ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ

Fakulta strojní

Ústav přístrojové a řídící techniky

# Inovace laboratorního modelu Soustava tří nádob

Bakalářská práce

Jonáš Cikhart

Bakalářský program: Teoretický základ strojního inženýrství

Vedoucí: Ing. Pavel Trnka, Ph.D.

Praha, leden 2020

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Cikhart** Jméno: **Jonáš** Osobní číslo: **467339**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídicí techniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Inovace laboratorního modelu Soustava tří nádob**

Název bakalářské práce anglicky:

**Inovation of laboratory model Three tanks**

Pokyny pro vypracování:

1) Popište současný stav řídicího sw laboratorní úlohy Soustava tří nádob, a to jak na straně řídicího PLC (Rex Controls), tak na straně vizualizačního PC (Matlab Simulink).
2) Vytvořte nové uživatelské rozhraní pro řešení tří typů laboratorních cvičení předmětu Automatické řízení (logické řízení, frekvenční vlastnosti, uzavřený regulační obvod).
3) Navrhněte úpravy úlohy umožňující využití průtokoměrů zabudovaných do laboratorního modelu a pokuste se navržené řešení realizovat.

Seznam doporučené literatury:

[1] Produktová dokumentace Rex. Dostupné z: https://www.rexcontrols.cz/clanky/nova-verze-nove-jmeno
[2] Vrána, Stanislav. Frekvenční charakteristika soustavy tří nádrží. Dostupné z: http://vlab.fs.cvut.cz/navody/files/kaskada_frekvencni.pdf
[3] Vrána, Stanislav. Logické řízení výšky hladiny v nádržích. Dostupné z: http://vlab.fs.cvut.cz/navody/files/kaskada_logicka.pdf
[4] Vrána, Stanislav. Laboratorní úloha Seřízení PI regulátoru. Dostupné z: http://vlab.fs.cvut.cz/navody/files/kaskada_serizeniPI.pdf

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**Ing. Pavel Trnka, Ph.D., U12110.3**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **31.10.2019** Termín odevzdání bakalářské práce: **17.01.2020**

Platnost zadání bakalářské práce: _____

| Ing. Pavel Trnka, Ph.D.<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Ing. Michael Valášek, DrSc.<br>podpis děkana(ky) |
|---|---|---|

## III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

31.10.2019
Datum převzetí zadání

Podpis studenta

*Název práce:* **Inovace laboratorního modelu Soustava tří nádob**

**Anotace**

Cílem této práce je popsat současný stav laboratorního modelu Soustava tří nádob. Seznámit čtenáře se softwarovým řešením tohoto modelu, respektive jeho třech částí, a to jak na úrovni řídícího PLC v softwaru REX Controls, tak na straně vizualizační v softwaru MATLAB Simulink. Mezi dílčí cíle patřilo i mé seznámení s používaným softwarem. Dále pak navrhnout vlastní uživatelské prostředí určené pro studenty pracující na modelu, použitelné pro ovládání jednotlivých úloh. Nakonec se zamyslet a navrhnout použití vrtulkového průtokoměru v modelu a v případě možností tento návrh i realizovat.

*Klíčová slova:* laboratorní model, PLC, REX Controls, MATLAB Simulink, uživatelské prostředí, frekvenční vrtulkový průtokoměr

*Title:* **Inovation of laboratory model Three tanks**

**Abstract**

The aim of this work is to describe the current state of the laboratory model Three tanks. Also to acquaint the reader with software solution of this model, or more so its three parts, both at the level of the control PLC in the REX Controls software and at the level of visualization in the MATLAB Simulink software. Partial objectives were also to get familiour with the used software. Furthermore, to design my own user interface designed for students working on the model, which is usable for controlling individual tasks. Finally, consider and propose the use of the propeller flowmeter in the model and possible implementation of this proposition.

*Key words:* laboratory model, PLC, REX Controls, MATLAB Simulink, user interface, frequency propeller flowmeter

# Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze kterých jsem čerpal.

V Praze, dne 17.1.2020 ……

Jonáš Cikhart

# Poděkování

Děkuji všem lidem kolem mě, kteří mě podporovali při psaní této bakalářské práce. Osobně bych pak chtěl poděkovat mému vedoucímu Ing. Pavlovi Trnkovi, Ph.D., za jeho čas a chuť mi radit a za jeho připomínky k mé práci.

# Obsah

# Seznam použitých zkratek

IP – Internet Protocol

IPC – Industrial PC

PLC – Programmable Logic Controller

RDC – Remote Desktop Control

REX – Rapid Development Excellent

WPC – ovladač v RexDraw pro jednotku WinPAC

# 1 Úvod

Mým hlavním cílem při zpracovávání této práce bylo seznámení s reálným projektem, na kterém si mohu vyzkoušet základy teorie, které jsem se naučil během bakalářského studia. Zjistit, kde bývají hlavní problémy, při uvádění takovéhoto modelu zpět do provozu a zároveň co vše je k tomu potřeba. Od této práce jsem očekával možnost získání znalostí jak z hardwaru, tak softwaru.

## 1.1 Struktura práce

Práce je rozdělena na 6 hlavních části. Tyto části reflektují, v jaké časové posloupnosti jsem se s úlohou seznamoval a řešil jí. Zároveň tvoří logické celky, které jsou reprezentovány pomocí hlavních kapitol.

V kapitole **Současný stav modelu** popisuji stávající stav a hardwarové řešení laboratorního modelu Soustava tří nádob. Trochu více se zabývám řídící jednotkou, která je v modelu použita.

V druhé a třetí části popisuji software, který je na modelu použitý. Jedná se o **REX Control a MATLAB Simulink**. Zmiňuji se o základní struktuře systémů, uživatelském prostředí a používání. Především u Simulinku se zabývám více částmi, které jsem v úloze sám použil.

Ve čtvrté části **Původní řešení jednotlivých úloh** popisuji současné řešení laboratorních úloh na úrovni řízení a vizualizace v softwarech představených v druhé a třetí části. Přesněji se jedná o úlohy logické řízení, frekvenční vlastnosti a regulační obvod.

V páté části **Vytvoření nového uživatelského prostředí** řeším vytvoření nového ovládacího prostředí v programu MATLAB Simulink, použitelného k ovládání modelu, pomocí připojeného stolního počítače. Ovládání by mělo být uzpůsobené pro zmíněné úlohy v předmětu automatické řízení.

V šesté části **Frekvenční průtokoměr** se zabývám uvedením vrtulkového frekvenčního průtokoměru do provozu, a to jak hardwarově, tak softwarově. Dále pak navrhuji použití průtokoměru v modelu a zároveň se zamýšlím nad realizací některých z mých návrhů.

# 2 Současný stav modelu

## 2.1 Obecný popis

Úloha se skládá ze tří válcových nádob, které jsou vzájemně v dolní části spojeny přes kulové ventily. Zároveň jsou z nádrží, také přes kulové ventily, vyvedeny odtokové gumové trubičky. Voda do nádob je čerpána pomocí tří čerpadel, z nichž jedno je hlavní a dvě vedlejší [1]. Do nádrže 1 je čerpána voda pomocí hlavního čerpadla a jednoho z vedlejších čerpadel. Druhé vedlejší čerpadlo čerpá vodu do nádrže 3. Každé z čerpadel má k sobě ještě připojen operační zesilovač, který zajišťuje stabilitu napětí na čerpadlech. Do každé z nádob jsou nainstalovány tlakoměry, zavěšené na horní hraně desky. Tyto tlakoměry fungují na principu, že rostoucí hladina vody stlačuje vzduch ve svislé kovové trubičce, která má otvor těsně nad dnem nádoby. Druhá strana této trubičky je připojena k tlakoměru. Tlakoměr nádoby 2 je mechanicky poškozený, proto bude muset být nahrazen novým. Analogové vstupy jsou připojeny do průmyslové řídící stanice WinPAC WP-8841, o které se více zmiňuji v kapitole **2.2**. Potřebné prvky úlohy jsou napájeny pomocí dvojitého laboratorního zdroje *Diametral P230R51D*.


*Obr. 1: Laboratorní model Soustava tří nádob*

## 2.2 WinPAC WP-8841

WP-8841 je modulární řídící jednotka, která v sobě obsahuje operační systém Windows CE 5.0. Variantní připojení umožňuje spojení pomocí ethernetu, dále připojení RS-232/485, což jsou rozhraní pro digitální přenos dat po sériové lince [7] [8]. Dále lze pak přímo k jednotce připojit monitor skrze VGA koncovku a klávesnici s myší přes USB port. Takto ji lze následně ovládat přímo ve zmíněném operačním systému. V mém případě byla ovšem využita možnost komunikace přímo ze stolního počítače, což je pochopitelně uživatelsky pohodlnější. WP-8841 umožňuje připojení až 8 paralelních zásuvných modulů, které se dají jednoduše přidávat zasunutím do zadní strany jednotky. Každý z modulů má navíc vyjímatelnou svorkovnici, což se vzhledem k přístupnosti zadní strany modelu ukázalo velmi užitečné.

V současném stavu je v jednotce 5 zásuvných paralelních modulů:

- i-8017HW – 8 analogových vstupů
- i-8024W – 4 analogové výstupů
- i-8064W – 8 digitálních výstupů
- i-8084W – 4/8 digitálních vstupů (podle nastavení modulu)
- i-8084W – 4/8 digitálních vstupů (podle nastavení modulu)


*Obr. 2: Paralelní zásuvné moduly*

# 3 Software REX Controls

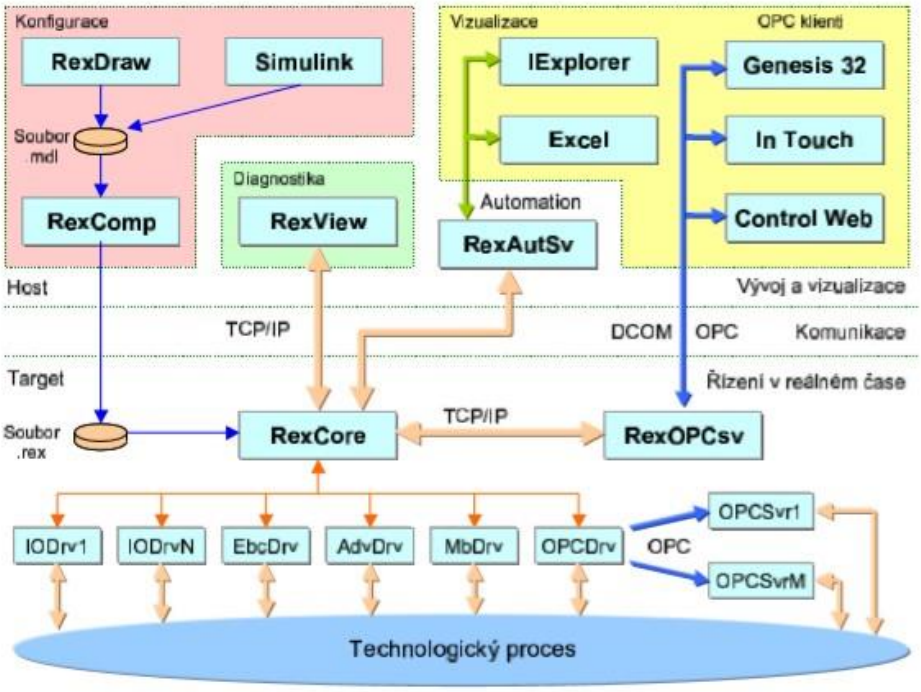
V našem použitém modelu, se pracuje se starší verzí programu, přesněji REX Controls v1.42. Některé pojmenování se proto nyní liší, ale celkový koncept programu zůstává stejný. Na svůj počítač se mi ovšem tuto starší verzi povedlo stáhnout z archivu, tudíž nebyl problém soubory upravovat a vyměnit za mnou vytvořené.

## 3.1 Základní informace

REX je řídící systém, který umožňuje zrychlovat a usnadňovat instalaci a uvedení řídících procesů pro různé technologické procesy do provozu. Jedna z nejdůležitějších věcí, které pomáhají k právě zmíněnému urychlování uvedení do provozu je možnost systém odsimulovat dopředu a tím výrazně snížit šanci možných problémů při samotné instalaci. REX proto velmi úzce spolupracuje se softwarem Matlab-Simulink, který je jedním z nejrozšířenějších nástrojů pro vývoj a testování nových algoritmů. [4] [5]

## 3.2 Struktura řídícího systému REX

Celková struktura řídícího systému REX je poměrně komplexní, proto ji zde přiblížím za pomoci obrázku Obr. 3. Při mojí reálné práci s modelem Soustava tří nádob jsem se ovšem setkal pouze s některými z ukázaných částí. Převážně se jednalo o grafický editor RexDraw, kompilátor RexComp, RexView a nepřímo jsem přišel do styku i s jádrem systému RexCore.

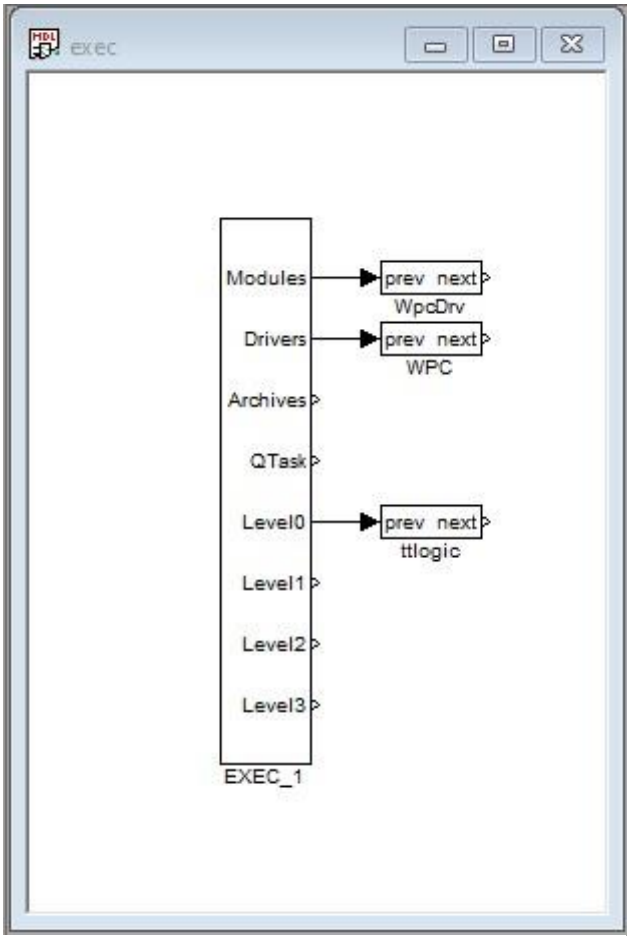



*Obr. 3: Celková struktura řídícího systému REX [5]*

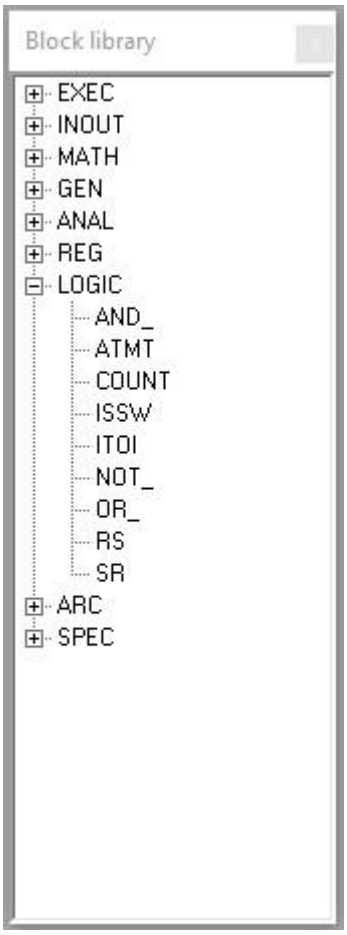
### 3.2.1 RexDraw

RexDraw je grafický editor, který umožňuje navrhovat funkční schémata řídícího systému REX. Za použití bloků, které jsou obsaženy v rozsáhlé knihovně systému REX [6], se vytváří finální soubor, který je vygenerován ve formátu *.mdl*. Tyto bloky pracují v diskrétním čase a jejich velká část je automaticky diskreditována pro danou periodu vzorkování.

