Uporaba odprtokodnih spletnih tehnologij v krmilnih sistemih

Iztok Fekonja
DIKA d.o.o., Na Ajdov hrib 14, 2310 Slovenska Bistrica
iztok.fekonja@gmail.com

Using open source web technologies in control systems

In this contribution is represented concept and implementation of web browser interface for remote supervising and controlling of factory automation control, based on usage of GPL open source project libraries. Represented are also development requirements for each interface parts. Shown is also brief explanation of problematics at choosing of proper JavaScript libraries. Technological frame of solution is based on TCP/IP HTTP protocol and jQuery JavaScript libraries. Software model is following MVC design and is implemented by JSON data exchange format, HTML graphical objects and AJAX data refreshing.

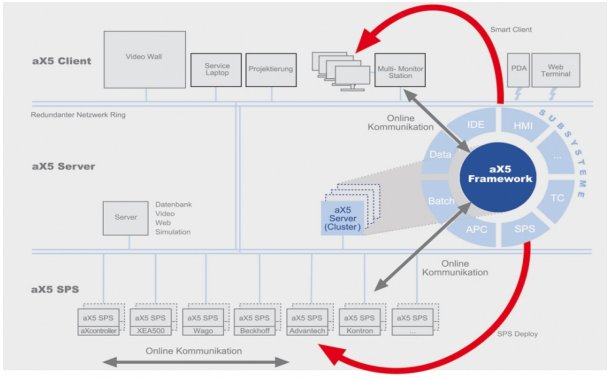
Kratek pregled prispevka

V članku je predstavljena zasnova in izvedba spletnega vmesnika za daljinski nadzor in upravljanje industrijskega avtomatizacijskega krmilja na osnovi odprtokodnih spletnih knjižnic. Prikazana so tudi izhodišča in zahteve za celoto in posamične dele vmesnika. Obrazložena je tudi problematika izbire primernih programskih modulov in pogojenost izvedbe s strojnim okoljem, kjer se platfoma izvaja. Tehnološki okvir se bazira na TCP/IP HTTP protokolu ter jQuery/JavaScript odprtokodnih knjižnicah. Programski model MVC je implementiran z JSON podatkovnim formatom, HTML obliko in AJAX podatkovnim osveževanjem.

1 Uvod

1.1 Opis avtomatizacijskega okolja

Kot avtomatizacijsko okolje je v tem članku mišljeno predvsem platforma »ax5« podjetja AutomationX [1], ki je sedaj del industrijskega koncerna GAW [2]. Shematska zasnova tega okolja je prikazana na spodnji sliki 1:



Slika 1: Shematska zasnova »ax5« krmilja.

Krmilje je razdeljeno na sledeče nivoje:

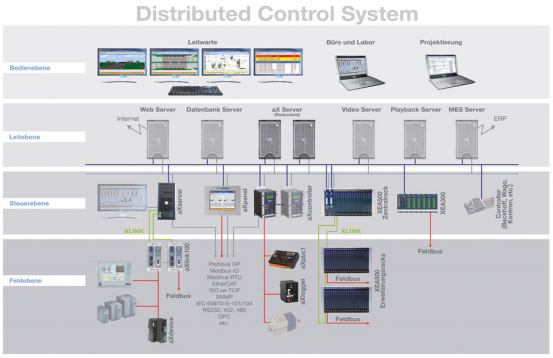
- ax5/Client uporabniški programski moduli
- ax5/Server za zajem in shrambo podatkov
- ax5/PLC krmilje (kombinacija krmilnikov)

Strojno avtomatizacijsko krmilje je izvedeno z AutomationX PLC krmilniki v kombinaciji z krmilniki in vhodno/izhodnimi enotami drugih proizvajalcev. Sistemska programska platforma je večinoma Windows, v kritičnih okoljih pa Linux. Procesorska platforma krmilnika je večinoma klasični Intel Procesor, ponekod pa vedno bolj pogosto ARM9 [3].

