Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Vývoj uživatelského rozhraní v HTML5 pro různé struktury řízení klimatizačních jednotek a vytápění

PLZEŇ, 2013 Vojtěch Bohman

Zde přilož originální zadaní

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejíchž úplný seznam je její součástí.

V Plzni dne 19. srpna 2013

.....

VOJTĚCH BOHMAN

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Pavlu Baldovi, Ph.D. za cenné profesionální rady a připomínky. Dále pak všem pracovníkům katedry kybernetiky, kteří se podíleli na projektu, za poskytnuté rady a materiály.

Abstrakt

Práce se zabývá vývojem uživatelského rozhraní pro řízení klimatizačních a otopných jednotek za použití internetové technologie HTML5. V první části práce jsou prezentovány technologie použité při vývoji rozhraní. Mezi ně patří budoucí standard HTML5, škálovatelná vektorová grafika SVG, programovací jazyk JavaScript a další. Hlavním úkolem práce je za pomoci představených technologií vytvořit komponenty, ze kterých lze sestavit schémata pro zobrazení klimatizačních a otopných systémů. Práce popisuje vývoj komponent, jejich strukturu, vlastnosti a použití. Pro praktickou ukázku byla vytvořena schémata, která simulují běh klimatizační soustavy. V závěrečné části práce byl vytvořen tutoriál, který na konkrétních případech popisuje, jak se ovládají komponenty a tvoří vizualizační schémata.

The work deals with the development of the user interface for the steering of air conditioning and heating units use of HTML5 web technologies. The first part describes technologies used in developing of the interface. These include future standard HTML5, Scalable Vector Graphics SVG, JavaScript programming language and more. The main goal of this work is to create components which were built with help of introduced techonologies and use them for building schemes which display air conditioning and heating systems. The aim of the work is to describe development, structure, properties and applications of the components and schemes that have been created for a practical demonstration. The final part of this work represents tutorial which describes in examples how the components are controlled and how the schemes are formed.

Key words: HTML5, SVG, SCADA, HMI, JavaScript, InkScape, REX, uživatelské rozhraní, klimatizační jednotky

Obsah

Úvod

Na základě projektu firmy Mikroklima vznikl požadavek na vývoj uživatelského rozhraní pro různé struktury klimatizačních jednotek a vytápění. Toto rozhraní má být vytvořeno za pomoci nejmodernějších technologií. Proto byl zvolen budoucí standard HTML5, spolu s programovacím jazykem JavaScript, škálovatelnou vektorovou grafikou SVG a komunikací probíhající prostřednictvím protokolu WebSocket.

Po analýze požadavku se došlo k závěru vytvořit knihovnu komponent, které by reprezentovaly jednotlivé klimatizační a otopné zařízení (např.: ventilátor, rekuperátor, klapka, kotel, radiátor, ...). Z těchto komponent by uživatel mohl následně složit schéma, sloužící pro komunikaci člověk-stroj (neboli HMI). K tvorbě a editaci schémat by měl být použit vektorový editor Inkscape, který lze využívat zdarma. Uživatel by měl být schopen přehledně ovládat systém a měl by být informován o důležitých událostech v celé klimatizační (respektive otopné) soustavě.

K pochopení použitých technologií bylo třeba nastudovat celou řadu specifikací (HTML5, SVG, XML), dostupných na internetových stránkách W3Schools. Internetové zdroje také poskytly mnoho užitečných tutoriálů, které usnadnily první kroky při vývoji komponent. Vzhledem k vysokému přínosu těchto tutoriálů, bylo další cílem vytvořit jeden v závěru práce. Účelem tutoriálu má být seznámení uživatele s nejnutnějšími programy a postupy, které budou třeba k tvorbě a zprovoznění vizualizačních schémat.

Zmíněné technologie byly použity z důvodu nových možností, které přináší. Při použití starších specifikací by bylo nutné využívat různé zásuvné moduly (neboli externí technologie). Kdežto HTML5 nativně podporuje JavaScript a SVG, dovolující vytvářet funkční grafické prvky. Veškeré tyto technologie jsou volně šiřitelné.

V úvodní části práce budou představeny použité technologie a příklady jejich užití. Najdeme zde informace o HTML5, JavaScriptu, knihovně jQuery, formátu zápisu JSON, značkovacímu jazyku XML a škálovatelné vektorové grafice SVG. Další zmínka bude o systémech SCADA, vektorovému editoru Inkscape a řídícím systému REX. V následujících kapitolách se seznámíme s vývojem komponent, jejich popisem a funkcemi. Podrobně je představena knihovna a schémata, která byla vytvořena pro praktickou ukázku. Závěr práce tvoří tutoriál, který usnadní tvorbu uživatelského rozhraní. V příloze je uvedena ukázka zdrojového kódu komponenty (ostatní se nacházejí na přiloženém CD-ROM), schéma v programu REX a často používané zkratky v editoru Inkscape.

1 Použité softwarové technologie

1.1 HyperText Markup Language

SGML[1] je standardní jazyk určený k formálnímu popisu dokumentů. Hypertextový[2] značkovací jazyk je příkladem jazyka definovaného v SGML. HMTL patří mezi značkovací jazyky, které se využívají k tvorbě webových stránek.

Tento jazyk je charakterizován množinou značek (tzv.*tags*) a jejich atributů definovaných pro danou verzi. Názvy jednotlivých značek se uzavírají mezi lomené závorky < a >. Tyto značky se obvykle používají jako párové (<**tagname**> *content*</**tagname**>). První značka se nazývá otevírací a druhá zavírací. Za HTML[2] element považujeme vše, co se nachází mezí otevírající a zavírající značkou (včetně).

Účelem webového prohlížeče je přečíst HTML dokumenty a zobrazit je jako webové stránky. Prohlížeč nezobrazuje HTML[4] značky, ale používá je k interpretaci obsahu stránky. Na příkladu 1.1 vidíme správnou deklaraci (verze HTML 4.01,1999) a strukturu HTML stránky. <!DOCTYPE> deklarace pomáhá prohlížeči správně zobrazit webové stránky a pro různé verze HTML jsou i různé typy deklarací.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
     <title>Titulek</title>
  </head>
  <body>
     <h1>Hlavní nadpis</h1>
     <p>Odstavec</p>
     <p>Další odstavec</p>
  </body>
</html>
```

Obr. 1.1: Struktura HMTL stránky

[1]SGML (Standard Generalized Markup Language) – univerzální značkovací jazyk
[2]Hypertext – odkaz

1.1.1 HyperText Markup Language 5

Od roku 1999 se web velice změnil, proto je třeba vyvinout nový standard, HTML5. Podle současného plánu by měla být konečná specifikace HTML 5.0 schválena do konce roku 2014 (k datu 7.2013). Vzniká ve spolupráci W3C[3] a WHATWG[4].

Hlavní důraz je kladen na jednoduchost a bezpečnost. „Nejlepší řešení je to nejjednodušší.“[9] Tato specifikace je stále ve vývoji, ale většina prohlížečů podporuje nové HTML5 elementy.

Byla stanovena některá pravidla:

- Nové funkce by měly být založeny na HTML, CSS[5], DOM a JavaScriptu
- Redukce potřeby externích pluginů (Flash)
- Vylepšení správy chyb
- Používání značkování místo skriptování
- HTML5 by mělo být nezávislé na zařízení

Snaha o jednoduchost a přívětivost pro uživatele je také vidět z deklarace HTML stránky, která se zmenšila z původní (viz 1.1) na pouhý: `<!DOCTYPE html>`.
Mezi nové a zajímavé prvky této specifikace patří: kreslící plocha `<canvas>`, elementy `<video>` a `<audio>`, které slouží pro vkládání multimédií přímo do prohlížeče a nahrazují tak různé pluginy či Flash. Dále pak elementy pro lepší strukturu stránky: `<article>`, `<footer>`, `<header>`, `<section>`, `<nav>` a formulářové elementy: `calendar`, `date`, `time`, `email`, `url`, `seacher`[3].

[3]W3C (World Wide Web Consortium)
[4]WHATWG (Web Hypertext Application Technology Working Group)
[5]CSS (Cascading Style Sheets) – kaskádové styly – jazyk pro popis zobrazení stránek

1.2 Programovací jazyk

1.2.1 JavaScript

JavaScript je multiplatformní skriptovací[6] jazyk vyvinutý Brendanem Eichem v roce 1995. Tento interpretovaný programovací jazyk se často vkládá přímo do HTML stránky (jednoduchost), kde může různým způsobem interaktivně komunikovat s uživatelem (validace formulářů) nebo oživovat animace. JavaScript[11] je podporován všemi moderními webovými prohlížeči. Program vytvořený v tomto jazyce se většinou spouští na straně klienta (po stažení webové stránky) na rozdíl od jazyků běžících na straně serveru (např. PHP nebo ASP).

Programový kód JavaScriptu může být implementován přímo do webové stránky. Pro tuto možnost existuje značka (tag) `<script>`. Máme dvě možnosti umístění skriptu v HTML stránce: v sekci `<body>` nebo `<head>`. V praxi se dává přednost umístění skriptu do sekce `<head>` nebo na úplný konec stránky, abychom měli přehled o veškerém kódu a zároveň nedocházelo k interakcím s obsahem stránky. Pokud umístíme skript do sekce `<body>`, spustí se jakmile na něj při čtení dokumentu dojde řada.

Jinou možností jak připojit JavaScript k HTML (či SVG[7]) je externě. Soubory obsahující zdrojový kód JavaScriptu používají příponu `.js`. Poté může tag vypadat následujícím způsobem: `<script src="nazevSouboru.js"/>`. Protože se při vývoji komponent pracuje s několika JavaScriptovými soubory, bude využita tato varianta.

Velkou přednost tohoto jazyka tvoří možnost přistupovat k objektovému modelu webové stránky neboli DOM[8].

DOM

Tato platforma byla vyvinuta W3C tak, aby umožnila programům a skriptům dynamicky přistupovat a měnit obsah, strukturu a styl dokumentu (HTML, XML nebo v tomto případě SVG). Po načtení stránky vytvoří prohlížeč DOM reprezentující stromovou strukturou celý dokument.

Pro psaní JavaScriptového kódu postačuje jednoduchý textový editor nebo se dají využít komplexnější IDE[9] prostředí např. Visual Studio, Eclipse. Velmi důležitou částí psaní programového kódu je i jeho ladění. K tomuto účelu má většina prohlížečů vlastní nástroje. Mezi uživateli je velmi oblíben program Firebug, který se instaluje jako doplněk pro Firefox. Tento prohlížeč nejlépe podporuje grafiku v SVG a je tedy potřebnou aplikací k vytváření a testování vizualizačních komponent.

V této práci je JavaScript využit k interakci s SVG objekty a jejich animování. Implementace bude přiblížena v kapitole 2.1.

[6] Skriptovací jazyk – odlehčený programovací jazyk
[7] SVG (Scalable Vector Graphics) – škálovatelná vektorová grafika
[8] DOM (Document Object Model) – objektový model dokumentu
[9] IDE (Integrated Development Environment) – vývojové prostředí

1.2.2 jQuery – „write less, do more“

jQuery[6] je JavaScriptová knihovna vytvořená k usnadnění práce s JavaScriptem. Je to open source software pod licencí MIT univerzity a je podporován mnoha prohlížeči i např. firmou Microsoft (ASP.NET AJAX) nebo Google.

jQuery:

- dokáže manipulovat s CSS,
- dokáže vybrat a změnit objekty DOM, upravit HTML události,
- usnadňuje práci s efekty a animacemi,
- podporuje Utility (informace o prohlížeči),
- dokáže pracovat s AJAX[10] technologií

Abychom mohli knihovnu používat můžeme se odkázat na online CDN[11] nebo si můžeme knihovnu stáhnout do lokálního úložiště. V této práci je využito lokálního úložiště, díky tomu se dají vyvíjet komponenty bez přístupu na internet. Knihovna je připojena elementem:

```
<script xlink:href="jquery-1.10.1.min.js" type="application/ecmascript" id="script3"/>
```

jQuery je uzpůsobené pro práci s HTML elementy, využívá tzv. jQuery Selector[7]. Ten byl navržen tak, aby dokázal najít nebo vybrat HTML elementy podle jejich id, třídy, typu nebo atributu.

Základní syntaxe vypadá následovně: `$(selector).action()`. Symbol pro dolar definuje přístup k jQuery knihovně a selector hledá HTML element, na kterém se následně provede daná funkce (action()). Na příkladu 1.2 vidíme selektor $c. Ten nám substituuje tzv. connection string, nebo-li řetězec reprezentující přípojný bod. V objektu *Alias-defs* vyhledá algoritmus alias `DIRTY` a poté nahradí selektor $c přípojným řetězcem, který může vypadat následovně: `HVAC.MODEL_FILTER`.

```
that.$c.DIRTY.on('change',function (i){
 if(i.getValue()){
 ...
 }
});
```

Obr. 1.2: Příklad použití jQuery

[10] AJAX (Asynchronous JavaScript and XML) – načítání obsahu serveru bez nutnosti obnovení stránky
[11] CDN (Content Delivery Network) – síť pro doručování obsahu

1.2.3 JavaScript Object Notation

„JSON[8] neboli JavaScriptový objektový zápis je odlehčený formát pro výměnu dat. Podobnou funkci plní značkovací jazyk XML, ale JSON je jednoduše čitelný i zapisovatelný člověkem a snadno analyzovatelný i generovatelný strojově.“

Přestože je tento standard odvozen z JavaScriptu (používá JavaScriptovou syntaxi pro popis objektů), je na něm zcela nezávislý a může být zpracováván mnoha jinými jazyky. Data jsou ukládána v párech *jméno:hodnota*.

Seznámíme se s dvěma strukturami:

- **Objekt** se zapisuje mezi složené závorky `{}`. Může obsahovat několik párů *jméno:hodnota* oddělené čárkami.

```
{
 "OHEAT": "$T_OHEAT",
 "RUNNING": "$T_RUNNING",
 "L1": "$T_L1",
 "L2": "$T_L2"
}
```

- **Pole** zapisujeme mezi hranaté závorky `[]` a může se skládat z několika objektů.