Na rozdíl od Matlab-Simulink, kde celá konfigurace může být tvořena pomocí jediného souboru, který obsahuje různé podsystémy, v systému je REX konfigurace tvořena pomocí alespoň dvou souborů, z nichž jeden je tzv. hlavním souborem projektu (obvykle exec.mdl). Hlavní soubor projektu specifikuje konfiguraci jednotlivých úloh, ovladačů, priorit, časování a další [5]. [4]




*Obr. 4: Hlavní soubor exec.mdl logické úlohy tří nádob*




*Obr. 5: Knihovna bloků v prostředí RexDraw*

### 3.2.2 RexComp

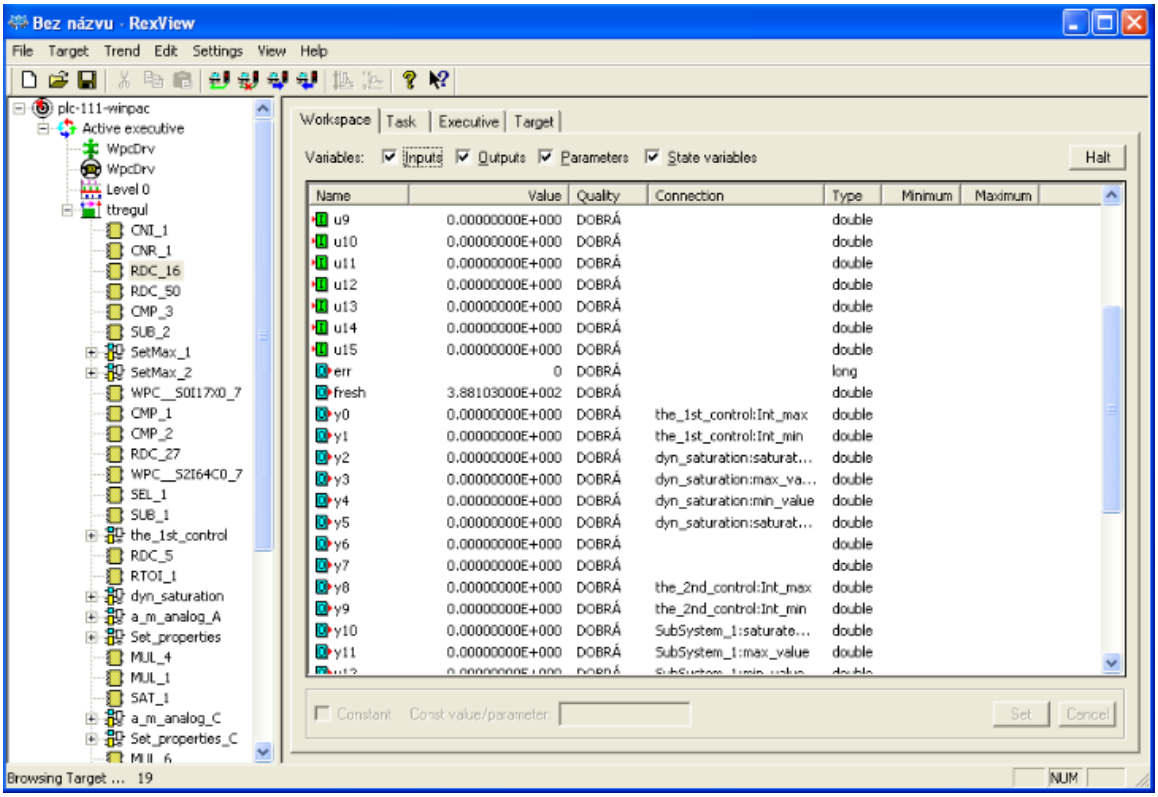
RexComp je část struktury REX, která se stará o překlad konfigurací. Překládá zdrojový soubor, který obdrží ve formátu .mdl na binární konfigurační soubor *.rex* řídícího systému REX.

Překlad konfigurace se skládá ze sedmi po sobě jdoucích kroků [5]:

1) Nalezení bloku exekutivy reálného času EXEC v hlavním souboru projektu, kontrola parametrů exekutivy.
2) Nalezení a kontrola všech objektů řídícího systému, které jsou nakresleny v hlavním souboru projektu.
3) Přidání jednotlivých objektů do konfigurace: modulů, ovladačů, archivů, rychlé úlohy a úloh jednotlivých použitých výpočetních úrovní.
4) Alokace paměti a nastavení parametrů bloků z konfigurace.
5) Kontrola a propojení řídících úloh.
6) Kontrola správnosti celé konfigurace.
7) Uložení přeloženého souboru s příponou *.rex* na disk.

### 3.2.3 RexView

RexView je program, který slouží ke sledování dějů v jádře systému REX v reálném čase. To je obzvláště důležité při uvádění procesů do provozu, případného vzniku chyb nebo při řešení problému v již zavedeném systému. Zároveň umožňuje ruční nastavení proměnných, což je velmi užitečné při testování.




*Obr. 6: Zobrazení proměnných v programu RexView*

### 3.2.4 RexCore

RexCore je jádro řídícího systéme REX. Na rozdíl od doposud zmíněných částí v kapitolách 3.2.1 až 3.2.3 běží RexCore na cílovém zařízení. V našem modelu se jedná o WP-8841, ale obecně to může být PC, IPC (Industrial PC) nebo například WinCon. Toto jádro provádí paralelně několik činností, obvykle v řídících systémech. Tyto činnosti jsou vykonávány na základě priorit v režimu preemptivního multitaskingu pomocí jednotlivých subsystémů jádra [5].

Jádro obsahuje tyto subsystémy [5]:

- Subsystém reálného času – stará se o spouštění jednotlivých úloh a v nich vložených funkčních bloků, řídí spouštění ovladačů, získává a poskytuje diagnostické informace o časování úloh a ovladačů a vytížení systému.

- Vstupně-výstupní subsystém – poskytuje rozhraní pro ovladače technických prostředků pro získávání vstupů z procesu a nastavování výstupů.

- Algoritmický subsystém – obsahuje algoritmy funkčních bloků, které jsou volány z úloh subsystému reálného času.

- Diagnostický subsystém – poskytuje diagnostické informace o běhu řídicího systému, umožňuje download a ladění aplikace.

- Archivační subsystém – slouží pro archivaci událostí, alarmů a historických trendů veličin řídicího systému.

# 4 Software MATLAB Simulink

Počítač, na kterém se model řídí, má nainstalovanou verzi R2006a programu MATLAB. Ta je velmi stará a nenabízí tudíž tolik možností jako novější verze R2019a, na kterou je poskytnutá licence z ČVUT, a kterou jsem používal já. Při uvádění nových souborů, které jsem vytvořil, do provozu jsem se musel potýkat s problémy kompatibility obou verzí. Problém se po konzultaci s vedoucím této bakalářské práce nakonec s největší pravděpodobností vyřeší nainstalováním nové verze programu na tento počítač.

## 4.1 Základní informace

MATLAB Simulink je stejně jako RexDraw prostředí, které využívá blokových schémat při sestavování systémů. Podporuje tzv. „system-level design“, simulace, automatické generování kódu nebo nepřetržitou možnost testování a ověřování vestavěných systémů. Součástí Simulinku je grafický editor, upravitelná knihovna bloků, řešiče pro modelování, simulaci a analýzu lineárních nebo dynamických systémů. Jednotlivé bloky mohou reprezentovat části systému, fyzický komponent, subsystém nebo například funkci [10].

Důležitou součástí Simulinku je to, že je integrován s MATLABEM, což umožňuje zahrnout do modelů matlabovské algoritmy, případně exportovat výsledky simulací zpět přímo do MATLABU pro další zpracování, ať už se jedná o další výpočty, nebo zanesení do grafů, nebo databází [9].

## 4.2 Používání

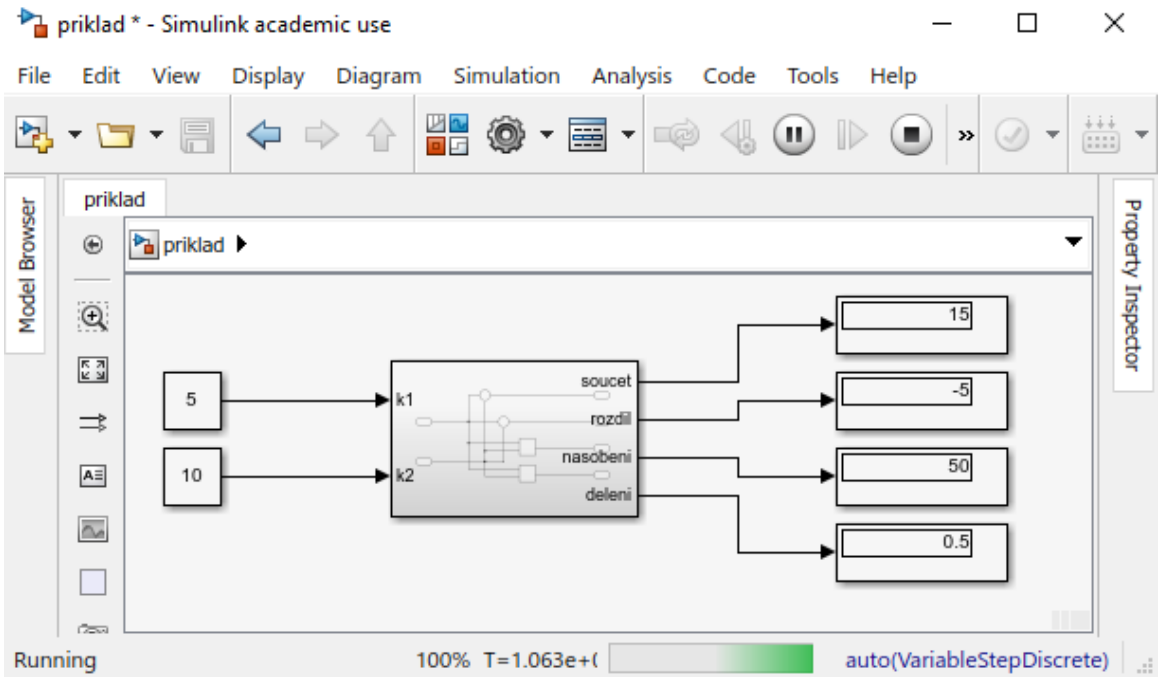
Základem při používání softwaru MATLAB Simulink je vytvoření simulačního modelu procesu v grafickém editoru. Ten se může skládat jen z jednotlivých bloků, které mohou tvořit nějaké například logické podmínky podle vstupů, nebo spojité řízení. Většinou je ale vhodnější použít, kvůli větší komplexnosti sytému, ke klasickým blokům ještě subsystémy, které stručně popíši v následující podkapitole **4.2.1**.

Při sestavování simulačního modelu procesu se z knihovny vyberou potřebné bloky, kterých je v Simulinku opravdu veliké množství. K základním se dají navíc ještě doinstalovat různá rozšíření, ve kterých jsou bloky pro více specifické účely. Jako příklad bych uvedl rozšíření, které jsem já sám použil. Do Simulinku přidá mimo jiné blok RDC, který je potřebný pro komunikaci s řídící jednotkou WinPac, respektive pro komunikaci s RexDraw. Instalační soubor této knihovny je možno najít v příloze této bakalářské práce pod názvem RL_2_10_ML8_0_x64.

Jednotlivé bloky se mezi sebou propojují standartně jako v ostatních grafických prostředích pomocí spojovacích čar, které vedou z výstupních portů bloků do vstupních. Většina z bloků má ještě vnitřní nastavení, ve kterém se mohou nastavit jeho parametry. Po sestavení je následující krok spuštění simulace, což je možné jak za pomoci tlačítka v prostředí Simulinku, tak pomocí matlabovského skriptu. Po spuštění simulace začnou jednotlivé bloky vykonávat svojí funkci, ať už se jedná o posílání nebo příjímání hodnot na výstupních a vstupních portech, nebo vykonávání nějaké vnitřní funkce definované v příslušném bloku. Simulace jako taková má své vlastní nastavitelné parametry, z nichž bych jako hlavní zmínil vzorkovací periodu určující, jak často dojde k aktualizaci celého modelu a jako druhý parametr čas, po který má simulace probíhat. Ten může být dán konkrétní hodnotou v sekundách, nebo lze simulaci nastavit hodnotou *inf*, aby probíhala, dokud nedojde k ručnímu ukončení, nebo pomocí nějakého scriptu.

## 4.3 Subsystém

Subsystém je skupina bloků, která je seskupena do jednoho obecného bloku, který tento subsystém reprezentuje. Svojí strukturou má definovaný počet vstupů a výstupů. Názorná ukázka je vidět na Obr 7. Jedná se o velmi základní subsystém, který má za úkol přiblížit fungování subsystémů. Má dva vstupní porty, do kterých přijme dvě konstanty. Ve vnitřní struktuře dojde postupně k sečtení obou z konstant, odečtení druhé z konstant od první, jejich vynásobení a vydělení první z konstant tou druhou. Z toho je zřejmá výhoda subsystémů, jenž spočívá ve značném zjednodušení a uspořádání celkové simulace.




*Obr. 7: Ukázkový subsystém*

### 4.3.1 Maska a callbacky

Důležitým prvkem subsystémů je i možnost vytvoření a editace tzv. masky a callbacků. Maska subsystému umožňuje měnit vzhled příslušného bloku, ať už se jedná o obrázek nějaké ikony, jeho barvu nebo popis a zobrazení hodnoty některé z používaných proměnných. Druhým důležitým prvkem masky je menu, které vyskočí při otevření daného subsystému. V něm se dá nastavit definice některých proměnných subsystému uživatelem. Vzhled masky je opět možno konfigurovat za pomoci matlabovských příkazů, tudíž je možné ho měnit na základě různých podmínek a vnitřních proměnných, které se během simulace můžou měnit.

Callbacky bloků umožňují definovat, co se má s blokem stát v případě, že nastane některá z předem nadefinovaných situací a umožňuje tak změnit například vzhled. Jako příklad bych uvedl funkce *OpenFnc* nebo *MoveFnc,* které se spustí v případě, že je blok otevřen, respektive se s ním v prostředí Simulinku pohne.

Možnosti nastavení masky přiblížím na subsystému *Regulator* v kapitole **6.3.1***,* který jsem vytvořil pro uživatelské prostředí u uzavřeného regulačního obvodu. Princip callbacků pak přiblížím na příkladu, který jsem vytvořil pro uživatelské prostředí u logické úlohy. Jedná se o subsystém s názvem *Prepinac* v kapitole **6.1.1**.

# 5 Původní řešení jednotlivých úloh

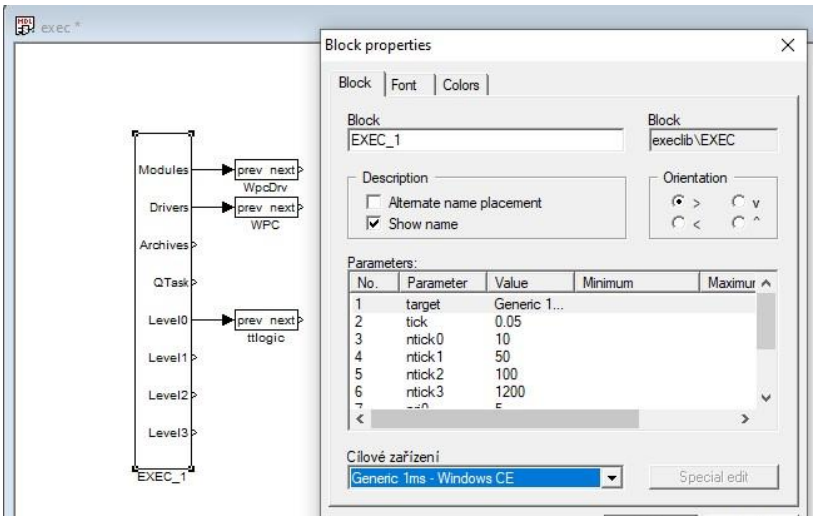
Oba softwary představené v kapitolách **3** a **4** byly na modelu Soustava Tří Nádob použity ve všech třech úlohách, proto jsem se rozhodl úlohy Logické řízení, Frekvenční vlastnosti a Uzavřený regulační obvod rozebrat postupně. Popíší jejich současnou formu a pokusím se přiblížit jednotlivé soubory a jejich funkci. Pro porovnání budou fotky původních řešení v obou softwarech v přílohách této bakalářské práce.

Ve všech třech úlohách se využívají některé stejné funkční bloky. Proto ty nejdůležitější představím v úvodní kapitole a v jednotlivých popisech úloh se na ně v případě potřeby jen odkážu. Především se jedná o hlavní soubor projektů v RexDraw a bloky zajišťující komunikaci s RexDraw jak mezi řídícím PLC, tak mezi Simulinkem. Nejzákladnější bloky jako je násobení apod. jsem vynechal.

## 5.1 Důležité bloky

### Hlavní soubor projektu v RexDraw

Hlavní soubor určující konfiguraci, viditelný na Obr. 8, má tři použité prvky, a to *Modules, Drivers a Level0.* Blok *WpcDrv* vystupující z *Modules* je jedním z modulů řídícího systému REX. Společně s prvkem Drivers určuje, jaký ovladač je použit, což je velmi důležité pro následnou komunikaci s PLC. Blok *WPC*, který vystupuje z *Drivers,* je blokem vstupně výstupního ovladače, jehož jedním z parametrů je jméno modulu v předchozím bloku, který implementuje daný ovladač. Další parametr je jméno konfiguračního souboru ovladače. Jeho hlavní funkcí je především zjišťovat odkud se mají číst hodnoty vstupů a stejně tak určovat, kam se mají zapisovat hodnoty výstupů. *Level0* určuje rychlost vzorkování. V tomto případě je na něj připojen soubor *ttlogic*, ale u zbylých dvou popisovaných úloh je to analogické. Tento soubor je druhý ze souborů tvořící celkový systém. Na rozdíl od hlavního souboru, tento určuje přímo konfiguraci řídící úlohy, nikoliv komunikace, či výběr ovladačů apod. Na pravé straně Obr. 8 jsou vidět násobky původního taktu jednotlivých levelů v milisekundách, kde je vidět, že s rostoucím levelem se zvyšuje i čas mezi jednotlivými takty. [4] [5]




*Obr. 8: Hlavní soubor projektu exec.mdl včetně vyskakovacího menu*

OCTIN, OCTOUT

Jedná se o bloky zajišťující čtení vstupu, respektive zapisování výstupů z řídící nebo do řídící jednotky. To je patrné již z názvu, kde *Oct,* respektive *Quad* reprezentuje počet vstupů nebo výstupů, se kterými blok pracuje (osm, respektive čtyři) a *IN* a *OUT,* jestli se jedná o jejich čtení, respektive zapisování. Komunikace je zajištěna pro mě poměrně netradičně, ale zároveň si dovolím tvrdit elegantně, a to přímo pomocí názvu bloku. Jméno musí být přesně napsané podle požadavků, protože každý z použitých symbolů v něm určuje jeden z parametrů, které se při komunikaci využívají.