Vedno bolj je v uporabi krmilnik s pasivnim hlajenjem, zaradi uporabe v agresivnih okoljih, ki lahko povzročajo zastoj ventilatorja, kot je recimo v ladjedelništvu (NGSM- ZDA [4])

Zahteve po funkcionalnosti krmilnika so velike in večinoma dosegajo polno zmogljivost strojne opreme. Zato je potrebno izvajanje vseh procesov krmilnika zelo skrbno optimirati in omejiti njihovo potrebo po strojni zmogljivosti.

Velikostni razred večjih projektov je okoli 40.000 podatkovnih objektov z okoli 300.000 podatkovnimi spremenljivkami. Konceptualna shema strojne platforme avtomatizacije je podana na sliki 2.



Slika 2: Shematski prikaz ax5 strojne platforme.

Iz prikaza je razvidna raznolikost strojne opreme, ki hkrati tudi zahteva uporabo številnih programskih komunikacijskih protokolov za izmenjavo in shrambo podatkov. Izvajanje teh dejavnosti je zato združeno v skupine opravil, ki se odvijajo v procesih avtomatizacijske platforme. Primer nekaj takih procesov:

- ax5device (nadzorni proces),
- ax5plc (časovno kritična PLC koda),
- ax5slowplc (nekritična PLC koda),
- ax5datarec (dostop do podatkov) in
- ax5web (spletni strežnik).

Običajna konfiguracija vsebuje 10-20 takih krmilnih procesov, ki so medsebojno povezani preko notranje lastne podatkovne baze. Zato je zelo pomembno, da ima uporabnik vedno možnost preverjanja in upravljanja delovanja krmilja in posamičnih modulov.

1.2 Oris uporabniških zahtev

Za izvedbo osnovnega nivoja upravljanja potrebujemo vmesnik, ki obsega hiter pregled stanja izvajalnih procesov in zagotovljen poseg v nenačrtovano stanje, kot je na primer nepričakovano iztirjenje katerega od procesov, ki zaradi nepredvidene porabe procesorskih in pomnilniških zmogljivosti lahko privede do blokade drugih kritičnih programskih modulov in posledično do zastoja celega krmilja. Popolni zastoj upravljanja krmilnega vodila je sicer

varovan v obliki redundančnega sistema, vendar v primeru hkratne odpovedi obeh, preostane samo ponovni zagon celotnega krmilja, ki mora biti čimkrajši, saj je od trajanja zagona lahko odvisno tudi človeško življenje.

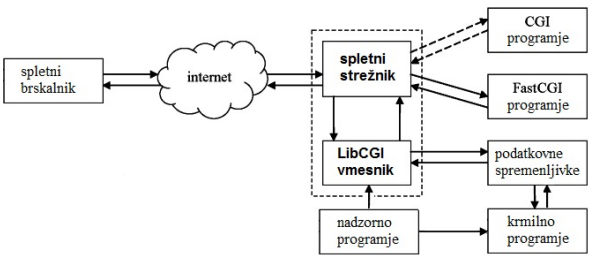
Vsi vgrajeni procesi krmilnika morajo stremeti k čimbolj enostavni zgradbi koncepta in karseda omejeni uporabi zunanjih knjižnic oziroma programskih modulov, ki bi vnesli dodatne nepotrebne zahteve in nestabilnosti.

Prav tako mora vsak proces zadostiti zahtevi po prenosljivosti izvorne programske kode, ki je zato zapisana v C in C++, s premišljeno uporabo specifičnih zunanjih programskih knjižnic, saj je tudi samo okolje krmilnika lahko zelo omejeno. Konfiguracija platforme krmilnika je namreč zelo raznolika in zato mora bit nabor vseh uporabljenih programskih modulov (procesov) skalirno zastavljen, da se le-ti lahko izvajajo na majhnem krmilniku, pa tudi na mnogo večjem industrijskem strežniku.