```
[
 {"alias":"FAN1_OHEAT","cstring":"$T.MODEL_FAN_VO:OHEAT"},
 {"alias":"FAN1_RUNNING","cstring":"$T.MODEL_FAN_VO:RUNNING"},
 {"alias":"FAN1_L1","cstring":"$T.MODEL_FAN_VO:L1"},
 {"alias":"FAN1_L2","cstring":"$T.MODEL_FAN_VO:L2"}
]
```

Jelikož se v tomto formátu zápisu dá udělat mnoho chyb, doporučuje se využívat validátor, který veškeré chyby ohlásí. Jedním z nich je např. JSONLint[12].

JavaScriptový objektový zápis se využije v připojování jednotlivých vizualizačních komponent k běžícímu modelu (respektive k reálnému systému). Tato technika bude vysvětlena v kapitole 2.1 a 5.3.

[12] JSONLint – http://jsonlint.com/

1.3 Rozšiřitelný značkovací jazyk (Extensible Markup Language)

XML[15] patří mezi značkovací jazyky (podobně jako HTML). Hlavním účelem toho jazyka je přenášení a ukládání dat, na rozdíl od HTML, který slouží k zobrazování dat. XML je softwarově a hardwarově nezávislý formát pro přenášení informací. Autor XML dokumentu si musí nadefinovat vlastní značky podle potřeby (na rozdíl od HTML, které je má jasně definované). Pomocí XML můžeme dosáhnout snadno aktualizovatelných dat v HTML.

XML má veliký význam v nově vznikajících internetových jazycích:

- **XHTML**[13] je HTML definované jako XML (přísnější a čistější verze)
- **RSS**[14] slouží k snadné aktualizaci tzv. feedů[15] neboli novinek z webu
- **SMIL**[16] umožňuje integrovat multimediální prvky (hudbu, video) do statických dokumentů HTML nebo XML

Za XML element považujeme vše co se nachází mezi otevírací a zavírací značkou (včetně). Může to být jiný element, text, atribut nebo jejich různá kombinace. Každý XML dokument vytváří stromovou strukturu (od kořene k listům). Na začátku musí být XML správně deklarován verzí a použitým kódováním (viz 1.3).

XML dokument musí dodržovat přísné syntaktické formátování:

- obsahuje kořenový element (root)
- každý XML element obsahuje uzavírací značku
- značky rozlišují malá a velká písmena (*case sensitive*)
- elementy musí být správně vnořeny
- hodnoty atributů musí být v uvozovkách (`""` nebo `''`)

Pokud dodržíme toto formátování, můžeme dokument považovat za „well formed“. Dokument nazveme „validním“ pokud je „well formed“ a zvalidován podle DTD[17].

[13] XHTML (Extensible HyperText Markup Language) – rozšířitelný hypertextový značkovací jazyk
[14] RSS (Rich Site Summary)
[15] Web feed nebo news feed
[16] SMIL (Synchronized Multimedia Integration Language) – jazyk pro integraci a synchronizaci multimediálního obsahu
[17] DTD (Document Type Definition) – definice typu dokumentu

```
<?xml version="1.0" encoding="iso-8859-2"?>
<clanek>
 <autor>Jan Novak</autor>
 <datum>19/07/2013</datum>
 <obsah>
  <nadpis>Úvod do XML</nadpis>
  <text>XML je další ze značkovacích jazyků. Tento jazyk...</text>
 </obsah>
</clanek>
```

Obr. 1.3: „Well formed“ XML

1.3.1 Škálovatelná vektorová grafika (Scalable Vector Graphics)

SVG patří mezi značkovací jazyky v XML formátu, sloužící pro popis dvojrozměrné vektorové grafiky. Existují dva základní druhy počítačové grafiky: vektorová a rastrová. Hlavní výhodou vektorové grafiky je možnost vysokého přiblížení bez ztráty jakékoliv kvality.

- **Rastrová grafika** (gif, jpeg) – pole barevných bodů (pixelů), které dohromady tvoří obrázek. Každý bod je určen přesnou polohou v mřížce a barvou v barevném modelu (např. RGB)
- **Vektorová grafika** (svg, eps) – složena ze základních geometrických útvarů (body, přímky, křivky), které si nesou informace, kde začínají, končí, barvu výplně či okraje

W3C vyvinulo SVG[1] v roce 2001 jako otevřený standard, který na internetu chyběl (konkurence komerční Flash[18] od Adobe). Všechny moderní prohlížeče jej podporují, nejlepší integraci má však Mozilla Firefox (který budeme využívat při vývoji komponent). K vytvoření SVG dokumentu postačuje jednoduchý textový editor. Ale existuje celá řada komerčních editorů (Adobe Illustrator, CorelDRAW) pro komplexnější práci nebo jednodušších open-source editorů (InkScape, SVG-edit).

SVG obrázky jsou definovány pomocí XML, proto mohou být prohledávány, indexovány, skriptovány a komprimovány. Opět musí být dokument správně deklarován (viz 1.4).

Máme několik možností připojení SVG k HTML stránce.

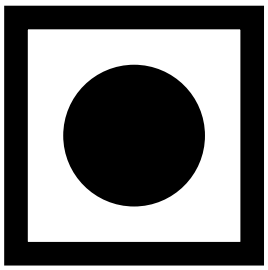
- <**embed**> – podpora všech prohlížečů; podpora skriptování; schváleno pouze v HTML5
- <**object**> – podpora všech prohlížečů; nepodporuje skriptování; schváleno v HTML4/5 a XHTML
- <**iframe**> – podpora všech prohlížečů; vytváří rámeček okolo obrázku; zakázáno v HTML4 a XHTML

[18] Flash – grafický vektorový program pro tvorbu animací

SVG definuje tři základní typy grafických elementů: **vektorové tvary** (`<rect>`, `<circle>`, `<ellipse>`, `<line>`, `<polyline>`, `<polygon>`, `<path>`), **text** a **rastrové obrázky**. Každý element a atribut v SVG[12] může být animován.
Na příkladu 1.4 vidíme, že se má vykreslit čtverec o hraně 20, bez výplně s černými okraji tloušťky 2 a kruh se středem v bodě $[11, 11]$, poloměrem 6 a černou výplní.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg"
 id="ctverec" version="1.1">
<rect width="20" height="20"
 style="fill:none;stroke:black;stroke-width:2"/>
<circle cx="11" cy="11" r="6" style="fill:black"/>
</svg>
```

Obr. 1.4: SVG kód



Obr. 1.5: SVG obrázek

Výhoda SVG formátu tkví v jeho reprezentaci pomocí XML jazyka. Veškeré elementy tak vytváří stromovou strukturu, ke které může být přistupováno pomocí DOM a v tomto případě JavaScriptem. Tím máme možnost snadno měnit veškeré SVG atributy jako například od barvy pozadí až po různé translace a transformace. Vytvoření komponent a jejich funkcí bude popsáno v kapitole 3.1.

1.4 Supervisory Control And Data Acquisition Human-machine interface

Supervisory Control and Data Acquisition (SCADA) můžeme přeložit jako dispečerské řízení a sběr dat. Jedná se o systém, který má za úkol sbírat, zpracovávat data a následně je zobrazovat na operátorské obrazovce. To vše v reálném čase. V období označované Monolithic (viz. níže) se jednalo o plnohodnotný řídící systém, postupem času se však systém vyvinul do podoby programu pro kontrolu procesů a sběr dat ze vzdálených zařízení. Díky dnes již rozšířenému internetu si tak snadno můžeme například nastavovat a sledovat teplotu vytápění v domě prostřednictvím mobilního telefonu či počítače. Hlavní uplatnění se však nachází v průmyslových zařízeních, kde se v případě jakéhokoliv problému spustí alarm a dispečer tak může rychle vyřešit nebo lokalizovat problém. Mezi známé SCADA systémy patří Promotic, Reliance a In Touch.

SCADA systém se skládá z několika subsystémů:

- **HMI**[19] – zařízení zobrazující data uživateli.
- **RTU**[20] – zařízení, které se připojuje k senzorům v procesu. Převádí signály na digitální data a ty posílá dispečerskému systému.
- **PLC**[21] – průmyslový automat používaný k automatizaci procesů v reálném čase.
- **Komunikační síť** – přenáší data mezi senzory a centrálním počítačem. Přenos může být zprostředkován pomocí satelitního, kabelového, mobilního připojení nebo různou kombinací.
- **MTU**[22] – centrální počítačový server, někdy nazývaný SCADA Center nebo hlavní stanice (master station)

Vývoj SCADA systémů se rozděluje do třech generací: monolithic, distributed, networked.

Monolithic

Scada systémy v tomto období byly centralizované na tzv. mainframe počítačích a síť prakticky neexistovala. Připojování k hlavní SCADA stanici bylo velmi limitované.

Distributed

V další generaci byl využit vývoj v počítačovém odvětví, hlavně miniaturizace součástek a využití LAN[23] – lokální počítačové sítě. Sledování procesů bylo rozděleno do více menších

[19] HMI (Human-machine interface) – rozhraní mezi člověkem a strojem
[20] RTU (Remote terminal unit)
[21] PLC (Programmable logic controller) – programovatelný logický automat
[22] MTU (Master Terminal Unit)
[23] LAN (Local Area Network)

stanic komunikujících pomocí LAN v reálném čase. Problém byl v jednoduchých síťových protokolech a jejich bezpečnostním rizikem.

Networked

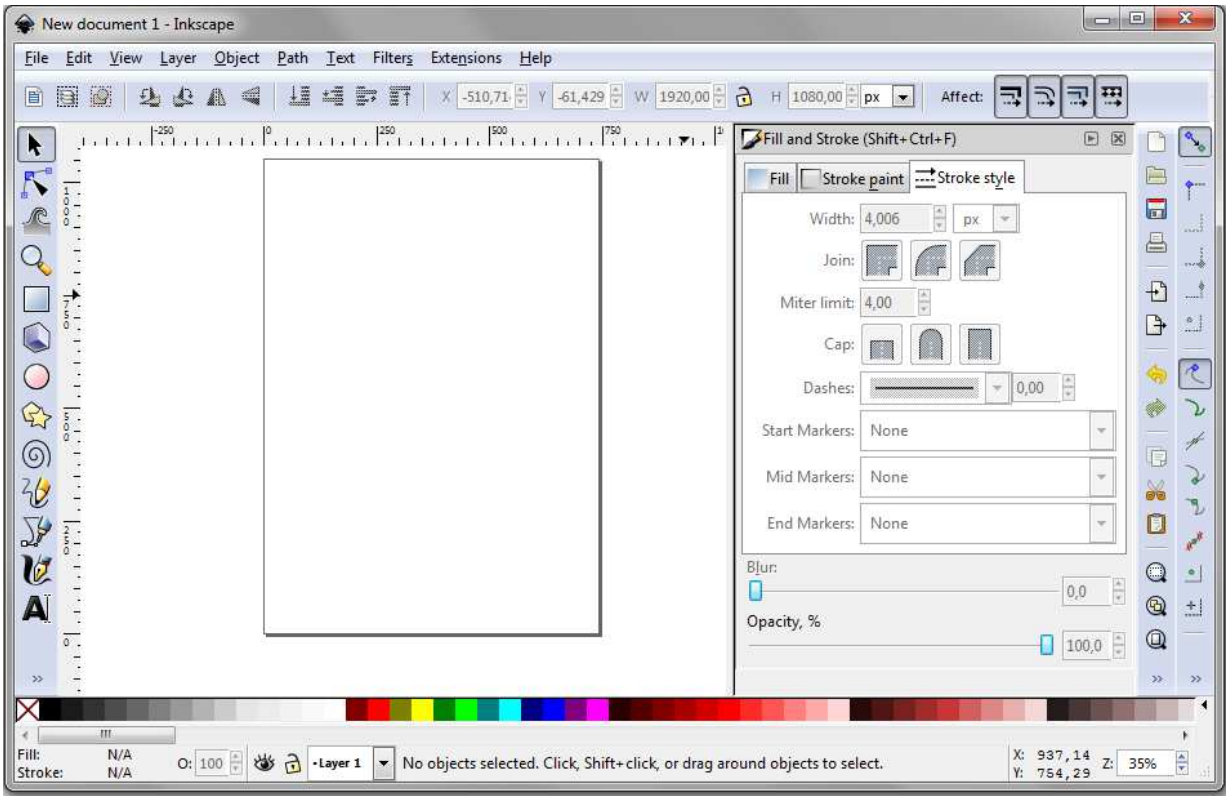
Třetí generace využívá podobnou architekturu jako druhá s tím rozdílem, že oddělené stanice komunikují s hlavní stanící SCADA pomocí WAN[24]. Díky tomuto spojení se k síti snadno připojí další zařízení jako jsou monitory, tiskárny a veškeré systémy, které jsou přístupné z internetu.

Cílem práce je snaha přiblížit se podobou a funkcí k systému HMI, tedy pomocí vizualizačního schématu zobrazit uživateli důležité informace, kontrolovat a ovládat větrací, klimatizační a otopné systémy.

[24]WAN (Wide Area Network) – rozlehlá počítačová síť

1.5 Inkscape

„Inkscape[5] je open source vektorový grafický editor, schopnostmi podobný komerčním programům jako Illustrator, Freehand, CorelDraw, nebo Xara X a to za použití W3C standardu škálovatelné vektorové grafiky (SVG).“ Editor nabízí možnost tvorby cest, textů (i po křivce), vrstev, barevných přechodů (gradientů), mnohoúhelníků či elips. Dále umožňuje komplexní operace s křivkami, aplikaci boolovských operací (sjednocení, rozdíl, ...), trasování bitmap a přímé editování XML.



Obr. 1.6: Prostředí Inkscape

Velkou výhodou editoru Inkscape je licence open source. Na stránce vývojářů[25] můžete program stáhnout pod záložkou *Ke stažení*, kde si lze vybrat mezi verzí pro Mac OS nebo Windows. Použití editoru k tvorbě vizualizačních schémat a jeho ovládání bude probráno v kapitole 5.

[25]Stažení Inkscape – http://inkscape.org/

1.6 Řídící systém REX

Firma REX Controls vyvíjí řídící systém REX[10], který je schopen realizovat komplexní algoritmy automatického řízení. V případě této práce se stará o simulaci klimatizačních jednotek a jejich řízení. K simulačnímu schématu se poté připojují jednotlivé komponenty, odkud získávají údaje zobrazované na vizualizační obrazovce. REX se skládá z několika dílčích programů:

RexDraw

Schémata řídícího algoritmu se vytvářejí v programu RexDraw. Ty se skládají z jednotlivých bloků a k tomuto účelu je určena rozsáhlá knihovna RexLib.

RexView

Tento program umožňuje sledovat veškeré činnosti v jádře systému REX při jeho běhu. Tato vlastnost se využije při ladění chyb algoritmu, během jeho uvádění do provozu.

RexComp

Překladač zkompiluje schéma vytvořené v RexDraw do binárního souboru. Při detekování závažné chyby je překlad ukončen a výsledný binární soubor není vytvořen, ostatní chyby jsou vypsány.

RexCore

Jádro je komplexní program běžící na cílovém zařízení (PC, WinCon, WinPAC, ...). Na základě priorit vykonává jednotlivé úlohy navrženého řízení.