WPC__S0I17X0_7

- WPC – Označuje použitý driver pro komunikaci.
- __ – Dvě podtržítka jsou použita jako povinný oddělovač.
- S0 – S označuje zkráceně slot a následné číslo určuje reálné místo zasunutí modulu v jednotce.
- I17 – I označuje, že popisujeme právě zásuvný modul a dvojčíslí za ním je poslední dvojčíslí z označení modulu (toto je dobře vidět na Obr. 2, v tomto případě i-8017).
- X – Toto písmeno označuje tzv. typ vlajky neboli v jakém napěťovém rozsahu případně proudovém rozsahu se pohybujeme (viz Obr 9.).
- 0_7 – Tato čísla určují, s jakými pracujeme vstupy nebo výstupy. Tento zápis udává rozsah hodnot od 0 do 7, ale je možné použít i jednotlivá čísla, např. 0. Načítání více vstupů najednou je ovšem rychlejší a přehlednější varianta.

1. $\pm 10\ V$ vlajka typu `X` (extra large)
2. $\pm 5\ V$ vlajka typu `L` (large)
3. $\pm 2.5\ V$ vlajka typu `M` (medium)
4. $\pm 1.25\ V$ vlajka typu `S` (small)
5. $\pm 20\ mA$ vlajka typu `C` (current)

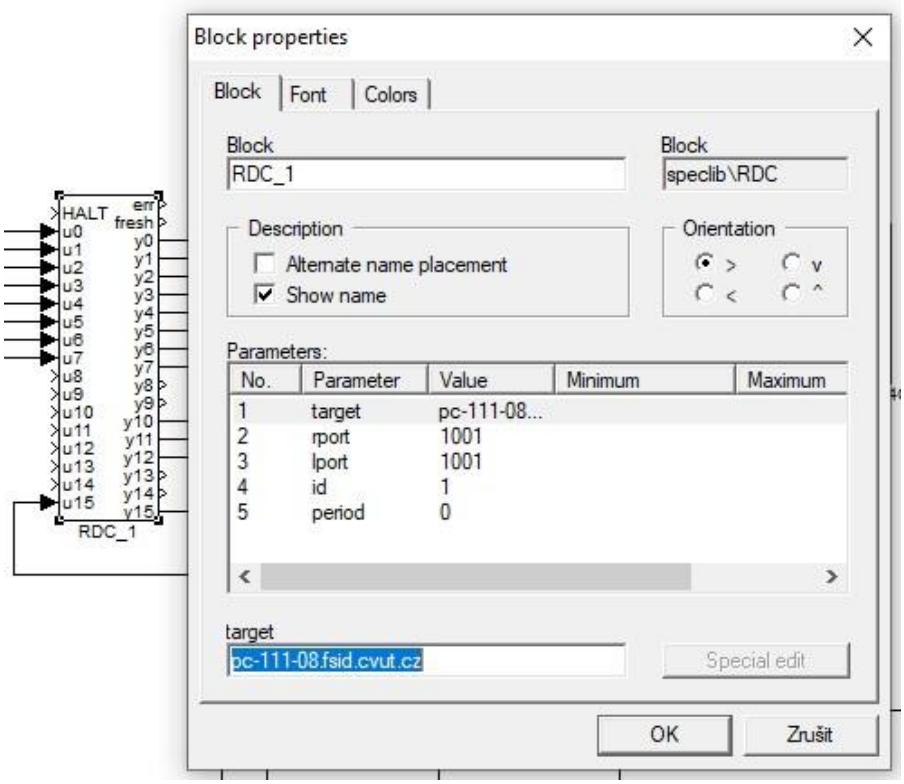
*Obr. 9: Typy vlajek pro vstupní bloky [11]*

QUADOUT

Blok *QUADOUT* s názvem WPC__S1I24V0_3 nám říká, že se jedná o komunikaci pomocí driveru WPC, paralelní zásuvný modul je ve slotu číslo 1, jeho označení je i-8024 a zapisujeme do všech čtyř výstupů. Rozdíl je tady v použité vlajce, jelikož se jedná o výstupní blok, respektive modul. Každý kanál lze totiž použít buď jako proudový s rozsahem ± 20 *mA*, nebo napěťový s rozsahem ± 10 *V* [11]. Proudové mají na pozici pro označení vlajky písmeno I, napěťový je označen pomocí V, což je i tento případ. Pokud by se za běhu vyskytla hodnota mimo dané meze, ovladač ji automaticky zaokrouhlí na krajní hodnotu. Z tohoto je vidět, že u každého modulu je důležité se podívat na jeho konkrétní specifikaci v příručce [11] a podle toho komunikaci nastavovat.

RDC

Tento blok, jehož zkratka představuje *Remote Desktop Control,* je blok určený pro komunikaci se Simulinkem. Funguje tak, že vstupy, které jsou načteny do hodnot u0 až u15 jsou poslány do druhého bloku RDC. To může být v RexDraw, ale i v Simulinku, na výstupy y0 až y15. Přenos dat je tedy obousměrný. Komunikace je vidět na Obr. 10, kde bych vyzdvihnul řádky *target a id. Target* určuje doménové jméno počítače, případně jeho ip adresu, na kterém Simulink běží a *id* určuje, se kterým blokem RDC se má ten původní spárovat. Hodnoty *id* musí být logicky u obou použitých bloků stejné.




*Obr. 10: Blok RDC s otevřeným menu*

## 5.2 Logické řízení

### 5.2.1 Zadání

V této úloze jde o logické řízení výšky hladiny v nádržích, respektive řízení čerpadel na základě stávající výšky hladiny v jednotlivých nádržích. Cílem bylo, aby studenti v předpřipraveném okně sestavili pomocí funkčních bloků grafické schéma pro zapínání čerpadel, na základě několika zadaných logických podmínek. Úplné zadání je možné najít v literatuře [1].

### 5.2.2 Původní řešení – Simulink

V Simulinku jsou tři hlavní soubory. Jedná se o *ttlogic.mdl.* To je základní soubor úlohy, co se týče právě MATLABU. Je v něm nastavení úlohy, komunikace a prostor pro řešení logické úlohy. Dále pak *ttlogicstop.mdl* a *bloky.mdl*. Tyto dva soubory doplňují základní soubor. Při spouštění se spustí script *run.m.* Ten v případě, že na určitém místě v počítači nejsou požadované soubory vytvoří novou složku a potřebné soubory tam nakopíruje. Následně spustí soubory *ttlogic.mdl* a *bloky.mdl.*

Pro úplné pochopení úlohy je důležité si ze začátku zjistit, co znamenají jednotlivé vstupy a výstupy a myšlenkově propojit spárované RDC bloky v RexDraw s těmi, co jsou použité v Simulinku. Dále pak postupně projít napojení jednotlivých bloků za sebou a zjistit jejich účel.

Pro přehlednost jsem vypsal smysl jednotlivých vstupů a výstupů z RDC bloků, které popisuji, do tabulky (viz Tab. 1 a Tab. 2):

Tab. 1: Význam vstupů a výstupů v bloku RDC

| Simulink | RexDraw | Význam |
|---|---|---|
| y1 | u1 | hodnota napětí na tlakoměru v nádrži 2 |
| y2 | u2 | hodnota napětí na tlakoměru v nádrži 3 |
| y3 | u3 | hodnota napětí na tlakoměru v nádrži 1 |
| u10 | y10 | logické zapnutí/vypnutí čerpadla u1a |
| u11 | y11 | logické zapnutí/vypnutí čerpadla u1b |
| u12 | y12 | logické zapnutí/vypnutí čerpadla u2a |

Tab. 2: Význam vstupů a výstupů v bloku RDC1

| Simulink | RexDraw | Význam |
|---|---|---|
| u0 | y0 | nastavení otáček čerpadla u1a |
| u1 | y1 | nastavení otáček čerpadla u1b |
| u2 | y2 | nastavení otáček čerpadla u2a |
| u3 | y3 | nastavená hodnota napětí pro maximální výšku hladiny v nádobě 1 a 3 |
| u4 | y4 | nastavená hodnota napětí pro maximální výšku hladiny v nádobě 2 |

V souboru *ttlogic.mdl* je ošetřeno nastavení úlohy pomocí subsystému *Servis*. V tomto subsystému se nastavují otáčky všech tří čerpadel pomocí sliderů s rozsahem od 0 do 10. Tyto hodnoty jsou přivedeny na vstupy u0 až u2 do bloku pojmenovaného RDC1, který je ovšem v RexDraw spárovaný s blokem RDC_2. Proto je nutné kontrolovat id bloků, o kterých jsem se zmínil v kapitole **RDC.** Vstupy u3 a u4 jsou nastavené hodnoty maximálního napětí na tlakoměrech, které tak nepřímo ovládají maximální povolenou výšku hladiny v nádobách.

Subsystém *Logicka uloha* obsahuje pouze tři vstupy reprezentující výšku hladiny v nádobách a tři výstupy reprezentující, jestli mají být jednotlivá čerpadla nezávisle na sobě zapnuta, nebo vypnuta. K tomu, aby měl tento subsystém nějaký smysl je soubor *bloky.mdl*, který představuje logickou knihovnu bloků. Nejedná se pouze o logické funkce typu *and* a *or* a jiných, ale je zde i například blok, který řeší to, jak současnou výšku hladiny, což je spojitá veličina, přeměnit na binární hodnotu, kterou je možno v úloze použít. Přesněji dělá to, že po nastavení sledované hodnoty z bloku vrátí, zda je současná výška hladiny vyšší nebo nižší než ta zadaná [1]. Pokud je hodnota nižší než zadaná, vrací 0. Pokud je vyšší než zadaná, vrací 1. Další blok, který stojí za to zmínit, je generátor signálů jedničky a nuly s možností přepínání pomocí vypínače. Tento způsob řešení generování signálu mi ovšem nepřijde příliš uživatelsky přátelský a je to něco, co bych rád ve svém uživatelském prostředí změnil.

Posledním subsystémem v souboru je *Kaskada tri nadrzi*, který reprezentuje fyzickou soustavu. Ten má tři vstupy, které vycházejí z popsaného subsystému *Logicka uloha* a říkají, zda má být dané čerpadlo zapnuté nebo vypnuté. Tyto proměnné jsou uvnitř subsystému přivedeny na vstupy u10, u11 a u12 na druhý použitý RDC blok, který je v RexDraw pojmenován jako RDC_1. Dále je v tomto subsystému série bloků, které zpracovávají signál tří tlakových senzorů zasílaných z RexDraw. Přes blok Selector se vyberou pouze požadované tři hodnoty a přepočtem z nich je získávána výška hladiny.

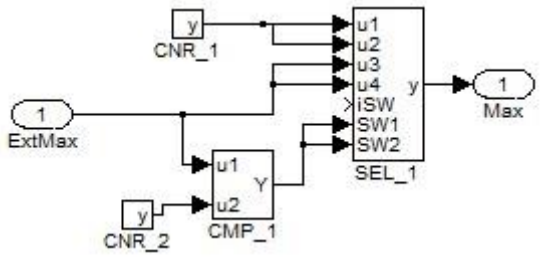
### 5.2.3 Původní řešení – RexDraw

Co se týče hlavního souboru exec.mdl v REX Controls, platí vše, co bylo řečeno v kapitole 5.1. Proto se nyní budu především zabývat strukturou druhého souboru ttlogic.mdl.

Nyní už můžeme poměrně lehce dojít k funkcím jednotlivých bloků v souboru a následně i fungování celého modelu. Z bloku RDC_1, nám z výstupů y10 až y12 jde signál přes saturační bloky SAT_4, SAT_5 a SAT_6, které omezují tuto hodnotu v rozmezí od 0 do 1. Následují bloky násobení MUL_1, MUL_2 a MUL_3, které spojují velikost otáček z bloku RDC_2 a logické zapnutí z bloku RDC_1. Další bloky jsou opět násobící, jsou označené MUL_4, MUL_5 a MUL_6 a zapojují do obvodu logickou podmínku, jestli nebyla čerpadla vypnuta z důvodu příliš vysoké hladiny v nádržích, respektive příliš vysokého napětí na tlakových senzorech. Princip tohoto ošetření je popsán v následujícím odstavci. Následují další saturační bloky SAT_1, SAT_2 a SAT_3, které mají rozsah od 0 do 5. Na konci už je jen zápis na vstupy bloku QUADOUT komunikujícím s čerpadly, čímž je uzavřen celý řídící systém modelu.

Zmíněný princip ošetření maximální výšku hladiny funguje pomocí subsystémů SetMax_1 (viz Obr. 11) a SetMax_2, který funguje analogicky. Do těchto subsystémů jsou přivedeny maximální hodnoty napětí nastavené v Simulinku. Tato hodnota je porovnána s konstantou, která je v bloku CNR_2 nastavena jako 0. Pokud je větší, z bloku CMP_1 vyjde logická 1, pokud menší tak 0. V bloku CNR_1 je nastavena konstantní hodnota 3. Právě tato konstanta a námi nastavená hodnota jsou přivedeny do nejdůležitějšího bloku tohoto subsystému. Jedná se o tzv. *analog signal selector* [6], který na základě hodnot přivedených na vstupy SW1 a SW2 určuje, jaký ze čtyř vstupů u1 až u4 pošle na výstup y. Vzhledem k tomu že na obě svorky SW1 a SW2 je připojena stejná hodnota, může dojít pouze ke dvěma případům. Buď bude nastavená hodnota napětí v Simulinku větší než 0, pak na výstupu komparátoru CMP_1 bude logická 1, tudíž i na svorkách SW1 a SW2, a na výstup se pošle hodnota u4, tedy ta námi nastavená. Pokud ovšem bude naše nastavená hodnota nulová, z CMP_1 bude vyslána logická 0, tudíž i na svorky SW1 a SW2, a na výstup se pošle hodnota u1, tedy ta nastavená v bloku CNR_1 na hodnotu 3 *V*.

Z tohoto subsystému takto získáme maximální dovolenou hodnotu napětí, která je pak porovnávána v blocích CMP_1, CMP_2 a CMP_3 přímo s hodnotou napětí na tlakových senzorech. Pokud by hodnota na senzorech byla příliš vysoká, z těchto bloků by začala vycházet logická 0 a příslušné čerpadlo by se vyplo.




*Obr. 11: Subsystém SetMax_1*

## 5.3 Frekvenční charakteristika

### 5.3.1 Zadání

V této úloze mají studenti za úkol změřit 3 body amplitudové a fázové frekvenční charakteristiky jedné ze soustav, které lze nastavit konfigurací ventilů. Ve zkratce jde o spuštění úlohy pomocí předpřipraveného uživatelského prostředí a odečtení hodnot z grafů, které jsou v MATLABU utvořeny. Tím si přiblížit probranou teorii v předmětu Automatické řízení. Celé zadání je možné najít v [2].

Celková softwarová realizace je několikanásobně komplexnější, než tomu bylo u logického řízení. To je zřejmé už jen z toho, že se jedná o spojité řízení a regulaci, tudíž mnohem složitější problematiku, než sekvenční logické řízení zapínání a vypínání čerpadel. Celkový popis takového systému přesahuje rámec této bakalářské práce. Z tohoto důvodu

nebudu popisovat každý jednotlivý blok, jako v kapitole 5.2, ale pokusím se vždy přiblížit jednotlivé části a popsat jejich funkci a smysl pro celkový systém. Zároveň úlohy v kapitolách 5.3 Frekvenční charakteristika a 5.4 Uzavřený regulační obvod pracují se stejným souborem na úrovni řídícího PLC, ale mají rozdílné uživatelské prostředí v MATLAB Simulinku. Proto se pokusím soubor v RexDraw přiblížit více v této kapitole a v té následující se na něj jen odkázat.

### 5.3.2 Původní řešení – Simulink

Pokud se podíváme na hlavní soubor v Simulinku pro tuto úlohu *ttfreq.mdl*, můžeme si všimnout, že je tvořen jedním velkým subsystémem *Soustava tri nadrzi*. Jeden z jeho vstupů je generátor harmonického signálu. Ten má nastavitelnou amplitudu a frekvenci. Právě frekvence je parametr, který se má během měření této úlohy měnit, aby došlo ke změnám pozorovaného obrazce a možnosti naměření několika různých hodnot. Druhým vstupem subsystému je signál z generátoru logických 1 a 0, který zapíná, respektive vypíná harmonické buzení. To je v subsystému vyřešeno pomocí bloku *Product*, do kterého jsou oba tyto vstupy přivedeny. Tento blok mezi sebou elementárně vynásobí přivedené vstupy, tudíž pokud je na přepínači nastavená nula, harmonické buzení nebude v simulaci započítáváno. Co se týče výstupů tohoto subsystému, jedním z nich je blok *zapisovadlo* s přepínačem. To v případě dokončení ukládání zobrazí graf s Lissajousovými obraci, které jdou následně uložit. Druhý výstup je graf s výškou hladin v nádržích.

