Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Plzeň, 2014

David Smolík

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 literatury a pramenů, jejíž úplný seznam je její součástí.

V Plzni dne: ….....

vlastnoruční podpis

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Pavlu Baldovi, Ph.D. za trpělivost a vstřícnost a Ing. Ondřeji Severovi za možnost konzultací, odborné rady a věnovaný čas.

Anotace

Tato práce se zabývá problematikou vytváření operátorského rozhraní pro modely sestavené v řídicím systému REX, formou webové vizualizace. Cílem je vytvořit knihovnu grafických komponent, které lze snadno editovat a lze je jednoduše vkládat do webové stránky. Práce je zaměřena na technologie a postupy, využívané při tvorbě interaktivních grafických komponent. Jedná se zejména o technologie, jež jsou součástí specifikace HTML5. Porovnává již existující možnosti vytvoření vizualizace s vlastním řešením a vysvětluje základní principy. Analyzuje výhody a nevýhody vlastního řešení. Součástí je aplikace výsledků na teoretickém příkladu a testování funkčnosti v nejrozšířenějších webových prohlížečích.

Klíčová slova: HMI, SCADA, REX, Inkscape, HTML5, SVG, JavaScript, JSON, WebSocket

Abstract

This thesis deals with the issue of creating an operator interface in the form of a website, for models that have been compiled in REX control system. Task is to create a library of graphical components, that can be edited easily and can be inserted to web page simply. The work is focused on technologies and procedures that are used in the creation of graphical components. Especially discussed about technologies that are a part of HTML5 specification. The work compares existing possibiblities to create a visualization with own solution, explains basic principles and analyzes advantages and disadvanteges of a custom solution. Part of this work is an application of results to theoretical example and functional testing in the most common browsers.

Keywords: HMI, SCADA, REX, Inkscape, HTML5, SVG, JavaScript, JSON, WebSocket

Obsah

1 Úvod

Důležitým aspektem v oblasti řízení je zpětná vazba, neboli zhodnocování získaných výsledků a následná reakce na ně. Zásadní roli v této oblasti tedy hraje, mimo jiné, interakce mezi uživatelem a systémem. Jednou z možností interakce „uživatel-systém“ je operátorské rozhraní, v angličtině známé jako „Human-Machine Interface“. Ve své bakalářské práci se zabývám problematikou vytváření operátorského rozhraní řídicích systémů formou webové vizualizace. Cílem je navrhnout sadu interaktivních grafických komponent, u kterých lze snadno editovat jejich základní parametry, a které lze jednoduše vkládat do webové stránky. Tyto komponenty umožní rychlé a snadné vytvoření elementárního operátorského rozhraní, přičemž některé prvky plní informační funkci a jiné slouží pro ovládání. Implementace v HTML zajišťuje nezávislost na platformě, díky tomu bude možné vizualizaci spustit v jakémkoli zařízení s webovým prohlížečem, který podporuje specifikaci HTML5.

Tato práce je zaměřena na technologie a postupy, využívané při tvorbě interaktivních grafických komponent, jež bude možné následně použít pro webovou vizualizaci k modelům, vytvořeným v řídicím systému REX. Porovnává již existující možnosti s vlastním řešením, vysvětluje základní principy a analyzuje výhody a nevýhody vlastního řešení. Součástí je aplikace výsledků na teoretickém příkladu a testování funkčnosti v nejrozšířenějších webových prohlížečích.

2 Současná situace

V současné době již existuje několik možností, jak pro modely řídicího systému REX vytvořit vizualizaci. Jednou z nich je použití komerčních SCADA/HMI systémů, to je umožněno díky podpoře průmyslového standardu OPC (OLE for Process Control). OPC je několik specifikací definujících komunikaci mezi klientem a serverem nebo mezi dvěma servery a zajišťuje přístup k datům v reálném čase.

Další možností je implementace v programovacím jazyce Java, který je řídicím systémem REX také podporován. Následně lze vytvořit virtuální laboratoř vložením již hotové aplikace do webové stránky, čímž je zajištěn přístup odkudkoliv.

2.1 SCADA/HMI

SCADA je zkratka pro „Supervisory Control and Data Acquisition“, viz [1]. Jde o systém shromažďující data z čidel v reálném čase, pracující nad reálným řídicím systémem. Získaná data jsou posílána na centrální počítač, kde jsou zpracována pro další řízení. Obsahuje vstupně-výstupní hardware, regulátory, HMI, sítě, komunikace, databáze.

HMI je zkratka pro „Human Machine Interface“, neboli rozhraní mezi člověkem a strojem. Jedná se o grafické uživatelské prostředí s řadou vizualizačních komponent, zobrazující informace o stavu řízeného procesu a umožňující zadávat operátorské příkazy.

Mezi typické SCADA/HMI systémy patří například Reliance, InTouch, ControlWeb, Promotic, Indusoft, Labview.

2.2 Virtuální laboratoř

Jedná se o webovou aplikaci s možností vzdáleného přistupu. Aplikace je vytvořena na míru konkrétnímu problému a je vhodná spíše k propagačním a prezentačním účelům.

2.3 Výhody vlastního řešení

Hlavní výhodou oproti SCADA/HMI systémům je plná funkčnost bez nutnosti zakoupení licence. Není třeba kupovat placené nástroje, všechny potřebné nástroje pro vytvoření vizualizace jsou zdarma, volně dostupné. Další výhodou je „otevřenost“ zdrojových souborů (open-source), vizualizační komponetny lze tedy rozšiřovat a přizpůsobovat vlastním potřebám.

Výhodou oproti virtuálním laboratořím je snadné a rychlé vytvoření webové vizualizace bez nutnosti kódování, jednoduché vkládání hotových komponent a manipulace s nimi.

3 Použité technologie a nástroje

3.1 HTML5

Jde o pátou, zatím poslední, generaci značkovacího jazyka HTML. V roce 2007 byl zahájen vývoj této specifikace, která doposud nebyla oficiálně vydána. Vzniká ve spolupráci konsorcia „World Wide Web Consortium“ (zkráceně „W3C“) a „WHATWG“, viz [2]. HTML5 je založeno na technologiích HTML, CSS, DOM a JavaScript.

3.1.1 Obecně o HTML

HTML je zkratka pro „HyperText Markup Language“. Jde o jazyk vhodný pro vytváření a popis webových stránek, jehož první verze vyšla v roce 1991. Je nazýván jako značkovací jazyk, viz [5].

Základním prvkem jsou elementy (tagy), které určují obsah webových stránek. Elementy jsou ve většině případů párové, jsou tvořeny počáteční a ukončovací značkou. Píší se do „špičatých“ závorek, přičemž ukončovací značka obsahuje lomítko za otevírací závorkou. Obecně to vypadá takto: *<jmenoTagu>Tělo elementu</jmenoTagu>*. Tělo elementu může být tvořeno vlastním textem nebo dalšími vnořenými elementy. Pokud element neobsahuje žádné tělo, nemusí se psát ukončovací značka, stačí napsat počáteční značku s lomítkem před uzavírací závorkou, obecně: *<jmenoTagu/>*. Jednotlivé elementy mohou mít ještě rozšiřující vlastnosti, tzv. atributy. Atributů může mít element několik a píší se do počáteční značky elementu, obecně: *<jmenoTagu atribut1=hodnota1 atribut2=hodnota2 ... >*.

HTML disponuje danou strukturou, která se musí dodržet. Každý HTML kód musí obsahovat kořenový element *<html>* a element *<body>*, který je vnořen do kořenového elementu a tvoří tělo dokumentu.

```
<html>
  <body>
    <p>Prvni odstavec</p>
    <p>Druhy odstavec</p>
  </body>
</html>
```

Příklad 1 – Ukázka struktury HTML kódu

3.1.2 Nové vlastnosti HTML5

Výhodou oproti starším verzím je omezení potřeby externích modulů (tzv. „pluginů“), které mají vlastní jazyk a je potřeba mít v prohlížeči nainstalovanou aktuální verzi (typickým externím pluginem je Flash). Je vylepšena správa chyb, upřednostňuje se značkování pomocí elementů oproti skriptování v JavaScriptu a měla by být zajištěna nezávislost na zařízení.

3.1.3 Nové elementy HTML5

Vznikla celá řada nových elementů, viz [5]. Zde je výčet těch nejvýznamějších.

Element pro vykreslování bitmapové grafiky ve 2D:

- <*canvas*> – obdélníková plocha, ve které lze vykreslit každý pixel, k vykreslení se používá JavaScript

Elementy pro lepší strukturalizaci stránky:

- <*article*> – definuje článek
- <*header*> – definuje záhlaví dokumentu nebo jeho části
- <*footer*> – definuje zápatí dokumentu nebo jeho části
- <*section*> – definuje určitou část dokumentu
- <*nav*> – definuje odkazy v navigační liště
- <*aside*> – definuje obsah v postranní části dokumentu

Elementy pro přehrávání médií:

- <*audio*> – slouží pro přímé vkládání audia do dokumentu bez nutnosti pluginu pro přehrávání, v současnosti jsou podporovány tři formáty: MP3, Ogg a Wav, přičemž každý prohlížeč může podporovat jiný formát
- <*video*> – slouží pro přímé vkládání videa do dokumentu bez nutnosti pluginu pro přehrávání, stejně jako u audia jsou v současnosti podporovány tři formáty: MP4, Ogg a WebM, opět není úplná podpora ve všech prohlížečích

3.1.4 HTML5 APIs

Jedná se o řadu rozšiřujících funkcí, viz [4]:

- *Geolocation* – funkce umožňující určit geografickou polohu uživatele, pokud je to uživatelem povoleno

- *Drag & Drop* – funkce umožňující uchopení objektu a jeho přetáhnutí na jiné místo, přičemž tato funkce může být aplikována na jakýkoli element, rozšiřující je možnost uchopit soubor v adresáři počítače a přetáhnout ho do webového prohlížeče
- *Web Storage* – funkce vylepšující lokální ukládání dat, nahrazuje současné „cookies“, data lze ukládat na neomezenou dobu (pomocí objektu *localStorage*) nebo jen na určitou dobu, dokud je aktuální záložka otevřená (pomocí objektu *sessionStorage*)
- *App Cache* – umožňuje uživateli používat webovou aplikaci v offline režimu s rychlejším načtením dat
- *Web Worker* – funkce umožňující vytvořit JavaScript běžící v pozadí, který ničím nezdržuje načtení (běh) stránky
- *SSE* – „Server-Sent Event“, umožňuje webové stránce automaticky získávat aktualizace ze serveru (ve starších specifikacích HTML se musela webová stránka dotazovat serveru, zda jsou nějaké aktualizace dostupné)
- *WebSocket* – technologie poskytující obousměrnou komunikaci mezi klientem (webovým prohlížečem) a serverem v protokolu TCP/IP, umožňuje výměnu informací v reálném čase.
- *WebGL* – knihovna se schopností vytvářet interaktivní 3D grafiku bez použití pluginů, je součástí elementu <*canvas*>

3.2 SVG

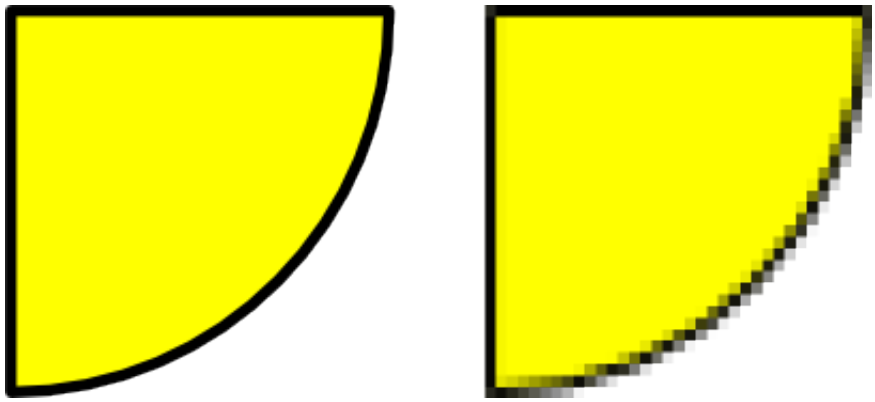
SVG je zkratka pro „Scalable Vector Graphics“ neboli škálovatelnou vektorovou grafiku. Vývoj této technologie začal už v roce 1999, zasloužilo se o to konsorcium „W3C“. Jedná se o značkovací jazyk založený na formátu XML a slouží pro vykreslení 2D vektorové grafiky. Jak je uvedeno v [6], SVG je využíváno především ve webových aplikacích jako interaktivní prvek, který neztrácí kvalitu při změně velikosti nebo přiblížení na rozdíl od bitmapové (rastrové) grafiky. HTML5 navíc umožňuje přímé vkládání této grafiky do HTML kódu pomocí elementu <*svg*>. Díky struktuře XML lze objekty dynamicky upravovat (například pomocí JavaScriptu), každý element a atribut v SVG dokumentu může být animován. Existuje celá řada grafických editorů, které umí ukládat obrázky do formátu SVG. Je to například „CorelDraw“, „Amaya“ od konsorcia „W3C“ nebo volně dostupný a hojně rozšířený „InkScape“.

3.2.1 Rozdíl bitmapové a vektorové grafiky

Bitmapová (rastrová) a vektorová grafika jsou dvě základní techniky, které se používají pro ukládání a zpracování obrazových dat.

U bitmapové grafiky je dána mřížka, která převádí grafický objekt na čtvercovou síť. Jednotlivé čtverce se nazývají pixely (obrazové body), přičemž každý pixel disponuje přesně danou polohou a barvou (někdy se definuje i jas). Jak se uvádí v [6], pokud jde o černobílý obrázek, každý pixel nese informaci o velikosti jednoho bitu. U obrázků, které jsou popsány pomocí barev RGB se datový objem jednoho pixelu zvětšuje s celkovým počtem použitých odstínů a tónů. Tato technika je využívána především u digitálních zařízení jako je fotoaparát, kamera, skener apod., tedy tam kde je obtížné popsat daný objekt známými tvary, tudíž nelze aplikovat vektorový popis. Typickými formáty bitmapové grafiky jsou JPEG, GIF, TIFF nebo PNG.

Vektorová grafika je založená na matematickém popisu. Grafický objekt je složen ze základních geometrických tvarů jako je kružnice nebo mnohoúhelník, jejichž základem je křivka nebo rovná čára. Křivka je v tomto případě definována minimálně počátečním bodem, vektorem, který určuje směr a koncovým bodem, dále se udává tloušťka a barva čáry, viz [6]. Křivka může být otevřená, uzavřená, s výplní nebo bez výplně. Při úpravě velikosti se pracuje pouze s křivkami, které určují tvar objektu (u bitmapové grafiky se pracuje se všemi obrazovými body). Tato technika se využívá zejména pro tvorbu ikon, log, diagramů nebo počítačových animací, viz [6]. Typickými formáty vektorové grafiky jsou CDR, SVG, EPS, PDF (metaformát).



a) **b)**

Obrázek 1 – Ukázka přiblížení vektorové (a) a bitmapové (b) grafiky

Výhodou bitmapové grafiky je velké rozšíření a podpora, dostačující zachycení reality (fotografie) a možnost použití efektů. Nevýhodou je vyšší velikost souboru a snižování kvality při zvětšování obrázku, viz [6].

