhlo Arduino celosvětové popularity a rozšíření a dnes je mimo výukové účely používáno mnoha amatérskými i profesionálními vývojáři pro svou všestrannost, dostupnost a otevřenost a snadné použití.

5.1 FILOZOFIE A KONCEPT

Projekt Arduino byl vyvinut jako jednoduchá elektronická platforma pro studenty technických oborů a z toho důvodu byl a nadále je šířen pod licencí open-source, a jeho veškerá technická dokumentace je tak volně přístupná. To se týká hardwaru, kde je možné dohledat originální projekty popisující DPS, použité součástky a celkové vlastnosti, i softwaru, kde jsou k dispozici kompletní zdrojové kódy. Obojí je možné svobodně zkoumat, měnit a dále distribuovat, díky čemuž je na trhu k dostání množství legálních kopií (tzv. klonů) desek Arduina vyrobenými různými výrobci po celém světě, které s originály sdílí většinu vlastností obvykle za nižší cenu.

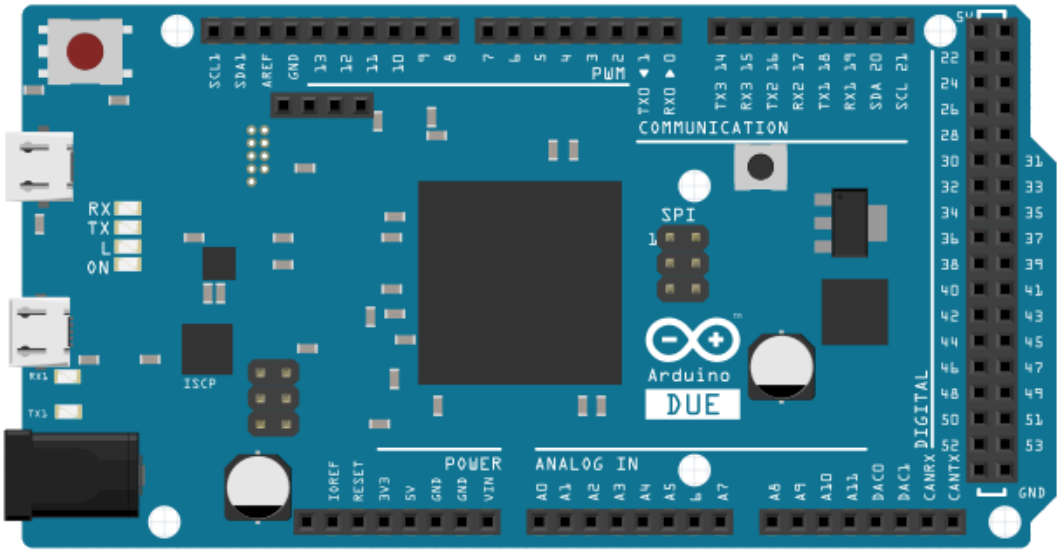
Jedním z důležitých atributů konceptu Arduino je možnost snadného rozšíření základní desky o takzvané shieldy. Tato zařízení s jednotným rozložením konektorů jsou určena k připojení k většině základních desek jednoduchým nasazením a poskytují hardwarový základ k různým dalším funkcím této platformy. Mezi nejčastěji využívané patří např. Ethernet Shield, který umožňuje komunikaci po ethernetové síti i dále do internetu, Motor Shield, určený pro řízení různých typů motorů, a GSM/GPRS Shield pro komunikaci přes mobilní síť.

5.2 HARDWARE

Pod označením Arduino je k dostání celá řada jednodeskových počítačů, které se vzájemně liší osazeným procesorem, rozložením pinů, velikostí desky a dalšími vlastnostmi.

5.2.1 Arduino Due

Arduino Due je jednou z výkonnějších desek, které řada Arduino nabízí. I přes některé rozdíly s ostatními modely této platformy zachovává kompatibilitu s většinou používaných rozšíření (shieldů). Rozložení desky Arduino Due je na obr. 5.1. Parametry desky čerpány z (Arduino Store, 2018).



Obr. 5.1 – Arduino Due

Arduino Due je vybaveno 32bitovým RISC **procesorem** Atmel SAM3X8E ARM Cortex-M3 s taktem 84 MHz. Tento procesor harvardské architektury má k dispozici 512 KB paměti Flash určené pro nahrání programu a dalších 96 KB paměti SRAM pro data.

Napětí je oproti jiným Arduino (typicky 5 V) deskám 3,3 V, což má z uživatelského hlediska vliv především na maximální napětí, které lze na pinech desky generovat a měřit. Připojení většího napětí může desku poškodit. Napájení je možné z konektoru USB či externími (7 ÷ 12) V (Arduino, 2017).

USB port je na desce osazen dvakrát. **Programovací USB port** je k procesoru připojen přes převodník USB-to-Serial a je primárním portem pro nahrávání programu a komunikaci s Arduinem. Zahájení komunikace s tímto USB vždy provede reset zařízení. **Nativní USB port** je propojen přímo s procesorem desky a umožňuje připojit Arduino k PC jako běžnou periferii (chová se jako myš, klávesnice atd.) nebo naopak připojovat jiné periferie k Arduinu (myš, klávesnici, telefon s Androidem apod.). Zahájení komunikace s tímto portem provede restart desky pouze při rychlosti přenosu 1200 bps.

Na desce je 54 **digitálních pinů**, které slouží jako vstupy nebo výstupy. 12 z nich dovede poskytovat analogový výstup ve tvaru **PWM**.

Arduino Due může díky **A/D** a **D/A převodníkům** na dvanácti **analogových pinech** měřit a na dvou analogových pinech generovat spojitý signál. Výchozí rozlišení je 10 bitů pro čtení a 8 bitů pro generování napětí, lze však u obou nastavit rozlišení až 12bitové.

Hardwarovým omezením D/A převodníku je možnost generovat napětí jen mezi (0,6 ÷ 2,6) ± 0,1 V dle konkrétní desky.

Vnitřní **časovače** umožňují měřit čas od začátku běhu programu a tím přeneseně i čas potřebný pro běh funkcí a jiných volání. Pro Arduino Due je možné měřit čas s přesností na 1 mikrosekundu

Rozměry desky jsou udávány jako 101,5 × 53,3 × 21,1 mm.

5.2.2 Srovnání desek Arduino

V tab. 5.1 jsou uvedeny základní vlastnosti nejrozšířenějších desek Arduino, nabízející jejich srovnání. Parametry čerpány z (Arduino, 2018a).

Tab. 5.1 – Přehled desek Arduino

Deska	Takt procesoru	Šířka slova	Paměť Flash	Paměť SRAM	Paměť EEPROM	Počet pinů		
						DI/DO	AI	AO
Arduino Uno Rev3	16 MHz	8 bitů	32 KB	2 KB	1 KB	14	6	-
Arduino Leonardo	16 MHz	8 bitů	32 KB	2,5 kB	1 KB	20	12	-
Arduino Micro	16 MHz	8 bitů	32 KB	2,5 kB	1 KB	20	12	-
Arduino Nano	16 MHz	8 bitů	32 KB	2 kB	1 KB	22	8	-
Arduino Mega 2560 Rev3	16 MHz	8 bitů	256 KB	8 KB	4 KB	54	16	-
Arduino Due	84 MHz	32 bitů	512 KB	96 KB	-	54	12	2
Arduino Yún	16 MHz	8 bitů	32 KB	2,5 kB	1 KB	20	12	-
	400 MHz	32 bitů	16 MB	2,5 kB	1 KB			

Všechny uvedené desky rovněž dovedou na některých svých DI/DO pinech generovat PWM signál. Počet těchto pinů závisí na konkrétní desce.

Zařízení Arduino Yún se od ostatních uvedených desek odlišuje druhým procesorem (Atheros AR9331) pro běh systému Linux a ethernetovým a Wifi portem a je určené pro

použití v oblasti IoT (Internet of Things). Kromě paměti uvedené v tab. 5.1 má navíc ještě 64 MB paměti RAM (DDR2).

