Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra

Bakalářská práce

Operátorské rozhraní řídicích systémů využívající standard HTML5

Plzeň 2016 Jan Cibulka

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, jejichž úplný seznam je její součástí.

V Plzni dne 10. května 2016

.....

Poděkování

Rád bych zde poděkoval Ing. Pavlu Baldovi, Ph.D., za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Anotace

Cílem této práce je vytvořit sérii příkladů s grafickými vizualizacemi z oblasti řízení pohybu. Nejprve jsou popsány jazyky a technologie, které jsou běžně používány při tvorbě webových stránek. Také je popsán řídicí systém REX a knihovna bloků Motion Control. Dále je porovnání již existujících prostředků pro řešení problematiky operátorského rozhraní ve světě a popis užívaných technologií. Navazující část se zabývá návrhem grafických vizualizačních bloků a přidruženého skriptu. Na konci práce je přehled vytvořených příkladů a shrnutí postupu jejich tvorby.

Klíčová slova

Vizualizace, řízení pohybu, animace, funkční bloky.

Annotation

Purpose of this thesis is to create collection of examples with graphical visualization and motion control background. At the beginning, languages and technologies used in web development are described. The REX control system and the Motion Control block library are also explained. Followed by comparison of already existing forms of solutions to this issue and description of used technologies. Next part is focused on design of graphical visualisation blocks and associated script. Also, overview of created examples and summarized process of their creation is included.

Keywords

Visualization, motion control, animation, function blocks.

Obsah

1 Úvod

Hlavním cílem této práce je seznámit čtenáře se způsobem návrhu operátorského rozhraní řízeného procesu. Toto rozhraní bude realizováno formou webové stránky. Řízený proces bude využívat řídicího systému REX, který obsahuje vývojové a diagnostické nástroje a také procesní jádro, kde probíhá samotné řízení procesů, sběr informací a předávání dat dále.
Pro systém REX je možné vyvinout zajímavé úlohy, protože je navržen pro řízení nejrůznějších technologických procesů. Zajímavé úlohy pocházejí například z oblasti řízení pohybu. Pro tento účel je v systému REX implementována knihovna funkčních bloků Motion Control podle standardu od skupiny PLCopen. Je tedy vhodné krátce seznámit čtenáře i s problematikou použití funkčních bloků řídicích pohyb.

Součástí práce je i porovnání již používaných přístupů pro realizaci vizualizace a řízení.

Výsledkem práce je série příkladů z oblasti řízení pohybu, které jsou tvořeny modely v systému REX a webovou vizualizací. Je vhodné, aby tato stránka byla úspěšně zobrazitelná na různých platformách, které se běžně používají k řízení procesů, tedy v prostředí osobních počítačů s operačním systémem Windows nebo Linux, dále pak na tabletu a mobilním telefonu.

Pro tvorbu těchto vizualizací je použito standardu HTML5, kaskádových stylů CSS a skriptovacího jazyka JavaScript. Je důležité seznámit čtenáře i s těmito technologiemi. Pro připojení k řídicímu systému REX byla již v jazyce JavaScript vyvinuta knihovna *RexHMI.js*. Dalším prvkem použitým při tvorbě vizualizace bude diagram bloků vygenerovaný přímo z vývojového prostředí RexDraw. Zde budou užity prvky vektorové grafiky. Vektorová grafika se upravuje v programu Inkscape, který také poskytuje rozšíření RexHMI určené přímo pro vývoj vizualizací. Toto rozšíření obsahuje připravenou sadu ovládacích prvků, které mohou být zakomponovány do vizualizace.

2 Teoretická část

2.1 Řídicí systém REX

Řídicí systém REX [1] je produktem firmy REX Controls s.r.o. Slouží k návrhu a realizaci pokročilých úloh z oblasti automatického řízení. Výsledný algoritmus se sestává z umístěných funkčních bloků, kde každý blok plní svoji danou funkci. Rozmanitý obsah těchto bloků zajišťuje pokrytí požadavků pro reprezentaci řízení i méně běžných technologických procesů.

2.1.1 Popis komponentů systému

Celý řídicí systém se skládá z několika částí [2].

RexDraw

RexDraw je grafické prostředí umožnující vytváření a editaci algoritmů. K dispozici jsou rozsáhlé knihovny funkčních bloků. Základem je vytvoření modelu *.mdl*, který se zkompiluje na soubor *.rex* a ten je možno dále předat RexCore. Po kompilaci a nahrání souboru do cílového zařízení je možné RexDraw změnit na diagnostický nástroj, který poslouží ke sledování stavu funkčních bloků a změnám jejich parametrů za běhu na cílovém zařízení.

RexView

RexView je program, který po připojení k cílovému zařízení poskytuje přehled a diagnostiku běžícího procesu. Je zde možné měnit parametry signálu, prohlížet si trend nebo zkoumat historii v archivním souboru. Také je možné řídit samotné jádro RexCore, například kontrolovat periodu provádění jednotlivých úkolů.

RexCore

Jádro RexCore zajišťuje běh úloh na cílovém zařízení. Stará se o správné spouštění a časování jednotlivých úloh na základě jejich priorit v režimu preemptivního multitaskingu. Velkou výhodou je pokrytí různých platforem, na kterých je možné spustit RexCore. Na běžném notebooku je možné dosáhnout periody vzorkování 5ms, na průmyslových počítačích až 1ms.

RexCore je složen z následujících subsystémů:

- Subsystém reálného času *RT*,
- Vstupně-výstupní subsystém *I/O*,
- Algoritmický subsystém *LB* (Informace o knihovně funkčních bloků) ,
- Diagnostický subsystém *Diag*,
- Archivační subsystém *Arc*.

2.1.2 Obecný popis bloků

Všechny bloky mají několik jednotících rysů. Každý blok má svojí grafickou reprezentaci v programu RexDraw. Uživatel programu RexDraw, který má zkušenosti se základními kombinacemi funkčních bloků tak může snadno, intuitivně a rychle vytvořit řídicí algoritmus.

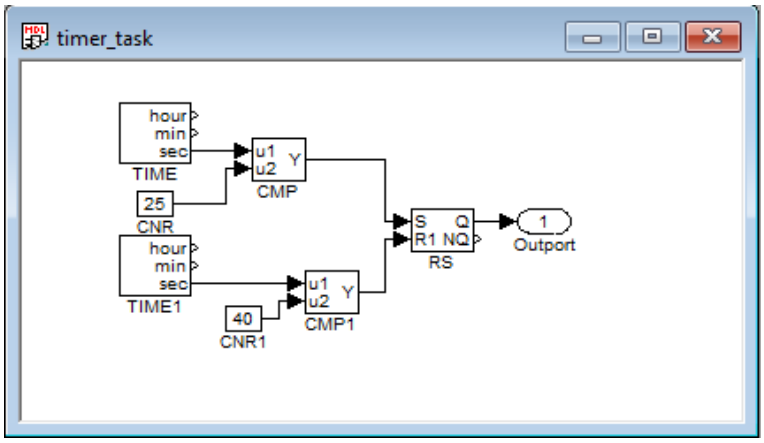
Každý blok má přesně definovanou množinu vstupních a výstupních signálů a také svoje vnitřní parametry [3]. Každá z těchto veličin musí být správného datového typu a musí být vždy v intervalu mezi minimální a maximální přípustnou hodnotou, jinak dojde k chybě.

Pro veličiny bloků se používají nejčastěji tyto tři datové typy:

bool : Tento datový typ se používá pro veličiny, kde jsou přípustné pouze dvě hodnoty, logická nula a jednička, zapnuto a vypnuto. Podle konvence systému REX se pro veličiny tohoto typu používají názvy složené z velkých písmen.

long : Používá se tam, kde jsou celočíselné hodnoty, například tedy číslo sady parametrů, délka trendového bufferu, typ generovaného signálu, chybový kód nebo výstup čítače. Podle konvence se zde používají názvy, které jsou složené z malých písmen a začínají většinou na jedno písmeno ze sady (i, k, l, m, n, o).

double : Datový typ double se používá pro vyjádření čísel s pohyblivou řádovou čárkou. Toho je potřeba u analogových signálů, pro matematické operace a na mnoha jiných místech. Podle konvence je vhodné tyto veličiny pojmenovávat názvy složenými z malých písmen.



Obrázek 2.1: Jednoduché schéma s klopným obvodem ukazující způsob propojení REX bloků.

2.1.3 Popis jednotlivých kategorií funkčních bloků

EXEC : Tato část obsahuje zejména bloky sloužící pro celkové sestavení struktury řízené úlohy a nastavení časování jednotlivých objektů obsažených v programu RexCore. Tyto bloky nemají ekvivalent v programu Matlab-Simulink, protože ten slouží pouze pro simulace a není zde potřeba řídit režii runtime jádra.

INOUT : Tato část obsahuje vstupně-výstupní bloky určené pro vazbu mezi systémem REX, jeho řídicími úlohami a vstupně výstupními ovladači kontrolujícími data přicházející a odcházející na periférie.

MATH : Tato část obsahuje bloky pro základní matematické funkce a operace.

ANALOG : Tato část obsahuje bloky zaměřené na zpracování analogových signálů. Patří sem integrátor, derivátor, dopravní zpoždění a například frekvenční filtry.

GEN : Tato část obsahuje bloky generující signál. Může se jednat o analogové i logické hodnoty, šum nebo složitou sekvenci hodnot.

REG : Tato část je nejobsáhlejší a obsahuje mnoho typů regulátorů, od jednoduchých typů založených na dynamické kompenzaci regulační odchylky po několik verzí PID regulátorů.

LOGIC : Tato část obsahuje bloky pro kombinační a sekvenční logické řízení.

ARC : Tato část obsahuje bloky pro archivaci dat a trendů v systému REX přímo v cílovém zařízení. Tyto bloky nemají žádný ekvivalent v programu Matlab-Simulink.

PARAM : Tato část obsahuje bloky usnadňující modifikaci a získávání parametrů z běžícího procesu.

MODEL : Tato část obsahuje bloky sloužící pro tvorbu matematických modelů dynamických systému pracujících v reálném čase.

MATRIX : Tato část obsahuje bloky, které umí pracovat s vícerozměrnými signály ve vektorovém nebo maticovém tvaru.

SPEC : V této části jsou dva speciální bloky. Blok RDC, umožnující komunikaci mezi dvěma programy Matlab-Simulink nebo pro komunikaci s OPC serverem. Druhým je blok REXLANG, do kterého je možno zaznamenávat vlastní algoritmy v jazyce velmi podobném jazyku C.

Dále jsou zde dvě další sekce, *MC_SIGNLE* a *MC_MULTI*, které se zabývají řízením pohybu na jedné a na více osách.

2.1.4 Knihovna Motion Control

Knihovna funkčních bloků Motion Control [4] je vytvořena podle světového standardu PLCopen-Motion Control. Motivace pro vznik tohoto standardu byla velká rozmanitost průmyslových zařízení pro řízení pohybu. Standardizace zvýšila znovupoužitelnost schémat při nákupu nových zařízení.

Knihovna se zabývá řízením pohybu. Základní myšlenkou je, že existuje blok reprezentují osu pohybu. Na tento blok jsou připojené další bloky, každý s definovanou akcí. Existují tedy příkazy typu *Přesuň osu do pozice x.* nebo *Zastav veškerý pohyb a vrat' se do výchozí pozice.* . Blok má vstup *Execute*, kde čeká na náběžnou hranu signálu, který značí požadavek na změnu stavu osy, nastavení parametru nebo začátek pohybu. Po skončení akce se blok sám deaktivuje a čeká na další impuls.

Funkční bloky této knihovny se dále dělí do dvou skupin. V první jsou administrativní bloky, které zajišťují obsluhu parametrů a osy. Mezi tyto bloky patří například *MC_Power* nebo *MC_WriteParameter*.

Ve druhé skupině jsou bloky zabývající se pohybem samotným. Mezi tyto bloky patří například *MC_MoveAbsolute*, *MC_PositionProfile* a *MC_Home*. Je umožněno řízení pohybu ve více osách. Propojení mezi více osami zajišťují bloky *MC_CamIn* (vačka) a *MC_GearIn* (převodovka).

Základem je použití bloku *RM_Axis*, který představuje samotnou osu. K jeho výstupu *axisRef*, který obsahuje referenci na osu, se připojí ostatní bloky, které manipulují s parametry osy nebo vykonávají pohyb.

