SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-72807

INTERNETOM PODPOROVANÁ TVORBA BLOKOVÝCH SCHÉM

DIPLOMOVÁ PRÁCA

2018 **Bc. Richard Führich**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-72807

INTERNETOM PODPOROVANÁ TVORBA BLOKOVÝCH SCHÉM

DIPLOMOVÁ PRÁCA

Študijný program:	Aplikovaná informatika
Číslo študijného odboru:	2511
Názov študijného odboru:	9.2.9 Aplikovaná informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci záverečnej práce:	doc. Ing. Katarína Žáková, PhD.
Konzultant:	doc. Ing. Katarína Žáková, PhD.

Bratislava 2018 **Bc. Richard Führich**

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Bc. Richard Führich
Diplomová práca:	Internetom podporovaná tvorba blokových schém
Vedúci záverečnej práce:	doc. Ing. Katarína Žáková, PhD.
Konzultant:	doc. Ing. Katarína Žáková, PhD.
Miesto a rok predloženia práce:	Bratislava 2018

Cieľom tejto práce je vytvorenie webovej aplikácie do vzdialeného laboratória, ktorá umožní zostavovanie blokových schém na základe preddefinovaných blokov. Aplikácia by mala byť vytvorená pomocou aktuálnych technológii určených na tvorbu webových aplikácii. Na základe analýzy problému a existujúcich riešení bol vytvorený návrh aplikácie, a boli vybrané technológie potrebné na úspešné vytvorenie navrhnutej aplikácie. Výsledná aplikácia bola vytvorená za pomoci spojenia technológii HTML, CSS a JavaSript, ktoré boli rozšírené prostredníctvom dostupných knižníc. Požadovaná funkcionalita “drag and drop“ bola zabezpečená knižnicou Fabric.js. Súčasťou aplikácie na vytváranie blokových schém je možnosť dané schémy priebežne ukladať.

Kľúčové slová: bloková schéma, matlab, rlc

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Bc. Richard Führich
Master's thesis:	Web supported block scheme creation
Supervisor:	doc. Ing. Katarína Žáková, PhD.
Consultant:	doc. Ing. Katarína Žáková, PhD.
Place and year of submission:	Bratislava 2018

Abstract

Keywords: block scheme, matlab, rlc

Poďakovanie

Chcem sa poďakovať vedúcemu záverečnej práce, ktorým bola doc. Ing. Katarína Žáková, PhD., za odborné vedenie, rady a pripomienky, ktoré mi pomohli pri vypracovaní tejto diplomovej práce.

Obsah

Zoznam obrázkov a tabuliek

Zoznam algoritmov

Úvod

V našej práci, internetom podporovanej tvorby blokových schém, je cieľom navrhnutie mechanizmu vytvárania blokových schém a ich ukladania do vhodného formátu spracovateľného v iných aplikáciách. Tento návrh by mal zohľadniť dostupné technológie a ich možnosti, vrátane knižníc pre tvorbu web aplikácii využívajúcich technológiu "drag and drop". Na základe predchádzajúceho návrhu a prostredníctvom preskúmaných technológii sa v tejto práci popisuje proces a spôsob vytvorenia webovej aplikácie do vzdialeného laboratória, ktorá umožňuje zostavenie a priebežné ukladanie blokových schém. Táto aplikácia rovnako obsahuje najčastejšie používané typy blokových schém a ich zodpovedajúce bloky, ktoré sa používajú na riadenie dynamických systémov. Taktiež zovšeobecňuje postup pre prípadnú implementáciu ďalších druhov blokových schém a ich blokov do tejto aplikácie.

Analýzou existujúcich riešení v kapitole 2 bolo možné určiť technológie použiteľné na riešenie problému internetom podporovanej tvorby blokových schém, vrátane vytvorenia mechanizmu skladania blokových schém, ich uloženia a následného vyhodnotenia. Nevýhodami skúmaných existujúcich riešení a ich aplikácii bola nejednotnosť jednotlivých spôsobov riešení danej problematiky. Vzhľadom na túto nejednotnosť je nemožné dať tie aplikácia dokopy a vytvoriť spoločné rozhranie na vytváranie a vyhodnocovanie jednotlivých druhov blokových schém, ktoré sú riešené v skúmaných aplikáciách.

Vytvorená webová aplikácia založená na poznatkoch získaných z aplikácii [1], [2], [3] pozostáva z používateľského prostredia navrhnutého pomocou technológii HTML, CSS a Boostrap a obsahuje pracovnú plochu, na ktorú je možné vykresľovať jednotlivé prvky blokových schém prostredníctvom JavaScript knižnice Fabric.js. Všetky podstatné údaje tejto aplikácie je možné uložiť a načítať v ktoromkoľvek bode vytvárania blokovej schémy prostredníctvom súboru obsahujúcom informácie vo formáte JSON, ktorý je možné vytvoriť v aplikácii. Jednotlivé technológie, programovacie jazyky, ich podporujúce knižnice a súborové formáty použité v tejto aplikácii sú popísané v kapitole 3.

Táto aplikácia rieši vyššie spomenutý problém nejednotnosti riešení jednotlivých skúmaných aplikácii a vytvára jednotné rozhranie na riešenie rôznych druhov blokových schém a taktiež umožňuje prípadné rozšírenie dostupných typov blokových schém a ich skupín blokov v budúcnosti.

1 Analýza problému

Prvým krokom riešenia problému internetom podporovanej tvorby blokových schém bolo bližšie analyzovať daný problém a možné spôsoby riešenia jednotlivých častí tohto problému. Presnejšie ide o podstatu a spôsob vytvárania blokových schém, popisovanie týchto schém matematicky a graficky, a následné riešenie jednotlivých typov blokových schém a získavanie požadovaných výstupov.

1.1 Riadiace systémy

Riadiace systémy [4] sú systémy, ktoré poskytujú požadovanú odozvu za pomoci ovládania výstupu. Na obrázku 1 je znázornený jednoduchý blokový diagram riadiaceho systému, ktorý je reprezentovaný jediným blokom.

Obr. 1: Jednoduchý riadiaci systém

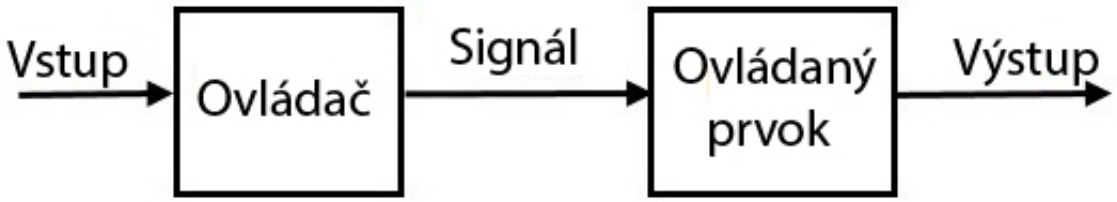
Príkladom takéhoto riadiaceho systému je riadenie dopravného osvetlenia. V tomto prípade je na riadiacu jednotku aplikovaná sekvencia vstupných signálov a výstupom je jedna z troch možných svetiel, ktoré bude svietiť určitý čas a ostatné dve svetlá budu počas tejto doby vypnuté. Z toho vyplýva, že riadiaci systém dopravného osvetlenia funguje na princípe času.

Riadiace systémy vieme rozdeliť na základe niekoľkých parametrov:

- Spojité a diskrétne riadiace systémy
  - toto rozdelenie závisí od typu použitého signálu
  - v spojitých riadiacich systémoch sú všetky signály nepretržité
  - v diskrétnych riadiacich systémoch existuje jeden alebo viac signálov s určitým časovým intervalom
- SISO a MIMO riadiace systémy
  - toto rozdelenie závisí od počtu vstupov a výstupov systému
  - Single Input and Single Output systémy obsahujú iba jeden vstup a výstup

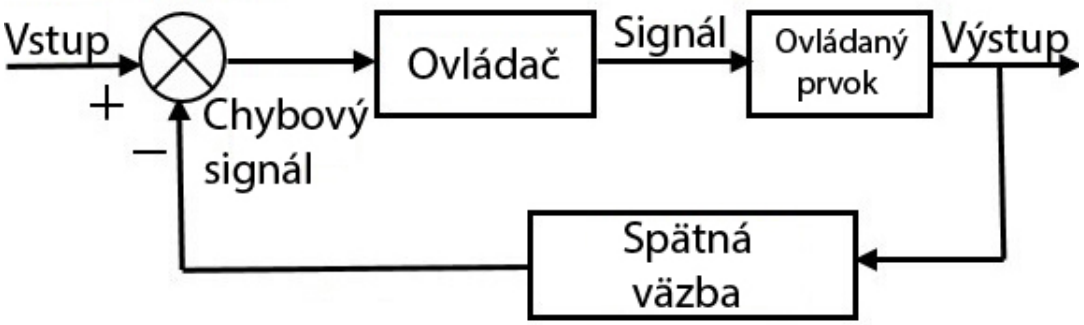
– Multiple Inputs and Multiple Outputs systémy obsahujú viac ako jeden vstup a výstup

Ďalším spôsobom klasifikácie riadiacich systémov je rozdelenie na zatvorené a otvorené riadiace systémy, ktoré sa určuje na základe cesty spätnej väzby. V otvorených riadiacich systémoch sa výstup nevracia späť na vstup, čo znamená, že riadenie je nezávislé od požadovaného výstupu.



Obr. 2: Otvorený riadiaci systém

V prípade otvoreného riadiaceho systému na obrázku 2 sa vstup posiela na ovládač, ktorý generuje signál. Tento signál je ďalej posielaný na prvok systému, ktorý sa má ovládať a ten produkuje výstup.



Obr. 3: Zatvorený riadiaci systém

Zatvorený riadiaci systém zobrazený na obrázku 3, tiež známy ako spätnoväzobný riadiaci systém, využíva princíp priamej cesty otvoreného riadiaceho systému, no obsahuje jednu alebo viac spätnoväzobných slučiek. To v jednoduchosti znamená, že časť výstupu je vracaná späť na vstup. Tieto riadiace systémy sú navrhované tak, aby automaticky dosahovali a udržiavali požadovaný výstup. Toto dosahuje pomocou generovania chybového signálu, rozdielu medzi výstupom a referenčným vstupom, čo znamená, že riadenie je istým spôsobom závislé od výstupu.

1.2 Matematické modely riadiacich systémov

Matematický model je skupina matematických rovníc, pomocou ktorých je možné popísať riadiace systémy. Tieto modely je možné použiť na analýzu a návrh daných systémov. Pod analýzou rozumieme postup, pri ktorom sa snažíme nájsť výstup, pričom poznáme vstup a matematický model, zatiaľ čo pri návrhu riadiaceho systému je zámerom nájsť matematický model na základe známeho vstupu a výstupu. Najčastejšie používané matematické modely sú:

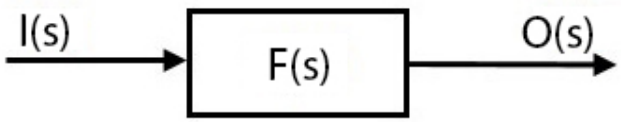
- model diferenciálnej rovnice
- model prenosovej funkcie
- model v stavovom priestore

1.3 Blokové schémy

Na popisovanie riadiacich systémov v obrázkovej forme sa používajú blokové schémy, ktoré sa skladajú z jedného bloku alebo kombinácie blokov. Medzi základné časti blokovej schémy patria blok, sčítavací bod a bod vetvenia. V zatvorenom riadiacom systéme zobrazenom na obrázku 3 je možné tieto tri prvky blokovej schémy identifikovať. Táto schéma pozostáva z troch blokov (*Ovládač*, *Ovládaný prvok* a *Spätná väzba*), jedného sčítavacieho bloku, do ktorého vchádza vstup systému a signál vysielaný spätnou väzbou a nakoniec jedeného bodu vetvenia pred výstupov systému, z ktorého sa vysiela spätná väzba.

Definícia bloku

Prenosová funkcia komponentu je popisovaná blokom, ktorý má jeden vstup a jeden výstup. Na obrázku 4 je znázornený jednoduchý blok s prenosovou funkciou `F(s)`, ktorého vstupom je `I(s)` a výstupom `O(s)`.



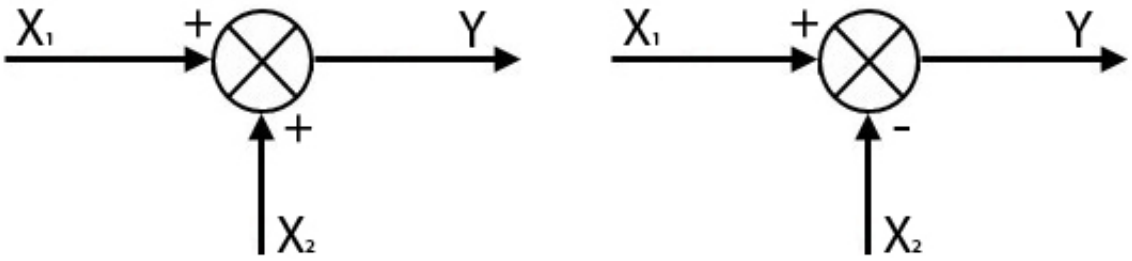
Obr. 4: Jednoduchý blok blokovej schémy

Na základe tohto obrázku je možné určiť výstup tohto bloku ako súčin jeho prenosovej funkcie a vstupu nasledovne:

$$O(s) = F(s)I(s)$$

Definícia bodu sčítavania

Pre účel definície uvažujeme, že bod sčítavania je reprezentovaný kruhom, v ktorom sa nachádza znak `X`. Tento prvok obsahuje dva alebo viac vstupov, jeden výstup a produkuje algebraický súčet alebo rozdiel jeho vstupov.

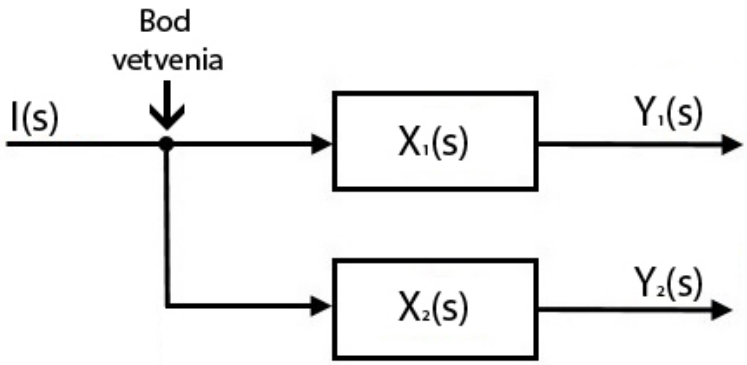


Obr. 5: Ukážka bodov sčítavania

Na obrázku 6 sú znázornené dva prípady bodov sčítavania s dvoma vstupmi. V týchto prípadoch sú týmito vstupmi X_1 a X_2, kde v prípade obrázku na ľavo sú oba vstupy kladné, zatiaľ čo v druhom prípade je jeden vstup kladný a druhý záporný. Výsledky ukážkových bodov sčítavania by boli $Y = X_1 + X_2$ pre ľavý obrázok a $Y = X_1 + (-X_2) = X_1 - X_2$ pre pravý obrázok. Rovnaký postup výpočtu výsledku bodu sčítavania sa používa v prípadoch, kedy do tohto bodu vchádza tri a viac vstupov.

Definícia bodu vetvenia

Bod vetvenia je bod v blokovej schéme, pomocou ktorého je možné jeden vstupný signál rozdeliť na viac ako jednu vetvu. Vďaka tomu je možné použiť ten istý vstupný signál na viac ako jeden blok alebo sčítavací bod. Týmto spôsobom je možné docieliť napríklad aj spätnú väzbu.