V subsystému *Soustava tri nadrzi* si můžeme všimnout čtyř dalších subsystémů, které jsou pro tuto úlohu velmi zásadní. Jedná se o dva regulátory, servis úlohy a subsystém popisující nádrže. Co se týče subsystému *Regulator1* a *Regulator2,* jsou naprosto identické. Každý z nich řídí jednu ze sledovaných nádrží. Mezi jejich vstupy patří konstanty pro nastavení regulátoru. Přesněji tedy proporcionální *r0*, integrační rI a derivační rD konstanty. Následující vstup je žádaná hodnota regulované veličiny označená *w.* Dále pak vstup říkající, o jakou nádrž se jedná, což je uvnitř tohoto subsystému vyřešeno pomocí bloku *Multi-Port Switch,* který zvolí, jaký ze vstupů pošle na výstup, na základě číselné hodnoty přivedené na řídící vstup. Tudíž přímo číslem nádrže (počítáno od nuly – 0,1,2) zvolíme, s jakými hodnotami je následně pracováno. Další ze vstupů tohoto subsystému předává hodnotu akční veličiny pro manuální řízení, ať už s nebo bez harmonického buzení. Posledním vstupem je určováno, jestli se mají hodnoty akční veličiny nastavovat automaticky pomocí regulátoru, nebo se má používat manuální hodnota $u_{man}$.

Subsystém *Servis* se podobá tomu v logické úloze. V prvním bloku *RDC50* se nastavují maximální hodnoty dovoleného napětí tlakoměrů pro první a třetí nádrž, dále pak offsety a zesílení pro čerpadla. V druhém bloku *RDC16* pak mezní hodnoty pro saturaci a antiwind-up pro dva regulátory.

Antiwindup je v regulátoru důležitý pro to, aby nedovolil zapojení integrační složky do regulace, v případě překročení fyzických limitů akčního zásahu. Pro příklad čerpadlo nemůže mít větší průtok, než na které je nadimenzované, ale zároveň nemůže být více zavřené, než když je jeho průtok nulový. Proto, když dojde k tomuto jevu, antiwindup může

integrátor udržovat vypnutý, do té doby, než se jeho hodnota dostane zpět do použitelného rozsahu.

Posledním subsystémem je pak blok s názvem *Nadrze.* Ten nám přijímá hodnoty z RexDraw o současném napětí na tlakoměrech. Jeho tři výstupy jsou přepočtené hodnoty napětí jako výšky hladin v nádržích.

V této úloze jsou je ještě použito několik pomocných skriptů a Simulinkových modelů. Jedná se o skript *run_freq.m,* který funguje analogicky jako u logické úlohy s tím rozdílem, že spouští pouze hlavní soubor *ttfreq.mdl.* Dále se jedná o skripty *ttload.m, ttinit.m.* Co se týče výstupů tohoto subsystému, jedním z nich je zapisování dat do definovaného souboru, což je vyřešeno pomocí samostatného matlabovského skriptu *ttfreqzapis.m.* Druhým výstupem je graf zobrazující časový průběh výšek hladin v nádržích.

### 5.3.3 Původní řešení – RexDraw

V prostředí RexDraw máme opět dva soubory. Jeden hlavní *exec.mdl* a jeden vedlejší *ttregul.mdl.* V souboru *ttregul.mdl* si můžeme všimnout bloku WPC__S0I17X0_7. Ten přijímá z řídící jednotky hodnoty z tlakoměrů. Tyto hodnoty se opět porovnávají s hodnotami maximálního napětí, které se nastavují v Simulinku v subsystému *Servis.* Logické ošetření proti přetečení je pak přes bloky MUL_1, MUL_2 a MUL_5 připojeno do celkového systému a přes tři saturační bloky nastavuje chod čerpadel pomocí bloku WPC_S1I24V0_3. Zároveň jsou tyto původní hodnoty tlakoměrů odesílány do Simulinku pomocí komunikačního bloku RDC_27. Právě maximální dovolené napětí tlakoměrů a další hodnoty jako je offset nebo sklon jsou posílány z bloku RDC_50. Zbylé nastavené parametry jsou poslané ze Simulinku do RexDraw pomocí bloků RDC_5 a RDC_20 a popisují regulátory. Dále si lze v souboru všimnou několika dalších subsystémů.

Subsystém *the_1st_control* přijímá jako vstupy regulační odchylku, konstanty regulátoru, hodnotu pro reset, integrační maxima a minima, což souvisí s popsaným antiwindupem a informaci, jestli se má použít manuální hodnota pro akční veličinu nebo ne. Jeho výstupy jsou přepočítaná celková akční veličina a zároveň její proporcionální, integrační a derivační složka. Vnitřní schéma představuje PID regulátor s antiwindupem, jehož nejdůležitějšími složkami jsou bloky zajišťující proporcionální, derivační a integrační část regulace.

Dalším subsystémem je blok dynamické saturace s názvem *dyn_saturation.* Ta nám do vstupů bere nastavené hodnoty ze Simulinku pro maximální a minimální saturaci a dále pak samotnou hodnotu proměnné a její meze. Výstupem je nastavená proměnná. Vnitřek tohoto bloku je tvořen skupinou podmínek, které porovnávají pomocí bloku CMP (compare) aktuální hodnotu s nastavenou a případně jí omezuje na hodnotu maximální nebo minimální meze.

Další subsystém, který popíši má název a*_m_analog_A.* Tento blok je použit v souboru třikrát. Poslední písmeno v názvu reprezentuje, o jakou nádobu se jedná. Tento subsystém má tři vstupy. Akční veličiny regulátoru nastavené automaticky a manuálně a informaci

nastavenou v Simulinku, jaká z nich se má použít. To je uvnitř bloku vyřešeno pomocí bloku *analog signal selector,* o kterém jsem se zmiňoval na konci kapitoly 5.2.2. Výstupem je proměnná *value*, která reprezentuje příslušnou akční veličinu pro současné nastavení úlohy.

Poslední subsystém *Set_properties* je také použit pro každou z nádrží zvlášť. Má tři vstupy pro hodnotu offsetu a zesílení nastavené v Simulinku a dále proměnnou *value,* která vychází ze subsystému popsaného v předchozím odstavci. Výstupem je proměnná, která má stejné jméno, ale je přepočítaná za použití offsetů a sklonu.

## 5.4 Uzavřený regulační obvod

### 5.4.1 Zadání

Cílem této úlohy je stanovení optimálních parametrů PI regulátoru. Jedná se o zesílení $r_0$ a časovou integrační konstantu $T_I$. Tyto parametry se dají zjistit pro třemi metodami:

1. Seřízení PI regulátoru metodou přechodové odezvy regulované soustavy za změnu akční veličiny.
2. Seřízení PI regulátoru metodou relé.
3. Seřízení PI regulátoru metodou kritických kmitů Zieglera a Nicholse.

### 5.4.2 Původní řešení – Simulink

Základním souborem pro ovládání této úlohy je v Simulinku ttregul.mdl. Tento soubor má jeden hlavní subsystém *Soustava tri nadrzi*. Ten má dohromady deset vstupů, které jsou přiřazené pro řízení dvou nádrží, tudíž každé z nich náleží právě pět vstupů. Prvním z nich je nastavení konstant regulátoru *r0*, *rI* a *rD*, následně pak referenční veličina *w*. Dále pak zda je akční veličina nastavena automaticky nebo se bere hodnota $u_{man}$. Další vstup je právě nastavená hodnota $u_{man}$. Poslední vstup reprezentuje, jestli je zapnuta metoda relé nebo ne. Co se týče výstupů, subsystém má čtyři. První dva výstupy zobrazují výstupní hodnoty pro řízené nádoby. Jedná se o výšku hladiny, akční veličinu daného regulátoru a regulační odchylku. Dalším výstupem je zápis dat, který je spojen s přepínačem pro zapínání. Zápis je opět možno vygenerovat do samostatného souboru pomocí externího matlabovského scriptu ttzapis.m, o kterém jsem se zmínil v kapitole 5.3.2. Posledním výstupem je opět přehled všech výšek hladin, kde každá různě barevná křivka reprezentuje jednu z nádrží.

Pokud se do tohoto subsystému podíváme, na první pohled je vidět veliká podobnost s úlohou v předchozí kapitole 5.3.2. Opět jsou zde čtyři použité subsystémy, které jsou naprosto identické a jedná se tedy o dva regulátory, servis a subsystém popisující nádrže. Hlavním rozdílem je ovšem výskyt záporné zpětné vazby, která vede z výstupu regulované veličiny *y* a vede do vstupu manuálně nastavené hodnoty akční veličiny regulátoru.

### 5.4.3 Původní řešení – RexDraw

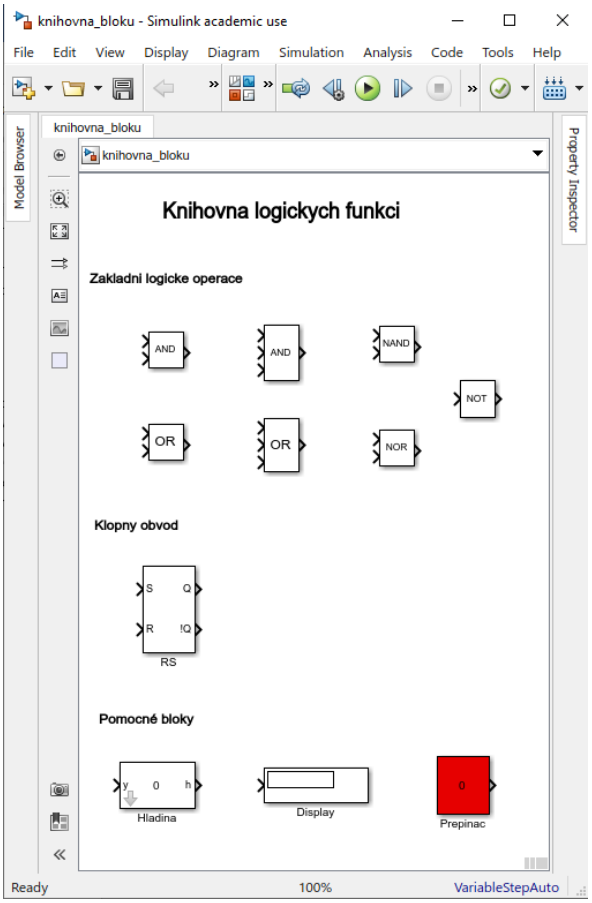
Co se týče spojení této úlohy s řídícím systémem RexDraw, jedná se o totožný soubor *ttregul.mdl,* který jsem popisoval v kapitole 5.3.3, proto se na ní pouze takto odkáži.

# 6 Vytvoření nového uživatelského prostředí

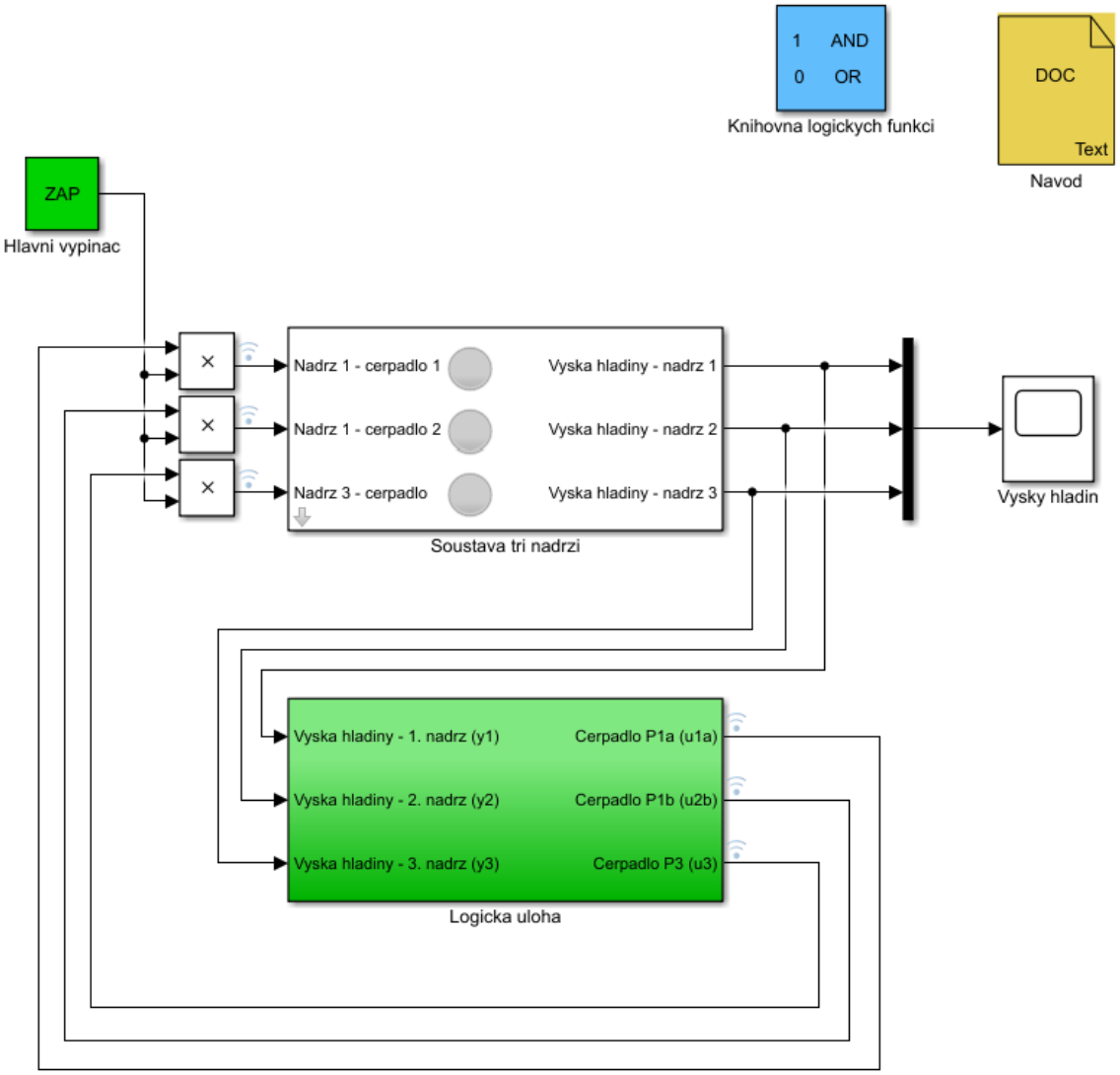
Při vytváření nového uživatelského prostředí pro všechny tři úlohy jsem se zaměřil na dvě hlavní kritéria. Mým prvním cílem bylo co nejvíce znázornit vizualizací úlohy probírané téma v předmětu automatické řízení a přiblížit tak probíranou teorii na reálném modelu. Tím druhým pak udělat prostředí co nejvíce uživatelsky přátelské, a to jak pro samotné užívání studenty, tak pro případné úpravy úloh a měnění jejich parametrů v případných změnách v úloze. U úloh jsem převzal původní skripty pro načítání, zastavení, vykreslení grafů apod. Pouze jsem je přizpůsobil svým podmínkám. Při finálním spuštění je potřeba jen spárovat ikonu zástupce pro danou úlohu se scripty run, respektive run_freq.

## 6.1 Logické řízení

Původní řešení logické úlohy mi přišlo velmi dobře zpracované a neviděl jsem důvod zásadně vzhled úlohy měnit. Proto jsem ve svém řešení zvolil stejné rozložení, ve kterém jeden ze subsystému reprezentuje laboratorní model, druhý subsystém pak dává možnost vyřešit řízení čerpadel na základě logických podmínek. Potřebné bloky se tahají z logické knihovny, kterou je možné otevřít pomocí tlačítka *Knihovna logických funkcí*, které jsem do mé verze přidal, aby bylo možno knihovnu otevřít přímo z hlavního souboru v případě zavření. Při spuštění hlavního souboru ovšem dojde k otevření knihovny automaticky.




*Obr. 12: Knihovna logických funkci*




*Obr. 13: Řešení vizualizace Logické úlohy*

Dále jsem přidal tři žárovky, které mění barvu z červené na zelenou na základě logických hodnot, které jsou do žárovek posílány. Každá z žárovek je připojena k jednotlivému čerpadlu, tudíž je za běhu simulace vidět, jaké čerpadlo je v daný moment zapnuté a jaké ne. Dále jsem přidal tlačítko s návodem přímo do hlavního souboru. Při dvojkliku na něj se otevře v MATLABu stručný návod pro řízení úlohy.

V knihovně jsem předělal uspořádání bloků, rozdělil je do tematických skupin a přidal k nim nadpisy. Dále jsem předělal generátor logické 0 a 1 pro řešení úlohy. V původním řešení byl tento generátor vyřešen pomocí subsystému, ze kterého vycházely hodnoty 1 a 0 a mezi nimi se přepínalo pomocí externího přepínače. Ačkoliv je toto řešení přehledné a je vidět jaká hodnota je zrovna sepnutá, vzhledově není příliš hezké. Zároveň se při přetahování z knihovny musí označit oba dva bloky najednou, jinak dojde k rozpojení. Při svém řešení jsem proto vytvořil blok *Prepinac*, který nyní rozeberu trochu více do hloubky pro ukázku možností callbacků, které jsem popisoval v kapitole 4.3.1.

### 6.1.1 Subsystém Prepinac

Tento subsystém jsem použil jako reprezentaci nějakého tlačítka nebo páčky, která umožňuje sepnutí některého z čerpadel za splnění příslušných podmínek ze zadání. Studenti ho tak můžou použít v řešení logických podmínek. Vnitřek tohoto bloku obsahuje pouze vnitřní proměnou *val,* která je přímo napojena na výstup bloku a její hodnota je tedy přímo výstupem tohoto subsystému.