5.3 SOFTWARE

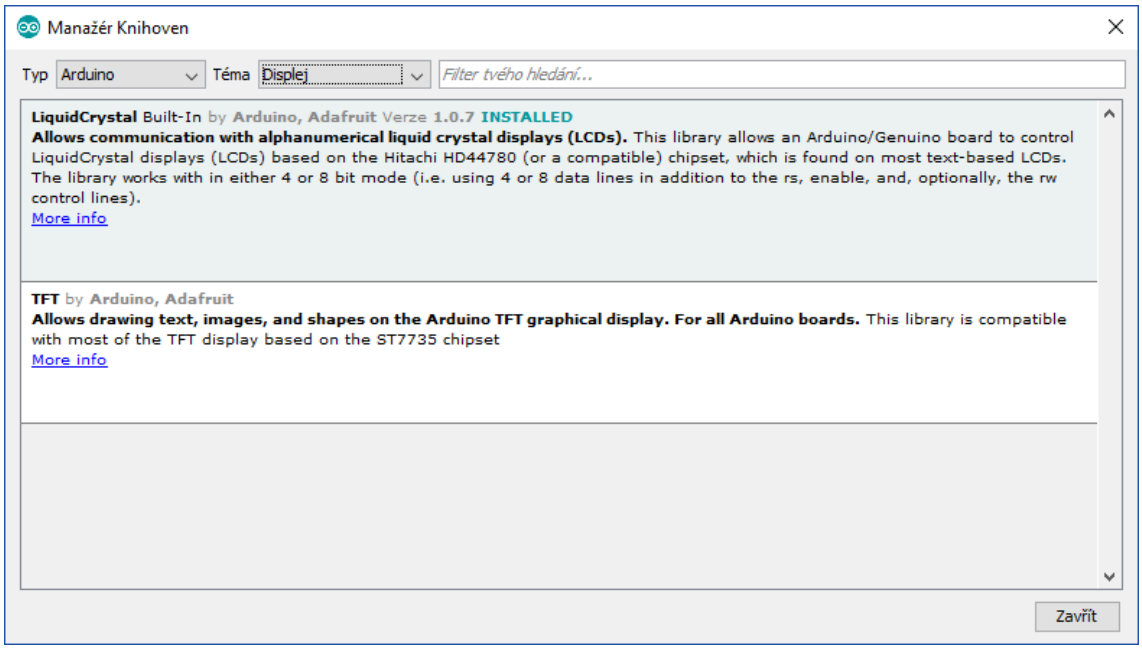
Jednodeskový počítač Arduino je možné programovat mnoha způsoby s využitím různých vývojových prostředí a externích programátorů (např. Atmel Studio pro desky s procesory AVR a SAM nebo Eclipse s nainstalovaným modulem pro podporu Arduina). Pro většinu aplikací je však plně dostačující využití již připravených nástrojů navržených specificky pro platformu Arduino a to Arduino IDE s jazykem Arduino Programming Language a programátor umístěný přímo na desce Arduino.

5.3.1 Arduino Desktop IDE

Arduino Desktop IDE je open-source grafické vývojové prostředí pro vývoj a zavádění aplikací pro desky Arduino. Je jednotné a univerzální pro všechny desky z platformy Arduino a všechny PC platformy, na kterých je spouštěno.

Editor zdrojového kódu standardně zvýrazňuje syntaxi a před kompilováním provádí její kontrolu. Pro nahrávání kódu do zařízení podporuje Arduino Desktop IDE možnost využití klasického připojení přes USB kabel nebo přes externí programátor (v programu je třeba zvolit ten právě používaný). Komunikaci se zařízením je možné realizovat přes obsažený sériový monitor a sériový plotter.

Manažér knihoven (na obr. 5.2) udržuje nainstalované knihovny aktuální a nabízí přehled všech knihoven dostupných v tematicky dělených online repozitářích.



Obr. 5.2 – Manažér knihoven

Oficiální variantou k Arduino Desktop IDE je Arduino Web Editor, který nabízí stejnou funkcionalitu (rozšířenou o možnost ukládání projektů do cloudu) bez nutnosti instalace.

5.3.2 Arduino Programming Language

Programování zařízení Arduino probíhá v zjednodušené podobě jazyka C++ označovaného Arduino Programming Language (dále APL). To umožňuje použití i mnoha standardních C/C++ knihoven. Základní syntaxe jazyka C++ (práce s proměnnými, řízení běhu programu apod.) zůstala zachována.

Součástí jazyka je kolekce knihoven a funkcí vytvořených cíleně pro Arduino. To značně usnadňuje práci s interním hardwarem, jako je např. obsluha I/O portů, časovačů a komunikačních rozhraní, i externím hardwarem, jako jsou např. LCD displeje a různé typy motorů a čidel.

Oproti standardnímu C++ má APL několik rozdílů. Po formální stránce byly změněny názvy některých datových typů (např. `bool` → `boolean` a `unsigned char` → `byte`) a ve struktuře programu se již dále nevyskytuje funkce `main()`. Ta byla nahrazena dvojicí funkcí `setup()`, která se spustí jako první na začátku běhu programu a provede se pouze jednou, a `loop()`, která se spustí automaticky po funkci `setup()` a opakuje se stále dokola (Arduino, 2018b).

5.3.3 Použité funkce

Specifické APL funkce (Arduino, 2018b) použité v této práci (zpravidla funkce obsluhující hardware) a některé další jsou uvedené v tab. 5.2.

Tab. 5.2 – Specifické APL funkce

funkce	popis
`Serial.begin();`	otevření sériového portu se zvolenou rychlostí přenosu
`Serial.print();`	zápis dat do sériového portu ve formátu ASCII
`Serial.println();`	zápis dat do sériového portu a přidání symbolu konce řádku
`Serial.available();`	zjištění počtu znaků dostupných ve vstupním bufferu
`Serial.parseInt()`	načtení celého čísla z vstupního bufferu
`Serial.parseFloat();`	načtení reálného čísla z vstupního bufferu
`analogWriteResolution();`	konfigurace rozlišení pro funkci `analogWrite()`
`analogReadResolution();`	konfigurace rozlišení pro funkci `analogRead()`
`analogWrite();`	generování analogové hodnoty nebo signálu PWM na jednom pinu
`analogRead();`	měření analogové hodnoty na jednom pinu
`delay();`	pozastaví program na zadaný počet milisekund
`micros();`	vrátí uplynulý počet mikrosekund od začátku běhu programu
`pinMode();`	konfigurace pinu na digitální vstup nebo výstup
`digitalWrite();`	zápis hodnoty HIGH nebo LOW na digitální výstup
`digitalRead();`	čtení hodnoty HIGH nebo LOW z digitálního vstupu

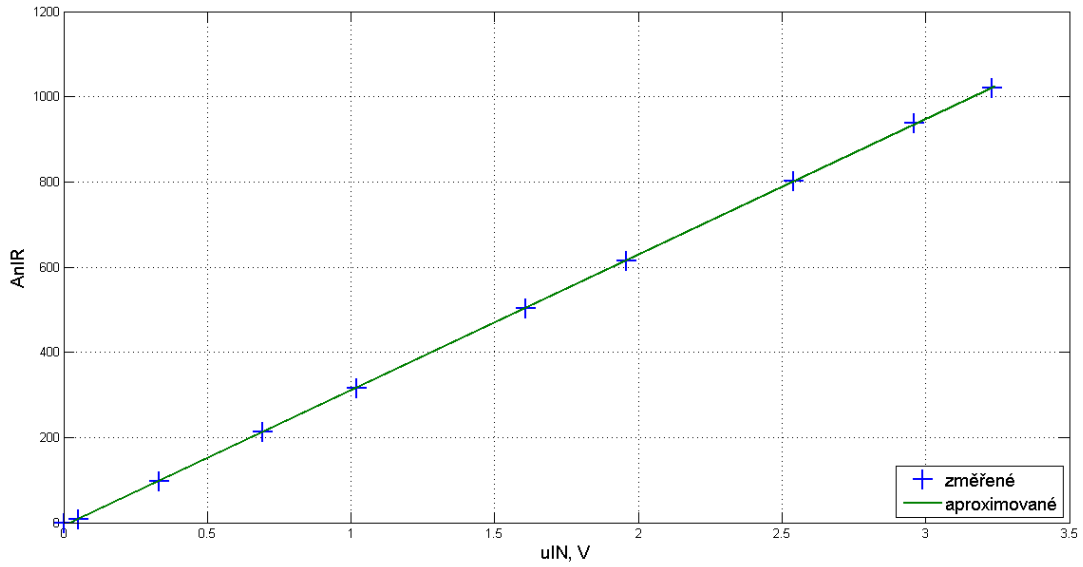
Některé zde uvedené funkce nejsou dostupné pro všechny desky Arduino. Obecně se tato omezení týkají především funkcí pro měření a generování analogových hodnot, a tedy např. funkce `analogWriteResolution()` je použitelná pouze v zařízení (z uvedených v tab. 5.1) Arduino Due.

6 NÁVRH A PRŮBĚH EXPERIMENTU – HW ČÁST

Pro vyhodnocení správnosti implementovaného řešení stavového pozorovatele musela být navržena a realizována vhodná soustava, na které mohla být funkčnost ověřena experimentálně. Požadavkem na soustavu byla její snadná realizovatelnost, možnost vytvoření jejího lineárního časově invariantního modelu s měřitelnými stavovými veličinami a možnost tyto veličiny jednoduše měřit zařízením Arduino Due.