Veškeré důležité informace o tomto softwaru jsou dostupné na internetových stránkách firmy[26].

1.6.1 WebSocket

WebSocket[14] je protokol, sloužící pro obousměrnou komunikaci mezi webovým prohlížečem (s podporou HTML5) a vzdáleným serverem. V případě této práce komunikuje vizualizační schéma s běžící exekutivou v řídícím systému REX. Hlavním účelem protokolu WebSocket je poskytovat komunikaci nezávislou na otevírání velkého počtu HTTP připojení.

[26] REX Controls – http://www.rexcontrols.cz/

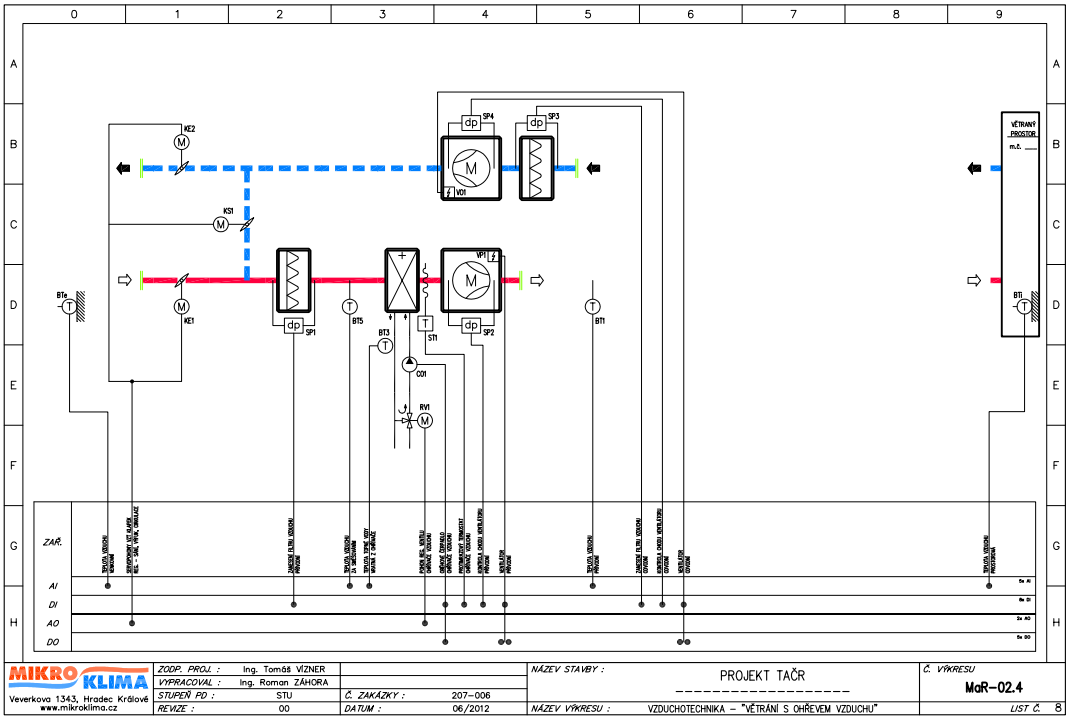
2 Návrhy řešení problému

2.1 Požadavky

Hlavním cílem této práce je vytvořit uživatelské rozhraní pro vizualizaci a řízení klimatizačních jednotek a vytápění s využitím nejmodernější technologie HTML5 (SVG, JavaScript). Tímto způsobem bude zaručena funkčnost na všech PC, bez využití speciálního softwaru, pro zobrazení postačuje pouze prohlížeč.

Vývoj uživatelského rozhraní, se snahou přiblížit se vzhledem a funkčností k systémům HMI, začal na základě požadavku firmy Mikroklima. Firma se komplexně zabývá klimatizacemi, vzduchotechnickými zařízeními a topnými systémy, od projektu přes dodávku až po následný servis.

Hlavním požadavkem firmy bylo na základě předložených technických výkresů vytvořit jednotlivé komponenty tak, aby vzhledově odpovídaly předloze (obr. 2.1) a vizualizačně plnili jejich funkce (např. při zapnutém ventilátoru se točí listy).



Obr. 2.1: Technický výkres vzduchotechniky

Výhodou tohoto řešení bude snadné zaškolení techniků ve firmě, kdy skládání schémat na vývojovém PC bude probíhat jako návrh technického výkresu.

Požadavky:

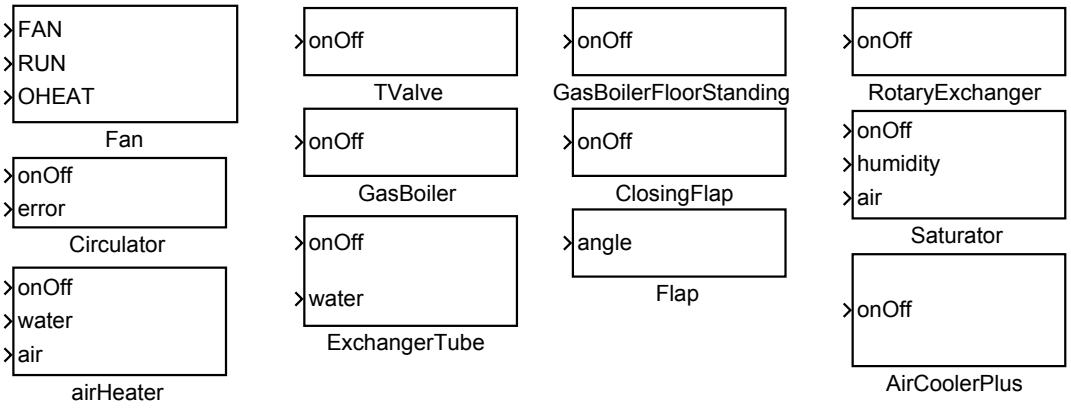
- podpora webových prohlížečů Firefox, Chrome
- komunikace se serverem pomocí WebSocket
- využít open source software
- jednoduchá implementace

2.2 Fáze vývoje

Celé uživatelské rozhraní se skládá z knihovny komponent, které jsou vytvořené pomocí SVG a JavaScriptu, a modelu sestaveném v řídícím systému REX. Součástí modelu je řídící algoritmus, který reguluje klimatizační (otopné) systémy a další subsystémy simulující jednotlivá zařízení. Snahou bylo vyvinout rozhraní tak, aby bylo možné z knihovny poskládat vizualizační schéma a zároveň co nejjednodušeji sestavit model v systému REX. Z tohoto důvodu byl model rozdělen na tři části: algoritmus řízení, model TZB[1] a model místnosti.

2.2.1 První fáze

Při prvních pokusech byly ke komponentám vytvořeny odpovídající subsystémy v programu REX, reprezentující daná zařízení, do kterých byly přiváděny potřebné signály (obr. 2.2). Tato varianta usnadňovala připojení vizualizačních komponent, ale na druhou stranu neumožňovala modelování složitějších schémat v programu REX.

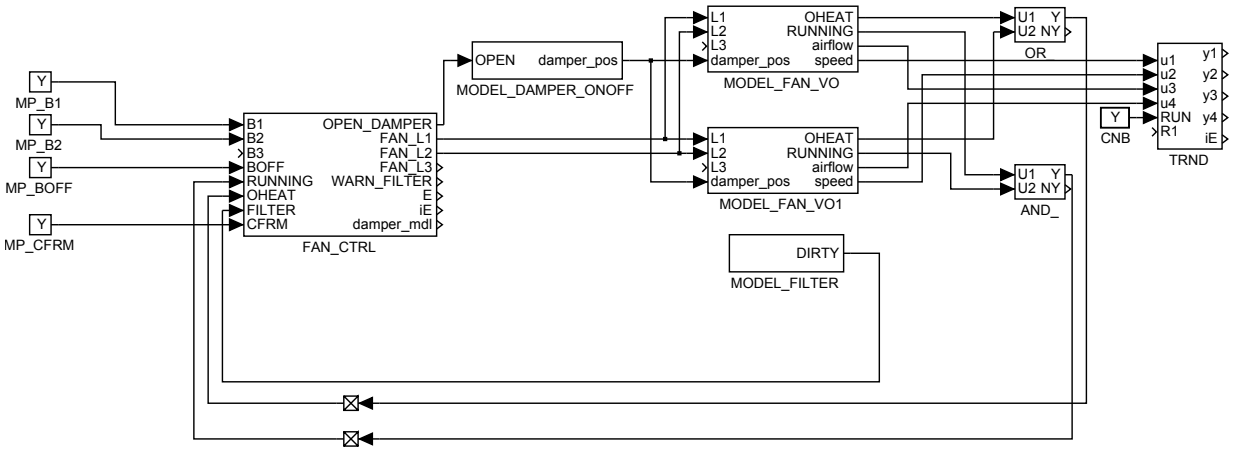


Obr. 2.2: Subsystémy jednotlivých zařízení

[1]TZB[13] (Technické Zařízení Budov) – zahrnuje obory instalace (vytápění, vzduchotechnika), elektrotechnické rozvody (měření, regulace) a další...

2.2.2 Druhá fáze

Z tohoto důvodu byla metoda připojování přímo na subsystémy zavrhnuta. V novém metodě se komponenty připojují na konkrétní signály REX exekutivy pomocí přípojných řetězců neboli connection string's. Takto vytvořená schémata mají podobnou strukturu jako příklad obrázku 2.3. Veškerou tvorbu schémat a řídících algoritmů v systému REX obstarává Ing. Jaroslav Sobota, Ph.D. Princip přípojných řetězců je vysvětlen v následující kapitole a používání je s několika ukázkami probráno v kapitole Tutoriál.



Obr. 2.3: Používaný systém

3 Implementace

3.1 Vývoj komponenty

Jako základní stavební kámen uživatelského rozhraní bude knihovna, skládající se z jednotlivých komponent. Jak již bylo představeno v kapitole 1.3.1, komponenty byly vytvořeny pomocí vektorové grafiky (SVG) v programu InkScape. Každá komponenta je tedy poskládaná z jednotlivých grafických elementů (čar, obdélníků, čtverců, elips a kružnic). Tyto elementy mají své vlastnosti určené atributy. Například obdélník je určen jeho pozicí (souřadnice: x, y), velikostí (width, height), tloušťkou okrajů, jejich zaoblením nebo barvou výplně. Atributy elementů za nás zavádí samotný InkScape a práce je tím usnadněna.

Při vytváření komponent byly uplatněny tzv. kontejnery neboli „g“ elementy. „g“ element je zkratkou slova *group*. Z důvodu častého používání tohoto kontejneru je využit krátký zápis, aby zůstal dokument přehledný. Pomocí kontejneru máme možnost jednotlivé elementy shlukovat a tím je označit pod jediným *id*. Hlavní využití shlukování se najde při animaci, kdy budou animovány veškeré elementy uvnitř kontejneru.

V této práci je celá komponenta uzavřena v „g“ elementu tak, aby mohl být definován její název, ten se skládá z anglického výrazu, pomlčky a čísla (`id="AirFilter-1")`, a dále atribut `rexsvg:module="nazevKomponenty"`. Další elementy potřebné pro běh jsou *title* a *desc*. Oba elementy slouží pro popis kontejneru. Titulek se zobrazuje při najetí ukazatele myši na komponentu a slouží při substituci, kdy se v připojovacích bodech nahradí `$T` názvem v titulku (`<title id="nejakeId">AIRFILTER</title>`).

V elementu *description* (desc) jsou uloženy přípojné řetězce (connection string's) a další volitelné informace (*options*) v jazyku JSON. Skládá se ze dvou objektů: „connections“ a „options“. Podle funkce komponenty musí být definován typ přípojného řetězce. Pokud funkce požaduje čtení hodnot z programu REX, necháme přípojný řetězec bez označení nebo přidáme řetězec `"type":"R"` (viz. čtvrtý řádek obr. 3.1). Jestliže je nutné zapisovat hodnoty do programu REX, je nutné přidat řetězec `"type":"W"` (čtvrtý řádek obr. 3.2).

```
"connections":{
  "prvni":"alias-prvni-polozky",
  "DIRTY":"$T_DIRTY",
  DIRTY:{alias:"$T_DIRTY",cstring:"HVAC.MODEL_FILTER:DIRTY"}
 },
  "options":{}
```

Obr. 3.1: Element description s přípojnými řetězci

Byly vytvořeny dvě možnosti zapsání přípojných řetězců:

- První možností, která se využije spíše při vývoji komponent, je definovat celý objekt přípojného bodu. Tak je tomu na třetím řádku obr. 3.1. Zde je definován objekt `DIRTY` s aliasem `AIRFILTER_DIRTY` (po provedení substituce za `$T`) a přípojným řetězcem `HVAC.MODEL_FILTER:DIRTY`, který je dán zapojením v modelovém příkladu (úplná adresa). Pokud toto dodržíme, komponenta bude fungovat a zobrazovat informace v samostatné stránce prohlížeče (bez použití *SVG_ Template*, viz dále).
- Druhá možnost, tzv. uživatelská, spočívá v tom, že přípojný řetězec bude definován v objektu `Alias-Defs` (v připraveném schématu *SVG_ Template*). Tato varianta je vidět na třetím řádku obrázku 3.1. Po otestování komponenty použije vývojář tuto variantu. Tím bude docílena snadná uživatelská manipulace, kdy se v jednotlivých komponentách nebude nic nastavovat a veškeré přípojné řetězce se zapíší pouze do objektu `Alias-Defs`, jako je vidět na obrázku 3.2.

```
    "connections": [
{"alias":"AIRFILTER_DIRTY","cstring":"$T.MODEL_FILTER:DIRTY"},
{"alias":"CLOSINGDUMPER_POSITION","cstring":"$T.FAN_CTRL:OPEN_DAMPER"},
{"alias":"MP_B1", "cstring":"HVAC.MP_B1:BSTATE", "type":"W"},
{"alias":"MP_B2", "cstring":"HVAC.MP_B2:BSTATE", "type":"W"}
     ],
    "options": {}
```

Obr. 3.2: Objekt Alias-Defs s přípojnými body

Druhý objekt v elementu *description* s názvem *options* slouží pro zadávání doplňujících informací pro jednotlivé komponenty. Mezi tyto vlastnosti můžeme vyjmenovat např: požadovanou barvu zapnuté LED kontrolky; vlastní barvu pro chybové nebo varovné hlášení; nastavení doby, po kterou se má zavírat klapka s digitálním vstupem. Veškeré vstupy mají své default nastavení v jednotlivých skriptech, nejprve se načte hodnota uvedená v objektu *options* a pokud nic neobsahuje, aplikuje se default hodnota.