1.3 Tehnološke zahteve in omejitve

Ob poglobljeni analizi tehnoloških zahtev se izkaže, da je za namen spletnega dostopa bolje uporabiti posebej za ta namen razvit lasten programski modul, kot kak standardni strežnik, recimo Apache [16]. Prav tako so iz koncepta izvzete zahtevnejše tehnološke rešitve kot je recimo PHP [5], saj se je v dosedanjih projektih pokazalo, da niso ustrezna razvojna pot.

Običajna tehnološka rešitev te problematike je v obliki uporabe CGI [6] in FastCGI [7], mehanizma, ki opazno reducira strežniško obremenitev, kot lahko vidimo na sliki 3.



Slika 3: Standardni CGI oz. FastCGI spletni dostop.

FastCGI koncept ima veliko uporabno vrednost, vendar v našem primeru zahteva dodatno namestitev klasičnega spletnega strežnika, kar pa ni sprejemljivo. Najbolj smiselno je tako kodiranje lastnega, čeprav omejenega, spletnega strežnika, ki mora ustrezati sledečim zahtevam:

- prenosljiva izvorna koda (Linux,Windows),
- preprosta in pregledna zasnova,
- podpora konceptu uporabniške seje,
- samozadostnost programske namestitve,
- preverljivost uporabljenih modulov,
- uporabljivost šibkih internetnih povezav,
- hiter dostop do posamične funkcionalnosti,
- zmogljiv pregled velike količine podatkov in
- uporaba povprečno zmogljivih terminalov.

Prednost prenosljivosti avtomatizacijskega paketa AutomationX je ravno v dejstvu, da je opremljena z zgodovino programskih sprememb in potrebnimi certifikati zrelosti kode, kar je zahteva pri uporabi avtomatizacije v ladjedelništvu (NGSM [4]). Posledično pa je tudi zahteva po preverljivosti in preglednosti vgrajenih programskih modulov, ki morajo biti pod nadzorom lastne programske hiše.

Zato je bilo potrebno razviti koncept lastnega spletnega strežnika, ki pa to dejansko ni, ampak je le zelo omejen programski modul, ki prestreza HTTP [8] GET/POST spletne zahteve in jih izvaja na podoben način, kot to počne standardni spletni strežnik. Na ta način je bil dosežen izjemno majhen odtis strojne porabe, prav tako pa je procesni modul popolnoma pod nadzorom jedra krmilnega avtomatizacijskega modula. Naloge takega spletnega strežnika so sledeče:

- izmenjava zahtevanih datotek,
- standardiziran podatkovni dostop in
- varnostni nadzor izvajanja zahtevkov.

Tak minimalistični koncept omogoča spletnemu programskemu modulu, da se lahko izvaja tudi v kritičnih okoliščinah, kjer so običajne oblike izvajanja uporabniškega vmesnika že odpovedale (vizualizacija preko X11/Motif [17], .NET [18], ...).

Na uporabniški strani je posledično s tem povezano tudi zahteva po preprostem konceptu spletnega vmesnika, ki prednostno uporablja

računalniške zmogljivosti na uporabniški strani. Strežniške rešitve tipa PHP namreč preveč obremenjujejo strojne zmogljivosti krmilnika.

Ena od možnih rešitev problematike bi lahko bila tudi uporaba Java programskega modula, kar pa ima sledeče pomanjkljivosti, ki so se že pokazale v nekem drugem podobnem projektu:

- Java vtičnik mora bit posebej nameščen,
- zaprtost kode, težavnost nadgradnje in
- certificiranje verzij.

Z uporabo običajnega zmogljivega spletnega brskalnika vsi ti problemi odpadejo, saj izvajanje JavaScript programske kode ne zahteva dodatnih namestitev, povrhu pa je odprta do uporabnika in jo lahko le-ta nadgrajuje po lastnih potrebah.

