ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

Katedra řídicí techniky

Rozšíření webových stránek laboratoře K26

Bakalářská práce

Praha 2009

Student: **Jan Hájek**

Vedoucí práce: **Ing. Jiří Roubal, Ph.D.**

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 30. června 2009 …….…….

podpis

Poděkování

Především bych chtěl poděkovat Ing. Jiřímu Roubalovi, Ph.D., vedoucímu této bakalářské práce, za rady, návrhy, připomínky a ochotu. Také děkuji Aleši Kapicovi za odborné konzultace týkající se autentifikace uživatelů pomocí protokolu LDAP.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jan Hájek**

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný
Obor: Kybernetika a měření

Název tématu: **Rozšíření webových stránek laboratoře K26**

Pokyny pro vypracování:

1. Naprogramujte v PHP stránky pro administrátorský přístup na stránkách laboratoře.
2. Naprogramujte přístup na stránky laboratoře pomocí fakultního serveru usermap.
3. Vytvořte popis k dalšímu modelu ve stylu, ve kterém jsou již modely popsány (text, fotografie, videa).
4. Navrhněte a naprogramujte nějakou motivační internetovou hru pro budoucí studenty, která souvisí s modelováním a řízením (s laboratorní výukou).

Seznam odborné literatury:

Dorf, R. C. and Bishop, R. H. Modern Control Systems, 11. vydání, Prentice Hall, 2007, ISBN-10: 0132270285, ISBN-13: 978-0132270281.
Petr Horáček, Systémy a modely, Praha 2000
Jiří Kosek, PHP - tvorba interaktivních internetových aplikací, Grada publishing s. r. o., 1999
Welling Luke, Laura Thomson, PHP a MySQL - rozvoj webových aplikací, 2003

Vedoucí: Ing. Jiří Roubal, Ph.D.

Platnost zadání: do konce letního semestru 2008/09

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 27. 2. 2009

Abstrakt

Tato bakalářská práce popisuje úpravy provedené na internetových stránkách Laboratoře teorie automatického řízení K26. Jako první jsou zde popsány úpravy textů a zdrojového kódu stránek. Poté je vysvětlen princip a postup programování motivační hry. Na závěr popisuje autentifikaci uživatelů pomocí školní sítě na Karlově náměstí.

Absctract

This bachelor thesis describes modifications that were made to the website of the Laboratory of Automatic Control Theory K26. At first text and source code changes of the web are described here. Then thesis explains the principle and practice of programming motivational game. Finally, it describes authentication of users using the school network at the Charles square.

Obsah

Seznam obrázků

Seznam výpisů

1. Úvod

Tato práce navazuje na bakalářskou práci [1], vypracovanou na Katedře řídicí techniky v roce 2008, týkající se zhotovení webových stránek Laboratoře teorie automatického řízení K26 [2] v budově E, Fakulty elektrotechnické Českého vysokého učení technického v Praze, na Karlově náměstí 13.

Pro účely naplnění zadání jsou zde využity programovací jazyky především pro webové technologie, a to JavaScript a PHP. Pro obohacení obsahu stránek je také nutné HTML (*HyperText Mark-up Language*) společně s CSS (*Cascading Style Sheeds*). Pro přehlednost jsou zdrojové kódy v následujícím textu umístěny v blocích jako je Výpis 1.1. Zvýraznění syntaxe bylo vytvořeno pomocí [14].

```
<?php
$promenna = 'retezec';
echo "jiny retezec";                                    //komentář
?>
<script>
function novaFunkce(argument){
  if (argument == 0) argument = "hodnota";
  return argument;
}
</script>
```

Výpis 1.1: Vzor zdrojového kódu

Cílem mé bakalářské práce je obohatit internetovou prezentaci Laboratoře teorie automatického řízení K26 [2]. Hlavním bodem je realizace motivační hry, která má za úkol představit potenciálním zájemcům o studium krásu automatického řízení. Pro tento účel byl zvolen fyzikální model „kulička na tyči“, který se fyzicky nachází v laboratoři K26. Náplní hry je snaha o ruční regulaci tohoto modelu. Výsledek je pak porovnán s automatickým řízením. Další body se týkají doplnění stručného popisu fyzikálního modelu v českém i anglickém jazyce a realizace autentifikace uživatelů stránek pomocí školní sítě na Karlově náměstí.

Tato práce je organizována následujícím způsobem. V kapitole 2 jsou popsány změny provedené na stránkách Laboratoře teorie automatického řízení K26 [2]. Tato kapitola se dále dělí na úpravy vzhledu a kódu stránek a na popis Tepelné soustavy. V kapitole 3 je popsána motivační hra – kulička na tyči, matematický popis modelu, jeho řízení a následné modelování pomocí JavaScriptu. V poslední kapitole je popsána autentifikace uživatelů pomocí školní sítě.

2. Úpravy stránek

Tato kapitola popisuje grafické úpravy a programové doplňky internetových stránek Laboratoře teorie automatického řízení K26 [2]. V první podkapitole jsou definovány použité pojmy, následující podkapitola je věnována úpravám vzhledu, další úpravám zdrojového kódu stránek, poslední pak doplnění popisu modelu tepelné soustavy.

2.1. Názvosloví

Pro následující text jsou důležité tyto pojmy: URL adresa, GET proměnná a administrátorské rozhraní. Tyto pojmy budou proto nejprve vysvětleny.

2.1.1. URL adresa

URL adresa je zkratkou pro Uniform Resource Locator, což je řetězec znaků, který slouží k přesné specifikaci umístění zdrojů informací na internetu. Typicky je složen z protokolu, doménového jména, portu, požadovaného dokumentu a parametrů. Funkci URL adresy a jejích jednotlivých částí nejlépe objasní praktický příklad:

URL adresa dle Výpisu 2.1 znázorňující volání hlavní strany internetových stránek Laboratoře teorie automatického řízení K26 [2] pomocí internetového prohlížeče, který přeloží zdrojový kód do požadované podoby. Lze ji rozložit následujícím způsobem:

protokol: http

doménové jméno: support.dce.felk.cvut.cz

port: 80

dokument: lab26/index.php

parametry: ?page=main

```
http://support.dce.felk.cvut.cz:80/lab26/index.php?page=main
```

Výpis 2.1: Příklad URL adresy včetně portu

Je-li port implicitní pro daný protokol (např. pro http je to port 80), nemusí být v zápisu URL uveden. URL adresa ve Výpisu 2.2 v tomto případě ukazuje na stejný dokument jako adresa

z Výpisu 2.1 s tím rozdílem, že pro případné zapamatování je tento zápis praktičtější a pro drtivou většinu uživatelů internetu známější.

```
http://support.dce.felk.cvut.cz/lab26/index.php?page=main
```

Výpis 2.2: Příklad URL adresy

2.1.2. GET proměnné

GET proměnnou jsou nazvány parametry URL adresy. Tento pojem je zde využit pro odlišení lokálních proměnných a těch, které přišly jako parametry URL v požadavku na zobrazení dokumentu. Název je odvozen od HTML formulářů a také od adresace těchto proměnných v PHP.

Součástí HTML jsou formuláře, které může návštěvník vyplnit a odeslat. HTML element *form,* označující HTML formulář, má atribut *method*, který může nabývat hodnot *post* a *get*. Je-li hodnota atributu *method* nastavena na *get*, odešlou se data vyplněného formuláře jako parametry URL adresy. Odtud tedy pojem *GET proměnná.*

2.1.3. Administrátorské rozhraní

Administrátorské rozhraní je neveřejná sekce stránky vyžadující autorizaci, tedy ověření identity uživatele s využitím přihlašovacího jména a hesla. Uvnitř tohoto rozhraní lze upravovat či měnit chování webu bez zásahu do zdrojového kódu stránky.

2.2. Úpravy vzhledu stránek laboratoře K26

Na stránkách Laboratoře teorie automatického řízení K26 [2] byly provedeny drobné grafické úpravy jako změna barvy pozadí či stylu odkazů. Nejdůležitější estetickou, ale i praktickou změnou je prodloužení viditelné části stránky na celou výšku prohlížeče dle rozlišení uživatele, díky tomu bylo dosaženo větší přehlednosti oproti původní, fixní výšce. Aby nebylo příliš zasaženo do struktury stránek, byla tato změna provedena pomocí technologie JavaScript. Zdrojový kód funkce včetně popisu činnosti jednotlivých řádků ukazuje Výpis 2.3.

Pro zjednodušení přístupu k jednotlivým prvkům stránky, byla ve Výpisu 2.3 využita knihovna jQuery [4]. Její hlavní výhodou je zajištění funkčnosti adresace objektů ve většině běžně používaných internetových prohlížečů. Tato knihovna také nabízí funkce dalece rozšiřující možnosti internetových prezentací.

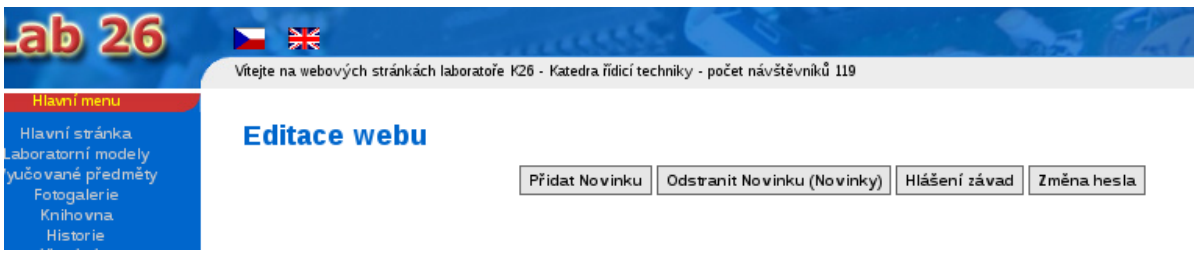
```
function forceLength(){
      //výška zobrazitelné plochy okna a odečtení výšky loga
  var windowHeight = $(window).height() - 135;
      //550px je minimální výška stránky
  if(windowHeight <  550) windowHeight =  550;
      //prodloužení jednotlivých elementů webu
  $("#text").height(windowHeight);
  $("#lista21").height(windowHeight + 21);
  $("#main").height(windowHeight + 125);
}

//posluchač změny velikosti okna prohlížeče
$(window).resize(forceLength);
```

Výpis 2.3: Java Script pro prodloužení stránky v závislosti na velikosti zobrazitelné plochy

Dle zadání byl také vylepšen vzhled administrátorského menu a bylo zajištěno, aby toto menu bylo viditelné i v případě výběru některé z možností nabídky. Pro efektnější zobrazení byl napsán CSS kód (Výpis 2.4) zajišťujícího zobrazení dle Obrázku 2.1.

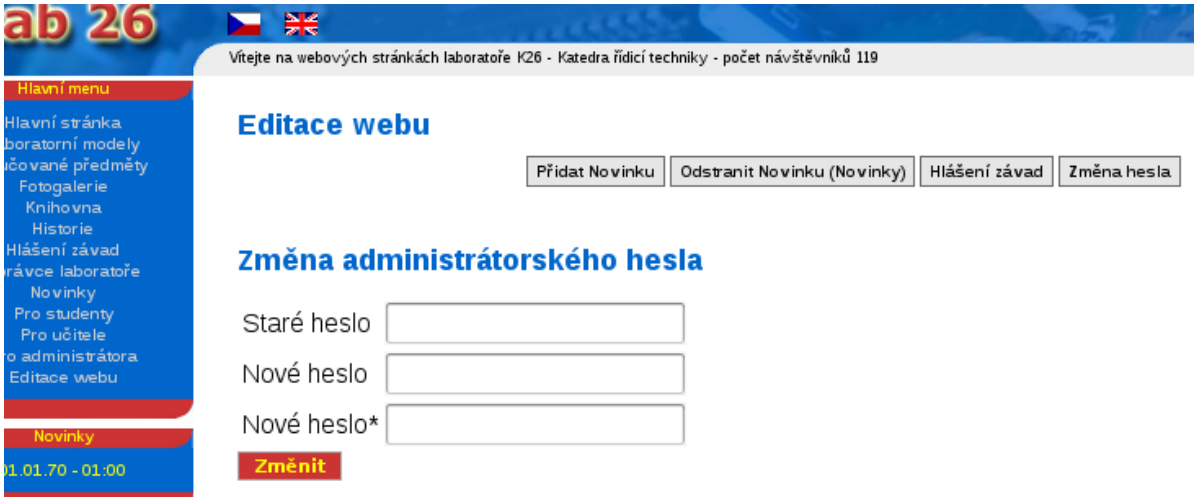
```
#admin_menu {
  display:block;
  list-style-type:none;
  text-align:center;
}
#admin_menu li {
  background-color:#E6E6E6;
  border:1px solid #676767;
  display:inline;
  margin:0;
  padding:2px 5px;
}
#admin_menu li:hover {
  border:1px solid #AAAAAA;
}
#admin_menu li a {
  color:#000000;
  text-decoration:none;
}
#admin_menu li a:hover {
  color:#E64000;
}
```

Výpis 2.4: CSS styl administrátorského menu



Obrázek 2.1: Nový vzhled administrátorského menu

Aby bylo menu stále zobrazeno, byly odkazy původního menu obohaceny o *get proměnnou* (*show*) udávající, která položka menu má být zobrazena. Proměnná *page* má stále hodnotu *edit,* což je původní stránka, obsahující menu samotné. Do tohoto souboru byla přidána podmínka zajišťující vložení patřičné stránky (název souboru v proměnné *show*), obsahující správnou jazykovou verzi požadované volby. Příklad vybrané položky (změna administrátorského hesla) včetně menu ukazuje Obrázek 2.2. Odkaz vedoucí na tuto stránku je naznačen ve Výpisu 2.5.



Obrázek 2.2: Vybraná položka administrátorského menu – změna hesla

```
http://support.dce.felk.cvut.cz/lab26/index.php?
                                    page=edit&show=admin_password
```

Výpis 2.5: URL adresa položky menu „Změna hesla“

2.3. Úpravy kódu

Modifikace uvedené v předchozí podkapitole spadají také do úprav kódu, ale jejich důsledkem jsou vizuální změny, proto byly vysvětleny samostatně. Jediná změna provedená v kódu stránek, která neovlivnila jejich vzhled, byl požadavek na vynucení stahování vektorových obrázků (se vzorci) ve formátu *wmf* (*Windows MetaFile*) [14]. MS Internet Explorer má tento typ souboru asociován pro přímé zobrazení. Po klepnutí na odkaz se tedy obrázek přímo zobrazil místo toho, aby se objevila nabídka pro stažení či otevření cílového souboru. Pro tento účel zde bylo využito scriptovacího jazyka PHP a jeho schopnosti modifikace hlavičky (headers).