Jak je možné vidět z uvedených příkladů, funkčnost komponenty a následného schématu záleží na správném zapsání dat ve formátu JSON. Tato část může být pro uživatele málo přehledná a existuje velká šance udělat chybu. Proto byl ve druhé kapitole zmíněn JSON validátor, který upozorní na špatně zformátovaný řetězec. Nicméně optimálního řešení nebylo dosaženo z několika důvodů. Za prvé je zde vysoký potenciál chybného zapsání, který povede k nefunkčnosti komponent a za druhé se zde míchají dva různé jazyky: JSON jako textový formát a SVG definované ve značkovacím jazyce XML. Pokud byl vybrán SVG jazyk k tvorbě grafiky bude vhodné využít XML pro přenášení informací. Celé uživatelské rozhraní je stále ve vývoji (k datu 31.7.2013). Nutností bude vytvořit plugin[1] do programu

[1] Vytvořením pluginu se zabývá kolega Bc. Lubomír Kristek.

InkScape, který usnadní zadávání přípojných řetězců a dalších vlastností. Zároveň bude odstraněna závislost na jazyce JSON a bude používán značkovací jazyk XML.

3.1.1 Externí skripty

Ke každé komponentě je nutné připojit čtyři JavaScriptové soubory, které se starají o běh všech komponent a schémat.

- jquery-1.10.1.min.js – knihovna jQuery pro usnadnění práce s JavaScriptem
- rex-weblim.min.js – slouží pro komunikaci komponent se systémem REX pomocí protokolu WebSocket
- rex-ui-svg.js – umožňuje manipulaci s komponentami (viz. níže)
- nazevKomponenty.js – obsahuje algoritmus komponenty (viz. níže)

3.1.2 Jmenný prostor (Namespace)

První skript vytvořil kolega Ing. Ondřej Severa. `rex-ui-svg` vytváří vlastní jmenný prostor `xmlns:rexsvg="http://www.rexcontrols.com/rexsvg"`. Namespace neboli jmenný prostor byl definován, aby se zabránilo konfliktu s označením elementů. Deklarace jmenného prostoru probíhá podle dané syntaxe: `xmlns:prefix="URL"`. URL[2] definuje umístění zdroje na serveru a většinou jsou zde uvedeny odkazy na webové stránky obsahující informace o jmenném prostoru. Tento skript obsahuje funkce, které umožňují veškerou manipulaci s SVG komponentami.

Jsou zde funkce:

- registrující jednotlivé komponenty
- pro načtení dat z elementu description a převedení jej z jazyka JSON
- pro načtení přípojných řetězců z objektu Alias-defs, pokud jsou definované
- getChild(nazevElementu), která nalezne element podle jeho id

Poslední skript je speciální pro každou komponentu. Obsahuje základní funkce zděděné z hlavního skriptu `rex-ui-svg` a další pro ovládání komponenty. Ty budou podrobně představené v další kapitole 4.

3.1.3 ViewBox

ViewBox je dalším důležitým atributem v definici samotného („root“) SVG elementu, který umožňuje vývojáři zobrazit jen část z celkové plochy. Toho se dá využít pro přizpůsobení obsahu stránky různým formátům zobrazovacího zařízení jako jsou monitory, tablety

[2]URL (Uniform Resource Locator)

nebo smartphony, kdy se dané schéma automaticky přizpůsobí (roztáhne či zmenší) velikosti displeje. Po vytvoření komponenty musí být atribut ručně dopsán a musí být nastaveny jeho správné parametry. Jakmile Inkscape zjistí, že je tento atribut přítomen, dále s ním pracuje a upravuje jej. Proto se uživatelé nemusí zabývat jeho definicí v nově vytvořených schématech.

3.1.4 Ukládání dat

Inkscape umožňuje mnoho způsobů uložení hotové grafiky. Mezi základní tři patří SVG soubory (inkscape, plain a optimized), které si dále rozebereme. Dalšími možnostmi jsou pdf (Portable Document Format), eps (Encapsulated PostScript), png (Portable Network Graphics), ps (PostScript), tex a mnoho dalších. Nyní se podíváme na rozdíly mezi SVG soubory, které nám Inkscape nabízí.

Plain SVG

Tento formát se pro použití absolutně nehodí. Odstraňuje námi definovaný namespace `xmlns:rexsvg` a atributy s ním spojené, namespace `xmlns:xlink`, který poskytuje funkčnost přiložených skriptů a s tím i veškeré skripty. Jako poslední odstraňuje atribut `onload`.

Optimized SVG

Optimized SVG odstraňuje pouze jediný namespace `xmlns:svg`. Bohužel přepisuje adresy umístění skriptů na úplné (od kořenového adresáře). Tím by se nám komplikovala práce s komponentami, pokud by byla potřeba přesouvání souborů v adresářích PC.

Inkscape SVG

Poslední formát je vhodný k ukládání jednotlivých komponent a schémat. V programu je nastaven jako default. Neodstraňuje žádné namespace, ani funkce námi definované. Naopak přidává dodatečné informace poskytnuté programem Inkscape jako je verze programu, použité přiblížení při poslední editaci, velikost stránky, použité jednotky a mnoho dalšího.

3.1.5 Dávkový soubor

Po vytvoření nebo jakékoliv úpravě JavaScriptového kódu je nutné použít textový soubor s příponou `.bat` neboli dávkový soubor. Ten obsahuje sérii příkazů, které se provedou po spuštění. V případě této práce se po spuštění souboru `compile.bat` spojí veškeré skripty jednotlivých komponent s kódem, obsaženým v `rex-ui-svg.js`, a vytvoří nový soubor `rew-svgvis.js`.

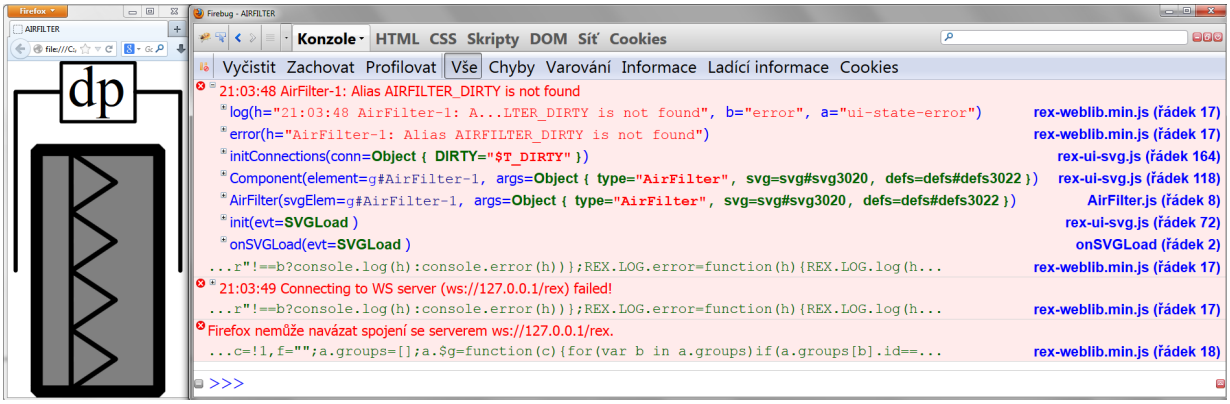
3.1.6 Debugger

Vytváření nových schémat a práce s komponentami může přinést řadu chyb. Jak při vývoji, zde je ladění samozřejmostí, tak i při samotné implementaci. Abychom zabránili výskytu co nejvíce chyb, používají se ladící programy, tzv. debuggery. Jelikož se v této práci využívá převážně prohlížeč Firefox od Mozilly, byl použit i jeho doplněk Firebug. Pomocí

Firebugu lze sledovat, upravovat a ladit veškeré CSS, HTML, DOM a JavaScript. Právě JavaScript debugger je zde velikým pomocníkem. Hlavní části kódu, pracující s připojováním WebSocketu nebo překladem JSON, obsahují příkazy pro chybové výpisy (pokud chybí definice přípojného řetězce ⇒ error). Pomocí klávesy `F12` vyvoláme panel pro zobrazení těchto výpisů.

Firebug byl hojně využíván kvůli možnostem:

- pozastavit vykonávání kódu v jakémkoliv okamžiku,
- přidání podmínek pro zastavení vykonávání kódu,
- zadání proměnné, která je sledována po celou dobu běhu kódu,
- tooltipu, kdy se po najetím myši na proměnou zobrazí její vlastnosti



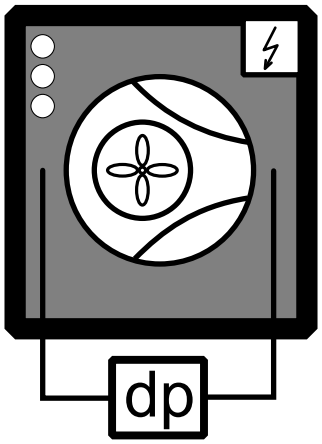
Obr. 3.3: Komponenta AirFilter se špatně definovaným připojením

Na obrázku 3.3 je zobrazena komponenta AirFilter, která se nedokázala připojit k běžící exekutivě v programu REX. V tomto případě se jedná o (úmyslně) nezapnutou komunikaci WebSocket. Na výpisu vidíme problém s nalezením přípojného řetězce proměnné `AIRFILTER_DIRTY` a nenalezený WebSocket server. Jiným případem by mohl být špatně definovaný přípojný řetězec v JSON (opět by zde byl zobrazen error). Šedá výplň komponenty upozorňuje na problém s navázáním spojení, poté je nutné otevřít debugger a najít příslušnou chybu.

4 Knihovna komponent

V této kapitole bude popsána knihovna vytvořených komponent, jejich funkčnost a použití. Každá komponenta je složena z SVG souboru, ten je na obrazcích, a JavaScriptového kódu, který je uveden v příloze. U každé komponenty je nejprve uveden stručný popis komponenty, poté seznam *connection stringů* a seznam *options* s vysvětlením jejich funkcí. Kde se nevyskytují žádné možnosti z objektu *options* není tento seznam uveden. U signálů se objevují zkratky DI (digital input), DO (digital output), AI (analog input) a AO (analog output). V případě digitálních vstupů (resp. výstupů) lze číst (resp. nastavovat) pouze hodnoty true/false (tedy on/off) na rozdíl od analogových, kde jsou přenášeny číselné hodnoty (jako teplota potrubí nebo natočení klapky).

4.1 Ventilátor (Fan)



Obr. 4.1: Fan

Komponenta Fan simuluje ventilátor s různými stupni rychlostí. Počet rychlostí je určen podle počtu definovaných přípojných bodů. Ty jsou zobrazeny pomocí LED kontrolek v rohu komponenty. Kontrolky se rozsvěcují či zhasínají podle aktuálního stupně a společně s tím se otáčejí listy ventilátoru. Tato komponenta obsahuje stavovou logiku: jestliže přijde signál zapnutí, ale listy se netočí, svítí žlutě, jakmile se začnou točit, rozsvítí se zeleně. Opačný postup je při vypínání.
Při přehřátí ventilátoru (`OHEAT="1"`) proběhne proces vypínání a okolí listů spolu s napájením svítí červeně.
Ventilátor obsahuje čidlo pro kontrolu diference tlaku vstup/výstup. Pokud se obě hodnoty neshodují, kontrolka diference `DP` se rozsvítí červeně.

connection:

OHEAT (DI) – přehřátý ventilátor; hodnota true ⇒ červená kontrolka (listy a napájení)

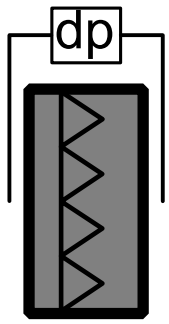
RUNNING (DI) – diference tlaku vstup/výstup; zanesení ⇒ červená kontrolka `DP`

L1 (DO) – jeden stupeň větrání

L2 (DO) – dva stupně větrání

L3 (DO) – tři stupně větrání

4.2 Vzduchový filtr (AirFilter)



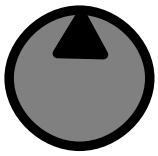
Obr. 4.2: AirFilter

Filtr obsahuje čidlo, které podle diference tlaku na vstupu a výstupu pozná, jestli je průchozí či zanesený. Při problému je uživatel upozorněn varovnou barvou.

connection:

DIRTY (DI) – zanesený filtr; hodnota true ⇒ varovná kontorlka

4.3 Oběhové čerpadlo (Circulator)



Obr. 4.3: Circulator

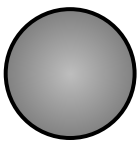
Oběhové čerpadlo, které vhání vodu do systému.

connection:

ENABLE (DO) – běh čerpadla; zelená kontrolka

OHEAT (DI) – přehřátí čerpadla; červená kontrolka

4.4 Kontrolka (Led)



Obr. 4.4: Led

Kontrolku lze připojit k jakémukoliv signálu a nastavit požadovanou barvu. Ta se zadává do objektu *options* šestnáctkovým RGB zápisem ("#FFFFFF"- bílá barva).

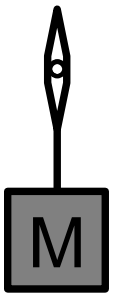
connection:

LIGHT (DI) – signalizace zapnuta/vypnuta

options:

color – zadání požadované barvy

4.5 Klapka (ClosingDamper)



Obr. 4.5: ClosingDamper

VZT klapka se servopohonem obsahuje pouze digitální výstup, což znamená možnost úplného otevření nebo úplného zavření. Natáčení klapky je animováno otáčejícím se listem klapky, proto se v proměnné `open_time` nastavuje požadovaný čas naklápění klapky.

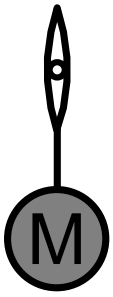
connection:

POSITION (DO) – otevření/uzavření klapky

options:

open_time – čas otevírání/zavírání klapky v sekundách

4.6 Klapka (Damper)



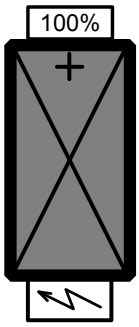
Obr. 4.6: Damper

VZT klapka se servopohonem obsahující analogový vstup, což dovoluje plynule řídit otevírání/zavírání klapky. Hodnoty v rozmezí $0 - 1$ (1 otevřeno) jsou načítány z REXu podle stupně větrání.

connection:

POSITION (AO) – hodnota $0 - 1$ je přepočtena na stupně pro otočení listu klapky

4.7 Ohřívač vzduchu (ElAirHeater)



Obr. 4.7: ElAirHeater

Komponenta ElAirHeater reprezentuje elektrický přímotopný ohřívač vzduchu. Horní kontrolka udává nastavený výkon ohřívače. Při přetížení vypadne jistič (červené světlo napájení).