2.2 Základní popis jazyka HTML

2.2.1 Co je to HTML

HTML (HyperText Markup Language) je jazyk navržený pro tvorbu webových stránek. Jednou z jeho mnoha výhod je jednoduchost, takže základní konstrukce jsou zřejmé ze samotného náhledu na zdrojový kód. Běžný člověk je schopen se naučit tyto základy za krátkou dobu.

Důležitou vlastností je obrovská rozmanitost realizovatelných prvků obsahu, který tvoříme. Jelikož v oblasti webových stránek dochází stále k velmi rychlému rozvoji, je důležité, aby byl tento jazyk neustále aktualizován podle nejnovějších trendů a potřeb ve světě vývoje webových stránek. Tuto činnost vykonává organizace W3C (World Wide Web Consortium).
Vhodný popis principů jazyka vyplývá již z jeho názvu :

HyperText : Způsob, jakým je realizována navigace po webové stránce. Kliknutím na označený text se přenesete na jinou HTML stránku kdekoliv na Internetu.

Markup : Funkce HTML „tagu“. Tag označí určitou část textu jako určitý typ textu s obecně definovanými vlastnostmi.

Language : Jedná se o jazyk. Vyskytuje se zde tedy konkrétní předepsaná syntaxe a objevují se zde klíčová slova.

Tento jazyk funguje tak, že se vytvoří soubor, jehož obsah velmi připomíná běžný text. Do něho se vloží krátké úseky kódu, které specifikují způsob, jak je text interpretován webovým prohlížečem. Toto jsou *tagy*. Text je pak uložen jako soubor s příponou *.html*. Při požadavku webového prohlížeče na čtení stránky je přenesen tento soubor. Interpretace tohoto souboru pak probíhá v prohlížeči na straně klienta. Dojde ke přečtení souboru, vyhodnocení tagů a převedení textu do vizuální podoby.

Z tohoto procesu vyplývá další silná stránka HTML, totiž že pro vytváření dokumentů není třeba speciálního software nebo editačního studia, ale stačí obyčejný textový editor.

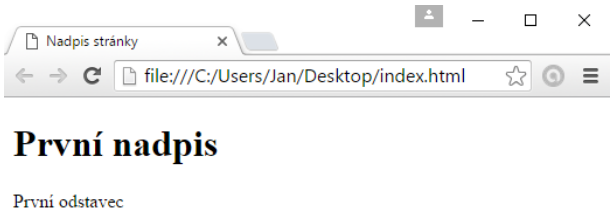
HTML tagy jsou tím, co odlišuje HTML soubor od běžného textu. Jsou to klíčová slova mezi ostrými závorkami < a >. Značky se neobjevují ve výsledné vizualizaci, jde o informace určené hlavně prohlížeči. Tyto značky umožňují označovat části textu, které chceme, aby měly určitou vlastnost, například že se jde o nadpis, že má být text podtržený, tučný nebo že se jedná o odkaz. Mimo přiřazování vlastností textu je možné vkládat speciální objekty jako například tabulky, obrázky nebo navigační menu.

Příklad základních prvků HTML
Tento velmi jednoduchý příklad ukáže, jakým způsobem funguje označování textu v HTML.

Příklad 2.1: Ukázka použití naprosto základní HTML struktury. Obsah je rozčleněn do hlavičky (head) a těla (body) stránky.

```
<!DOCTYPE html>
<html>
<head>
<title>Nadpis stránky</title>
</head>
<body>
<h1>První nadpis</h1>
<p>První odstavec</p>
</body>
</html>
```

Toto je obsah souboru pojmenovaného například *index.html*. Po otevření ve webovém prohlížeči vypadá zobrazená stránka jako na obrázku 2.2.



Obrázek 2.2: Základní HTML stránka.

2.3 Nový obsah v HTML5

Představení HTML verze 5 znamenalo obrovskou změnu ve využití HTML. Verze 1.0 až 4.0 (a XHTML) se zaměřovaly a používaly na rozmísťování textu a obrázků na webové stránce. Novější verze zahrnovaly určité pokusy o poskytnutí sémantických dat, které by pomohly automaticky rozpoznávat a kategorizovat HTML dokumenty. Podporu médií zajišťují pluginy Microsoft-Silverlight nebo Adobe-Flash, u kterých se ukázalo, že představují významné bezpečnostní riziko.

Oproti tomu HTML5 přidává hodně nového obsahu. Využívá HTML spíše jako aplikační platformu, která dokáže nejen zobrazovat text a obrázky, ale daleko více se zaměřuje na „aktivní“ obsah. Umožňuje přehrávat video a audio, hrát hry, obsahovat

interaktivní 2D a 3D grafické prvky, řeší ukládání dat v prohlížeči a přístup k nim v online i offline režimu. Vývoj tohoto standardu trval několik let a v roce 2014 dosáhl své finální podoby.

2.3.1 Nové tagy v HTML5

Nové tagy [5] v HTML5 se dají rozdělit do několika kategorií.

- Nové strukturální elementy umožňují detailnější rozčlenění stránky.

  <**article**> : Definuje článek(article) v dokumentu.

  <**figure**> : Definuje vložený obsah, například obrázek, fotku, diagram nebo graf.

  <**header**> **a** <**footer**> : Definuje hlavičku a patičku dokumentu.

- Nové typy vstupů. V klasickém HTML je pouze několik typů vstupních polí, například tlačítko nebo textová oblast. HTML5 přináší mnoho nových možností, například:

  <**color**> : Umožňuje výběr barev.

  <**datetime**> : Umožňuje výběr datumů.

  <**url**> : Umožňuje zadání internetové adresy.

  Mimo typy vstupních polí přidává i mnoho nových atributů jaké můžou vstupní pole mít, například výšku a šířku nebo maximum a minimum (omezení obsahu).

- Přidání grafického obsahu.

  <**svg**> : Slouží pro vložení vektorové grafiky v syntaxi SVG.

  <**canvas**> : Slouží pro kreslení grafiky na webové stránce.

- Přidání médií.

  <**audio**> : Vloží přehrávač zvuku.

  <**video**> : Vloží přehrávač videí.

  <**embed**> : Vloží externí aplikaci (plugin).

2.4 Podpůrné jazyky pro HTML

I přes to, že je HTML často aktualizováno a jsou vydávány verze se stále více rozšířeními, používají se v kombinaci s ním podpůrné jazyky.

Kaskádové styly (CSS) se využívají pro úpravy vizualizace stránky, přidávání speciálních atributů prvkům na stránce, což pomáhá při tvorbě stránek a zlepšuje jejich přehlednost.

Interakci mezi uživatelem a prezentovanými daty zajišťuje jazyk JavaScript, který velmi rozšiřuje použitelnost webových stránek. Z „označkovaného“ statického textu se stává dynamická stránka s obsluhou událostí, kterou je možné použít například pro tvorbu uživatelských rozhraní.

2.4.1 JavaScript

JavaScript [6] je programovací jazyk, který přidává interaktivitu do webových stránek. Například se používá pro hraní her, oživuje tlačítka nebo vykonává animaci. Je možné s ním vytvářet mnoho typů aktivních prvků, od obrázkových galerií, přes animovaní obsahu, po klienta pracujícího nad databází. Hlavní výhodou jazyka je možnost psát podpůrné nástroje postavené na tomto jazyce, které poskytují mnoho užitku za málo úsilí. Mezi ně patří:

- API (Application Programming Interface) zabudované do webových prohlížečů, které poskytují různou funkcionalitu, jako například dynamické generování HTML obsahu nebo přepisování CSS vlastností.
- API třetích stran umístěné na webové stránce umožní používat její funkcionalitu. Příkladem jsou prvky Facebooku na jiných stránkách.
- Frameworky a knihovny třetích stran aplikované na HTML dokáží velmi zrychlit proces tvorby a údržby webu.

Psát kód v jazyce JavaScript je velmi snadné, syntaxe je podobná ostatním rozšířeným jazykům, například Javě. Oproti ní však JavaScript využívá dynamické typy a není tak snadné s ním přistupovat k databázi. Podobné je však užití metod a objektů.

Důležitou funkcí JavaScriptu je zachytávání událostí (eventů). Nejlepší ukázka této funkcionality je v následujícím příkladě.

Příklad 2.2: V příkladě můžeme vidět deklaraci metody, která se zavolá při každém kliknutí na specifikovaný obsah. Její funkcí je, že při kliknutí změní obrázek a při dalším kliknutí jej změní zpět.

```
var myImage = document.querySelector('img');
myImage.onclick = function() {
var mySrc = myImage.getAttribute('src');
if(mySrc === 'images/firefox-icon.png') {
myImage.setAttribute ('src','images/obrazek1.png');
} else {
myImage.setAttribute ('src','images/obrazek2.png');
}
}
```

Tento text je vložen do souboru, který je vedle HTML stránky a má příponu *.js*, která ukazuje, že se jedná o skript v jazyce JavaScript. Následně dojde k propojení pomocí HTML tagu *<script>*.

```
<script type="text/javascript" src="skript.js">
```

Prohlížeč se poté sám postará o exekuci skriptu.

2.4.2 SVG

SVG (Scalable Vector Graphics) [7] je způsob, jak se definuje vektorová grafika. Jedná se o otevřený standard vyvíjeným společenstvím W3C od roku 1999. Mezi jeho dobré vlastnosti patří, že při přiblížení obrázku nedojde ke ztrátě kvality obrázku.

Jeho syntaxe je postavena na základě XML (eXtensible Markup Language). To znamená, že jeho obsah může být indexován, což usnadňuje vyhledávání a navigaci. Také je možné prvky skriptovat a animovat. Další výhodou je jejich snadná úprava, pro úpravu je možno použít jakýkoliv textový editor. Existují však i grafické editory, například Inkscape, které mohou do určité míry práci zjednodušit.

SVG obrázek tvoří různé elementy, například křivky, obdélníky, kruhy, texty, barevné přechody a jiné. Každý element v sobě nese informace o tvaru a umístění. Prvky je možno seskupovat do skupin a manipulovat s nimi hromadně.

Kombinace HTML s jazykem JavaScript otevírá mnoho nových možností. Tagem *<svg>* je možno vložit grafiku přímo do obsahu webové stránky. JavaScript dokáže dynamicky měnit atributy SVG elementů. Tato vlastnost se využívá při animaci objektů [8].

Příklad 2.3: V této ukázce je do HTML obsahu vloženo SVG, ve kterém je obdélník. Pod ním je tlačítko, které po kliknutí spustí JavaScriptovou metodu, která v

dokumentu vyhledá požadovaný element a změní jeho atribut.

```
<svg width="200" height="200">
<rect id="obdelnik1" x="5" y="15" width="40"
height="60" style="stroke:#00ff00; fill:none;"/>
</svg>
<input id="tlacitko1" type="button"
value="Roztahni obdelnik" onclick="zmenAtribut()"/>

<script>
function zmenAtribut() {
document.getElementById("obdelnik1")
.setAttribute("width", "50");
}
</script>
```

2.5 Přínos HTML5 pro HMI

Průmyslové vizualizační systémy HMI (Human Machine Interface) a SCADA (Supervisory Control And Data Acquisition) se využívají již velmi dlouho. O zvětšení jejich potenciálu se postaralo rozšíření operačního systému Windows, protože dodavatelé vizualizačního softwaru se mohli soustředit na vývoj vizualizace pro jednu hlavní platformu.

Výzvou bylo zrealizování vzdáleného přístupu k datům. Nyní však mohou uživatelé přistupovat ke svým datům odkudkoliv, například ze svých osobních počítačů, mobilních telefonů nebo tabletů. Data jsou prezentována nejčastěji formou přehledového okna (Dashboard View), na kterém je vyobrazen řízený proces pomocí KPI (Key Performance Indicator) prvků. Tyto vizualizace jsou k uživateli přenášena pomocí technologie HTML5, nového standardu značkovacího jazyka hojně využívaného v praxi.

Časový vývoj poskytování vzdáleného přístupu k datům
Nejprve se využívala softwarová řešení, nacházející se v těsné blízkosti procesu nebo stroje [9]. Ta často běžela na platformě Windows. Dostupnost dat byla omezená na zařízení připojená do průmyslové sítě, tedy nejdále ve velíně nebo dispečinku.

Aby mohl být proces řízen z jiného místa, bylo potřeba pronajmout datovou linku, která byla dostatečně spolehlivá a poskytovala uspokojivou přenosovou rychlost. Připojení se realizovalo pomocí sítě VPN (Virtual Private Netwok), která umožňovala nastavit přijatelnou úroveň zabezpečení. Její nevýhodou byla nutnost udržovat funkci software i hardware na obou koncích přenosu.