V hlavičce souboru lze nastavit mnoho voleb a změnit tak chování zobrazeného dokumentu (stránky). Script z Výpisu 2.6 zajistí, že se dotaz na uložení objeví vždy. Funkce *header()* modifikuje hlavičku a přesvědčí tak prohlížeč, že cílový soubor není obrázek, který dokáže

zpracovat, ale soubor ke stažení. Podobným způsobem lze nastavit přesměrování, vykreslovat obrázky pomocí scriptů apod. Pozor se musí dát pouze na to, aby v kódu před příkazem *header()* nebyl žádný výpis ani žádné jiné znaky na výstupu, jinak je soubor označen za textový a hlavičku již nelze modifikovat.

Funkce scriptu z Výpisu 2.6 spočívá v načtení GET proměnné *file*, jejíž hodnota je definována URL dotazem a vyhodnocení, zda soubor existuje, zda nenastal pokus o stažení souboru s příponou *php* a jestli proměnná *file* obsahuje lomítka, tedy snahu o přechod mezi adresáři. Pokud je vše správně, script nastaví v hlavičce typ souboru - příloha a relativní odkaz na obrázek (*content-disposition*), velikost přílohy *content-length,* aby prohlížeč mohl zobrazovat procentní ukazatel stahování, a dodatečné proměnné *content-type* a *content-description*. Soubor se poté odešle na standartní výstup v binární podobě funkcí *readfile()*. V případě pokusu o stažení neexistujícího nebo PHP souboru se vypíše jednotná HTML stránka informující uživatele o tom, že požadovaný soubor nebyl nalezen.

```
<?php
$DIRECTORY="models/equations/";
$soubor = $_GET["file“];
if(substr($soubor,-3)!="php" && file_exists($DIRECTORY.$soubor)
            && !empty($soubor) && strpos($soubor,"/") === false){
    $velikost = filesize($DIRECTORY.$soubor);
    header("Content-Disposition: attachment;filename=$soubor");
    header("Content-Type: application/octet-stream");
    header("Content-Description: File Transfer");
    header("Content-Length:".$velikost);
    readfile ($DIRECTORY.$soubor);
}else{
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//CZ">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html;
                                                  charset=utf-8">
  <meta lang="cs" name="description" content="Stazeni souboru">
  <title>Stažení souboru <?=$soubor?> selhalo</title>
  </head>
  <body>
    <strong>Chyba: soubor <?=$soubor?> nenalezen</strong>
  </body>
  </html>
<?php
}
?>
```

Výpis 2.6: Soubor download.php pro vynucení stahování

2.4. Popis modelu Tepelná soustava TQ

V kategorii modelů u fyzikálního systému Tepelné soustavy chyběly informace, byly tedy doplněny ve stejném stylu jako u ostatních laboratorních modelů.

Úvodní text v českém jazyce:

„Laboratorní model tepelné soustavy TQ (Thermal Control) je fyzikální systém vyrobený firmou TecQuipment Ltd určený k výukovým účelům modelování dynamických systémů a jejich řízení. Model slouží k demonstraci, identifikaci a testování řídicích algoritmů pro soustavy s velmi dlouhými časovými konstantami. Typické aplikace jsou různé tepelné systémy, vytápěcí systémy, sklářské pece, kalcinační rotační pece a podobně.“

Úvodní text v anglickém jazyce:

„Laboratory model, Thermal Control TQ is physical system, made by TecQuipment Ltd intended for learning purposes, it's used for explain controling of dynamic systems. The model is used for demonstration, identification and testing of control algorithms for systems with a very long time konstants. Typical applications are the different thermal systems, heating systems, glass furnace, calcinating rotary kilns and etc.“

Obrázky z praxe, ukazující využití tohoto modelu:

Obrázek 2.3: Obrázky ukazující praktické využití modelu tepelné soustavy

Český text seznamující se skutečným modelem v laboratoři K26:

„Laboratorní model se skládá z potrubí s ventilátorem, hliníkovým chladičem a klapkou. Ventilátor žene vzduch potrubím přes chladič, na němž jsou umístěny dva teplotní senzory. Jeden je přímo na chladiči a druhý snímá teplotu vzduchu v okolí chladiče. Dále je na chladiči umístěn zdroj tepla. Potrubí je zakončeno klapkou, která umožňuje měnit rychlost proudění vzduchu modelem. Vstupy laboratorního modelu jsou napětí na ventilátoru u_v, napětí zdroje tepla u_t a napětí motorku klapky u_m. Výstupy laboratorního modelu jsou teploty na senzorech T_1, T_2. K ovládání modelu můžeme použít PC s programem Matlab/Simulink s Real Time Toolboxem, kde jsou všechny veličiny převedeny na bezrozměrná čísla obvykle v intervalu (-1, +1).

Využití pro výuku

Laboratorní model můžeme například využít k aplikaci základního řízení (například pomocí PID regulátorů) teploty okolí chladiče T_2 pomocí vstupu u_t při proměnném otevření klapky či rychlosti ventilátoru. Vzhledem ke své pomalosti je model vhodný pro aplikaci prediktivního řízení.“

Anglický text seznamující se skutečným modelem v laboratoři K26:

„The laboratory model consists of a pipe with a fan, aluminum cooler and flap. Fan drives air throught pipeline across the cooler where are placed two temperature sensors. One is directly on the cooler and the second senses the ambient temperature around the cooler. On the cooler is also located a heat source. The pipeline is completed flap, which allows you to change speed of airflow throught the model. Inputs of laboratory model of the voltage at the fan u_v, voltage source heat u_t and voltage motor throttle u_m. Outputs of the laboratory model are temperatures on sensors T_1, T_2. For control can be used PC with Matlab/Simulink software with Real Time Toolbox. All signals are converted to non-dimensional numbers (-1, +1).

Education use

Laboratory model can be used for applications such as the basic control (for example by PID controler) of ambient temperature around the cooler T_2 of the input through the u_t at varying speeds and throttle fan. Due to model speed is suitable for the application of predictive control.“

Schematický obrázek naznačující princip fungování modelu:

Obrázek 2.4: Schéma modelu tepelné soustavy

$u_v(t)$ … napětí na ventilátoru

$T_1(t)$ … teplota chladiče

$T_2(t)$ … teplota okolí chladiče

$u_t(t)$ … napětí na topném tělese

$u_m(t)$ … napětí určující polohu klapky

3. Motivační hra „Kulička na tyči“

V této kapitole je popsána mnou naprogramovaná motivační hra, její náplň, cíl a výběr vhodného programovacího jazyka. Dále je zde naznačen potřebný fyzikální princip a návrh regulátoru. V závěru kapitoly je vysvětlena vlastní programová realizace a implementace do webu Laboratoře teorie automatického řízení K26 [2].

3.1. Náplň hry a volba scriptovacího jazyka

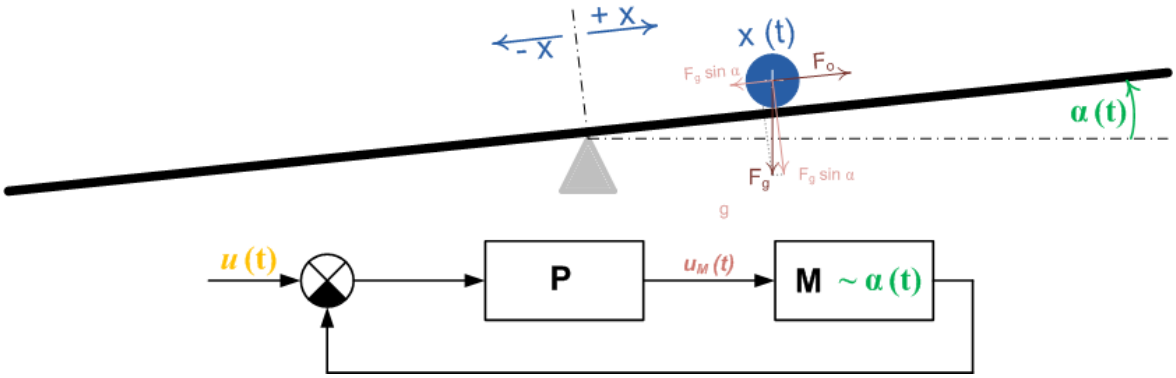
Motivační hra pro budoucí studenty by měla být zajímavá, názorná a zábavná. Všechny tyto vlastnosti sdružuje fyzikální model kuličky na tyči. Kuličku na tyči je možné regulovat ručně, a právě proto je ideálním modelem pro hru. Navíc je tento model součástí vybavení laboratoře, kde se vyučuje identifikace a následné řízení fyzikálních modelů.

Protože má být hra motivační, měla by být hratelná ve webovém prohlížeči, nejlépe přímo na stránkách Laboratoře teorie automatického řízení K26 [2]. Existuje celá řada způsobů jak naprogramovat hru hratelnou v prohlížeči, například pomocí Java Appletu nebo v dnešní době velice populárním komerčním produktem Flash od společnosti Macromedia®. Pro účely této práce byla vybrána méně známá technologie, která však plnohodnotně konkuruje ostatním možnostem. Byl zvolen programovací jazyk JavaScript a HTML 5 element *canvas*. Jedná se o poměrně novou technologii, i přesto je však její podpora webovými prohlížeči dostatečná. Nespornou výhodou JavaScriptu je možnost objektového přístupu a skutečnost, že kompilace probíhá až na klientově straně, a tedy výkonově ani paměťově nezatěžuje server, na kterém se zdrojový kód scriptu nachází.

Element *canvas* byl vyvinut společností Apple a později implementován do prohlížečů s jádrem Gecko, do kterých patří např. Mozilla Firefox. Tento prohlížeč je tedy zároveň i doporučený pro zajištění správné funkce hry. Velmi dobře je *canvas* podporován také běžně používanými prohlížeči Safari a Opera verze 9 nebo vyšší. Microsoft Internet Explorer tento objekt bohužel nepodporuje ani v nejnovější verzi (aktuálně 8). Nicméně pro Internet Explorer 6 a novější zajišťuje alespoň částečnou podporu ActiveX objekt Excanvas (více v [9], [10]), který simuluje objekt *canvas* a umožňuje využívat většinu jeho funkcí. Je však výpočetně náročný a na většině stolních počítačů tedy pomalejší. Hra tím ztrácí realistický dojem. Na stránce s hrou je tedy uvedeno, že pro správnou funkci je doporučený prohlížeč Mozilla Firefox.

3.2. Teoretický úvod ke hře

Kulička na tyči představuje systém s rychlou odezvou. Schématicky jej znázorňuje Obrázek 3.1. Tyč je upevněna ve svém středu a spojena s hřídelí motoru **M**. Motor **M** se otáčí, tím vychyluje tyč podle vstupního napětí u_M na motoru. Na tyči je pohyblivě umístěna kulička. Vstupní napětí u_M na motoru **M** pootočí tyč o úhel **α**. Motor **M** je umístěn ve zpětné vazbě s proporcionálním regulátorem **P**, takže hmotnost kuličky neovlivní úhel natočení α. Na kuličku působí síla F_g, která udává směr a rychlost pohybu kuličky, a zároveň ve směru proti pohybu kuličky brzdící síla odporu prostředí F_o.



Obrázek 3.1: Schematický nákres modelu kuličky na tyči

Aby kulička z tyče nespadla, jsou na skutečném modelu po celé délce tyče nataženy dva vodiče bez izolace, které vymezují prostor určený pro pohyb kuličky. Jeden z těchto vodičů je připojen na zdroj napětí. Kulička zde funguje jako jezdec potenciometru. Úbytek napětí udává polohu kuličky.

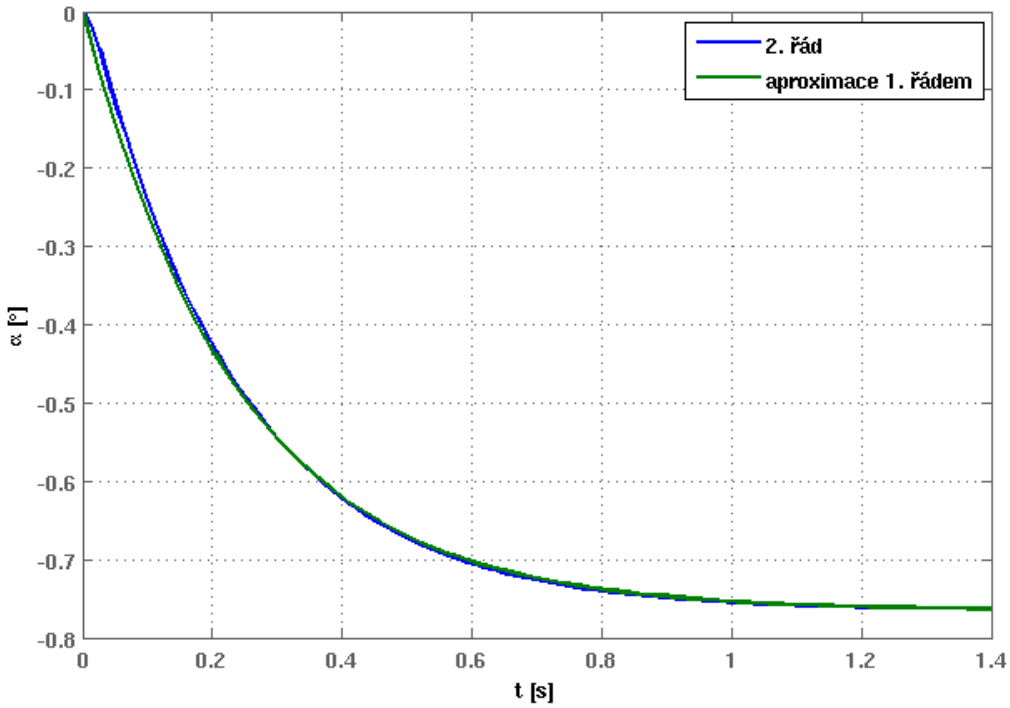
3.2.1. Matematický popis motoru

Přenos mezi žádaným náklonem tyče u [°] a skutečným náklonem tyče α [°] popisuje soustava druhého řádu [3]

$$P(s)=\frac{\alpha(s)}{u(s)}=\frac{-267{,}85}{s^2+85s+350} \tag{1}$$

pro hru je však využita aproximace prvním řádem

$$P_b(s)=\frac{\alpha(s)}{u(s)}=\frac{-0{,}7654}{0{,}24s+1} \quad . \tag{2}$$



Obrázek 3.2: Odezvy na jednotkový skok přenosů popisujících náklon tyče

Obrázek 3.2 ukazuje odezvy na jednotkový skok vstupní veličiny obou přenosů. Pro účely herní simulace je shoda dostatečná. Potřebná je také diferenční rovnice popisující závislost náklonu tyče na vstupním napětí. Ta vznikne přepisem rovnice (2) do tvaru

$$0{,}24\,\dot{\alpha}(t) = -\alpha(t) - 0{,}7654\,u(t) \quad . \tag{3}$$

Odtud

$$\dot{\alpha}(t) = \frac{-\alpha(t) - 0{,}7654\,u(t)}{0{,}24} \quad . \tag{4}$$

Ve hře je periodicky spouštěna výpočetní funkce. Pro numerické řešení diferenční rovnice bude tedy potřeba její diskretizovaná podoba. Využitím Eulerovy aproximace [3] se získá vztah

$$\frac{\alpha(k+1) - \alpha(k)}{T_s} = \frac{-\alpha(k) - 0{,}7654\,u(k)}{0{,}24} \quad , \tag{5}$$

kde T_s [s] je perioda vzorkování.