Z uporabo AJAX [12] metode podatkovnega osveževanja postane grafično prikazovanje tekoče in prijetno na pogled.

Programski jezik JavaScript [9] se izkaže kot smiselna izbira, saj je v zadnjem času pridobil na popularnosti zaradi široke programske podpore v odprtokodni skupnosti v obliki zelo zmogljive knjižnice jQuery [10]. Ta v sebi podpira rešitve kot so: AJAX, XML, JSON,...

Osnovne zamisli pri iskanju kombinacije izvedbe spletnega vmesnika so bile:

- enostavna in pregledna zgradba,
- hitra in učinkovita uporaba ter
- odprtokodni koncept.

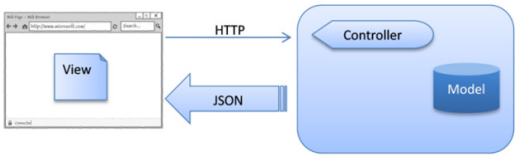
Idejni koncept je tako predvideval osnovni vstopni menu, ki uporabniku omogoči takojšen pregled stanja krmilnika, bolje rečeno sistema, kajti na vpogled je stanje obeh redundančnih krmilnikov hkrati.

Spletna stran sistemskega menija se mora naložit hipno na kakršnemkoli JavaScript terminalu (mobilnem telefonu, namiznem ali tabličnem računalniku, ...), saj s tem omogoča hiter poseg v problematično stanja krmilja in preprečitev nastajanja večje škode.

Osnovno vodilo pri sestavljanju koncepta je bilo preprostost uporabe in odzivnost, zato je sama estetika podobe vmesnika enostavna, saj bi vsaka dodatna nepotrebna dekoracija samo omejevala uporabnost vmesnika.

1.4 Izbira koncepta

Pri razvoju idejne zasnove je bil uporabljen koncept MVC (model-view-controller), ki je poenostavljeno predstavljen na sliki 4:



Slika 4: MVC (model-view-controller) koncept.

Posamične komponente so zasnovane tako:

- »Model«: Podatkovni format JSON [11] omogoča standardizirani dostop do kontrolnega nivoja, ne glede na dejansko izvedbo spletnega vmesnika, ki ga lahko uporabnik prilagaja neodvisno od različice programske opreme krmilnika.
- »View«: HTML grafični in tekstovni objekti so definirani in poimenovani ločeno od ostalega programja, kar omogoča enostavnejše prilagajanje uporabniškim zahtevam. Mehko in tekoče prehajanje prikaza stanj je podprto s strani brskalnika.
- »Control«: Spletni koncept AJAX [12] je v bistvu samodejni osveževalnik, ki se izvaja kot notranja JavaScript zanka in v ozadju spletnega vmesnika periodično osvežuje tabelo krmilnih podatkov, ki navzkrižno združuje reference HTML objektov in podatkovnih spremenljivk. Tako lahko uporabnik poljubno nadgrajuje nabor nadzorovanih spremenljivk brez poseganja v sam koncept podatkovne povezave.

1.5 Izbor programskih modulov

Po preizkusih vseh možnih kombinacij različnih odprtokodnih knjižnic se je oblikoval sledeč nabor komponent:

- »jquery.js«: [10] JavaScript knjižnica, verzija 1.11, ki je še v prilagodljivi in

prenosljivi obliki. Verzija 2.0 je namreč že omejena na najnovejše spletne brskalnike.

- »json2.js«:[11] podatkovni format JSON služi za podporo standardizirane podatkovne izmenjave in nadgradnji spletnega vmesnika.
- »dataTables.js«:[13] zmogljiva podatkovna tabela s podporo sortiranju in listanju.
- »fancytree.js«:[14] omogoča drevesni prikaz podatkovnih spremenljivk in drugih programskih komponent krmilnika.
- »easytabs.js«:[15] omogoča uporabo različnih jQuery objektov na isti spletni strani preko preklapljanja zavihkov. Bistvena prednost je zmožnost hkratnega izvajanja jQuery modulov, kar omogoča prepletanje predstavitvenih stanj spletnih podstrani.