Obr. 6: Ukážka bodu vetvenia

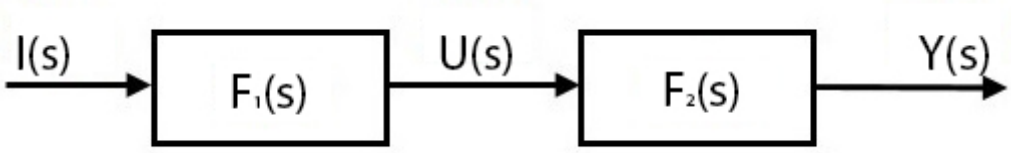
1.4 Bloková algebra

Vzťah medzi obrazmi vstupnej veličiny a výstupnej veličiny môžeme vyjadriť pomocou prenosu systému a jeho komponentov. Na vyjadrenie väzieb komplexnejších systémov sa používajú blokové schémy, pomocou ktorých vieme väčšie a komplexnejšie systémy rozložiť na spojenie elementárnych členov. Spôsob výpočtu prenosu celého systému z prenosov čiastkových, nazývame bloková algebra. Pre túto metódu výpočtu prenosu je podstatný spôsob zapojenia jednotlivých blokov, ktoré v blokových schémach poznáme tri:

- sériové
- paralelné
- antiparalelné

Sériové zapojenie

Pri sériovom zapojení je vstupom jedného z blokov výstup predchádzajúceho. Výstupný prenos je daný súčinom čiastkových prenosov systému.



Obr. 7: Sériové zapojenie

Podľa obrázka 7 teda môžeme výsledný prenos vypočítať vynásobením jednotlivých prenosov:

$$Y(s) = F_2(s)U(s) = F_1(s)F_2(s)I(s)$$

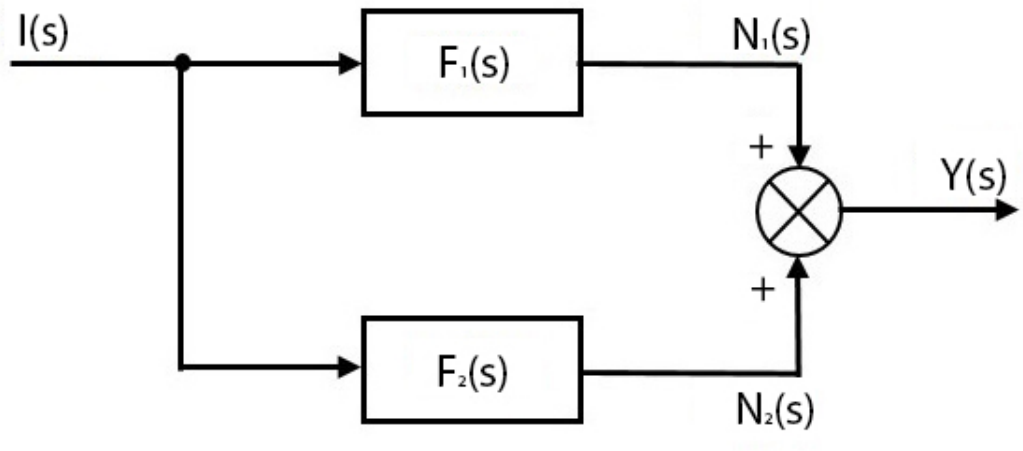
$$F_s = \frac{Y(s)}{I(s)} = F_1(s)F_2(s)$$

Na základe tohto výpočtu prenosu vieme takúto časť systému zjednodušiť do jedného bloku s prenosovou funkciou $F_1(s) * F_2(s)$ so vstupom *I(s)* a výstupom *Y(s)*.

Paralelné zapojenie

Bloky v paralelnom zapojení majú rovnakú vstupnú veličinu a výstupy týchto blokov sa sčítavajú. Pre obrázok 8 sú hodnoty čiastočných výstupov rovné:

$$N_1(s) = F_1(s)I(s) \quad \text{a} \quad N_2(s) = F_2(s)I(s)$$



Obr. 8: Paralelné zapojenie

Ďalej je možné dokončiť výpočet celkového prenosu tejto schémy, ktorý je pri paralelnom zapojení rovný súčtu čiastočných prenosov:

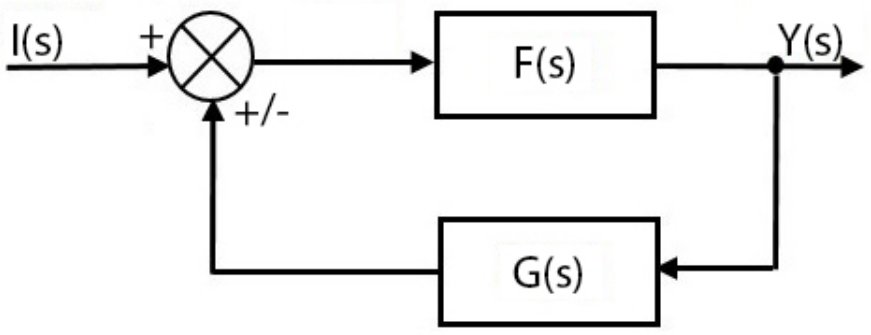
$$Y(s) = [F_1(s) + F_2(s)]I(s)$$

$$F_s = \frac{Y(s)}{I(s)} = F_1(s) + F_2(s)$$

V tomto prípade je teda možné paralelné zapojenie dvoch blokov zjednodušiť na jeden blok s prenosovou funkciou $F_1(s) + F_2(s)$, vstupom *I(s)* a výstupom *Y(s)*.

Antiparalelné zapojenie

Posledným z troch zapojení je antiparalelné, tiež nazývané spätnoväzobné zapojenie, pri ktorom sa výstupná veličina napája na sčítavací bod na začiatku zapojenia, kde sa pričíta alebo odčíta od vstupu schémy.



Obr. 9: Antiparalelné zapojenie

V prípade antiparalelného zapojenia na obrázku 9 je prenosom celej schémy zlomok zlo-

žený z prenosovej funkcie na priamej vetve, vydelenej súčinom prenosovej funkcie priamej vetvy a spätnej väzby:

$$Y(s) = F(s)[I(s) - G(s)Y(s)] \Rightarrow Y(s)[1 + F(s)G(s)] = F(s)I(s)$$

$$F_s = \frac{Y(s)}{I(s)} = \frac{F(s)}{1 \pm F(s)G(s)}$$

Podľa toho, či sa na vstupe schémy vstupný signál pričíta alebo odčíta, rozlišujeme medzi spätnými väzbami kladnými a zápornými. Prostredníctvom zápornej spätnej väzby sa získava takzvaná regulačná odchýlka, pomocou ktorej sa dosahuje zhoda vstupnej a výstupnej veličiny.

1.5 Elektrické obvody

Elektrický obvod [5] je obvod obsahujúci uzavretú trasu pre poskytovanie toku elektrónov zo zdroju napätia alebo prúdu. Prvky v elektrickom obvode môžu byť zapojené v sériovom zapojení, paralelnom zapojení alebo v kombinácii týchto zapojení. Jednotlivé prvky v týchto obvodoch sa môžu deliť podla niekoľkých kategórii:

- aktívne a pasívne prvky
- lineárne a nelineárne prvky
- jednosmerné a obojsmerné prvky

Rozdelenie na aktívne a pasívne prvky je určené podľa schopnosti prvku dodávať elektrickú energiu do obvodu:

- **Aktívne prvky** dodávajú elektrickú energiu iným prvkom, ktoré sa nachádzajú v obvode, ale zároveň sú schopné danú energiu aj spotrebovať. Príkladom sú zdroje napätia a prúdu.
- **Pasívne prvky** nie sú schopné do obvodu dodávať energiu, ale sú schopné ju absorbovať. Danú energiu uchovajú vo forme magnetického alebo elektrického poľa, prípadne ju premenia na teplo. Príkladom pasívnych prvkov sú rezistor, kondenzátor a cievka.

Rozdelenie jednosmerných a obojsmerných prvkov záleží od smeru toku prúdu cez jednotlivé prvky obvodu. Rozdelenie lineárnych a nelineárnych prvkov určuje voltampérová charakteristika, ktorá znázorňuje závislosti prúdu a napätia.

1.5.1 RLC obvod

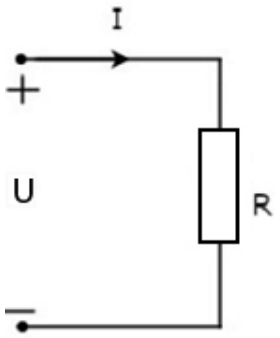
RLC obvod [6], inač nazývaný aj rezonančný obvod, je elektrický obvod pozostávajúci z rezistoru, kondenzátoru a cievky v sériovom alebo paralelnom zapojení.

Rezistor

Hlavnou úlohou rezistora ako pasívnej elektrickej súčiastky je obmedzenie toku prúdu, upravenie úrovne signálu, rozdelenie napätia a podobne. Podľa Ohmovho zákonu je napätie rezistora na obrázku 10 rovné súčinu prúdu tečúceho cez rezistor a jeho odporu, matematicky vyjadrené nasledovne:

$$U = IR \quad \Rightarrow \quad I = \frac{U}{R}$$

kde U označuje napätie, I označuje prúd a R je odpor rezistora. Na základe tohto vzťahu je zjavné, že prúd tečúci cez rezistor je priamo úmerný napätiu zdroja a nepriamo úmerný odporu obvodu.



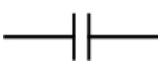
Obr. 10: Zapojenie rezistora

Kondenzátor

Kondenzátor je druhou pasívnou elektrickou súčiastkou, ktorá sa skladá z dvoch vodivých platničiek oddelených nevodivým prvkom a uchováva energiu v závislosti od napätia, ktoré na ňu pôsobí. Hodnota elektrického náboja kondenzátora závisí od napätia:

$$Q = CU$$

kde Q je elektrický náboj a C je kapacita kondenzátora.



Obr. 11: Schematická značka kondenzátora

Cievka

Poslednou z pasívnych elektrických súčiastok je cievka, ktorá uchováva energiu v magnetickom poli, keď ňou prechádza elektrický prúd. Celková veľkosť magnetického toku vytvorená cievkou závisí od elektrického prúdu, ktorý preteká súčiastkou:

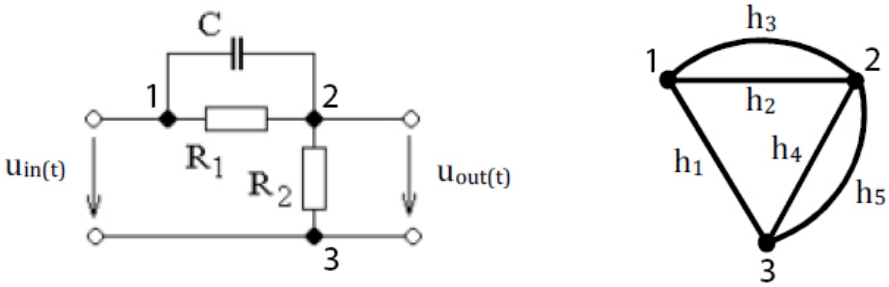
$$\phi = LI$$

kde ϕ je celkový magnetický tok a L je indukčnosť cievky.

Obr. 12: Schematická značka cievky

1.6 Grafy elektrických obvodov

Ide o grafickú reprezentáciu používanú na analýzu zložitých elektrických obvodov. V grafoch sa uzol považuje za spoločný bod dvoch a viacerých hrán, v niektorých prípadoch môže byť k uzlu pripojená jediná hrana. Za hranu sa považuje čiara spájajúca dva uzly. Akýkoľvek elektrický obvod je možné konvertovať na graf reprezentujúci tento obvod tak, že sa všetky pasívne súčiastky a zdroje napätia nahradia hranami.



Obr. 13: Prevod RLC obvodu na graf [7]

V obvode na obrázku 13 sú tri hlavné uzly označené číslami od jedna po tri a päť hrán, ktoré spájajú tieto uzly, medzi ktoré patria tri hrany obsahujúce pasívne súčiastky RLC obvodu a dve hrany obsahujúce napätia. Na pravej strane toho istého obrázku je zobrazený ekvivalentný graf pre tento obvod. V tomto grafe sú tri uzly označené číslami od jedna po tri, zhodné s uzlami v RLC obvode. V grafe sa tiež nachádza päť hrán označených kombináciou písmena h a čísla od jedna po päť, ktoré reprezentujú jednotlivé prvky obvodu.

1.6.1 Druhy grafov

Medzi základné druhy grafov rozdelených na základe vlastností ako spojitosť a orientácia grafov patria:

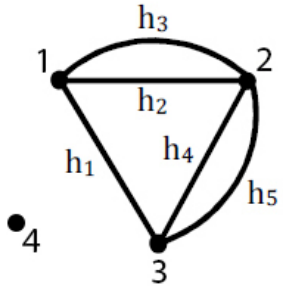
- orientovaný graf
- neorientovaný graf
- súvislý graf
- nesúvislý graf

Súvislý graf

V prípade, že existuje aspoň jedna hrana spájajúca každý z uzlov grafu, jedná sa o súvislý graf. Toto znamená, že všetky uzly v súvislom grafe majú na seba napojenú jednu alebo viac hrán, a ani jeden z uzlov v grafe nezostane oddelený. Graf zobrazený na obrázku 13 je príkladom súvislého grafu.

Nesúvislý graf

Ak existuje aspoň jeden uzol v grafe, na ktorý nie je napojená ani jedna hrana, jedná sa o nesúvislý graf. V tomto prípade existuje jeden alebo viac uzlov, ktoré sú oddelené.



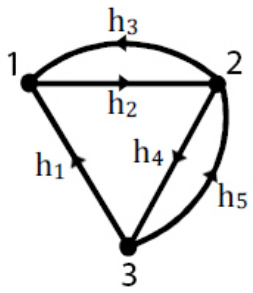
Obr. 14: Nesúvislý graf

Príklad nespojitého grafu je znázornený na obrázku 14, kde sa na každý z uzlov jedna až tri napája aspoň jedna hrana, no uzol štyri zostáva oddelený, bez napojenej hrany.

Orientované a neorientované grafy

Rozdiel medzi orientovanými a neorientovanými grafmi spočíva v značení smeru toku prúdu jednotlivých hrán. Ak jednotlivé hrany nie sú označené šípkami smeru toku prúdu, jedná sa o neorientovaný graf, ktorého príklad je znázornený na obrázku 13.

Naopak v prípade, že všetky hrany sú označené šípkami smeru toku prúdu, jedná sa o orientovaný graf, ktorého príklad je zobrazený na obrázku 15.



Obr. 15: Orientovaný graf

1.7 Matica incidencie

Matica incidencie [8] reprezentuje graf konkrétneho elektrického obvodu, a pomocou tejto matice je možné daný graf spätne vytvoriť. Ako už bolo spomenuté, grafy sa skladajú zo skupiny uzlov, ktoré sú spojené pomocou hrán. Spojenie medzi hranou a uzlom sa v tomto prípade nazýva incidencia. V prípade grafu s počtom uzlov n a počtom hrán h je možné vytvoriť maticu incidencie s n riadkami a h stĺpcami. Ak sa jedná o **orientovaný** graf, jeho ekvivalentná matica $n \times b$ môže nadobudnúť tri hodnoty:

- ak hrana pripojená na uzol smeruje od daného uzla, hodnota prvku je -1
- ak hrana pripojená na uzol smeruje do daného uzla, hodnota prvku je +1
- ak daná hrana nie je pripojená na daný uzol, hodnota prvku je 0

Podobne pri **neorientovanom** grafe je možné vytvoriť maticu incidencie, ktorej prvky môžu nadobudnúť hodnoty:

- ak je hrana spojená s daným uzlom, hodnota prvku je 1
- ak hrana nie je spojená s daným uzlom, hodnota je 0

Postup tvorenia matice incidencie spočíva v zvolení jedného z uzlov daného grafu a vyplnení riadku matice incidencie, ktorý korešponduje s číslom vybraného uzlu pomocou vyššie spomenutých pravidiel. To znamená, že pre uzol i a hranu j sa do matice A na pozícii A_{ij} zapíše hodnota zodpovedajúca stavu spojenia danej dvojice v grafe.