Úpravou vznikne diskrétní rovnice popisující natočení tyče v závislosti na žádané hodnotě náklonu tyče

$$\alpha(k+1)=\alpha(k)-\frac{T_s}{0{,}24}(\alpha(k)+0{,}7654\,u(k)) \quad . \tag{6}$$

3.2.2. Matematický popis pohybu kuličky

Pohyb kuličky lze popsat diferenciální rovnicí [3] druhého řádu

$$\ddot{x}(t)=-\frac{g}{1+\frac{2}{5}\left(\frac{r}{dr}\right)^2}\sin(\alpha(t))-b_k\dot{x}(t) \quad , \tag{7}$$

kde x [m] je poloha kuličky, α [°] úhel natočení tyče, b_k [s^{-1}] odpor prostředí, g [m/s^2] gravitační konstanta, r [m] poloměr kuličky a dr [m] polovina vzájemné vzdálenosti vodičů, mezi kterými je kulička umístěna. Pro zjednodušení se nahradí funkce sinus úhlem natočení tyče v obloukové míře

$$\ddot{x}(t)=-\frac{g}{1+\frac{2}{5}\left(\frac{r}{dr}\right)^2}\frac{\pi}{180}\alpha(t)-b_k\dot{x}(t) \quad . \tag{8}$$

Z rovnice (8) lze vyjádřit přenos úhlu natočení tyče na pozici kuličky

$$P_k(s)=\frac{x(s)}{\alpha(s)}=\frac{K_b}{s^2+b_k s}=\frac{K_b}{s(s+b_k)} \quad , \tag{9}$$

kde

$$K_b=-\frac{g}{1+\frac{2}{5}\left(\frac{r}{dr}\right)^2}\frac{\pi}{180} \quad . \tag{10}$$

Ekvivalentní zápis rovnice (8) je dvojice následujících rovnic

$$\dot{v}(t)=-\frac{g}{1+\frac{2}{5}\left(\frac{r}{dr}\right)^2}\frac{\pi}{180}\alpha(t)-b_k v(t) \quad , \tag{11}$$

$$\dot{x}(t)=v(t) \quad . \tag{12}$$

Opětovným použitím Eulerovy aproximace vznikne diskrétní popis ryhlosti pohybu kuličky a její pozice

$$\frac{v(k+1)-v(k)}{T_s}=-\frac{g}{1+\frac{2}{5}\left(\frac{r}{dr}\right)^2}\frac{\pi}{180}\alpha(k)-b_k v(k) \quad , \tag{13}$$

$$\frac{x(k+1)-x(k)}{T_s}=v(k) \quad , \tag{14}$$

kde T_s [s] je perioda vzorkování.

Úpravou se získají rovnice, které bude herní script společně s rovnicí (6) numericky řešit

$$v(k+1)=v(k)+T_s\left(-\frac{g}{1+\frac{2}{5}\left(\frac{r}{dr}\right)^2}\frac{\pi}{180}\alpha(k)-b_k v(k)\right) \quad , \tag{15}$$

$$x(k+1)=x(k)+T_s v(k) \quad . \tag{16}$$

Výsledný přenos celé soustavy je dán součinem přenosů (9) a (2)

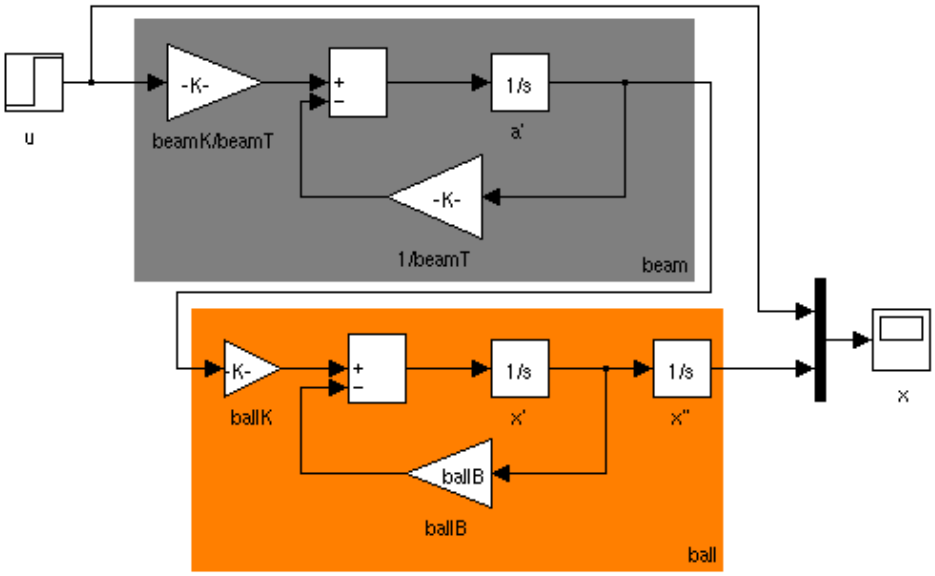
$$P_s(s)=P_k(s)\cdot P_b(s)=\frac{x(s)}{\alpha(s)}\cdot\frac{\alpha(s)}{u(s)}=\frac{K_b}{s^2+b_k s}\cdot\frac{-0{,}7654}{0{,}24s+1}=$$
$$=\frac{-0{,}7654\cdot K_b}{0{,}24s^3+(1+0{,}24b_k+b_k)s^2+b_k s} \quad . \tag{17}$$

Pro hru jsou využity konstanty ze skutečného modelu [3]. Odpor prostředí b_k je 0,25 s^{-1}, gravitační konstanta g je 9,81 m/s^2, poloměr kuličky r je 0,0095 m a vzdálenost mezi vodiči je 0,0134 m, dr je tedy 0,0067 m. Číselně vyjádřený přenos žádaného úhlu náklonu tyče na pozici kuličky je tedy

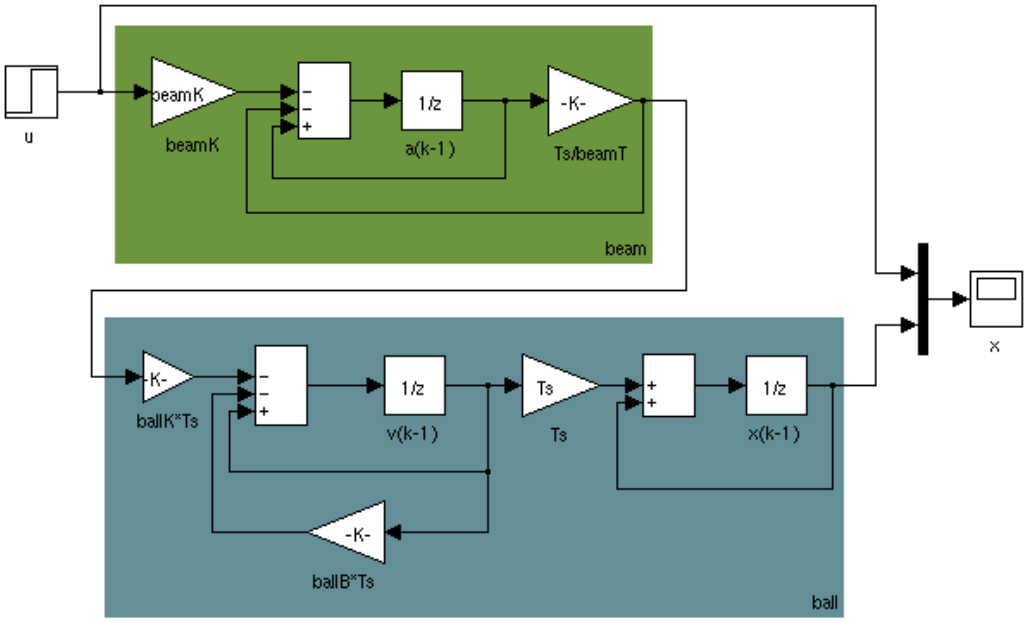
$$P_s(s)=\frac{x(s)}{u(s)}=\frac{-0{,}07249}{0{,}24s^3+1{,}06s^2+0{,}25s}=\frac{-0{,}07249}{s(0{,}24s^2+1{,}06s+0{,}25)} \quad . \tag{18}$$

Pro ověření správnosti výsledného přenosu lze využít nástroj Simulink z prostředí MATLAB, kde se dá soustava kuličky na tyči namodelovat a porovnat s vypočteným přenosem. Schéma spojitého modelu je znázorněno na Obrázku 3.3, kde šedě podbarvená část představuje rovnici tyče (4) a oranžová část rovnici kuličky (8). Diskrétní model sestavený z rovnic (6), (15) a (16) je na Obrázku 3.4. Rychlost vzorkování diskrétního modelu T_s byla zvolena *0,05* s, tomu odpovídá

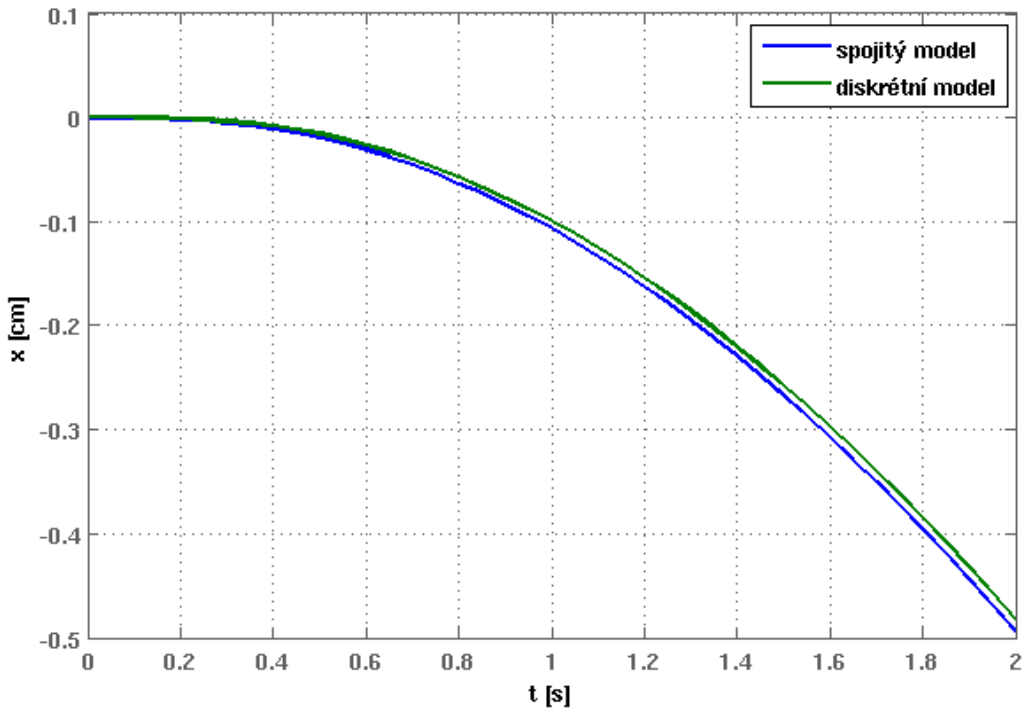
20 vzorků za sekundu. Porovnání výstupní odezvy na změnu žádaného úhlu natočení tyče z *0* na *5* stupňů spojitého a diskrétního modelu je zobrazeno na Obrázku 3.5.



Obrázek 3.3: Simulinkové schéma spojitého modelu kuličky na tyči



Obrázek 3.4: Simulinkové schéma diskrétního modelu kuličky na tyči



Obrázek 3.5: Porovnání odezvy spojitého a diskrétního modelu na skok vstupní veličiny z 0° na 5°

3.2.3. Regulátor pro pozici kuličky

V druhé části hry se uplatňuje regulátor, který má zdůraznit schopnosti automatického řízení. Pro efektivní řízení je postačující PD regulátor. Proporcionální složka slouží k zesílení nebo zeslabení vstupní veličiny a derivační složka zvyšuje rychlost regulátoru, což je vyváženo velikostí akčního zásahu. Integrační složka zajišťující nulovou regulační odchylku zde není potřeba, protože model je možné stabilizovat s nulovou regulační odchylkou pouze proporcionálním regulátorem.

Spojitý obecný PID regulátor má přenos

$$P_R(s) = P + \frac{I}{s} + D\,s \quad . \tag{19}$$

Ideální spojitý PD regulátor má nulovou Integrační složku, tedy

$$P_R(s) = P + D\,s = D(s + \omega_d) \quad . \tag{20}$$

Reálný spojitý PD regulátor označovaný také jako PDf je rozšířen o filtrační složku, která omezuje derivaci. Lze jej vyjádřit následujícím přenosem

$$P_{Rf}(s)=\frac{D(s+\omega_d)}{\frac{1}{\omega_f}s+1} \quad , \tag{21}$$

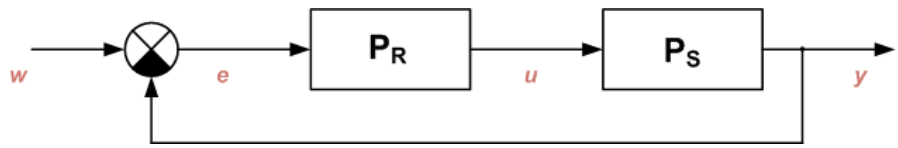
kde ω_f je filtrační frekvence (pól) [rad/s].