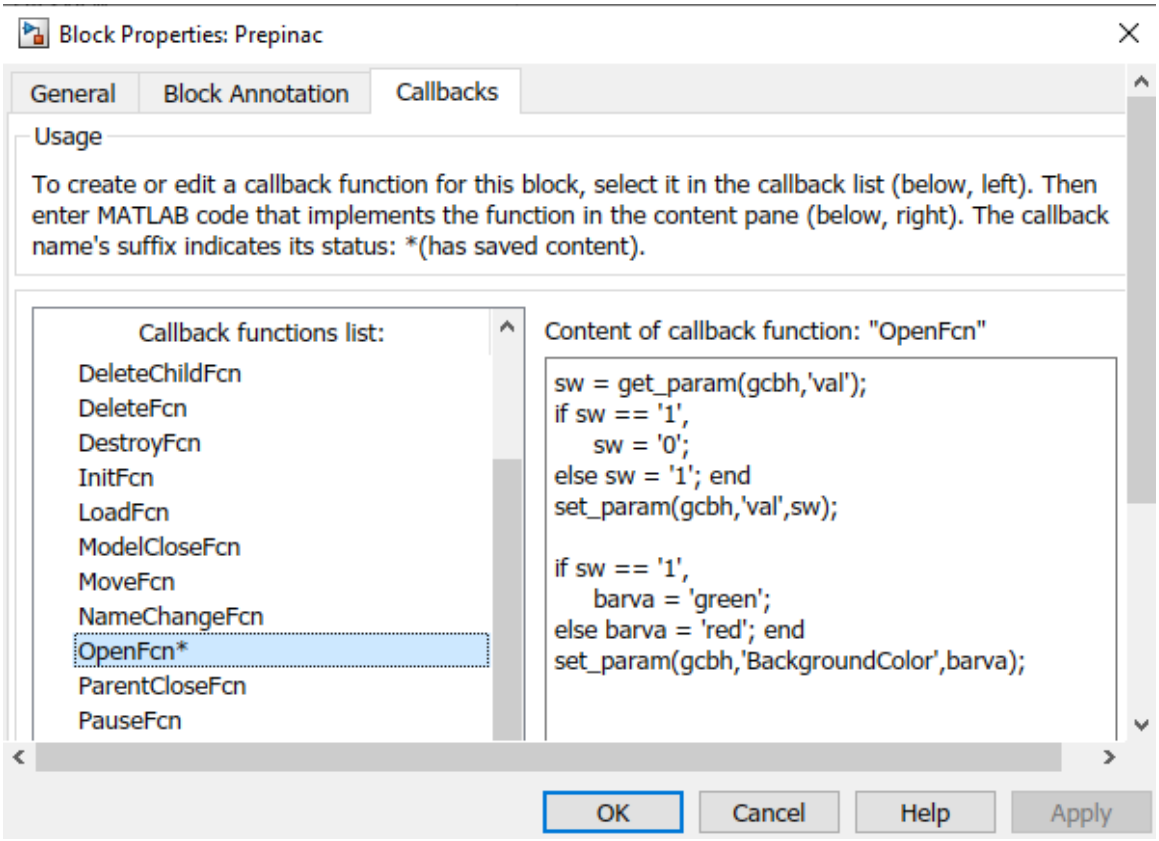
Princip tohoto bloku spočívá v tom, že mění svou barvu a text pokaždé, co na něj uživatel dvojklikne. Z toho, že je potřeba zásah uživatele je zřejmé, že bylo potřeba použít funkcí callbacku, konkrétně funkce *OpenFnc,* která se spustí při otevření bloku, respektive tedy dvojkliku na něj. Kód tohoto callbacku je vidět na Obr. 14.

Při dvojkliku se tedy v prvním řádku načte do proměnné *sw* současná hodnota proměnné *val* pomocí funkce get_param. Její první parametr je *gcbh,* což je zkratka pro *Get Current Block Handle*, druhý parametr je pak název proměnné, kterou chci načíst. V následujících třech řádcích je podmínka *if,* která proměnnou *sw* změní z jedničky na nulu v případě, že její současná hodnota je právě jedna. V opačném případě se využije druhá část podmínky *else* a proměnná *sw* se nastaví na jedničku. V pátém řádku se zapíše současná hodnota proměnné *sw* do vnitřní proměnné *val*. Těchto pět řádků tedy provede přečtení současné hodnoty, její změnu na opačnou hodnotu a následný zápis. V následujících řádcích je vyřešena změna barvy bloku. V podmínce *if* se porovná současná hodnota proměnné *sw.* Pokud je rovna jedné, nastaví se nová proměnná *barva* na zelenou, v opačném případě na červenou. Poslední řádek pak nastaví barvu bloku na základě této proměnné, a to pomocí funkce *set_param*, která má stejné parametry, jako funkce *get_param,* kterou jsem popisoval v tomto odstavci.

Co se týče masky subsystému, je zde velmi krátká. Jedná se o pouze jeden řádek, který na blok vypíše současnou hodnotu proměnné *val,* která je z bloku posílána na výstup.

Já tento blok použil jako hlavní vypínač celé úlohy, který umožňuje sepnutí čerpadel. Pouze jsem předělal jeho masku, aby místo hodnoty proměnné *val* vypisoval v její závislosti slova *ZAP a VYP.* To stačilo udělat pomocí jednoduché podmínky if, jejíž kód vypadal následovně:

```
if val == 0
    fprintf('VYP');
else
    fprintf('ZAP');
end
```




*Obr. 14: Callback bloku Prepinac*

## 6.2 Frekvenční charakteristika

V této úloze jsem se rozhodl celkové řešení změnit tak, aby při používání bylo možné trochu více porozumět, co se v úloze právě děje a zároveň celou úlohu upravit do lepší vizuální podoby. Jedním ze základních prvků v úloze je generátor harmonického signálu, který je sčítán s manuálně nastavenou hodnotou akční veličiny. Původně bylo přidání harmonického signálu vyřešeno pomocí přepínače s generátorem logické 0 a 1, jehož hodnota byla jedním ze vstupů subsystému. V mém řešení jsem použil stejný přepínač jako v logické úloze, pouze s pozměněným textem na bloku na *ZAP/VYP.* Zároveň jsem vyvedl sčítání manuálně nastavené akční veličiny s harmonickým signálem mimo subsystém reprezentující reálný model, aby bylo přímo na první pohled vidět, co se během simulace děje při použití zmíněného přepínače.

V úloze se několikrát mění hodnota frekvence pro harmonický signál, aby došlo k tvorbě rozdílných obrazců v grafu. Z toho důvodu jsem přidal do oblasti nastavení úlohy slider, který je propojený s nastavením frekvence v generátoru harmonického signálu. Tento slider má omezené hodnoty v rozmezí, které je studentům doporučeno nastavit. Řešení slideru v Simulinku mi přijde ovšem poněkud nepraktické, protože před tím, než se dá s jezdcem hýbat, musí se na blok nejprve kliknout. To má za následek, že při prvním použití se většinou hýbe s celým sliderem, nikoliv pouze jeho jezdcem. Proto jsem přidal callback,

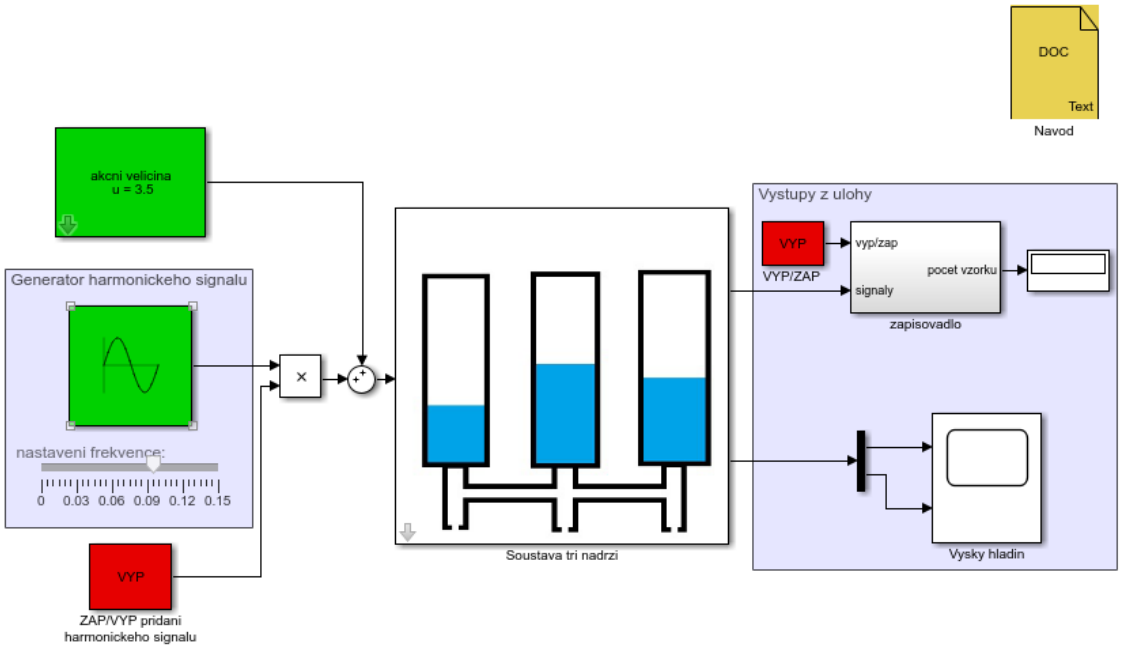
který se spustí ve chvíli, kdy se s blokem pohne. Vyskočí dialogové okno, ve kterém je krátká vysvětlivka, že na slider se musí nejprve kliknout a až poté je možné hýbat s jezdcem. Kód tohoto callbacku je:

```
helpdlg(sprintf('Nedjrive kliknout na slider, potom tahnout.'))
```

V subsystému *Soustava tri nadob,* který tvoří hlavní tělo programu, jsem předělal vnitřní strukturu, aby mohlo dojít ke změnám které jsem popsal v předchozím odstavci. Jednalo se především o předělání struktury, která byla spojena se vstupy, protože jsem oproti původnímu řešení jeden odebral. Dále jsem pak vytvořil masku subsystému. Je tvořena schématickou ikonou tří nádrží zobrazenou přímo na bloku, aby bylo na první pohled vidět, co daný blok reprezentuje.

Zobrazování a zápis výstupů jsem neměnil a použil stávající řešení. Jen jsem opět nahradil původní přepínač, který byl stejný jako ten původní u logické úlohy a nahradil ho tím svým popsaným dříve. Opět jsem akorát pozměnil popisky, aby odpovídal dané situaci.

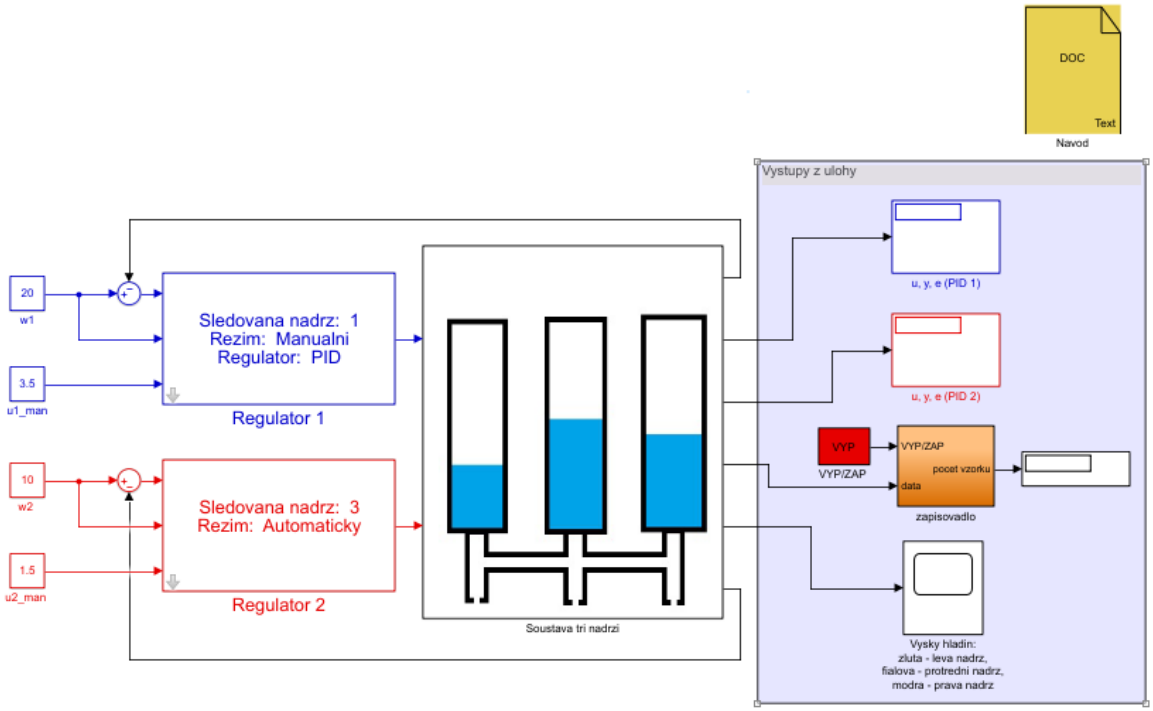
Nakonec jsem přidal stručný návod na ovládání úlohy, do kterého lze zapsat pokyny pro tuto úlohu.




*Obr. 15: Řešení vizualizace úlohy Frekvenční charakteristika*

## 6.3 Uzavřený regulační obvod

Při vytváření nového uživatelského prostředí pro tuto úloho bylo potřeba předělat matlabovský soubor prakticky od základů. Původní řešení mi přišlo hodně nepřehledné, proto jsem chtěl práci s modelem co nejvíce zjednodušit a zároveň tak udělat názornější, co se během simulace děje. Nejzásadnější pro mne tedy bylo, aby moje uživatelské prostředí připomínalo uzavřený regulační obvod, s jehož schématickým obrázkem se studenti automatického řízení setkávají velmi často, a tudíž aby bylo na první pohled jasné, co daná úloha reprezentuje. Reálně to znamená výskyt referenční hodnoty, která je spojená se zápornou zpětnou vazbou, regulační odchylku, regulátor, akční veličinu, blok reprezentující systém a regulovanou veličinu přímo v hlavním okně simulace.




*Obr. 16: Řešení vizualizace úlohy Uzavřený regulační obvod*

Při vytváření nového uživatelského prostředí jsem tedy vytvořil dva subsystémy reprezentující regulátory, které popíši v samostatné kapitole 6.3.1, abych mohl rozebrat jejich strukturu a masku trochu více dopodrobna.

Subsystém *Soustava tri nadrzi* symbolizuje reálnou soustavu. Ve skutečnosti se v něm neděje nic zásadního, co se týče řízení úlohy a jeho význam je opravdu spíše pro lepší vizuální zpracování úlohy. Výjimkou by snad mohl být jen blok pro servis úlohy, kde se nastavuje maximální dovolené napětí na tlakoměrech, tak jako v předchozích úlohách. Dále pak blok, který komunikuje s řídícím systémem v RexDraw a získává z něj aktuální výšky hladin. Hodnoty, které přijme z regulátorů se pomocí bloku DEMUX rozdělí a následně jsou rozděleny na šest příslušných výstupů. Jedná se o dva výstupy y, které jsou použité na zpětnou vazbu. Dále pak dva výstupy, které umožňují sledovat hodnoty akční veličiny,

řízené veličiny a regulační odchylky pro oba regulátory. Dalším výstupem je zapisovadlo, které při sepnutí tlačítka začne zaznamenávat aktuální hodnoty. Posledním výstupem je pak graf, který ukazuje současnou výšku hladin v nádržích.

Výstupy jsem opět nechal v prakticky původním rozpoložení. Pouze jsem trochu poupravil velikosti, opět použil svůj přepínač s odpovídajícími popisky a ohraničil pole výsledků s nadpisem *Vystupy z ulohy*.

Jako v předchozích úlohách jsem ještě přidal blok s návodem, který opět stručně popisuje cíl a ovládání dané úlohy.

### 6.3.1 Subsystém Regulator

Subsystém regulátor má jako vstupy referenční hodnotu *w,* regulační odchylku *e*, která je vytvořená zápornou zpětnou vazbou a jako polední manuálně nastavenou akční hodnotu *u_man.* Z regulátoru vychází pouze jeden výstup, který pro studenty reprezentuje nastavení akční veličiny *u.* Reálně je to ovšem pět hodnot, které jsou spojeny do jednoho signálu pomocí bloku MUX uvnitř subsystému, a to z důvodu, aby studenty nemátlo, že z regulátoru vychází pět hodnot, když to má být podle teorie pouze jedna.

Vnitřní struktura subsystému je tvořena z dalšího menšího subsystému, ve kterém je zrealizovaná komunikace s řídícím systémem RexDraw. Ten má jako vstupy parametry příslušného regulátoru, referenční veličinu, číslo sledované nádrže, manuálně nastavenou hodnotu akční veličiny a nastavení, zda pracuje v automatickém nebo manuálním režimu. Jeho výstupy jsou výsledná hodnota, akční veličina z regulátoru a regulační odchylka.

Základem pro to, aby se v masce mohli vůbec nastavovat nějaké hodnoty je mít uvnitř subsystému místo konkrétních hodnot parametry. V masce se pak přidá možnost, jak tento parametr zadat. Může to být přímo zadáním konkrétní hodnoty, vybrání položky ze seznamu, nebo například pomocí slideru, a další. Tyto možnosti zadávání se spojí s daným parametrem a při vybrání uživatelem se nastaví v samotném subsystému. Podle volby zadání se do parametru zapíše příslušná hodnota. Pro příklad u vybírání z vyskakovací nabídky je to číslo volby, tedy 1 a výš. U zadávání konkrétní hodnoty je přímo ta poslána do příslušného parametru.