Príklad ekvivalentnej matice neorientovaného grafu zobrazeného na obrázku 13:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Na základe tejto matice incidencie je zjavné, že jednotlivé riadky reprezentujú tri uzly grafu. Stĺpce tejto matice reprezentujú päť prvkov grafu elektrického obvodu z obrázku 13, a práve preto je táto matica veľkosti 3×5. Hodnota prvku matice je 1 v prípadoch, kde sa hrana spája s konkrétnym uzlom. V ostatných prípadoch, kde sa na uzol daná hrana nenapája, sú hodnoty prvkov 0. Na základe tejto matice by teda bolo možné spätne zostaviť ekvivalentný graf reprezentujúci elektrický obvod.

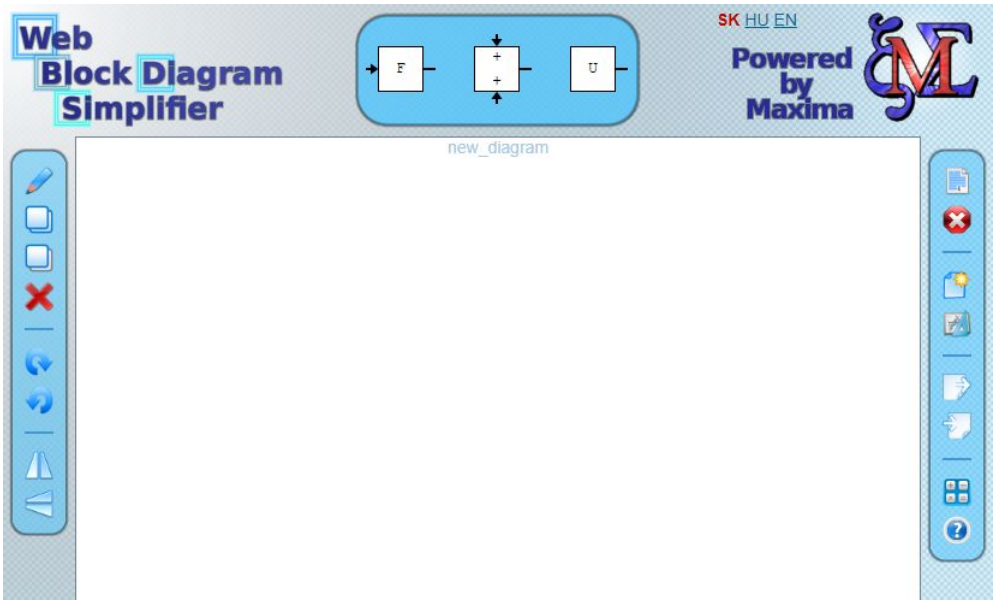
2 Analýza existujúcich riešení

2.1 Internetom podporované riešenie blokových schém

Zámerom tejto práce [1] je návrh a vytvorenie aplikácie slúžiacej na vytváranie a zjednodušovanie blokovej schémy za účelom získania výslednej prenosovej funkcie celého systému. Táto aplikácia sa skladá z dvoch časti - serverovej a klientskej. Serverová časť fungujúca na servery s operačným systémom Ubuntu má za úlohu spracovať a zjednodušiť blokovú schému pomocou aplikácie Maxima, ktorá sa používa na manipuláciu so symbolickými a číselnými hodnotami a výpočty s nimi. Na umožnenie práce s aplikáciou Maxima sa v tejto práci využíva zbierka PHP a PERL skriptov, ktoré medzi iným zabezpečujú vykonávanie Maxima príkazov s názvom MaximaPHP.

Serverová časť spolupracuje s klientskou časťou, ktorá poskytuje používateľské rozhranie na zostavovanie blokových schém a funguje za pomoci technológii ako sú:

- XHTML
- CSS
- JavaScript
- Javascript knižnice jQuery



Obr. 16: Používateľské rozhranie internetom podporovaného riešenia blokových schém

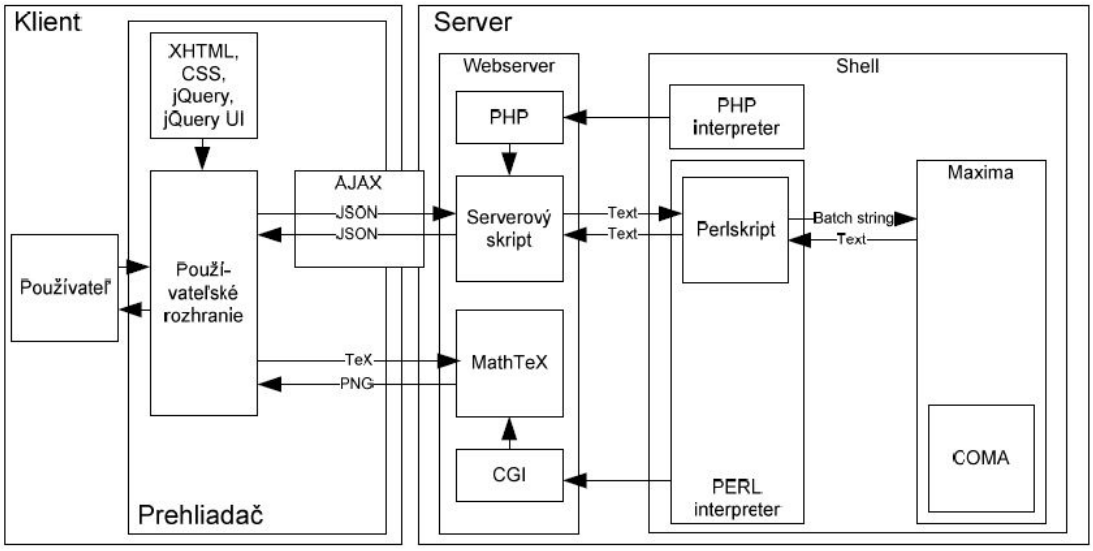
Používateľské rozhranie na obrázku 16 sa skladá z prvkov, ktoré umožňujú jednoduchú navigáciu a vytváranie blokových schém, ako sú menu a plátno, na ktorom je možné

za pomoci “Drag and drop“ funkcionality dané blokové schémy skladať. Používanie tejto funkcionality je umožnené za pomoci JavaScript knižnice jQuery. Základnou časťou tvorby blokových schém sú samotné bloky, ktoré táto práca rozdeľuje na dve časti - grafickú časť, ktorá sa zobrazuje používateľovi a časť obsahujúcu informácie o bloku a jeho parametre na aplikačnej úrovni práce. Ďalej samotné bloky rozdeľuje do dvoch kategórii:

- Základné bloky
  - blok s prenosovou funkciou
  - sčítavací blok
  - vstupno-výstupný blok
- Vedľajšie bloky
  - spojenia medzi blokmi
  - uzol

Podstatnou súčasťou základných blokov sú takzvané “piny“, ktoré reprezentujú vstupné a výstupné porty daného bloku s preddefinovanými pozíciami, pomocou ktorých je následne možné konkrétne bloky spájať.

Komunikácia medzi jednotlivými časťami aplikácie sa vykonáva za pomoci technológie AJAX, ktorá zabezpečuje prenos informácii o blokovej schéme medzi serverovou a klientskou časťou aplikácie.

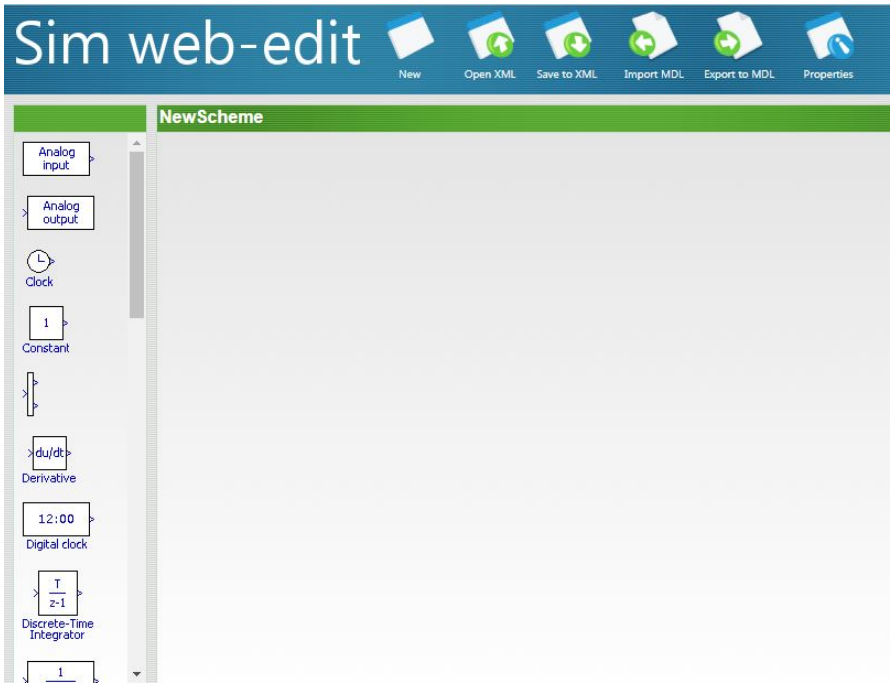


Obr. 17: Komunikácia aplikácie internetom podporovaného riešenia blokových schém [1]

Pomocou komunikácie na obrázku 17 a využitím prvkov používateľského prostredia na obrázku 16 je používateľ schopný zostaviť blokovú schému, ktorú neskôr pošle na spracovanie serverovej časti aplikácie, ktorá v prípade úspešného zjednodušenia a vyriešenia späť používateľovi odošle výslednú prenosovú funkciu zostaveného systému. V aplikácii je možné na plátno postupne pridávať bloky, upravovať ich parametre, rotáciu a polohu na plátne, vytvárať spojenia medzi jednotlivými blokmi, a v prípade správne zapojenej a nastavenej blokovej schémy požiadať o zjednodušenie a vyhodnotenie.

2.2 Internetom podporovaná modifikácia blokových schém

Podstatou práce internetom podporovanej modifikácie blokových schém [2] je webová aplikácia na vytváranie blokových schém za pomoci aplikácie Simulink, ktorá slúži ako prostriedok na riadenie dynamických systémov vo virtuálnych a vzdialených laboratóriách. Podobne ako pri aplikácii internetom podporovaného riešenia blokových schém, aj táto aplikácia sa skladá zo serverovej a klientskej časti. Technológie použité na zostrojenie klientskej časti a používateľského rozhrania sú XHTML, CSS, JavaScript a JavaScript knižnica jQuery. Dáta a parametre schémy a jednotlivých blokov sú ukladané vo formátoch JSON a XML, ktoré sú prenášané pomocou technológie AJAX medzi klientskou a serverovou časťou, ktorá je riešená za pomoci programovacieho jazyka PHP.



Obr. 18: Používateľské rozhranie internetom podporovanej modifikácie blokových schém

Používateľské rozhranie na obrázku 18 je zložené z troch oddelených častí - hlavné menu, zoznam použiteľných blokov a samotného plátna, na ktorom sa bloková schéma skladá. Hlavné menu obsahuje nastavenia a operácie so schémou, a ponuka blokov je zostavovaná dynamicky na základe dostupných blokov v konfiguračnom súbore. Ovládanie blokovej schémy a jednotlivých blokov je zabezpečené pomocou technológie "drag and drop", pomocou ktorej je možné pohybovať s jednotlivými prvkami aplikácie. Bloky tejto aplikácie sú rozdelené na dva druhy, podobne ako pri predchádzajúcej aplikácii:

- Základné bloky modelovania dynamických systémov
- Vedľajšie bloky doplňujúce blokovú schému - uzol a čiara spájajúca bloky

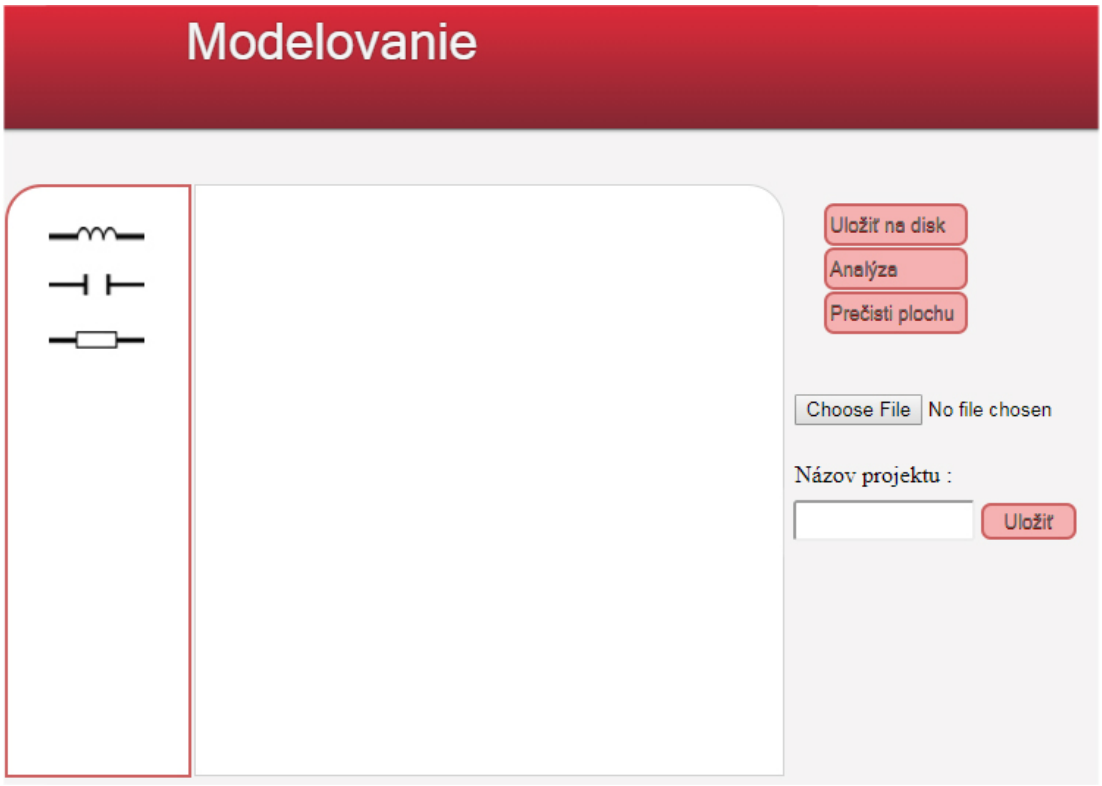
Každý zo základných blokov obsahuje parametre, ktoré je možné meniť pomocou formulára v dialógovom okne po kliknutí na daný blok, v ktorom je tiež možné nastaviť rotáciu zvoleného bloku. Tieto parametre sú ukladané v konfiguračnom súbore formátu XML. Podstatnou časťou tejto práce je vytvorenie výstupu vo formáte, ktorý je možné neskôr simulovať v prostredí aplikácie Simulink. V tomto prípade ide o súbor vytváraný na serverovej časti aplikácie podľa vytvorenej blokovej schémy a parametrov jednotlivých blokov, vo formáte `.MDL`.