Výhodou vektorové grafiky je nižší velikost souboru a zachování kvality při zvětšování obrázku. Nevýhodou je vyšší náročnost na paměť a na procesor u složitějších grafických objektů, viz [6].

3.2.2 Struktura SVG dokumentu

Díky čistému XML formátu může být jednodušší SVG vytvořeno v jakémkoli textovém editoru. Pro složitější případy je vhodné použít některý z grafických editorů.

První řádka SVG dokumentu obsahuje XML deklaraci s atributem *standalone*, který říká zda je SVG samostatné nebo zahrnuje referenci na externí soubor. V další řádce jsou definovány případné reference. Následuje kořenový element <*svg*>, který je v dokumentu jediný. V kořenovém elementu se definuje namespace a případně atributy jako *width* nebo *height,* které určují výšku a šířku zobrazované plochy SVG dokumentu. Pokud element <*svg*> nemá tyto atributy definovány, obrázek se přizpůsobí velikosti okna prohlížeče. Element <*svg*> dále obsahuje vnořené elementy, které již slouží pro vykreslení vlastních objektů.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2"
  fill="red" />
</svg>
```

Příklad 2 – Ukázka struktury SVG dokumentu s referencí na externí soubor, výsledkem je kruh

3.2.3 Způsob vkládání SVG do HTML stránky

Existuje několik způsobů jak vložit SVG do webové stránky, viz [7], přičemž každý z nich se vyznačuje specifickými vlastnostmi.

- vložení pomocí <*embed*> elementu
- vložení pomocí <*object*> elementu

- vložení pomocí <*iframe*> elementu
- přímé vložení elementu <*svg*> do HTML kódu
- využití odkazu pomocí elementu <*a*>

3.2.4 Vložení do HTML stránky pomocí <embed> elementu

Výhodou této metody je podpora všech prohlížečů a povolení ve specifikaci HTML5. Není však schválena ve straších specifikacích HTML 4 a XHTML, viz [7].

```
<embed src="jmeno.svg" type="image/svg+xml" />
```

Příklad 3 – Vložení SVG do HTML pomocí elementu *<embed>*

3.2.5 Vložení do HTML stránky pomocí <object> elementu

Výhodou této metody je opět podpora všech prohlížečů a povolení ve specifikaci HTML 4, XHTML i HTML5. Neumožňuje však skriptování, viz [7].

```
<object data="jmeno.svg" type="image/svg+xml"></object>
```

Příklad 4 – Vložení SVG do HTML pomocí elementu *<object>*

3.2.6 Vložení do HTML stránky pomocí <iframe> elementu

Tuto metodu taktéž podporují všechny prohlížeče, viz [7]. Nevýhodou je vytvoření rámečku, nemožnost stylování a metoda není povolena ve straších specifikacích HTML 4 a XHTML.

```
<iframe src="jmeno.svg"></iframe>
```

Příklad 5– Vložení SVG do HTML pomocí elementu *<embed>*

3.2.7 Vložení do HTML stránky přímo

Tento způsob je povolen pouze ve specifikaci HTML5 a podporuje ho většina prohlížečů.

```
<html>
  <body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <circle cx="100" cy="100" r="50" stroke="black" stroke-width="2"
    fill="red"/>
  </svg>
  </body>
</html>
```

Příklad 6 – Přímé vložení SVG do HTML stránky

3.2.8 Odkaz na SVG soubor

Jedním ze způsobů je také odkaz na externí SVG.

```
<a href="jmeno.svg">Odkaz</a>
```

Příklad 7 – Odkaz na externí SVG

3.2.9 Základní tvary – obdélník

Obdélník je definován pomocí tagu <*rect*>. Mezi povinné atributy patří počáteční souřadnice *x, y* (levý horní roh), šířka *width* a výška *height*. Dalšími atributy mohou být například zaoblení rohů *rx, ry*, viz [7].

```
<rect x="20" y="20" width="100" height="40" rx="10" ry="10"/>
```

Příklad 8 – Vykreslení obdélníku

3.2.10 Základní tvary – kruh

Kruh je definován pomocí tagu <*circle*>. Mezi povinné atributy patří souřadnice středu c*x, cy* a poloměr *r,* viz [7].

```
<circle cx="50" cy="50" r="30"/>
```

Příklad 9– Vykreslení kruhu

3.2.11 Základní tvary – elipsa

Elipsa je definována pomocí tagu <*ellipse*>. Atributy, které musí být součástí jsou souřadnice středu c*x, cy* a radiusy *rx, ry*, viz [7].

```
<ellipse cx="100" cy="50" rx="80" ry="40"/>
```

Příklad 10 – Vykreslení elipsy

3.2.12 Základní tvary – čára

Čára (úsečka) je definována pomocí tagu <*line*>. Povinnými atributy jsou počáteční souřadnice *x1, y1* a koncové souřadnice *x2, y2*, viz [7].

```
<line x1="20" y1="20" x2="70" y2="70"/>
```

Příklad 11– Vykreslení úsečky

3.2.13 Základní tvary – polyline

Tvar složený z několika úseček, nemůže být vyplněn barvou. Je definován pomocí tagu <*polyline*>. Povinným atributem je atribut *points,* který obsahuje souřadnice vrcholů úseček, viz [7].

```
<polyline points="0,50 50,50 50,100 100,100"/>
```

Příklad 12– Vykreslení polyline úsečky

3.2.14 Základní tvary – mnohoúhelník

Mnohoúhelník je definován pomocí tagu <*polygon*>. Povinným atributem je atribut *points* (stejně jako u polyline)*,* který obsahuje souřadnice vrcholů úseček. Narozdíl od *polyline* může být vyplněn, viz [7].

```
<polygon points="150,20 280,120 180,250 111,225"/>
```

Příklad 13– Vykreslení mnohoúhelníku

3.2.15 Základní tvary – cesta

Pomocí cesty lze vytvořit libovolný tvar objektu, který může být vyplněn. Je definována pomocí tagu <*path*>. Důležitým atributem je atribut *d,* který určuje vlastní podobu cesty.

Existuje několik příkazů, které se do atributu *d* zapisují, viz [6]:

- M – moveto, přesun na dané souřadnice
- L – lineto, vykreslení úsečky z aktuální polohy k daným souřadnicím
- H – horizontal lineto, vykreslení úsečky horizontálně
- V – vertical lineto, vykreslení úsečky vertikálně
- C – curveto, kubická Bézierova křivka
- S – smooth curveto, kub. Béz. křivka vycházející z koncového bodu předchozí křivky
- Q – quadratic curve, kvadratická Bézierova křivka
- T – smooth quadratic curve, kvad. Bézierova křivka, podobně jako „S“
- A – elliptical arc, vykreslení oblouku z aktuální pozice k daným souřadnicím
- Z – close path, uzavření cesty

```
<path d="M 100 0 L 50 200 L 125 200 Z"/>
```

Příklad 14 – Vykreslení cesty

3.2.16 Tvorba textu

Text lze vytvořit pomocí tagu <*text*>. Povinným atributem jsou souřadnice *x, y,* viz [7].

```
<text x="50" y="50" fill="blue">Text</text>
```

Příklad 14– Vytvoření textu

3.2.17 Element <g>

Element sloužící pro seskupení několika objektů, přičemž následně je umožněno manipulovat s celou skupinou, viz [6].

```
<g>
  <rect x="20" y="20" width="100" height="40"/>
  <circle cx="20" cy="20" r="30"/>
  <line x1="20" y1="20" x2="70" y2="70"/>
</g>
```

Příklad 15 – Seskupování objektů

3.2.18 Element <defs>

Umožňuje předdefinovat objetky nebo vlastnosti, které mohou být posléze v dokumentu opakovaně využívány pomocí referencí. Objekty v tomoto elementu se tedy nevykreslují hned, ale je na ně odkazováno, viz [6].

```
<defs>
  <linearGradient id="gradient1">
    <stop  style="stop-color:#000000;stop-opacity:1;"
           offset="0"/>
    <stop  style="stop-color:#ffffff;stop-opacity:1;"
           offset="1"/>
  <linearGradient/>
</defs>
<circle cx="20" cy="20" r="30" style="fill:url(#gradient1)"/>
```

Příklad 16 – Ukázka použití elementu <defs> , <circle> obsahuje referenci na gradientní přechod

3.2.19 Element <use>

Tento element je využíván pro opakované vykreslování („klonování“) stejného objektu předdefinovaného v elementu <*defs*>, viz [6].

```
<defs>
  <rect id="obdelnik1" x="20" y="20" width="100" height="40"/>
</defs>
<use xlink:href="(#obdelnik1)" x="50" y="50"/>
```

Příklad 17 – Ukázka použití elementu <use> , obsahuje referenci na obdélník

3.2.20 Elementy <title> a <desc>

Tyto elementy slouží k popisu objektu pomocí textového řetězce. Každý objekt může mít svůj vlastní popis, přičemž popisující text není vykreslován. Element *<title>* by měl být vždy prvním „potomkem“ nadřazeného elementu, viz [6].

```
<g>
  <title>Obdélník</title>
  <desc>Obdélník s výškou 40 a šířkou 100.</desc>
  <rect x="20" y="20" width="100" height="40"/>
</g>
```

Příklad 18 – Ukázka použití elementu <title> a <desc>

3.3 JavaScript

JavaScript (standardizovaný jako ECMAScript) je objektově orientovaný skriptovací jazyk, jehož vývoj začala společnost „Netscape“ v roce 1995. Umožňuje vytvářet webové stránky s řadou interaktivních prvků jako jsou například tlačítka nebo textová pole, lze pomocí něj tvořit animace, viz [8]. Jedná se o „odlehčený“ programovací jazyk, jehož kód může být vložen přímo do HTML stránky.

JavaScript je interpretovaný jazyk, což znamená, že se spouští bez předchozího překladu (za běhu). Ke spuštění programu dochází až po stažení webové stránky klientem z internetu. Je odvozen od programovacích jazyků C, C++ a Java, viz [8].

Typickým využitím JavaScriptu je vytváření, mazání, kopírování a úprava HTML elementů. Pro přístup k HTML elementům slouží aplikační rozhraní DOM („Document Object Model“), které je jazykově nezávislé. DOM je vytvořen v paměti prohlížeče po načtení webové stránky, má stromovou strukturu a obsahuje prvky jako HTML elementy, atributy, texty, komentáře, příčemž každý prvek ve stromové struktuře může být nalezen a modifikován JavaScriptem.

3.3.1 Vložení do HTML stránky přímo

Pro přímé vložení JavaScriptu do HTML dokumentu se používá element *<script>,* který může být vnořen do elementu *<body>* nebo do elementu *<head>,* viz [9]. Počet skriptů v HTML dokumentu je libovolný.

```
<html>
  <body>
  <script>
    alert('Textová zpráva');
  </script>
    …
  </body>
</html>
```

Příklad 19 – Přímé vložení skriptu do HTML

3.3.2 Vložení do HTML stránky odkazem na externí skript

JavaScript může být uložen jako samostatný externí soubor s příponou .js a následně vložen do HTML dokumentu. Element *<script>* v tomto případě obsahuje atribut *src* s názvem externího skriptu. Umístění je obdobné jako u přímého vkládání, viz [9].

```
<html>
  <body>
  <script src="nazevSkriptu.js"></script>
    …
  </body>
</html>
```

Příklad 20 – Vložení externího skriptu do HTML

3.3.3 Syntaxe

JavaScript je posloupnost příkazů oddělených středníkem, přičemž prohlížeč vykonává jednotlivé příkazy postupně, podle pořadí v jakém jsou napsány. Příkazy mohou být seskupeny do bloku pomocí složených závorek, a poté provedeny zároveň, typickým příkladem seskupení jsou funkce, viz [9]. JavaScript rozlišuje malá a velká písmena, tedy například proměnná s názvem *prom* a jiná proměnná s názvem *Prom* jsou dvě různé proměnné. Bílé mezery jsou v kódu povoleny, jsou jím ignorovány.

Komentáře v kódu mohou být jednořádkové nebo víceřádkové. Pro jednořádkové komentáře se používá pouze počáteční značka // (dvě lomítka). Pro víceřádkové komentáře se jako počáteční značka používá /* a jako ukončovací značka */, viz [9].

```
<html>
<body>
  <p id="odstavec"></p>
  <script>
    function soucinCisel(a, b) {
         return a * b;
    }
    var s = soucinCisel(2, 5);
    document.getElementById("odstavec").innerHTML = s;
  </script>
</body>
</html>
```

Příklad 21 – Ukázka syntaxe a struktury JavaScriptu

3.3.4 Proměnné

Proměnné jsou v JavaScriptu deklarovány klíčovým slovem *var* a jejich datový typ lze libovolně měnit. Základními datovými typy jsou String, Number, Boolean, Array, Object, Null, Undefined, viz [9].

```
var x = "hodnota";
var y;
x = 5;
```

Příklad 22 – Proměnné v JavaScriptu

3.3.5 Funkce

Funkce jsou definovány klíčovým slovem *function* následuje vlastní jméno funkce a jednoduché závorky. V jednoduchých závorkách se případně uvádí parametry oddělené čárkou. Nakonec je vložen blok kódu ve složených závorkách. Příkazy funkce jsou prováděny pouze tehdy, pokud je funkce vyvolána (většinou nějakou událostí nebo jinou funkcí). Funkce mohou dále obsahovat návratovou hodnotu, která následuje za klíčovým slovem *return* (tato hodnota je vrácena na místo, kde byla funkce vyvolána).

```
function minimum (a,b){
  var min = 0;
  if (a < b) min = a;
  else min = b;
  return min;
}
```

Příklad 23 – Funkce v JavaScriptu, vrací minimum ze dvou hodnot

3.3.6 Události

Události (events) se dají popsat jako akce provedené uživatelem nebo samotným prohlížečem, viz [9]. JavaScript může na události nějakým způsobem reagovat. Příkladem události může být kliknutí na tlačítko nebo načtení stránky. Události se přidávají jako atributy a mohou být přiřazeny jakémukoli elementu v libovolném množství. Pokud daná událost nastane, dojde k vykonání námi definované činnosti.

```
<html>
<body>
  <button onclick="getElementById('p1').innerHTML='NovýObsah'">
  </button>
  <p id="p1">PůvodníObsah</p>
</body>
</html>
```

Příklad 24 – Události v JavaScriptu, kliknutím na tlačítko se změní obsah odstavce

Nejpoužívanějšími HTML událostmi jsou, viz [9]:

- *onclick* – kliknutí na element
- *onchange* – změna HTML elemntu
- *onmouseover* – myš se pohybuje nad elementem
- *onmouseout* – myš opustila pozici nad elementem
- *onkeydown* – stisknutí tlačítka na klávesnici
- *onload* – prohlížeč dokončil načtení webové stránky

3.3.7 Objekty

Téměř vše v JavaScriptu může být objektem. Objekty jsou data (proměnné) s přídavnými vlastnostmi a metodami. Lze vytvářet vlastní objekty.

```
var osoba = {
  jmeno: "Jan",
  prijmeni: "Novák",
  vek: 35,
  id: 7535,
  celeJmeno: function(){return this.jmeno + " " + this.prijmeni}
};
```

Příklad 25 – Vytvoření objektu osoba v JavaScriptu

Pro přístup k vlastnostem objektu lze použít více způsobů.