Obecný přenos uzavřené regulační smyčky, jejíž schéma je naznačeno na Obrázku 3.6, je následující

$$G(s)=\frac{Y(s)}{W(s)}=\frac{P_R P_S}{1+P_R P_S} \quad , \tag{22}$$

kde P_R je přenos regulátoru a P_S přenos soustavy.

Přenos otevřené regulační smyčky (odpojená zpětná vazba) lze vyjádřit jako součin přenosů soustavy a regulátoru.



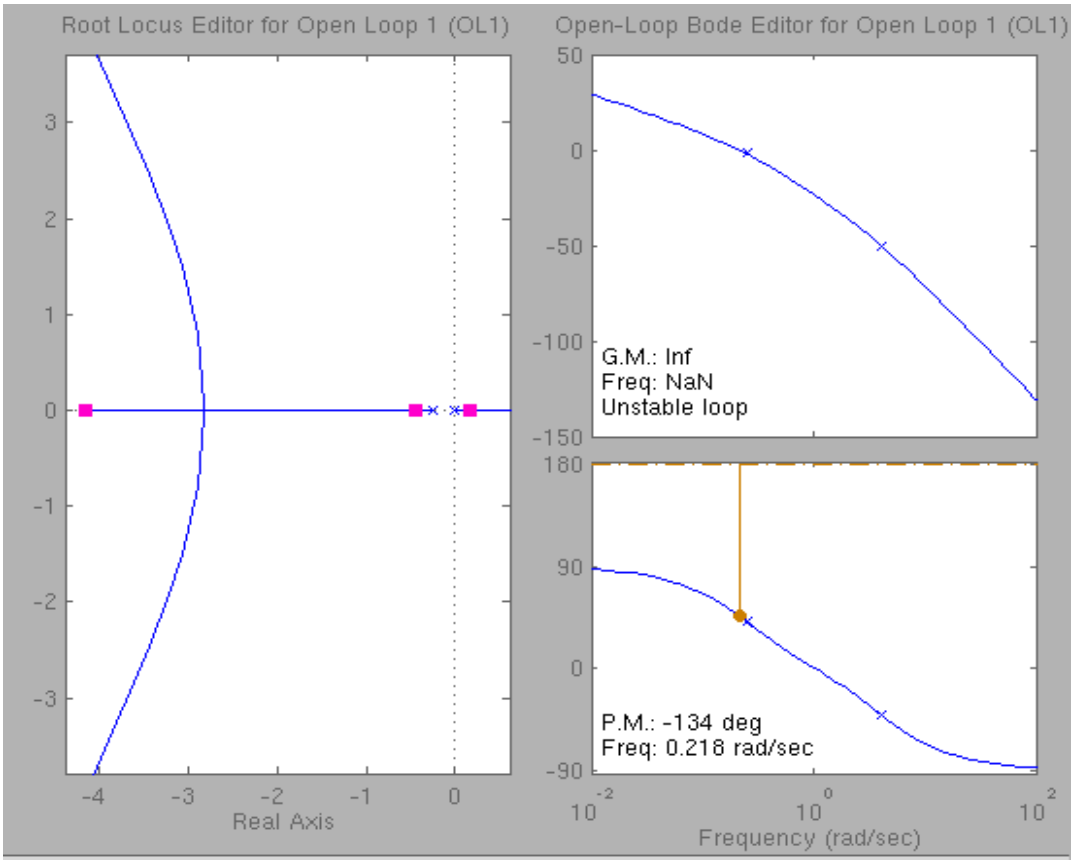
Obrázek 3.6: Obecné schéma uzavřeného regulačního obvodu

Pro návrh regulátoru je v této práci využit nástroj *sisotool*[1] z prostředí MATLAB, který zobrazuje geometrické místo kořenů, Bodeho charakteristiku, odezvu na skok, velikost akčního zásahu atd. *Sisotool* lze, díky možnosti přidávat nuly a póly na zadané souřadnice, využít pro návrh regulátoru v otevřené smyčce.

Obrázek 3.7 vlevo ukazuje geometrické místo kořenů přenosu (18), zde je v *sisotoolu* nastavena hodnota zesílení (regulátoru) na *1*. Geometrické místo kořenů, zobrazené na tomto obrázku, je tedy samotné soustavy.

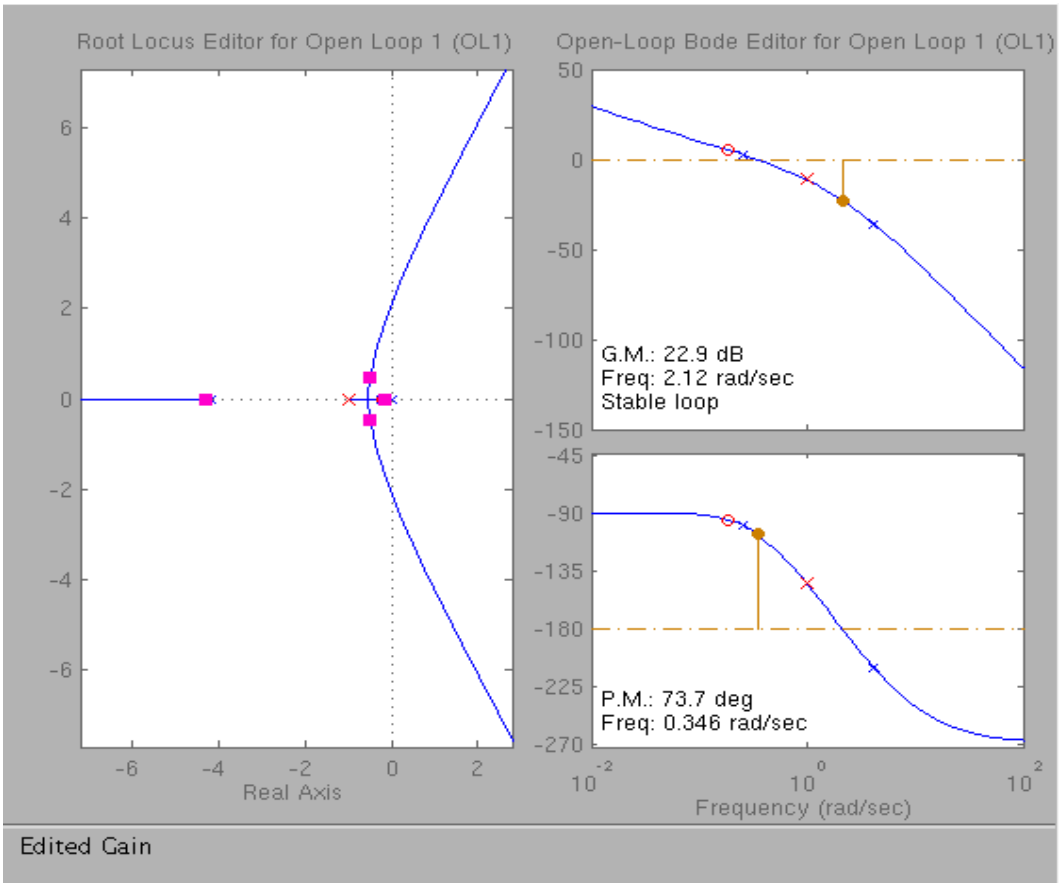
Stabilita celého systému je dána polohou nul a pólů. Stabilní systém je musí mít všechny nalevo od imaginární osy. Regulátor je tedy nutné navrhnout tak, aby přesunul nuly a póly do záporné oblasti reálné osy geometrického místa kořenů.

1 sisotool … siso značí single input, single output. Je to tedy nástroj pro práci se systémy s jedním vstupem a jedním výstupem.



Obrázek 3.7: Okno sisotoolu zobrazující geometrické místo kořenů a Bodeho charakteristiku přenosu modelu kuličky na tyči pro $k_p = 1$

Pro návrh PD regulátoru s filtrací (21) metodou geometrického místa kořenů se přidává jedna nula a jeden pól. Přenos (18), pro který má být regulátor navržen, má záporné znaménko. Regulátor tedy musí mít záporné zesílení, aby byla soustava stabilní. Pól byl zvolen v -1 a nula -0,186, tím došlo ke změnám dle Obrázku 3.8. Přesná hodnota (záporného) zesílení byla donastavena experimentálně tak, aby byla odezva celého systému, tedy odezva *y* i akční zásah *u*, na jednotkový skok žádané hodnoty *w* pro navržený regulátor co nejlepší. Odezvu na jednotkový skok i akční zásah lze zobrazit přímo v nástroji *sisotool.* Zobrazené průběhy se překreslí při každé změně parametrů soustavy. Díky tomu lze sledovat zhoršení či vylepšení návrhu a není nutné testovat regulátor v samostatné simulaci. Polohu přidaných pólů, nul a velikost zesílení lze zadávat velmi přesně. Záložku hlavního okna *sisotoolu* s možností přesného zadávaní parametrů ukazuje Obrázek 3.9. Geometrické místo kořenů finálního návrhu PDf regulátoru pro model soustavy kuličky na tyči je na Obrázku 3.8 vlevo.



Obrázek 3.8: Geometrické místo kořenů přenosu s navrženým PD regulátorem

Přenos navrženého PD regulátoru (v horní části Obrázku 3.9) je

$$P_R(s)=\frac{U(s)}{E(s)}=\frac{-5{,}5s-1{,}025}{s+1} \quad . \tag{23}$$

Tomu odpovídá

$$\dot{u}(t)=-5{,}5\dot{e}(t)-1{,}025e(t)-u(t) \quad , \tag{24}$$

kde u [°] je akční zásah a e [°] regulační odchylka.

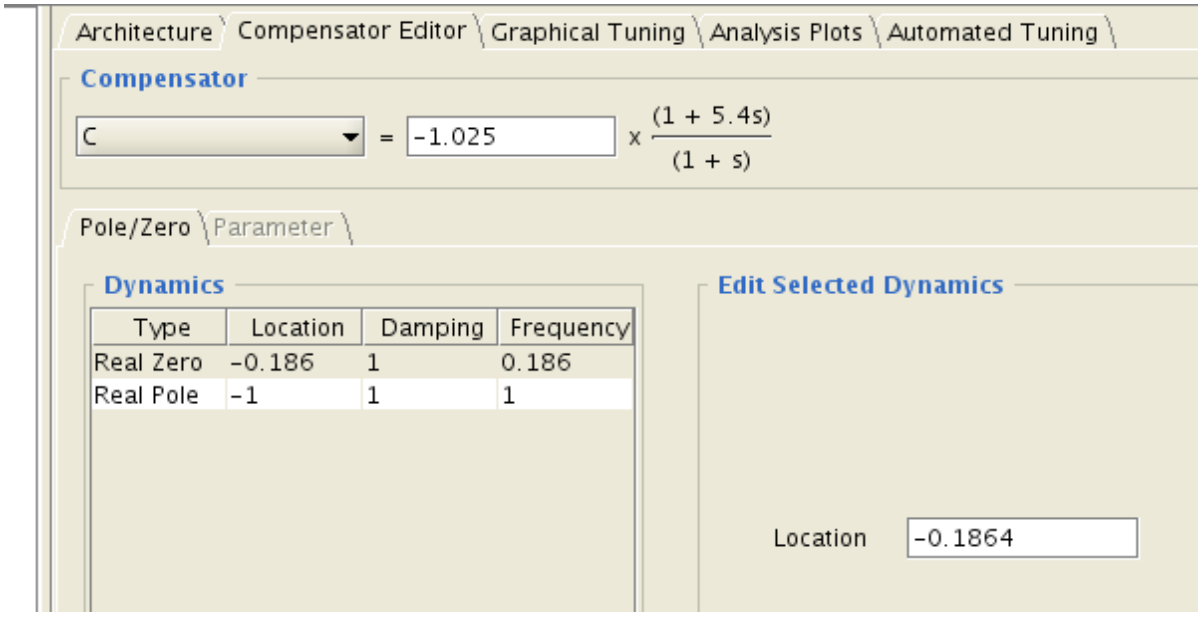
Opětovným užitím Eulerovy aproximace se získá diskrétní podoba regulátoru (24)

$$u(k+1)=-T_s u(k)+u(k)-5{,}5(e(k+1)-e(k))-1{,}025T_s e(k) \quad . \tag{25}$$

Číselně vyjádřená rovnice regulátoru, potřebná pro výpočetní algoritmus hry

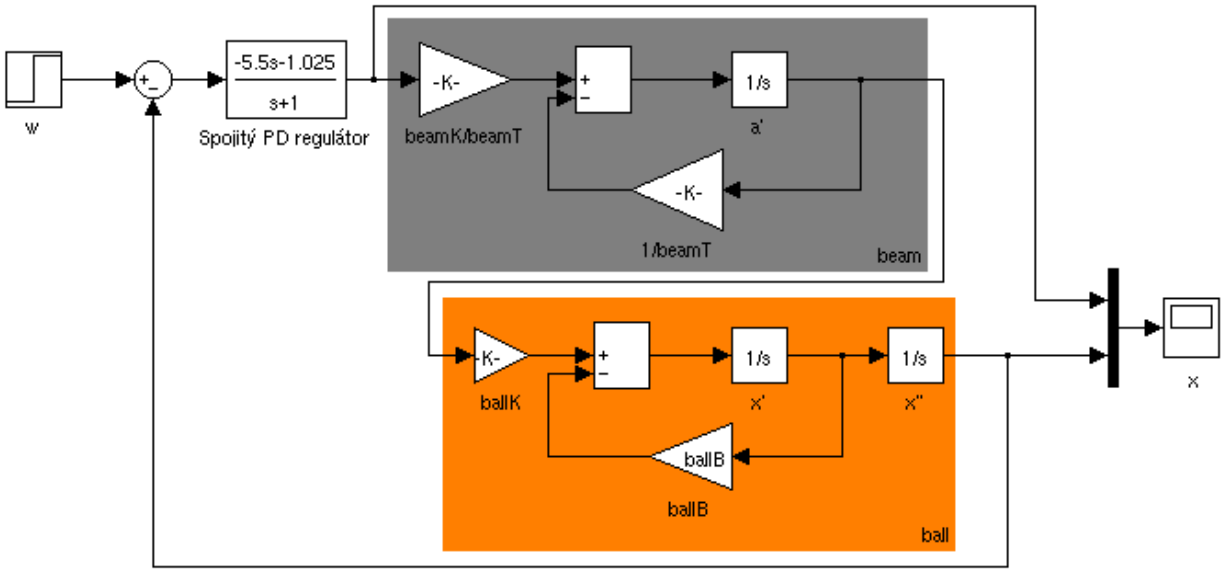
$$u(k+1)=0.95u(k)-5.5e(k+1)+5.4488e(k) \quad , \tag{26}$$

kde u [°] je akční zásah a e [°] regulační odchylka, pro Ts rovno $0{,}05$ s .