Lepším způsobem přenosu dat se ukázalo býti použití kombinace serveru a webového prohlížeče. Tato technologie je od přelomu tisíciletí používána prakticky všude. Použití všudypřítomného Internetu jako média umožnilo uživatelům přístup ke svým datům odkudkoliv. Další výhodou je, že klient nevyžaduje žádný dodatečný software pro vizualizaci, vše důležité se řeší na HMI/SCADA serveru. Povaha přenosu dat po internetu také nevyžaduje udržování stálého přístupu klienta k internetové síti.

Tento systém fungoval dobře, dokud cílové zařízení pro zobrazení dat byl osobní počítač. Vysílaná vizualizace mohla mít tedy stejnou podobu, jakou má v operátorském PC přímo v řídicí místnosti. Pokud ovšem cílové zařízení je mobil nebo tablet, je třeba upravit podobu zobrazovaných dat, a také přizpůsobit ovládací prvky. Vizualizace na osobním počítači může využívat velké monitory a není třeba tolik šetřit místem. Také se k ovládání používá klávesnice a myš. Oproti tomu na mobilním zařízení, například na PDA, které muže operátor nosit u sebe v kapse, nebo tabletu, je třeba přizpůsobit zobrazovaná data rozměrům displeje a také ovládacím prvkům, kterými může být speciální „tužka“ nebo ovládání pomocí dotyky prstů.

Nahrazení myši a klávesnice dotykovým ovládáním nebylo těžké na implementaci. Nabízelo se nahradit poslouchání událostí kliknutí na myš poklepáním na obrazovku, stejně, jako je to u běžného užívání webového prohlížeče na mobilu a na počítači. Toto se však neukázalo být praktické, protože některé prvky navržené pro ovládání přes počítač se ovládaly dotyky velmi nemotorně.

To způsobilo, že dodavatelé softwaru začali poskytovat různé verze HMI pro osobní počítače a pro dotyková zařízení. Tím se vyřešil problém s funkcionalitou komponent, ale zároveň se výrazně zvýšily nároky na údržbu software, protože při každé změně firmwaru zařízení nebo jejich API bylo nutné obměňovat balíčky určené pro konkrétní cílové modely. Tyto nové verze také bylo nutné pokaždé kompletně testovat.

Nástup HTML5 a jeho přínos pro HMI

Naštěstí se vývojáři dočkali standardu HTML5, na kterém mohli snadněji vyvíjet a udržovat aplikace se vzdálenou projekcí vizualizace řízeného procesu. Při komunikaci mezi lokálním serverem a vzdáleným klientem se využívá HTML5 jako prostředníka a rozhraní. Vývojáři mohou také snadno poskytnout nástroje, díky kterým si mohou sami uživatelé z poskytnutých komponent sestavit svoji vlastní vizualizaci. Protože téměř všechny webové prohlížeče podporují funkce HTML5, výsledná vizualizace se zobrazí správě na všech cílových platformách.

HTML5 tedy přináší do tvorby HMI hned několik výhod.

- Zaručení kompatibility na cílových zařízeních. Nezávislost na operačních systé-

mech, vytvořená vizualizace bude fungovat na Windows stejně jako na Linuxu, Mac OS a mobilních operačních systémech jako Android, iPhone iOS a jiných. Toto je nejdůležitější přínos HTML5.

- Žádný přidaný software není vyžadován na klientské části. Není potřeba adaptéru pro příjem dat. Internetem se posílá pouze výsledná vizualizace a reakce uživatele.
- Snazší přístup k datům mobilními zařízeními. Operátoři mohou pozorovat proces na svém tabletu v terénu bez nutnosti připojení k PC.
- Snazší řešení zabezpečení na mobilních platformách. Komunikace po internetu probíhá na základě relativně bezpečných protokolů.

Při použití HTML5 dojde k automatickému přizpůsobení zobrazovaného obsahu velikosti okna. To znamená, že komponenty vytvořené ve vysokém rozlišení na počítači se při zobrazení na malém telefonu přeorganizují a změní velikost.

HTML5 řeší problémy spojené s umožněním vzdáleného ovládání a přenosem informací. Při každém upgradu vizualizace, jako například vylepšení bezpečnosti nebo změně obsahu, je změna okamžitě přijata serverem a vysílána do cílových zařízení.

Přestože HTML5 značně zjednodušuje celkovou tvorbu a manipulaci s HMI, stále je zde nutnost nastavení a obsluha serveru a také samotná tvorba grafické vizualizace jedním z grafických API podporovaných v HTML5 (SVG, Canvas nebo WebGL) [10]. Zde se otevírá prostor pro komerční firmy, z nichž každá nabízí své vlastní řešení technických otázek.

2.6 Nejčastěji využívané technologie

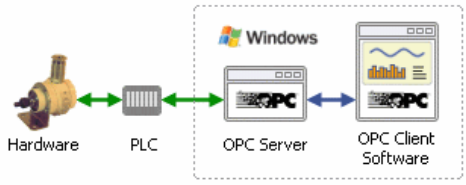
Firmy, například [9], [11] a [12], se snaží přizpůsobit potřebám zákazníka, často tak poskytují více možností jak vyřešit určitou technickou část. Společným rysem je použití OPC standardu pro získání dat z průmyslových PLC (Programmable Logic Controller). Zpřístupnění může probíhat na cloudové službě, která může být součástí software podporujícího běh OPC serveru nebo využívající cloudových serverů třetích stran. Pro zobrazení dat na klientské straně se používá tenký klient ve webovém prohlížeči, webová služby nebo spojení s malou aplikací, například Java Appletem nebo WindowsForms formulářem.

Různí se také přístup k návrhu samotné vizualizace. Některé firmy si tuto službu nechávají pouze pro sebe a vizualizaci sestrojí podle zákazníkových požadavků. Jiné

vydaly vývojová prostředí, ve kterých je možno vizualizace vytvářet.

2.6.1 Technologie pro přenos dat - standard OPC

OPC (Object Linking and Embedding for Process Control) je softwarový standard, který definuje interface(rozhraní), jakým se bude komunikovat s průmyslovými hardwarovými zařízeními. Standard OPC je postaven na jiném standardu, Windows COM [13] (Component Object Model), který definuje interface chování softwarových komponent. OPC tvoří páry programů klient-server [14]. Jelikož je OPC rozšířený průmyslový standard a pouze interface, existuje mnoho způsobů implementace tohoto softwaru a většina firem používajících OPC vytváří vlastní řešení OPC aplikace tak, aby předával informace dále v požadovaném formátu [12]. Dokonce pokud implementace firmy splní test od OPC Foundation, které spravují OPC standard, tak mohou pro svojí implementaci tohoto rozhraní používat logo směrnice OPC Compliance.



Obrázek 2.3: OPC architektura [15].

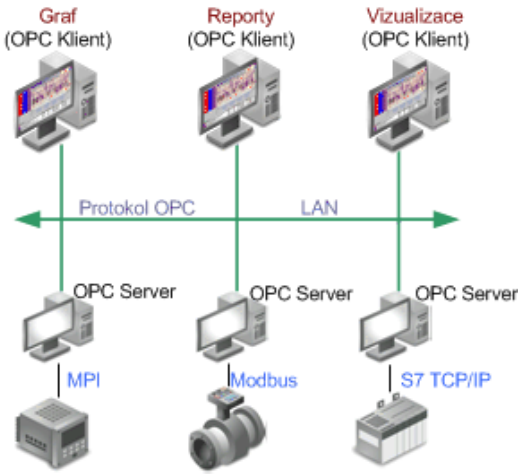
OPC server je software, který překládá informace z protokolu hardwarových komponent do datového formátu OPC. Tato data dále propaguje do vyšších vrstev použitého software.

OPC klient je software, který je využíván pro připojení k OPC serveru a dokáže z něj získat informace nebo posílat příkazy. Získaná data prezentuje uživateli ve formě vizualizací a trendů. Pokud není žádost prezentovat data graficky, je možné data přímo exportovat, například do Microsoft Excel.

OPC charakterizuje požadované chování specifikovaných objektů, jejich vlastnosti a metody, které se starají o připojení, odpojení, zjišťování, zda jsou data přístupná a čtení z dat nebo zápis do dat. OPC servery a klienti jsou nejčastěji implementovany v jazyce C nebo C++.

Toto řešení se může vyskytovat v několika formách [16]. Nejjednodušší je, pokud se klient i server nalézají na jednom počítači. Další variantou je, že jeden OPC server v síti sbírá data z PLC a posílá je po síti Ethernet OPC klientům, například do řídicích center operátorům nebo na mobilní telefony.

V rozsáhlých průmyslových sítích se používá více OPC serverů pro sběr mnoha veličin, které jsou prezentovány na ještě větším počtu OPC klientů.



Obrázek 2.4: OPC na větších sítích [16].

Dříve bylo OPC rozděleno do několika skupin podle tématu, například OPC Data Access, OPC Historical Data Access, OPC Alarms and Events nebo OPC XML-DA. V roce 2004 však bylo toto rozdělení nahrazeno standardem OPC UA(Unified Architecture) [18], které v sobě obsahuje původní specifikace a přidává k nim některé důležité vlastnosti:

- Platformově nezávislý. Původní OPC používalo operace specifické pro Microsoft Windows (COM/DCOM). Tento přístup ovšem měl bezpečnostní problémy. Také byla platformová nezávislost vyžádána velkým počtem serverů běžícím na operačním systému Linux a rozšiřováním aplikace HMI na mobilní zařízení s operačním systémem Android.
- Zlepšení zabezpečení. Transportní protokoly nyní umožňují chytřejší šifrování dat. Umožnění přenosu po protokolech HTTPS (HyperText Transfer Protocol Secure) a SOAP (Simple Object Access Protocol).
- Standardizace. Protože je OPC UA standardizovaný pod IEC-62541, je snadnější propojení mezi zařízeními od různých výrobců.
- Zjednodušení. Při změně nebo přidání obsahu na klasickém OPC bylo potřeba měnit server i klienta na mnoha úrovních (v rámci OPC Data Access, OPC Historical Data Access, ...), díky spojení těchto standardů je změna obsahu daleko rychlejší.

2.6.2 Cloudová řešení pro SCADA systémy

Cloudové řešení se využívá, pokud není možno zaručit stabilní přenos mezi OPC servery a klienty. Cloud-computing servery jsou služby od třetích stran, které poskytují své výpočetní kapacity zákazníkům. Ti tento prostor využívají pro přístup a hosting pro své servery, sítě, úložiště, aplikace a služby. Provozující zaručuje, že tato data budou dostupná vždy na vyžádání a o veškerou infrastrukturu se stará provozovatel. Cloudová řešení umožňují zákazníkům připojení přes jakékoliv mobilní zařízení vybavené internetem, komunikaci přes SMS zprávy a emaily.

Firmy, které se zabývají tvorbou a údržbou vizualizace (a řízením), se tedy přizpůsobují potřebám a možnostem zákazníka. Ten může a nemusí mít k dispozici svojí vlastní síťovou infrastrukturu s veřejnou IP adresou a zabezpečením, která je vhodná pro instalaci SCADA systému. Často tak dochází ke kombinacím cloudových řešeních s klasickými servery. Mnoho firem má také své vlastní cloudové servery.

Cloudová řešení poskytují mnoho variant, jak kombinovat běh SCADA systému [19]. Cloudy mohou být privátní nebo veřejné, sloužit jednomu zákazníkovi nebo více, existují také varianty, které sdruží více cloudů dohromady, každý s jinými vlastnostmi.