```
var krestniJmeno = osoba.jmeno;
var krestniJmeno = osoba["jmeno"];
var celeJmeno = osoba.celeJmeno();
```

Příklad 26 – Přístup k vlastnostem objektu v JavaScriptu

3.4 jQuery

jQuery je knihovna jazyka JavaScript, vyvíjená od roku 2006. Jedná se o „odlehčenou“ verzi JavaScriptu, jejíž hlavním účelem je zjednodušení kódování. Usnadňuje například manipulaci s objekty, které jsou součástí aplikačního rozhraní DOM. Umožňuje obsluhovat události, manipulovat s CSS, vytvářet různé efekty a animace, použít zásuvné moduly s doplňujícími funkcemi, viz [10].

3.4.1 Vložení knihovny do HTML

Existují dva způsoby, jak knihovnu připojit k webové stránce, viz [10]. Prvním způsobem je stažení knihovny[1] s příponou *.js* do vlastního úložiště a vložení do HTML dokumentu jako externí skript.

```
<script src="jquery-1.11.0.min.js"></scirpt>
```

Příklad 27 – Vložení stažené knihovny jQuery do HTML

Druhým způsobem je vložení knihovny z „Content Delivery Network“ (systém serverů sdílející svůj obsah). Hostitelem je například „Google“ nebo „Microsoft“

1 Dostupné z: http://jquery.com

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js">
</scirpt>
```

Příklad 28 – Vložení knihovny jQuery do HTML z Google CDN

```
<script
src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.0.min.js">
</scirpt>
```

Příklad 29 – Vložení knihovny jQuery do HTML z Microsoft CDN

3.4.2 Syntaxe

Syntaxe je přizpůzobena snadnému výběru HTML elementů a následné manipulaci s nimi. Pro výběr elementu se používá znak *$*, následovaný selektorem v „kulatých“ závorkách. Dále pokračuje „tečka“ a funkce (událost), aplikovaná na daný element, viz [10]. Selektorem mohou být elementy, atributy elementu (*id, class, type, ...*) nebo samotné hodnoty atributů.

```
$("button").click(funciton(){
  $("#idOdstavce").hide();
});
```

Příklad 30 – Syntaxe v jQuery

Nejpoužívanějšími jQuery událostmi jsou, viz [10]:

- *$(document).ready()* – prohlížeč dokončil načtení webové stránky
- *click()* – kliknutí na element
- *dblclick()* – dvojklik na element
- *mouseenter()* – myš se pohybuje nad elementem
- *mouseleave*() – myš opustila pozici nad elementem
- *mousedown()* – stisknutí levého tlačítka myši nad elementem
- *mouseup()* – uvolnění levého tlačítka myši nad elementem
- *hover()* – kombinace událostí *mouseenter()* a *mouseleave()*

3.5 WebSocket

WebSocket je technologie v rámci specifikace HTML5. Obecně se jedná o komunikační spojení mezi dvěma procesy, navázané pomocí IP technologie. Socket je popsán dvěma IP

adresami, mezi kterými probíhá komunikace, dále protokolem a portem. Tento socket bývá využíván k obousměrnému přenášení dat, viz [11].

Pomocí WebSocket technologie je možné vytvořit webovou aplikaci s obousměrnou komunikací mezi klientem (webovým prohlížečem) a serverem v protokolu TCP/IP, viz [11]. Díky této technologii je možná výměna informací v reálném čase. Ve starších specifikacích se klient musel dotazovat serveru, zda jsou dostupná nová data, což značně prodlužovalo dobu komunikace. Pomocí WebSocket technologie lze vytvářet „real-time“ aplikace, viz [11].

WebSocket vychází ze starších technologií, jako je například AJAX („asynchronní jednorázový požadavek od klienta na server v protokolu HTTP) nebo COMET („dlouhodobě vyhrazené HTTP spojení), viz [11]. Na straně klienta jsou sockety implementovány v JavaScriptu třídou WebSocket. Lze tedy vytvořit instanci této třídy a navázat spojení se serverem, který ovšem musí tuto technologii podporovat.

```
var wSocket = new WebSocket ("ws://www.cilovyserver.cz");
```

Příklad 31 – Ukázka vytvoření instance třídy WebSocket v JavaScriptu

Po vytvoření nové instance je možné se serverem komunikovat posíláním zpráv ve formě textových řetězců. Objekt WebSocket disponuje třemi událostmi – *onopen, onmessage, onclose*. Událost *onopen* je vyvolána při otevření spojní a signalizuje možnost komunikace, *onmessage* se zavolá po přijetí zprávy ze serveru a *onclose* je vyvolána po uzavření spojení, ukončení komunikace. Pro zaslání zprávy na server slouží metoda *send()*, parametrem této metody je textový řetězec. K uzavření spojení na straně klienta se může použít metoda *disconnect(),* viz [11].

```
wsocket.send("Textový řetězec");
```

Příklad 32 – Ukázka zaslání zprávy klientem na server pomocí WebSocket technologie

3.6 JSON

JSON je zkratka pro „JavaScript Object Notation“. Jedná se o odlehčený formát pro výměnu dat, který je založen na JavaScriptu. Disponuje jednoduchostí, je snadno zpracovatelný člověkem i strojem. JSON je jazykově i platformě nezávislý textový formát, využívá principy programovacích jazyků C, C++, C#, Java, JavaScript, Python a mnoha dalších, viz [12].

JSON je založen na dvou strukturách. První strukturou je kolekce párů název/hodnota, ve většině jazycích bývá realizována jako objekt, struktura, hash tabulka, slovník, klíčový seznam, asociativní pole, viz [12]. Druhou strukturou je seřazený seznam hodnot, ve většině jazycích bývá realizován jako pole, vektor, seznam, posloupnost, viz [12]. Tyto základní struktury podporuje většina programovacích jazyků, jsou tedy považovány za univerzální a vhodné pro výměnný formát.

JSON může být uložen do samostatného souboru, častěji je ale využíván pro přenos dat v prostředí internetu. Je považován za jednodušší (odlehčenou) alternativu XML formátu, které z velké části obsahuje kontextové značky s jejich atributy.

3.6.1 Syntaxe

Objekt je neuspořádaná množina párů název/hodnota, je ohraničen složenými závorkami. Za otevírací závorkou následují dvojice *název:hodnota,* jednotlivé dvojice jsou odděleny čárkou, na konec je umístěna uzavírací složená závorka, viz [12].

```
{"nazev1":hodnota1,"nazev2":hodnota2}
```

Příklad 33 – Objekt ve formátu JSON

Pole je posloupností hodnot, je ohraničeno hranatými závorkami, jednotlivé hodnoty jsou odděleny čárkou, viz [12].

```
[hodnota1,hodnota2,hodnota3,hodnota4]
```

Příklad 34 – Pole ve formátu JSON

Hodnota je řetězec uzavřený dvojicí uvozovek. Hodnotou může být *číslo*, *true*, *false*, *null*, *objekt*, *pole*. Tyto struktury moho být dále vnořovány, viz [12].

```
"3"
"null"
```

Příklad 35 – Hodnota ve formátu JSON

Řetězcem je nula a více znaků kódování Unicode, jsou uzavřeny dojicí uvozovek, využívá únikových sekvencí pomocí zpětného lomítka.

Číslo je podobné jako v programovacích jazycích s tím, že JSON nepoužívá hexadecimální ani oktalový zápis, viz [12].

Ke kontrole syntaxe formátu JSON existují různé nástroje. Jedním z nich je například online JSON validátor „JSONLint“ [2].

3.7 REX

Řídicí systém REX je nástroj vyvíjený společností REX Controls s.r.o., slouží pro návrh a realizaci algoritmů automatického řízení. Jeho součástí je knihovna funkčních bloků RexLib, z nichž lze následně daný algoritmus sestavit. Jak je uvedeno v [13], řídicí systém je kompatibilní s prostředím *Matlab-Simulink*, vytvořené řídicí algoritmy lze tedy jednoduše odsimulovat před jejich praktickou realizací. Podporuje průmyslový standard OPC, díky tomu je umožněno vytvářet vizualizace a operátorská rozhraní (HMI) ve všech běžných SCADA/HMI systémech, příkladem mohou být systémy jako Reliance, Promotic, Indusoft nebo Labview. Dále je zajištěna podpora jazyka Java, díky kterému lze vytvářet HMI a virtuální laboratoře, přenositelné na různé platformy a schopné přímého vložení do webových stránek. REX dále podporuje technologii WebSocket, která umožňuje obousměrnou komunikaci mezi klientem a serverem v reálném čase. Pro tvorbu HMI je tedy možné využít jádro webového prohlížeče a veškeré technologie specifikace HTML5.

REX je otevřený systém, dovoluje rozšíření a přizpůsobení potřebám uživatele. Je podporován operačními systémy reálného času jako je Windows CE, Windows Embedded, Linux/Xenomai, viz [13]. Pro méně náročné operace, které nevyžadují komunikaci v reálném čase, je možné provozovat řídicí systém na běžných operačních systémech jako je Windows8/7/Vista/XP, viz [13].

Řídicí systém se skládá z šesti hlavních komponent RexDraw, RexView, RexComp, RexCore, RexRun a RexOPCSv, viz [13].

3.7.1 RexDraw

Program umožňující návrh funkčních schémat pomocí bloků knihovny RexLib. Výstupem je soubor s příponou *.mdl*.

2 Dostupné z: http://jsonlint.com

3.7.2 RexView

Program určený pro diagnostiku, umožňuje sledovat aktuální dění v jádře řídicího systému. Poskytuje hierarchicky uspořádané informace o všech subsystémech jádra. Díky komunikaci pomocí protokolu TCP/IP se lze k jádru připojit na lokálním počítači, v lokální síti i ve vzdálené síti na vzdáleném počítači.

3.7.3 RexComp

Po vytvoření hlavního souboru projektu s příponou *.mdl* generuje program RexComp binární konfigurační soubor s příponou *.rex*. Při spuštění programu RexComp vypisuje překladač informace o překládaných souborech a případných chybách. Pokud je zjištěna závažná chyba, překlad se ukončí a binární konfigurační soubor se nevygeneruje. Překladač lze spustit přímo v programu RexDraw pomocí tlačítka Compile.

3.7.4 RexCore

RexCore je jádro řídicího systému REX a běží na cílovém zařízení (PC, IPC, WinCon, WinPAC). Jedná se o komplexní program provádějící paralelně různé činnosti, přičemž jednotlivé úlohy jsou vykonávány na základě priorit v režimu preemptivního multitaskingu.

3.7.5 RexRun

Program sloužící ke spuštění jádra se zvolenou konfigurací v souboru s příponou *.rex*.

3.7.6 RexOPCSv

OPC server umožňující zobrazení dat v běžých vizualizačních SCADA/HMI systémech.

3.8 Inkscape

Inkscape je nástroj pro vytváření vektorové grafiky, jež patří pod neziskovou organizaci „Software Freedom Conservancy“, viz [14]. Je využíván pro tvorbu celé řady grafických objektů jako jsou ilustrace, loga, ikony, diagramy, mapy nebo webové grafiky. Jako nativní formát pro ukládání souborů je používán formát SVG, viz [14]. Inkscape je volně dostupný, otevřený software, lze do něj vkládat různá vlastní rozšíření a přizpůsobit ho svým potřebám.

Schopnosti kreslících nástrojů v Inkscapu jsou srovnatelné s nástroji alternativních grafických editorů jako jsou Adobe Illustrator, CorelDraw nebo Xara Xtreme, viz [14].

Lze importovat a exportovat soubory různých formátů jako jsou SVG, AI, EPS, PDF, PS nebo PNG, viz [14].

3.8.1 Nástroje pro tvorbu objektů

- kreslící nástroje – tužka, pero, kaligrafický nástroj
- základní tvary – obdélník, elipsa (kruh), mnohoúhelník (hvězda), spirála
- textový nástroj – jednořádkový/víceřádkový text
- nástroj klonování – vytváří kopii pomocí odkazu na daný objekt

3.8.2 Nástroje pro manipulaci s objekty

- transformace – translace, rotace, škálování, zkroucení
- pořadí objektů – posunutí v hierarchii nahoru nebo dolů
- seskupení objektů – seskupení více objektů do jedné skupiny
- vrstvy – mohou vytvářet stromovou strukturu

3.8.3 Nástroje pro výplň a obrys

- vzorník barev – RGB, HSL, CMYK, CLS
- kapátko – výběr barvy z objektu
- gradientní přechod – lineární nebo kruhový gradient
- výplň vzorem – bitmapovým nebo vektorovým
- obrys – nepřerušovaný nebo přerušovaný

3.8.4 Nástroje pro manipulace s křivkou

- tvorba uzlů – přidávání, odstraňování uzlů na křivce
- objekt na křivku – převede libovolný objekt na křivku
- zjednodušení křivky
- převod bitmapy na křivky

3.8.5 Textové nástroje

- víceřádkový text
- Kerning, mezery mezi znaky, mezery mezi řádky
- text na křivku – text opisuje trasu křivky

3.8.6 Ostatní nástroje

- editor XML – úprava objektů v XML editoru
- export – export do formátů PNG, ODD, DXF, PDF, EPS, PS, sk1

3.9 Inkscape REX WebHMI extension

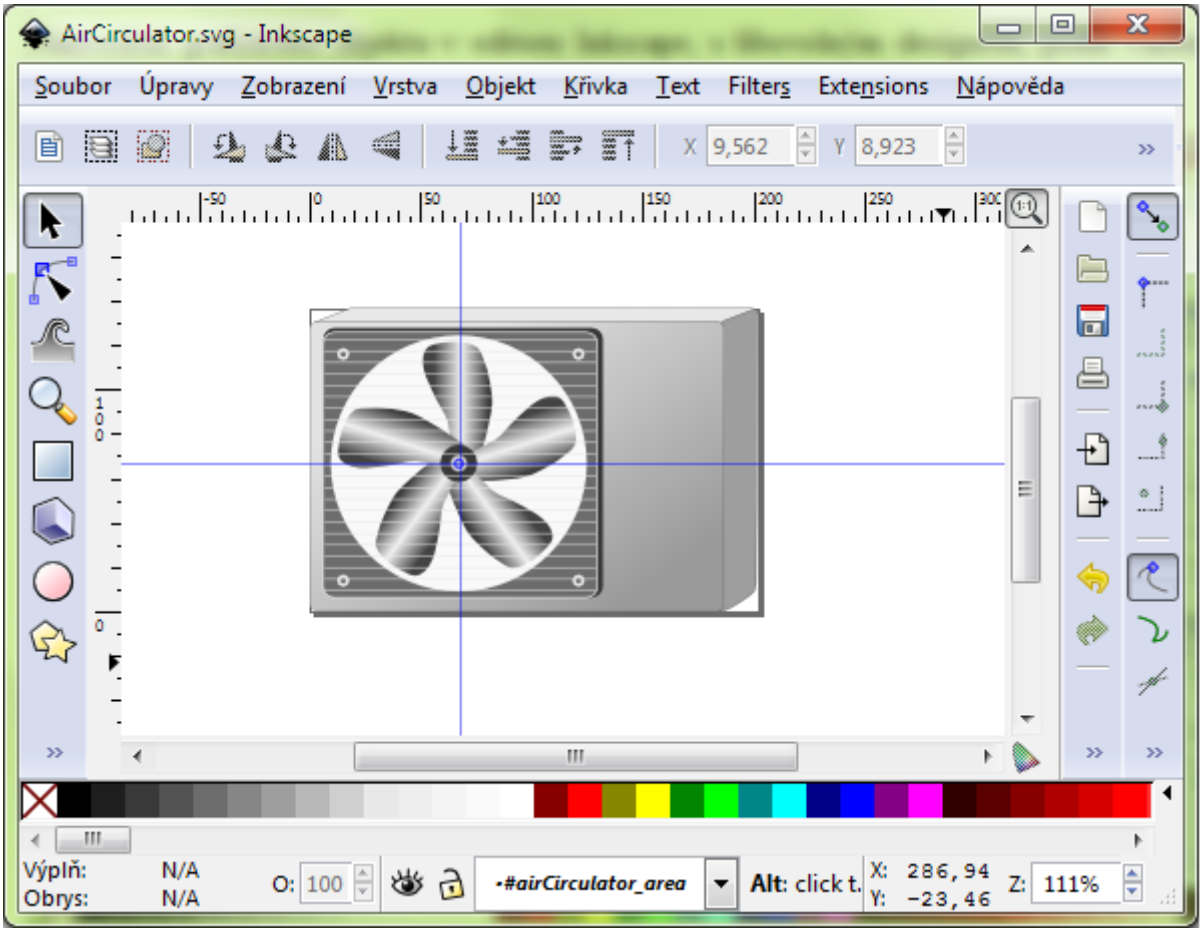
Jedná se o sadu rozšíření grafického editoru Inkscape, jelikož je Inkscape otevřený (open-source) software, je možné vytvářet a vkládat vlastní rozšíření. Do grafického editoru přidává nástroje, pomocí nichž lze vytvořit webovou vizualizaci k modelům sestavených v řídicím systému REX. Pro zakomponování rozšíření je nutná instalace s umístěním do kořenového adresáře Inkscape na pevném disku. Hlavní součátí jsou nástroje *Component Maker*, *Element Edit, Build HTML a Library Browser*, jejichž funkce jsou popsány v následujících kapitolách.