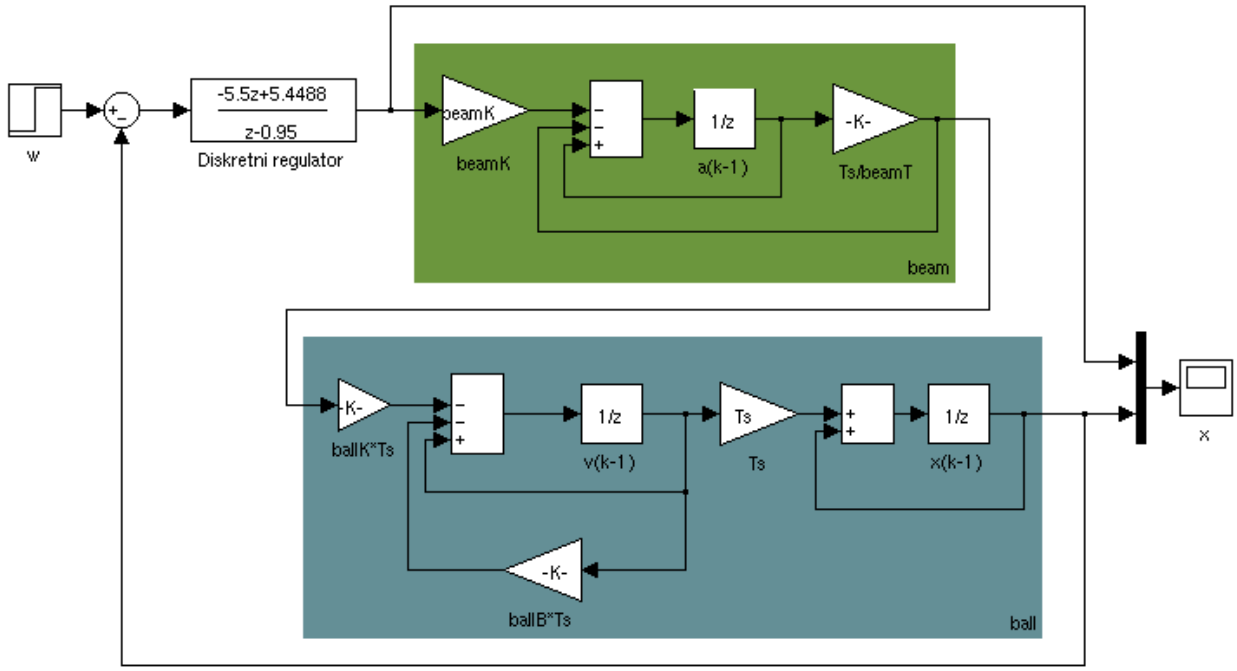


Obrázek 3.9: Okno zobrazující polohu přidaných pólů, nul a výsledný přenos navrženého regulátoru pro spojitý model kuličky na tyči

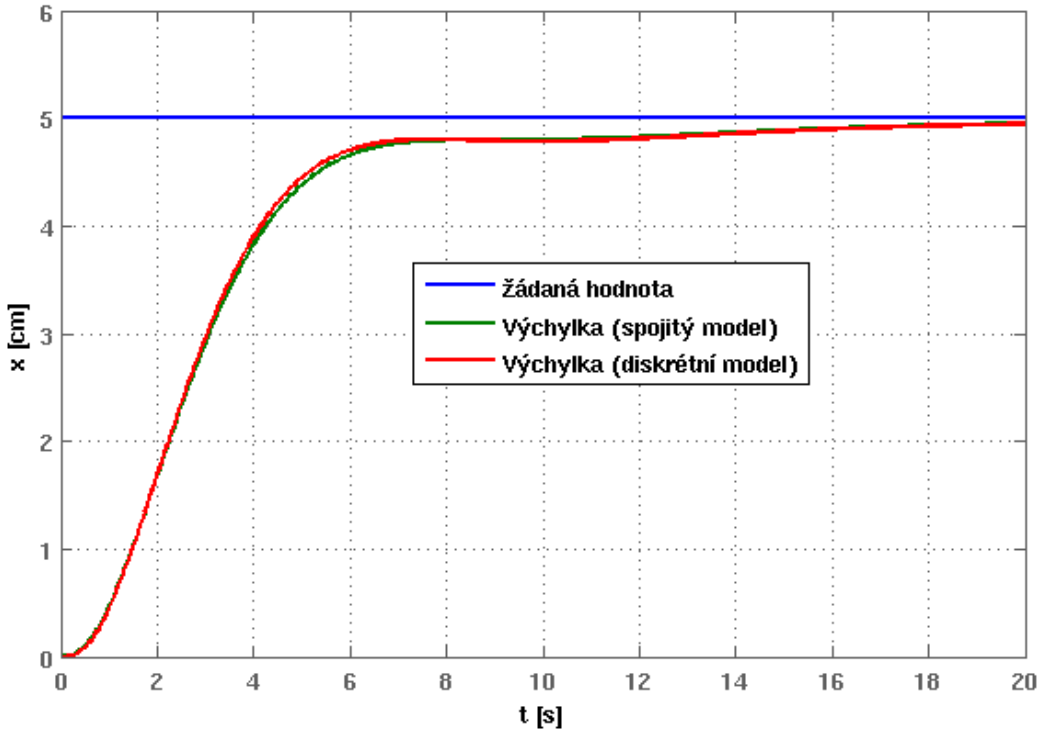
Navžený PD regulátor byl otestován na spojitém modelu (Obrázek 3.10) a diskrétní regulátor na diskrétním modelu (Obrázek 3.11). Porovnání odezvy spojitého a diskrétního modelu řízeného tímto regulátorem je zobrazeno na Obrázku 3.12. Akční zásahy těchto regulátorů jsou na Obrázku 3.13. Rychlost vzorkování je dostatečná a lze tedy použít diskretizovanou verzi spojitého regulátoru.



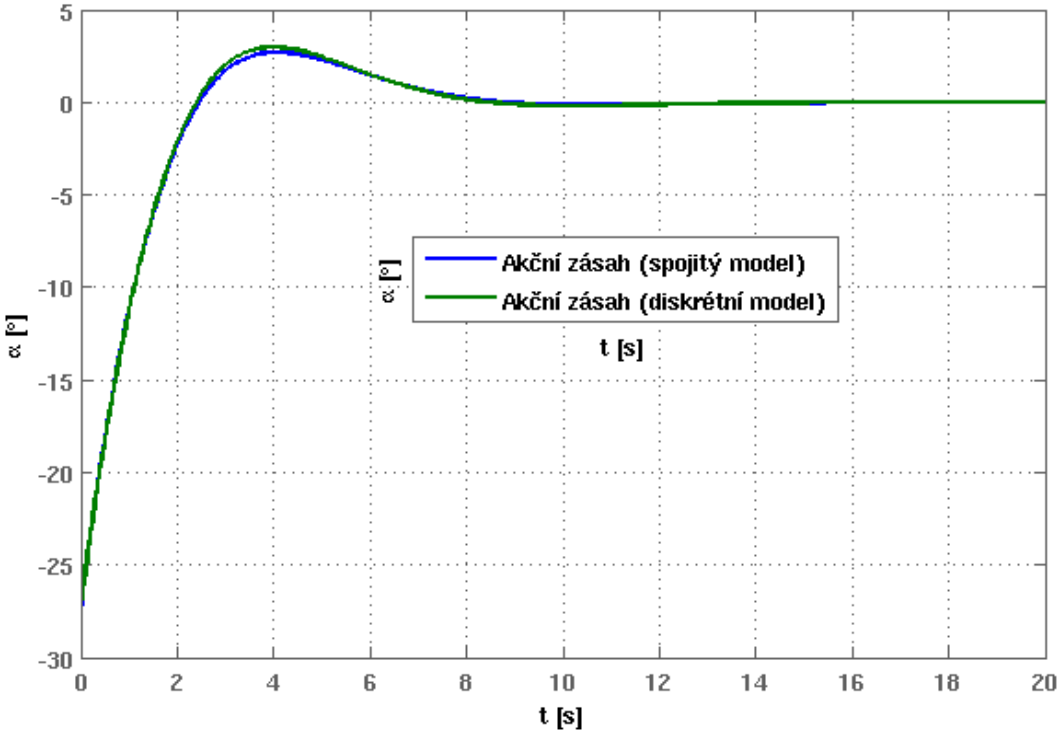
Obrázek 3.10: Schéma spojitého modelu kuličky na tyči s regulátorem



Obrázek 3.11: Schéma diskrétního modelu kuličky na tyči s regulátorem



Obrázek 3.12: Porovnání odezvy diskrétního a spojitého modelu kuličky na tyči s regulátorem



Obrázek 3.13: Porovnání akčních zásahů diskrétního a spojitého regulátoru pro model kuličky na tyči

3.3. Programová realizace

Pro lepší pochopení vlastního kódu hry bude nejprve ukázána základní práce s objektem *canvas* a knihovnami jQuery [6] a CanvasText [7]. Pak bude stručně vysvětlen zdrojový kód kuličky na tyči a použitý algoritmus na vykreslení grafických závislostí. Na závěr bude ukázána implementace hry do webu laboratoře K26.

3.3.1. Canvas, jQuery a CanvasText

Jak již bylo uvedeno, *canvas* je součástí nejnovější verze standardu HTML. Obsluha se provádí pomocí JavaScriptu. K tomu je nejprve potřeba získat referenci na patřičný objekt ve stránce, se kterým bude dále pracováno. V HTML existuje mnoho způsobů, jak přesně definovat určitý objekt. Při programování byl využit způsob identifikace podle argumentu *id*, který musí být pro každý objekt jedinečný (nebo prázdný), má-li být dokument validní[2]. Adresace pomocí *id* v JavaScriptu se liší podle používaného prohlížeče, proto zde byla využita knihovna jQuery [6], která usnadňuje přístup k HTML elementům a přidává nadstandartní funkce.

2 Validita dokumentu značí syntaktickou správnost zápisu zdrojového kódu. Dáno standardem W3C [12]

Adresace HTML objektu *canvas*, který má hodnotu *id* stanovenu na „*canvas*“ (Výpis 3.1) se v JavaScriptu pomocí jQuery provede příkazem ve Výpisu 3.2.

```
<canvas id=“canvas“ width=“20“ height=“50“></canvas>
```

Výpis 3.1: HTML objekt jehož id je rovno canvas

```
$('#canvas')[0]
```

Výpis 3.2: Adresování HTML objektu podle id pomocí jQuery

Pomocí standardních funkcí JavaScriptu je ekvivalentní zápis ve Výpisu 3.3. Předpokladem je podpora DOM (objektový model), ta však ve starších verzích prohlížečů (např. MS IE 6) chybí. Z tohoto důvodu byla využita knihovna jQuery.

```
document.getElementById('canvas')
```

Výpis 3.3: Adresování HTML objektu podle id

Tím je získán HTML objekt. Inicializační funkce *getContext* (Výpis 3.4) vrátí instanci objektu, který patřičný element *canvas* obsluhuje, a lze nad tímto objektem volat všechny podporované funkce.

```
ctx = $('#canvas')[0].getContext('2d');
```

Výpis 3.4: Získání instance objektu pro obsluhu příslušného canvasu

Nejprve bude popsáno vykreslování tvarů. To se zde řeší pomocí tzv. *cesty* (Path). Vykreslení a vyplnění cesty je ukázáno ve Výpisu 3.5.

```
//začátek cesty
ctx.beginPath();
//zde je cesta
//konec cesty
ctx.closePath();
//vykreslit hranice (okraje) cesty
ctx.stroke();
//nebo vyplnit cestu
ctx.fill();
```

Výpis 3.5: Vykreslení a vyplnění cesty

Vlastní cestu lze definovat pomocí příkazů *moveTo()* a *lineTo()*, argumenty těchto funkcí jsou souřadnice vůči mřížce v *canvasu,* která má počátek v levém horním rohu [8].

Mimo funkcí pro vyznačení a vykreslení cesty zde budou potřebné tyto metody (jejich použití je podrobněji vysvětleno v [8]):

rect(x,y,width,height) – vykreslení obdélníku s levým horním rohem v [x,y] o šířce *width* a výšce *height.*

clearRect(x,y,width,height) – vymazání patřičné oblasti.

drawImage(image,x,y) – vykreslení obrázku (objektu Image) na souřadnice.

arc(x, y, radius, startAngle, endAngle, anticlockwise) – vykreslení části nebo celé kružnice se středem v [x,y], poloměrem *radius*, počátečním a koncovým úhlem. Anticlockwise udává, zda funkce kruhovou výseč vykreslí po nebo proti směru hodinových ručiček.

translate(x,y) – posun plátna uvnitř *canvasu* na zadané souřadnice.

rotate(α) – rotace plátna o zadaný úhel v obloukové míře.

strokeStyle – není to funkce, ale vlastnost - proměnná. Umožňuje nastavení nebo změnu barvy pro vykreslování.