Vytvorenie tohto súboru v prvom rade spočíva v úprave blokovej schémy do tvaru, ktorý skript vykonávajúci prevod na súbor dokáže spracovať. Ďalej aplikácia pokračuje v prenesení upravenej blokovej schémy od používateľa na serverovú časť aplikácie pomocou AJAX metódy `POST`. Táto upravená bloková schéma vo formáte JSON obsahuje informácie o jednotlivých blokoch a spojeniach medzi blokmi, a je potrebné ju na strane servera upraviť do požadovaného tvaru, zodpovedajúceho formátu `.MDL`. Štruktúra takéhoto súboru obsahuje informácie ako:

- Základné informácie o schéme
  - názov schémy
  - verzia aplikácie Simulink
  - informácie čase vytvorenia schémy
  - rozmery schémy
- Predvolené hodnoty nových blokov
- Predvolené parametre blokov
- Zoznam blokov a ich prepojení

2.3 Modelovanie lineárnych dynamických systémov

Obsahom aplikácie modelovania lineárnych dynamických systémov [3] je prostredie vytvorené na navrhnutie blokových schém rezonančných obvodov. Základom aplikácie je používateľské rozhranie zostavené pomocou technológii HTML5, CSS3, JavaScript a knižníc jQuery a Fabric.js, ktorá sa v tomto prípade využíva na jednoduchšiu prácu s HTML elementom Canvas. Na elemente Canvas sa vykonáva celý proces vykresľovania blokov lineárnych dynamických systémov, ich spájania a kompletizácie blokovej schémy pomocou technológie "drag and drop". Jednotlivé bloky blokovej schémy reprezentujúcej lineárne dynamické systémy sú realizované pomocou obrázkov na plátne, pridané funkcionalitou knižnice Fabric.js. Mimo toho aplikácia plne využíva riadenie udalostí ponúkané touto knižnicou, ktorou zabezpečuje väčšinu interakcie v prostredí plátna.



Obr. 19: Používateľské rozhranie modelovania lineárnych dynamických systémov

V prípade potreby poskytuje aplikácia možnosť uloženia vytvorenej blokovej schémy priamo v aplikácii alebo do súboru obsahujúceho informácie o plátne a jednotlivých blokoch vo formáte JSON. Neskôr je tak možné uloženú schému znova načítať. Po vytvorení kompletnej blokovej schémy lineárneho dynamického systému je možné v aplikácii získať výstupnú maticu incidencie, ktorá tento systém popisuje a vektor všetkých prvkov použitých v tomto systéme.

2.4 Zhrnutie

Na základe analýzy jednotlivých aplikácii v tejto kapitole je možné zhrnúť niekoľko bodov, ktoré vedú k riešeniu problému internetom podporovanej tvorby blokových schém. Podstatnou časťou riešenia tohto problému je výber technológii a programovacích jazykov použiteľných na vytvorenie vhodnej aplikácie na tvorbu, spracovanie a vyhodnotenie blokových schém rôznych druhov. Medzi najčastejšie používané technológie na vytvorenie funkčnej prezentačnej stránky webovej aplikácie patria HTML, CSS, JavaScript a ich rozširujúce knižnice, či frameworky. V spolupráci s týmito technológiami sa používajú dátové formáty JSON a XML na uchovávanie a prenášanie dát používaných aplikáciou. Spomenuté a ďalšie technológie použiteľné na riešenie daného problému sú vysvetlené a popísané v kapitole 3.

Každá z prác popísaných v tejto kapitole vytvára riešenie daného problému na základe svojho návrhu a z toho dôvodu je takmer nemožné dané aplikácie spojiť a vytvoriť jednotné rozhranie, v ktorom by používateľ bol schopný pracovať so schémou svojho výberu. Zámerom našej práce je vytvorenie jednotného rozhrania na vykresľovanie blokov a tvorbu blokových schém podľa vlastného výberu a následnej možnosti získania výstupu pre vytvorené schémy. Taktiež je potrebné zabezpečiť možnosť rozšírenia druhov vytvoriteľných blokových schém pridaním potrebných skupín blokov.

3 Použité technológie

V tejto kapitole sa píše o technológiách použiteľných na vytvorenie grafickej a aplikačnej časti internetom podporovanej tvorby blokových schém. Vysvetľuje zámer danej technológie spolu s jej výhodami a prípadnými nedostatkami.

3.1 HTML

HyperText Markup Language je najzákladnejší stavebný kameň všetkých webových aplikácii a stránok. Popisuje a definuje obsah webovej stránky spolu s jej základným rozložením. Po jeho boku sú na tvorbu funkčných webových stránok používané technológie ako CSS a JavaScript, ktoré daný základ upravujú vizuálne a pridávajú mu možnosť interakcie. HTML používa takzvané "značky" na ohraničovanie jednotlivých prvkov webovej stránky ako sú obrázky alebo text, a tie zobrazuje vo webovom prehliadači. Medzi elementárne HTML značky patria napríklad `<html>`, `<head>`, `<body>`, ale aj `<div>`, `<p>` a mnoho iných. Tieto elementy môžu ďalej obsahovať atribúty, ktoré sú často potrebné pre ich správne fungovanie alebo ich jednoducho upravujú. Pre našu aplikáciu je významný element `<canvas>`, ktorý je základom riešenia problému internetom podporovanej tvorby blokových schém.

3.1.1 Canvas

Spolu s piatou verziou HTML štandardu bol vydaný aj element canvas [9]. Tento element umožňuje dynamické vykresľovanie 2D tvarov a bitmapových obrázkov primárne pomocou JavaScript-ového kódu. Oblasť plátna, na ktorú možno kresliť definuje HTML kód pomocou atribútov výšky a šírky. K tejto oblasti sa dá pristupovať pomocou JavaScript kódu a prostredníctvom niekoľkých funkcii je možné dynamicky generovať požadovanú grafiku. Základná funkcionalita tohto elementu je však veľmi obmedzená, a preto je potrebné využitie externých knižníc, ktoré umožnia jednoduchšiu prácu a omnoho viac možností.

3.2 CSS

Na popisovanie a upravovanie prezentačnej stránky dokumentu napísaného v jazyku HTML sa používajú takzvané kaskádové štýly. Táto technológia bola vytvorená s účelom oddeliť prezentačnú a obsahovú stránku pomocou vecí ako sú usporiadanie, štýl písma a farby, čím zvyšuje dostupnosť informácii a flexibilitu [10]. Všetky tieto zmeny je možné vykonať jednoducho a rýchlo priamo v danom CSS dokumente bez potreby zasahovať do HTML kódu.

3.2.1 Bootstrap

Na zjednodušenie procesu vytvárania vizuálne vyhovujúcej prezentačnej časti webovej stránky sa často používajú front-end frameworky, akým je napríklad Bootstrap. Jedná sa o open-source framework obsahujúci šablóny na štylizáciu textu, formulárov, navigácie a iných prvkov webovej stránky na základe HTML a CSS. Rovnako poskytuje využitie JavaScript rozšírení, ktoré umožňujú implementáciu animácii a iných interaktívnych prvkov. Bootstrap je podporovaný na všetkých aktuálnych webových prehliadačoch a poskytuje takzvaný responzívny dizajn, ktorý dynamicky upravuje obsah webovej stránky na základe veľkosti okna webového prehliadača a rozlíšenia obrazovky zariadenia.

3.3 JavaScript

JavaScript [11] je interpretovaný programovací jazyk, ktorý podporuje rôzne programovacie štýly, vrátane objektovo orientovaného, funkcionálneho a riadeného udalosťami. Ako už v predchádzajúcich častiach bolo spomenuté, JavaScript je najčastejšie používaný vo webových stránkach a aplikáciach, kde poskytuje používateľovi mieru interakcie, pracuje s webovým prehliadačom a upravuje obsah HTML dokumentu [10]. Syntakticky je podobný jazykom ako sú C, C++ a Java. Ide o takzvaný voľne písaný jazyk, čo znamená, že premenné nemusia mať definované dátové typy. Medzi základné podporované dátové typy tohto jazyka patria čísla, textové reťazce a boolovské hodnoty. Aj napriek tomu, že bol tento jazyk pôvodne implementovaný iba vo webových prehliadačoch na strane klienta, aktuálne sa často nachádza aj na webových serveroch, v databázach, či dokonca PDF aplikáciách.

3.3.1 jQuery

Ide o JavaScript knižnicu, ktorej úlohou je zjednodušiť vytváranie skriptov [12] na prácu s HTML kódom na strane klienta a uľahčenie narábania s JavaScript kódom. jQuery rieši mnoho častých úkonov, ktoré by vyžadovali niekoľko riadkov JavaScript kódu a obalí ich do funkcie, ktorú stačí zavolať jedným riadkom kódu. Medzi hlavné vlastnosti jQuery knižnice tiež patria:

- výber a práca s elementami v HTML dokumente
- manipulácia s CSS kódom
- AJAX
- metódy na obstarávanie udalostí v HTML
- efekty a animácie

3.3.2 AJAX

Asynchronous JavaScript and XML [13] je skript na strane klienta komunikujúci so serverom alebo databázou, ktorý poskytuje spôsob výmeny dát a aktualizovanie časti webovej stránky bez potreby obnovenia celej stránky. AJAX je v podstate všeobecný názov pre rôzne JavaScript techniky používané na pripojenie sa na webový server, dynamicky bez potreby načítania viacerých stránok, primárne za pomoci `XmlHttpRequest` objektu. Hlavné výhody používania AJAX sú:

- asynchrónne volania - umožňujú webovému prehliadaču vyhnúť sa čakaniu na odpoveď zo servera
- zvýšenie rýchlosti - bez potreby obnovenia stránky a čakania na odpoveď zo servera sa zvyšuje rýchlosť aplikácie
- zníženie zátaže na sieť - keďže sa medzi klientom a serverom neposiela celá stránka, ale iba časť obsahu, ktorý sa mení, znižuje sa zátaž na server

3.4 JSON

JSON, skratka pre JavaScript Object Notation [14] je formát súboru používaný na uchovávanie a prenášanie dát medzi serverom a webovou aplikáciou. Tento formát je často používaný na synchrónnu a asynchrónnu komunikáciu, a ako alternatíva pre formát XML. Je to jazykovo nezávislý dátový formát a mnohé jazyky, vrátane jazyku JavaScript, obsahujú kód na generovanie a spracovanie JSON formátu. Jeho oficiálny MIME[1] typ je `application/json` a jeho obsah pozostáva z dvoch prvkov - kľúča a hodnoty, ktoré dokopy vždy tvoria dvojice. V prípade kľúča ide o textový reťazec uzavretý v úvodzovkách, zatiaľ čo hodnota môže byť typu:

- Pole - skupina hodnôt
- Boolovská hodnota - true alebo false
- Číslo
- Objekt - pole párov kľúča a hodnoty
- Textový reťazec

[1] `https://www.iana.org/assignments/media-types/media-types.xhtml`

Formát zápisu jednotlivých typov hodnôt:

```
Jednoduchá hodnota ako číslo, reťazec alebo boolovská hodnota
"kľúč": "hodnota"

Pole hodnôt
"kľúč": ["hodnota1", "hodnota2"]

Objekt zabalený v inom objekte
"kľúč": {
        "kľúč zabaleného objektu1":"hodnota",
        "kľúč zabaleného objektu2":"hodnota"
        }
```

3.5 Drag and drop

Drag and drop, po slovensky ťahaj a pusti, je jedna z kľúčových častí riešenia aplikácie na tvorenie blokových schém. Táto funkcionalita umožní používateľovi aplikácie pohybovať s jednotlivými prvkami na plátne, avšak podpora drag and drop v HTML je pre potreby tejto práce nedostačujúca, preto bolo potrebné využitie externých knižníc.

3.5.1 Fabric.js

Ako už bolo spomenuté v kapitole 3.1.1, HTML canvas umožňuje vykresľovanie 2D tvarov a obrázkov, ale kvalita jeho API je na veľmi nízkej úrovni. V prípade potreby komplexnejšej práce s plátnom tento problém rieši práve knižnica Fabric.js [15]. Táto JavaScript knižnica výrazne uľahčuje prácu s HTML plátnom - automaticky poskytuje potrebnú funkcionalitu drag and drop, možnosť práce s SVG grafikou, objektový model pre plátno a jeho ovládanie je založené na udalostiach na plátne. Medzi objekty v tejto knižnici sa radia základné tvary ako čiary, kruhy, trojuholníky, štvorce, text a ich zoskupenia, ale aj komplexnejšie objekty ako obrázky a cesty, ktoré sa podobajú SVG `<path>` elementom. Jednoduchý príklad vykreslenia objektu štvorca je znázornený v nasledujúcej ukážke:

```
Získanie referencie na objekt plátna
var canvas = new fabric.Canvas('c');

Vytvorenie objektu štvorca
var rect = new fabric.Rect({
  left: 100,
```

```
    top: 100,
    fill: 'red',
    width: 20,
    height: 20
});

Pridanie štvorca na plátno
canvas.add(rect);
```

Podstatnou časťou tejto knižnice je jej architektúra založená na udalostiach, medzi ktoré patria napríklad:

- Udalosti na základe použitia myši
  - `mouse:down/mouse:up` - pri zatlačení/pustení ľavého tlačítka myši
  - `mouse:over/mouse:out` - pri nabehnutí/opustení objektu myšou
- Udalosti na základe zmeny stavu objektov
  - `object:added/object:removed` - pri vytvorení/zmazaní objektu
  - `object:moving` - pri pohybe objektu
  - `selection:created` - pri zvolení jedného z objektov
  - `selection:updated` - pri zmene zvoleného objektu
  - `path:created` - pri vytvorení objektu typu Path
- Udalosti na základe zmien na plátne

Tieto udalosti sa zachytávajú vždy ako sa vykoná daný úkon, a je možné pomocou nich riadiť väčšinu funkcionality, ktorá sa vykonáva na plátne. Flexibilita tejto knižnice zabezpečuje, že je možné upravovať jednotlivé prvky podľa požiadaviek a je možné ich rovnako zakázať alebo povoliť.

3.6 KaTeX

KaTeX je JavaScript knižnica [16], pomocou ktorej je možné vo webových prehliadačoch jednoducho zobrazovať rovnice, funkcie a iné matematické zápisy v tvare TeX - formátovacom systéme navrhnutom za účelom jednoduchého písania kvalitne štylizovaného textu. Hlavnými bodmi tejto knižnice sú:

- Rýchlosť - všetky zápisy skladá synchrónne a nie je potrebné obnoviť celú stránku
- Kvalita výpisov - rozvrhnutie knižnice je založené na štandarde pre vypisovanie TeX matematiky
- Je nezávislá od iných zdrojov
- Produkuje výpisy nezávisle od prostredia a prehliadača

Táto knižnica podporuje všetky často používané webové prehliadače a jej inštalácia spočíva iba v stiahnutí a pridaní knižnice do projektu. Zabezpečenie výpisu je vykonávané pomocou jednoduchého príkazu `katex.render`, napríklad:

```
katex.render("\dfrac{2s^3+s^2+s}{s^2+s+1}", HTMLelement);
```

Na základe tohto príkladu sa pomocou knižnice na stránke vypíše rovnica v tvare:

$$\frac{2s^3 + s^2 + s}{s^2 + s + 1}$$

4 Realizácia návrhu

V tejto kapitole sa píše o konkrétnej realizácii návrhu a jednotlivých častiach riešenia aplikácie na vytváranie blokových schém. Presnejšie ide o spôsob skladovania a vykresľovania jednotlivých blokov schémy a ich častí, vytváranie spojení medzi týmito blokmi, popis dátových štruktúr a ich parametrov využívaných pri práci s aplikáciou a vysvetlenie komunikácie medzi touto a externými aplikáciami, ktoré sú využívané na získanie potrebných výstupov pre túto aplikáciu.