4 Vytvoření komponenty a její struktura

Pro vytvoření vizualizační komponenty je nutné mít nainstalovaný grafický editor Inkscape, dále je potřeba do Inkscapu přidat rozšíření Inkscape REX WebHMI extension (dále jen REX WebHMI). Součástí každé komponenty je zároveň vytvoření skriptu, kterým je inicializována a později ovládána. K vytvoření skriptu lze použít libovolný textový editor nebo některé vývojové prostředí jako je například NetBeans.

4.1 Grafický návrh

Nakreslení grafického objektu v editoru Inkscape, s libovolným designem, podle vlastních potřeb pro vizualizaci.



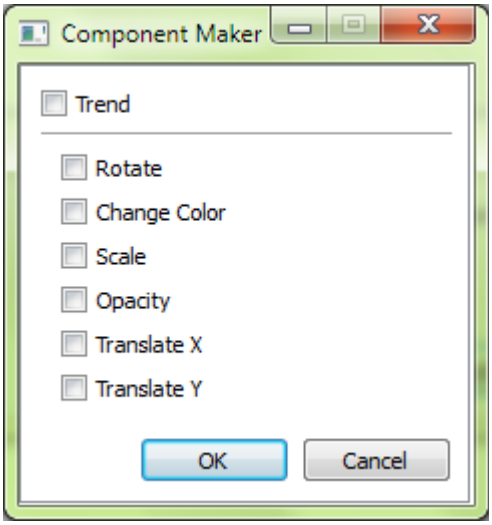
Obrázek 2 – Grafický návrh pomocí nástroje Inkscape

4.2 Aplikace nástroje Component Maker

Component Maker je součástí rozšíření REX WebHMI a jde o nástroj umožňující vytvořit z libovolného grafického objektu (vytvořeného v editoru Inkscape) komponentu, kterou lze následně zobrazit ve webové vizualizaci řídicího systému REX. Nutnou podmínkou při vytváření je seskupení grafického objektu, grafický objekt musí být součástí nějaké skupiny (grupy), jinak řečeno musí být vnořen do elementu <*g*>. Seskupení lze docílit v nabídce *Objekt|Seskupit* nebo pomocí zkratky *CTRL+G*. Nástroj Component Maker lze vyhledat v nabídce *Extensions|RexHMI|Component Maker* a pro jeho použití je nutné mít objekt (skupinu) označenou.

V dialogovém okně nástroje *Component Maker* je nabízeno vytvoření trendu (Trend) z grafického objektu, dále pak přidání základních manipulací s objektem, mezi nimiž se nachází translace ve směru osy x (TranslateX), translace ve směru osy y (TranslateY), rotace kolem daného středu (Rotate), škálování (Scale), změna barvy (Change Color) a průhlednost (Opacity). Vytvoření trendu a základní manipulace je možné ignorovat.

Potvrzením dialogového okna se v XML kódu grafické komponenty automaticky vygenerují potřebné součásti pro vizualizaci. Jedná se o přidání jmenného prostoru *rexsvg*, přidání atributu *module* a vnoření elementů <*title*> a <*desc*> do hlavního elementu.



Obrázek 3 – Dialogové okno nástroje Component Maker

4.2.1 Připojení jmenného prostoru rexsvg

Do kořenového elementu <*svg*> je přidán jmenný prostor (namespace) *rexsvg.*

```
<svg xmlns:rexsvg="http://www.rexcontrols.com/rexsvg" … > … </svg>
```

Příklad 36 – Automaticky generovaný namespace rexsvg nástrojem Component Maker

4.2.2 Přidání modulu do hlavního elementu

V hlavním elementu grafického objektu (grupa nadřazená všem grafickým objektům v dokumentu), který většinou bývá umístěn za elementem <*metadata*>, se dále automaticky vytvoří atribut *module* (modul), který je součástí jmenného prostoru *rexsvg.* Modul slouží k propojení grafického objektu se skriptem komponenty a disponuje výchozí hondnotou *"GeneralComponent".*

```
<svg>
  <defs> … </defs>
  <metadata> … </metadata>
  <g id="idKomponenty"
  rexsvg:module="GeneralComponent" … >
   …
  </g>
</svg>
```

Příklad 37 – Automaticky generovaný atribut module nástrojem Component Maker

4.2.3 Vnoření elementu <title> a <desc>

Dále jsou do hlavního elementu automaticky vnořeny elementy <*title*> a <*desc*>, přičemž oba elementy obsahují textové řetězce. V elementu <*title*> je obsažen název komponenty.(výchozí je"*Title*"). V elementu <*desc*> (descriptions) je řetězec ve formátu JSON, který obsahuje dva objekty *connections* a *options,* ve kterých se později definují připojení a základní parametry komponenty.

```
<g id="idKomponenty"
rexsvg:module="GeneralComponent" … >
  <title>Title</title>
  <desc>
  {
  "connections":{},"options":{}
  }
  </desc>
</g>
```

Příklad 38 – Automaticky generovaný element <title> a <desc> nástrojem Component Maker

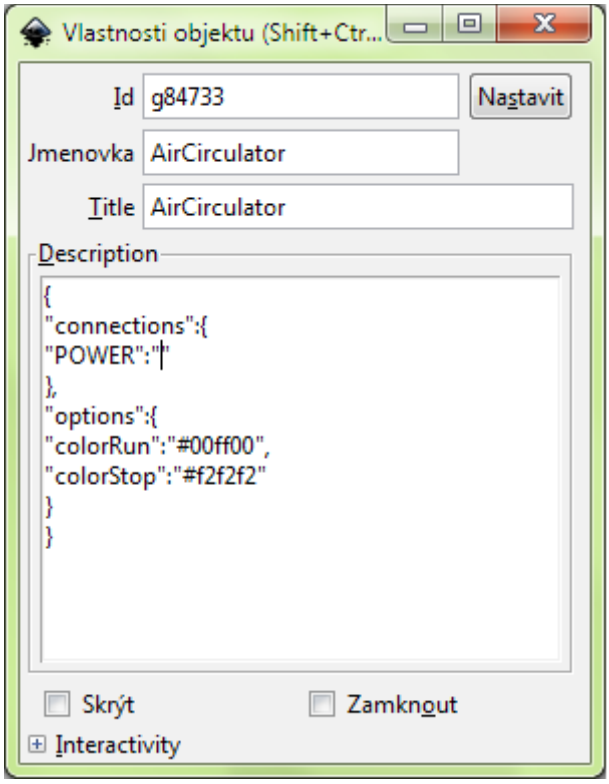
4.3 Definice připojení a základních parametrů

Připojení a základní parametry jsou definovány v elementu *<desc>*, který je využit k výměně informací mezi řídicím systémem a webovovu stránkou. Z tohoto důvodu je použit řetězec ve formátu JSON, který je pro přenos dat vhodný. V automaticky generovaném elementu *<desc>* je potřeba definovat prázdné objekty *connections* (připojení) a *options* (základní parametry). K definování objektů elementu *<desc>* se dostaneme z nabídky Inkscapu *Objekt*| *Vlastnosti objektu* nebo pravým kliknutím myši na objekt s výběrem *Vlastnosti objektu.* V dialogovém okně v části *Descriptions* lze prázdné objekty doplnit.

V objektu *connections* se definují připojení k jednotlivým veličinám řídicího systému. Zvolí se název připojení a přiřadí se mu defaultní hodnota, která je nazývána jako alias. Počet připojení je libovolný, přičemž jednotlivé aliasy musí disponovat unikátními názvy. Hodnota aliasu nemusí být nutně definována, lze ji doplnit až při vytváření konečné vizualizace.

V objektu *options* se definují základní parametry, vlastnosti komponenty, se kterými dále pracuje skript a komponenta je na jejich základě modifikována. Syntaxe je shodná s objektem *connections*, zvolí se název parametru, kterému může být přiřazena výchozí hodnota.

Pro kontrolu syntaxe formátu JSON lze použít různé nástroje jako je například online validátor „JSONLint“.



Obrázek 4 – Definice objektů connections a options

```
<desc>
{
  "connections": {
    "VALUE": "ALIAS"
  },
  "options": {
    "parametr1": "hodnota1",
    "parametr2": "hodnota2"
  }
}
</desc>
```

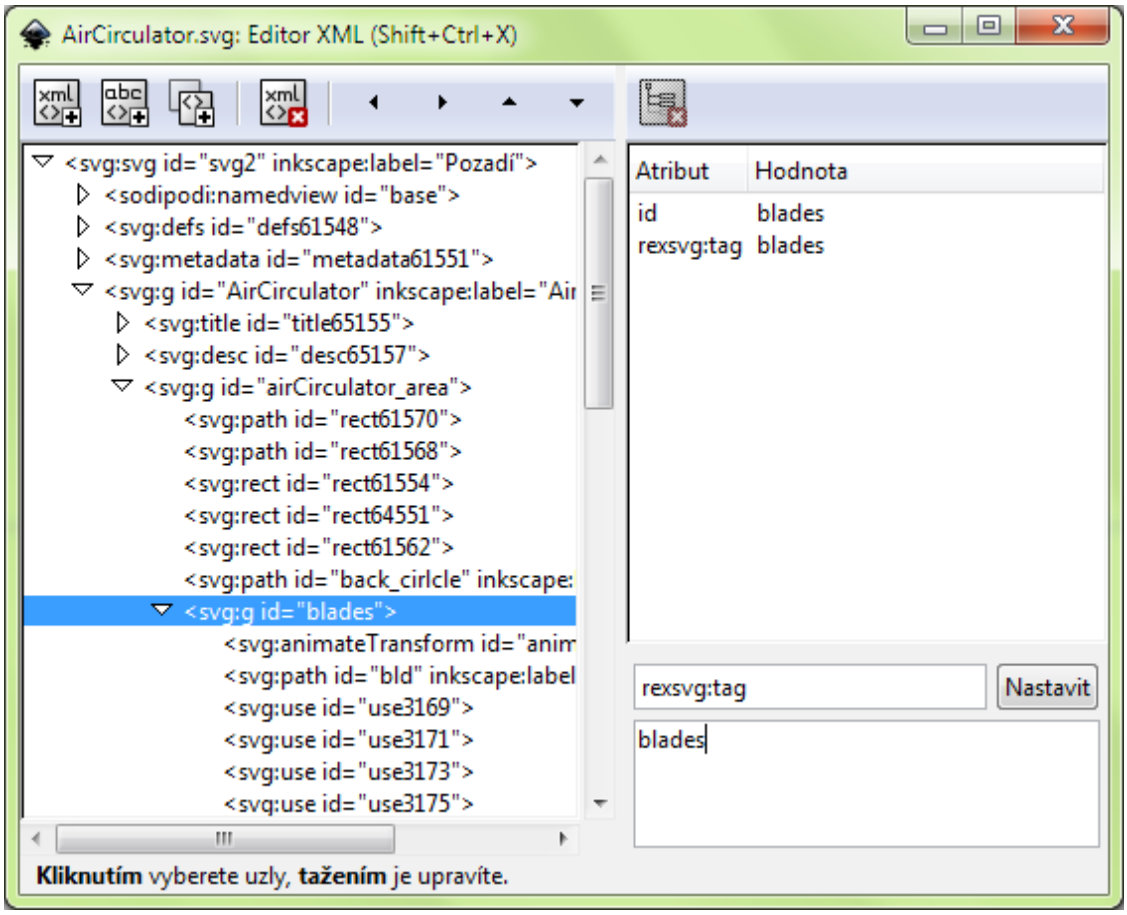
Příklad 39 – Definice připojení a základních parametrů

4.4 Pojmenování objektů pro manipulaci

Následně je potřeba nějakým způsobem označit objekty, se kterými bude ve vizualizaci manipulováno. Pojmenování je možné provést dvěma způsoby, buď pomocí atributu *id* nebo *tag*. Přičemž není vhodné použít atribut *id*, neboť případné zkopírování objektu změní

hodnotu *id* v duplikátu. To způsobí nefunkčnost a chyby ve skriptu, kterým je komponenta ovládána. Atribut *id* musí být unikátní v rámci celého SVG dokumentu.

Pro pojmenování manipulovaných objektů se tedy používá atribut *tag*, který je součástí jmenného prostoru *rexsvg*. Atribut lze přidat například v XML editoru, který nalezneme v nabídce Inkscapu *Úpravy|Editor XML*. V dialogovém okně XML editoru se v levé části zobrazí stromová struktura grafického objektu a v pravé části atributy označeného objektu s jejich hodnotami s možností přidání libovolného atributu. Při označení nějakého objektu ve stromové struktuře se zároveň označí daný objekt přímo v grafickém editoru. Atribut *tag* přitom musí být unikátní pouze v rámci daného objektu.



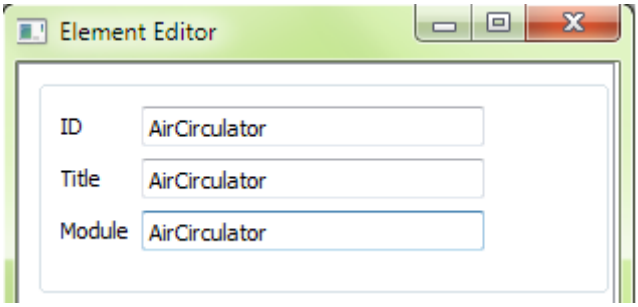
Obrázek 5 – Připojení atributu tag ke grafickému objektu pomocí Editoru XML

```
<rect
  rexsvg:tag="obdelnik"
  id="idObdelniku"
  x="20"
  y="20"
  width="100"
  height="40"/>
<circle
  rexsvg:tag="kruh"
  id="idKruhu"
  cx="50"
  cy="50"
  r="30"/>
```

Příklad 40 – Pojmenování objektů přidáním atributu tag

4.5 Editace modulu a titulu

V této fázi je vhodné použít nástroj *Element Edit* pro změnu výchozí hodnoty modulu (atribut *module*) a výchozí hodnoty titulu (element *<title>*). Tento nástroj nalezneme v nabídce *Extensions|RexHMI|Element Edit* nebo pomocí zkratky *CTRL+E,* používá se na označený objekt. V dialogovém okně tohoto nástroje nás zatím zajímá pouze položka *Title,* ve které definujeme název a dále položka *Module,* ve které definujeme modul komponenty, sloužící k propojení se skriptem. Potvrzením dialogu je editace dokončena.