Hra kulička na tyči je vykreslena schématicky, tyč je tedy naznačena úsečkou. Tloušťka tyče je zde simulována vykreslením potřebného počtu úseček. Vykreslení tyče znázorňuje Výpis 3.6.

```
//volba barvy v hex formatu – např. #000000 … černá
ctx.strokeStyle = beamColor;
ctx.beginPath();
//přesun ukazatele na levý krajní bod tyče
ctx.moveTo(x-beamWidth , y – beamWidth);
//beamHeight udává tloušťku tyče
for(var i=0;i<beamHeight/2;i++){
  //vykreslení čáry na pravý okraj tyče
  ctx.lineTo(x+beamWidth,y+beamWidth+i);
  //vykreslení čáry na levý okraj tyče
  ctx.lineTo(x-beamWidth, y-beamWidth+i);
}
//uzavření cesty
ctx.closePath();
//vykreslení cesty
ctx.stroke();
```

Výpis 3.6: Vykreslení tyče

Obdobným způsobem probíhá vykreslení kuličky. Tedy vyplněním cesty tvořené uzavřenou kružnicí o poloměru *r*. Jak bylo uvedeno výše, kružnici lze zobrazit pomocí funkce *arc()* nad patřičnou instancí objektu *canvasu*. Zdrojový kód vykreslující kuličku ukazuje Výpis 3.7.

```
//začátek cesty
ctx.beginPath();
//vykreslení kruhu na pozici x,y o poloměru r
ctx.arc(x, y, r, 0, Math.PI*2, true);
//uzavření cesty
ctx.closePath();
//vyplnění kružnice
ctx.fill();
```

Výpis 3.7: Vykreslení kuličky na souřadnice [x,y]

CanvasText [7] je knihovna umožňující vykreslování znaků do objektu *canvas*. Tuto knihovnu je nutné nejprve aktivovat. Pak je možné využívat její funkce:

drawTextCenter(font, size, x, y, text) – vykreslí vycentrovaný text daným fontem (řetězec s názvem fontu, např. arial), *size* je velikost textu, *x*, *y* souřadnice pozice, na kterou má být text vykreslen a *text* je textový řetězec, který má být vykreslen.

fontAscent(font,size) – šířka jednoho znaku o velikosti *size* fontem *font*

fontDescent(font,size) – výška znaku o velikosti *size* fontem *font*

3.3.2. Kulička na tyči

Po spuštění hry je úkolem hráče udržet kuličku na místě vyznačeném šipkou. Ke změně referenční hodnoty dochází v pravidelných časových intervalech. V pravém dolním rohu hracího pole je herní čas, po jehož vypršení se objeví věta: „Nyní uvidíte automatickou regulaci, pro pokračování klepněte do hrací plochy“. Poté hraje PID regulátor se stejnou referencí. Jakmile regulátor dohraje, objeví se sdělení informující hráče o tom, že uvidí porovnání své a automatické regulace v grafu. Opět čas ubíhá v pravém dolním rohu. Během této části hry dochází k výpočtu kvality regulace pomocí součtu kvadratických odchylek [11]. Jakmile uplyne herní čas, grafy se zastaví a v horní části se zobrazí procentní ohodnocení regulace. Toto ohodnocení je stanoveno na základě PID regulace podle vztahu

$$q = 100 \cdot e^{-0.005\,(J_{hrac} - J_{PID})} \quad . \tag{27}$$

Konstanta v mocnině exponenciely byla stanovena porovnáním hodnot nejlepší a nejhorší regulace tak, aby procentní ohodnocení bylo úměrné výsledku.

Kvadratické odchylky jsou definovány následujícím vztahem

$$J = \sum (odchylka)^2 + \rho \cdot \sum (akční\ zásah)^2 \quad . \tag{28}$$

Konstanta ρ snižuje váhu akčního zásahu v celkové odchylce. Hodnota této konstanty byla stanovena na *0,1*.

Jednotlivé herní úseky jsou definovány v proměnné *gameStatus*. Na začátku hry je hodnota proměnné rovna *0* a na konci každé etapy je inkrementována. Hra končí, má-li *gameStatus* hodnotu *5*.

Zdrojový kód hry je složen z knihovny (*game.lib.js*) obsahující definice proměnných, všechny důležité funkce a obsluhu událostí. Hlavní výpočetní funkce *draw()* je v samostatném souboru (*game.js*). Pro vykreslení grafických závislostí byl napsán objekt *Graph* (*Graph.js*). Protože stránky laboratoře obsahují dvě jazykové mutace, byla i hra vytvořena tak, aby bylo možné měnit její jazyk. Všechny texty hry jsou v samostatném souboru *game.lang.js*.

Pro správnou funkci hry musí být jako první spuštěna funkce *preInit()*, která nastaví všechny potřebné proměnné, jako jsou rozměry hrací plochy nebo reference na objekt *canvas*. Vlastní hra se spustí funkcí *init()* (Výpis 3.8). Zde se zavolá JavaScriptová funkce *setInterval()*, která nastaví periodické spouštění zadané metody. Interval spouštění určuje druhý parametr, ten je zde roven převrácené hodnotě rychlosti vzorkování (diskrétních výpočetních funkcí z kap. 3.2.1).

```
function init() {
  //proměnná preinitvariable udává, zda funkce preInit()
  //proběhla v pořádku
  if(!preInitVariable) return false;
  //povolení knihovny CanvasText
  CanvasTextFunctions.enable(ctx);
  //aktivace intervalove funkce draw()
  intervalId = setInterval(draw, 1/Ts);
  return intervalId;
}
```

Výpis 3.8: Inicializační funkce

Výpis 3.9 ukazuje aktivaci posluchačů událostí. Událost (*Event*) je svým způsobem přerušení. Její kód se vykoná pokaždé, když událost nastane. Zde je tato možnost využita pro ovládání hry (změnu akčního zásahu). Zápis použitý ve Výpisu 3.9 je možný pouze při využití knihovny jQuery [6]. Také jsou zde předvedeny dvě možnosti zápisu přiřazení funkcí, reagujících na událost myši. Pro událost pohybu je přiřazena funkce *onMouseMove(),* zatímco pro kliknutí je funkce vepsána přímo do přiřazení. Oba tyto zápisy jsou možné. Nepsaná pravidla říkají, že první zápis je vhodnější v případě, že prováděná funkce je rozsáhlá nebo složitá. Druhá možnost je vhodná spíše pro krátké úseky kódu, jako nastavení hodnoty proměnné atd.

Kliknutí myši je v této hře využito dvakrát, jednou pro pokračování hry a jednou pro znovu načtení stránky na konci hry. Ve Výpisu 3.9 ve funkci *click()* je opět rozdělení, podle proměnné *gameStatus*. Pokud má tato proměnná hodnotu *1* (hráč dohrál, po kliknutí hraje regulátor) zavolá se funkce *setinterval* na metodu *draw()*. Je-li v proměnné *gameStatus* hodnota *5* (konec hry) a kurzor myši je na určitých souřadnicích, bude proveden příkaz pro znovu načtení aktuální URL adresy. Funkce *numIn()* používaná v kódu Výpisu 3.9 nepatří mezi standartní funkce. Byla vytvořena pro zkrácení zápisu podmínky a její návratová hodnota je závislá na tom, zda hodnota prvního parametru je větší než hodnota druhého a zároveň menší než hodnota třetího parametru. Pokud je podmínka splněna, funkce vrací *true*. V opačném případě *false*. Například výsledkem volání *numIn(2,1,5)* bude *true,* zatímco *numIn(5,1,2)* vrátí *false*. Protože číslo *5* je větší než *2*. Proměnná *evt* je vytvořena na základě vzniku události. Obsahuje mimo jiné informace o tom, které tlačítko myši bylo stisknuto a aktualní souřadnice pozice kurzoru (*pageX* a *pageY*). Tyto údaje jsou pro zpracování události nezbytné, a proto je tato proměnná předána obsluhujícím funkcím jako parametr.

```
//onMouseMove handler
$('#canvas').mousemove(onMouseMove);
//onClickEvent handler pro pokracovani hry po kliknuti
$('#canvas').click(function (evt) {
  if(gameStatus == 1){
    intervalId = setInterval(draw, 1/Ts);
  }
  if(gameStatus == 5){          //tlacitko znovu
    if(numIn(evt.pageX,LEFT + WIDTH/2 - fontHalphWidth,LEFT +
   WIDTH/2 + fontHalphWidth) && numIn(evt.pageY,TOP+40,TOP+80))
                                        window.location.reload();
  }
});
```

Výpis 3.9: Událostní funkce

Výpočet uvnitř funkce *onMouseMove* (Výpis 3.10) zpracovávající pohyb myši, je nutný pouze tehdy, když hraje hráč - hodnota proměnné *gameStatus* je *0*. V tomto případě funkce určí velikost akčního zásahu podle polohy kurzoru myši. Výpočetní algoritmus musí pouze počítat s tím, že uvnitř proměnné evt.pageX je aktuální vzdálenost kurzoru od levého okraje okna prohlížeče v pixelech, zatímco v proměnné *beamCenterX* je vzdálenost středu tyče od okraje hry. Rozdíl mezi levým okrajem hry (*canvasu*) a oknem prohlížeče vyrovnává proměnná *LEFT,* do které je při inicializaci uložena tato vzdálenost. Velikost akčního zásahu může nabývat hodnot od *-1* do *1*, proto je těsně před předáním (resp. před uložením do globální proměnné *controlAction*) vypočteno signum pomocné proměnné. Tím jsou ošetřeny hodnoty větší než *1* a menší než *-1*.

Činnost funkce z Výpisu 3.10 v případě, že má *gameStatus* hodnotu *5* (konec hry), je pouze slovně naznačena v komentáři, protože kód je příliš rozsáhlý (kompletní kód je na přiloženém DVD).

```
function onMouseMove(evt) {
  if(gameStatus == 0){
    //je-li hra v rezimu 0 - manualni rizeni
    //pasmo necitlivosti 1px na kazdou stranu
    if((evt.pageX – LEFT - 1) >= beamCenterX && (evt.pageX +
    LEFT + 1) <= beamCenterX){
      controlAction = 0;
    }else{
       //normalizovaná poloha myši (vůči středu tyče)
      tmp = -(beamCenterX + LEFT - evt.pageX) / ((beamCenterX +
                                                        LEFT) / 4);
       //je-li v abs hodnotě větší než 1 přepíše vypočtenou
       //hodnotu funkcí sign(x) tj pro x > 1 = 1 pro x < 1 = -1
      if(Math.abs(tmp) > 1) tmp = sign(tmp);
       //konečnou hodnotu zapsat jako akční zásah
      controlAction = tmp;
    }
  }
  if(gameStatus == 5){
    //režim 5 - překreslení textu „hrát znovu“ pokud je kurzor
      na daných souřadnicích
  }
}
```

Výpis 3.10: Část funkce onMouseMove()

Pro vykreslení závislostí, díky kterým lze pouhým okem porovnat kvalitu regulace, byl vytvořen objekt *Graph*. Do instance tohoto objektu lze zaznamenávat hodnoty v jednotlivých časových intervalech pro libovolný počet průběhů. Nad objektem lze zavolat metodu *paint(),* která graf vykreslí. Také je možné měnit barvy průběhů nebo měnit hodnoty funkcí v daných časových okamžicích. Rozměry, polohu a počet funkcí grafu je nutné zadat při vytváření instance objektu. Vytvoření jednoho z objektů (pro výchylku) ukazuje Výpis 3.11. V proměnné *graphDeflection* bude tedy objekt *Graph* pro dva průběhy o šířce *200 px,* výšce *100 px* a levým horním rohem v [*260*, *150*].

```
//Graph(x,y,width,height,f)
var graphDeflection = new Graph(260, 150, 200, 100, 2);
```

Výpis 3.11: Vytvoření instance objektu Graph pro záznam výchylky

Funkce draw() zajišťující výpočty a vykreslení je v Příloze A. Tato funkce je opět rozdělena podle hodnoty proměnné *gameStatus,* která říká, jak se má hra chovat. Pokud je její hodnota menší než dvě, je zobrazena kulička na tyči, graf výchylky a akčního zásahu. Také se provádí

výpočetní operace pro pohyb kuličky a natočení tyče. Pokud má *gameStatus* hodnotu *4,* zobrazí se porovnání výchylek a akčních zásahů naměřených během hry regulátoru a hráče. Ostatní hodnoty proměnné *gameStatus* jsou pomocné pro přechod mezi jednotlivými etapami hry.

Vykreslení tyče již bylo vysvětleno v kapitole 3.3.1, aby se tyč natáčela, byla využita rotace bodu kolem počátku v $\mathbb{R}^2$ [5]. K tomu je nutné vypočítat rotační matici

$$R=\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} , \tag{29}$$

kde α je úhel natočení bodu. Vlastní rotace bodu se provede vynásobením sloupcového vektoru souřadnic tohoto bodu s rotační maticí

$$\begin{bmatrix} x' \\ y' \end{bmatrix}=\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} . \tag{30}$$

Střed rotace je zde střed tyče a její konec je rotovaným bodem. I když jednou ze základních funkcí *canvasu* je rotace plátna, zde ji nebylo možné využít, protože pro výpočetní algoritmus polohy kuličky je nutné znát souřadnice umístění tyče v každém okamžiku. Vestavěná funkce rotace však otáčí celým plátnem a souřadnice bodů cest a všech objektů jsou tedy stále stejné. Rotace plátna byla využita pro popis vertikálních os u grafů. Není to však operace nezbytná pro tuto práci (podrobně vysvětleno v [6]).

Aby kulička byla stále na tyči, v každém cyklu (spuštění funkce *draw()*) dochází k výpočtu y-ové souřadnice kuličky z rovnice přímky (tyče). X-ová souřadnice se počítá dle rovnic (15) a (16). V každém kroku je uložena aktuální souřadnice do proměnné pro předchozí hodnotu. Stejným způsobem se počítá úhel natočení tyče z rovnice (6).

V JavaScriptu nelze vytvářet výpočetní vlákna v pravém slova smyslu, a proto se pohybem myši (vznik události) přerušuje činnost scriptu, a tím se prodlužuje doba výpočtu. Aproximace přenosu motoru prvním řádem v kapitole 3.2.1 byla provedena právě z tohoto důvodu.

V režimu PID protivníka (*gameStatus* rovný *1*) se počítá rovnice regulátoru (26), která mění hodnotu akčního zásahu tj. proměnné *controlAction*. Zbytek probíhá stejně jako v režimu hráče.

Pro jazykové mutace bylo vytvořeno proměnná *LANG* obsahující pole, v němž jsou uloženy všechny texty, popisky a oznámení zobrazované ve hře. Díky tomu je možné například přidání dalšího jazyka nebo jednoduchá modifikace vnitřních textů bez zásahu do kódu hry.