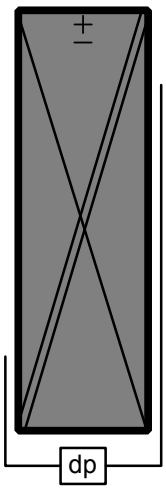
connection:

ENABLE (DO) – zapnutí/vypnutí ohřívače; zelená kontrolka

POWER (AO) – nastavený výkon ohřívače

CIRCUITBREAKER (DI) – jistič ohřívače; hodnota true ⇒ vypnutí; červená kontrolka napájení

4.8 Rekuperační výměník (Exchanger)



Obr. 4.8: Exchanger

Exchanger představuje rekuperační tepelný výměník. V případě namrznutí se rozsvítí červeně.

connection:

ICE (DI) – namrznutí výměníku; hodnota true ⇒ červená kontrolka

4.9 Display

Obr. 4.9: Display

Display slouží pro zobrazení jakékoliv číselné hodnoty běžícího systému. Ve volitelných parametrech lze nastavit počet desetinných míst a hraniční body, které slouží k obarvení číselného údaje, pokud hodnota překročí horní nebo klesne pod spodní mez. Všechny barvy se zapisují v šestnáctkovém RGB formátu.

connection:

VALUE (AI) – hodnota sledované proměnné

options:

number_rounding – počet desetinných míst

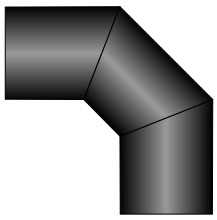
color_min – barva spodní meze ("#0000FF"– modrá)

color_max – barva horní meze ("#FF0000"– červená)

range_min – hodnota spodní meze

range_max – hodnota horní meze

4.10 Potrubí (Pipe)



Obr. 4.10: Pipe

Komponenta Pipe reprezentuje potrubí zobrazující aktuální teplotu uvnitř kolujícího vzduchu. Podle zadaných počátečních hodnot vypočítává algoritmus výslednou barvu pomocí lineární interpolace. V knihovně se nacházejí tři komponenty: rovné, „T“ potrubí a koleno.

connection:

VALUE (AI) – aktuální teplota vzduchu v potrubí

options:

range_min – počáteční teplota

range_max – koncová teplota

color_min – počáteční barva v šestnáctkovém RGB zápisu

color_max – koncová barva v šestnáctkovém RGB zápisu

4.11 Termostat 1

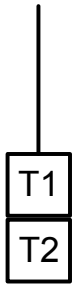
Obr. 4.11: Thermostat

Termostat, sloužící k udržování stálé teploty, je zde reprezentován jedním až třemi čtverci podle použití. Termostat s jedním digitálním vstupem signalizuje dosažení provozní teploty zelenou kontrolkou.

connection:

THERMO (DI) – dosažení provozní teploty

4.12 Termostat 2



Obr. 4.12: Thermostat

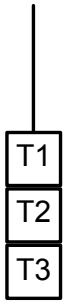
Termostat se dvěma digitálními vstupy signalizuje dosažení provozní teploty zelenou kontrolkou (`T1`) a kritické přehřátí červenou kontrolkou (`T2`).

connection:

THERMO (DI) – dosažení provozní teploty

THERMO2 (DI) – kritické přehřátí

4.13 Termostat 3



Obr. 4.13: Thermostat

Termostat se třemi digitálními vstupy signalizuje dosažení teploty vychlazení zelenou kontrolkou (`T1`), provozního maxima žlutou kontrolkou (`T2`) a kritické přehřátí červenou kontrolkou (`T3`).

connection:

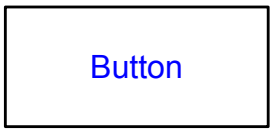
THERMO (DI) – signál pro vychlazení

THERMO2 (DI) – provozní maximum

THERMO3 (DI) – kritické přehřátí

Následující tlačítka byly vytvořeny Ing. Ondřejem Severou pro ovládání vizualizací. Obě komponenty jsou, po spuštění vizualizace v prohlížeči, nahrazeny příslušnými tlačítky a boxy z jazyka HTML. Tyto komponenty jsou ve stádiu vývoje a správného zobrazení a funkčnosti je docíleno pouze v prohlížeči Firefox.

4.14 Tlačítko (Button)



Obr. 4.14: Button

Tlačítko je využíváno pro ovládání dalších komponent (zapnutí/vypnutí ventilátoru). Při stisknutí je zapsána hodnota 1 a při puštění tlačítka je zapsána hondonta 0.

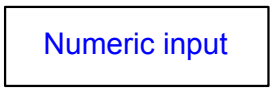
connection:

btn (DO) – zápis hodnoty

options:

type – povinná hodnota button

4.15 Zadávací box (Input)



Obr. 4.15: Input

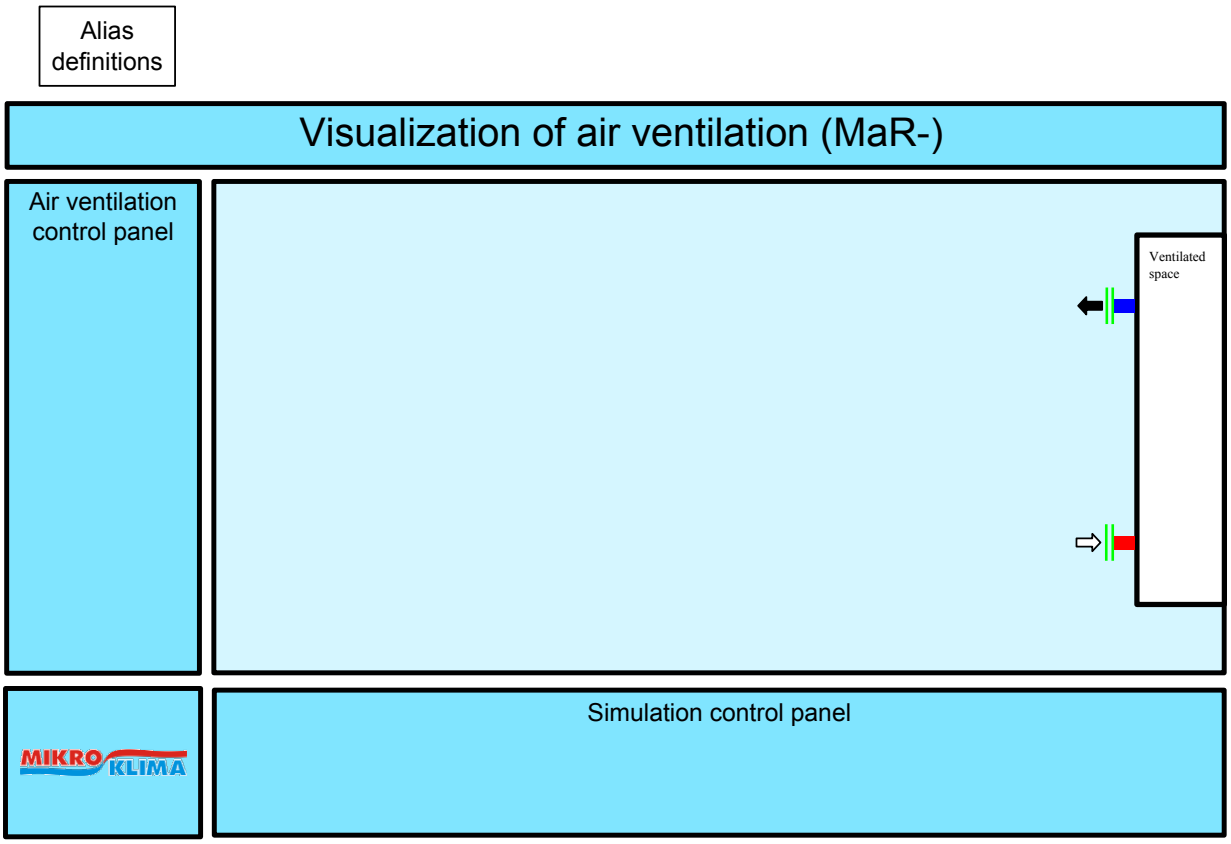
Komponenta Input slouží pro zadávání číselných hodnot (např. požadovaná teplota) do systému REX.

connection:

value (AO) – zápis hodnoty

4.16 Ukázková schémata

Po seznámení s jednotlivými komponenty je třeba představit vytvořené vzorové plátno, které se využije pro vkládání komponent a jejich připojení. Základní plátno, s předdefinovaným designem, je pojmenované `SVG_Template`. Na rozdíl od komponent přidružuje pouze tři JavaScriptové soubory: knihovnu `jQuery`, `rex-weblib.js` pro práci s WebSocket serverem a knihovnu všech JavaScriptových kódů uložených v `rex-svgvis.js`.



Obr. 4.16: SVG_Template

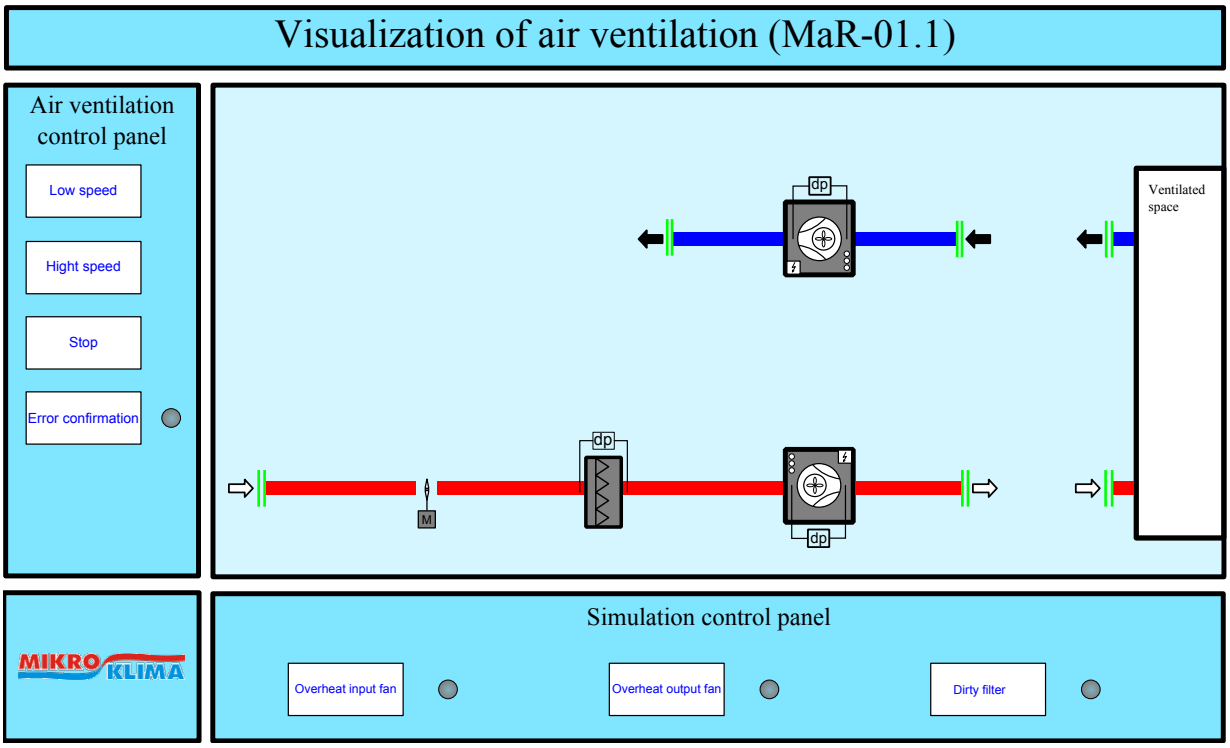
Do tohoto plátna se vkládají vytvořené komponenty, ovládací tlačítka a zadávací boxy, které společně vytvoří vizualizační schéma. Velikost a umístění komponent se přizpůsobí danému technickému výkresu, reprezentující klimatizační systém. Následně se nastaví připojné řetězce v objektu *Alias definitions*. Po uložení a spuštění schématu v prohlížeči se objekt *Alias definitions* skryje. Veškerá manipulace s komponentami, tvorba schémat a nastavení připojení je podrobně probráno a názorně ukázáno v kapitole 5. Pro praktickou ukázku bylo vytvořeno několik různě složitých vizualizačních schémat, které jsou popsány na dalších stránkách. Jelikož jsou obrázky vygenerovány z editoru InkScape, nemají tlačítka vzhled jako v prohlížeči (komponenta tlačítka je v prohlížeči nahrazena HTML tlačítkem).

4.16.1 MaR-01.1

První vytvořené schéma představuje jednoduché větrání v místnosti. Pokud je v místnosti horko, zapne uživatel ventilátor (nízké nebo vysoké otáčky). K vytvoření byly použity dva ventilátory (přívodní a odvodní), vzduchový filtr, klapka s digitálním výstupem a několik ovládacích tlačítek. Ve schématu není žádné měřící čidlo, tudíž nelze zadávat žádné teploty a potrubí je pouze statickým prvkem.

V levém panelu (slouží pro ovládání větrání, viz kapitola 5) lze zvolit nízké či vysoké otáčky nebo zastavení ventilátoru a potvrzení chyby v případě, že nějaká nastala. Ve spodním panelu (ovládání simulace chyb) se nacházejí tlačítka pro přehřátí ventilátorů a zanesení filtru.

Při zapnutí větrání se otevře klapka a roztočí se listy ventilátoru. V případě přehřátí nebo neshodě vstupního a výstupního tlaku bude větrání zastaveno a klapka uzavřena. Rozsvítí se příslušná kontrolka(ky) a v levém panelu bude svítit *Error confirmation*, dokud se neodstraní závada a nepotvrdí kliknutím na stejnojmenné tlačítko.