Obrázek 6 – Editace modulu a titulu

4.6 Vytvoření skriptu

Ke každé komponentě se zhotoví skript, který jí přidává interaktivitu a stará se o manipulaci jejích součástí. K vytvoření skriptu lze použít libovolný textový editor nebo některé vývojové prostředí, například NetBeans. Nově vytvořené skripty komponent jsou obsluhovány nadřazeným skriptem, pojmenovaným *rex-ui-svg.js*, jež je součástí instalace řídicího systému REX. Nadřazený skript slouží k inicializaci přípojných bodů řídicího systému, k vytvoření objektu komponenty a definuje několik funkcí, které lze na objekt komponenty posléze aplikovat.

Mezi funkce nadřazeného skriptu patří například *getChild("id")*, která nalezne „potomka“ grafického objektu s daným atributem *id.* Další podobnou funkcí je *getChildByTag("tag"),* která opět nalezne „potomka“ objektu podle daného atributu *tag*, jež je součástí jmenného prostoru *rexsvg*. Jak již bylo zmíněno, vhodnější je vyhledání „potomka“ pomocí atributu *tag,* jekopírování (duplikování) komponenty v rámci jedné vizualizace. Pokud použijeme vyhledávání součástí pomocí atributu *id*, v rámci jedné vizualizace nelze komponentu duplikovat, neboť *id* každé duplikované komponenty se změní. Změna *id* vede k chybě, neboť je znemožno nalezení „potomka“ ve skriptu.

Ke konečné vizualizaci je připojeno ještě několik skriptů, které plní funkci výchozích, pomocných nástrojů. Patří mezi ně *rex-webvis.min.js*, *jquery-2.0.3.min.js*, *jquery-ui.min.js*, *jquery-disabler.min.js*, *jquery.ui.touch-punch.min.js*, *jquery-ui-timepicker-addon.min.js*, *globalize.js*, *globalize.culture.cs.js*. Všechny tyto skripty jsou k webové stránce automaticky připojeny při vytváření vizualizace a není nutné s nimi manipulovat.

4.6.1 Definice modulu

Skript komponenty se vyznačuje danou strukturu, která vychází z nadřazeného skriptu *rex-ui-svg.js* a z pomocných skriptů.

Základním prvkem skriptu komponenty je definice modulu, kterému je přiřazena funkce vytvářející objekt komponenty. Toho docílíme příkazem, který je uveden v příkladu 35. Za *NazevModulu* použijeme hodnotu (řetězec), přiřazenou atributu *module* (atribut hlavního elementu komponenty), která byla editována při úpravě grafického objektu nástrojem *Element Edit*. Zásluhou definování modulu dojde k propojení XML kódu komponenty se skriptem.

```
REX.UI.SVG.NazevModulu = function (svgElem, args){ … };
```

Příklad 35 – Základní prvek skriptu komponenty

4.6.2 Vytvoření objektu komponenty

Funkce přiřazená modulu obsahuje parametry *svgElem* a *args*. Uvnitř funkce se dále vytvoří objekt komponenty a uloží se do proměnné *that.* Objekt komponenty, neboli proměnná *that*, je zároveň návratovou hodnotou této funkce. V další fázi jsou všechny příkazy definovány uvnitř funkce.

```
function (svgElem, args){
var that = Object.create(REX.UI.SVG.Component(svgElem,args));
 …
return that;
};
```

Příklad 41 – Vytvoření objektu komponenty s návratovou hodnotou

4.6.3 Načtení základních parametrů

Následně mohou být načteny a uloženy základní parametry komponenty, definované v objektu *options,* jenž je součástí elementu *<desc>*. Kompletní sadu parametrů lze přiřadit jediné proměnné s možností pozdějšího jednotlivého výběru, to lze uskutečnit příkazem *that.options* (název proměnné *options* lze zkrátit na tvar *$o*). Každému parametru lze ve skriptu zároveň přiřadit výchozí hodnotu aniž by musel být předem definován. Výchozí hodnota následuje za znakem ||.

```
var $o = that.options || {};
var param1 = $o.parametr1 || 1;
var param2 = $o.parametr2 || 2;
```

Příklad 42 – Načtení základních parametrů komponenty

4.6.4 Načtení grafických součástí

Podobným způsobem jako parametry mohou být dále načteny grafické součásti komponenty, se kterými má být manipulováno při vizualizaci. Načtení lze provést pomocí již zmiňovaných funkcí *getChild("id")* nebo *getChildByTag("tag")*. Jinak řečeno pomocí atributu *id* nebo atributu *tag*, jejichž hodnotu musíme předem definovat.

```
var objekt1 = that.getChildByTag("tag1");
var objekt2 = that.getChildByTag("tag2");
```

Příklad 43 – Načtení grafických součástí komponenty dle atributu tag

```
var objekt1 = that.getChild("id1");
var objekt2 = that.getChild("id2");
```

Příklad 44 – Načtení grafických součástí komponenty dle atributu id

4.6.5 Obsluha událostí

Existují dvě základní události, na které lze ve skriptu reagovat. První z nich je událost *read*, která je vyvolána pokaždé, když je položka (veličina řídicího systému) přečtena. Druhou událostí je událost *change*, ta je vyvolána na začátku (po načtení webové stránky), a poté při každé změně sledované položky (veličiny).

Pro obsluhu události se používá funkce *on(typ, funkce)*, která obsahuje dva parametry. Prvním parametrem je jeden ze dvou typů událostí. Druhým parametrem je funkce, která se vykoná při vyvolání dané události. Tato funkce disponuje jedním parametrem, který představuje sledovanou položku (veličinu). Hodnotu položky lze přečíst pomocí funkce *getValue*().

```
function (itm){
  var hodnotaPolozky = itm.getValue();
}
```

Příklad 45 – Přečtení hodnoty položky

V syntaxi může být použita zkratka *$c* pro objekt *connections*. Na místo NAZEV je vložen název připojení, definovaný v objektu *connections* v grafické komopnentě.

```
that.$c.NAZEV.on('read', function (itm){ … });
that.$c.NAZEV.on('change', function (itm){ … });
```

Příklad 46 – Obsluha událostí read a change

4.6.6 Zápis hodnoty do položky

Ve skriptu je možné změnit hodnotu položky užitím funkce *setValue(hodnota,true)*, prvním parametrem funkce je požadovaná hodnota libovolného typu (dle typu položky), druhým parametrem je vždy logická hodnota *true.* Syntaxe je podobná obsluze událostí, opět může být použita zkratka *$c* představující objekt *connections.* Na místo NAZEV je vložen název

připojení, definovaný v objektu *connections* v komponentě. V tomto případě je zapotřebí, aby byla položka zápisového typu, tzn. alias připojení musí mít definovaný objekt *type* s hodnotou *W* (write).

```
that.$c.NAZEV.setValue(hodnota,true);
```

Příklad 47 – Zápis hodnoty do položky

4.6.7 Ostatní součásti skriptu

Skript může dále obsahovat vlastní kód, zahrnující libovolné příkazy a funkce, pro samotnou manipulaci s grafickými objekty. Důležitými a často používanými součástmi jsou například funcke *setAttributeNS(namespace, atribut, hodnota)* a funkce *createElementNS(namespace, element),* sloužící k úpravě libovolného atributu, resp. k vytvoření a vložení nového elementu.

4.6.8 Ukázkový skript komponenty AirCirculator

```
REX.UI.SVG.AirCirculator = function (svgElem, args) {
var that = Object.create(REX.UI.SVG.Component(svgElem, args));
var $o = that.options || {};

var runColor = $o.colorRun || "#00ff00",
  stopColor = $o.colorStop || "#f2f2f2";

var backCircle = that.getChildByTag("back_circle"),
  blades = that.getChildByTag("blades"),
  animation = that.getChildByTag("blades_animation");

var begin = false;

that.$c.POWER.on('change', function (itm) {
  if (itm.getValue()) {
    backCircle.style.fill = runColor;
    if (!begin) {
      animation.beginElement();
      begin = true;
    }
  }
```

```
  else {
    backCircle.style.fill = stopColor;
    if (begin) {
      animation.endElement();
      begin = false;
    }
  }
});
return that;
};
```

5 Vytvoření vizualizace

Pro zhotovení vizualizace je nutné použít nástroje rozšíření REX WebHMI, knihovnu vlastních komponet a knihovnu základních komponent, která je součástí instalace řídicího systému REX (obsahuje základní komponenty a nadřazený skript *rex-ui-svg.js*).

Knihovnu vlastních komponent musíme předem vytvořit. Toho docílíme seskupením všech vytvořených vizualizačních komponent do jednoho adresáře s libovolným umístěním a názvem. Tento adresář disponuje danou strukturou, obsahuje podadresář s názvem *js,* do kterého jsou umístěny veškeré skripty (soubory s příponou *.js*), náležící vlastnoručně vytvořeným komponentám. Dále jsou do adresáře umístěny veškeré grafické komponenty (soubory s příponou *.svg*). Knihovna ještě může obsahovat podadresář s názvem *css*, ve kterém mohou být definovány kaskádové styly ke každé komponentě (soubory s příponou *.css*).

5.1 Vizualizační plocha a vložení komponent

Zpočátku je potřeba vytvořit vizualizační plochu. Ta se vytvoří jako prázdný dokument v Inkscapu, je možné v ní vytvořit libovolnou grafiku, která bude sloužit jako pozadí vizualizace. Do dokumentu se z vytvořené knihovny (adresáře) vloží, přetažením, grafické komponenty. Nalezení knihovny komponent usnadňuje nástroj *Library Browser*, který je v nabídce *Extensions|RexHMI|Library Browser*. Při jeho spuštění se zobrazí průzkumník.

5.2 Připojení konfigurační komponenty HMIConfig

Pomocí nástroje *Element Edit* se vytvoří potřebná konfigurační komponenta *HMIConfig,* sloužící k nastavení základních vlastností vizualizace. V konfigurační komponentě je dále umožněno definovat připojení, která budou později přidělena jednotlivým vizualizačním komponentám. Nástroj *Element Edit* se spouští z nabídky *Extensions|RexHMI|Element Edit* nebo pomocí zkratky *CTRL+E*.

Obrázek 7 – Komponenta HMIConfig

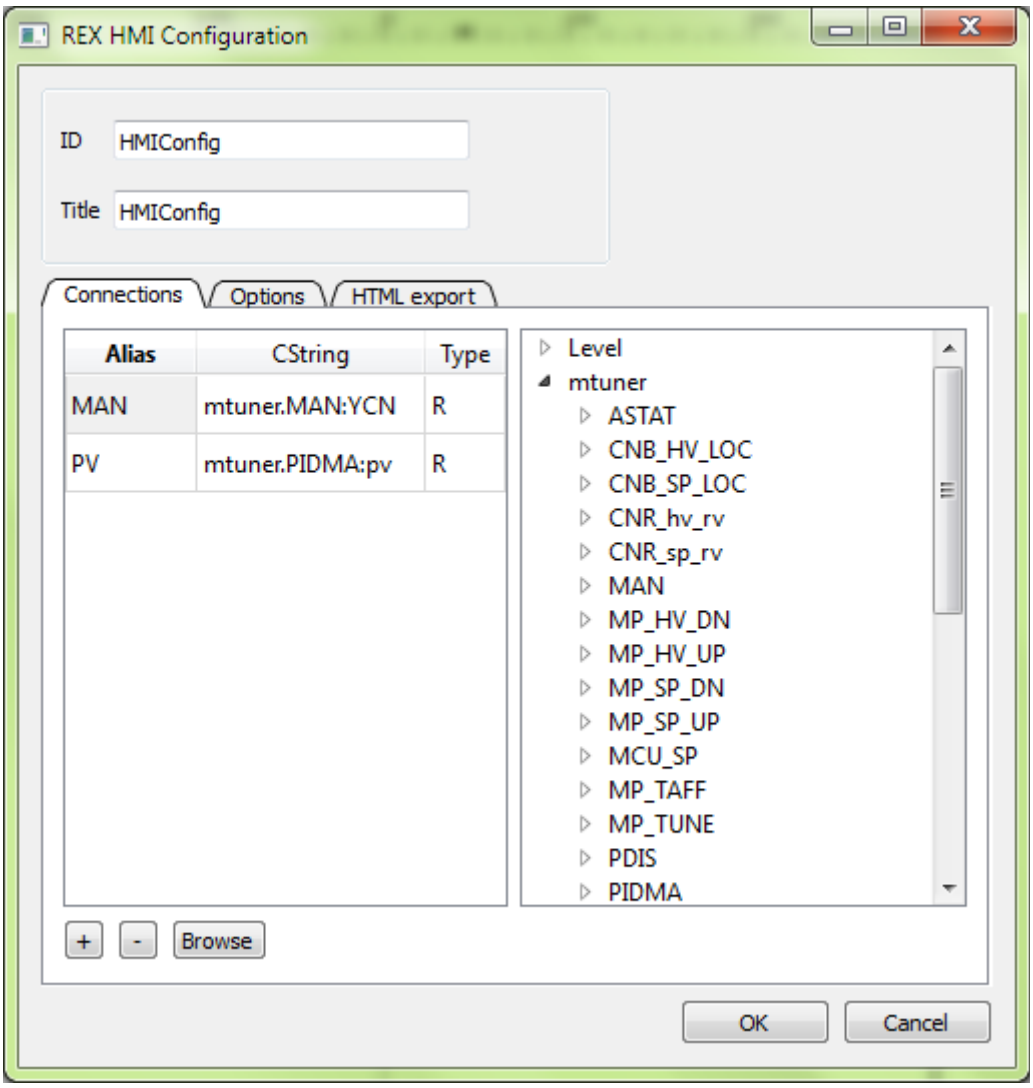
Konfigurační komponenta umožňuje definovat přípojné body řídicího systému s jejich aliasy, název a umístění vizualizace na disku, knihovny komponent nebo například IP adresu cílového zařízení. Konfigurační komponenta je automaticky vytvořena spuštěním nástroje *Element Edit* za podmínky, že není označen žádný grafický objekt. Jako každá jiná komponenta i konfigurační má definovaný modul (*HMIConfig*), do hlavního elementu jsou vnořeny <*title*> a <*desc*>. V objektu *connections* elementu <*desc*> jsou, v tomto případě, definovány aliasy s připojovacími řetězci a typem připojení. V konečné vizualizaci se konfigurační komponenta nezobrazuje, je skryta.

Pokud označíme konfigurační komoponentu a následně spustíme nástroj *Element Edit*, zobrazí se dialogové okno se třemi záložkami, přičemž záložka *Options* může zůstat nezměněna.