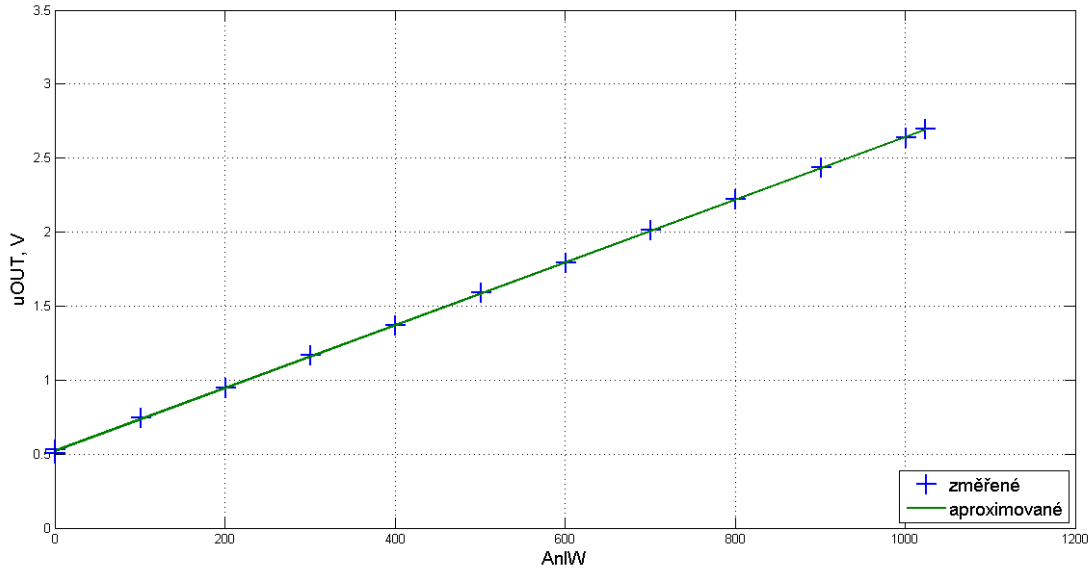
6.1 KALIBRACE VSTUPŮ A VÝSTUPŮ

Byla provedena kalibrace A/D a D/A převodníků desky Arduino Due. Kontrolní měření bylo zajištěno multimetrem DT9205A a externím zdrojem napětí. Pro rozlišení převodníků 10 bitů byly naměřeny charakteristiky na obr. 6.1 a obr. 6.2.



Obr. 6.1 – Kalibrace A/D převodníku

A/D převodník byl kalibrován pro hodnoty od 0 do 3,25 V. Toto rozmezí pokrývá celou využitelnou oblast, neboť při připojení napětí většího než 3,3 V hrozí poškození zařízení.



Obr. 6.2 – Kalibrace D/A převodníku

D/A převodník byl kalibrován pro hodnoty od 0 do 1023, což při zvoleném rozlišení 10 bitů pokrývá celou využitelnou oblast. Byly zjištěny i horní a dolní mez napěťového výstupu převodníku, které jsou uvedeny v tab. 6.1.

Tab. 6.1 – Omezení D/A převodníku

omezení	hodnota napětí, V
dolní mez	0,503
horní mez	2,693

Obě změřené charakteristiky jsou přímky, které neprochází nulou, a byly tedy aproximovány rovnicí ve tvaru $y = a \cdot x + b$. Výsledné koeficienty rovnic (6.1) a (6.2) byly získány použitím funkce MATLAB `polyfit()`.

$$u_{\mathrm{IN}} = 0{,}003142 \cdot AnlR + 0{,}022773, \tag{6.1}$$

$$AnlW = 473{,}314509 \cdot u_{\mathrm{OUT}} - 252{,}445510. \tag{6.2}$$

kde u_{IN} je skutečná hodnota měřeného napětí, V,

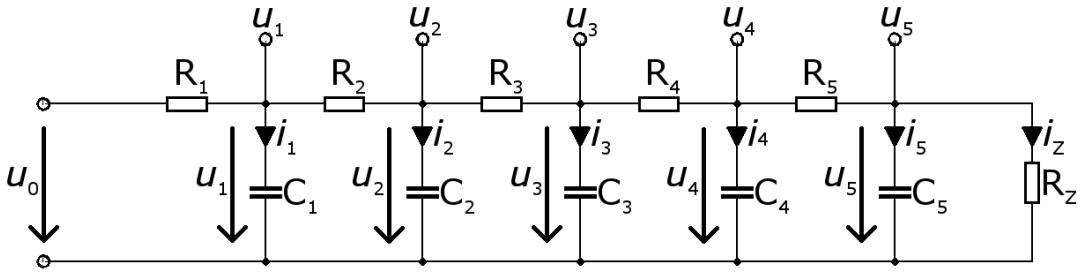
u_{OUT} je žádaná hodnota generovaného napětí, V,

AnlR je hodnota získaná funkcí `analogRead()`,

AnlW je parametr funkce `analogWrite()`.

6.2 RC SOUSTAVA

Jednou ze soustav splňujících požadavky kladené v tomto experimentu je soustava několikanásobného RC článku, jejíž zapojení je znázorněné na obr. 6.3.



Obr. 6.3 – Elektrické schéma soustavy

Řád soustavy je dán počtem RC článků v sérii a jejich přidáním či odebráním může být lehce změněn.

6.2.1 Použité součástky

Hodnoty součástek použitých při realizaci RC soustavy jsou uvedeny v tab. 6.2.

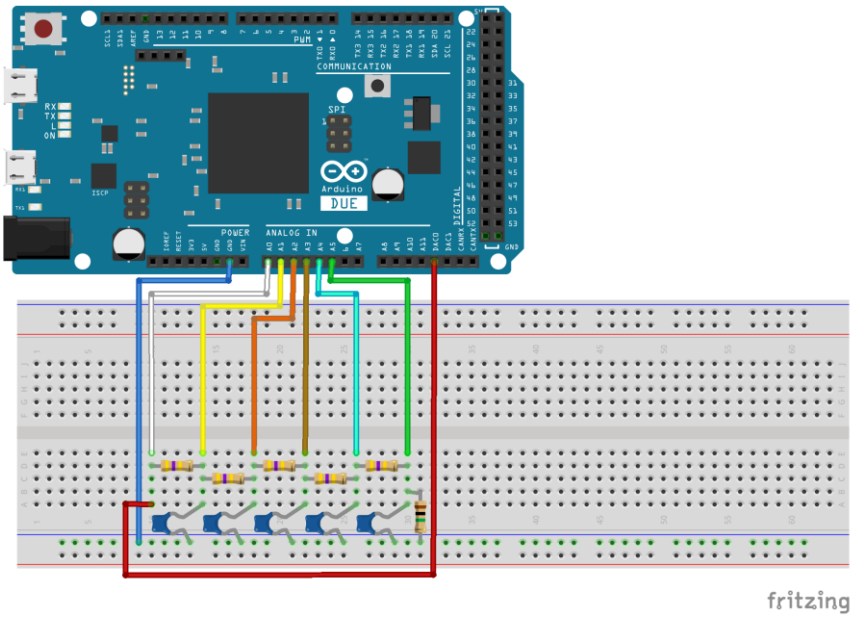
Tab. 6.2 – Hodnoty součástek

Součástka	Hodnota
$R_1 – R_5$	470 kOhm
$C_1 – C_5$	10 nF
R_Z	1 MOhm

Volba hodnot součástek byla provedena s ohledem na přibližnou očekávanou rychlost estimace.

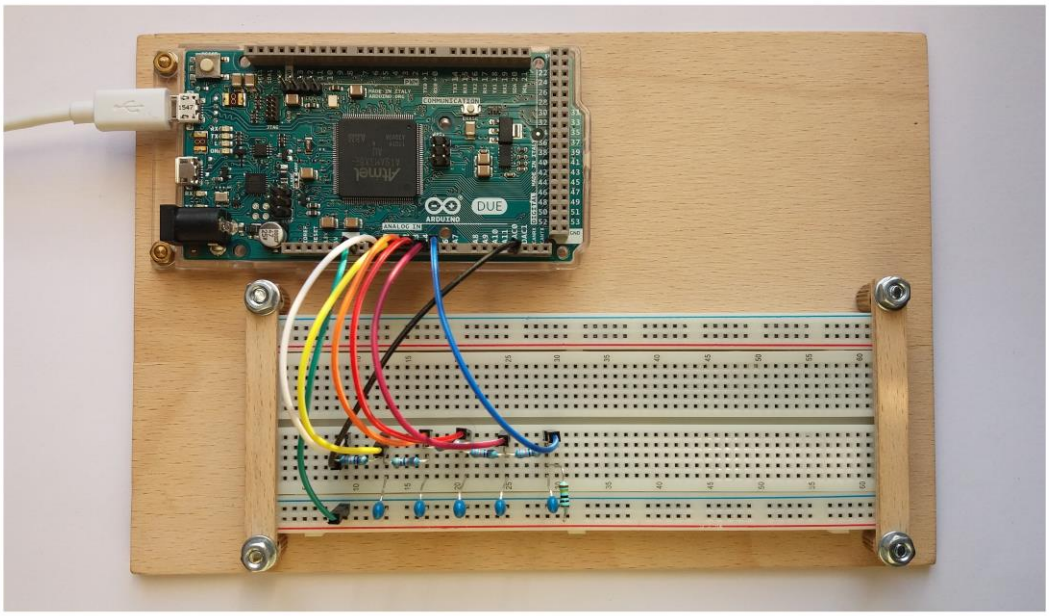
6.2.2 Zapojení

Tuto soustavu je možné snadno připojit k desce Arduino Due, která slouží jako generátor vstupního napětí u_0 a zároveň měří hodnoty napětí u_1 až u_5 na kondenzátorech C_1 až C_5.