Obr. 4.17: Větrání

4.16.2 MaR-01.2

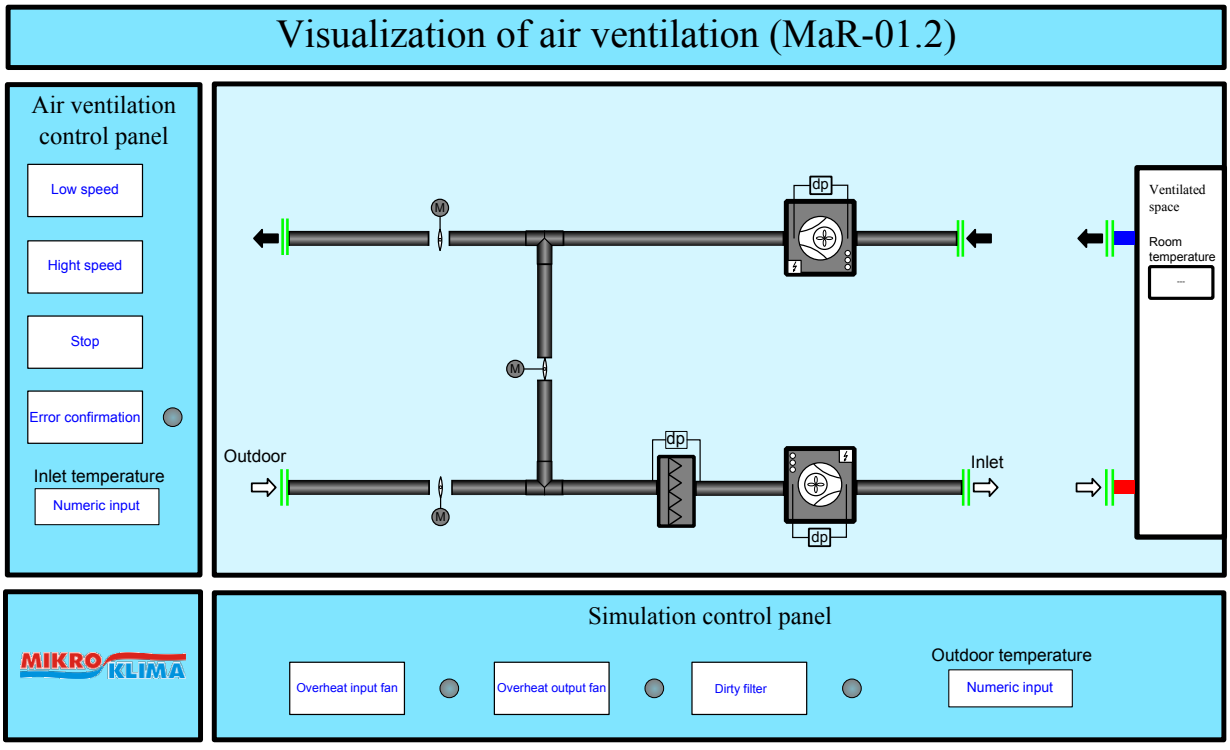
V následujícím schématu je vytvořené větrání s mícháním čerstvě příchozího a z místnosti odchozího vzduchu. Větraná místnost má vytápění regulované na teplotu 22°C s hysterezí 1 stupeň.

Základní princip a ovládání je stejné jako na předešlém schématu, s rozdílem, že zde uživatel nastavuje požadovanou teplotu vzduchu, který je vháněn do místnosti. Tento vzduch se namíchá z venkovního vzduchu a vzduchu z místnosti pomocí natáčení tří klapek. Ty jsou řízeny PI regulátorem.

V levém panelu lze ovládat dvě rychlosti ventilátorů a nastavovat teplotu vstupního vzduchu. Panel pro simulaci poruch je téměř stejný jako v předešlém příkladě. Navíc je pouze box pro zadání venkovní teploty.

Toto schéma již obsahuje čidlo pro měření teploty, a proto jsou zde užity komponenty Pipe, tedy dynamické potrubí, které se přebarvuje podle teploty vzduchu v něm.

Na pravé straně v objektu *Ventilated space* se nachází displej, zobrazující aktuální teplotu v místnosti.



Obr. 4.18: Větrání

5 Tutoriál

Kapitola tutoriál slouží k seznámení a pochopení základního použití programu Inkscape se zaměřením na tvorbu a editaci vizualizačních schémat klimatizačních a otopných systémů. Co to Inkscape je, co dokáže a jak ho pořídit bylo podrobně probráno v kapitole 1.5. V této části budou představeny nástrojové panely, základní fungování a zkratky, které usnadní práci s komponentami. V této kapitole bude také probráno jak vložit a vhodně upravit komponentu, nastavení přípojných bodů a uložení schématu.

5.1 Seznámení s InkScape

Pro seznámení s pracovním prostředím dobře poslouží připravený obrázek 5.1, ve kterém jsem zvýraznil a pojmenoval jednotlivé panely. Pokud při práci není třeba jakýkoli panel a pouze „zabírá místo“ na obrazovce, velice snadno může být skryt v menu *Views* → *Show/Hide*.

Panely

V horní části se nacházejí dva panely, *Commands* a *Tool control*. *Commands panel* nabízí zástupce za různé položky v menu tak, aby byly přístupné k rychlému použití (např. vytvoření nového dokumentu, uložení, tisk, vrácení poslední akci, atd.). Naproti tomu *Tool control* panel je vždy specifický pro použitý nástroj. Nabízí možnosti pro změnu velikosti, umístění nebo tloušťky v případě vytváření nového elementu nebo nastavení fontu, velikosti a zarovnání při použití nástroje text, apod.

Tool Box, přichycený k levému okraji plátna, obsahuje nástroje pro vybírání, kreslení nebo jinou modifikaci objektů. Jednoduše se vybere požadovaný nástroj levým kliknutím myši, v *Tool Control* panelu naskočí možnosti pro daný nástroj, a použije se na plátno.

Na opačné straně plátna je umístěn tzv. *Snap Control* panel, ve kterém se nastavuje přichycování objektů k bodům, jiným objektům, ohraničení stránky nebo vodicím linkám.

Jako poslední a důležitý panel je uveden *Status Bar*. Ten informuje uživatele o barvě výplně, ohraničení a průhlednosti vybraného objektu, aktuální kreslící vrstvě (informace, jestli je uzamčená a viditelná), pozici ukazatele a o aktuálním zvetšení.

SVG_Template

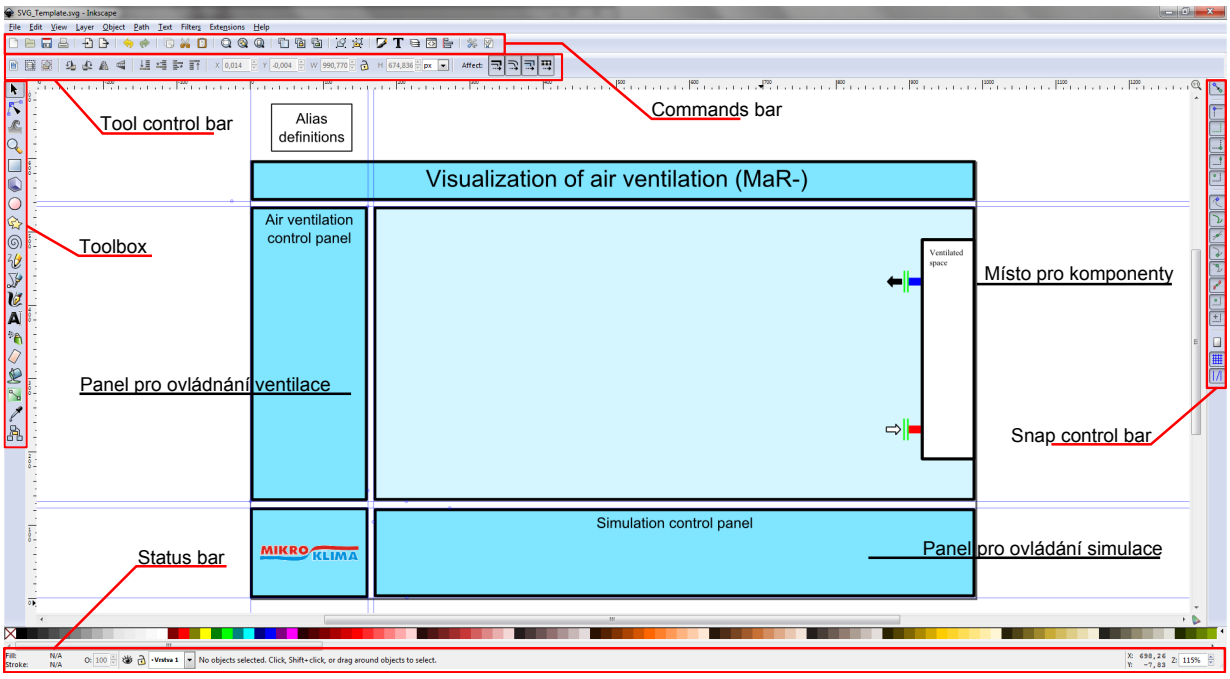
Nyní jsme seznámeni s panely a pracovní plochou programu InkScape. Pokud budeme chtít vytvořit nové schéma, potřebujeme otevřít předpřipravený SVG_Template (obr. 4.16). To provedeme pomocí *File* → *Open...*, ikonkou na *Command panelu* nebo za pomoci zkratky `[Ctrl+O]` a vyhledáme template v průzkumníku. Abychom předešli

nechtěné úpravě SVG_Template, doporučuji ho hned uložit pod jiným názvem (název vytvářené vizualizace) pomocí položky *File* → *Save As...* nebo zkratkou `[Ctrl+Shift+S]`. Nové schéma musí být uloženo jako Inkscape SVG (*.svg). Důvody, proč je používán tento formát, byly objasněny v kapitole 3.1.4.

Plocha SVG_Template je rozdělena do několika menších oddílů a každý z nich slouží pro jiný účel. V horní části se nachází předdefinovaný nadpis nového schématu, není ovšem problém přizpůsobit si jej vlastní potřebě jednoduchým kliknutím a přepsáním. Světlá plocha pod nadpisem tvoří hlavní část schématu. Tento prostor je určen pro vkládání vybraných komponent. Do objektu *Ventilated space*, který reprezentuje větraný prostor, mohou být přidány displeje pro zobrazování teploty v místnosti, příchozího vzduchu, atd.

Levý panel s názvem *Air ventilation control panel* slouží pro přidání ovládacích prvků ventilace, jako jsou tlačítka zapnout/vypnout ventilátor, boxy pro zadání požadované teploty/vlhkosti nebo různé kontrolky.

Do spodního *Simulation control panel* jsou taktéž přidávána tlačítka, kontrolky (apod.), která ovšem ovládají simulační model dané úlohy. Pro příklad zde mohou být tlačítka umožňující přehřátí ventilátorů, ucpání filtrů nebo boxy pro změnu venkovní teploty. Využití tohoto panelu najdeme při testování schématu nebo správnosti ovládacího algoritmu.



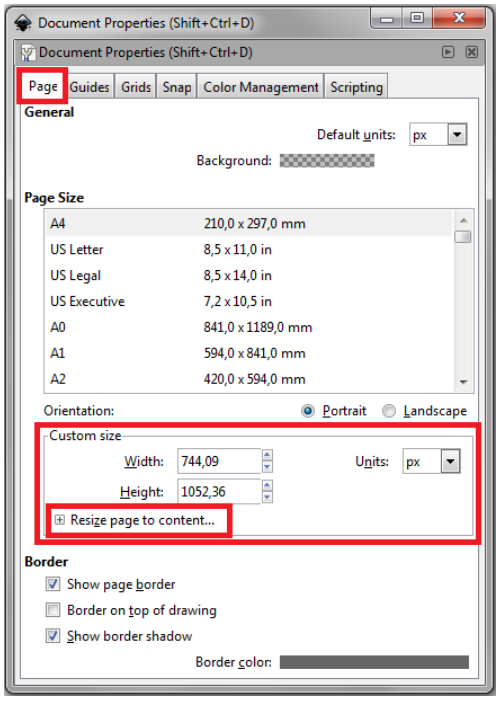
Obr. 5.1: Inkscape

Měřítko pracovní plochy

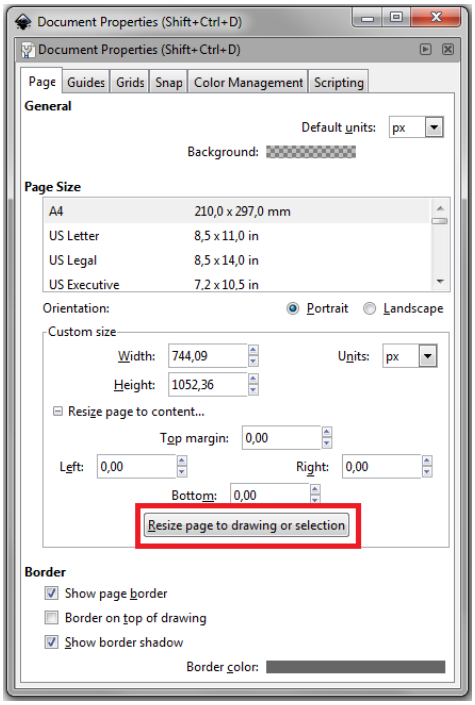
Jestliže potřebujeme větší užitnou plochu, ať už z důvodu velkého množství komponent nebo zvětšení jednotlivých komponent za účelem lepší přehlednosti, kliknutím a přidržením levého tlačítka myši označíme celé schéma a roztahováním objektu pomocí šipek v rohu

upravíme výslednou velikost. Pokud chceme zachovat všechny poměry stran držíme klávesu [Ctrl]. Pokud ne, a požadujeme například užší schéma (bez zachování poměru stran), provedeme úpravu bez klávesy [Ctrl].

Jestliže byla upravena velikost schématu, je třeba nastavit velikost zobrazované stránky. Tím budou nastaveny parametry atributu viewBox a vizualizace bude vždy roztažena přes celou plochu prohlížeče. Nejprve označíme celé schéma kromě objektu *Alias definiton* a použijeme zkratku [Ctrl+Shift+D] nebo vyhledáme volbu *File* → *Document Properties...*, vyskočí dialogové okno s panelem *Page* (obr. 5.2), kde v rámečku *Custom size* rozklikneme nabídku *Resize page to content...* a použijeme tlačítko *Resize page to drawing or selection* (obr. 5.3).



Obr. 5.2: Document Properties → Custom size



Obr. 5.3: Custom size → Resize

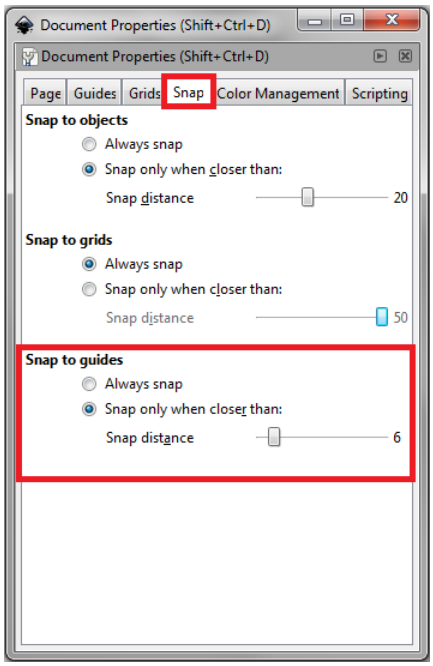
Zoom a pohyb pracovní plochy

Ovládání InkScape je velmi intuitivní, proto si představíme další ovládací prvky, které usnadní a zpříjemní tvorbu schémat. Posouvání plochy je možné díky bočnímu a spodnímu posuvníku ([Ctrl+B]) nebo pomocí klávesových šipek se stisknutým [Ctrl]. Nejsnazší varianta je skrolovací kolečko myši. Pokud ho používáme se klávesou [Shift] pohyb je horizontální, jinak vertikální.