V záložce *Connections* lze libovolně přidávat přípojné body k řídicímu systému, v položce *Alias* se definuje alias (název), v položce *CString* se definuje připojovací řetězec, *[jmeno_tasku].[jmeno_bloku]:[parametr]* a v položce *Type* se definuje typ připojení. Pokud máme navíc v řídicím systému REX spuštěný nějaký proces, můžeme použít tlačítko *Browse*, které vytvoří kompletní stromovou strukturu všech přípojných bodů spuštěného procesu a následně je lze, dvojklikem, přidat do položky *CString*. Typ připojení může nabývat pouze dvou hodnot, *"R"* nebo *"W"*. *"R"* znamená, že hodnota položky je určená pouze ke čtení, *"W"* znamená, že hodnotu položky lze ve skriptu přepsat.

V záložce *HTML Export* v položce *HTML Header* se definuje hlavička webové stránky a v položce *HTML Title* titulek webové stránky. V *HTML Target* se definuje umístění a název souboru s příponou *.html*. Do položky *Library Path* je nutné přidat knihovny, které obsahují použité vizualizační komponenty, navíc je potřeba přidat knihovnu s názvem *General*[3], která je součástí instalace řídicího systému REX a obsahuje základní komponenty a nadřazený skript *rex-ui-svg.js*.

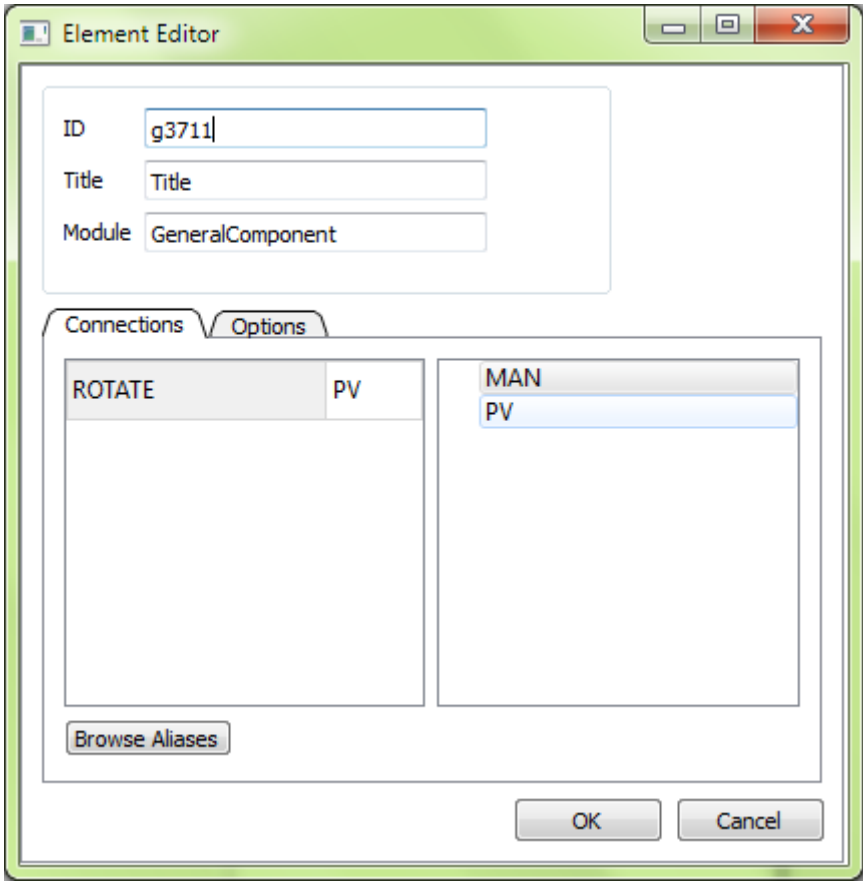
3 C:\ProgramData\REX Controls\Rex WebHMI\libs



Obrázek 8 – Dialogové okno konfigurační komponenty

5.3 Editace vizualizačních komponent

U všech komponent vložených do vizualizační plochy musí být nastaveny hodnoty připojení. Nastavení (editace) docílíme označením komponenty a následnou aplikací nástroje *Element Edit*. V dialogovém okně je záložka *Connections* a záložka *Options*. Záložka *Connections* představuje připojení, do kterých je nutné přiřadit některý z aliasů, definovaných v konfigurační komponentě. Pokud již máme aliasy v konfigurační komponentě definované, můžeme použít tlačítko *Browse Aliases*, které zobrazí seznam všech dostupných aliasů. Následně je možné alias ze seznamu přiřadit. Záložka *Options* představuje základní parametry komponenty, které lze nastavit, přičemž není nutné je vyplňovat, pokud mají ve skriptu nastaveny výchozí hodnoty.



Obrázek 9 – Dialogové okno nástroje Element Editor

5.4 Aplikace nástroje Build HTML

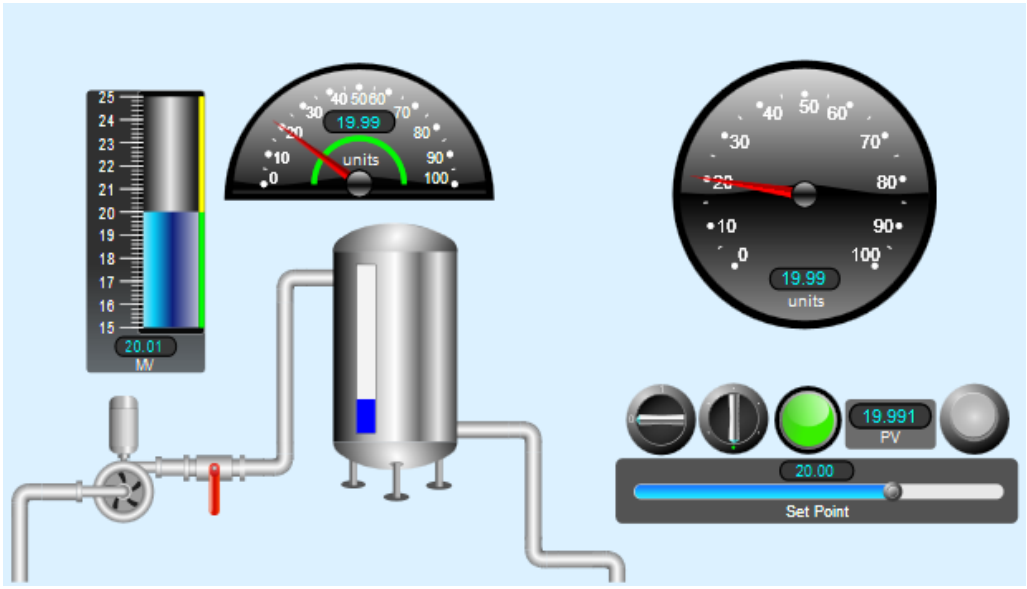
Na závěr je potřeba připravenou vizualizaci exportovat do umístění, které bylo zvoleno v konfigurační komponentě. Export je možné uskutečnit pomocí nástroje *Build HTML* z nabídky *Extensions|RexHMI|Build HTML* nebo pomocí zkratky *CTRL+H*. Tímto krokem je proces vytvoření vizualizace ukončen a lze ji zobrazit v podporovaném prohlížeči.

Ke správné funkčnosti vizualizace a ke komunikaci s řídicím systémem je nutné před jejím spuštěním aktivovat webový server *Lighttpd* a WebSocket server *RexWSTcp,* které jsou součástí řídicího systému a lze je spustit pomocí nástroje *HMI services*[4].

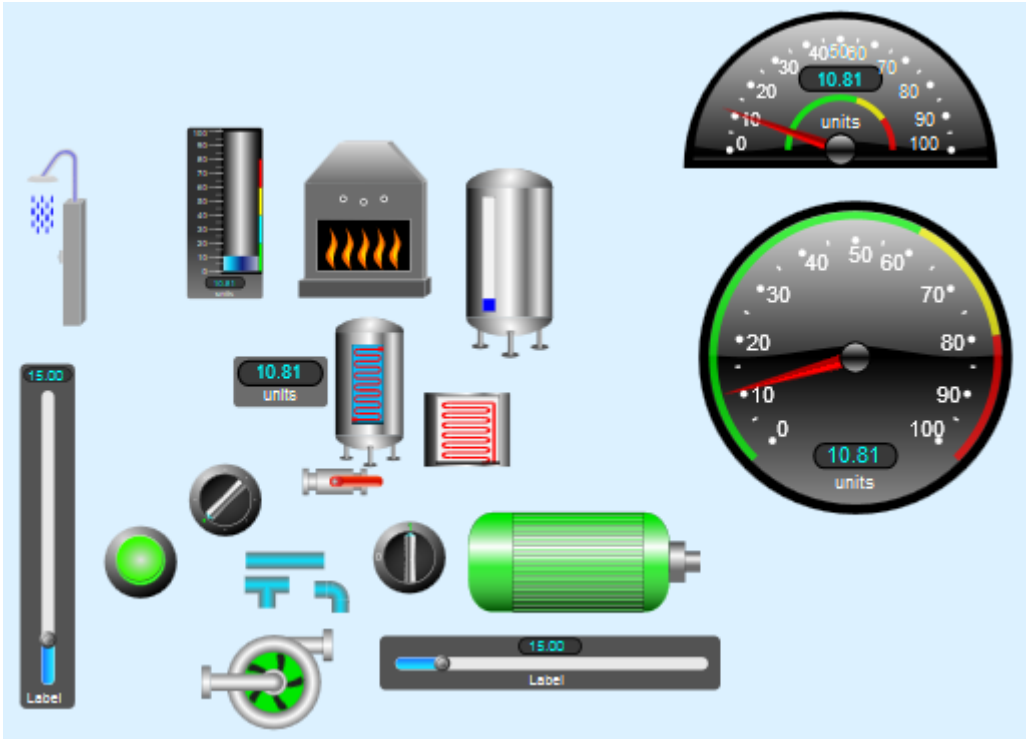
4 Start/Rex Controls/REX_X_XX_X_XXXX_XXX/Nastroje/HMI services

5.5 Ukázkové vizualizace komponent

Ukázkové vizualizace byly připojeny k příkladu *MTUNER*, jež je součástí řídicího systému REX.



Obrázek 10 – Ukázková vizualizace



Obrázek 11 – Ukázková vizualizace komponent

6 Seznam vytvořených komponent

6.1 AirCirculator

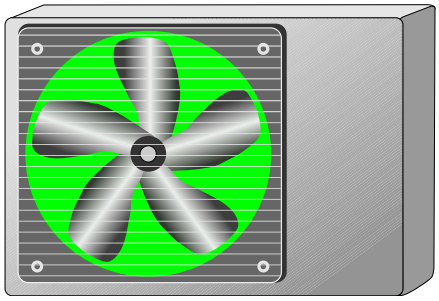
Venkovní jednotka tepelného čerpadla, která disponuje jedinou funkcí, otáčením vrtule a nastavenou barvou je signalizován stav, kdy je jednotka v chodu.

6.1.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)

6.1.2 Základní parametry

- *colorRun* – barva v pozadí vrtule, zobrazovaná při zapnuté jednotce
- *colorStop* – barva v pozadí vrtule, zobrazovaná při vypnuté jednotce



Obrázek 12 – Venkovní jednotka tepelného čerpadla

6.2 BarGraph

Sloupcový graf umožňující zobrazit aktuální číselnou hodnotu položky. Zahrnuje nastavitelnou číselnou osu, digitální hodnotu, barevné rozsahy a popisek.

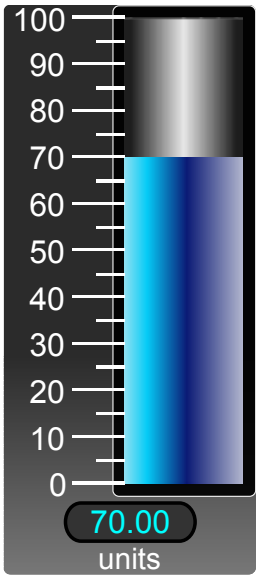
6.2.1 Připojení

- *VALUE* – položka může nabývat pouze číselné hodnoty (v pohyblivé řádové čárce)

6.2.2 Základní parametry

- *rangeMin* – minimální hodnota rozsahu (osy)
- *rangeMax* – maximální hodnota rozsahu (osy)

- *tickStep* – krok dílků na ose
- *mainTickStep* – krok hlavních dílků s číselným popiskem
- *digitalPrecision* – počet desetinných míst u digitální hodnoty
- *units* – jednotky nebo jiný popisek
- *zoneStartValues* – pole počátečních hodnot barevných rozsahů, jednotlivé hodnoty se oddělují buď čárkou nebo mezerou a je nutné dodržet jeden způsob v rámci parametru, lze definovat libovolný počet barevných rozsahů
- *zoneEndValues* – pole konečných hodnot barevných rozsahů, způsob zápisu je shodný jako v případě počátečních hodnot, přičemž počet konečných hodnot musí odpovídat počtu počátečních hodnot
- *zoneColors* – pole barev pro definované rozsahy, opět mohou být odděleny čárkou nebo mezerou, barvy jsou v hexadecimálním tvaru (#ffffff)



Obrázek 13 – Sloupcový ukazatel

6.3 Boiler

Plynový kotel, který se může ocitnout ve třech stavech se signalizací ohně: zapnutý na maximální výkon, zapnutý na minimální výkon, vypnutý.

6.3.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)
- *CURRENTTEMP* – číselná položka aktuální teploty
- *SETTEMP* – číselná položka požadované teploty

6.3.2 Základní parametry

Základní parametry nejsou definovány.

Obrázek 14 – Plynový kotel

6.4 DigitalValue

Digitální hodnota (display), umožňující zobrazit číselnou hodnotu položky

6.4.1 Připojení

- *VALUE* – položka může nabývat pouze číselné hodnoty (oborem jsou reálná čísla)

6.4.2 Základní parametry

- *precision* – počet zobrazovaných desetinných míst
- *units* – jednotky nebo jiný popisek

Obrázek 15 – Display s číselnou hodnotou

6.5 Filter

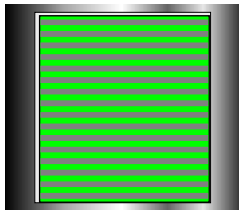
Vzduchový filtr, schopný dvou stavů: zapnutý, vypnutý.

6.5.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)

6.5.2 Základní parametry

- *colorOn* – barva zapnutého filtru
- *colorOff* – barva vypnutého filtru



Obrázek 16 – Vzduchový filtr

6.6 Gauge180

Ručičkový ukazatel („budík“) se stupnicí v rozsahu 180°, umožňující zobrazit číselnou hodnotu položky. Součástí je nastavitelný rozsah stupnice, digitální hodnota, barevné rozsahy a popisek.