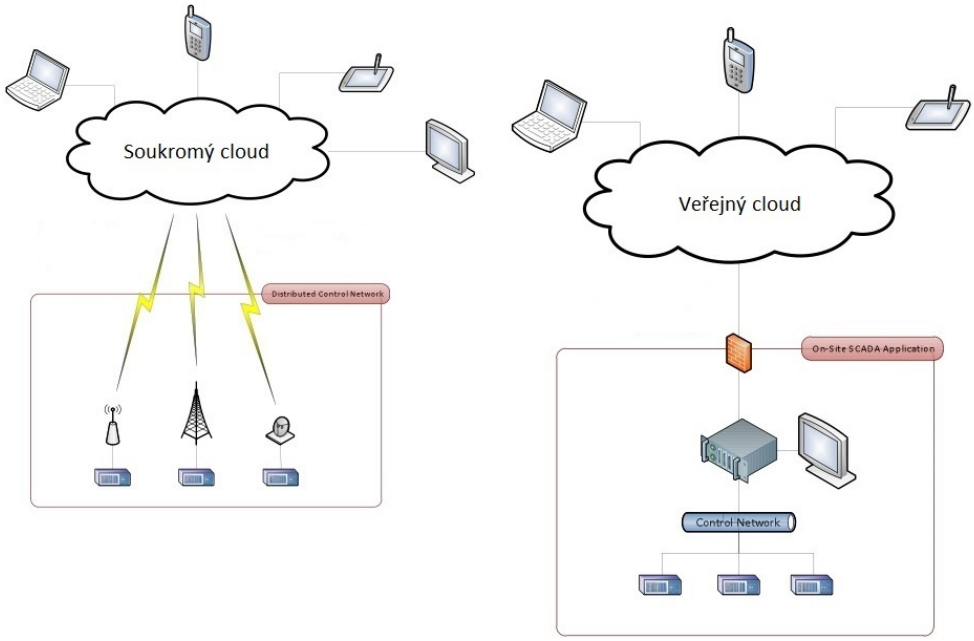
Cloud může podporovat SCADA ve dvou režimech:

- SCADA běží v řídicím systému a výstupy z ní jsou propagovány do cloudového úložiště.
- SCADA běží na cloudovém uložišti a je vzdáleně připojená k řídicím systémům.

Nevýhody cloudových řešení

Migrací ze zabezpečeného lokálního síťového obvodu do cloud-computiongového úložiště se svěřuje přístup ke všem citlivým datům poskytovateli služby. To vyžaduje rozhodnutí, jakým způsobem budou data chráněna a jak k nim bude přistupováno. Citlivá data se někdy podaří odkrýt i nedopatřením, například chybou konfigurace.

Cloudové služby jsou také oblíbeným terčem hackerských útoků. Sice se snaží těmto útokům bránit ale i přes to se útočníkům podaří DDoS (Distributed Denial of Service) útokem, který spočívá v přehlcení serveru požadavky a odepření přístupu běžným uživatelům. Často se také hacker nejdříve vydává za zájemce o systém, aby mohl zevnitř odhalit slabiny systému.



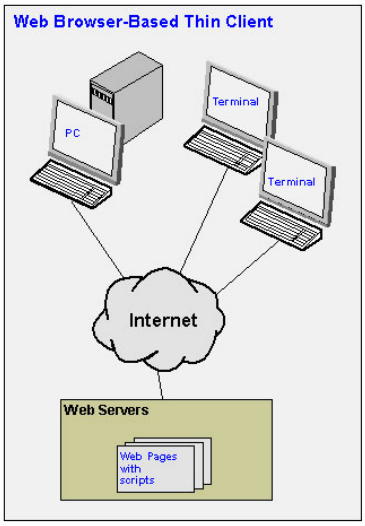
Obrázek 2.5: Umístění SCADA systému [19].

Výhody cloudových řešení
Výhody cloudových řešení se dají shrnout do několika bodů:

- Snadné řízení aktualizací a bezpečnostních záplat. Nové verze jsou uvedeny do služby během několika minut.
- Pokud se jedná o veřejnou službu, tak je možné zobrazit data na libovolném zařízení na celém internetu.
- Jsou spolehlivé a podporují i velké objemy dat, protože cloudové servery často běží na více nezávislých strojích najednou. Také zálohování probíhá na více místech.
- Je možné snadno navyšovat kapacitu.
- Není třeba řešit technické otázky nebo licence kolem použitých hardwarových a softwarových komponent.

2.6.3 Vizualizace v tenkém klientovi

Když jsou data zpřístupněna pomocí OPC klienta, je třeba je předložit uživateli. Dnes jsou často využívá „tenkého klienta“. Jedná se o aplikaci, která přenechává výpočet dat serveru a její náplní je pouze prezentace uživateli. Tencí klienti se realizují ve webových prohlížečích, které zajišťují komunikaci se serverem pomocí protokolu HTTP. Pro vytváření obsahu webových stránek se používá jazyk HTML5. Pro přidání další logiky, například prvků uživatelského rozhraní, se používá skriptovacího jazyka JavaScript. Zásadní výhodou je obsluha softwaru vývojáři, protože místo dvou aplikací je veškerá logika centralizována na serveru. Pokud se vše důležité nalézá pouze na serveru, zlepšují se podmínky zabezpečení a také je možné dělat zálohy celého systému bez potřeby stažení dat ze software na straně klienta.



Obrázek 2.6: Provedení tenkého klienta [20].

2.6.4 Základní rysy tvorby webových HMI v HTML5

Základem tvorby webových stránek je použití jazyka HTML, dnes velmi často verze 5. Definuje se jím rozložení celé stránky, nastavuje se, jaká data se budou zobrazovat, kde jsou přístupná a konfigurují se všechny prvky uživatelského rozhraní.

Rozčlenění stránky do sekcí

Pro lepší orientaci při návrhu webové stránky se obsah rozdělí podle požadované struktury. Na nejvyšší vrstvě se například oddělí menu, lišta nástrojů, odkazy na jiné části webu a na samotnou vizualizaci. Samotná vizualizace se dále může například dělit na přehledové okno, ovládací prvky a grafickou reprezentaci procesu. V následujícím příkladu je ukázáno, jakým způsobem se stránka může dělit do sekcí pomocí HTML5 a jakým způsobem se používají atributy *class* a *id*.

Příklad 2.4: Klasické rozložení vizualizace použitím HTML5. Je patrné rozložení do několika částí.

```
<body>
...
<div class="menu">...</div>
<div class="panel_nastroju">...</div>
<div class="reference">...</div>
<div class="vizualizace">
    <div class="prehledove_okno">
                <div id="polozka1">...</div>
                <div id="polozka2">...</div>
        </div>
        <div class="ui">
                <div id="tlacitko1">...</div>
                <div id="tlacitko2">...</div>
        </div>
        <div class="grafika">
                <img id="obrazek1" ...>
                <img id="obrazek2" ...>
        </div>
</div>
</body>
```

Atributy *class* a *id* slouží jako „háčky“, které umožňují navigaci mezi všemi prvky. Rozdíl mezi nimi je ten, že atribut *class* může být stejný pro mnoho elementů(tagů). Tak vzniká skupina objektů jedné třídy. Této třídě se pak snadno pomocí CSS dodají požadované vizualizační parametry. Oproti tomu *id* by měl být unikátní pro každý element, slouží jako identifikační klíč. Aby se zamezilo kolizím, jsou *id* systematicky přiřazována například podle pořadí na stránce nebo jiných vlastností.

Příklad 2.5: Ukázka použití atributu id, kde dojde k zavolání metody getElementById(), která vrátí kolekci prvků, které jsou tímto id označeny.

```
var tlacitko = document.getElementById("tlacitko");
```

Konfigurace v hlavičce stránky

Po vytoření HTML5 obsahu v části *body*, je třeba dále uvést doplňující informace od sekce *head*. Mimo běžný obsah, který je v každé webové stránce, tedy uvedení nadpisu, jednoduchých skriptů v jazyce JavaScript a metadat, je třeba právě zde dodat informace, které jsou zapotřebí pro konfiguraci vizualizace. Také se do části *head* stránky přidávají odkazy na použité JavaScriptové knihovny a CSS styly. Ukázka tohoto vložení je v následujícím příkladu, kde dochází k propojení jazyků HTML, JavaScript a CSS.

Příklad 2.6: Ukázka hlavičky HTML dokumentu, kde dochází ke konfiguraci spojení a zabezpečení, k importu JavaScriptových knihoven a vložení CSS souborů.

```
<head>
...
<script type="text/javascript">
konfigurace = {
token_zabezpeceni:'1e5dfgb130-476d-151-c78a-ba904d2',
serverURL: 'http://192.168.0.1:8000'
};
</script>
...
<script type="text/javascript" src="js/hmi-lib.js"></script>
...
<link rel="stylesheet" type="text/css" href="css/hmi-style.css"/>
</head>
```

Režie pomocí JS knihoven

Jednou z hlavních částí produktů firem zabývajících se tvorbou HMI v HTML5 je JS knihovna, která je zároveň páteří celé logiky za vizualizací. Na jedné straně jsou data v běžných datových typech zpřístupněna OPC klientem nebo webovou službou, na cloudu nebo jinde. Na druhé straně čekají na propojení s těmito daty HTML prvky na stránce.

Pomocí konfigurace, kterou jsme nastavili v hlavičce stránky, se knihovna připojí k serveru, který jí poskytne vyžádaná data. JavaScript poskytuje nástroje, jakými je možno poslouchat a reagovat na změny. Tyto akce aktivují metody, které propagují data do HTML a grafického obsahu na stránce.

HTML a grafické prvky však musí být označeny, aby knihovna věděla, jaká data jsou spojena s kterou reprezentací. Při vytváření stránky je potřeba specifikovat, s jakými daty jsou propojená jednotlivá *id*. Tato musí být obsažena ve vytvořené knihovně v jazyce JavaScript, která je určená pro animaci vizualizace.

Při změně informace na zdroji tedy knihovna dokáže rozpoznat, jaký webový obsah se váže na tato data a správně upraví HTML a grafický obsah, aby odpovídal realitě. Reakce uživatele jsou realizovány obráceným způsobem, kdy interakce s uživatelským rozhraním spustí metodu knihovny, která podle typu vstupu předá informace dále směrem k řízenému procesu.

Přístupy k realizaci grafických prvků

Existují dva typy obrázků, rastrové a vektorové. Zatímco u rastrového typu je v souboru uložena informace, jakou má který pixel barvu, u vektorové grafiky je obsaženo více informací o tom, jaké obrazce jsou na obrázku, například že se jedná o červený kruh, a prohlížeč vygeneruje na základě této informace pixelovou reprezentaci.

Poněkud zastaralý, ale stále obvyklý způsob, jak animovat objekt, je pomocí série rastrových obrázků. Například otáčení ventilátoru je realizováno třeba desíti obrázky s různou polohou lopatek. Ty jsou uspořádány do smyčky a po opakovaném časovém intervalu se obsah přepíše následujícím obrázkem, až tak vznikne iluze pohybu.

Příklad 2.7: Ukázka vložení rastrového obrázku. Rychlá změna obrázku může vyvolat dojem plynulé animace.

```
...
<div id="animovana_komponenta1">
<img src="obrazky/ventilator/1.png" id="1_ventilator1" >
</div>
...
```

K tomuto HTML kódu přísluší část v JavaScriptové knihovně, kde je definováno, že každý časový interval se má inkrementovat hodnota v atributu *src* komponenty s *id 1_ventilator1*, pokud se nemá vynulovat.

Alternativou k tomuto přístupu je použití vektorové grafiky. Hodnoty se zde mění daleko jednodušeji, elegantněji a uložené obrázky zabírají méně paměti.

Příklad 2.8: Ukázka vložení jednoduché vektorové grafiky přímo do kódu HTML.

```
...
<div id="animovana_komponenta1">
<svg>
<g transform="rotate(1)" id="lopatky_ventilatoru">
... svg obrazek lopatek ventilatoru ...
</g>
</svg>
</div>
...
```

K tomuto HTML, který obsahuje SVG prvek, je přiřazen kód, který každou periodu změní argument ve funkci *rotate()* u elementu, kde je *id* rovno *lopatky_ventilatoru*.

3 Praktická část

3.1 Analýza možností vizualizace Motion Control příkladů

3.1.1 Analýza příkladů na MotionControl

Základní instalace programu RexDraw obsahuje skupinu příkladů, která předvádí možnosti řídicího systému a funkci některých bloků. Část této skupiny ilustruje použití bloků z knihovny Motion Control. Protože cílem práce je vytvořit grafické komponenty, které mají být součástí vizualizace sady příkladů, je vhodné hledat společné rysy, které se vyskytují v příkladech z oblasti Motion Control, analyzovat je a definovat požadavky na základní sadu komponent pro vizualizaci příkladů.

Po zběžném prozkoumání příkladů bylo rozhodnuto, že je výhodné vytvořit tři základní komponenty. Jedna bude sloužit pro obsluhu osy, druhá pro obecné příkazy pohybů a třetí pro převodovku.

3.1.2 Blok osy - Axis

Hlavním vizualizačním blokem je blok reprezentující veličiny osy, *RM_Axis*. Existuje několik základních požadovaných vlastností a hodnot, které jsou stejné pro všechny použité osy a které je vhodné zobrazovat.