Pri návrhu riešenia tejto aplikácie bolo potrebné zamerať sa na niekoľko požiadaviek, ktoré sa delia na funkcionálne a nefunkcionálne. Úlohou tejto práce je vytvorenie jednotného rozhrania na vykresľovanie blokov a tvorbu kompletných blokových schém podľa výberu používateľa, a následnú možnosť spracovania a získania výstupu vytvorenej blokovej schémy. Rovnako bolo potrebné navrhnúť aplikáciu tak, aby bolo možné jednoducho rozšíriť druhy schém, s ktorými môže používateľ pracovať a to pridaním skupiny blokov reprezentujúcich daný typ schémy.

Funkcionálne požiadavky

- Možnosť vykreslenia blokov vybranej blokovej schémy
- Podpora viacerých vstupných portov blokov
- Zabezpečenie dostatočnej interakcie s vytvorenými blokmi
  - pohybovanie blokov po plátne
  - rotácia blokov
  - mazanie blokov
  - možnosť nastavenia parametrov blokov
- Možnosť spájania blokov a vytvárania bodov vetvenia na týchto spojoch
- Export a import vytvorených blokových schém vo vlastnom formáte
- Vytvorenie výstupov pre jednotlivé druhy blokových schém
  - matica incidencie a vektor elementov blokovej schémy
  - prenosová funkcia systému
  - export do súboru formátu `.mdl` pre aplikáciu Simulink

Nefunkcionálne požiadavky

Aplikácia by mala byť podporovaná všetkými často používanými, aktuálnymi webovými prehliadačmi, bez potreby dodatočnej inštalácie akejkoľvek technológie na strane používateľa. Taktiež by mala byť dostatočne jednoduchá, intuitívna a nenáročná na sieťové prostriedky, či prostriedky používateľa. Z grafickej stránky by mala aplikácia obsahovať responzívny dizajn, ktorý zabezpečí kvalitné zobrazenie na akomkoľvek zariadení.

4.1 Práca s blokmi

Jedným z najpodstatnejších elementov aplikácie zameranej na vytváranie blokových schém sú samotné bloky, preto bolo potrebné zamerať sa na správny spôsob skladovania a narábania s nimi. Tieto bloky sú skladované v HTML dokumentoch zodpovedajúcich jednotlivým schémam vo formáte dvoch JSON objektov. Jednotlivé JSON objekty zodpovedajú štruktúram nasledovne:

- `blockDrawData` - obsahuje informácie a dáta potrebné pre vykresľovanie jednotlivých blokov pomocou JavaScript knižnice Fabric.js
- `blockParameters` - obsahuje dáta špecifické pre konkrétny typ bloku, počty vstupných a výstupných portov, parametre bloku a iné informácie potrebné pre dátovú štruktúru `scheme`

Kľúčom JSON objektov sú v oboch prípadoch zhodné názvy blokov a ich hodnoty pozostávajú z ďalších objektov, ktoré popisujú jednotivé parametre a informácie.

4.1.1 Dáta pre vykresľovanie blokov

V JSON objekte `blockDrawData` je možné definovať dáta pre väčšinu objektov podporovaných knižnicou Fabric.js. Objekty tejto knižnice podporované našou aplikáciou sú:

- `rect` - objekt pre vykreslenie štvorca
- `circle` - objekt pre vykreslenie kruhu
- `triangle` - objekt pre vykreslenie trojuholníka
- `path` - objekt pre vykreslenie cesty na základe súradníc na plátne
- `text` - základný textový objekt
- `name (text)` - vedľajší textový objekt, rovnako sa jedná o objekt textu, no je možné nim odlíšiť zámer písaného textu (v prípade názvu pod blokom), ale nie je nevyhnutné ho používať

Je potrebné definovať jednotlivé objekty a všetky ich nevyhnutné parametre na základe toho, ako má daný blok vyzerať. Medzi základné parametre objektov často patria:

- `height` - číselný parameter zodpovedajúci výške objektu
- `width` - číselný parameter zodpovedajúci šírke objektu
- `fill` - textový reťazec definujúci farbu vnútra objektu, v prípade potreby je možné vypnúť pomocou hodnoty `false`
- `stroke` - textový reťazec definujúci farbu okrajov objektu, v prípade potreby je možné vypnúť pomocou hodnoty `false`
- `left/top` - číselný parameter odsadenia objektu
- `changeable` - boolovská hodnota špecifická pre textové objekty, ktorá v prípade hodnoty `false` zabraňuje prepisovaniu obsahu tohto objektu
- `invisible` - boolovská hodnota pre netextové objekty, ktorá v prípade hodnoty `false` zabraňuje zmene farby daného objektu

Príklad definície jednoduchého objektu v tomto prípade je znázornený na ukážke:

```
"Multiply":[
    {
        "type":"rect",
        "data":{
            "width":40,
            "height":40,
            "fill":"white",
            "stroke":"black"
        }
    },
    {   "type":"text",
        "Text":"F",
        "data":{
            "fontFamily": "Arial",
            "top":16,
            "left":16,
            "fontSize": 12,
```

```
            "fill": "black",
            "changeable":true
        }
    }
]
```

4.1.2 Dáta pre parametre blokov

Podobne ako pri dátach pre vykresľovanie blokov v kapitole 4.1.1 sa na uchovávanie dát parametrov blokov používa JSON objekt s názvom `blockParameters`. Tieto parametre sa po pridaní konkrétneho bloku na plátno ukladajú do štruktúry `scheme`, na základe ktorej funguje veľká časť aplikácie. Tento objekt sa delí na povinnú časť, uchovávajúcu informácie potrebné pre funkčnosť jednotlivých blokov a nepovinnú časť, ktorá reprezentuje jednotlivé HTML prvky ako sú textový input, select alebo checkbox. Povinná časť tohto JSON objektu pozostáva hlavne z prvkov ako sú:

- `io` - textová hodnota, na základe ktorej aplikácia rozhoduje aké porty budú bloku pridelené. Na výber sú hodnoty `in/out` v prípade, ak blok obsahuje iba vstup alebo výstup a `both` v prípade oboch
- `ports` - podobne ako pri parametri `io`, akurát sa jedná o horný a dolný port bloku
- `NumberOfInputs/Outputs` - číselná hodnota reprezentujúca počet vstupov a výstupov
- `BlockType` - textová hodnota upresňujúca typ bloku
- `hasExtra` - boolovská hodnota, určujúca či pre konkrétny typ bloku existujú nastaviteľné parametre.
- `defaultExtra` - v prípade hodnoty `true` parametra `hasExtra`, obsahuje tento parameter pole nastaviteľných parametrov, ktoré sa viažu na druhú, nepovinnú časť JSON objektu

Príklad povinnej časti JSON objektu je znázornený v nasledujúcej ukážke kódu:

```
"Multiply":
[
    {   "io":"both",
        "NumberOfInputs":1,
        "NumberOfOutputs":1,
```

```
            "MaxInputs":1,
            "ports":null,
            "NumOfTop":0,
            "NumOfBot":0,
            "hasExtra":true,
            "defaultExtra":{0:"F",1:"1"},
            "BlockType":"Multiply",
            "specific":true
        }
    ]
```

Druhá, nepovinná časť objektu obsahuje informácie o HTML elementoch, ktoré sa v aplikácii zobrazujú v dialógových oknách. Pomocou týchto HTML elementov je možné zmeniť nastaviteľné parametre ako sú meno bloku, počet vstupných portov a parametre obsiahnuté v poli `defaultExtra`. Využiteľné HTML elementy v tomto prípade sú textový `input`, `checkbox` a `select`. Všetky tri elementy zdieľajú páry kľúč-hodnota:

- `title` - textový reťazec, ktorý funguje ako popis daného elementu a vypisuje sa pri ňom v modálnom okne
- `id` - textový reťazec reprezentujúci HTML atribút `id`
- `number` - číselná hodnota korešpondujúca s pozíciou konkrétneho nastaviteľného parametru v poli `defaultExtra`

Príklad druhej, nepovinnej časti JSON objektu obsahujúcej jeden HTML element textového poľa:

```
{
    "type":"input",
    "data":
        {
            "title":"Numerator",
            "id":"multiply-citatel",
            "default_value":"F",
            "number":0
        }
}
```

4.1.3 Pridávanie blokov na plátno

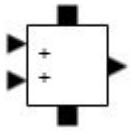
Na základe parametrov definovaných v spomenutých JSON objektoch je možné dané bloky vykresliť na plátno. V prvom rade sú tieto bloky dynamicky vypisované do hlavného menu aplikácie na ľavej strane. Toto je vykonávané vo funkcii `loadBlocks`, ktorá načítava jednotlivé časti blokov uchovávaných v objekte `blockDrawData` a vykresľuje ich na samostatné plátna v hlavnom menu. Na bloky v hlavnom menu je následne možné kliknúť, čím sa zavolá funkcia s názvom `addBlock`, ktorá má za úlohu konkrétny blok vykresliť na hlavné plátno, kde sa skladá bloková schéma. Táto funkcia pozostáva z viacerých častí, ktoré dajú dohromady blok a všetky jeho časti, potrebné pre následný správny chod aplikácie. Prvým krokom je vykreslenie samotného bloku na základe prislúchajúceho JSON objektu - blok je po častiach pridávaný na hlavné plátno a následne sú tieto časti spojené do skupiny, ktorá reprezentuje daný blok. V prípade častí ako sú `rect`, `circle` a `triangle` sú potrebné iba základné údaje ako výška, šírka a odsadenie častí, no pri častiach ako `text` a `path` sú potrebné špecifické parametre ako súradnice danej čiary a text na vypísanie.
Ďalším krokom kompletného vykreslenia bloku na plátno je pridanie jeho zodpovedajúcich portov na základe parametrov v JSON objekte `blockParameters`.

Obr. 20: Ukážka blokov s portami

Po určení toho, ktoré porty majú byť bloku priradené, sa tieto porty taktiež vykresľujú na hlavné plátno a ich pozícia je určovaná nasledovne:

- `Vstupný port`
  - `súradnica x` = pozícia ľavého horného rohu bloku
  - `súradnica y` = pozícia ľavého horného rohu bloku + výška bloku / 2
- `Výstupný port`
  - `súradnica x` = pozícia ľavého horného rohu bloku + šírka bloku
  - `súradnica y` = pozícia ľavého horného rohu bloku + výška bloku / 2

V prípade horného a dolného portu, ktoré sú znázornené na obrázku 21, sa výpočet pozície vykonáva podobným spôsobom.



Obr. 21: Ukážka bloku s portami, vrátane horného a spodného portu

Jedinou výnimkou výpočtu pozície portu sú bloky s viacerými vstupnými portami znázornené na obrázku 20. Pri týchto blokoch sa výpočet pozície vstupných blokov počíta dynamicky na základe daného počtu nasledovne:

$$x = výška\ bloku/počet\ vstupov - (výška\ bloku/počet\ vstupov/2)$$

$$pozícia = počiatočná\ pozícia + (poradie\ portu * x)$$

V tomto prípade `x` reprezentuje odsadenie jednotlivých portov od seba, `počiatočná pozícia` reprezentuje ľavý horný roh bloku, ku ktorému port patrí a `pozícia` je finálna pozícia portu podľa jeho poradia. Po vykreslení všetkých požadovaných portov na plátno zostáva posledná časť funkcie, ktorá sa stará o pridanie vytvoreného bloku a jeho vstupných portov do štruktúry `scheme`. Na základe tejto štruktúry sa vykonáva väčšina funkcionality vytvárania a spracovávania blokovej schémy, ale aj následné výstupy a exporty blokových schém.

4.2 Dátová štruktúra

Na ukladanie podstatných informácii o blokovej schéme a jednotlivých blokoch sa používa štruktúra s názvom `scheme`. Táto štruktúra, podobne ako dátové štruktúry jednotlivých blokov, pozostáva z párov kľúča a hodnôt, kde kľúč je spojenie typu objektu a poradia daného objektu na plátne. Hodnoty týchto párov závisia od druhov objektov, ktoré sa v štruktúre nachádzajú. Druhy objektov v štruktúre môžu byť:

- `Blok` - reprezentuje blok pridaný na plátno, jeho kľúč pozostáva z názvu bloku a čísla, napríklad `'blok1'`
- `Vstupný port` - vstupné porty patriace blokom, kľúče sú zapísané v tvare `'blok1I'` a v prípade, že vstupných portov je viac, sa za týmto kľúčom nachádza číslo daného portu

- `Čiara` - spojenie medzi jednotlivými blokmi, kľúče sú v tvare `'line1'`
- `Bod vetvenia` - bod vetvenia vytvorený na jednom zo spojení medzi blokmi, kľúč pozostáva z reťazca `'point'` a čísla čiary, na ktorej sa nachádza
- `Informácie o schéme` - uchováva názov schémy, na ktorej sa práve pracuje, napríklad `rlc`

Po procese vykreslenia bloku na plátno popísané v kapitole 4.1.3 sa parametre daného bloku zapíšu do tejto štruktúry. To, aké parametre sa do štruktúry zapíšu môže závisieť od konkrétneho typu blokovej schémy, no všeobecne bloky zdieľajú nasledovné parametre:

- Typ a názov bloku
- Celkový počet vstupných a výstupných portov
- Počet plných vstupných a výstupných portov
- Pole súradníc bloku
- Rotáciu bloku v stupňoch
- Číselnú hodnotu reprezentujúcu poradie bloku
- Názvy blokov a čiar pripojených na porty tohto bloku
- Parametre a názvy týchto parametrov špecifické pre tento blok

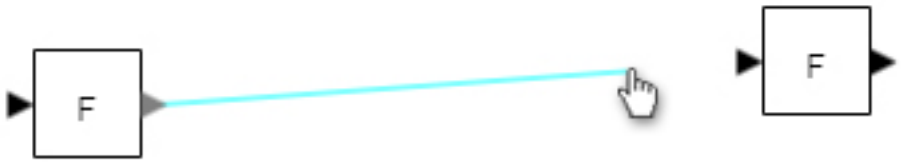
V prípade záznamu v štruktúre pre vstupné porty patriace blokom sa zapisujú parametre:

- Boolovská premenná popisujúca, či je na port pripojená čiara
- Názov pripojenej čiary
- Názov a poradie portu
- Názov rodičovského bloku

Parametre pre čiary a body vetvenia sú popísané neskôr v kapitole 4.3 a 4.4, v ktorých sa bližšie popisuje spôsob vytvárania čiar, spájania blokov a vytvárania bodov vetvenia na týchto čiarach.

4.3 Spájanie blokov

Spájanie blokov a vytváranie čiar, ktoré tieto bloky spájajú sa vykonáva na základe udalostí knižnice Fabric.js. V prípade umiestnenia myši na objekt reprezentujúci výstupný port bloku je možné zatlačiť ľavé tlačítko myši, kedy knižnica zaregistruje udalosť `mouse:down`. Na základe tejto skutočnosti sa zapíšu potrebné údaje o bloku a pozícii výstupného portu, a vytvorí sa počiatočný bod pomocnej čiary. Po získaní týchto údajov je možné myšou pohnúť a udalosť s názvom `mouse:move` zabezpečí neprestajné získavanie koncového bodu pomocnej čiary, ktorú aj vykreslí. Táto pomocná čiara je znázorňovaná ako neprerušovaná svetlomodrá čiara od stredu výstupného portu, po aktuálnu pozíciu myši na plátne.