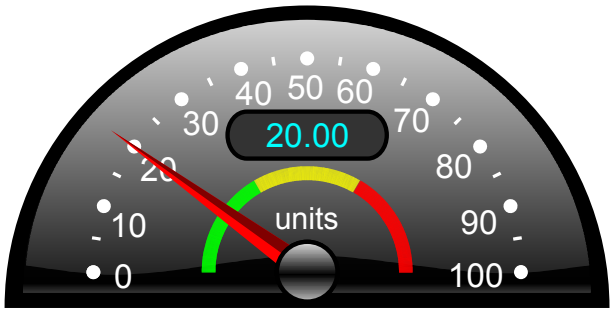
6.6.1 Připojení

- *VALUE* – položka může nabývat pouze číselné hodnoty (oborem jsou reálná čísla)

6.6.2 Základní parametry

- *rangeMin* – minimální hodnota rozsahu (osy)
- *rangeMax* – maximální hodnota rozsahu (osy)
- *tickStep* – krok dílků na ose
- *mainTickStep* – krok hlavních dílků s číselným popiskem
- *digitalPrecision* – počet desetinných míst u digitální hodnoty

- *units* – jednotky nebo jiný popisek
- *zoneStartValues* – pole počátečních hodnot barevných rozsahů, jednotlivé hodnoty se oddělují buď čárkou nebo mezerou a je nutné dodržet jeden způsob v rámci parametru, lze definovat libovolný počet barevných rozsahů
- *zoneEndValues* – pole konečných hodnot barevných rozsahů, způsob zápisu je shodný jako v případě počátečních hodnot, přičemž počet konečných hodnot musí odpovídat počtu počátečních hodnot
- *zoneColors* – pole barev pro definované rozsahy, opět mohou být odděleny čárkou nebo mezerou, barvy jsou v hexadecimálním tvaru (#ffffff)



Obrázek 17 – Ručičkový ukazatel se stupnicí v rozsahu 180°

6.7 Gauge270

Ručičkový ukazatel („budík“) se stupnicí v rozsahu 270°, umožňující zobrazit číselnou hodnotu položky. Součástí je nastavitelný rozsah stupnice, digitální hodnota, barevné rozsahy a popisek.

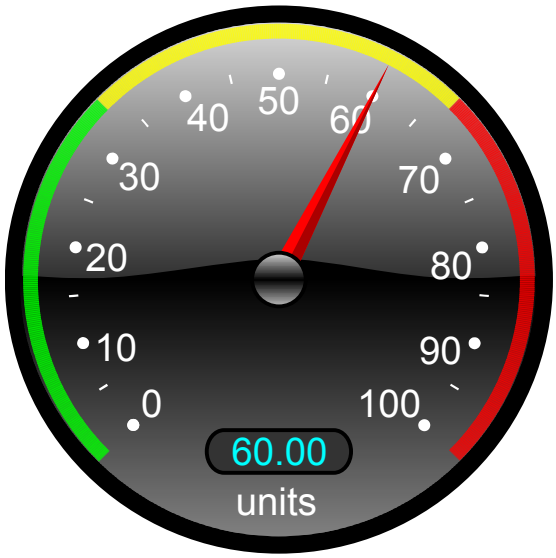
6.7.1 Připojení

- *VALUE* – položka může nabývat pouze číselné hodnoty (oborem jsou reálná čísla)

6.7.2 Základní parametry

- *rangeMin* – minimální hodnota rozsahu (osy)
- *rangeMax* – maximální hodnota rozsahu (osy)
- *tickStep* – krok dílků na ose
- *mainTickStep* – krok hlavních dílků s číselným popiskem
- *digitalPrecision* – počet desetinných míst u digitální hodnoty

- *units* – jednotky nebo jiný popisek
- *zoneStartValues* – pole počátečních hodnot barevných rozsahů, jednotlivé hodnoty se oddělují buď čárkou nebo mezerou a je nutné dodržet jeden způsob v rámci parametru, lze definovat libovolný počet barevných rozsahů
- *zoneEndValues* – pole konečných hodnot barevných rozsahů, způsob zápisu je shodný jako v případě počátečních hodnot, přičemž počet konečných hodnot musí odpovídat počtu počátečních hodnot
- *zoneColors* – pole barev pro definované rozsahy, opět mohou být odděleny čárkou nebo mezerou, barvy jsou v hexadecimálním tvaru (#ffffff)



Obrázek 18 – Ručičkový ukazatel se stupnicí v rozsahu 270°

6.8 HandleValve

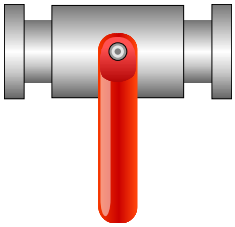
Dvoucestný ventil, který se může ocitnout ve dvou stavech: otevřený, zavřený.

6.8.1 Připojení

- *FLOW_R* – položka může nabývat logické hodnoty true nebo false(otevřený/zavřený), slouží k přečtení hodnoty
- *FLOW_W* – položka může nabývat logické hodnoty true nebo false(otevřený/zavřený), slouží k zápisu hodnoty

6.8.2 Základní parametry

- *initialPosition* – počáteční poloha ventilu, může nabývat hodnoty *on* (otevřený), hodnoty *off* (zavřený) nebo může zůstat nevyplněna, přičemž dojde k přečtení aktuální polohy.



Obrázek 19 – Dvoucestný ventil

6.9 HandleValveT

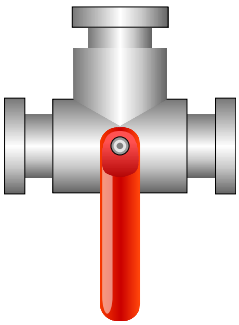
Třícestný spojitý ventil.

6.9.1 Připojení

- *FLOW1_R* – položka může nabývat logické hodnoty true nebo false(otevřený/zavřený), slouží k přečtení hodnoty otevření ventilu v prvním směru
- *FLOW1_W* – položka může nabývat logické hodnoty true nebo false(otevřený/zavřený), slouží k zápisu hodnoty, otevření ventilu prvním směrem
- *FLOW2_R* – položka může nabývat logické hodnoty true nebo false(otevřený/zavřený), slouží k přečtení hodnoty otevření ventilu v druhém směru
- *FLOW2_W* – položka může nabývat logické hodnoty true nebo false(otevřený/zavřený), slouží k zápisu hodnoty, otevření ventilu druhým směrem

6.9.2 Základní parametry

- *initialPosition* – počáteční poloha ventilu, může nabývat hodnoty „*1*“ (otevřený pvním směrem), hodnoty „*2*“ (otevřený druhým směrem) nebo může zůstat nevyplněna, přičemž dojde k přečtení aktuální polohy.



Obrázek 20 – Třícestný ventil

6.10 Heater

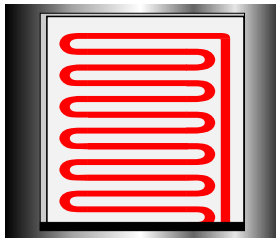
Potrubní ohřívač vzduchu se třemi stavy: zapnutý v režimu ohřívání, zapnutý v režimu bez ohřívání a vypnutý.

6.10.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)
- *CURRENTTEMP* – číselná položka aktuální teploty
- *SETTEMP* – číselná položka požadované teploty

6.10.2 Základní parametry

- *colorCooling* – barva, nastavená po překročení požadované teploty (aktuální teplota je vyšší než požadovaná)
- *colorHeating* – barva, nastavená při ohřívání (aktuální teplota je nižší než požadovaná)
- *colorOff* – barva vypnutého ohřívače



Obrázek 21 – Potrubní ohřívač vzduchu

6.11 Heating

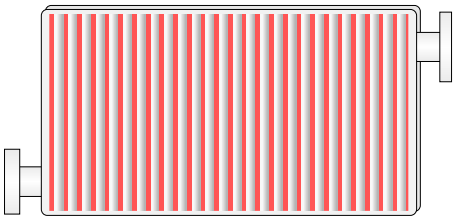
Topení (radiátor) se dvěma stavy: zapnuté (ohřívání), vypnuté

6.11.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)
- *CURRENTTEMP* – číselná položka aktuální teploty
- *SETTEMP* – číselná položka požadované teploty

6.11.2 Základní parametry

- *colorOn*– barva zapnutého topení
- *colorOff* – barva vypnutého topení



Obrázek 22 – Topení (radiátor)

6.12 Led

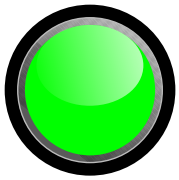
Led signalizace s připojením na logickou hodnotu položky, může nabývat dvou stavů: položka je ve stavu true (zapnuto), položka je ve stavu false (vypnuto).

6.12.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)

6.12.2 Základní parametry

- *colorRun* – barva, signalizující stav „zapnuto“
- *colorOff* – barva, signalizující stav „vypnuto“



Obrázek 23 – Led signalizace

6.13 Motor

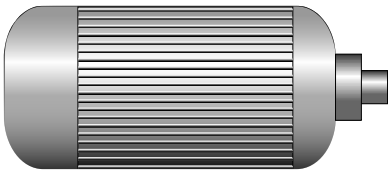
Motor se dvěma stavy: zapnutý, vypnutý.

6.13.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)

6.13.2 Základní parametry

- *colorRun* – barva zapnutého motoru
- *colorStop* – barva vypnutého motoru



Obrázek 24 – Motor

6.14 PipeStraight, PipeElbow, PipeT

Rovná trubka, koleno trubky a trubka ve tvaru „T“, které nabývají dvou stavů: prázdné, plné.

6.14.1 Připojení

- *PIPECONTENT* – připojená položka může nabývat pouze logické hodnoty true nebo false (prázdná/plná trubka)

6.14.2 Základní parametry

- *colorRun* – barva plné trubky
- *colorStop* – barva prázdné trubky

Obrázek 25 – Vedení trubek

6.15 Pump

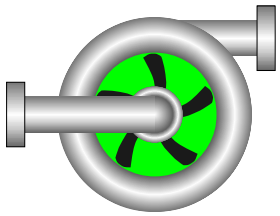
Vodní čerpadlo se dvěma stavy: zapnuté, vypnuté.

6.15.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)

6.15.2 Základní parametry

- *colorRun* – barva zapnutého čerpadla
- *colorStop* – barva vypnutého čerpadla



Obrázek 26 – Vodní čerpadlo

6.16 PushOnOff

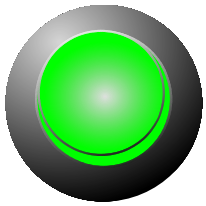
Tlačítko, sloužící k přepínání mezi dvěma stavy

6.16.1 Připojení

- *POWER_R* – připojená položka může nabývat pouze logické hodnoty true nebo false, slouží k přečtení aktuální hodnoty (zapnuto/vypnuto)
- *POWER_W* – připojená položka může nabývat pouze logické hodnoty true nebo false, slouží k zápisu hodnoty

6.16.2 Základní parametry

- *initialPosition* – počáteční poloha, může nabývat hodnoty *on* (zapnuto), hodnoty *off* (vypnuto) nebo nemusí být vyplněna, v tomto případě dojde k přečtení a nastavení aktuální polohy



Obrázek 27 – Tlačítko

6.17 Shower

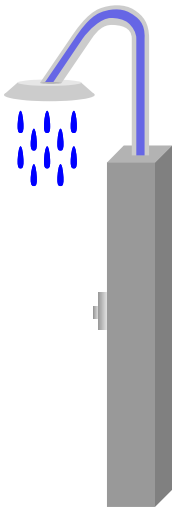
Sprcha se dvěma stavy: zapnutá (tekoucí voda), vypnutá.

6.17.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)

6.17.2 Základní parametry

Základní parametry nejsou definovány.



Obrázek 28 – Sprcha

6.18 SliderHorizontal

Horizontální slider („posuvník“), umožňující nastavit a zapsat hodnotu z definovaného rozsahu. Zahrnuje číselný ukazatel a popisek.

6.18.1 Připojení

- *VALUE_R* – připojená položka může nabývat číselné hodnoty, slouží k přečetení aktuální hodnoty
- *VALUE_W* – připojená položka může nabývat číselné hodnoty, slouží k zápisu hodnoty do položky

6.18.2 Základní parametry

- *minValue* – minimální hodnota rozsahu
- *maxValue* – maximální hodnota rozsahu
- *step* – krok slideru („posuvníku“)
- *initialValue* – počáteční hodnota, pokud není vyplněna, přečte se aktuální
- *digitalPrecision* – počet desetinných míst u digitální hodnoty
- *fontSize* – velikost písma popisku
- *label* – popisek

Obrázek 29 – Horizontální slider

6.19 SliderVertical

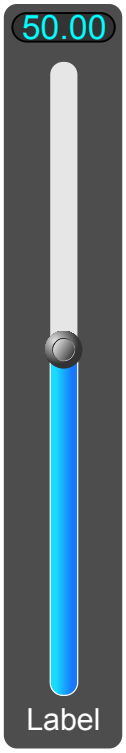
Vertikální slider („posuvník“), umožňující nastavit a zapsat hodnotu z definovaného rozsahu. Zahrnuje číselnou hodnotu a popisek.

6.19.1 Připojení

- *VALUE_R* – připojená položka může nabývat číselné hodnoty, slouží k přečetení aktuální hodnoty
- *VALUE_W* – připojená položka může nabývat číselné hodnoty, slouží k zápisu hodnoty do položky

6.19.2 Základní parametry

- *minValue* – minimální hodnota rozsahu
- *maxValue* – maximální hodnota rozsahu
- *step* – krok slideru („posuvníku“)
- *initialValue* – počáteční hodnota, pokud není vyplněna, přečte se aktuální
- *digitalPrecision* – počet desetinných míst u digitální hodnoty
- *fontSize* – velikost písma popisku
- *label* – popisek



Obrázek 30 – Vertikální slider

6.20 Switch

Přepínač s volitelným počtem poloh, sloužící pro zápis číselných hodnot.

6.20.1 Připojení

- *VALUE_R* – připojená položka může nabývat číselné hodnoty, slouží k přečetení aktuální hodnoty
- *VALUE_W* – připojená položka může nabývat číselné hodnoty, slouží k zápisu hodnoty do položky

6.20.2 Základní parametry

- *initialPosition* – počáteční poloha, v tomto případě jedna z definovaných hodnot
- *valuesOfPositions* – hodnoty v jednotlivých polohách, oddělené čárkou nebo mezerou, počet definovaných hodnot určuje počet poloh přepínače

Obrázek 31 – Přepínač s volitelným počtem poloh

6.21 SwitchOnOff

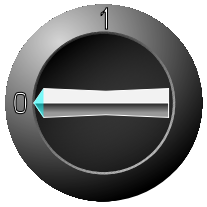
Přepínač, sloužící k přepínání mezi dvěma stavy.

6.21.1 Připojení

- *VALUE_R* – připojená položka může nabývat číselné hodnoty, slouží k přečetení aktuální hodnoty
- *VALUE_W* – připojená položka může nabývat číselné hodnoty, slouží k zápisu hodnoty do položky

6.21.2 Základní parametry

- *initialPosition* – počáteční poloha, může nabývat hodnoty „1“ (zapnuto) nebo hodnoty „0“ (vypnuto)



Obrázek 32 – Přepínač se dvěma stavy

6.22 Tank

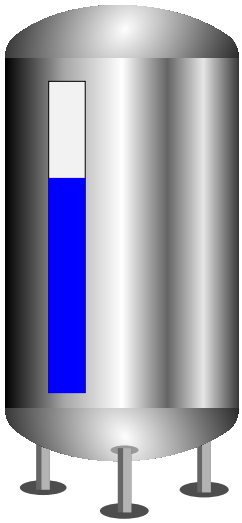
Nádrž, zobrazující aktuální hladinu kapaliny nebo plynu.