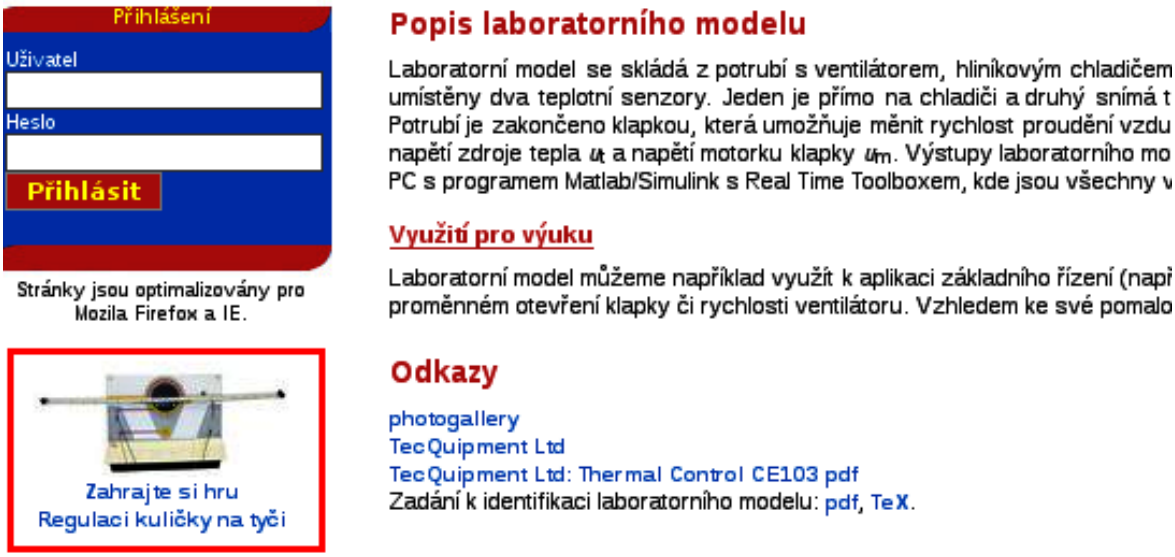
3.4. Implementace hry do webu laboratoře K26

Pro vložení hry do webu stačilo pouze načíst všechny potřebné soubory (Výpis 3.12) a vložit element *canvas*. Pomocí jQuery metody $(document).ready() zajišťující, že kód uvnitř bude proveden, jakmile budou všechny části dokumentu načteny. Tím lze předejít chybám vznikajícím v důsledku volání ještě nenačtených funkcí, objektů a podob.

Na webu Laboratoře teorie automatického řízení K26 [2] byla vytvořena stránka s hrou, jednoduchým popisem a návodem, jak hru hrát, v českém i anglickém jazyce. Do levé části stránky (pod hlavní menu) byl umístěn odkaz s obrázkem, který má za úkol upoutat návštěvníkovu pozornost (Obrázek 3.14 – červeně zvýrazněno).

```
<script type="text/javascript" src="lib/jquery-
                                          1.3.2.min.js"></script>
<script type="text/javascript" src="lib/canvastext.js"></script>
<!--[if IE]><script type="text/javascript"
                src="excanvas.compiled.js"></script><![endif]-->
<script type="text/javascript" src="lib/game.lib.js"></script>
<script>
var gameLanguage = 'cz';
</script>
<script type="text/javascript" src="lib/game.lang.js"></script>
<script type="text/javascript" src="lib/Graph.js"></script>
<script type="text/javascript" src="game.js"></script>
<script>
$(document).ready(function() {
  preInit();
 });
</script>
```

Výpis 3.12: Načtení potřebných funkcí



Obrázek 3.14: Odkaz na hru „Kulička na tyči“ (vyznačeno červeným obdélníkem)

4. Sjednocení přihlašovacích údajů

V této kapitole je popsán návrh autentifikace pomocí přihlašovacích údajů v rámci místní sítě na Karlově náměstí. Pro ověření či získávání údajů o členech akademické obce ČVUT-FEL se ve školní síti na Karlově náměstí používá protokol LDAP (*Lightweight Directory Access Protocol*) [13]. Všechny potřebné údaje lze tedy získat na základě komunikace s LDAP serverem.

4.1. LDAP

LDAP je protokol pro přístup k datům na adresářovém serveru. V případě ČVUT lze touto cestou procházet seznamy lidí, jejich přihlašovací jména, domovské adresáře atd. Data jsou na serveru uložena ve stromové struktuře. Jedná se o klient-server aplikaci. Implicitně probíhá komunikace na portu 389, po kterém klient požádá o komunikaci se serverem a ten na základě požadavku přijme, nebo odmítne spojení. Pokud je spojení navázáno, klient může provádět buď tzv. anonymní, nebo autorizované spojení (*bind*), případně vyhledávat a modifikovat adresářovou strukturu s daty.

4.2. Komunikace s LDAP serverem a autentifikace

Výpis 4.1 ukazuje funkci *ldap_search_user(„uživatelské jméno“)* pro ověření existence uživatele, resp. uživatelského jména v databázi serveru Katedry řídicí techniky. Funkce nejprve ověří, zda je uživatelské jméno mezi studenty a v případě, že ne, hledá jej mezi zaměstnanci katedry. Pokud je uživatel nalezen, LDAP vrátí jeho rozlišovací jméno (*DN - Distinguished Name*), kterým je jednoznačne definován každý záznam. Funkce *ldap_search_user* vrací pole, které obsahuje návratovou hodnotu pod klíčem *'result'*, studenti jsou určeni hodnotou *1* a zaměstnanci hodnotou *2* v poli *'type'*, rozlišovací jméno v *'dn'* a chybu (došlo-li k nějaké) v *'error'*.

Pro ověření hesla slouží funkce *ldap_get_auth(„*rozlišovací jméno“, „heslo“*)* (Výpis 4.2). Tato funkce opět vrací asociativní pole. V klíči *'result'* je výsledek autorizace, pokud je heslo i jméno správné, má hodnotu *true,* jinak *false*. Opět je zde klíč *'error',* který obsahuje patřičné hlášení v případě chyby.

Pro funkci *ldap_get_auth()* je nutná zabezpečená komunikace se serverem. Tu je možné realizovat pomocí SSL (Secure Sockets Layer) nebo TLS (Transport Layer Security). SSL i TLS jsou kryptografické protokoly zajišťující bezpečnost komunikace na internetu pro libovolné datové přenosy.

```
function ldap_search_user($login){
  $found = false;
  $dn = "";
  $error = "ok";
  $user_type = 0;
  //přihlášení k LDAP serveru
  $ds=ldap_connect("control.felk.cvut.cz");
  if ($ds) {
      //nastavení parametrů spojení
       ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);
       ldap_set_option($ds, LDAP_OPT_REFERRALS, 0);
      //anonymní přihlášení
       $r = @ldap_bind($ds);
       if($r){
          //student bind (nalezení jména v databázi studentů)
         $sr=ldap_search($ds, "cn=$login,ou=student, ou=felk,
                                          o=cvut, c=cz", "uid=*");
         $info = ldap_get_entries($ds, $sr);
         if($info["count"] > 0){
             $found = true;
             $user_type = 1;
              $dn = "cn=$login,ou=student,ou=felk,o=cvut,c=cz";
         }
         //teacher bind (nalezení v databázi zaměstnanců)
         $sr=ldap_search($ds, "cn=$login,ou=control, ou=felk,
                                          o=cvut, c=cz", "uid=*");
         $info = ldap_get_entries($ds, $sr);
         if($info["count"] > 0){
             //počet vrácených záznamů > 1 => uživatel existuje
             $found = true;
             $user_type = 2;
              $dn = "cn=$login,ou=control,ou=felk,o=cvut,c=cz";
         }
       }else{
             $error = @ldap_error($ds);
             $error = "error: ".$error."<br/>";
       }
       ldap_close($ds);
  } else {
       $error = "error: Unable to connect to LDAP server<br/>";
  }
  return Array( 0=>$found, 'result'=>$found, 'type'=>$user_type,
                1=>$user_type, 'dn'=>$dn, 2=>$dn, 'error'=>$error);
}
```

Výpis 4.1: Funkce pro ověření existence uživatele v síti LDAP

Tím je zajištěno, že citlivá data odesílaná na server mohou být jen těžko odposlechnuta. Tento krok se ovšem stal kamenem úrazu celé funkce. V rámci správy a údržby sítě na Karlově náměstí

byl server *support.felk.cvut.cz* přesunut na jiný segment páteřní sítě. Není tedy na stejném segmentu jako LDAP server. LDAP však nepříjmá zabezpečenou komunikaci z vnější sítě, není tedy možné z *support.felk.cvut.cz*, kde je webová stránka umístěna, uskutečnit zabezpečené sezení, a tedy ani autorizované spojení pro ověření uživatelského hesla. Nicméně dojde-li ke změně topologie sítě nebo nastavení LDAP serveru, bude dvojice funkcí *ldap_search_user() a ldap_get_auth()* schopná autentizovat uživatele v rámci sítě na Karlově náměstí a rozlišit studenty od zaměstnanců katedry.

```
function ldap_get_auth($dn,$password){
    $ret = false;
    $error = "";
      //navazání komunikace s ldap serverem
    $ds = ldap_connect("control.felk.cvut.cz");
      //nastavení parametrů připojení
    ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);
    ldap_set_option($ds, LDAP_OPT_REFERRALS, 0);
    if($ds){
      //spusteni zabezpecene komunikace - kamen urazu
      $r = @ldap_start_tls($ds);
      if($r){
            //autorizovane prihlaseni
            $r = @ldap_bind($ds,$dn,$password);
            if($r){
                  //pokud projde, jsou zadané údaje správné
                  $ret = true;
            }else{
                  $error = @ldap_error($ds);
            }
      }else{
            $error = @ldap_error($ds);
      }
      ldap_close($ds);
    }
    return Array(0=>$ret, 'result'=>$ret, 1=>$error,
                                                 'error'=>$error);
}
```

Výpis 4.2: Funkce pro ověření uživatelského hesla pomocí sítě LDAP

Jednoduchý (teoretický) příklad autentifikace pomocí uvedených funkcí ukazuje Výpis 4.3. Nejprve se pokusí o nalezení uživatele *hajekj7,* pokud je výsledek hledání *true,* vypíše numerickou hodnotu udávající, zda se jedná o studenta či zaměstnance katedry. Následuje pokus o ověření hesla. Je-li heslo správné, vypíše „*Identita ověřena*“. V případě chyby je zobrazeno patřičné chybové hlášení, získané z LDAP serveru.

Pro zajištění funkčnosti uvedeného příkladu by bylo nutné změnit nastavení LDAP serveru tak, aby příjmal TLS spojení i z druhého síťového adaptéru. Nebo jeden ze serverů (*support.dce.felk.cvut.cz* nebo *LDAP)* umístit na stejný segment páteřní sítě.

```
$ret = ldap_search_user('hajekj7');
if($ret['result']){
   //nedoslo k chybě
   echo $ret['type']; //student = 1, zaměstnanec = 2
   //ověření hesla
   $ret = ldap_get_auth($ret['dn'],'uživatelovo tajné heslo');
   if($ret['result']){
      //heslo ověřeno
      echo 'Identita ověřena';
   }else{
      //došlo k chybě, vypíše chybu
      echo $ret['error'];
   }
}else{
   //došlo k chybě, vypíše chybu
   echo $ret['error'];
}
```

Výpis 4.3: Příklad autentifikace uživatelů

5. Závěr

Cílem této práce bylo rozšířit a obohatit stávající internetovou prezentaci Laboratoře teorie automatického řízení K26 [2], napsat popis libovolného modelu ve stejném stylu a upravit vzhled stávajícího administrátorského rozhraní, naprogramovat motivační hru a autentifikaci s pomocí serveru usermap.

Administrátorské rozhraní bylo upraveno dle požadavků. Byl změněn vzhled administrátorského menu a bylo zajištěno jeho stálé zobrazení při výběru jednotlivých položek menu.

Fyzikální model, jehož popis byl na web doplněn, byl stanoven na Tepelnou soustavu. Byl přidán úvodní text s praktickým využitím systému, podrobnější text užitečný pro obsluhu modelu a schématický obrázek pro lepší představu o modelu.

Motivační hra byla naprogramována v jazyce JavaScript, který umožňuje přímé spuštění v prohlížeči bez nutnosti rozšíření nebo speciálních aplikací. Náplní hry je snaha o řízení soustavy „kulička na tyči“, která se nachází v Laboratoři teorie automatického řízení K26 [2], a následné porovnání s PID regulátorem navrženým pro tento model. Na konci hry je zobrazen procentní poměr úspěšnosti regulace. Tím je zajištěna hratelnost – hráč se může snažit vylepšit svou kvalitu regulace. Vzhledem ke kvalitě regulátoru není vyloučena stoprocentní úspěšnost regulace.

Autentifikace pomocí serveru usermap byla během činnosti na této části bakalářské práce změněna na ověření pomocí serveru LDAP uvnitř sítě na Karlově náměstí. Kvůli změnám ve struktuře počítačové sítě však nebylo možné uskutečnit autentifikační proces (ověření hesla). V této práci je tedy pouze naznačeno řešení, které bohužel z technických důvodů nelze využít. Pokud ovšem v budoucnu dojde ke změně struktury sítě nebo nastavení LDAP serveru, bude možné zde uvedené řešení využít.

Literatura

[1] HOLEČEK, J. *Webové stránky laboratoře K26*. Praha, 2008. 55 s. ČVUT FEL. Vedoucí bakalářské práce Roubal J.

[2] Roubal, J.,Holeček, J.,Hájek, J., *Laboratoř teorie automatického řízení K26*, [online]. [cit. 2009-5-12], http://support.dce.felk.cvut.cz/lab26

[3] Roubal, J., *Kulička na tyči TQ (BB1)*, 2008. [online]. [cit. 2009-5-12], http://support.dce.felk.cvut.cz/lab26/

[4] Fuka, J. – John, J. – Kutil, M. *Učebnice SARI*. [online]. [cit. 2009-5-12], *http://dce.felk.cvut.cz/sari*

[5] Horcik, R., *Točíme krychlí*. [online]. [cit. 2009-5-12], http://www2.cs.cas.cz/~horcik/Teaching/applications/node1.html

[6] jQuery, [online]. [cit. 2009-5-12], http://jquery.com

[7] Studt, J., *Canvastext,* [online]. [cit. 2009-5-12], http://www.federated.com/~jim/canvastext

[8] Kliehm, M., *Drawing Graphics with Canvas,* [online]. [cit. 2009-5-12], *https://developer.mozilla.org/en/drawing_graphics_with_canvas*

[9] Almaer, D., Canvas in IE, [online]. [cit. 2009-5-12], *http://ajaxian.com/archives/canvas-in-ie*

[10] Excanvas, [online]. [cit. 2009-5-12], http://excanvas.sourceforge.net

[11] Roubal J., Pekař J., Pachner D., Havlena V., *Moderní teorie řízení – Cvičení.* Praha, 2005. ČVUT.