Maska tohoto subsystému se dělí do dvou hlavních částí. První část je menu, které se zobrazí při rozkliknutí bloku, a ve kterém se může konfigurovat regulátor, respektive úloha jako taková. Druhá část je text, který se přímo na bloku zobrazuje a je tedy čistě informační.

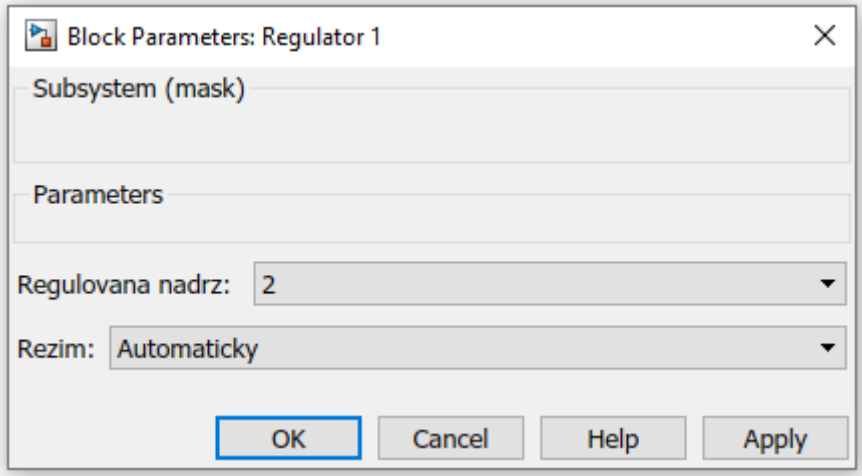
První část masky se nastavuje v parametrech a dialozích. Pro každý parametr, který jsem ve vnitřní struktuře subsystému zavedl, jsem přiřadil možnost, jak ho nastavit. Konkrétně se jedná o číslo sledované nádrže, kde lze z vyskakovacího menu vybrat možnosti 1, 2 a 3. Dále vyskakovací menu, ve kterém se nastaví, jestli má regulátor pracovat v automatickém nebo manuálním režimu. V dalším vyskakovacím menu se vybere typ použitého regulátoru, přesněji PID nebo dvoupolohový. Nakonec tři parametry, které se nastavují konkrétními hodnotami zadané uživatelem. Reprezentují proporcionální, integrační a derivační konstanty regulátoru rP, rI a rD. Pro lepší vzhled a celkovou přehlednost jsem ještě přidal

pro vyskakovací menu, kde se vybírá automatický nebo manuální režim callback. V něm jsem nastavil, aby došlo k schování ostatních možností nastavení regulátoru, pokud dojde k vybrání automatického režimu z nabídky. Názorný příklad je vidět na Obr. 17 a Obr. 18. To vyplývá z reálného používání této úlohy, kde nemá smysl zadávání konstanty regulátoru, když se v automatickém režimu stejně nepoužívají. Ke schování parametrů regulátoru dojde i v případě vybrání dvoupolohového regulátoru za pomocí identického callbacku, akorát s parametrem *PR*.

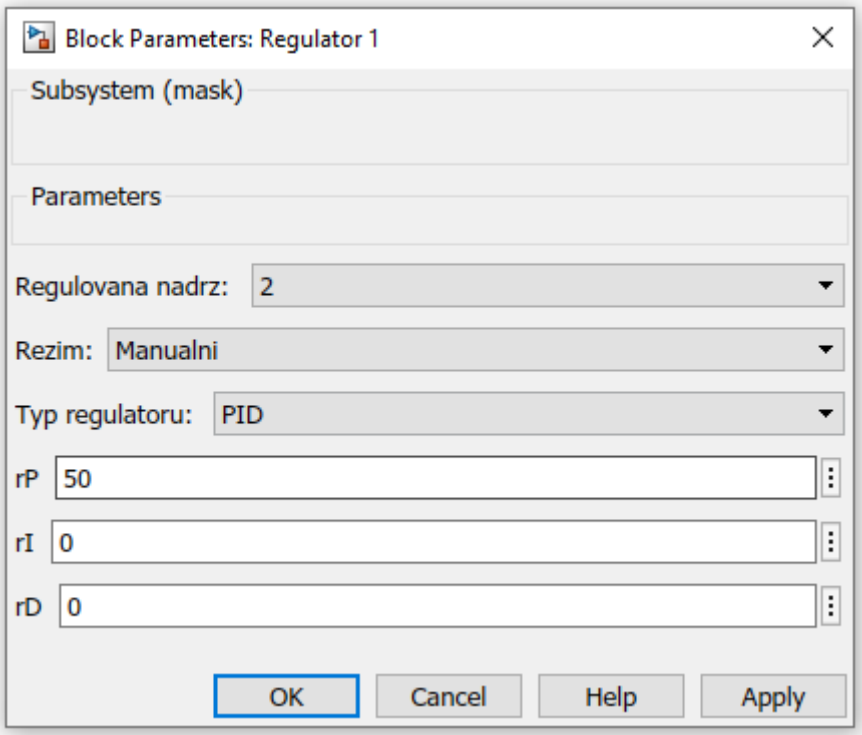
Kód tohoto callbacku je:

```
switch get_param(gcb,'AM')
    case 'Automaticky'

set_param(gcb,'MaskVisibilities',{'on','on','off','off','off','off'});
    case 'Manualni'

set_param(gcb,'MaskVisibilities',{'on','on','on','on','on','on'});
end
```

Na začátku se zjistí hodnota parametru *AM,* který odpovídá nastavování automatického nebo manuálního režimu. V obou případech podmínky dojde k nastavení viditelnosti jednotlivých možností v celkovém menu. V případě první volby se vypne viditelnost nastavení typu regulátoru a jeho konstant. V případě druhé volby se naopak jejich viditelnost zapne.




*Obr. 17: První varianta vyskakovacího menu subsystému Regulator*




*Obr. 18: Druhá ukázka vyskakovacího menu subsystému Regulator*

Druhá část masky tohoto subsystému se stará, aby na bloku bylo vždy napsána současná konfigurace daného regulátoru, respektive úlohy. První řádek říká, která z nádrží je daným regulátorem řízena. Druhý řádek, jestli je regulátor v automatickém režimu a řídí nádrž samostatně, nebo jestli je v manuálním režimu. Pokud je nastaven manuální režim, objeví se třetí řádek, ve kterém se zobrazuje, jaký typ regulátoru je nastaven.

Kód pro toto zobrazení je:

```
disp(sprintf('%s %d\n%s %s\n', 'Sledovana nadrz: ', nadrz, 'Rezim: ',
AM_popis{AM}));

if AM == 2
    disp(sprintf('\n\n%s %s', 'Typ regulatoru: ', PR_popis{PR}));
end
```

Pole *AM_popis* a *PR_popis* obsahují textové možnosti z menu *Rezim* a *Typ regulatoru*, jenž se následně, pomocí parametru AM a PR, vypíšou na subsystém. Tyto parametry se nastaví přímo vybráním některé z možností v menu, jak jsem už naznačil na začátku této kapitoly.

# 7 Frekvenční průtokoměr

V úloze je pro každou z nádob nainstalován vrtulkový průtokoměr. Jedná se o typ *mpm 2F66D55.* Průtokoměry posílají signál, který je roven 6900 impulsům na litr. Při jejich uvádění do provozu jsem nejdříve musel vyřešit jejich zapojení a následně pak komunikaci se Simulinkem, aby se mohli hodnoty průtoků zobrazovat při používání modelu.

Pro zapojení jsem si musel nejprve najít datasheet pro příslušné průtokoměry. Vzhledem k tomu, že průtokoměry už byly nainstalované, žádnou dokumentaci jsem k nim neměl. Na internetu se mi podařilo najít datasheet, který obsahoval mimo jiné i průtokoměry, které jsou v modelu použity. Celý datasheet je možné najít v literatuře [11].

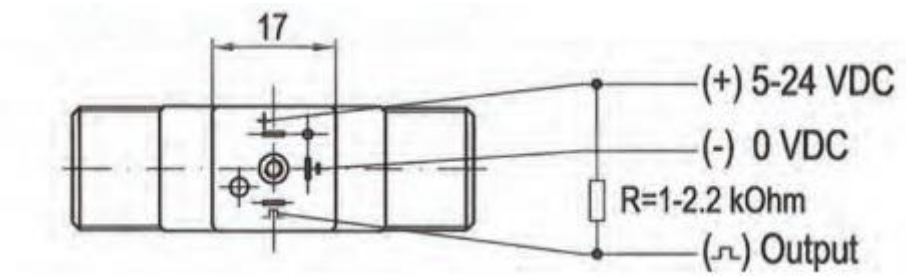

*Obr. 19: Nezapojený vrtulkový průtokoměr*

## 7.1 Zapojení průtokoměrů

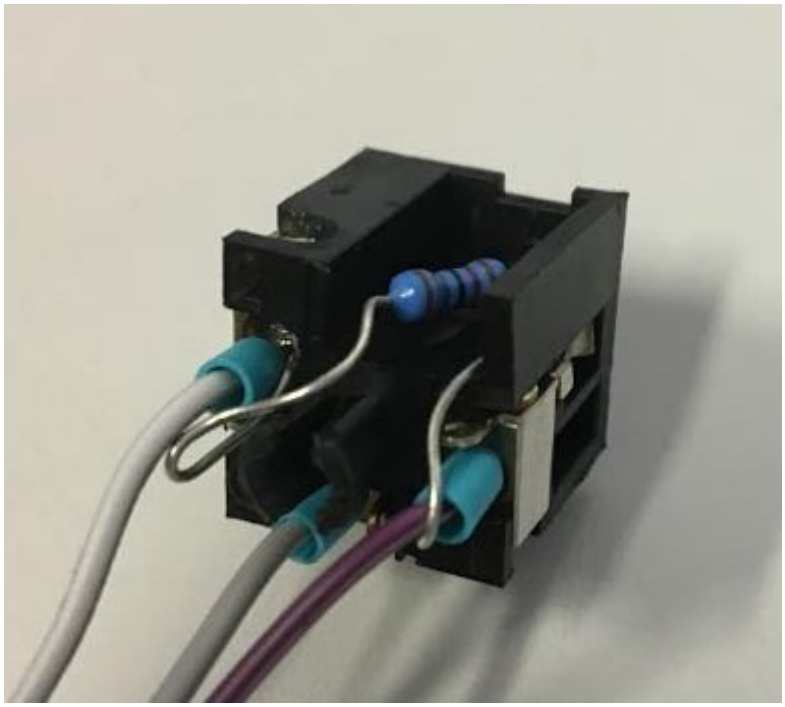
Průtokoměr má tři porty. Jeden napájecí, jeden s nulovým potenciálem a jeden, který vede signál. Pro mne toto zapojování bylo prvním setkáním s elektrickým zapojováním jakéhokoliv hardwaru, proto jsem si musel osvojit i naprosté základy, jako je svlékání izolace z drátů a krimpování. Poté co jsem si připravil všech 9 potřebných drátů, jsem je připojil k jednotlivým průtokoměrům a vyvedl na zadní stranu desky. Napájecí dráty jsem připojil k analogovému výstupu zásuvného modulu i-8024, který jsem takto použil jako zdroj. Dráty nesoucí signál připojil do tří portů v modulu i-8084 řídící jednotky WinPac. Nulové

potenciály jsem připojil k zemi a zároveň propojil mezi sebou, aby došlo k vyrovnání potenciálů.

Dalším krokem bylo připojení tzv. pull-up rezistoru o hodnotě 1400 Ω mezi napájecí drát a drát se signálem u každého z průtokoměrů, tak jak je ukázáno v datasheetu (viz Obr. 20). Toto jsem vyřešil způsobem, kde jsem rezistor připojil do příslušných svorek přímo v průtokoměru, jak je vidět na Obr. 21. Účel pull-up rezistoru je v určení logické hodnoty, když není definovaná žádným zařízením. Jinak řečeno, aby byla jasně určená logická hodnota ve stavu, kdy je spínač v průtokoměru rozepnut [12].




*Obr. 20: Schéma zapojení průtokoměru [12]*

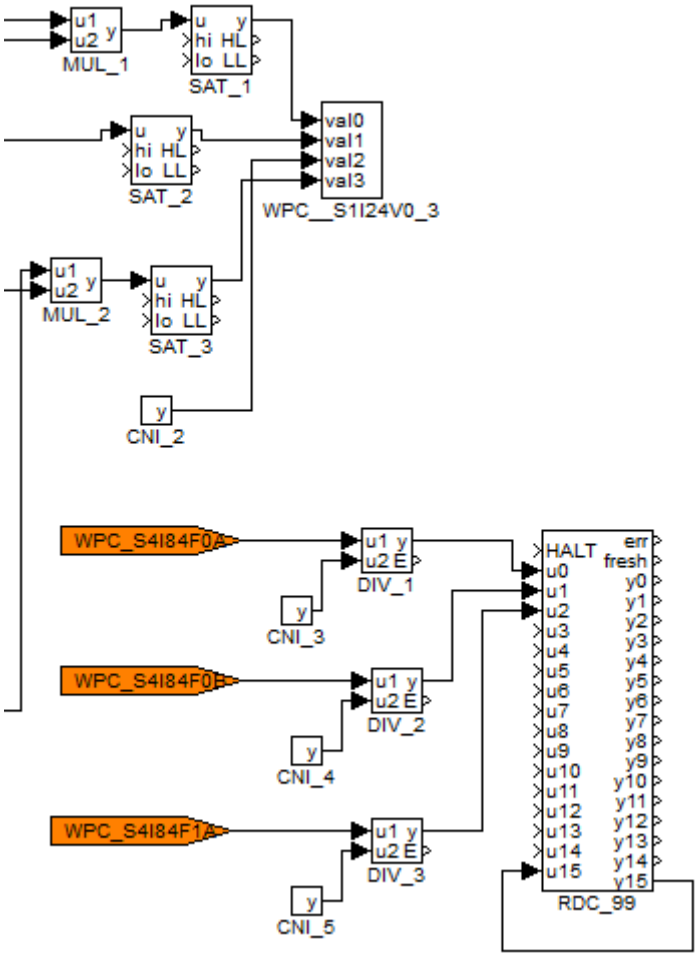


*Obr. 21: Zapojení průtokoměru s pull-up odporem*

## 7.2 Softwarové řešení průtokoměrů

Mým hlavním cílem bylo průtokoměry uvést do provozu, proto jsem se nezabýval příliš následným využíváním hodnot průtoku přímo v úlohách automatického řízení, ale pouze je připravil pro další použití a zprostředkoval komunikaci se Simulinkem.

### 7.2.1 Řešení – RexDraw




*Obr. 22: Softawrové řešení průtokoměrů v RexDraw*

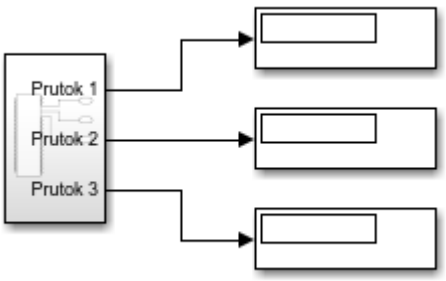
Pro řešení jsem použil paralelní zásuvný modul i-8084W. Ten může pracovat v pěti režimech. Přesnější popis lze najít v literatuře [13]. Pro můj účel snímání průtoku vrtulkovým průtokoměrem se nejvíce hodí frekvenční režim. Ten se v názvu bloků *From* (na Obr. 22 označené oranžově) projeví písmenem F, čímž se nastaví do požadovaného režimu. Výstupem je pak přímo frekvence průtokoměru. V řídícím systému RexDraw jsem tedy z bloků se jménem WPC__S4I84F0A, WPC__S4I84F0B a WPC__S4I84F1A vyvedl hodnoty, které přísluší průtokoměrům. Ty jsem vydělil hodnotou 6900, čímž jsem z frekvence dostal přímo průtok v litrech za sekundu. Přepočítané hodnoty jsem následně poslal do nového

RDC bloku s názvem *RDC_99* na porty u0 až u2. Následně jsem ještě přiřadil konstantní hodnotu 10 na vstup *val2* bloku WPC__S1I24V0_3, ze kterého jsou průtokoměry napájeny, aby bylo na výstupu při zapnutí požadované napětí právě 10 V.

Při zkoušení mého zapojení došlo při kompilaci k erroru 137, který byl pojmenován jako chyba ovladače WPC. Tato chyba byla způsobena nově přidanými bloky *From.* Začal jsem tedy zkoumat, v čem by mohla být chyba. Jelikož jsem při uvádění do provozu postupoval v RexDraw striktně podle návodu [13], domníval jsem se, že tam by mělo být vše správně. Stáhl jsem si proto novější verze programu RexDraw a RexView, konkrétně 2.10.8. V té došlo ke kompilaci prakticky stejného souboru bez problému, čímž se potvrdil můj předpoklad, že se nejednalo o syntaktickou chybu. Bohužel z novější verze programu RexView se nešlo připojit k řídící jednotce, a tak do ní zkompilovaný soubor nahrát. S největší pravděpodobností je nutné aktualizovat i ovladač v řídící jednotce.