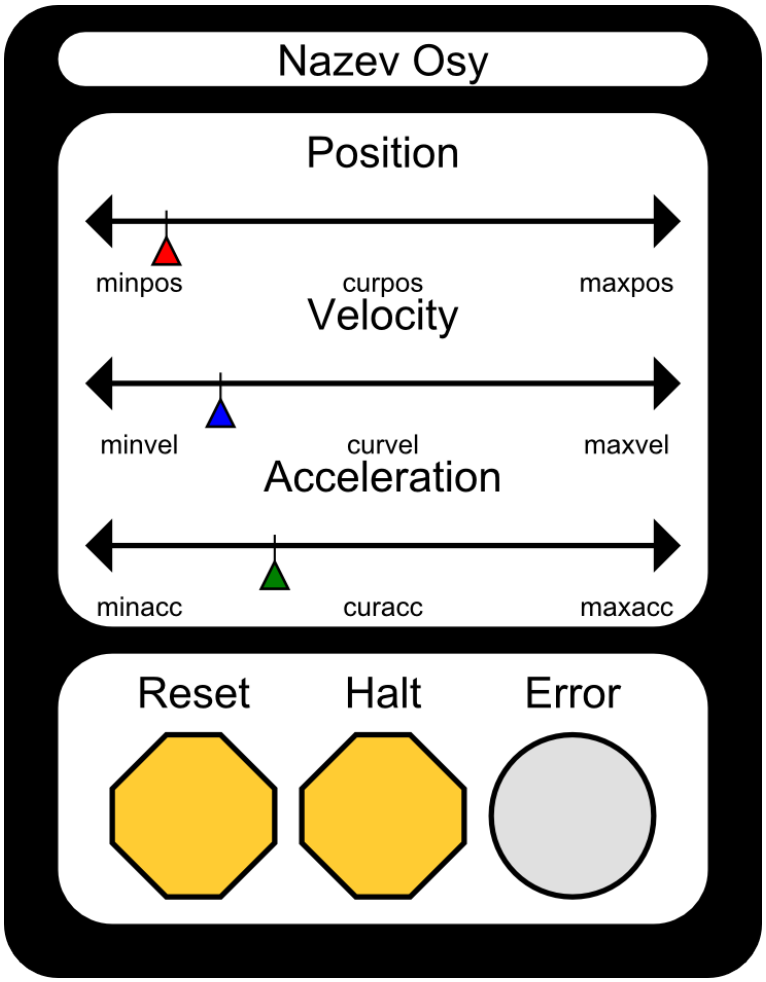
Název Osy : Každá osa má svůj název, který nemusí být stejný jako název bloku ve schématu, ale může více odrážet význam osy.

Sledované Veličiny : Každá osa má pozici, rychlost a akceleraci. Minimální a maximální hodnota je vypsána vhodným způsobem. Současná hodnota má kromě výpisu i grafické zobrazení, například formou bodu cestujícího po úsečce, která představuje přípustné hodnoty osy.

Tlačítko Reset : Sice to není základní funkce osy, ale v nových příkladech je umožněno vynulování veličin osy a odstranění chyb. V původních příkladech není tato funkce vytvořena, proto je v nich po vzniknutí chyby třeba nahrát celý model znovu.

Tlačítko Halt : Opět to není obsaženo v samotném bloku osy, avšak možnost pozastavit jakýkoliv pohyb prováděný na ose je šikovná.

Indikátor Chyby : Pokud vznikne na ose chyba, je třeba zřetelně zobrazit, že k této události došlo.



Obrázek 3.1: Návrh grafické reprezentace vizualizačního bloku Axis.

3.1.3 Blok příkazu pohybu - MotionCommand

Mnoho příkazů pohybů (*MoveAbsolute*, *MoveRelative*, *MoveVelocity*, atd.) má několik podobných rysů. Není třeba tedy vymýšlet zobrazení každého bloku zvlášť, postačí reprezentace vlastností obecných pro všechny typy.

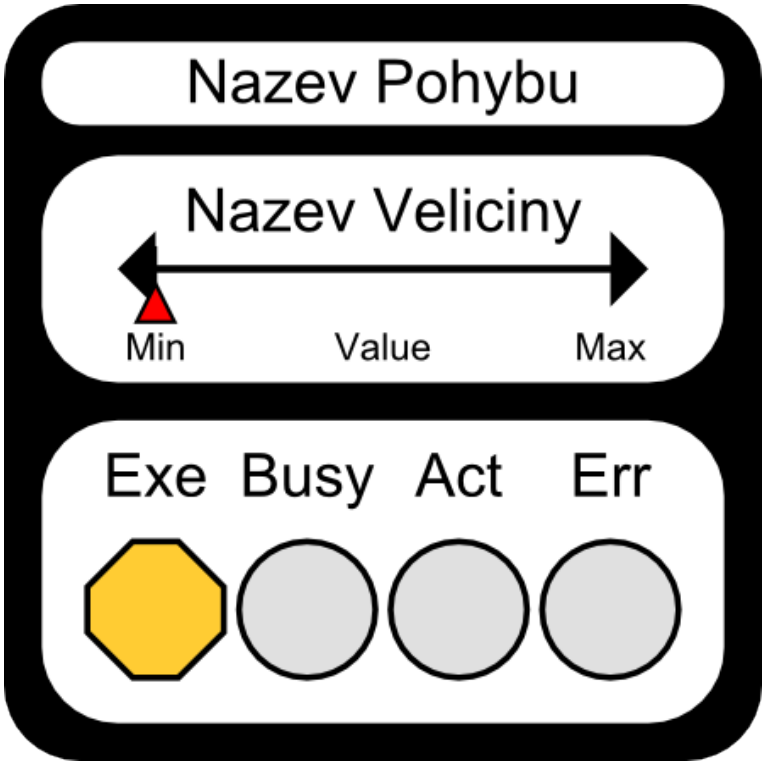
Název Pohybu : Každý pohyb má svůj název, který nemusí být stejný jako název bloku ve schématu, ale může více odrážet jeho význam.

Požadovaná hodnota : Bloky příkazů fungují tak, že se pokusí přesunout konkrétní veličinu osy na požadovanou hodnotu. Touto veličinou může být pozice, rychlost i zrychlení. Blok příkazu pohybu tedy zobrazí, s jakou veličinou blok manipuluje a jaká je jeho konkrétní hodnota.

Minimum a Maximum : Protože blok přísluší vždy jen k jedné ose, je vhodné zjistit minimální a maximální hodnotu osy pro manipulovanou veličinu. Tato hodnota je vypsána. Ve výsledku zde opět je úsečka s bodem reprezentující polohu požadované hodnoty mezi minimem a maximem osy.

Tlačítko Execute : Některé bloky se mohou spouštět automaticky, v tom případě tato funkce nebude nakonfigurována. Pokud se spouští pomocí tlačítka *MP*, jako je tomu ve vytvořených příkladech, potom je součástí vizualizace příkazu pohybu i toto tlačítko.

Indikátory : Blok může být najednou v několika stavech, může být aktivní (Active), spuštěný (Busy) nebo chybový (Error). Tyto stavy je vhodné zřetelně signalizovat.



Obrázek 3.2: Návrh grafické reprezentace vizualizačního bloku MotionCommand.

3.1.4 Blok převodovky - GearBox

Tato vizualizace slouží pro reprezentaci bloku *MC_GearIn*, který se používá pro propojení dvou os, hlavní (Master) a vedlejší (Slave), kdy pohyby na hlavní ose způsobují stejný pohyb na vedlejší ose ovlivněný převodovým poměrem.

Název Převodu : Převod má svůj název, který je zobrazený a nemusí se shodovat s názvem bloku.

Názvy Os : Obě osy, Master i Slave, mohou mít názvy, odlišné od názvů bloků os, které lépe vystihují jejich významy.

Převodový Poměr : Blok obsahuje hodnoty čitatele a jmenovatele převodového zlomku, který je dobré zobrazit.

Tlačítko Execute : Někdy je možné blok spustit automaticky, potom tato funkce nebude nakonfigurována. Pokud se spouští pomocí tlačítka *MP*, jako je tomu v příkladech, potom je součástí vizualizace převodovky toto tlačítko.

Indikátor GearIn : Pokud je blok spuštěný a aktivně ovlivňuje vedlejší osu, je vhodné toto indikovat.

Indikátor Error : Pokud během běhu bloku došlo k chybě, je vhodné jí zřetelně zobrazit.

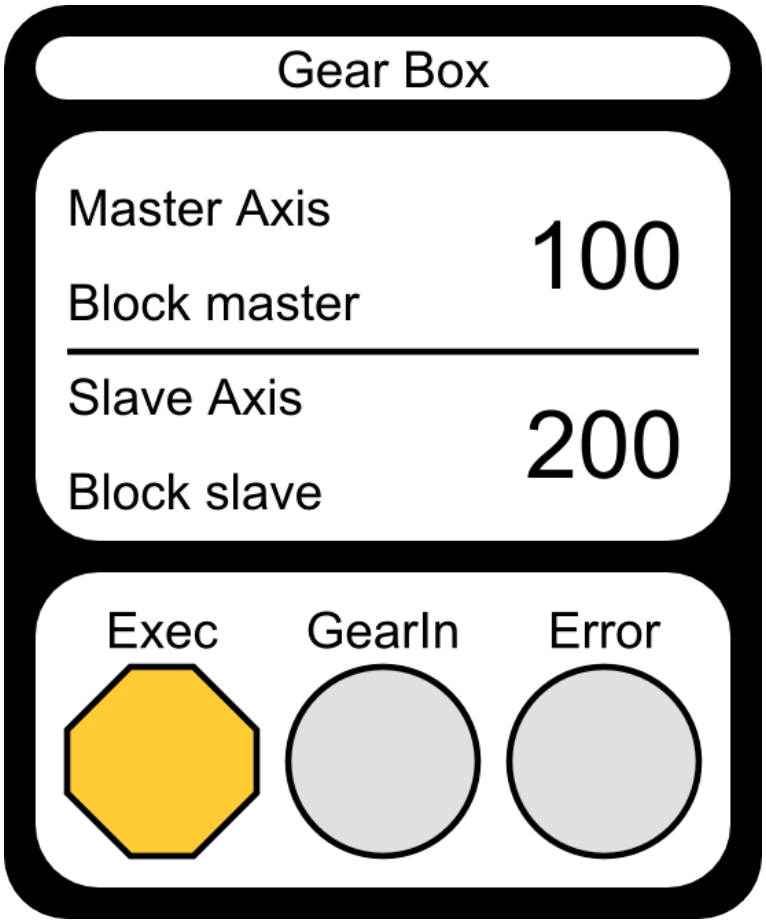
V této části byly definovány vlastnosti, které splňuje grafická vizualizace bloku osy (*RM_Axis*), převodovky (*MCP_GearIn*) a příkazu pohybu (*MCP_Move...*).

3.2 Strukturální Koncept

Zde je popsáno, z jakých částí se skládá každý vytvořený příklad. Také je uvedeno, jaké jsou potřebné konfigurační informace pro jednotlivé typy vizualizačních bloků a jakým způsobem konfigurace probíhá.

3.2.1 Složení příkladu

Každý příklad se skládá z několika částí. Nejdůležitější je soubor *.rex*, který obsahuje schéma modelu exekutivy a úlohy (*task*). Dále jsou ve složce *www* obsažené soubory spojené s vizualizací. Hlavním souborem je *index.html*, který obsahuje definici celé stránky, SVG obrázky a odkaz na několik potřebných knihoven v jazyce JavaScript



Obrázek 3.3: Návrh grafické reprezentace vizualizačního bloku GearBox.

pro propojení s jádrem RexCore. Vizualizace se nahrává automaticky společně s exekutivou, pokud je v exekutivě obsažen blok *WWW*.

3.2.2 Obsah a rozložení v *index.html*

Soubor *index.html* se skládá z několika částí.

Stejně jako v již vytvořených původních příkladech, hlavička (<*head*>) obsahuje mnoho informací. Například je zde obsažen hlavní nadpis stránky, odkazy na loga REXu online, CSS styly a na přiložené knihovny v jazyce JavaScript.

Hlavní přidanou částí v hlavičce je vytvoření konfiguračních objektů, které se využijí při inicializaci ve vytvořeném skriptu *mc-block-animation.js*.