Obr. 6.4 – Zapojení Arduina s RC článkem

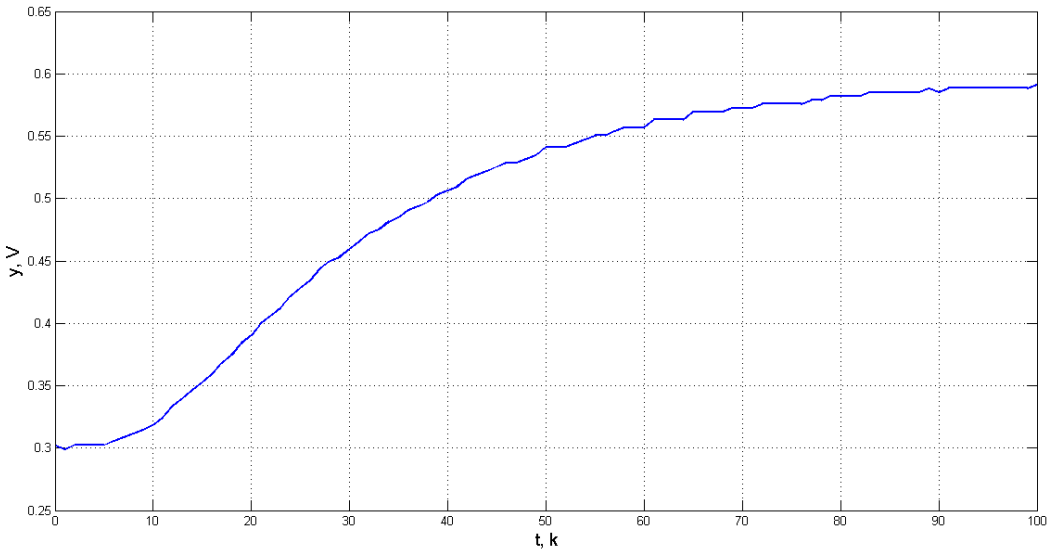
Na obr. 6.4 je zobrazeno propojení desky Arduino Due s nepájivým polem se součástkami tvořícími pětinásobný RC článek. Obrázek byl vytvořen v open-source programu Fritzing. Reálné zapojení je zobrazeno na obr. 6.5.



Obr. 6.5 – Fotodokumentace soustavy

6.2.3 Odezva soustavy

Po připojení soustavy bylo možné změřit její odezvu, která je zobrazená na obr. 6.6. Na signálu byl zjištěn pouze zanedbatelný šum a v pozdější části práce tedy bylo možné použít deterministické metody pro návrh estimátoru.



Obr. 6.6 – Odezva soustavy

6.3 TVORBA STAVOVÉHO MODELU

Matematický model experimentální soustavy RC článků byl vytvořen aplikací Kirchhoffových zákonů a následným dosazením rovnice kondenzátoru

$$u(t) = \frac{1}{C} \cdot \int_0^t i(t) \cdot \mathrm{d}t, \text{ z čehož} \tag{6.3}$$

$$i(t) = C \cdot \frac{\mathrm{d}u(t)}{\mathrm{d}t}. \tag{6.4}$$

Pro vyjádření matematického popisu ve tvaru stavového modelu s hodnotami napětí u_1 až u_5 jako stavovými veličinami bylo prvním krokem popsání těchto veličin jako

$$u_5 = R_\mathrm{Z} \cdot i_\mathrm{Z}, \tag{6.5}$$

$$u_4 = R_5 \cdot (i_5 + i_\mathrm{Z}) + u_5, \tag{6.6}$$

$$u_3 = R_4 \cdot (i_4 + i_5 + i_\mathrm{Z}) + u_4, \tag{6.7}$$

$$u_2 = R_3 \cdot (i_3 + i_4 + i_5 + i_\mathrm{Z}) + u_3, \tag{6.8}$$

$$u_1 = R_2 \cdot (i_2 + i_3 + i_4 + i_5 + i_\mathrm{Z}) + u_2, \tag{6.9}$$

$$u_0 = R_1 \cdot (i_1 + i_2 + i_3 + i_4 + i_5 + i_Z) + u_1. \tag{6.10}$$

Vyjádřením proudů i_1 až i_5 z rovnic (6.5) až (6.10) a dosazením rovnice proudu kondenzátoru (6.4) lze dále psát

$$i_5 = \frac{1}{R_5} \cdot u_4 - \left(\frac{1}{R_5} + \frac{1}{R_Z}\right) \cdot u_5 = C_5 \cdot \frac{\mathrm{d}u_5}{\mathrm{d}t}, \tag{6.11}$$

$$i_4 = \frac{1}{R_4} \cdot u_3 - \left(\frac{1}{R_4} + \frac{1}{R_5}\right) \cdot u_4 + \frac{1}{R_5} \cdot u_5 = C_4 \cdot \frac{\mathrm{d}u_4}{\mathrm{d}t}, \tag{6.12}$$

$$i_3 = \frac{1}{R_3} \cdot u_2 - \left(\frac{1}{R_3} + \frac{1}{R_4}\right) \cdot u_3 + \frac{1}{R_4} \cdot u_4 = C_3 \cdot \frac{\mathrm{d}u_3}{\mathrm{d}t}, \tag{6.13}$$

$$i_2 = \frac{1}{R_2} \cdot u_1 - \left(\frac{1}{R_2} + \frac{1}{R_3}\right) \cdot u_2 + \frac{1}{R_3} \cdot u_3 = C_2 \cdot \frac{\mathrm{d}u_2}{\mathrm{d}t}, \tag{6.14}$$

$$i_1 = \frac{1}{R_1} \cdot u_0 - \left(\frac{1}{R_1} + \frac{1}{R_2}\right) \cdot u_1 + \frac{1}{R_2} \cdot u_2 = C_1 \cdot \frac{\mathrm{d}u_1}{\mathrm{d}t}, \tag{6.15}$$

kde *u* jsou aktuální hodnoty napětí, V,

i jsou aktuální hodnoty proudů, A

R jsou hodnoty použitých odporů, Ω,

C jsou hodnoty použitých kondenzátorů, F

Rovnice (6.11) až (6.15) je možné uspořádat do tvaru stavového popisu soustavy RC článku s napětími na kondenzátorech jako stavovými proměnnými.

$$\underbrace{\frac{\mathrm{d}}{\mathrm{d}t} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \boldsymbol{A} \times \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix}}_{\boldsymbol{x}} + \boldsymbol{B} \cdot \underbrace{u_0}_{\boldsymbol{u}}, \tag{6.16}$$

$$\underbrace{u_5}_{\boldsymbol{y}} = \boldsymbol{C} \times \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix}}_{\boldsymbol{x}} + \boldsymbol{D} \cdot \underbrace{u_0}_{\boldsymbol{u}}, \text{ kde} \tag{6.17}$$

$$\boldsymbol{A} = \begin{bmatrix} -\frac{1}{C_1}\cdot\left(\frac{1}{R_1}+\frac{1}{R_2}\right) & \frac{1}{C_1}\cdot\frac{1}{R_2} & 0 & 0 & 0 \\ \frac{1}{C_2}\cdot\frac{1}{R_2} & -\frac{1}{C_2}\cdot\left(\frac{1}{R_2}+\frac{1}{R_3}\right) & \frac{1}{C_2}\cdot\frac{1}{R_3} & 0 & 0 \\ 0 & \frac{1}{C_3}\cdot\frac{1}{R_3} & -\frac{1}{C_3}\cdot\left(\frac{1}{R_3}+\frac{1}{R_4}\right) & \frac{1}{C_3}\cdot\frac{1}{R_4} & 0 \\ 0 & 0 & \frac{1}{C_4}\cdot\frac{1}{R_4} & -\frac{1}{C_4}\cdot\left(\frac{1}{R_4}+\frac{1}{R_5}\right) & \frac{1}{C_4}\cdot\frac{1}{R_5} \\ 0 & 0 & 0 & \frac{1}{C_5}\cdot\frac{1}{R_5} & -\frac{1}{C_5}\cdot\left(\frac{1}{R_5}+\frac{1}{R_Z}\right) \end{bmatrix}, \quad (6.18)$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{1}{C_1}\cdot\frac{1}{R_1} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (6.19)$$

$$\boldsymbol{C} = [0 \quad 0 \quad 0 \quad 0 \quad 1], \quad (6.20)$$

$$\boldsymbol{D} = [0]. \quad (6.21)$$

Pro soustavy vyšších řádů by matice analogicky dále rostly, soustavy nižších řádů lze vyjádřit jako submatice matic uvedených, např. matici $\boldsymbol{A}_3$ a $\boldsymbol{B}_3$ soustavy 3. řádu by tvořily submatice $\boldsymbol{A}_3 = \boldsymbol{A}_5(1{:}3,1{:}3)$ a $\boldsymbol{B}_3 = \boldsymbol{B}_5(1{:}3)$.