6.22.1 Připojení

- *LEVEL* – připojená položka může nabývat číselné hodnoty, slouží k přečetení aktuálního objemu (případně výšky hladiny)

6.22.2 Základní parametry

- *capacity* – objem (kapacita) nádrže, případně maximální výška hladiny
- *colorOfLevel* – barva hladiny v nádrži



Obrázek 33 – Nádrž na kapalinu/plyn

6.23 WaterBoiler

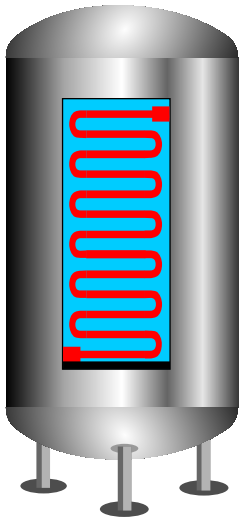
Nádrž („bojler“) sloužící k ohřevu vody, která se může ocitnout ve třech stavech: zapnuto v režimu ohřívání, zapnuto v režimu bez ohřívání a vypnuto.

6.23.1 Připojení

- *POWER* – připojená položka může nabývat pouze logické hodnoty true nebo false (zapnuto/vypnuto)
- *CURRENTTEMP* – číselná položka aktuální teploty
- *SETTEMP* – číselná položka požadované teploty

6.23.2 Základní parametry

- *colorCooling* – barva spirály, nastavená po překročení požadované teploty (aktuální teplota je vyšší než požadovaná)
- *colorHeating* – barva spirály, nastavená při ohřívání (aktuální teplota je nižší než požadovaná)
- *colorOff* – barva spirály vypnutého ohřívače

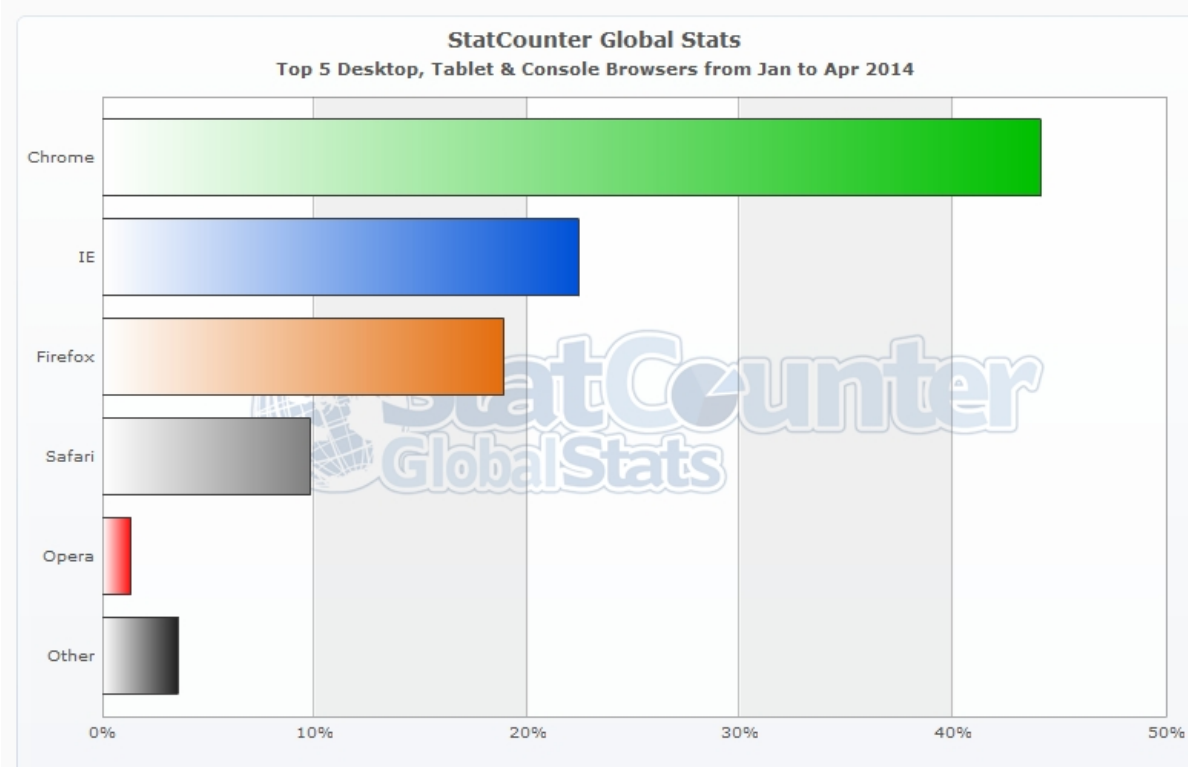


Obrázek 34 – Nádrž na vodu s ohřevem

7 Testování a podpora v prohlížečích

Veškeré komponenty byly testovány připojením na příklad *MTUNER*, který je součástí řídicího systému REX a slouží jako ukázkové funkční schéma.

K testování kompatibility a funkčnosti vizualizace byly využity aktuální verze vybraných nejrozšířenějších webových prohlížečů. Podle serveru „StatCounter GlobalStats“ patří (celosvětově) mezi nejpoužívanější prohlížeče nynější doby Chrome od společnosti „Google“, Internet Explorer od společnosti „Microsoft“, Firefox vyvíjený společností „Mozilla“, Safari od společnosti „Apple“ a Opera, která je produktem společnosti „Opera Software“.



Obrázek 35 – Statistika nejpoužívanějších webových prohlížečů od ledna do dubna v roce 2014[5]

Testování proběhlo v následujících prohlížečích:

- Google Chrome 34.0.1847.131 m
- Internet Explorer 11.0.9600.17105
- Mozilla Firefox 29.0.1
- Opera 21.0.1432.57

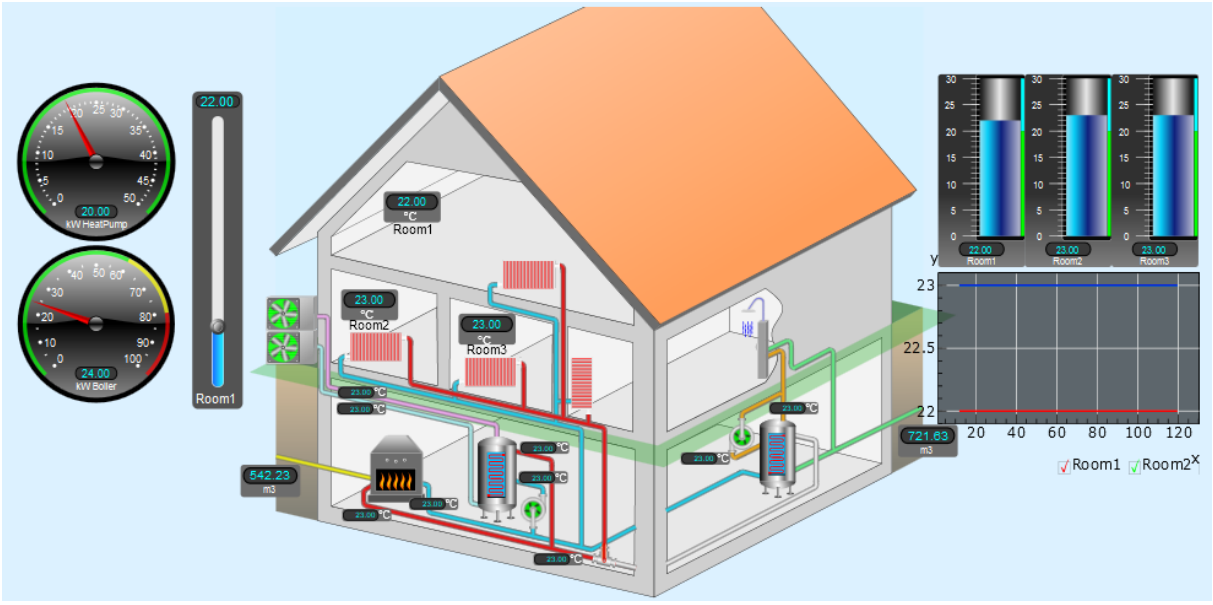
5 Dostupné z http://gs.statcounter.com/#browser-ww-monthly-201401-201404-bar

Výsledkem testu je úplná podpora v prohlížečích Chrome, Firefox a Opera, ve kterých při vizualizaci nevznikl žádný problém. Částečná podpora je i v prohlížeči Internet Explorer, kde se u některých komponent při vizualizaci vyskytly chyby. Jedná se zejména o animované komponenty, které v prohlížeči Internet Explorer zůstavají statické.

8 Vizualizace rodinného domu

Jako praktický příklad byla navržena a sestavena vizualizace rodinného domu. Základ je postaven na modelu řídicího systému REX, který vychází z práce Martina Tichého [16].

Sledovanými částmi domou jsou podkroví s jednou místností, přízemí se dvěma místnostmi a koupelnou a sklep. V každé obytné místnosti je radiátor a čidlo teploty. V koupelně je umístěno topné těleso a sprcha. Ve sklepě se nachází plynový kotel s plynoměrem na přívodu plynu, vnitřní jednotka tepelného čerpadla (tepelný výměník s čerpadlem), zásobník teplé vody (TUV). Tepelné čerpadlo získává energii ze vzduchu, proudícího z venkovní jednotky do tepelného výměníku a zpět. Proudění studené vody do tepelného výměníku zajišťuje čerpadlo. Kotel a tepelné čerpadlo topí do stejného výstupu, který je dále rozdělen třícestným ventilem. Teplá voda je tímto ventilem směrována buď do topné soustavy nebo do zásobníku (TUV), přičemž ventil se přepíná automaticky. Pokud je aktuální teplota vody v zásobníku nižší než požadovaná, ventil do něj automaticky nasměruje teplou vodu. Do zásobníku je přivedena studená voda s odbočkou do koupelny a s vodoměrem na jejím přívodu. Ze zásobníku (TUV) je dále veden oběhový okruh s čerpadlem s odbočkou teplé vody do koupelny. Vizualizace dále obsahuje ukazatele aktuálního výkonu kotle a aktuálního výkonu tepelného čerpadla, ukazatele teploty vzduchu v obytných místnostech, ukazatele teploty vody v trubkách nebo například trend teplot.



Obrázek 36 – Vizualizace rodinného domu

9 Závěr

V současnosti existuje několik přístupů pro vytvoření vizualizace k modelům řídicího systému REX. Tyto metody jsou spojeny s řadou nevýhod, z toho důvodu začal vývoj nové metody, která lépe vyhovuje vlasntím potřebám.

Jednou z výhod vlastního řešení je, že není potřeba pořizovat licenci, veškeré potřebné součásti jsou volně dosutpné. Druhou výhodou jsou „otevřené“ zdrojové soubory, tím je umožněno rozšíření a přizpůsobení knihovny i samotných grafických komponent dle vlastních potřeb pro vizualizaci. Další výhodou je snadné a rychlé vytvoření vizualizace, využitím knihovny hotových komponent.

Cílem bylo navrhnout a otestovat sadu vizualizačních komponent, které lze jednoduše editovat, lze jim nastavit základní parametry a lze je vkládat do webové stránky s možností skriptování. Ukázalo se, že vhodným řešením k tomuto účelu je použití webových technologií HTML5, JavaScript a technologie SVG, která je jimi podporována. Za pomoci těchto technologií se podařilo vytvořit knihovnu několika základních komponent s různými parametry a funkcemi.

V průběhu práce byly popsány použité technologie a nástroje, byl uveden postup vytvoření komponenty a vytvoření konečné vizualizace, byly popsány parametry a možnosti připojení jednotlivých komponent. Sada navržených komponent byla dále testována v nejrozšířenějších webových prohlížečích. Ktestování byl použit ukázkový příklad řídicího systému, ke kterému byly komponenty připojeny. Součástí práce je také demonstrační příklad, ve kterém jsou navržené komponenty zahrnuty.

Do budoucna by bylo vhodné knihovnu komponent rozšiřovat a doplňovat novými prvky tak, aby byla univerzální a dala se použít pro různá odvětví a různé typy řídicích systémů.

Seznam použité literatury

[1] Reliance. Co znamená SCADA/HMI? [online]. c2014 [cit. 2014-4-26]. Dostupné z WWW: <http://www.reliance.cz/cs/products/what-does-scada-hmi-mean?highlight_result=hmi>

[2] World Wide Web Consortium. HTML5 [online]. c2013, 15 April 2014 [cit. 2014-4-17]. Dostupné z WWW: <http://www.w3.org/TR/html5/>

[3] SMOLA, Martin. HTML5: Co přináší a proč se o něj zajímat. In: *Root.cz* [online]. 29. 8. 2012 [cit. 2014-4-17]. Dostupné z WWW: <http://www.root.cz/clanky/html5-co-prinasi-a-proc-se-o-nej-zajimat/>

[4] LARSEN, Rob. HTML5, CSS3, and related technologies [online]. 26 April 2011 [cit. 2014-4-19]. Dostupné z WWW: <http://www.ibm.com/developerworks/library/wa-webstandards/>

[5] HTML Tutorial [online]. c2014 [cit. 2014-4-19]. Dostupné z WWW: <http://www.w3schools.com/html/>

[6] TRANTÝR, Tomáš. Technologie SVG a HTML5 objekt Canvas jako perspektivní metody vkládání dynamické grafiky do www stránek [online]. České Budějovice, 2012. Bakalářská práce.Jihočeská univerzita v Českých Budějovicích. Fakulta pedagogická. Dostupné z WWW: <http://www.petrpexa.cz/diplomky/trantyr.pdf>

[7] SVG Tutorial [online]. c2014 [cit. 2014-4-20]. Dostupné z WWW: <http://www.w3schools.com/svg/>

[8] JavaScript. In: *Wikipedia: otevřená encyklopedie* [online]. Stránka byla naposledy editována 17. 4. 2014, 12:11 [cit. 2014-4-23]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/JavaScript>

[9] JavaScript Tutorial [online]. c2014 [cit. 2014-4-23]. Dostupné z WWW: <http://www.w3schools.com/js/>

[10] jQuery Tutorial [online]. c2014 [cit. 2014-25-4]. Dostupné z WWW: <http://www.w3schools.com/jquery/>

[11] MALÝ, Martin. Web Sockets. In: *Zdroják, o tvorbě webových stránek a aplikací* [online]. 14. 12. 2009 [cit. 2014-4-26]. Dostupné z WWW: <http://www.zdrojak.cz/clanky/web-sockets/>

[12] JSON. *Introducing JSON* [online]. [cit. 2014-4-26]. Dostupné z WWW: <http://json.org/>

[13] Rex Controls. Řídicí systém REX [online]. c2014 [cit. 2014-4-29]. Dostupné z WWW:http: <http://www.rexcontrols.cz/rex>

[14] Software Freedom Conservancy. Inkscape Overview. *What is Inkscape?* [online]. [cit. 2014-4-29]. Dostupné z WWW: <http://inkscape.org/en/about/>

[15] Software Freedom Conservancy. Features of Inkscape [online]. [cit. 2014-4-29]. Dostupné z WWW: <http://inkscape.org/en/about/features/>

[16] TICHÝ, Martin. Regulace teploty v budovách. Plzeň, 2012. Bakalářská práce. Západočeská univerzita v Plzni. Fakulta aplikovaných věd. Katedra kybernetiky.

Přílohy

- CD – přiložené CD obsahuje bakalářskou práci ve formátu *PDF*, ve složce *Img* jsou umístěny veškeré použité ilustrace a složka *ComponentLibrary* obsahuje knihovnu komponent (grafické soubory, skripty).