V těle (<*body*>) stránky se nachází viditelný obsah. V oddíle, který je určen pro vlastní obsah, tedy v *div* s *id="content"* je připravena tabulka (<*table*>), která slouží pro pojmutí dvou částí vizualizace.

Do první části je vložena vizualizace vygenerována z programu RexDraw (vysvětleno dále).

Ve druhé části se nachází SVG obsahující vizualizaci sestavenou z navržených bloků. Tento obrázek je možné obohatit o další animační prvky uživatelského rozhraní v programu Inkscape s rozšířením RexHMI.

3.2.3 Konfigurace a inicializace

Protože je vytvářena vlastní vizualizaci pro tři typy bloků, je třeba navrhnout tři typy konfiguračních objektů. Pro každý typ konfigurace je třeba definovat sadu parametrů.

Konfigurační objekt bloku Axis

Název modelu : Použije se při tvorbě řetězců pro přípojné body.

Název bloku *RM_Axis* : Název bloku v modelu, ke kterému se vztahuje tato vizualizace. Použije se ve tvorbě řetězců pro přípojné body.

Nadpis : Řetězec který se zobrazí ve vizualizaci na pozici nadpisu.

Tlačítko reset : Název bloku *MP* v modelu, které slouží k resetování osy.

Tlačítko halt : Název bloku *MP* v modelu, které slouží k pozastavení aktivity osy.

ID vizualizace : Identifikátor SVG skupiny, která reprezentuje tuto vizualizaci.

Konfigurace je použita v metodě *initAxis()*, ve které se nastavují přípojné body pro animaci.

Příklad 3.1: Příklad konfigurace vizualizačního bloku Axis. Tento kód je vložen do oblasti se skriptem v <*head*>.

```
var axis1Config={
model:"moveAbsoluteExample",
blok:"RM_Axis",
nazev:"Hlavni Osa",
tl_reset:"MP3",
tl_halt:"MP4",
```

```
id_vizualizace:"axis1"
}
initAxis(axis1Config);
```

Konfigurační objekt bloku MotionCommand

Název modelu : Použije se ve tvorbě řetězců pro přípojné body.

Nadpis : Řetězec který se zobrazí ve vizualizaci na pozici nadpisu.

Veličina : Název parametru, který určuje kam se má osa přesunout. Tedy například cílová rychlost (*Velocity*), cílová poloha (*Position_*) nebo relativní posun v poloze (*Distance*).

Blok osy : Název bloku osy ve schématu. Použije se pro získání minimální a maximální hodnoty.

Tlačítko aktivace : Název bloku *MP*, které je přivedeno na vstup *Execute*. Pokud se blok spouští automaticky, je možné jej nechat prázdný, nebo ho odkázat na nepřipojené *MP*.

Název bloku : Název bloku v modelu, který je reprezentován touto vizualizací.

ID vizualizace : Identifikátor SVG skupiny, která reprezentuje tuto vizualizaci.

Veličina Osy : Přípustnými hodnotami jsou { „pos“, „vel“ nebo „acc“}. Jedná se o zkratky *Position*, *Velocity* a *Acceleration*. Tato informace je důležitá, protože je nutné vědět, jakou veličinou bude příkaz manipulovat, aby bylo možné přečíst minimum a maximum správné veličiny na ose.

Konfigurace se aktivuje v metodě *initMotionCommand()*.

Příklad 3.2: Příklad konfigurace vizualizačního bloku MotionCommand. Metoda *initMotionCommand()* je definována v přiloženém souboru *mc-block-animation.js*, jak je popsáno dále.

```
var motion1Config={
model:"moveAbsoluteExample",
nadpis:"Move Absolute 1",
velicina:"Position_",
blok_osy:"RM_Axis",
blok_aktivace:"MP1",
nazev_bloku:"MCP_MoveAbsolute1",
id_vizualizace:"command1",
velicina_osy:"pos"}
initMotionCommand(motion1Config);
```

Konfigurační objekt bloku GearBox

Název modelu : Použije se ve tvorbě řetězců pro přípojné body.

Název bloku *RM_GearIn* : Název bloku v modelu, ke kterému se vztahuje tato vizualizace.

Tlačítko Execute : Název bloku *MP*, které je přivedeno na vstup *Execute*. Pokud se blok spouští automaticky, je možné jej nechat prázdný, nebo ho odkázat na nepřipojené *MP*.

Nadpis Řetězec který se zobrazí ve vizualizaci na pozici nadpisu.

ID vizualizace Identifikátor SVG skupiny , která reprezentuje tuto vizualizaci.

Název Master osy Zobrazí se jako statický text v části popisující hlavní osu, nemusí odpovídat reálnému názvu bloku.

Název Slave osy Zobrazí se jako statický text v části popisující vedlejší osu, nemusí odpovídat reálnému názvu bloku.

Konfigurace se pak aktivuje v metodě *initGearBox()*.

Příklad 3.3: Příklad konfigurace vizualizačního bloku GearBox. Tento vizualizační blok nepotřebuje ke svému chodu znát názvy bloků os, ke kterým se vztahuje.

```
var gear1Config={
model:"gearInExample",
blok:"MCP_GearIn1",
tl_exec:"MP2",
nadpis:"Prevod 1",
id_vizualizace:"gearBox1",
nazev_master:"prvni",
nazev_slave:"druhy"}
initGearBox(gear1Config);
```

3.2.4 Struktura SVG obrázku

Do zvolené části je vložen obrázek SVG, kterému je třeba nadefinovat určité vlastnosti.

SVG obrázek si dopočítá své rozměry, aby vyplňoval obsah obalujícího objektu.

```
<td width=900px height=700px>
```

V celém SVG je výchozí nastavení zarovnání textu na střed. Důležitý je parametr *viewBox*. Ten udává, jaké velký rozsah souřadnic bude zobrazen v SVG. V následujícím příkladě je možné používat souřadnice na X-ose od 0 do 400 a na Y-ose od 0 do 370. Tento interval se pak vykreslí do oblasti definované absolutním rozměrem, tedy 900px na 700px.

```
<svg text-anchor="middle"  viewBox="0 0 400 370"  >
```

Dovnitř SVG je možné vkládat objekty představující vizualizace bloků. Každý vizualizační blok je umístěn do párových tagů (*<g> - group*) vymezující skupinu se stejnými vlastnostmi. Tato skupina musí mít atribut *id* stejný jako je *id_vizualizace* v konfiguračním objektu, který se vztahuje k této skupině. *Id* slouží později v inicializační metodě při selekci této skupiny. Druhým atributem skupiny je *transform="translate(x,y)"*, který udává relativní posunutí vizualizačního bloku v obrázku SVG.

```
<g id="axis1" transform="translate(0,0)">
...
</g>
<g id=motionCommand1" transform="translate(150,0)">
...
</g>
```

3.3 Vypracování SVG komponent

V této sekci jsou kompletní zdrojové kódy SVG objektů, které byly vytvořeny podle navrhovaných parametrů.

3.3.1 SVG zápis Axis

Grafická reprezentace tohoto SVG je na obrázku 3.1.

```
<svg text-anchor="middle" viewBox="0 0 140 180">
<g id="axis1" transform="translate(0,0)">
<rect x="0" y="0" width="140" height="180" fill="black" ry="10" rx
   ="10"/>
<rect x="10" y="5" width="120" height="10" fill="white" ry="5" rx="5"/>
<rect x="10" y="20" width="120" height="95" fill="white" ry="10" rx
   ="10"/>
<rect x="10" y="120" width="120" height="50" fill="white" ry="10" rx
   ="10"/>
<g transform="translate(35,150)">
<polygon points="5,15 15,5 15,-5 5,-15 -5,-15 -15,-5 -15,5 -5,15" fill
   ="#ffcc33" stroke="black" stroke-width="1" id="reset"/>
```

```
</g>
<g transform="translate(70,150)">
<polygon points="5,15 15,5 15,-5 5,-15 -5,-15 -15,-5 -15,5 -5,15" fill
   ="#ffcc33" stroke="black" stroke-width="1" id="halt"/>
</g>
<circle cx="105" cy="150" r="15" fill="#e0e0e0" stroke="black" stroke-
   width="1" id="error"/>
<text x="70" y="13" font-size="8" id="nazev_osy">Nazev Osy</text>
<text x="35" y="130" font-size="8">Reset</text>
<text x="70" y="130" font-size="8">Halt</text>
<text x="105" y="130" font-size="8">Error</text>
<polygon points="15,40 20,35 20,45" fill="black"/>
<polygon points="125,40 120,35 120,45" fill="black"/>
<line x1="19" y1="40" x2="121" y2="40" stroke-width="1" stroke="black
   "/>
<polygon points="15,70 20,65 20,75" fill="black"/>
<polygon points="125,70 120,65 120,75" fill="black"/>
<line x1="19" y1="70" x2="121" y2="70" stroke-width="1" stroke="black
   "/>
<polygon points="15,100 20,95 20,105" fill="black"/>
<polygon points="125,100 120,95 120,105" fill="black"/>
<line x1="19" y1="100" x2="121" y2="100" stroke-width="1" stroke="black
   "/>
<text x="70" y="30" font-size="8">Position</text>
<text x="70" y="60" font-size="8">Velocity</text>
<text x="70" y="90" font-size="8">Acceleration</text>
<text x="25" y="53" font-size="5" id="min_pos">minpos</text>
<text x="25" y="83" font-size="5" id="min_vel">minvel</text>
<text x="25" y="113" font-size="5" id="min_acc">minacc</text>
<text x="115" y="53" font-size="5" id="max_pos">maxpos</text>
<text x="115" y="83" font-size="5" id="max_vel">maxvel</text>
<text x="115" y="113" font-size="5" id="max_acc">maxacc</text>
<text x="70" y="53" font-size="5" id="cur_pos">curpos</text>
<text x="70" y="83" font-size="5" id="cur_vel">curvel</text>
<text x="70" y="113" font-size="5" id="cur_acc">curacc</text>
<g id="jezdec1" transform="translate(10,0)">
<polygon points="17.5,48 22.5,48 20,43" fill="red" stroke="black"
 stroke-width="0.5"/>
<line x1="20" y1="43" x2="20" y2="38" stroke="black"
stroke-width="0.4"/>
</g>
<g id="jezdec2" transform="translate(20,0)">
<polygon points="17.5,78 22.5,78 20,73" fill="blue" stroke="black"
stroke-width="0.5"/>
<line x1="20" y1="73" x2="20" y2="68" stroke="black" stroke-width
   ="0.4"/>
</g>
<g id="jezdec3" transform="translate(30,0)">
<polygon points="17.5,108 22.5,108 20,103" fill="green" stroke="black"
```

```
stroke-width="0.5"/>
<line x1="20" y1="103" x2="20" y2="98" stroke="black" stroke-width
   ="0.4"/>
</g>
</g>
</svg>
```

3.3.2 SVG zápis MotionCommand

Grafická reprezentace tohoto SVG je na obrázku 3.2.

```
<svg text-anchor="middle" viewBox="0 0 120 145">
<g id="motionCommand1" transform="translate(0,0)">
<rect x="0" y="0" width="120" height="145" fill="black"
ry="10" rx="10"/>
<rect x="5" y="5" width="110" height="10" fill="white"
ry="5" rx="5"/>
<rect x="5" y="20" width="110" height="65" fill="white"
ry="10" rx="10"/>
<rect x="5" y="90" width="110" height="50" fill="white"
ry="10" rx="10"/>
<g transform="translate(25,120)">
<polygon points="5,15 15,5 15,-5 5,-15 -5,-15 -15,-5 -15,5 -5,15"
 fill="#ffcc33" stroke="black" stroke-width="1" id="exec"/>
</g>
<circle cx="60" cy="120" r="15" fill="#e0e0e0"
stroke="black" stroke-width="1" id="ingear"/>
<circle cx="95" cy="120" r="15" fill="#e0e0e0"
stroke="black" stroke-width="1" id="error"/>
<text x="25" y="102" font-size="8">Exec</text>
<text x="60" y="102" font-size="8">GearIn</text>
<text x="95" y="102" font-size="8">Error</text>
<text x="60" y="13" font-size="8" id="nazev_bloku">Gear Box</text>
<text x="10" y="35" text-anchor="start" font-size="8">Master Axis</text
   >
<text x="10" y="65" text-anchor="start" font-size="8">Slave Axis</text>
<text x="10" y="50" text-anchor="start" font-size="8" id="master">Block
    master</text>
<text x="10" y="80" text-anchor="start" font-size="8" id="slave">Block
   slave</text>
<text x="95" y="45" font-size="15" id="num">100</text>
<text x="95" y="75" font-size="15" id="den">200</text>
<line x1="10" x2="110" y1="55" y2="55" stroke-width="1" stroke="black
   "/>
</g>
</svg>
```