7 NÁVRH A PRŮBĚH EXPERIMENTU – SW ČÁST

Pro potřeby realizace navrženého řešení a experimentu byl vytvořen program pro desku Arduino Due využívající knihovnu vygenerovanou programem MATLAB Coder a samostatný program v MATLAB (GUI), který slouží pro předávání dat mezi deskou Arduino a PC.

7.1 VYGENEROVANÁ KNIHOVNA

Vygenerovaná knihovna byla navržena s požadavkem na co nejširší možnost uplatnění a to i mimo rámec této práce. Obsahuje čtyři funkce naprogramované v jazyce MATLAB, z nichž jedna je specifická pro realizovaný experiment a zbývající tři jsou univerzální.

7.1.1 Funkce ardSoustavaRC

Funkce `ardSoustavaRC()` je specifická pro realizovaný experiment a slouží k sestavení matic spojitého stavového modelu soustavy RC článků ze zadaných hodnot použitých součástek. Dále je uvedený kód funkce popsaný v jazyce MATLAB.

```
%kód v jazyce MATLAB pro vytvoření modelu
function [A,B,C,D] = soustavaRC(Rn,Cn,Rz,n)
  A=zeros(n,n);
  for i=1:n-1
    A(i,i)=-((1/Rn(i))+(1/Rn(i+1)))/Cn(i);
    A(i,i+1)=1/(Rn(i+1)*Cn(i));
    A(i+1,i)=1/(Rn(i+1)*Cn(i+1));
  end
  A(n,n)=-((1/Rn(n))+(1/Rz))/Cn(n);

  B=[(1/(Cn(1)*Rn(1)));zeros(n-1,1)];

  C=[zeros(1,n-1),1];

  D=0;
end
```

Vstupem funkce jsou pole **Rn**, **Cn** a proměnná **Rz** obsahující parametry součástek a proměnná **n**, udávající řád soustavy.

Výstupem funkce jsou matice **A**, **B**, **C** a **D** určující spojitý stavový model soustavy.

7.1.2 Funkce ardCon2Dis

Funkce `ardCon2Dis()` slouží k přepočtu matic spojitého modelu na matice modelu diskrétního a k výpočtu pólů tohoto modelu. K výpočtu diskrétních matic jsou využity vztahy (2.21) a (2.23). K výpočtu pólů modelu je využita funkce MATLAB `eig()`.

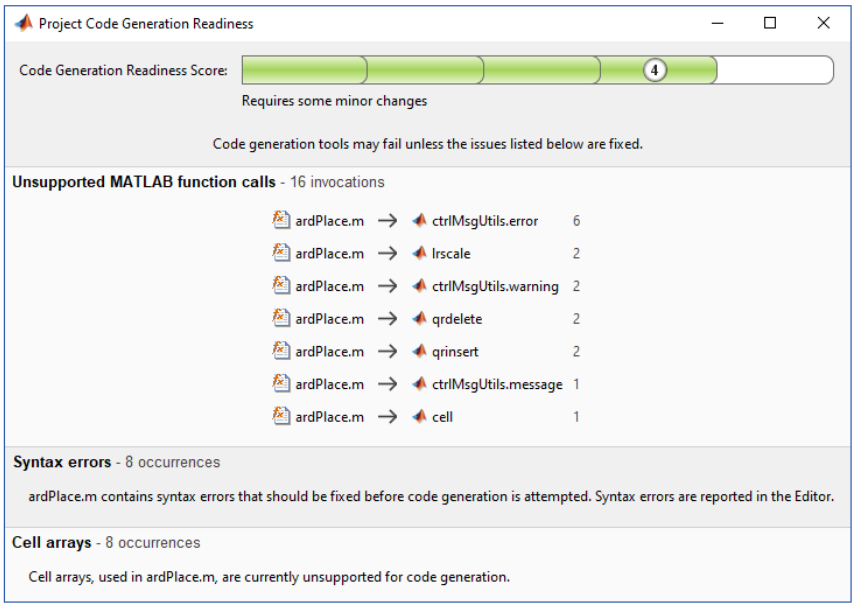
Vstupem funkce jsou matice **A** a **B** určující rovnici dynamiky spojitého stavového modelu soustavy a proměnná **Tvz** udávající periodu vzorkování.

Výstupem funkce jsou matice **M** a **N** určující rovnici dynamiky diskrétního stavového modelu soustavy a vektor **p** obsahující póly diskrétního modelu.

7.1.3 Funkce ardPlace

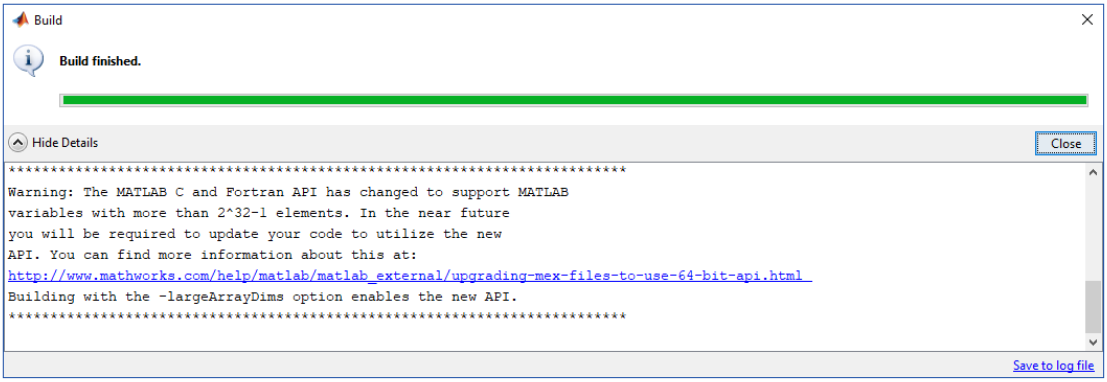
Funkce `ardPlace()` je upravenou verzí funkce MATLAB `place()`, která není podporována programem MATLAB Coder, a slouží k výpočtu zesílení pozorovatele metodou umístění pólů.

Při úpravě původní funkce `place()` bylo nutné vyřešit několik problémů, kdy některé její vnitřní operace narážely na omezení programu MATLAB Coder. Problémy, které byly indikovány (na obr. 7.1) ještě před pokusem o konverzi, byly spojené s voláním nepodporovaných funkcí. Některé z nich, např. volání funkcí `ctrlMsgUtils()`, bylo možné odstranit bez vlivu na funkčnost algoritmu, neboť jejich účelem je vypsání chybového hlášení do terminálu MATLAB a toto lze řešit jinak (v této práci např. výstupní proměnnou *err* indikující chybu). Funkce nutné pro běh programu, např. `qrinsert()`, bylo nutné zkopírovat dovnitř souboru funkce `ardPlace()` a dále je upravovat podobným způsobem, aby bylo možné jejich použití (typicky obsahují volání `error(message())`, které opět pouze indikuje chybu). Problematičtější bylo obejití omezení s použitím Cell Array (pole buněk), se kterým nainstalovaná verze Coderu (R2013b) neumí pracovat. Řešení tohoto problému je značně individuální a závislé na situaci, v této práci bylo možné nahradit Cell Array trojrozměrnou maticí (S{i} → S(:,:,i)), neboť všechny její buňky byly matice stejné velikosti (zjištěno experimentálně pro různé vstupní parametry).