### 7.2.2 Řešení – MATLAB Simulink

V Simulinku jsem pro přehlednost vytvořil nový samostatný subsystém, ve kterém jsem z bloku *RDC_99* vyvedl hodnoty z portů y0 až y2 do výstupů subsystému. Ty jsem připojil na tři displeje, na kterých bude zobrazena hodnota průtoku v litrech za sekundu. Vzhled subsystému lze vidět na Obr. 23. Celý soubor s komunikací se nazývá *prutok.mdl.*



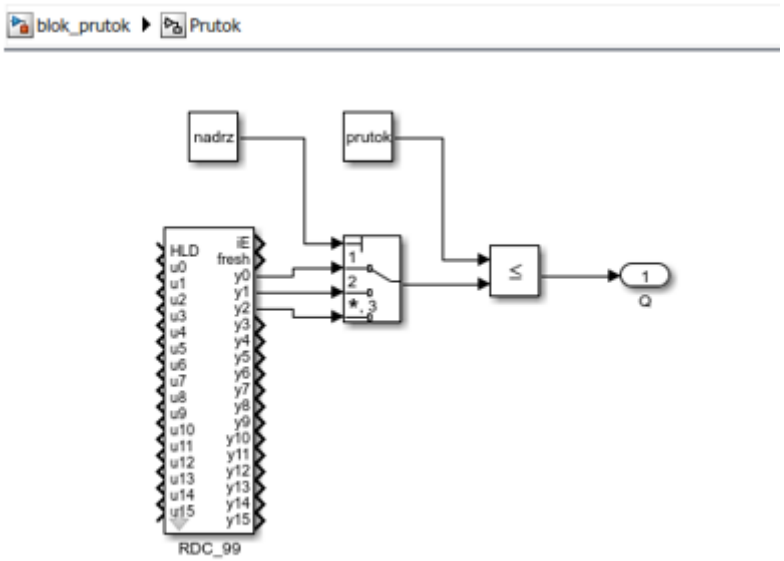

*Obr. 23: Softwarové řešení průtokměrů v Simulinku*

## 7.3 Použití průtokoměrů

Využití průtokoměrů v úloze může být hned několik. Obecně průtokoměry slouží jako další zdroj informací o systému. To se může využít pro zjištění současné konfigurace modelu za běhu, kdy může nedopatřením dojít k uzavření ventilu. Další příkladem je zjišťování poruchy systému, kdy může dojít k ucpání některého z výtoků nádob. To mohou využít studenti předmětů, které se zabývají řízením takových to systémů trochu více do hloubky.

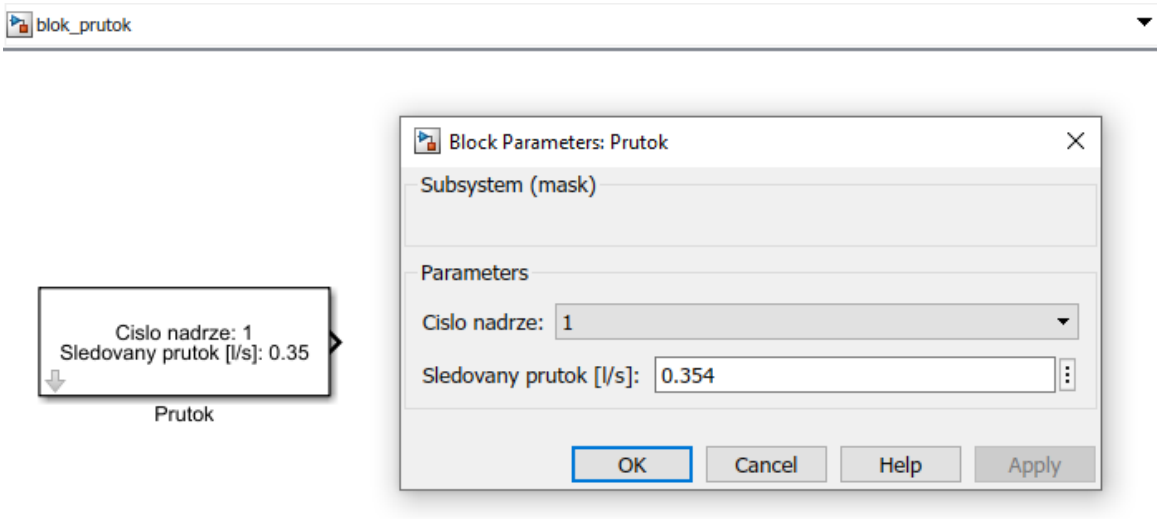
Co se týče úlohy logického řízení, může být hodnota průtoku jedné z nádrží využita jako další podmínka pro sepnutí nebo vypnutí některého z čerpadel. Stačí pouze vytvořit subsystém, který se velmi podobá tomu s názvem *hladina*, který jsem popisoval v kapitole 6.1. Pro ukázku jsem se ho rozhodl vytvořit. Pojmenoval jsem ho *prutok.*

V RexDraw není potřeba cokoliv měnit nebo přidávat a stačí tedy pouze popsat princip v Simulinku. Uvnitř tohoto subsystému se z bloku RDC_99 přijme současná hodnota průtoků. Pomocí parametru *nadrz* se vybere, ze které nádrže chce uživatel průtok sledovat. Následně se daná hodnota porovná s nastavenou hodnotou uživatele. Pokud je průtok menší, z bloku vychází nula. Pokud je průtok naopak větší, z bloku vychází jednička. Struktura subsystému je vidět na Obr. 24.




*Obr. 24: Vnitřní struktura bloku prutok*

Vytvořil jsem i masku subsystému, která na blok vypíše současné nastavení tohoto bloku. Na první řádek se vypíše číslo sledované nádrže a na druhý hodnota průtoku zaokrouhlená na dvě desetinná místa. Při rozkliknutí bloku uživatel vybere nádrž ze seznamu. Požadovanou hodnotu průtoku zadává ručně. Názorná ukázka je vidět na Obr. 25.




*Obr. 25: Maska bloku prutok*

Další způsob využití může být jednoduše dodělán v případě, že bychom chtěli průtokoměry jako takové ovládat. Vzhledem k tomu, že jsou napájeny z výstupů jednoho ze zásuvných modulů jednotky WinPac, místo permanentního sepnutí daného vstupu pomocí konstanty lze poměrně snadno nastavit, aby docházelo k zapnutí a vypnutí pomocí přepínače v Simulinku. Stačilo by hodnotu z přepínače, tedy jedničku nebo nulu, vynásobit s hodnotou, která je do vstupu přivedena v RexDraw a průtokoměry by byly napájeny vždy jen v případě, že dojde k sepnutí přepínače v Simulinku.

# 8 Závěr

Práce obsahuje celkový postup mé inovace laboratorního modelu Soustava tří nádob. Ta se skládala z popsání a pochopení současného stavu modelu, kde jsem se musel seznámit jak s jeho hardwarovou, tak softwarovou stránkou. Obzvláště u softwaru bylo ze začátku problém se zorientovat, vzhledem k tomu, že to bylo mé první setkání s daným softwarem. složitosti a nepřehlednosti některých ze souborů, ale nakonec se mi t

Dílčím úkolem této práce bylo popsání základního fungování použitého softwaru. Jmenovitě se jednalo o řídící systém REX Controls, jehož hlavní funkcí bylo řízení modelu a komunikace s jeho řídící jednotkou. Dále pak software MATLAB Simulink, ve kterém se zprostředkovala vizualizace a ovládání úloh.

Následoval popis původního řešení úloh v obou zmíněných softwarech, kde jsem rozebral vnitřní strukturu souborů a uživatelského prostředí a v některých případech se zamyslel nad problémy současného řešení.

Hlavní částí bylo vytvoření nového uživatelského prostředí pro každou z úloh, které se dá použít pro jejich ovládání. Mým cílem nebylo vše předělat od základů, nýbrž co nejvíce ovládání zjednodušit a zpřehlednit, tudíž jsem předělával a vytvářel nové ovládání pouze v některých částech úloh. Během uvádění mých řešení do provozu nastával problém s kompatibilitou souborů, vzhledem k velmi staré verzi MATLABU na počítači, který se v úloze používá. Do odevzdání této práce se problémy nepodařilo zcela vyřešit, a to ani přes ukládání do starších verzí programu. Prvním řešením tohoto problému mohla být instalace nové verze programu MATLAB na používaný počítač. Ovšem vzhledem k tomu, že tento počítač běží na Windows XP, nejnovější verze, kterou tento operační systém podporuje je MATLAB R2015a. Od školy je ovšem nejstarší poskytnutá verze MATLAB R2015b. Tudíž nemám možnost nainstalovat kompatibilní verzi programu. Druhou možností bylo předělat celé uživatelské prostředí na počítači v laboratoři. To by bylo vzhledem k rychlosti počítače velmi náročné a zároveň bych musel odebrat některé z použitých prvků, protože starší verze těmito komponenty nedisponuje. Řešení je tedy připravené k použití v mnou použité verzi R2019a, pouze stačí dořešit novější verzi programu MATLAB.

Poslední částí bylo zprovoznění nainstalovaných průtokoměrů v každé z nádob. Původně jsem předpokládal, že zprovoznění budu řešit spíše teoreticky, ale ukázalo se, že zapojení hardwaru mi hodně prospělo v získání nových zkušeností s elektrickým zapojením. Pro průtokoměry jsem udělal i základní softwarové řešení, abych otestoval jejich funkčnost a dále se pak zamyslel nad jejich dalším využití na modelu. Bohužel z důvodu staré verze ovladače na řídící jednotce jsem moje řešení nemohl otestovat, ale po konzultaci s vedoucím práce jsme se shodli, že řešení by mělo být správné, a je tak připravené k použití.

# Reference


[1] V. Stanislav: *Logické řízení výšky hladiny v nádržích.* [online], 2006 [cit. 2019-11-06]. Dostupné z: http://vlab.fs.cvut.cz/navody/files/kaskada_logicka.pdf

[2] V. Stanislav: *Frekvenční charakteristika soustavy tří nádrží.* [online], 2006 [cit. 2019-11-06]. Dostupné z: http://vlab.fs.cvut.cz/navody/files/kaskada_frekvencni.pdf

[3] V. Stanislav: *Laboratorní úloha Seřízení PI regulátoru.* [online], 2006 [cit. 2019-11-06]. Dostupné z: http://vlab.fs.cvut.cz/navody/files/kaskada_serizeniPI.pdf

[4] *Začínáme s řídícím systémem REX.* REX Controls s.r.o., Plzeň, březen 2009

[5] *Stručný popis řídícího systému REX.* REX Controls s.r.o., Plzeň, březen 2009

[6] *Function Blocks of the REXYGEN System*. *REXYGEN* [online]. [cit. 2019-11-07]. Dostupné z: https://www.rexygen.com/doc/ENGLISH/MANUALS/BRef/BRef_ENG.html

[7] *Standar WinPAC 8000 controller*. icpdas [online]. [cit. 2019-11-11]. Dostupné z: http://www.icpdas.com/products/PAC/winpac/wp-8x41.htm

[8] *Datasheet wp-8141*. icpdas [online]. [cit. 2019-11-11]. Dostupné z: https://files.icpdas-europe.com/products/K119483/web/icpdas/datasheet-wp-8141-en-g.pdf

[9] *Simulink*. MathWorks [online]. [cit. 2019-11-12]. Dostupné z: https://www.mathworks.com/help/simulink/

[10] *Ovladač WpcDrv systému REX pro WinPAC 8000.* REX Controls s.r.o., Plzeň, duben 2009

[11] *Turbin Flowmeter Vision*. platon-direct [online]. [cit. 2019-12-27]. Dostupné z: http://www.platon-direct.eu/pdf/LICO_Vision1000_2000_EN_V2_2014_compressed.pdf

[12] *Pull-up rezistory.* Elektroráj [online]. [cit. 2020-1-11]. Dostupné z: http://www.elektroraj.cz/2015/12/16/pull-up-rezistory/

[13] *Ovladač WpcDrv systému REX pro WinPAC 8000.* REX Controls s.r.o., Plzeň, únor 2016

# Seznam obrázků



# Seznam tabulek

# Seznam elektronických příloh

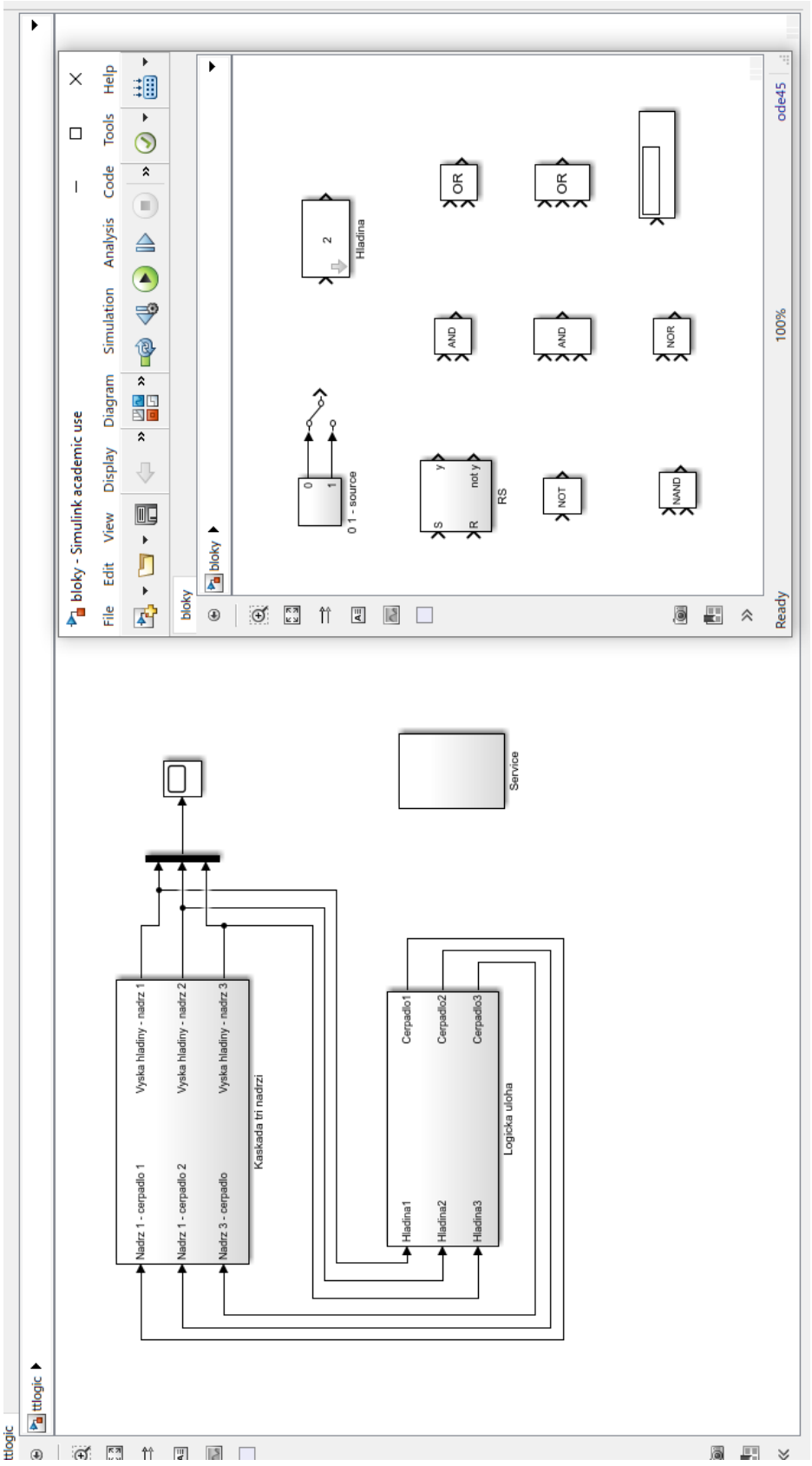
Všechny elektronické přílohy jsou zabaleny v archivu *elektronicke_prilohy.zip*, který je vypálen na přiložené CD.