3.3.3 SVG zápis GearBox

Grafická reprezentace tohoto SVG je na obrázku 3.3.

```
<svg text-anchor="middle" viewBox="0 0 120 145">
<g id="gearBox1" transform="translate(0,0)">
<rect x="0" y="0" width="120" height="145"
 fill="black" ry="10" rx="10"/>
<rect x="5" y="5" width="110" height="10"
 fill="white" ry="5" rx="5"/>
<rect x="5" y="20" width="110" height="65"
 fill="white" ry="10" rx="10"/>
<rect x="5" y="90" width="110" height="50"
 fill="white" ry="10" rx="10"/>
<g transform="translate(25,120)">
<polygon points="5,15 15,5 15,-5 5,-15 -5,-15 -15,-5 -15,5 -5,15"
 fill="#ffcc33" stroke="black" stroke-width="1" id="exec"/>
</g>
<circle cx="60" cy="120" r="15" fill="#e0e0e0"
 stroke="black" stroke-width="1" id="ingear"/>
<circle cx="95" cy="120" r="15" fill="#e0e0e0"
 stroke="black" stroke-width="1" id="error"/>
<text x="25" y="102" font-size="8">Exec</text>
<text x="60" y="102" font-size="8">GearIn</text>
<text x="95" y="102" font-size="8">Error</text>
<text x="60" y="13" font-size="8" id="nazev_bloku">Gear Box</text>
<text x="10" y="35" text-anchor="start"
font-size="8">Master Axis</text>
<text x="10" y="65" text-anchor="start"
font-size="8">Slave Axis</text>
<text x="10" y="50" text-anchor="start"
font-size="8" id="master">Block master</text>
<text x="10" y="80" text-anchor="start"
font-size="8" id="slave">Block slave</text>
<text x="95" y="45" font-size="15" id="num">100</text>
<text x="95" y="75" font-size="15" id="den">200</text>
<line x1="10" x2="110" y1="55" y2="55" stroke-width="1"
 stroke="black"/>
</g>
</svg>
```

3.4 Tvorba inicializačního skriptu

Celý skript je uložen v souboru *mc-block-animation.js* a je zapsán v jazyce JavaScript. Skládá se ze tří hlavních částí; jedna pro každý typ bloku. Mnohokrát se zde opakuje několik typů funkcí a konstrukcí, je tedy popsán každý typ, poté pořadí a argumenty, se kterými se volají.

3.4.1 Vkládání statického textu

Jedním typem úkonu je vložení textu z konfigurace na určité místo, který se po dobu běhu příkladu nebude měnit. Příklady statického textu mohou být nadpis bloku nebo pojmenování os u převodovky (GearBox).

Příklad 3.4: Příklad vložení názvu do nadpisu. V proměnné *config* je uložena konfigurace, v proměnné *vizualizace* pak SVG obsah týkající se daného vizualizačního bloku.

```
function vlozNadpisMotion(config,vizualizace){
var text_el=$("#nazev_pohybu",vizualizace);
text_el.html(config.nadpis);
}
```

3.4.2 Svázání textu s veličinou z procesu

Další záležitostí je svázání textu s proměnnou přímo z REXu. K tomuto účelu se využívá knihovna *REX.HMI*, která poskytuje způsob jak asynchronně reagovat na změnu v datech. Z konfigurace se sestaví řetězec přípojného bodu. Pak se definuje funkce, která se zabývá samotnou reakcí na podnět.

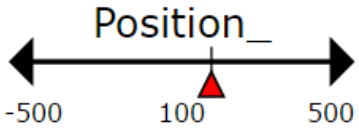
Příklad 3.5: Příklad svázání proměnné s textem. Zde se při každé změně v procesu *RM_Axis* změní vypsaná hodnota maximální akcelerace ve vizualizaci.

```
function propojMaximumAccelerationAxis(config,vizualizace){
REX.HMI.addItem(config.id_vizualizace+"-max_acc",
config.model+"."+config.blok+":"+"MaxAccelerationAppl" , 'R');
REX.HMI.$i(config.id_vizualizace+"-max_acc")
        .on("change", function(itm) {
        var text_el=$("#max_acc",vizualizace);
        text_el.html(itm.getValue().toFixed(0));
});}
```

3.4.3 „Jezdci“

Jedním z hlavních prvků vizualizačních bloků Axis a MotionCommand je grafická reprezentace toho, jaká je hodnota ve vztahu k maximu a minimu osy.

Například, pokud minimum osy je -500, maximum osy je +500 a současná hodnota je +100, tento jezdec na ose se bude nacházet v poloze 60% zleva, jako je tomu na obrázku 3.4.



Obrázek 3.4: Osa s Jezdcem.

Příklad 3.6: Funkce pro výpočet a změnu absolutní polohy jezdce podle současné pozice.

```
function aktivujJezdcePosition(val,config,vizualizace){
var jezdec = $("#jezdec1", vizualizace);
var delkaCilovehoZobrazeni = 100;
var min = $("#min_pos",vizualizace).html();
var max = $("#max_pos",vizualizace).html();
if((0>=min-val) && (0<=max-val)){
var hodnotaSouradnic = ((val-min)*delkaCilovehoZobrazeni)/(max-min);
jezdec.attr("transform", "translate(" +hodnotaSouradnic+ ",0)");}}
```

Pokud se hodnoty mění velmi rychle, což u souvislého pohybu lze očekávat, vznikne iluze plynulého přesunu.

3.4.4 Tlačítka

Tlačítka jsou reprezentována osmiúhelníky, které mění barvu při najetí myši (*hover*) a při stisknutí myši (*mousedown*). Vše je vytvořeno asynchronním posluchačem událostí nad tímto SVG objektem, který po stisknutí tlačítka zapíše logickou „1“ pomocí knihovny *REX.HMI* tam, kam je upřesněno z konfigurace. Tato funkce funguje pouze, když je cílový blok typu MP (Manual Pulse, blok tedy obsahuje parametr *BSTATE*).

Příklad 3.7: Příklad propojení tlačítka Execute v GearBox. Knihovna *REX.HMI* se stará o přenos dat mezi vizualizací a modelem.

```
function propojExecuteGear(config,vizualizace){
REX.HMI.addItem(config.id_vizualizace+"-exec",
```

```
         config.model+"."+config.tl_exec+":BSTATE", 'W');
var circle_el = $("#exec",vizualizace);
circle_el.hover(function(){
        circle_el.css({fill:"#33CCFF"});},
                                function(){
        circle_el.css({fill:"#FFCC33"});});
circle_el.mousedown(function(){
        circle_el.css({fill:"#3366FF"});});
circle_el.mouseup(function(){
        circle_el.css({fill:"#33CCFF"});
        REX.HMI.write(config.id_vizualizace+"-exec",1);});}
```

3.4.5 Indikátory

Indikátory jsou reprezentovány kruhy, které mění barvu podle stavu proměnné. Opět se používá knihovna *REX.HMI*, kde se využívá asynchronního poslouchání událostí. Zde se předpokládají pouze hodnoty logické „0“ a „1“. Pokud je „0“, indikátor se deaktivuje a barva se změní na šedou. Pokud je „1“, barva se změní na zelenou (aktivita) nebo červenou (chyba).

Příklad 3.8: Ukázka propojení „zeleného“ indikátoru. CSS atribut *transition* způsobí, že se nová hodnota aplikuje plynuje, což vypadá daleko příjemněji.

```
function propojBusyMotion(config,vizualizace){
REX.HMI.addItem(config.id_vizualizace+"-busy",
         config.model+"."+config.nazev_bloku+":Busy", 'R');
REX.HMI.$i(config.id_vizualizace+"-busy").on("change", function(itm) {
var circle_el = $("#busy",vizualizace);
if(itm.getValue().toFixed(0) == 1){
        circle_el.css({fill: "#00F53D", transition: "0.5s"});
}else{
circle_el.css({fill: "#e0e0e0 ", transition: "0.5s"});}});}
```

3.4.6 Aktivace objektu a volání metod

Když se v *index.html* zavolají metody inicializace, v *mc-block-animation.js* to aktivuje připravenou posloupnost výše zmíněných funkcí, které zajistí propojení všech veličin s korespondujícími body v SVG vizualizaci.

Příklad 3.9: Ukázka posloupnosti aktivace MotionCommand bloku. Ostatní bloky se aktivují obdobně. Aby nedošlo k záměně názvů metod, všechny metody které se týkají MotionCommand bloku mají koncovku „Motion“, podobně u Axis a GearBox.

```
function initMotionCommand(config){
```

```
var vizualizace=$("#"+config.id_vizualizace);
vlozNadpisMotion(config,vizualizace);
vlozNazevVelicinyMotion(config,vizualizace);
propojMinimumOsyMotion(config,vizualizace);
propojMaximumOsyMotion(config,vizualizace);
propojCilovouHodnotuMotion(config,vizualizace);
propojBusyMotion(config,vizualizace);
propojActiveMotion(config,vizualizace);
propojErrorMotion(config,vizualizace);
propojExecMotion(config,vizualizace);}
```

3.5 Obohacení obsahu

3.5.1 Generování schématu z RexDraw

Celá vizualizace by byla nekompletní, pokud by v ní bylo „pouze“ několik vizualizačních bloků. Jedním ze způsobů, jak přidat další důležité prvky do vizualizace, je generování schématu přímo z programu RexDraw. Postup je velmi snadný. V hlavním menu je možnost „Export to HTML..“, která vytvoří soubory s vizualizací, tedy *index.html* a složky s potřebnými dodatečnými obrázky, styly a skripty. Po spuštění stránky se automaticky propojí s hodnotami z procesu a je tedy takto vytvořeno velmi efektivní uživatelské rozhraní.

Po nahlédnutí do vygenerovaného souboru, kde je cílová vizualizace (například *moveAbsoluteExample.html*) je vidět SVG obrázek vložený do * * s *id=”rex-dialog-container”*. Tento obsah (včetně *div*) je možné ze stránky vyjmout a vložit jej do libovolné jiné webové stránky, která obsahuje REXovský obsah, tedy základní kolekci skriptů, stylů a rozložení stránky. Funkcionalita těchto bloků je tedy zachována. Toto oživené schéma je součástí každého vytvořeného příkladu, protože doplňuje grafické zobrazení vytvořených vizualizačních bloků.

3.5.2 Generování animace z Inkscape RexHMI

Další zajímavou možností je obohacení o grafické ovládací prvky, které je možné přidat v programu Inkscape s rozšířením RexHMI. Podrobný návod jak pracovat s tímto nástrojem a jeho popis je přístupný on-line na webových stránkách RexControls [21]. Tyto prvky je možné kombinovat s vytvořenými vizualizačními bloky, tak jako je tomu například v *gearInExample*.

Příklad *gearInExample* vznikl tak, že byla vytvořena v Inkscape běžným způsobem REX vizualizace. Byly do ní vloženy připravené grafické ovládací prvky a jejich

konfigurace. Pak byl otevřen zdrojový kód tohoto SVG a na konec grafické části byly přidány vytvořené vizualizační bloky. Po vložení prvků je vhodné zkontrolovat celkovou vizuální kompozici. Pak byla ze souboru SVG exportována vizualizace, která obsahuje *index.html* a ostatní soubory. Do hlavičky souboru *index.html* třeba vložit potřebné konfigurační objekty a použít aktivační metodu vytvořeného skriptu. V souboru *index.html* je opět obsažen SVG obrázek, který je možno vyjmout a používat ho v jiných REX vizualizacích.

3.6 Shrnutí příkladů

3.6.1 Postup tvorby příkladů

V minulých kapitolách byly popsány jednotlivé elementy, ze kterých jsou tvořeny příklady k základním blokům knihovny Motion Control. Těmito elementy je myšleno popsání struktury *index.html*, použití inicializačního skriptu *mc-block-animation.js* a popis obsahu SVG komponent. Tvorba vizualizace příkladu, který obsahuje schéma vygenerované z RexDraw a SVG obohacené v Inkscape zabere nemnoho času.

Prvním krokem je tvorba modelu a získání *index.html* souboru podle RexHMI šablony. Toho může být například dosaženo použitím některého stávajícího příkladu nebo vizualizace vygenerované z RexDraw, ze které je odebráno vše odkazující se na konkrétní příklad. K tomuto HTML souboru budou zapotřebí samozřejmě i složky se skripty, styly a obrázky.