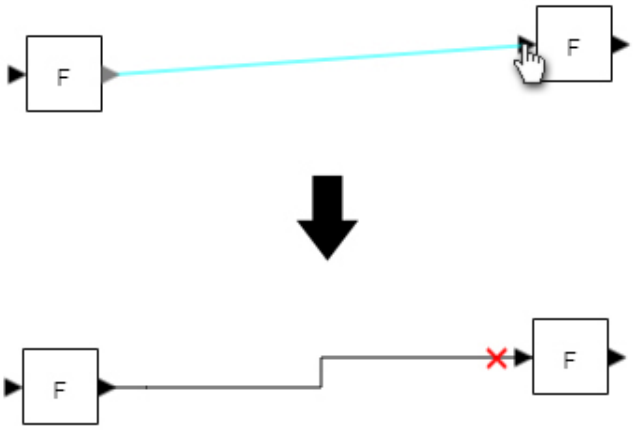
Obr. 22: Pomocná čiara pri spájaní blokov

Prostredníctvom tejto pomocnej čiary je možné zaznamenať konečný bod požadovanej čiary. Po vybratí koncového bodu a pustení ľavého tlačítka myši sa vyvolá udalosť `mouse:up` a v prípade tejto udalosti môžu nastať tri prípady:

- Koncový bod čiary sa nachádza v blízkosti vstupného portu bloku iného, ako je blok z ktorého je ťahaná pomocná čiara - v tomto prípade vykonávanie funkcie pokračuje
- Koncový bod čiary sa nenachádza v blízkosti vstupného portu - pomocná čiara sa vymaže
- Jeden z portov už je plný a nemôže presiahnuť maximálny počet čiar pripojený na neho - čiara sa zmaže, bloky sa nespoja a vypíše sa chybová hláška

V prípade, že koncový bod pomocnej čiary sa nachádza v blízkosti vstupného portu a kapacita portov nebola presiahnutá sa pomocná čiara zmaže a zavolá sa funkcia `createLine`. Táto funkcia zabezpečuje získanie správnej pozície vstupného a výstupného portu na základe pozície a rotácie ich rodičovského bloku a následne pomocou funkcie `createPath`

získa potrebné súradnice čiary. Ďalej sa táto čiara na základe vypočítaných súradníc vykreslí na plátno a spojí vybrané bloky.



Obr. 23: Vytvorenie spojenia medzi blokmi

Po spojení týchto blokov je potrebné upraviť ich údaje v dátovej štruktúre `scheme`. Obom blokom sa názov vytvorenej čiary zapíše medzi čiary napojené na blok a upravia sa kapacity napojených portov.
Vytvorené spojenie medzi blokmi a jeho parametre je následne potrebné zapísať do dátovej štruktúry pre ďalšie použitie. Pre objekt čiary sa do tejto štruktúry zapisujú:

- Číselná hodnota reprezentujúca poradie čiary
- Názov bloku, z ktorého vychádza a do ktorého vchádza
- Poradie vstupného portu, do ktorého vchádza
- Boolovnská hodnota popisujúca skutočnosť, či sa na čiare nachádza bod vetvenia
- Dodatočné informácie o type spojenia

Na konci vytvorenej čiary sa dodatočne vykresľuje tlačítko v tvare červeného `x`, ktoré je znázornené na obrázku 24 a slúži na vymazanie čiary. Po dokončení celého procesu vytvárania spojenia medzi blokmi a zápisu tohto spojenia do štruktúry sa pomocou udalosti `object:moving` sleduje, či sa blok a spolu s ním čiara hýbe. V prípade, že sa blok hýbe, sa volá funkcia s názvom `redrawLine`, ktorá na základe pozície bloku prekresľuje súradnice

čiary tak, aby sa počiatočný a koncový bod zachovali na správnom mieste zodpovedajúcich portov. Spolu s čiarou táto funkcia zabezpečuje prekresľovanie bodu vetvenia a bodu zmazania čiary.

4.4 Bod vetvenia

Pomocou dvojkliku ľavého tlačítka myši nad čiarou medzi dvoma blokmi a udalosti `mouse:dblclick` je možné na začiatku vybranej čiary vytvoriť bod vetvenia v prípade, že sa na tejto čiare ešte bod vetvenia nenachádza. Tento bod je v blokovej schéme zobrazený ako čierny, plný kruh a správaním je podobný výstupnému portu bloku.

Obr. 24: Bod vetvenia

Pri pridaní bodu vetvenia na plátno je potrebné tento bod zapísať do štruktúry `scheme` podobne ako pri objektoch bloku a čiary. V prípade tohto bodu sa do štruktúry zapisujú parametre:

- Názov čiary, na ktorej je bod umiestnený
- Názov bloku z ktorého rodičovská čiara vychádza
- Počet výstupov - podľa základných nastavení aplikácie je to vždy maximálne jeden výstup
- Počet plných výstupov
- Číselná hodnota reprezentujúca poradie bodu - je zhodná s poradím rodičovskej čiary
- Zoznam blokov, na ktoré je bod napojený

Použitie bodu vetvenia na spojenie bodu s blokom je zhodné s použitím výstupného portu bloku. Zatlačením ľavého tlačítka myši nad bodom vetvenia a následným pohnutím myši sa vytvorí pomocná čiara. Potiahnutím tejto čiary nad vstupný port nejakého z blokov a

následným pustením ľavého tlačítka myši sa v prípade voľných portov vytvorí spojenie, ktoré sa tiež zapíše do dátovej štruktúry. Ďalej sa čiara v prípade pohybu bodu, alebo koncového bloku prekresľuje podľa potreby.

4.5 Parametre blokov

Každý z blokov obsahuje parametre, ktoré sú uložené v dátovej štruktúre. Tieto bloky sú primárne využívané pri získavaní výstupov pre jednotlivé schémy a preto je pre správnu funkčnosť aplikácie potrebné tieto parametre upravovať. Väčšina blokov zdieľa dva hlavné parametre - názov bloku a v prípade potreby, počet vstupných portov. Zvyšné parametre sú definované v dátovej štruktúre a sú špecifické pre konkrétne typy schém a bloky. Všetky z týchto parametrov je možné nastaviť na jednom mieste, v dialógovom okne, ktoré je možné vyvolať pomocou dvojkliku ľavého tlačítka myši na daný blok. Zmena prvého z parametrov, mena bloku, spočíva v upravení záznamu v dátovej štruktúre a v prípade povolenia zmeny viditeľného mena na plátne pomocou hodnoty `changeable` sa zmení aj text viditeľný na obrázku 26.

Obr. 25: Zmena textu bloku

V prípade, že bloku je možné meniť počet vstupných portov, je v prvom rade potrebné blok odpojiť od všetkých čiar. Ak je blok odpojený od čiar a pole tohto parametru obsahuje akceptovanú hodnotu, upraví sa dátová štruktúra a prostredníctvom funkcie `redrawBlock` sa tento blok prekresľuje. Proces prekresľovania spočíva v uložení podstatných dát do dočasných premenných, zmazania pôvodného bloku z dátovej štruktúry a z plátna pomocou funkcie `deleteBlock`, a následného vykreslenia nového bloku na základe uložených dočasných premenných s novým počtom vstupných portov. Vykresľovanie požadovaného počtu portov funguje na princípe výpočtu pozície portu podľa celkového počtu portov a poradia daného portu, rovnako ako v kapitole 4.1.3.

Obr. 26: Zmena počtu vstupných portov

Zmena zvyšných parametrov špecifických pre konkrétne bloky spočíva primárne v úprave dátovej štruktúry, v zriedkavých prípadoch sa mení aj text bloku na plátne.

4.6 Ukladanie a načítanie schémy

Ak je v nejakom momente vytvárania schémy potrebné prerušiť danú činnosť, je možné schému v aktuálnom stave uložiť a neskôr načítať bez straty údajov. Uloženie vytvorenej blokovej schémy má na starosti funkcia `saveScheme`, ktorá po obdržaní požiadavky z používateľského prostredia načíta dátovú štruktúru `scheme` v celom rozsahu a prevedie ju do formátu JSON, ktorý uloží do `Blob` objektu. Následne v používateľskom prostredí poskytne možnosť pomenovania výstupného súboru, v základe pomenovaný `model`, a súbor ponúkne na stiahnutie na lokálny počítač.

V prípade požiadavky na načítanie schémy zo súboru sa aplikácia pomocou funkcie `loadModel` v prvom rade uistí, že používateľ chce premazať aktuálnu schému na plátne. Následne načíta používateľom vybraný súbor obsahujúci schému a overí jeho správnosť. Obsah súboru ďalej načíta a prevedie ho do potrebného tvaru, prostredníctvom ktorého najprv overí, či typ schémy korešponduje s typom schémy nastavenom v používateľskom prostredí, a následne schému začne vykresľovať na plátno. Po procese vykreslenia sa schéma na plátne a v dátovej štruktúre nachádza v stave identickom schéme pred jej uložením.

4.7 Riešenie schémy RLC obvodu

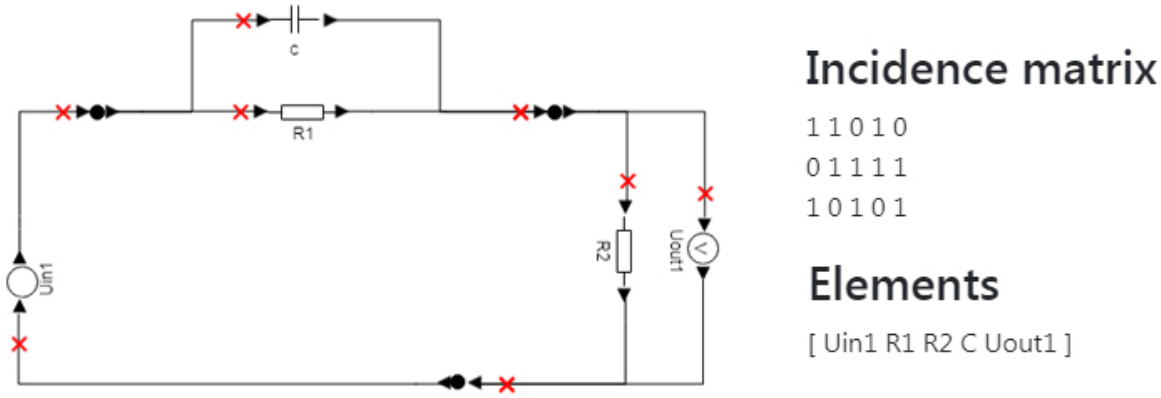
Riešenie schém RLC obvodov a vytvorenie výstupu spočíva vo vytvorení matice incidencie na základe grafu prevedeného z blokovej schémy vytvorenej používateľom spolu s vektorom prvkov, z ktorých sa táto bloková schéma skladá.

V prvom kroku riešenia tohto obvodu je podstatné rozdeliť objekty pridané na plátno na uzly a bloky, z ktorých neskôr budú hrany grafu. Toto je zabezpečené pomocou dvoch polí s názvami `nodes` a `branches`, do ktorých sa postupne pridávajú všetky prvky na plátne na základe ich typu. Zároveň s vytváraním polí obsahujúcich uzly a hrany sa vytvára vektor všetkých blokov, ktorý je taktiež obsiahnutý v poli s názvom `elementVector`, do ktorého sa zapisujú mená blokov na plátne.

Po naplnení zoznamov uzlov a blokov vytvorenej blokovej schémy sa vytvára pole núl o veľkosti *Počet uzlov* $\times$ *Počet blokov*, ktoré slúži na uskladnenie výsledkov matice incidencie. Výpočet výsledkov matice incidencie spočíva v postupnom prechádzaní zoznamu uzlov, pričom sa v prvom rade zaznačí poradie uzla, ktorý je aktuálne skúmaný. Ďalším krokom je nahliadnutie do dátovej štruktúry a pomocou parametra `connectedToBlocks`

je potrebné určiť, s ktorými blokmi je tento uzol spojený pomocou jeho výstupného portu. Následne sa prechádza zoznam hrán, a v prípade zhody názvu niektorej z hrán s blokom spojeným so skúmaným uzlom, sa zaznačí číslo tejto hrany. Posledným krokom tohto výpočtu je zapísanie hodnoty `1`, ktorá reprezentuje spojenie medzi skúmaným uzlom a nájdenou hranou, na správne miesto poľa matice incidencie. Toto je zabezpečené pomocou čísel daného uzla a hrany, ktoré boli zaznačené počas výpočtu a teda číslo uzla reprezentuje riadok a číslo hrany reprezentuje stĺpec v matici incidencie. Po zapísaní všetkých hrán spojených s výstupom uzla sa vykoná rovnaký postup s rozdielom parametra `connectedFromBlock`, ktorý určuje hrany s ktorými je uzol spojený pomocou jeho vstupného portu. Celý tento proces je znázornený v algoritme 1.

Po dokončení procesu výpočtu matice incidencie nasleduje vypísanie požadovaného výstupu na používateľské rozhranie pomocou dialógového okna. Toto dialógové okno sa najprv naplní jednotlivými hodnotami výstupnej matice incidencie a vektora elementov, ktoré sú formátované pomocou HTML kódu a následne sa používateľovi zobrazia. Ukážka vytvorenej RLC blokovej schémy a jej výstupu z príkladu na obrázku 13 je zobrazená na obrázku 27.



Obr. 27: Ukážka riešenia RLC blokovej schémy

Algorithm 1 Výpočet matice incidencie

```
Naplň pole uzlov
Naplň pole hrán
if počet uzlov > 0 AND počet hrán > 0 then
  Vytvor pole núl matica_incidencie
  for all uzly do
    i = číslo uzla
    for all bloky napojené z výstupného portu uzla do
      Zapíš názov bloku
      for all hrany do
        if názov napojeného bloku = názov hrany then
          j = číslo hrany
          matica_incidencie_ij = 1
        end if
      end for
    end for
    for all bloky napojené z vstupného portu uzla do
      Zapíš názov bloku
      for all hrany do
        if názov napojeného bloku = názov hrany then
          j = číslo hrany
          matica_incidencie_ij = 1
        end if
      end for
    end for
  end for
end if
```

4.8 Riešenie blokovej algebry a zjednodušovania blokových schém

Pri tomto type blokovej schémy je zámerom zjednodušenie vytvoreného systému a získanie celkovej prenosovej funkcie systému. Na vytvorenie systému sú v tomto prípade dostupné štyri typy blokov - blok prenosovej funkcie, sčítavací blok, vstupný a výstupný blok. Každý z týchto blokov má špecifické funkcie v systéme a nastaviteľné parametre, bližšie popísané v tabuľke 1.

Tabuľka 1: Popis blokov riešenia blokovej algebry

Názov bloku	**Názov v aplikácii**	**Špecifické parametre**
Blok prenosovej funkcie	`Multiply`	Čitateľ a menovateľprenosovej funkcie
Sčítavací blok	`Sumator`	Znamienka vstupov bloku
Vstupný blok	`IOin`	-
Výstupný blok	`IOout`	-

Vstupné a výstupné bloky v tejto blokovej schéme zabezpečujú ekvivalentné signály pre systém. V oboch prípadoch majú bloky jeden port, ktorý určuje ich funkciu. Vzhľadom na to, že ide o schému typu SISO (Single Input and Single Output), každá správne zapojená schéma tohto typu musí obsahovať presne jeden vstupný a jeden výstupný blok, ktoré slúžia ako vstup a výstup systému.