Obr. 7.1 – Upozornění programu MATLAB Coder

Po vyřešení prvotních problémů je možné spustit generování kódu, ještě před ním však dojde k automatické kontrole rozměrů proměnných. Při této kontrole program MATLAB Coder prochází všechny větve funkce a zjišťuje, zda mají proměnné ve všech alternativách zajištěné stejné rozměry a typ (reálný/komplexní atd.) nebo souhlasí-li rozměry a typ na obou stranách operátoru přiřazení. To může být problematické např. v případě, kdy je Coder schopný jednoznačně určit všechny vlastnosti proměnné na jedné straně, ale pouze některé na straně druhé, což je považováno za chybu. Typickým problémem řešeným i v této práci je nutná jednoznačnost vlastností výstupní proměnné (v této funkci je to matice zesílení estimátoru) i v případě, že funkce skončí předčasně. Kdekoliv je tedy v originální funkci použito volání `return()`, musí být před ním výstupní proměnná inicializována na očekávaný typ a rozměry. Úspěšné dokončení konverze kódu je indikovánu hlášením na obr. 7.2.



Obr. 7.2 – Výstup programu MATLAB Coder

Vstupem funkce jsou matice **M** a **C** diskrétního stavového modelu soustavy a vektor **p** obsahující požadovanou polohu pólů diskrétního estimátoru stavu.

Výstupem funkce je matice zesílení estimátoru **H** a proměnná **err** indikující výskyt chyby během výpočtu.

7.1.4 Funkce ardPozorovatel

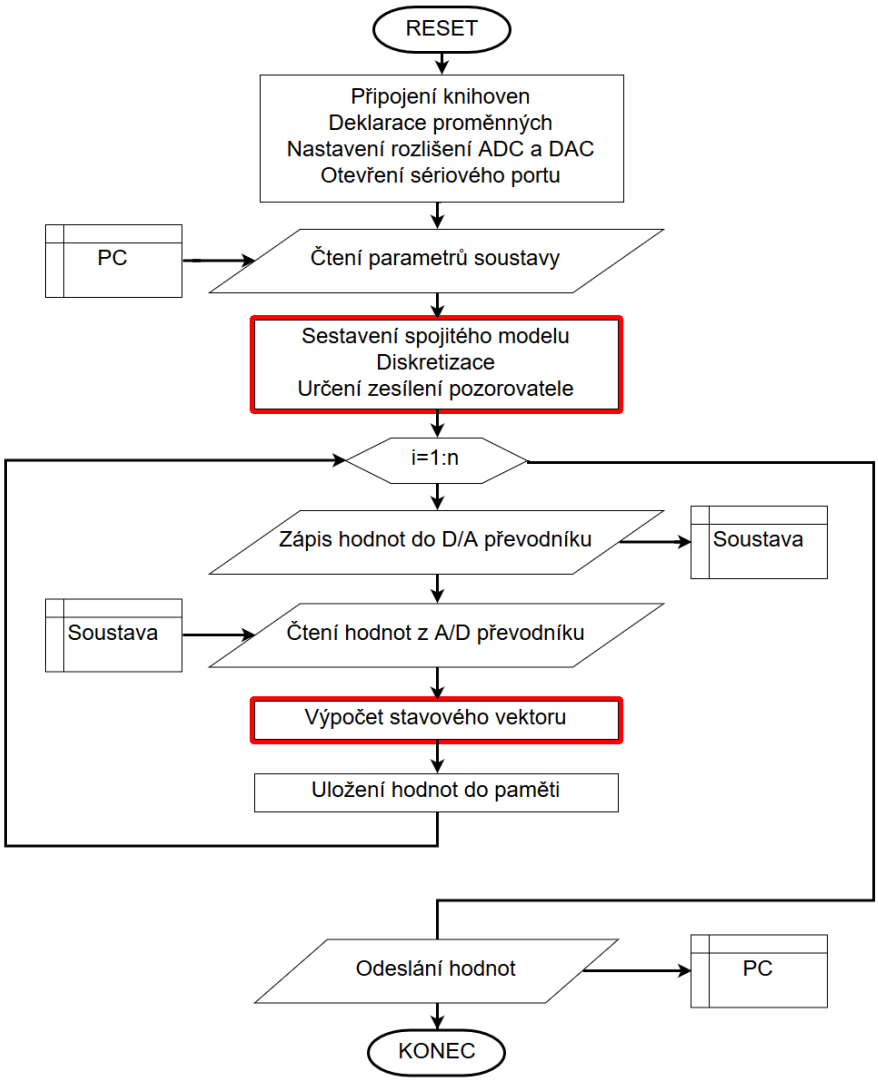
Funkce `ardPozorovatel()` obsahuje rovnici pozorovatele stavu (3.4) a vrací odhad stavového vektoru.

Vstupem funkce jsou matice **M**, **N**, **C** a **D** diskrétního stavového modelu soustavy, matice zesílení estimátoru **H**, vektor **xOld** obsahující odhad stavových veličin z předcházejícího kroku a vektory **u** a **y** obsahující informace o aktuální hodnotě vstupních a výstupních veličin.

Výstupem funkce je vektor **xNew** obsahující predikci odhadu stavových veličin do dalšího kroku.

7.2 PROGRAM PRO ARDUINO

Program v zařízení Arduino Due je napsán v jazyce APL a realizuje vlastní experiment. Je popsán vývojovým diagramem na obr. 7.3.

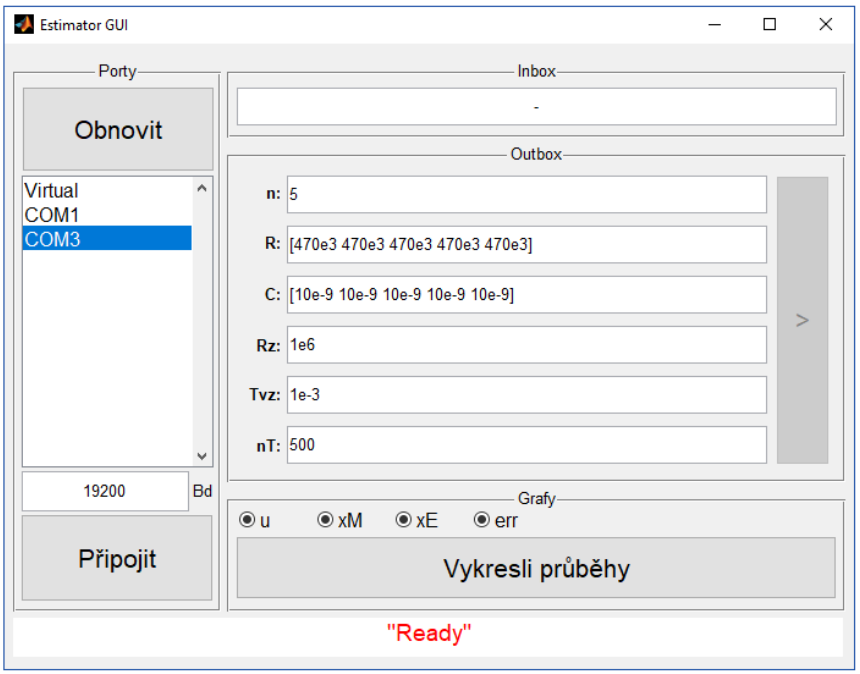


Obr. 7.3 – Program v zařízení Arduino

Části programu vyznačené červeně jsou realizovány implementovanými knihovními funkcemi vygenerovanými programem MATLAB Coder. Zbylé části program jsou realizovány funkcemi jazyka APL.

7.3 MATLAB GUI

GUI naprogramované v prostředí MATLAB slouží k předávání dat mezi deskou Arduino Due a běžným PC, jejich archivaci, zobrazení naměřených a vypočtených hodnot a výpočtu průběžné kvadratické chyby odhadu stavu. V GUI je implementována kontrola zadaných a přijatých dat a dostupnosti zařízení.



Obr. 7.4 – MATLAB GUI

V hlavním panelu GUI (na obr. 7.4) je možné vybrat port a nastavit rychlost komunikace. Po úspěšném zahájení komunikace je zpřístupněna funkcionalita odeslání dat (parametrů RC soustavy) do zařízení, po níž je automaticky zahájeno čtení sériového portu a ukládání přijatých dat do souboru, který je označen časovou známkou. Takto archivovaná data z posledního měření je možné ze souboru načíst a po úspěšné kontrole jejich celistvosti je zobrazit v grafu.

8 VÝSLEDKY MĚŘENÍ

Pro ověření funkčnosti navrženého řešení a zjištění jeho omezení bylo provedeno několik opakovaných měření, při kterých bylo zajištěno dostatečné množství dat pro jejich objektivní vyhodnocení. Oproti standardnímu postupu, kdy by byla nejprve ověřena správnost, bylo v této práci nutné tento postup porušit, neboť správný odhad stavových veličin závisí na správnosti stavového modelu, což pro diskrétní estimátor v důsledku znamená nutnost důsledného dodržení periody vzorkování, pro kterou byl diskrétní stavový model vytvořen. Správnost zkoušených možností byla tedy ověřována souběžně s měřením rychlosti.