[12] W3C, *World Wide Web Consortium,* [online]. [cit. 2009-5-12], http://validator.w3.org

[13] WIKIPEDIA, *LDAP*, [online]. [cit. 2009-5-12], *http://cs.wikipedia.org/wiki/LDAP*

[14] Hájek, J., *paste.cz*, [online]. [cit. 2009-5-12], *http://www.paste.cz*

[15] WIKIPEDIA, *Windows Metafile,* [online]. [cit. 2009-5-12], *http://en.wikipedia.org/wiki/Windows_Metafile*

[16] Dorf, R. C. and Bishop, R. H., *Modern Control Systems*, 11. vydání, Prentice Hall, 2007, ISBN-10: 0132270285, ISBN-13: 978-0132270281

[17] Kosek, J., *PHP – tvorba interaktivních internetových aplikací*, Grada publishing, 1999

[18] Horáček, P., *Systémy a modely*, Praha 2000

[19] Welling, L., Thomson, T., *PHP a MySQL – rozvoj webových aplikací*, 2003

Použitý software

- Operační systém Linux
- OpenOffice 2.6, *http://openoffice.cz*
- Gimp 2.6, *http://www.gimp.org*

Knihovny

- Excanvas, http://excanvas.sourceforge.net/
- jQuery, http://jquery.com/

Příloha A

Zdrojový kód výpočetní funkce

```
function draw() {
  ctx.fillStyle = backColor;
  clear();
  if(gameStatus == 4){
    //hra v rezimu 4 - porovnani automaticke a rucni regulace
    ctx.strokeStyle = "#000000";
    var font = "arial";
    var outY = HEIGHT - ctx.fontAscent(font,8);
    var time = LANG['time'] + Math.round(cykl*Ts) + LANG['time_fin'] +
                                    (gameTime - Math.round(cykl*Ts)) + " s";
    ctx.drawTextCenter( font, 8, WIDTH - 120, outY,time);
    ctx.drawTextCenter( font, 10, 55, outY,LANG['legend']);
    //popis grafu
      //popis osy x u akcniho zasahu
    ctx.drawTextCenter( font, 8, 230, 265,"t [s]");
      //popis osy x u vychylky
    ctx.drawTextCenter( font, 8, 470, 265,"t [s]");
    ctx.save();
    ctx.translate(WIDTH/2,HEIGHT/2);
    ctx.rotate(3/2 * Math.PI);
      //popis osy y u akcniho zasahu
    ctx.drawTextCenter( font, 8, 50, -240,"u [V]");
      //popis osy y u vychylky
    ctx.drawTextCenter( font, 8, 50, 0,"x [m]");
    ctx.restore();
    ctx.drawTextCenter( font, 12, 370 - ctx.fontDescent(font,12),
                                                    50,LANG['deflection']);
    ctx.drawTextCenter( font, 12, 120 - ctx.fontDescent(font,12),
                                                        50,LANG['action']);
    //legenda
    ctx.strokeStyle = "#0000ff";     //clovek
    ctx.drawTextCenter( font, 10, 110, outY,LANG['player']);
    ctx.strokeStyle = "#00ff00";    //regulator
    ctx.drawTextCenter( font, 10, 150, outY,LANG['pid']);

      //soucet odchylek pro ohodnoceni kvality regulace
    diff['player_deflection'] += (deflectionsGraph.values[cykl][0]-
         deflectionsGraph.values[cykl][1])*(deflectionsGraph.values[cykl][0]-
                                        deflectionsGraph.values[cykl][1]);
      //soucet kvadratu odchylek pro ohodnoceni kvality regulace
    diff['pid_deflection'] += (deflectionsGraph.values[cykl][0]-
         deflectionsGraph.values[cykl][2])*(deflectionsGraph.values[cykl][0]-
                                         deflectionsGraph.values[cykl][2]);
    diff['player_action'] += (actionsGraph.values[cykl][0]) *
                                           (actionsGraph.values[cykl][0]);
      //soucet kvadratu akcniho zasahu pro ohodnoceni kvality regulace
    diff['pid_action'] += (actionsGraph.values[cykl][1]) *
                                           (actionsGraph.values[cykl][1]);
```

```
    deflectionsGraph.paint(ctx,cykl);
    actionsGraph.paint(ctx,cykl);
  }
  if(gameStatus == 1){
    //hra v rezimu 1 - automaticke rizeni
    if(cykl + 1 < graphDeflection.values.length)
       reference = graphDeflection.values[cykl+1][0];
    ctx.strokeStyle = "#d8b970";
    //sipka
    arrow(beamCenterX - reference*beamWidth*vector[0],beamCenterY - ballR -
                                                                          40);
    //PD regulator
    controlError = -reference - x;
    controlAction = 0.7211*oldCa-5.56*controlError+5.245*oldCe;
    oldCa = controlAction;
    oldCe = controlError;
    if(controlAction > 1) controlAction = 1;
    if(controlAction < -1) controlAction = -1;
  }
  if(gameStatus == 0){
    //hra v rezimu 0 - rucni rizeni
    //generator nahodne reference, v urcitych casovych intervalech
    ctx.strokeStyle = "#d8b970";
    if(cykl % 2 != 0 || arrowBlink > 5){
      arrow(beamCenterX + reference*beamWidth*vector[0],beamCenterY - ballR -
                                                                  40);  //sipka
      arrowBlink++;
    }
    if(cykl*Ts % referenceChangePeriod == 0){
      oldRef = reference;
       //minimalni zmena reference o 0.2
      while(Math.abs(reference - oldRef) < 0.2){
         reference = 0.7 - Math.random()*1.4;
      }
    }
  }
  if(gameStatus < 2){
    //hra v rezimu 0 nebo 1
    ctx.strokeStyle = "#000000";
    var font = "times";
    var outY = HEIGHT - ctx.fontAscent(font,8);
    var time = LANG['time'] + Math.round(cykl*Ts) + LANG['time_fin'] +
                                  (gameTime - Math.round(cykl*Ts)) + " s";
    ctx.drawTextCenter( font, 8, WIDTH - 120, outY,time);
    //popis grafu
      //popis osy x u akcniho zasahu
    ctx.drawTextCenter( font, 8, 460, 125,"t [s]");
      //popis osy x u vychylky
    ctx.drawTextCenter( font, 8, 460, 265,"t [s]");
    outY = 5 + ctx.fontAscent(font,8);
    ctx.drawTextCenter( font, 8, 380, outY,LANG['action']);
    outY = 125 + ctx.fontAscent(font,8);
    ctx.drawTextCenter( font, 8, 380, outY,LANG['deflection']);
    //rovnice motoru, aprox soustavou 1. radu
    angle = oldAngle - Ts/beamT*(oldAngle + beamK*controlAction*10);
    if(Math.abs(angle) > MAX_ABS_ANGLE){
      angle = MAX_ABS_ANGLE*sign(angle);
    }
      //prepocet pro rotacni matici
    realAngle = 135 + angle;
    oldAngle = angle;
```

```
    if(gameStatus == 0){                                                //rezim 0
      graphDeflection.add(cykl/2,[-reference,-x]);
      graphAction.add(cykl/2,[-reference,controlAction]);
      deflectionsGraph.add((cykl/2),[-reference,-x, 0]);
      actionsGraph.add(cykl/2,[controlAction,0]);
      graphDeflection.paint(ctx,cykl);
// [0,1] znaci vyber vykreslovanych car(fci),prvni je reference zde vypnuta..
      graphAction.paintSelected(ctx,cykl,[0,1]);
    }
    if(gameStatus == 1){                                               //rezim 1
      graphPIDDeflection.add(cykl/2,[reference,-x]);
      graphPIDAction.add(cykl/2,[reference,controlAction]);
      deflectionsGraph.addVal((cykl/2),-x, 2);
      actionsGraph.addVal(cykl/2,controlAction, 1);
      graphPIDDeflection.paint(ctx,cykl);
      graphPIDAction.paintSelected(ctx,cykl,[0,1]);
    }
    //trojuhelnik, podstavec pod tyci
    ctx.fillStyle = "#8c8c8c";
    ctx.beginPath();
    ctx.moveTo(beamCenterX, beamCenterY );
    ctx.lineTo(beamCenterX + 5,beamCenterY + beamHeight + 5);
    ctx.lineTo(beamCenterX - 5,beamCenterY + beamHeight + 5);
    ctx.closePath();
    ctx.fill();
  //transformace
  //m1 = [Math.cos(angle),-Math.sin(angle)];
  //m2 = [Math.sin(angle),Math.cos(angle)];
  //rotateMatrix = [m1,m2];
  //[x,y]*rotateMatrix
    //rotace jednotkoveho vektoru o uhel realAngle
    vector[0] = baseVector[0] * Math.cos(realAngle * Math.PI / 180)-
                            baseVector[1]*Math.sin(realAngle * Math.PI / 180);
    vector[1] = baseVector[0] * Math.cos(realAngle * Math.PI /
                        180)+baseVector[1]*Math.sin(realAngle * Math.PI / 180);
    ctx.strokeStyle = beamColor;
    ctx.beginPath();
    //tyc
    ctx.moveTo(beamCenterX-beamWidth*vector[0] , beamCenterY -
                                                      beamWidth*vector[1]);
    for(var i=0;i<beamHeight/2;i++){
     ctx.lineTo(beamCenterX+beamWidth*vector[0],
                                          beamCenterY+beamWidth*vector[1]+i);
     ctx.lineTo(beamCenterX-beamWidth*vector[0], beamCenterY-
                                                  beamWidth*vector[1]+i);
    }
    ctx.closePath();
    ctx.stroke();
    //ukazatel prubehu velikosti akcniho zasahu
    ctx.drawImage(imageMouseBar,10,beamCenterY+100);
    ctx.strokeStyle = "rgba(255,0,0,0.8)";
    ctx.beginPath();
    //ukazatel mysi (cerveny)
    ctx.moveTo(beamCenterX + controlAction*beamWidth,beamCenterY+100);
    ctx.lineTo(beamCenterX + controlAction*beamWidth,beamCenterY+120);
    ctx.closePath();
    ctx.stroke();

    var x1 = beamCenterX - beamWidth*vector[0];
    var y1 = beamCenterY - beamWidth*vector[1];
```

```
    var k = vector[1]/vector[0];
    //rovnice primky
    y = k*(x*beamWidth*vector[0] + beamCenterX - x1) + y1;
//protoze je to pocitano ve stredu kulicky, je potreba odecist jeji polomer
    y -= ballR + 1;
    v = oldV + Ts*(-ballC*angle - ballB*oldV);
    x = oldX + Ts*oldV;
    if(Math.abs(x) > 1){
      v = 0;
      x = oldX;
    }
    oldX = x;
    oldV = v;
    ctx.fillStyle = ballColor;                //kulicka
    circle(beamCenterX + x*beamWidth*vector[0], y, ballR);
  }
  cykl++;
  //rozhodovaci cast hry
  //konec hry v rezimu 0, manualni rizeni (+gameTime s)
  if(gameStatus < 1 && cykl*Ts > gameTime){
      gameStatus = 1;                                   //zmena rezimu hry
      //vynulovat vsechny konstanty, priprava na automaticke rizeni
      reference = 0;
      cykl = 0;
      x = 0;
      v = 0;
      oldX = 0;
      oldV = 0;
      angle = 0;
      controlAction = 0;
      clearInterval(intervalId);
      ctx.fillStyle = backColor;
      clear();
      var font = "arial";
      var outY = HEIGHT/2 - ctx.fontAscent(font,10);
      ctx.strokeStyle = "rgba(50,0,0,0.75)";
      ctx.drawTextCenter( font, 10, canvas.width/2, outY,LANG['now_look']);
      ctx.drawTextCenter( font, 10, canvas.width/2, outY +
                                    ctx.fontAscent(font,14),LANG['for_cont']);
      //cekani na click (onClickHandler game.lib.js)
  //konec hry v rezimu 1, automaticke rizeni (+gameTime s)
  }else if(gameStatus == 1 && cykl*Ts > gameTime) {
      gameStatus = 2;
      cykl = 1;
      ctx.fillStyle = backColor;
      clear();
  //konec hry v rezimu 2 (+2 s)
  }else if(gameStatus == 2 && cykl*Ts < 2) {
      font = "arial";
      outY = HEIGHT/2 - ctx.fontAscent(font,10);
      ctx.strokeStyle = "rgba(50,0,0,0.75)";
      ctx.drawTextCenter( font, 10, canvas.width/2,
                                                  outY,LANG['now_you_will_see']);
  }else if(gameStatus == 2 && cykl*Ts >= 2){
      gameStatus = 4;
      cykl = 1;
  //konec hry v rezimu 4 (+gameTime s), prechod na zaver hry.
  }else if(gameStatus == 4 && cykl*Ts > gameTime){
      cykl = 1;
      gameStatus = 5;
      clearInterval(intervalId);
```

```
        ctx.fillStyle = backColor;
        //smazat jen nadpisy
       ctx.clearRect(0, 0, WIDTH, 80);
       var rho = 0.1;
       var qualityPID = rho * diff['pid_action'] + diff['pid_deflection'];
       var qualityPlayer = rho * diff['player_action'] +
                                                    diff['player_deflection'];
       var quality = Math.round(100*Math.exp(-0.005*(qualityPlayer -
                                                        qualityPID))*100)/100;
       summary = LANG['difference']+quality+' %';
       if(diff['player_deflection'] < diff['pid_deflection']){
          summary = LANG['you_are_better'];
       }

      var font = "arial";
      var outY = 30 + ctx.fontAscent(font,15);
      ctx.strokeStyle = "rgba(50,0,0,0.75)";
      ctx.drawTextCenter( font, 15, canvas.width/2, outY,LANG['game_over'] +
                                                                  summary);
      ctx.strokeStyle = "rgba(50,0,0,0.75)";
      ctx.drawTextCenter( font, 14, canvas.width/2,
                                                 outY+30,LANG['play_again']);
     }
}
```

Příloha B

Obsah přiloženého DVD

- Kompletní zdrojové kódy celého webu
- Zdrojové kódy motivační hry
- Zdrojové kódy autentifikace pomocí LDAP serveru