V prípade bloku prenosovej funkcie ide o blok s jedným vstupom a jedným výstupom, ktorého funkcionalita závisí od jeho parametrov - čitateľa a menovateľa funkcie. Ak čitateľ obsahuje premennú alebo konštantu, jedná sa o funkciu zosilnenia a táto hodnota sa taktiež zobrazuje v bloku na plátne. Tomuto bloku je však možné nastaviť aj prenosovú funkciu, ktorá pozostáva z dvoch polynómov. Toto je možné prostredníctvom zápisu čitateľa a menovateľa pomocou hodnôt jednotlivých koeficientov polynómu oddelených medzerami nasledovne:

$$2\ 1\ 1\ 0 \Rightarrow 2s^3 + s^2 + s$$

$$1\ 1\ 1 \Rightarrow s^2 + s + 1$$

V prípade bloku s parametrami prenosovej funkcie sa v bloku na plátne zobrazí text `f(s)`, ktorý informuje používateľa o tom, že blok reprezentuje prenosovú funkciu. Zápis tejto prenosovej funkcie je možné skontrolovať v dialógovom okne daného bloku, alebo pomocou ukážky prenosovej funkcie na spodku aplikácie pri nabehnutí myši na daný blok.

Posledným z blokov dostupných v tomto type schémy je sčítavací blok, ktorého úlohou je spojiť dva signály a vyprodukovať z nich jeden signál na jeho výstupe. Vytváranie tohto signálu je zabezpečené pomocou dvoch vstupných portov, ktorých znamienka ovplyvňujú parametre bloku. Nastavením parametrov tohto bloku je možné docieliť akúkoľvek kombináciu znamienok "+" a "-". Sčítavací blok taktiež obsahuje jeden výstupný port, pomocou ktorého odovzdáva vyprodukovaný signál.

Po vytvorení kompletnej schémy pomocou vyššie spomenutých blokov, spojov medzi jednotlivými blokmi a bodov vetvenia je možné v používateľskom rozhraní požiadať o výstup daného systému. V tomto prípade sa vytvorená schéma musí upraviť do správneho formátu zodpovedajúceho externej aplikácii [1] používanej na získanie výstupu, ktorým je JSON objekt obsahujúci informácie o schéme, plátne a jednotlivých častiach schémy. Prvou časťou tohto JSON objektu sú informácie o blokovej schéme a plátne ako napríklad názov schémy, počet blokov na plátne a výška a šírka plátna:

```
"block_diagram":{
    "name":"diagram",
    "counter":3,
    "canvas":{
        "width":1271,
        "height":989
    },
"blocks":{ ... }
```

Ďalšou časťou tohto objektu je zoznam všetkých blokov na vytvorenej blokovej schéme, spojení medzi týmito blokmi a bodov vetvenia. Medzi hlavné informácie o jednotlivých blokoch patria názov a typ bloku, ich pozícia na plátne, počet voľných portov a ich špecifické parametre. Informácie o blokoch taktiež obsahujú zoznam ich portov, ktoré definujú ich typ a názov bloku, s ktorým sú spojené. Rovnakým spôsobom sú štrukturované aj záznamy o spojeniach medzi blokmi a bodoch vetvenia. Príklad záznamu bloku v JSON objekte:

```
"Multiply1":{
    "block_type":"multiply",
    "position_x":379,
    "position_y":214,
    "numerator":"F",
    "denominator":"1",
```

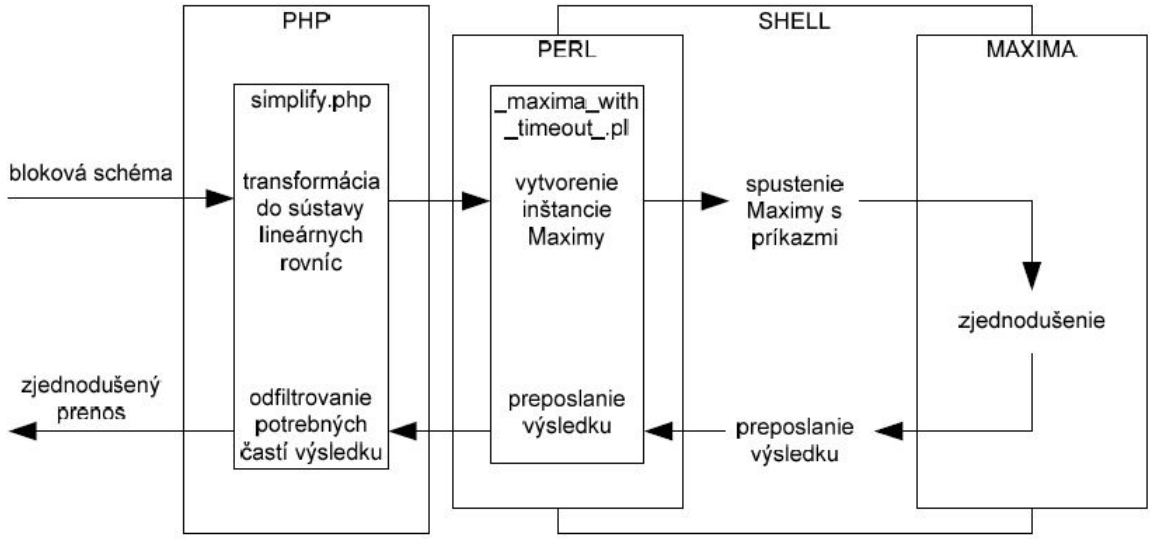
```
            "empty_pin_count":0,
            "pins":{
                "pin1":{
                    "type":"output",
                    "location":"right",
                    "connected_to_block":"line2",
                    "connected_to_pin":"input"
                },
                "pin2":{
                    "type":"input",
                    "location":"left",
                    "connected_to_block":"line1",
                    "connected_to_pin":"output"
                }
            }
        }
```

Po zostrojení celej štruktúry JSON objektu používateľom vytvorenej blokovej schémy je možné tento objekt odoslať na spracovanie a zjednodušenie serverovej časti externej aplikácie pomocou AJAX volania. Toto volanie zabezpečí odoslanie požiadavky na spracovanie upravenej blokovej schémy a je vykonávané pomocou jednoduchého skriptu:

```
.ajax({
    type:   "určenie HTTP metódy požiadavky, v našom prípade POST",
    url:    "URL smerujúce na serverový skript obstarávajúci našu
             požiadavku",
    data:   "určuje dáta, ktoré sa majú odoslať skriptu",
    async:  "boolovská premenná definujúca, či sa má požiadavka
             vykonať asynchrónne",
    success:"určuje čo sa má vykonať v prípade úspešného
             prijatia odpovede"
});
```

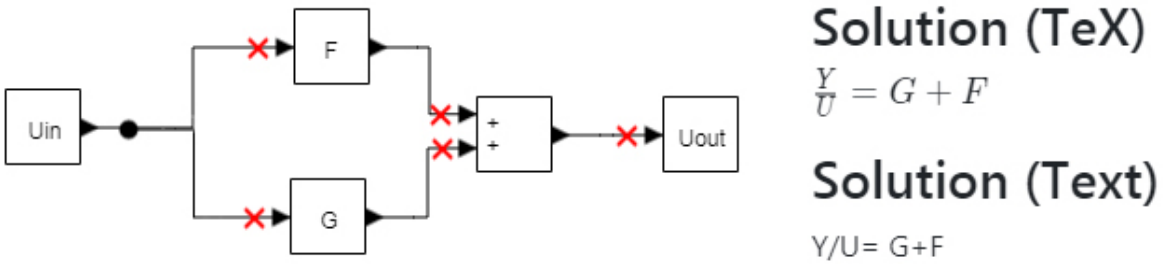
Odoslaním požiadavky na skript `simplify.php` externej aplikácie sa v prvom rade vyhodnotí, či je vytvorená schéma správne zapojená. V prípade, že obsahuje práve jeden vstupný a jeden výstupný blok, a všetky porty blokov sú naplnené, pokračuje sa v zjednodušovaní schémy vytvorením lineárnych rovníc a generovaní kódu pre program Maxima, ktorým

skript získa výslednú prenosovú funkciu. Táto prenosová funkcia je následne preposlaná na klientsku časť našej aplikácie a vypísaná do dialógového okna na používateľskom rozhraní pomocou knižnice KaTeX. Proces zjednodušenia blokovej schémy a vyhodnotenia prenosovej funkcie je znázornený na obrázku 28.



Obr. 28: Proces vyhodnotenia výslednej prenosovej funkcie systému [1]

Riešenie jednoduchého obvodu tohto typu schémy, ktorý obsahuje všetky dostupné druhy blokov a bod vetvenia je zobrazené na obrázku 29.



Obr. 29: Príklad vyhodnotenia výslednej prenosovej funkcie systému

4.9 Zostavovanie blokových schém pre simuláciu

Cieľom tohto typu schémy je vytvorenie blokovej schémy, ktorú je možné následne využiť na ďalšiu simuláciu pomocou prostredia Simulink [17]. Blokové schémy vytvárané v našej aplikácii zodpovedajú schémam v prostredí Simulink verzie 7.0.

Zostavenie schémy je možné pomocou 25 často používaných blokov z prostredia Simulink a nastavenia parametrov jednotlivých blokov. Zoznam všetkých dostupných blokov v tomto type schémy spolu s počtom ich vstupných a výstupných portov, a zoznamom

ich nastaviteľných parametrov je v tabuľke 2. Výstupom tohto druhu schémy je textový súbor vo formáte `.mdl` obsahujúci podstatné informácie o schéme, plátne, jednotlivých blokoch a ich parametroch, spojeniach medzi blokmi a bodoch vetvenia. Prostredníctvom tohto súboru je možné schému vytvorenú v našej aplikácii načítať a využívať na lokálnu simuláciu, alebo simuláciu na internete.

Vytvorenie tohto výstupu spočíva v transformovaní blokovej schémy vytvorenej používateľom do formy, ktorú je externá aplikácia [2] schopná spracovať. Preto je potrebné, podobne ako pri predchádzajúcom type blokovej schémy vytvoriť JSON objekt, obsahujúci všetky podstatné informácie na vytvorenie výstupu. Tento objekt vyzerá nasledovne:

```
{
    "modelName": "NewScheme",
    "systemName": "Simulink",
    "version": "7.0",
    "width": "1271",
    "height": "989",
    "blocks": [ ... ],
    "connections": [ ... ]
}
```

Medzi hlavné údaje patrí názov schémy, verzia aplikácie Simulink, pre ktorú je daná schéma vytváraná, výška a šírka plátna a zoznamy jednotlivých blokov a spojení medzi nimi. Zoznamy blokov a spojení obsahujú všetky prvky pridané používateľom na plátno spolu s dodatočnými informáciami o nich. Medzi tieto informácie v prípade blokov patrí názov a typ bloku, ich aktuálna pozícia na plátne, ktorá je využívaná pri vykresľovaní daného bloku v prostredí Simulink, alebo inej aplikácie na simuláciu. Príklad záznamu jedného z blokov v tomto objekte vyzerá nasledovne:

```
{
    "id": "Clock1",
    "name": "Clock",
    "type": "Clock",
    "x": "247",
    "y": "354",
    "attributes": {
        "Decimation": "10"
}}
```

Okrem vyššie spomenutých informácii tieto záznamy obsahujú zoznam parametrov špecifických pre jednotlivé typy blokov, ktoré je možné nastaviť pomocou dialógového okna v používateľskom rozhraní a sú obsiahnuté v zozname s názvom `attributes`.

V prípade spojení medzi blokmi obsiahnutými v zozname s názvom `connections` sa zaznamenávajú porty a názvy blokov, z ktorých dané spojenie vychádza, a do ktorých dané spojenie vchádza. Záznam spojenia medzi blokmi v tejto štruktúre môže vyzerať napríklad takto:

```
{
    "id": "line2",
    "startBlockId": "Clock1",
    "startPoint": "E",
    "endBlockId": "AnalogOutput1",
    "endPoint": "W"
}
```

Po zostavení požadovanej blokovej schémy a transformovaní tejto schémy na JSON objekt akceptovaný externou aplikáciou je možné požiadať o jej prevedenie na výstupný súbor formátu `.mdl`. Toto je zabezpečené, podobne ako pri predchádzajúcom type blokovej schémy, pomocou AJAX volania s požiadavkou na skript s názvom `export.php`. Prenášanými dátami požiadavky je samozrejme upravená bloková schéma vo formáte JSON. Úlohou spomenutého skriptu je danú schému dekódovať a rozložiť na elementárne časti, pomocou ktorých neskôr zloží požadovaný výstupný súbor. Na základe blokov použitých vo vytvorenej blokovej schéme vytvorí zoznam typov týchto blokov a do výstupného súboru zapíše predvolené parametre týchto typov blokov. Rovnako vypíše aj konkrétne bloky použité v schéme a jednotlivé spojenia medzi nimi. Okrem týchto údajov sa vo výstupom súbore nachádza hlavička, ktorá obsahuje všetky potrebné údaje na využitie tohto súboru v aplikácii Simulink.

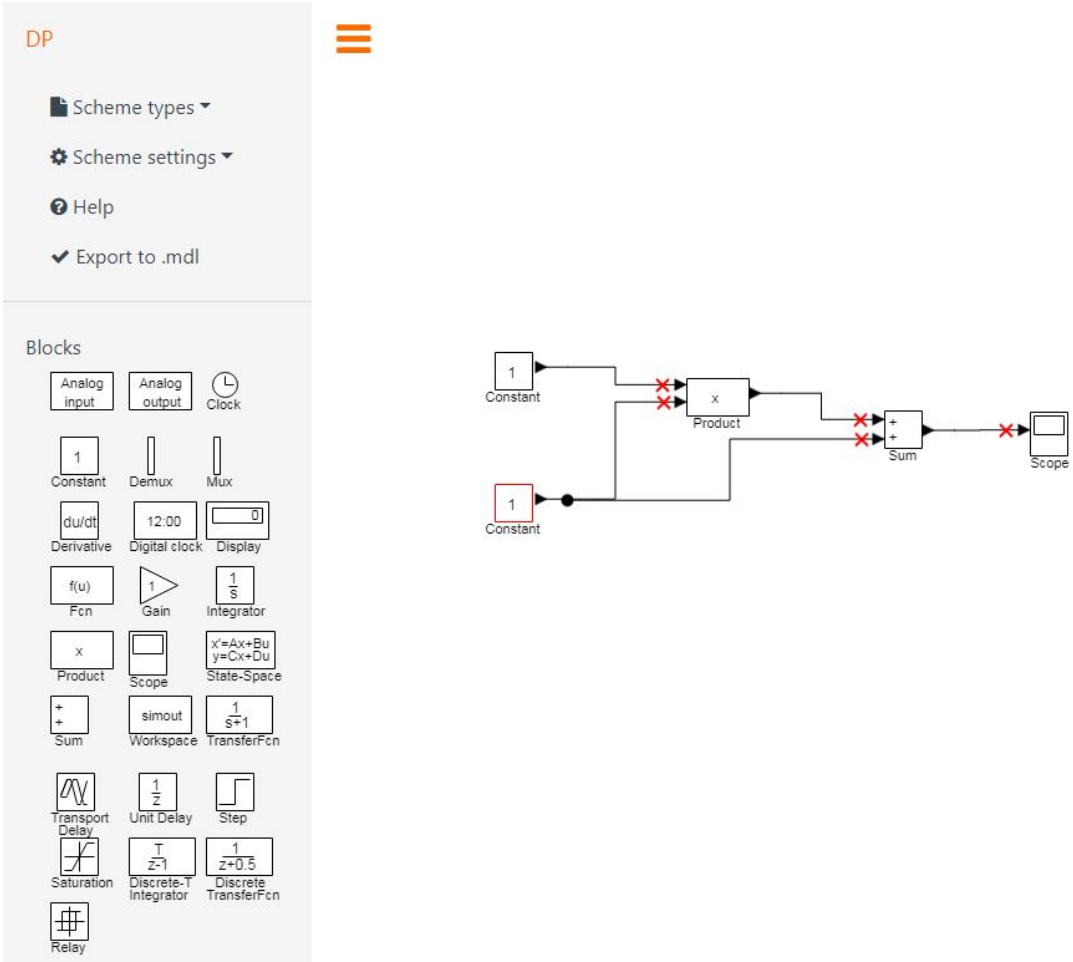
Tabuľka 2: Dostupné bloky schémy pre simulovanie