8.1 MĚŘENÍ RYCHLOSTI

Pro měření rychlosti vykonávání kódu (doby běhu jednotlivých funkcí) byla využita funkce APL `micros()`. Správnost hodnoty zjištěné touto funkcí byla ověřena kontrolou na osciloskopu Hameg HMO 3522 měřením změny hodnoty napětí generovaného na pinu v režimu digitálního výstupu funkcí `digitalWrite()` ve stejných místech zdrojového kódu. Zjištěný rozdíl (1 až 2 mikrosekundy) je dán dobou provádění samotných funkcí `micros()` a `digitalWrite()` a vzhledem k době provádění kódu je zanedbatelný. Protože čas provádění funkce `micros()` je kratší (měření je tedy přesnější) a její použití je jednodušší, je v této práci upřednostněna k určování rychlosti namísto použití osciloskopu.

8.1.1 Měření doby výpočtu

Byla změřena rychlost provádění kódu estimátoru stavu (výpočet odhadu stavového vektoru funkcí `ardPozorovatel()`) v závislosti na uvažovaném řádu soustavy (druhý až pátý) a optimalizaci kódu (v různých verzích pro konkrétní soustavu nebo obecnou). Tab. 8.1 udává průměrnou délku výpočtu v mikrosekundách.

Tab. 8.1 – Výsledky měření rychlosti

komplexnost	rozměr	datový typ	Řád estimátoru			
			2	3	4	5
real	n × n	float	27	51	80	117
real	n × n	double	40	76	127	182
real	Inf × Inf	float	53	81	117	161
real	Inf × Inf	double	68	107	159	223

Tab. 8.1 – Výsledky měření rychlosti - pokračování

komplexnost	rozměr	datový typ	Řád estimátoru			
			2	3	4	5
complex	n × n	float	78	156	237	379
complex	n × n	double	104	210	335	497
complex	Inf × Inf	float	112	194	303	438
complex	Inf × Inf	double	141	250	393	570

První tři sloupce udávají volitelné parametry programu MATLAB Coder. Komplexnost značí, zda je výpočet prováděn nad množinou reálných nebo komplexních čísel. Datový typ udává typ všech proměnných. Použité jsou `float` (`single`) a `double` (`double`). Rozměr je velikost vektorů a matic, pro které je funkce `ardPozorovatel()` přeložena. Hodnota n × n znamená, že byl překlad proveden pro konkrétní řád estimátoru a rozměry proměnných jsou tak pevně dané. Hodnota Inf × Inf značí, že řád nebyl předem znám, vygenerovaná funkce je realizována obecně pro pozorovatele „až nekonečného“ řádu a očekává na vstupu proměnnou speciálního složeného datového typu, který v sobě informaci o rozměru obsahuje.

Uvedené vyzkoušené varianty pokrývají většinu základních možností nastavení programu MATLAB Coder pro typ vstupní proměnné generované funkce. V rámci této práce nebyly zkoumány dopady např. změny nastavení cílového procesoru nebo důraz na rychlost / čitelnost generovaného kódu.

Uvedené časy jsou pouze dobou potřebnou pro běh kódu výpočtu neznámých stavů (doba provádění funkce `ardPozorovatel()`) a nejsou v nich zahrnuté další úkony prováděné během jednoho cyklu estimace, jako generování a měření napětí na konektorech Arduina, ukládání dat do paměti a další. Proto je minimální možná perioda vzorkování v realizovaném experimentu o něco delší.

8.1.2 Měření doby provádění I/O operací

Společně s rychlostí provádění kódu byla zkoumána i průměrná rychlost měření a generování analogového signálu v zařízení Arduino Due. Byla změřena průměrná doba provádění funkcí `analogRead()` a `analogWrite()` a také doba potřebná pro jejich první volání v programu (spuštění A/D a D/A převodníků), která je několikanásobně větší.

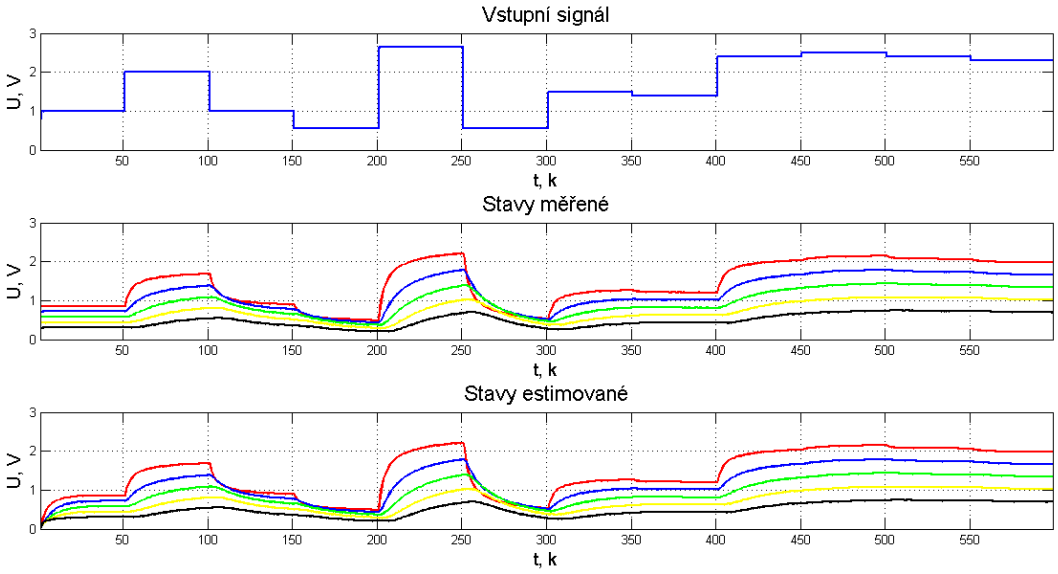
Tab. 8.2 – Rychlost ADC a DAC

funkce	rychlost, µs	první spuštění, µs
`analogRead()`	5	42
`analogWrite()`	4	33

Uvedené časy byly změřeny pro nastavené rozlišení 10 bitů, pro jiná nastavení jsou však totožné, neboť funkce `analogReadResolution()` a `analogWriteResolution()` nemění rozlišení převodníků, nýbrž pouze počet bitů čísla vrácené či zadaného funkcím `analogRead()` a `analogWrite()` (Arduino, 2018b).

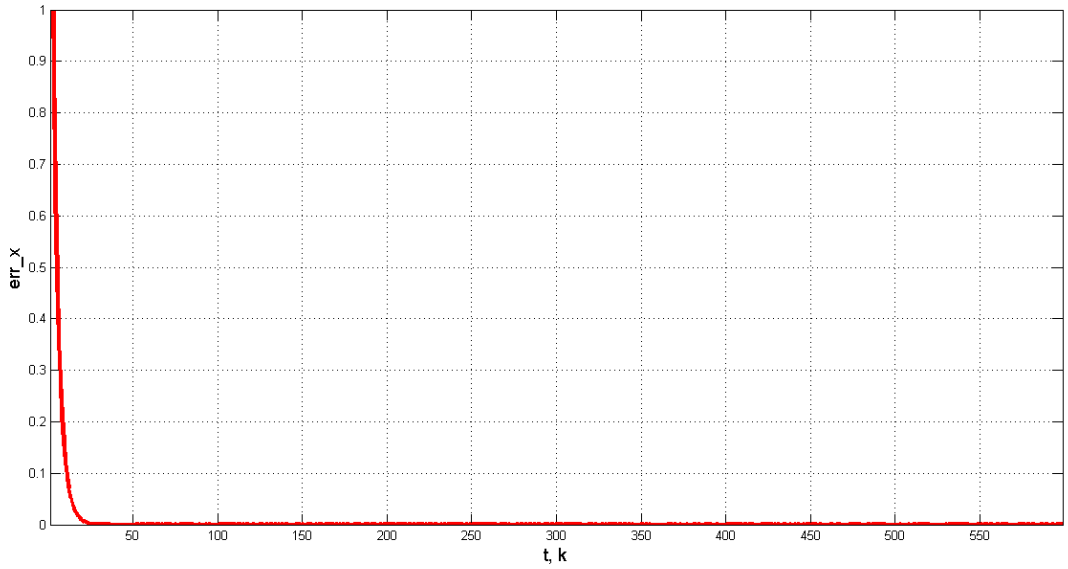
8.2 OVĚŘENÍ SPRÁVNOSTI ESTIMACE

Funkčnost stavového pozorovatele byla ověřena srovnáním naměřených a vypočtených hodnot a výpočtem průběžné kvadratické chyby odhadu, která od vysokých počátečních hodnot rychle konvergovala k nule.