K přiblížení či oddálení pracovní plochy slouží tlačítka + a -, box pro zadání přesné hodnoty ve *Status Baru* či skrolovací kolečko myši se stisknutou klávesou [Ctrl], kdy středem přibližování je vždy ukazatel myši.

Vodítka

Dobrým pomocníkem při umisťování komponent do volné plochy jsou vodítka (guides), která se jednoduše vytahují z horního a levého pravítka. Buď je umístíme podle potřeby myší nebo po dvojkliku nastavíme v dialogovém okně přesnou polohu. V panelu Snap Bar musí být aktivní první ikona *Enable snapping*, druhá *Snap bounding box corners*, čtvrtá *Snap to bounding box corners* (přitahování na rohy objektu) a ve spodní části *Snap to guides*. V document properties (`[Shift+Ctrl+D]`), v kartě Snap (obr. 5.4), lze upravit citlivost přichytávání podle vlastní potřeby. Pokud přichytávání překáží v práci, dá se jednoduše vypnout ikonou *Enable snapping* a jestliže vodítka dále nebudou třeba, stačí je přesunout zpět do pravítek.



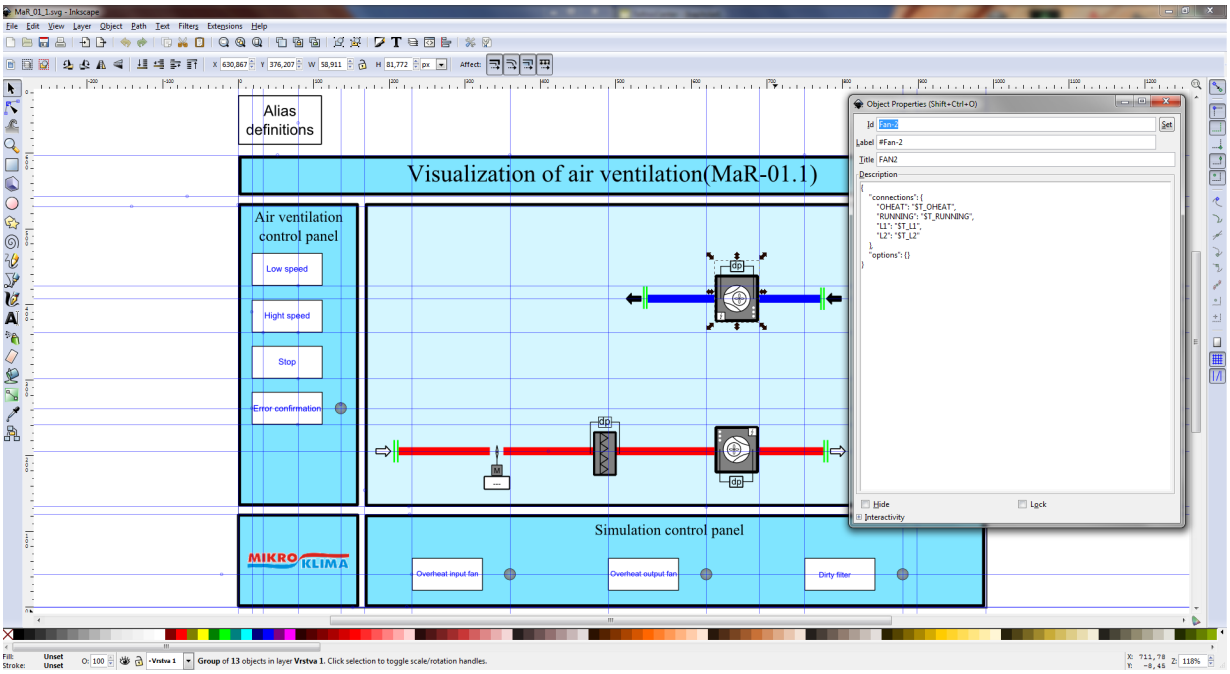
Obr. 5.4: Karta Snap v Document Properties

5.2 Vkládání a úprava komponent

Vložení komponenty se provádí pomocí metody *drag and drop*. Vedle okna programu InkScape otevřeme adresář s uloženými komponentami, uchopíme vybraný soubor s příponou .svg a přetáhneme jej do pracovní plochy editoru InkScape. Po té co stejným způsobem přidáme všechny potřebné komponenty, bude vhodné je rozmístit do podoby určené technickým výkresem. Doporučuji využít výše představená vodítka, udělat si jakousi mřížku a komponenty poté přichycovat na svá místa.

Transformace

Klasickou změnu měřítka jsme si ukázali při změně pracovního plátna. Pozicování komponent nám InkScape velmi usnadňuje s použitím předdefinovaných zkratek. Posunutí lze ovládat pomocí `šipek`, změnu měřítka mění klávesy `<, >` a rotaci provádí `[, ]`. Defaultní hodnoty pro změnu posunutí a měřítka jsou 2 pixely. Se stisknutým `[Shiftem]` se tato hodnota zvětší 10x. Otočení komponenty je definováno po 15°, avšak se stisknutým `[Ctrl]` bude objekt otočen o 90°.



Obr. 5.5: Připravené schéma s využitím vodítek

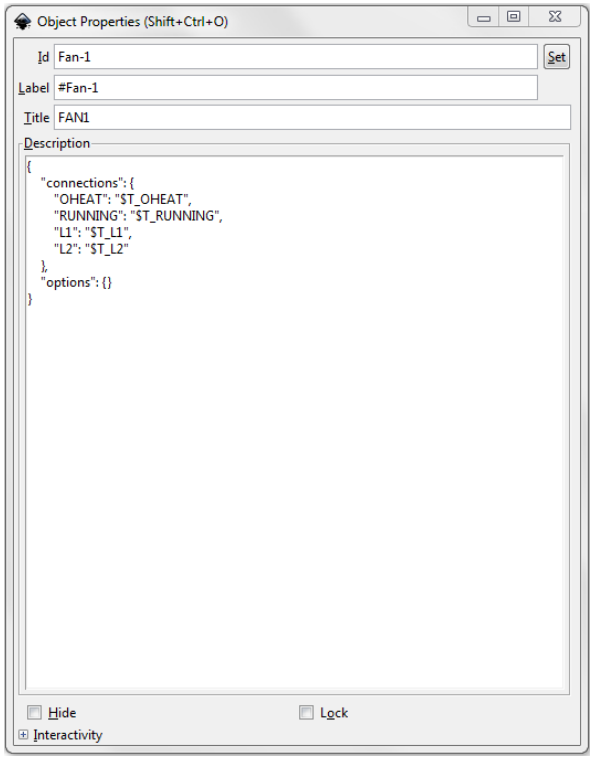
Velmi zajímavou funkci tvoří zkratky s použitím klávesy `[Alt]`. Ta v kombinaci s posuvem, rotací či změnou měřítka transformuje objekt o jeden pixel aktuálního zoomu. Při větším přiblížení bude pohyb na obrazovce vypadat jako změna o jeden pixel, avšak absolutní hnutí bude menší. To znamená, že při větším přiblížení lze docílit vyšší přesnosti transformací.

Duplikace

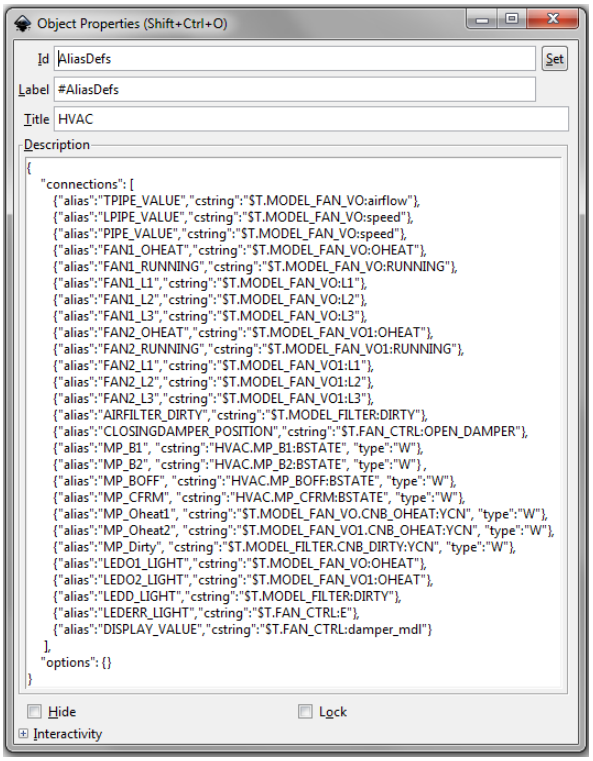
Jestliže se ve schématu objevuje více stejných komponent, lze je přidávat metodou *drag and drop* nebo lze využít známou kombinaci `[Ctrl+C]` a `[Ctrl+V]`. Nová komponenta se objeví v místě ukazatele. Inkscape má implementovanou duplikační funkci pod zkratkou `[Ctrl+D]`, kdy se komponenta objeví přímo nad originálem. Nicméně tato funkce přepisuje *id* komponenty, které je potřeba pro běh kódu v JavaScriptu, a nelze ji používat. Komponenta Pipe (potrubí) je vyjímka a nejlepší volbou je přetahování metodou *drag and drop* nebo použití duplikační funkce ([Ctrl+D]).

5.3 Připojení komponent k modelu

Ve chvíli, kdy jsme spokojeni s vizuální podobou schématu, přichází na řadu zapojení přípojných řetězců pro komunikaci prohlížeče s řídícím systémem REX.



Obr. 5.6: Komponenta Fan



Obr. 5.7: Objekt Alias definitions

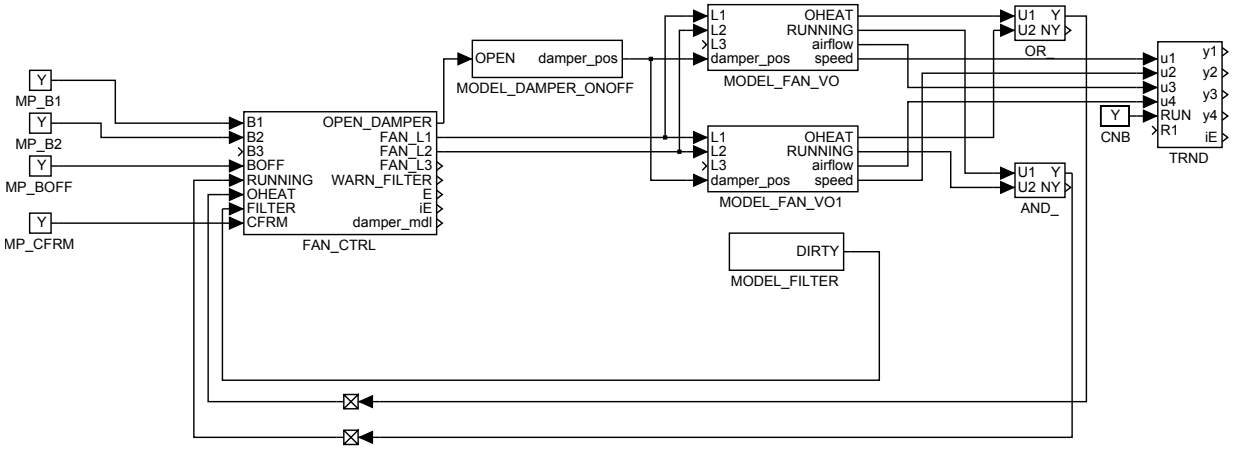
Nejprve se podíváme na připojení a nastavení komponent. Dialogové okno (obr. 5.6), dovolující úpravu elementu description, vyvoláme pravým kliknutím myši na komponentu a volbou *Object Properties* nebo označením komponenty a využití zkratky `[Ctrl+Shift+O]`. Plugin do editoru InkScape, který má nahradit toto dialogové okno a usnadnit tak zadávání přípojných řetězců, aliasů a volitelných informací, nebyl v době psaní této práce dokončen. Musíme se tedy spokojit se zadáváním údajů v jazyku JSON a jeho případnou validací pomocí JSONLint. O formátu JSON se více dozvíte v kapitole 1.2.3.

Definice Aliasu

Element description je u komponent předdefinovaný a tak není třeba velkých zásahů. V případě vícero komponent stejného druhu musí být změněn titulek (`Title:FAN1` ⇒ `Title:FAN2`) nebo mohou být doplněny či změněny volby v objektu *options* (např. barva kontrolky). Po každé provedené změně musí být použito tlačítko `Set`, jinak se tyto změny neuloží a celou úpravu bude třeba opakovat.

Definice přípojného řetězce

V dalším kroku se podíváme na komponentu *Alias definitions* (obr. 5.7). V té nás zajímá pouze objekt connections, kde jsou k určeným aliasům přiřazeny přípojné body, tedy řetězec, určující přesnou cestu k datům v modelovém schématu řídícího systému REX. Přípojné řetězce musí být vypsány uživatelem ručně podle daného schématu.