Názov bloku	**Vstupné porty**	**Výstupné porty**	**Nastaviteľné parametre**
Analog Input	-	1	Sample time, Input channels, Input range
Analog Output	1	-	Sample time, Output channels/range, Initial/Final Value
Clock	-	1	Decimation
Constant	-	1	Constant value
Mux	1-4	-	-
Demux	-	1-4	-
Derivative	1	1	Coefficient
Digital clock	-	1	Sample time
Display	1	-	Decimation
Function	1	1	Expression
Gain	1	1	Gain
Integrator	1	1	Initial condition, Limit output, Upper/Lower saturation limit
Product	1-4	1	-
Scope	1	-	-
StateSpace	1	1	A, B, C, D, Initial conditions
Sum	2	1	List of signs
To Workspace	1	-	Variable name, Limit data, Decimation, Output range
Transfer function	1	1	Numerator, Denominator
Transport delay	1	1	Time delay, Initial output
Unit delay	1	1	Delay length, Initial condition
Step	-	1	Step time, Initial/Final value
Saturation	1	1	Upper/Lower limit
Relay	1	1	Output on/off, Switch on/off point
Discrete-time integrator	1	1	Gain, Condition, Sample time, Limit output, Upper/Lower saturation limit
Discrete transfer function	1	1	Numerator, Denominator, Initial states

5 Používateľské rozhranie a používanie aplikácie

Cieľom tejto kapitoly je opísať používateľské rozhranie aplikácie, jeho jednotlivé prvky a používanie týchto prvkov. Používateľ má pomocou tohto rozhrania a jeho prvkov možnosť vytvárať grafickú reprezentáciu blokových schém a vykonávať operácie nad vytvorenou schémou, alebo jej časťami.

Toto rozhranie bolo navrhnuté a vytvorené tak, aby bolo intuitívne a jednoduché na používanie, no vyžaduje znalosť využívania klávesových skratiek, pomocou ktorých sa ovládajú niektoré operácie v blokových schémach.



Obr. 30: Používateľské rozhranie aplikácie

Používateľské rozhranie zobrazené na obrázku 30 sa skladá z troch hlavných častí - hlavnej ponuky, pomocou ktorej sa ovláda väčšina operácii nad schémou, ponuka dostupných blokov danej schémy a samotné plátno, na ktorom sa schéma skladá. Hlavné menu je

možné v prípade potreby schovať pomocou oranžovej ikony v ľavom hornom rohu plátna, čím sa rozšíri pracovná plocha. Opakovaným kliknutím na ikonu menu sa hlavné menu zobrazí. Veľkosť plátna sa dynamicky upravuje na základe veľkosti okna internetového prehliadača nastavením jeho výšky a šírky pomocou JavaScript príkazu `onresize`, ktorý sleduje udalosť zmeny veľkosti prehliadača.

Hlavná ponuka aplikácie zameraná na ovládanie vytváranej schémy na plátne je umiestnená v ľavom hornom rohu aplikácie a delí sa na následné položky:

- Typy schém
- Nastavenia schémy
- Nápoveda
- Získanie výstupu vytvorenej schémy

Prvou položkou tohto menu je výber dostupných typov schém, ktorých odkazy presmerujú používateľa na HTML súbor obsahujúci danú schému. V čase písania tejto práce naša aplikácia ponúka tri druhy blokových schém:

- RLC
- Bloková algebra a zjednodušovanie blokových schém
- Bloková schéma určenú na simulácie

Hlavné operácie nad schémou je možné zvoliť v druhej položke hlavnej ponuky, ktorá obsahuje tri možnosti:

- **Nová schéma** - prečistí pracovnú plochu a dátovú štruktúru a zabezpečí pôvodný stav schémy
- **Uložiť schému** - poskytne používateľovi aktuálny stav schémy na stiahnutie v súbore formátu, ktorý je popísaný v kapitole 4.6
- **Načítať schému** - umožní načítanie schémy zo súboru, v prípade načítania najprv uvedie plátno do pôvodného stavu

Položka nápoveda otvorí dialógové okno, ktoré približuje používateľovi spôsob práce s aplikáciou a klávesové skratky použiteľné v tejto aplikácii spolu s vysvetlením ich funkcionality. Poslednou zo štyroch položiek hlavnej ponuky je získanie výstupu vytvorenej

blokovej schémy, ktorá taktiež otvorí dialógové okno, ktorého obsah závisí od typu blokovej schémy s ktorou používateľ aktuálne pracuje.

V časti používateľského rozhrania pod hlavnou ponukou sa nachádza zoznam blokov dostupných pre zvolený typ schémy. Pomocou týchto blokov je možné zložiť používateľom požadovanú schému a pridávanie týchto blokov je zabezpečené pomocou kliknutia ľavého tlačítka myši na jeden z blokov v zozname. Po kliknutí na jeden z blokov v zozname sa daný blok vykreslí na plátne a je možné s ním ďalej pracovať. Takýmto spôsobom je na plátne možné vytvoriť neobmedzený počet blokov. Výpis celého zoznamu dostupných blokov je zabezpečený prostredníctvom dynamického pridávania všetkých blokov obsiahnutých v dátových štruktúrach pre vykresľovanie blokov a ich parametrov, ktoré boli popísané v kapitolách 4.1.1 a 4.1.2.

Poslednou z troch častí používateľského rozhrania je samotná pracovná plocha obsahujúca plátno, na ktoré je možné pridávať vyššie spomenuté bloky a pomocou nich vytvoriť požadovanú schému. Na plátne je možné vybrať jeden z vytvorených blokov pomocou ľavého tlačítka myši. Toto sa prejaví zmenou farby vonkajších okrajov vybraného bloku z čiernej na červenú, ako je zobrazené na obrázku 31, a pri zmene zvoleného bloku alebo zrušení voľby, sa farba pôvodne zvoleného bloku zmení späť na čiernu.

Obr. 31: Zmena farby pri zvolení aktívneho bloku

So zvoleným blokom je možné vykonávať nasledujúce operácie:

- Presúvať blok po plátne
- Zmazať blok
- Meniť rotáciu bloku
- Otvoriť dialógové okno obsahujúce parametre bloku

5.1 Klávesové skratky

Na zjednodušenie a urýchlenie práce s vytváranou schémou a jednotlivými blokmi schémy je možné využiť dostupné klávesové skratky. Klávesové skratky v našej aplikácii sa snažia držať štandardu klávesových skratiek používaných v bežných aplikáciách:

- **Del** - zabezpečuje vymazanie aktuálne zvoleného bloku, v prípade že na blok sú napojené nejaké čiary, zmažú sa aj tie
- **r** - mení rotáciu bloku v krokoch 90 stupňov. Zmena rotácie je zabezpečená prostredníctvom prekresľovania daného bloku a jeho portov pomocou údaju `rotation` v dátovej štruktúre. Zmena rotácie bloku je možná iba v prípade, že na blok nie sú napojené žiadne čiary
- **Ctrl+s** - uloženie stavu vytváranej blokovej schémy do súboru
- **Ctrl+j** - vytvorenie nového bloku podľa typu aktuálne zvoleného bloku
- **Smerové šípky** - posúvanie bloku o jednu pozíciu v smere zvolenej šípky

5.2 Dialógové okná aplikácie

Dialógové okná sú podstatnou časťou aplikácie a zabezpečujú štyri základné funkcie:

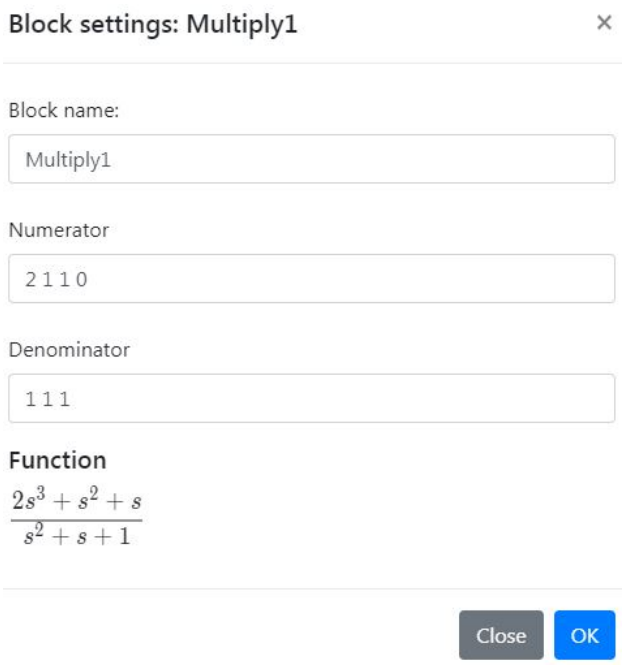
- Výpis nápovedy a dostupných klávesových skratiek v aplikácii
- Načítanie schémy so súboru
- Nastavovanie parametrov jednotlivých blokov
- Vypisovanie výstupu vytvorenej schémy

Prvým z týchto dialógových okien je okno nápovedy. Ako už bolo spomenuté, účelom tohto okna je priblíženie funkcionality aplikácie a klávesových skratiek používateľovi.

Dialógové okno načítania blokovej schémy sa zobrazí po kliknutí na odkaz načítania schémy v hlavnej ponuke a obsahuje pole, v ktorom sa dá zvoliť požadovaný súbor, ktorý chce používateľ načítať. Po potvrdení výberu sa aplikácia používateľa v prvom rade spýta, či chce daným súborom premazať aktuálny stav pracovnej plochy. Ak je načítaný vhodný formát schémy, táto schéma je následne vykreslená na plátno.

Parametre jednotlivých blokov je možné nastaviť pomocou dialógového okna, ktoré sa otvorí po dvojkliku ľavého tlačítka myši na konkrétny blok. Obsah tohto dialógového okna závisí od typu schémy s ktorou používateľ pracuje a vybraného bloku, no základnými údajmi tohto okna je meno bloku, a v prípade niektorých blokov zmena počtu vstupných portov. Tieto dialógové okná môžu obsahovať tri typy HTML vstupov - textový `input`, `select` a `checkbox`. V prípade parametrov bloku na obrázku 32 je možné nastaviť čitateľ a menovateľ funkcie bloku. Polynómy tejto funkcie sa následne vypíšu na mieste určenom pre túto funkciu pomocou knižnice KaTeX ako bolo popísané v kapitole 4.8. Po potvrdení

zmeny obsahu textových polí zodpovedajúcich jednotlivým parametrov sa tieto parametre odošlú na kontrolu správnosti. V prípade správnosti vstupov sa tieto hodnoty zapíšu do dátovej štruktúry a dialógové okno sa zavrie. Ak však obsah textových polí nezodpovedá potrebnému formátu, vypíše sa chybová hláška, ktorá používateľovi ohlási textové pole obsahujúce chybu a dialógové okno sa nezavrie.



Obr. 32: Dialógové okno parametrov bloku

Po vytvorení požadovanej blokovej schémy môže používateľ požiadať o výstup daného typu schémy pomocou hlavnej ponuky. Po tejto požiadavke sa vhodným spôsobom vyhodnotí schéma, vytvorí sa výstup vo formáte daného typu schémy a tento výstup sa vypíše do dialógového okna. Jednotlivé formáty výstupov sú popísané v kapitolách 4.7, 4.8 a 4.9, ktoré zodpovedajú jednotlivým typom schém. S dialógovým oknom obsahujúcim výstup schémy je možné hýbať po pracovnej ploche tak, aby bolo možné porovnať vstupnú schému a daný výstup. Po zavretí tohto okna je ho možné znova zobraziť po kliknutí na zodpovedajúce tlačítko v hlavnej ponuke.

Záver

Záver

Zoznam použitej literatúry

1. VÖRÖS, Ludovít. *Internetom podporované riešenie blokových schém*. Bratislava : FEI STU, 2011.
2. JANÍK, Zoltán. *Internetom podporovaná modifikácia blokových schém v Matlab/Simulinku*. Bratislava : FEI STU, 2010.
3. ARONOVÁ, Mária. *Online modelovanie lineárnych dynamických systémov*. Bratislava : FEI STU, 2015.
4. DORF, Richard C a BISHOP, Robert H. *Modern control systems*. Pearson, 2011.
5. MAHADEVAN, K. a CHITRA, C. *ELECTRICAL CIRCUIT ANALYSIS:* 2018. ISBN 9789387472341.
6. ROBBINS, Allan H a MILLER, Wilhelm C. *Circuit analysis: Theory and practice*. Cengage Learning, 2012.
7. VASS, Gábor. *Algoritmizovanie vytvárania modelov dynamických systémov*. Bratislava : FEI STU, 2015.
8. HALÁS, Miroslav. *Ročníkový projekt - Spracovanie témy „Modelovanie RLC obvodov“ pomocou programového systému MAPLE V*. Bratislava : FEI STU, 2000.
9. FULTON, Steve a FULTON, Jeff. *HTML Canvas, Second Edition*. 2. vyd. USA: O'Reilly Media, Inc., 2013. ISBN 978-1-449-33498-7.
10. GOODMAN, Danny. *Dynamic HTML: The Definitive Reference: A Comprehensive Resource for HTML, CSS, DOM & JavaScript*. Ö'Reilly Media, Inc.", 2002.
11. FLANAGAN, David. *JavaScript: The Definitive Guide*. 5. vyd. USA: O'Reilly Media, Inc., 2006. ISBN 978-0-596-10199-2.
12. OSMANI, Addy. *Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide*. Ö'Reilly Media, Inc.", 2012.
13. MAHEMOFF, Michael. *AJAX design patterns: creating Web 2.0 sites with programming and usability patterns*. Ö'Reilly Media, Inc.", 2006.
14. CROCKFORD, Douglas. The application/json media type for javascript object notation (json). 2006.
15. ZAYTSEV, Juriy, KIENZLE, Stefan a BOGAZZI, Andrea. *Dokumentácia ku knižnici Fabric.js* [online] [cit. 2018-01-02]. Dostupné z: `http://fabricjs.com/docs/`.

16. *Dokumentácia ku knižnici KaTeX* [online] [cit. 2018-04-04]. Dostupné z: `https://khan.github.io/KaTeX/`.

17. DABNEY, James B a HARMAN, Thomas L. *Mastering simulink*. Pearson, 2004.

Prílohy

A Štruktúra elektronického nosiča

```
/Diplomova_praca.pdf
/Pouzivatelska_prirucka.txt
/implementacia
    /css
        /style.css
    /js
        /blocks.js
        /controller.js
        /output.js
        /scheme.js
    /vendor
        /bootstrap
        /fabricjs
        /font-awesome
        /jquery
        /jquery-ui
        /katex
    /index.html
    /algebra.html
    /simulation.html
```