Obr. 8.1 – Ověření správnosti estimace

Na obr. 8.1 je průběh vstupního napětí RC soustavy a průběhy naměřených a vypočtených stavových veličin. Průběžná chyba odhadu vypočtená dle (3.21) je na obr 8.2. Na ose x obou grafů je uvedené číslo vzorku *k*.



Obr. 8.2 – Průběh chyby estimace

8.3 URČENÍ RYCHLOSTI IMPLEMENTACE ESTIMÁTORU STAVU

S využitím získaných znalostí lze provést úvahu o minimální periodě vzorkování stavového pozorovatele implementovaného na zařízení Arduino Due pro obecnou reálnou soustavu. I při využití nejvyššího rozlišení čísla získaného z A/D převodníků zařízení (12 bitů) je rozlišení datového typu `float` (32 bitů) pro výpočty dostatečné a využití typů s větší přesností nemá význam. Nebudou-li při určování zesílení pozorovatele metodou umístění pólů zvoleny póly komplexní, není důvod uvažovat výskyt komplexních čísel ani v jiných veličinách. Řád modelu obecně znám není a bude tedy použita varianta datových typů s neurčenými rozměry. Do periody vzorkování je nutné zahrnout i operace měření vstupních a výstupních veličin soustavy, jejich přepočet dle rovnice zjištěné při kalibraci a předání (zde pouze uložení do paměti) odhadu stavových veličin k dalšímu využití. Pro zvolenou soustavu 5. řádu je tedy minimální použitelná (experimentálně změřená) perioda vzorkování při uvažované konfiguraci zhruba 200 mikrosekund.

9 ZÁVĚR

Byla úspěšně ověřena možnost implementace estimátoru stavu na zařízení Arduino Due s využitím programu MATLAB pro jeho návrh.

V průběhu realizace úlohy byl vytvořen program v Arduino Programming Language, který je spouštěn v zařízení Arduino Due a s využitím statické knihovny v jazyce C vygenerované programem MATLAB Coder umožňuje výpočet parametrů stavového pozorovatele a následný cyklický výpočet odhadu aktuálního stavového vektoru. Knihovna je vygenerována na základě funkcí napsaných v jazyce MATLAB a funkce v ní obsažené umožňují: sestavení spojitého modelu soustavy, diskretizaci tohoto modelu, určení zesílení pozorovatele a jeho vlastní realizaci. K předávání dat mezi uživatelem a programem v zařízení Arduino Due bylo vytvořeno GUI v prostředí MATLAB, které implementuje i základní kontrolní funkce zadaných parametrů a přijatých dat.

Funkčnost řešení byla ověřena experimentem na soustavě několikanásobného RC článku, kdy zároveň byla měřena doba provádění výpočtu odhadu stavu a celková doba nutná k provedení jednoho cyklu estimace, který dále sestává z měření hodnot na vstupu a výstupu soustavy a archivace získaných dat. Na základě získaných informací byla stanovena minimální perioda vzorkování, čímž byly splněny všechny cíle této práce.

Vypracované řešení je do značné míry obecné a modulární a je tedy dále využitelné jako základ pro realizaci stavového řízení na zařízení Arduino Due nebo jiné platformě umožňující běh programu napsaného v jazyce C/C++.

Výhodou navrženého způsobu řešení je možnost implementace kvalitně zpracovaných a často velmi složitých algoritmů z prostředí MATLAB na embedded procesorech bez nutnosti porozumění jejich obsahu či mechanického přepisování do jazyka C/C++. Tuto výhodu lze uplatnit i u vlastních algoritmů, které je možné nejprve zapsat a otestovat v prostředí MATLAB s využitím veškerého komfortu, které toto prostředí nabízí, a teprve poté je spolehlivou metodou převést do cílového kódu v jazyce C/C++.

POUŽITÁ LITERATURA

ARDUINO, 2017. *Getting started with the Arduino Due* [online]. [cit. 2018-04-20]. Dostupné z: https://www.arduino.cc/en/Guide/ArduinoDue

ARDUINO, 2018a. *Arduino Products* [online]. [cit. 2018-05-11]. Dostupné z: https://www.arduino.cc/en/Main/Products

ARDUINO, 2018b. *Language Reference* [online]. [cit. 2018-05-12]. Dostupné z: https://www.arduino.cc/reference/en/

ARDUINO STORE. 2018. *Arduino Due* [online]. [cit. 2018-04-20]. Dostupné z: https://store.arduino.cc/arduino-due

BALÁTĚ, J. 2003. *Automatické řízení*. Praha: BEN - technická literatura, ISBN 80-7300-020-2

BUREŠOVÁ, L. 2017. *Úvod do Kalmanova filtru* [online]. [cit. 2018-05-13]. Dostupné z: http://docplayer.cz/38663649-Uvod-do-kalmanova-filtru.html

BURÝ, A. 2011. *Modelování a simulace dynamických systémů* [online]. Ostrava: Vysoká škola báňská - Technická univerzita. [cit. 2016-03-15]. Dostupné z: homen.vsb.cz/~neu10/.../Modelovn_a_simulace_dynamickch_systm.pdf

DUŠEK, F. 2012. *Pozorovatel stavu stavového modelu*. Pardubice: skripta Univerzity Pardubice

HLAVA, J. 2012. *Diskretizace spojitých systémů* [online] Liberec: skripta Technické univerzity v Liberci [cit. 2018-04-20] Dostupné z: www.fm.tul.cz/esf0247/index.php?download=668

HUMUSOFT, 2018. *Aplikační knihovny MATLABu a Simulinku* [online]. [cit. 2018-04-20]. Dostupné z: http://www.humusoft.cz/matlab/products/

MATLAB, 2018a. *MATLAB Coder* [online]. [cit. 2018-05-13]. Dostupné z: https://www.mathworks.com/products/matlab-coder.html

MATLAB, 2018b. *Documentation* [online]. [cit. 2018-05-13]. Dostupné z: https://www.mathworks.com/help/

OGATA, K. 1995. *Discrete-time control systems*. 2nd ed. Englewood Cliffs, N.J.: Prentice Hall. ISBN 0-13-034281-5.

ŠEBEK, M. 2016. *Pozorovatel a výstupní ZV* [online]. Praha: skripta Českého vysokého učení technického v Praze. [cit. 2018-04-20] Dostupné z: http://www.polyx.com/_ari/slajdy/Bas-ARI-16-Observer.pdf

ŠTECHA, J; HAVLENA, V. 1999. *Moderní teorie řízení* [online]. Praha: Editační středisko ČVUT, skripta Českého vysokého učení technického v Praze. [cit. 2018-05-13] Dostupné z: http://www.kirp.chtf.stuba.sk/~cirka/source/mtr_book.pdf

TUREK, M. 2007. *Stavové řízení* [online]. Brno: skripta Vysokého učení technického v Brně. [cit. 2018-04-20] Dostupné z: http://matlab.fei.tuke.sk/orhs/subory/podklady/stavove_rizeni_Matlab_LQ.pdf

VÍTEČKOVÁ, M; VÍTEČEK, A. 2016. *Stavové řízení* [online]. Ostrava: skripta Vysoké školy báňské - Technické univerzity Ostrava, 101 s. ISBN 978-80-248-3900-4. Dostupné z: http://books.fs.vsb.cz/ZRMS/stavove-rizeni.pdf

VROŽINA, M; JANČÍKOVÁ, Z; DAVID, J. 2012. *Identifikace systémů* [online]. Ostrava: Vysoká škola báňská - Technická univerzita. [cit. 2016-03-21]. ISBN 978-80-248-2594-6. Dostupné z: http://www.person.vsb.cz/archivcd/FMMI/IS/Identifikace%20systemu.pdf

Základní postupy při modelování a identifikaci. 2006. *Řízení technologických procesů* [online]. [cit. 2016-04-10]. Dostupné z: http://rtp.webzdarma.cz/model2.php#identifikace

PŘÍLOHY

A - CD

Příloha A

Příloha k diplomové práci

RYCHLOST IMPLEMENTACE ESTIMÁTORU STAVU V ZAŘÍZENÍ ARDUINO DUE

Jan Vojta

CD

Obsah

1 Text diplomové práce ve formátu PDF

2 Úplné zdrojové kódy vygenerované knihovny v jazyce C/C++ včetně zdrojového kódu originálních funkcí v jazyce MATLAB

3 Úplné zdrojové kódy programu pro zařízení Arduino Due v jazyce Arduino Programing Language

4 Úplné zdrojové kódy aplikace s grafickým uživatelským rozhraním v jazyce MATLAB pro komunikaci se zařízením Arduino Due