Obr. 5.8: Schéma v programu REX

Na obrázku 5.8 je schéma, ze kterého jsou vypsány přípojné řetězce v komponentě Alias definition, zobrazené na obrázku 5.7. Jako první slovo řetězce je HVAC, což označuje název exekutivy. Následuje tečka a název subsystému ze schématu (`MODEL_FAN_VO` = model ventilátoru). Dále chceme vybrat parametr `OHEAT` subsystému `MODEL_FAN_VO`. Místo tečky se použije dvojtečka. Touto adresou se připojíme k proměnné, informující o přehřátí ventilátoru. Celý přípojný řetězec bude vypadat následovně:

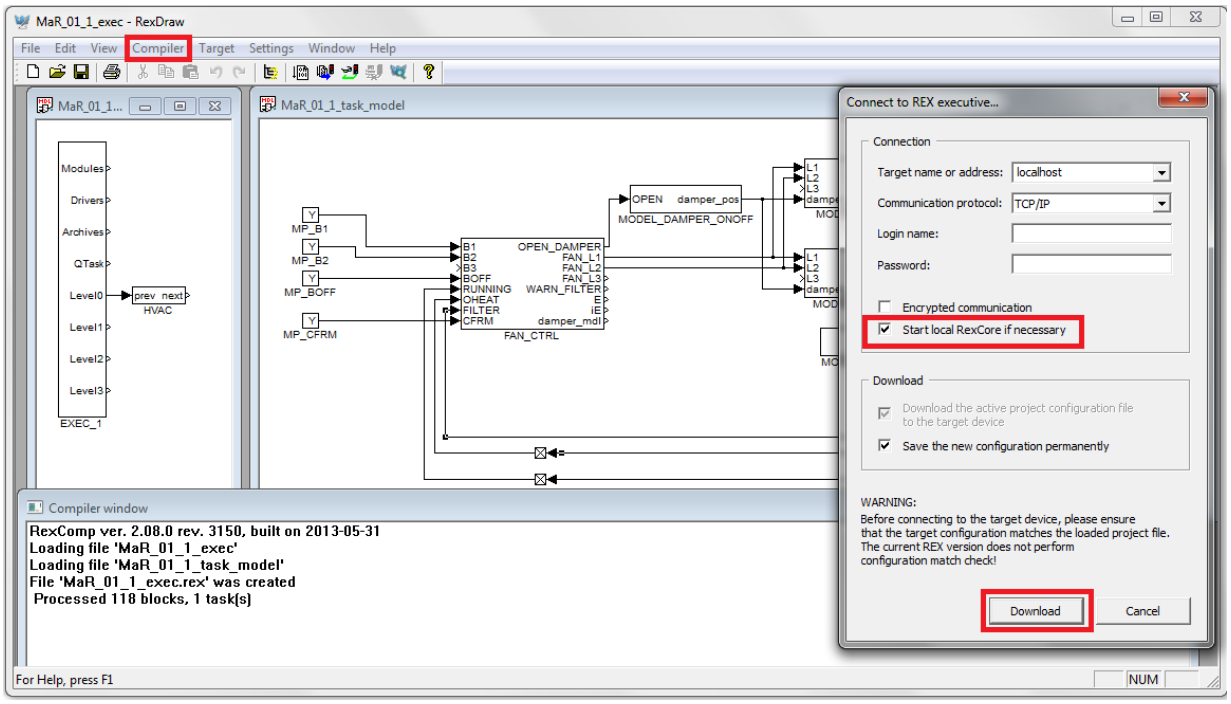
```
HVAC.MODEL_FAN_VO:OHEAT
```

5.4 Testování na vývojovém PC

Jakmile máme veškeré úpravy dokončené a uložené, bylo by vhodné nové schéma otestovat. Ověřit správnost připojení, vyzkoušet ovládání a zvážit jestli vizualizace poskytuje veškeré potřebné informace.

Předpokládám, že na vývojovém PC bude nainstalovaný REX. Z nabídky *Start* → *Programy* → *REX Controls* → *aktuálníVerze REXu* → *Nástroje* → spustíme aplikaci *HMI services*. Tímto se spustí webový server *Lighttpd* a WebSocket server *RexWSTcp*, které zprostředkují komunikaci mezi vizualizací v prohlížeči a běžící exekutivou v programu REX.

Každý model je tvořen dvěma soubory s příponou `.mdl`: `MaR_01_1_exec(a task_model)`, tedy názvem a jeho očíslováním. Otevřeme první soubor v programu RexDraw (více o systému REX viz. kapitola 1.6) a provedeme jeho překlad: *Compiler* → *Compile and Down-*



Obr. 5.9: Připojení exekutivy

load. Objeví se dialogové okno *Connect to REX executive...* (obr. 5.9), ve kterém zaškrtneme pole *Start local RexCore if necessary* a potvrdíme tlačítkem *Download*. Tímto proběhne nahrání exekutivy. Pokud vše proběhlo v pořádku objeví se okno *Download complete*, zde potvrdíme sledování algoritmu tlačítkem *Watch*. Nyní stačí otevřít patřičné schéma v prohlížeči a to začne komunikovat s běžící exekutivou. K odladění případných chyb se používá debugger, jehož použití je popsáno v kapitole 3.1.6.

Závěr

Vytvořené uživatelské prostředí, prezentované v této práci, je součástí projektu, na kterém se podíleli pracovníci katedry kybernetiky, jmenovitě: Ing. Jaroslav Sobota Ph.D., Ing. Ondřej Severa, Ing. Roman Pišl, Bc. Lubomír Kristek a další.

Hlavním přínosem je vytvořená knihovna komponent, ze které se dále vytvářejí schémata pro vizualizaci klimatizačních a otopných systémů. Většina požadavků kladených na komponenty byla splněna, nicméně knihovnu prozatím správně zobrazuje pouze webový prohlížeč Firefox od Mozilly. Další požadovaný prohlížeč Chrome od Google nemá tak rozsáhlou implementaci SVG specifikace. Proto je třeba počkat na vývoj prohlížeče nebo zvážit jiný způsob zadávání hodnot.

Připojování komponent se provádí pomocí tzv. přípojných řetězců, které jsou zapsány ve formátu JSON. Tento formát postačuje při vývoji rozhraní, avšak pro uživatele není jeho zpracování optimální, a to především kvůli složitému a k chybám náchylného zápisu. Pro budoucí použití je paralelně vytvářen plugin do vektorového editoru Inkscape, který poskytne přehledné GUI rozhraní a přejde se z používaného JSON formátu na značkovací jazyk XML.

V práci představené komponenty postačují k tvorbě základních vizualizačních schémat. Pro rozšíření použití je nutností pokračovat ve vývoji dalších tak, aby vznikla komplexní knihovna. Celý projekt bude nasazen v praxi, což povede k dalšímu vývoji a optimalizaci.

Jako velký přínos hodnotím vypracovaný tutoriál, který se zaměřuje na koncového uživatele. V tomto případě má tutoriál sloužit k tomu, aby uživatel rychle a snadno pochopil a ovládl použití programu Inkscape, práci s komponentami a tvorbu vizualizačních schémat.

Literatura

[1] FROST, J.; GOESSNER, S.; HIRTZLER, M.: *Learn SVG/The Web Graphics Standard*. Learn SVG:Web Graphics, 2003, ISBN 978-80-9741-7730-4.

[2] HTML tutorial. [Online], [cit.2013-07-16].
URL <http://www.w3schools.com/html/html_intro.asp>

[3] HTML5. [Online], [cit.2013-07-16].
URL <http://www.w3.org/TR/html5/>

[4] The Extensible HyperText Markup Language. [Online], [cit.2013-07-16].
URL <http://www.w3.org/TR/html/#html4>

[5] INKSCAPE. [Online], [cit.2013-07-22].
URL <http://inkscape.org/index.php?lang=cs>

[6] jQuery Learning Center. [Online], [cit.2013-07-19].
URL <http://learn.jquery.com/>

[7] jQuery Tutorial. [Online], [cit.2013-07-19].
URL <http://www.w3schools.com/jquery/default.asp>

[8] Úvod do JSON. [Online], [cit.2013-07-17].
URL <http://www.json.org/>

[9] LUBBERS, P.; ALBERS, B.; SALIM, F.: *HTML5/Programujeme moderní webové aplikace*. Brno: Computer Press, a.s., 2011, ISBN 978-80-251-3539-6.

[10] Začínáme se systémem REX. [Online], [cit.2013-07-23].
URL <http://www.rexcontrols.cz/media/documents/manuals/cz/REX_Getting_Started_CZ.pdf>

[11] SUEHRING, S.: *JAVASCRIPT/Krok za krokem*. Brno: Computer Press, a.s., 2008, ISBN 978-80-251-2241-9.

[12] Scalable Vector Graphics 1.1 (Second Edition). [Online], [cit.2013-07-22].
URL <http://www.w3.org/TR/SVG>

[13] TZB-info. [Online], [cit.2013-08-18].
URL <http://www.tzb-info.cz/>

[14] WebSocket.org. [Online], [cit.2013-08-13].
URL <http://www.websocket.org/>

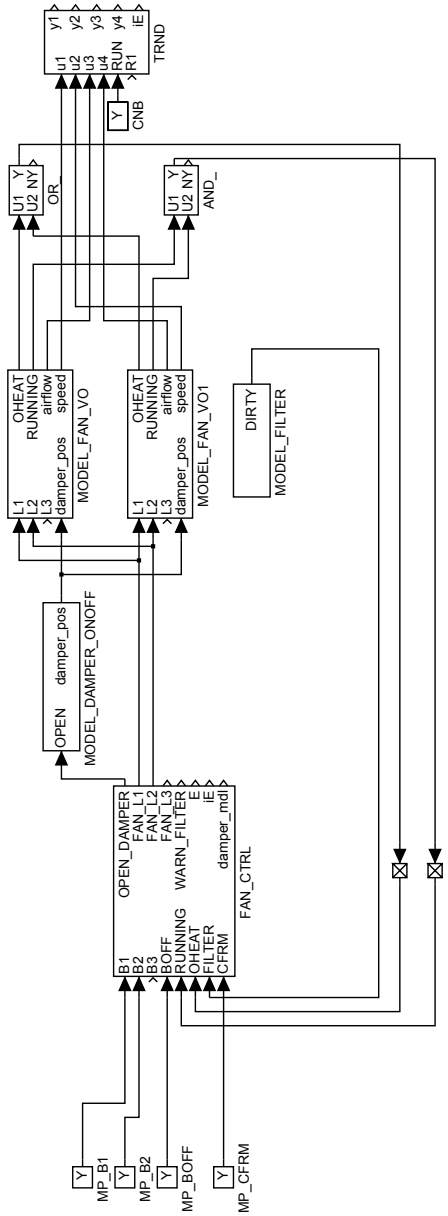
[15] XML Tutorial. [Online], [cit.2013-07-19].
URL <http://www.w3schools.com/xml/default.asp>

A Ukázka zdrojového kódu

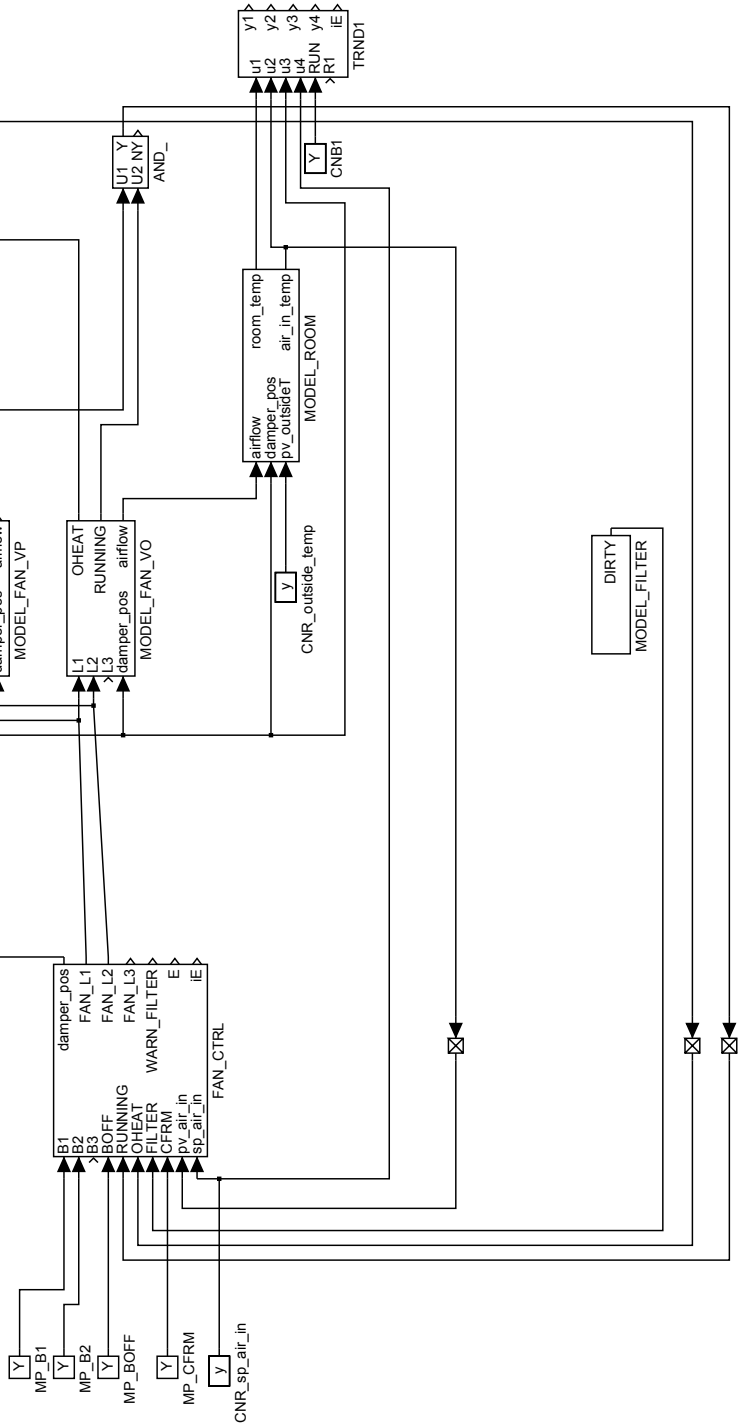
```
/**
 * SVG component represents Air filter.
 * @param {SVGElement} svgElem
 * @returns {REX.UI.SVG.Fan} New SVG Fan component
 */
REX.UI.SVG.AirFilter = function(svgElem,args) {
    // Inherit from base component
    var that = Object.create(REX.UI.SVG.Component(svgElem,args));
    // Store options for simple usage
    var $o = that.options || {};

    // Load options or default values
    var on_color = $o.on_color || '#00ff00';
    var error_color = $o.off_color || '#ff0000';
    var warning_color = $o.warning_color || '#ffff00';

    // Get SVG elements for manipulation
    var ofuse = that.getChild("upperRect"),
        ofilter = that.getChild("mainRect");

    // Add anonymous function as event listener. There are two events
    // 'read' - it is called every time when item is read
    // 'change' - called for the first time and every time item value is changed
    that.$c.DIRTY.on('change',function (i){
      if(i.getValue()){
        ofuse.style.fill = warning_color;
        ofilter.style.fill = warning_color;
      }else{
        ofuse.style.fill = "white";
        ofilter.style.fill = "white";
      }
      });

      return that;
     };
```

Obr. A.1: Ukázka JavaScriptového kódu – komponenta AirFilter

B Schéma v programu REX



Obr. B.1: MaR_01_1_task_model



Obr. B.2: MaR_01_2_task_model

C Seznam zkratek

Zkratka	*Funkce*
Ctrl+N	nový dokument
Ctrl+O	otevřít dokument
Ctrl+S	uložit dokument
Shift+Ctrl+S	uložit dokument jako...
Shift+Ctrl+D	Document Properties...
Shift+Ctrl+O	Object Properties
Ctrl+B	objevit/zmizet posuvník
+/-	přiblížení/oddálení
Ctrl+šipky	pohyb plochy
Ctrl+C	kopírovat
Ctrl+V	vložit

Tabulka C.1: Používané zkratky v editoru Inkscape