Do složky se skripty musí být vložen inicializační skript *mc-block-animation.js*. Tento skript musí být také importován do <*head*> v *index.html*. Do <*body*> stránky je vložena tabulka <*table*> o dvou řádcích, která slouží pouze k pojmutí objektů. Je možné použít i jinou strukturu, například <*div*>. Tomuto objektu je vhodné definovat, jak je velký. SVG pak automaticky vyplní jeho prostor.

Do první části tabulky je vložen obsah SVG z kódu vygenerovaného v RexDraw. Je vhodné vymazat z hlavičky <*svg*> informace o výšce a šířce v pixelech (*height* a *width*).

Do druhého řádku tabulky pak patří obsah SVG z kódu vytvořeného v Inkscape. Opět je vhodné vymazat z tagu <*svg*> informace o výšce a šířce (*height* a *width*). Vizualizační bloky, které jsme vložili, mají *id*, které je třeba změnit a zapamatovat si jej pro konfiguraci. Také mají parametr *translate=transform(0,0)*, který je třeba změnit tak, aby odpovídal požadovanému posunutí bloku po X a Y souřadnicích v rámci SVG.

Dalším krokem je vytvoření konfigurace v <*head*> stránky a zavolání konfigu-

rační metody z inicializačního skriptu *mc-block-animation.js*. Poté by příklad měl být hotový.

Součástí této práce je i šest příkladů, které byly touto metodou vytvořeny.

moveAbosluteExample : Ukázka použití bloku *MCP_MoveAbslute*.

movRelativeExample : Ukázka použití bloku *MCP_MoveRelative*.

moveAdditiveExample : Ukázka použití bloku *MCP_MoveAdditive*.

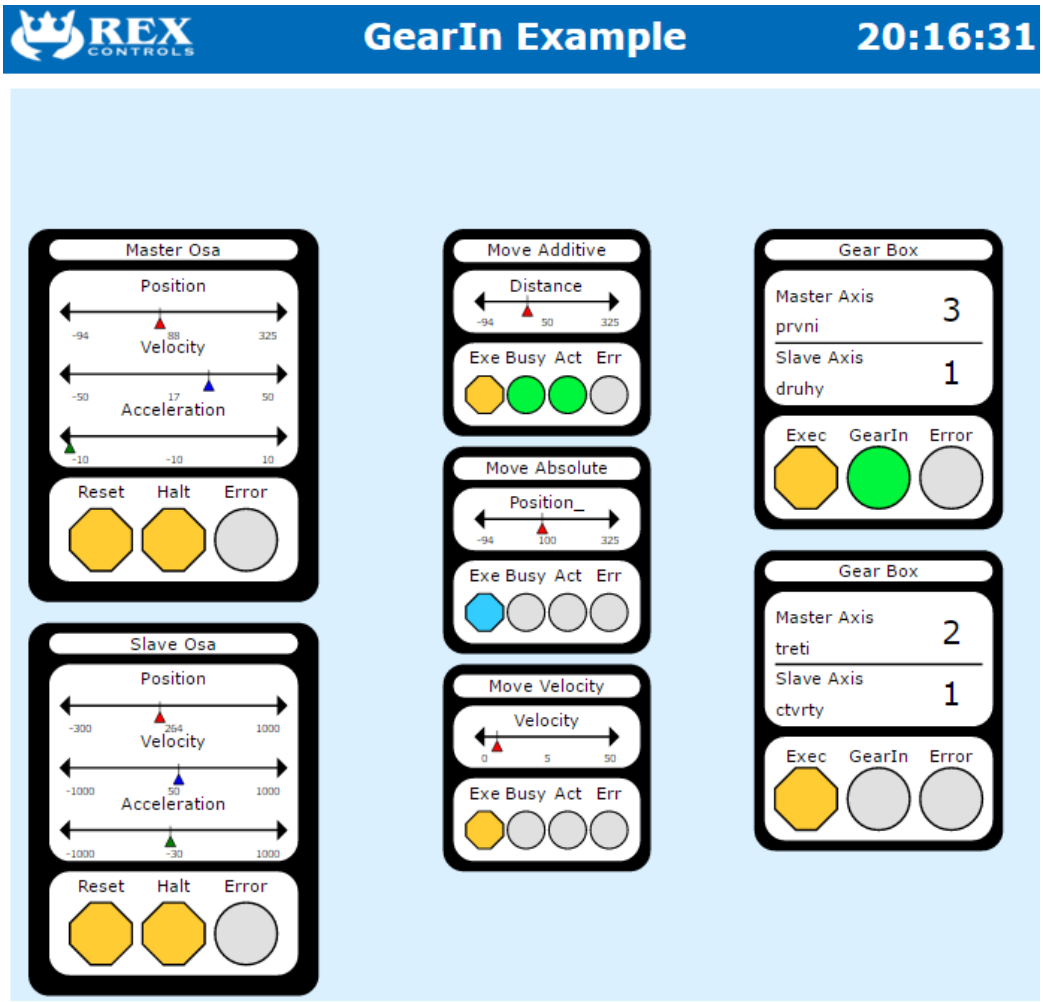
moveVelocityExample : Ukázka použití bloku *MCP_MoveVelocity*.

gearInExample : Ukázka použití bloku *GearIn* a všech předchozích bloků. Slouží jako souhrnný příklad.

camInExample : Ukázka použití bloku *CamIn* a *CamTableSelect*. Ukázka využití Inkscape RexHMI komponent.

3.7 Ukázka běhu

Po spuštění je vizualizace přístupná na cílové adrese. Všechny prvky jsou funkční. Protože je použito obecných technologii SVG a HTML, výsledná vizualizace se přizpůsobí přiblížení a rozlišení displeje na cílovém zařízení. Proto tato vizualizace je snadno zobrazitelná na všech platformách, které jsou propojeny j jejím zdrojem, například formou domácí či podnikové sítě. Nezáleží, zda je klient ve Windows nebo Linuxovém zařízení, nebo na mobilu či tabletu s OS Android.



Obrázek 3.5: Část vizualizace GearInExample.

4 Závěr

V této práci byl čtenář seznámen se základními postupy a technologiemi pro tvorbu grafických vizualizací řízených procesů. Na začátku proběhlo seznámení s jazyky webových stránek. Také byla popsána struktura řídicího systému REX.

Potom jsem vytvořil porovnání různých přístupů k vizualizaci a přenosu dat. Zde jsem procházel webové stránky firem zabývajících se vizualizací a HMI, snažil jsem se získat co nejvíce informací z poskytnutých materiálů a prospektů. V těchto technologických popisech jsem hledal společné rysy, ty se pokusil zapsat a porovnat rozdílné přístupy k tvorbě uživatelské vizualizace.

Protože jsem se rozhodl, že budu vytvářet vizualizaci pro sadu příkladů z oblasti Motion Control, bylo nutné se seznámit s obsahem této knihovny a také s příklady obsaženými v základní instalaci programu REX. Před tím, než bylo možné začít tvořit vizualizaci, bylo potřeba vytvořit novou sadu modelů, která ilustruje použití požadovaných bloků v příkladech. Nakonec jsem se spokojil s šesti příklady, které předvádějí základní bloky z knihovny Motion Control.

Nejobtížnější z celé práce bylo vymyšlení struktury a vlastností vizualizace. Napoprvé jsem samozřejmě neodhadl správný způsob ani styl provedení. Po vytvoření úspěšného konceptu v podobě MotionCommand vizualizačního bloku jsem byl schopen velmi rychle realizovat zbylé dva navržené vizualizační bloky. Věřím, že ve stejném stylu, jako jsou vytvořené bloky, je možné velmi rychle vytvořit další nové typy bloků, pokud by byl takový požadavek. Stačí se pouze držet návrhového postupu, kterému podléhala i tvorba vytvořených vizualizačních bloků.

Literatura

[1] REX CONTROLS s.r.o . *O společnosti* [online]. [cit. 10.2.2016]. Dostupný na WWW: https://www.rexcontrols.cz/o-spolecnosti

[2] REX CONTROLS s.r.o . *Co je to řídicí systém REX?* [online]. [cit. 10.2.2016]. Dostupný na WWW: https://www.rexcontrols.cz/rex

[3] REX CONTROLS s.r.o . *Funkční bloky systému REX* [online]. [cit. 11.2.2016]. Dostupný na WWW: https://www.rexcontrols.cz/media/DOC/CZECH/BRef_CZ.pdf

[4] PLCOPEN. *Motion Control* [online]. [cit. 20.3.2016]. Dostupný na WWW: http://www.plc.org/pages/tc2_motion_control/

[5] W3 SCHOOLS. *HTML5 New Elements* [online]. [cit. 1.3.2016]. Dostupný na WWW: http://www.w3schools.com/html/html5_new_elements.asp

[6] W3 SCHOOLS. *JavaScript Tutorial* [online]. [cit. 1.3.2016]. Dostupný na WWW: http://www.w3schools.com/js/

[7] W3 SCHOOLS. *SVG Tutorial* [online]. [cit. 1.3.2016]. Dostupný na WWW: http://www.w3schools.com/svg/

[8] JENKOV Jakob. *SVG Scripting* [online]. [cit. 1.3.2016]. Dostupný na WWW: http://tutorials.jenkov.com/svg/scripting.html

[9] ADVANTECH. *Evolving HMI/SCADA Software to HTML5 Business Intelligence Dashboard* [online]. [cit. 3.3.2016]. Dostupný na WWW: http://www.automation.com/pdf_articles/WebAccess8.0_Whitepaper_1031.pdf

[10] MICROSOFT. *WebGL, Canvas, or SVG? Choose the right API* [online]. [cit. 4.3.2016]. Dostupný na WWW: https://msdn.microsoft.com/en-us/library/dn265058(v=vs.85).aspx

[11] RELIANCE. *Struktura Systému* [online]. [cit. 11.3.2016]. Dostupný na WWW: https://www.reliance.cz/cs/products/reliance4-scada-hmi-system\#page=structure

[12] PROSYS OPC. *Prosys OPC UA Web Client released* [online]. [cit. 11.3.2016]. Dostupný na WWW: https://www.prosysopc.com/blog/prosys-opc-ua-web-client-released/

[13] MICROSOFT. *COM Clients and Servers* [online]. [cit. 20.3.2016]. Dostupný na WWW: https://msdn.microsoft.com/en-us/library/windows/desktop/ms683835(v=vs.85).aspx

[14] HONEYWELL INTERNATIONAL INC. *What is an OPC Server and an OPC Client?* [online]. [cit. 1.4.2016]. Dostupný na WWW: http://www.matrikonopc.com/resources/opc-server.aspx

[15] COGENT REAL-TIME SYSTEMS INC. *What is OPC?* [online]. [cit. 10.4.2016]. Dostupný na WWW: http://www.opcdatahub.com/WhatIsOPC.html

[16] FOXON S.R.O. *Prosys OPC UA Web Client released* [online]. [cit. 10.3.2016]. Dostupný na WWW: https://www.prosysopc.com/blog/prosys-opc-ua-web-client-released/

[17] KONDOR, Randy. *Understanding OPC: Basics for new users* [online]. [cit. 10.3.2016]. Dostupný na WWW: http://www.iebmedia.com/index.php?id=4467\&parentid=63\&themeid=255\&showdetail=true

[18] FREJBORG, Andreas; OJALA, Martti; HAAPANEN, Lauri. *OPC UA Connects your Systems* [online]. [cit. 10.2.2016]. Dostupný na WWW: https://downloads.prosysopc.com/downloads/automation_xx_seminar_opcua_connects_your_systems.pdf

[19] COMBS, Larry. *Cloud Computing for SCADA | Control Engineering* [online]. [cit. 29.3.2016]. Dostupný na WWW: http://www.controleng.com/single-article/cloud-computing-for-scada/8a6ea192e60cc626d2e3fc3a5686396d.html

[20] ACNODES. The Types of Thin Client and HMI Visualization Application | Acnodes Corporation [online]. [cit. 18.4.2016]. Dostupný na WWW: http://www.acnodes.com/blog/the-types-of-thin-client-and-hmi-visualization-application/

[21] REX CONTROLS. *Grafické HMI pro automatizaci bazénu pomocí Raspberry Pi* [online]. [cit. 15.4 .2016]. Dostupný na WWW: `https://www.rexcontrols.cz/clanky/graficke-hmi-pro-automatizaci-bazenu-pomoci-raspbe`