1. elektronicke_prilohy.zip
    1.1. moje_reseni
        1.1.1. freq
            1.1.1.1. freq_uloha.mdl
            1.1.1.2. run_freq.m
            1.1.1.3. schema.jpg
            1.1.1.4. ttinit.m
            1.1.1.5. ttload.m
            1.1.1.6. ttstart.m
            1.1.1.7. ttzapisfreq.m
        1.1.2. logika
            1.1.2.1. knihovna_blok.mdl
            1.1.2.2. logicka_uloha.mdl
            1.1.2.3. logicka_uloha_STOP.mdl
            1.1.2.4. run.m
        1.1.3. regul
            1.1.3.1. run.m
            1.1.3.2. schema.jpg
            1.1.3.3. ttinit.m
            1.1.3.4. ttload.m
            1.1.3.5. ttstart.m
            1.1.3.6. ttstop.mdl
            1.1.3.7. ttzapis.m
            1.1.3.8. uzavreny_regulacni_obvod.mdl
            1.1.3.9. vykresli.m
        1.1.4. rex
            1.1.4.1. logic
                1.1.4.1.1. exec.mdl
                1.1.4.1.2. ttlogic.mdl
                1.1.4.1.3. WpcDrv.rio
            1.1.4.2. regul
                1.1.4.2.1. exec.mdl
                1.1.4.2.2. ttregul.mdl
                1.1.4.2.3. WpcDrv.rio
        1.1.5. blok_prutok.mdl
        1.1.6. priklad.mdl
        1.1.7. prutok.mdl
    1.2. puvodni_reseni
        1.2.1. matlab
            1.2.1.1. logic
                1.2.1.1.1. bloky.mdl

1.2.1.1.2. run.m
1.2.1.1.3. ttlogic.mdl
1.2.1.1.4. ttlogicstop.mdl
1.2.1.2. regul
1.2.1.2.1. run.m
1.2.1.2.2. run_freq.m
1.2.1.2.3. ttfreq.mdl
1.2.1.2.4. ttinit.m
1.2.1.2.5. ttload.m
1.2.1.2.6. ttregul.mdl
1.2.1.2.7. ttstart.m
1.2.1.2.8. ttstop.mdl
1.2.1.2.9. ttzapis.m
1.2.1.2.10. ttzapisfreq.m
1.2.1.2.11. vykresli.m
1.2.2. rex
1.2.2.1. logic
1.2.2.1.1. exec.mdl
1.2.2.1.2. exec.rex
1.2.2.1.3. ttlogic.mdl
1.2.2.1.4. WpcDrv.rio
1.2.2.2. regul
1.2.2.2.1. exec.mdl
1.2.2.2.2. exec.rex
1.2.2.2.3. ttregul.mdl
1.2.2.2.4. WpcDrv.rio
1.3. RL_2_10_ML8_0_x64

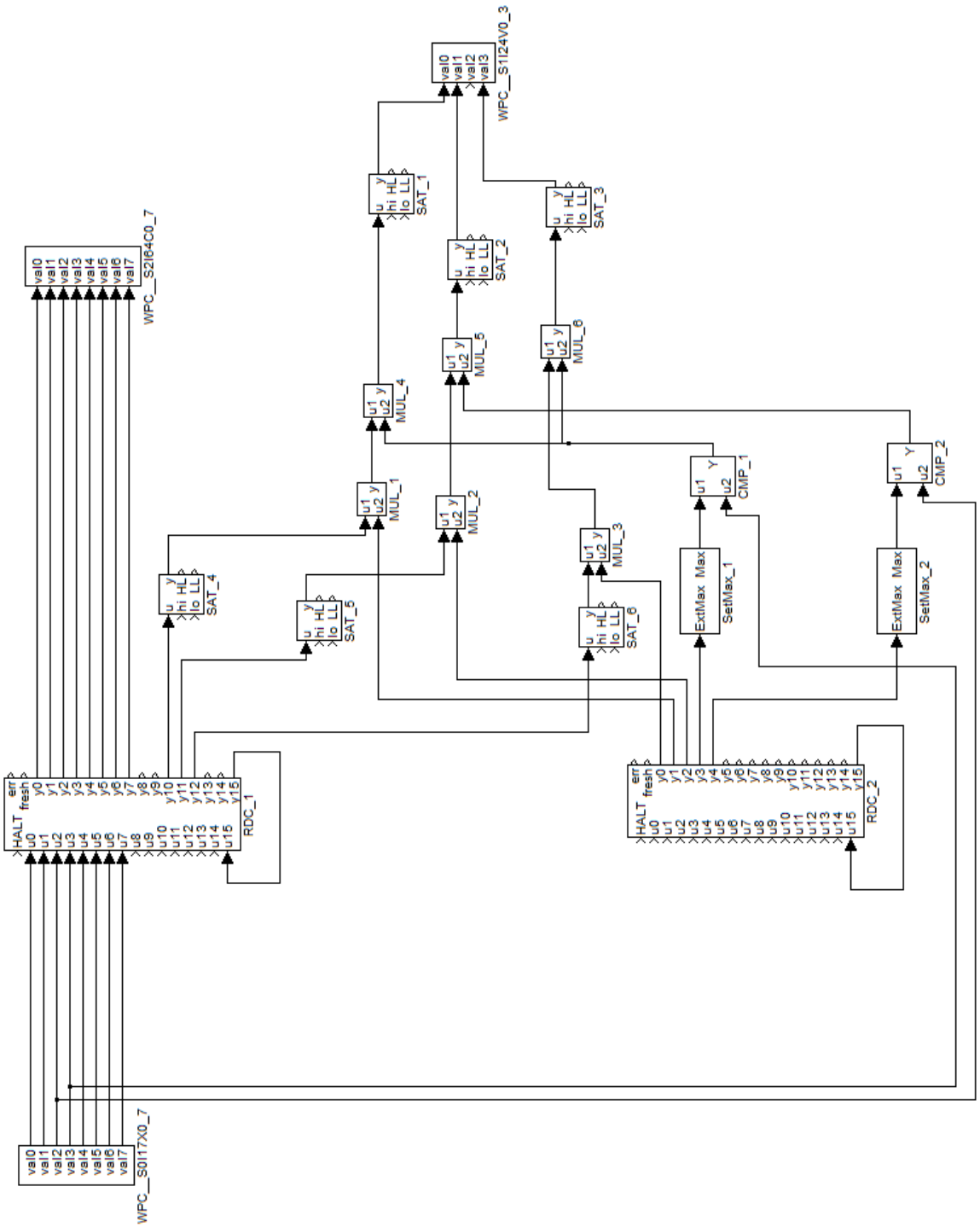
# Příloha 1

Původní řešení logické úlohy v MATLAB Simulink



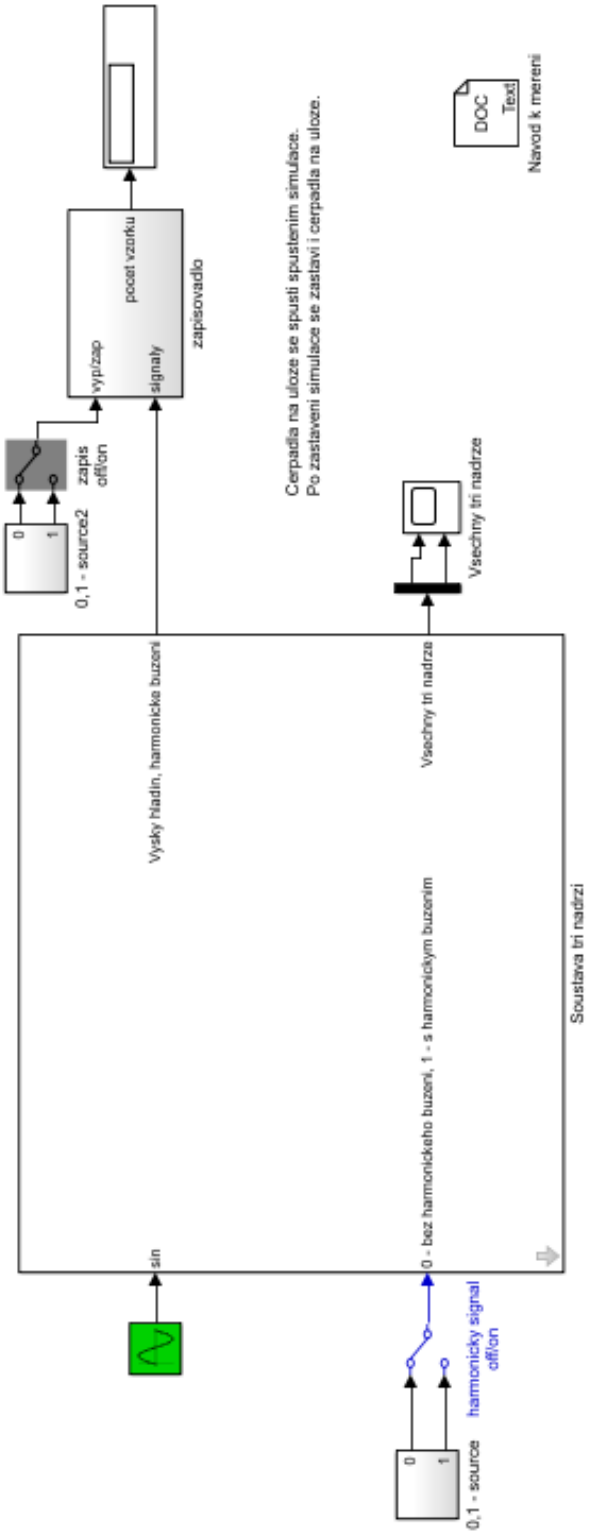
# Příloha 2

Původní řešení logické úlohy v RexDraw



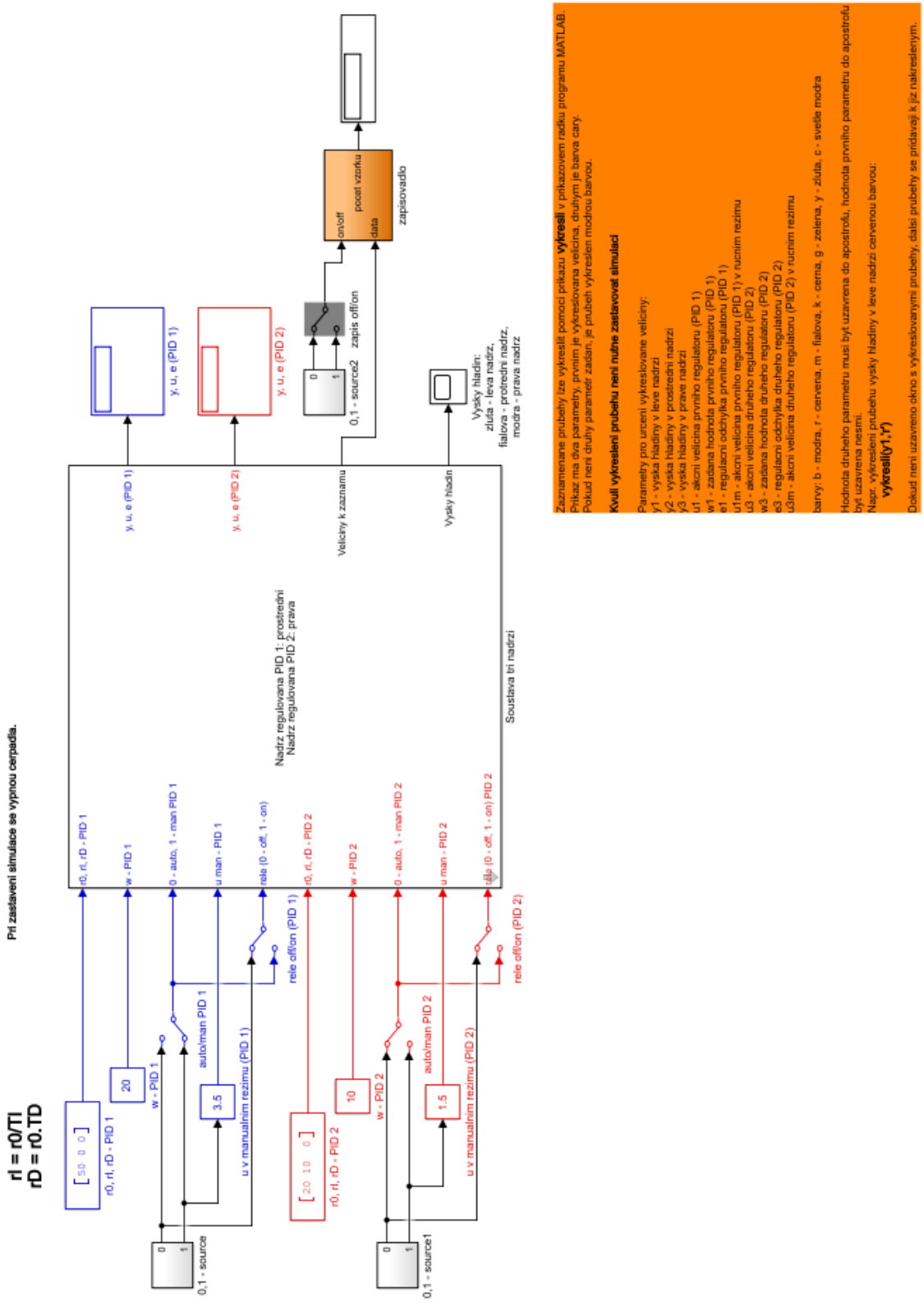
# Příloha 3

Původní řešení frekvenční úlohy v MATLAB Simulink



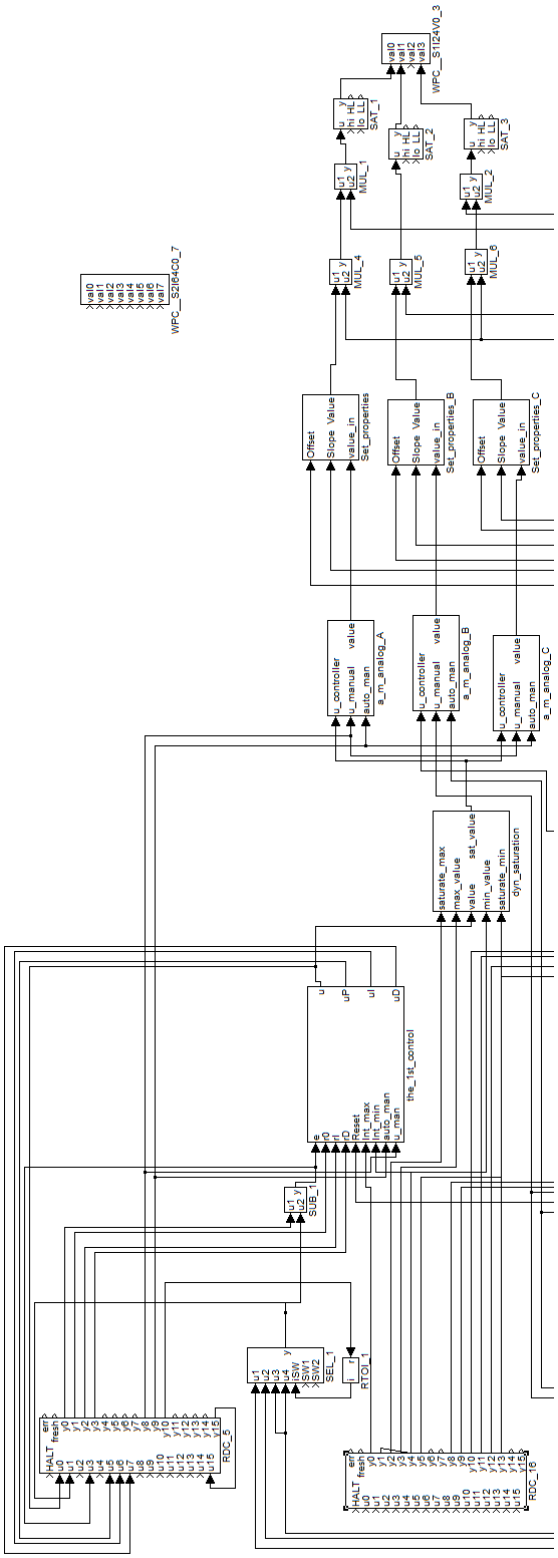
# Příloha 4

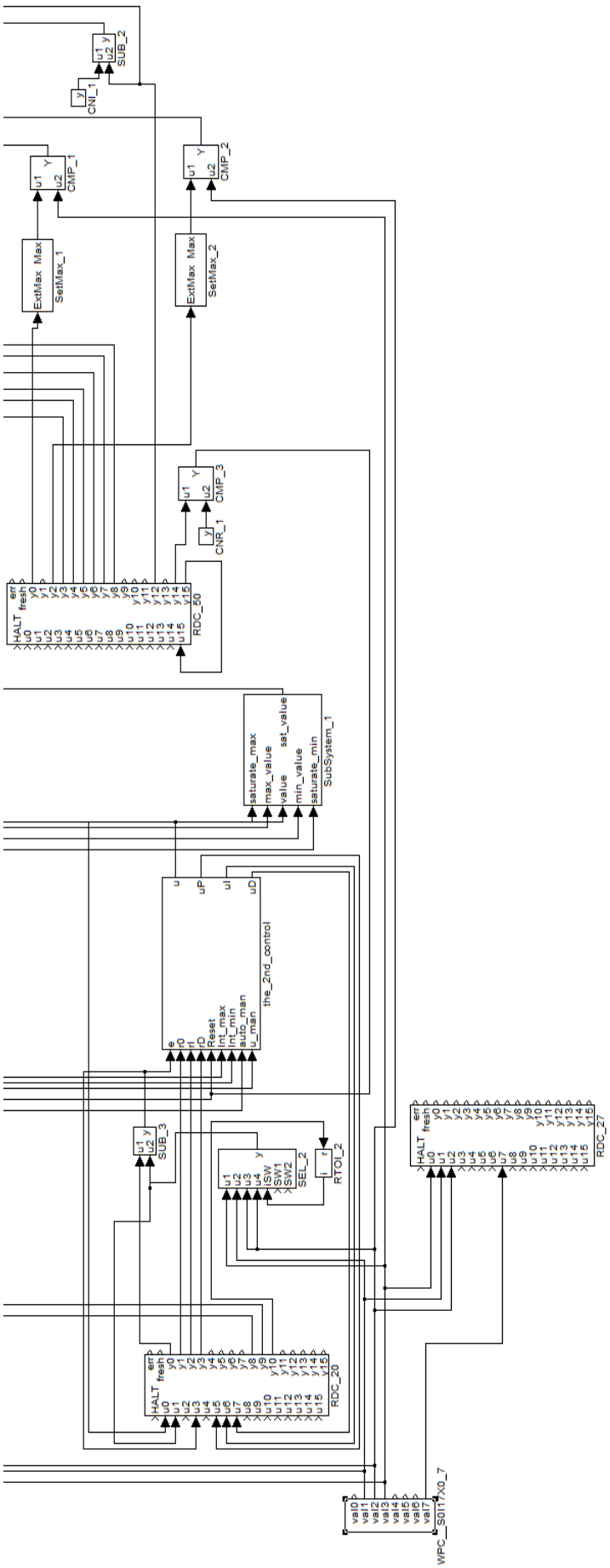
Původní řešení uzavřeného regulačního obvodu v MATLAB Simulink



# Příloha 5

Původní řešení frekvenční úlohy a uzavřeného regulačního obvodu v RexDraw




SUB_2
CNI_1
CMP_1
CMP_2
SetMax_1
SetMax_2
CMP_3
CNR_1
RDC_50
SubSystem_1
the_2nd_control
SUB_3
SEL_2
RTOI_2
RDC_20
RDC_27
WPC_S0117X0_7