Pri testiranju posamičnih kombinacij sklopov so zaradi medsebojne nezdružljivosti odpadle kakšne morebiti boljše rešitve od prej navedenih. Nezdružljivost se je pokazala kot manjkajoči predeli osveževanja zaslona, oziroma kot napačno izvajanje funkcionalnosti v primerjavi z izolirano namestitvijo tega modula.

Navedene komponente imajo zelo dobro prenosljivost in sposobnost hitrega zagona tudi v mobilnikih slabše kvalitete, povrh pa jih odlikuje majhna velikost datotek, kar je bistvena tudi za zmožnost uporabe slabših internetnih povezav.

Naloga spletnega strežnika je, da preko standardiziranih mehanizmov omogoči dostop do notranjih podatkovnih struktur aplikativnega programskega modula, v našem primeru krmilnika. Navadno se to izvaja preko CGI skript, oziroma preko FastCGI ukazov, ki omogočajo klice programskega vmesnika.

Slaba stran takega pristopa je oteženo sledenje konceptu »seje«, kjer se dostop izvaja v obliki avtomata stanj, kar je v primeru ločenih procesov in brez PHP platforme težje izvedljivo.

Zato je funkcionalnost spletnega strežnika in CGI vmesnika prevzel nov proces »ax5web«, ki je izveden v C izvorni kodi, ter uporablja lasten aplikativni programski vmesnik za izmenjavo podatkov in upravljanje krmilnih procesov.

2 Izvedba spletnega vmesnika

Spletni vmesnik je zastavljen kot preprosto, vendar učinkovito orodje, saj je glavno vodilo koncepta optimalni pretok podatkov ob čimmanjši porabi procesorskega časa na obeh straneh orodja (strežnika in brskalnika). Hkrati pa je tudi dovolj enostaven za nadaljnjo nadgradnjo in prilagoditev uporabniku. Tako je vstopna spletna stran »index.html« zastavljena na način, da je hkrati tudi edina spletna stran, ki jo je potrebno spreminjati, vsa ostala koda je izvedena v uvoženih JavaScript knjižnicah.

Problem pri programskem združevanju posamičnih komponent v celoto je bilo iskanje združljive kombinacije različnih JavaScript knjižnic, ki se praviloma ne združujejo najbolje, saj vsaka zase rešuje samo določen problem brez ozira na ostale knjižnice z iste jQuery platforme.

Žal na področju JavaScript jezika še ni definiranega industrijskega standarda, zato se večina programerjev zateka k uporabi najbolj uporabljene in podprte JavaScript knjižnice jQuery, ki dejansko uspeva pokrivat večino splošnih potreb, kar pa je za naš primer premalo. Ravno tako je problem marsikake knjižnice, da izraža prevec »kričeč« dizajn, ki v primeru industrijske avtomatizacije samo zavaja in upočasnjuje reakcijski čas uporabnika.

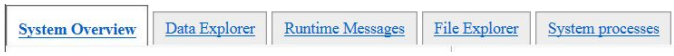
Prav tako je problem doseganje podobnosti obnašanja spletnega vmesnika na različnih spletnih brskalnikih (Firefox, MS-IE, Chrome), ki so vsi prisotni v različnih verzijah.

Zgradba vmesnika je razdeljena v več podsklopov, ki jih bomo spoznali v nadaljevanju.

2.1 Povezovalni sistemski meni z zavihki

Povezovalni meni , ki ga vidimo na sliki 5, je izveden na osnovi jQuery knjižnice »easyTabs« [15] in omogoča preklapljanje med različnimi spletnimi okni, ki se izvajajo znotraj iste uporabniške seje s spletnim strežnikom na

krmilniku. Tak koncept izboljšuje preglednost nad dogajanjem, hkrati pa združuje zaokroženost spletnih podstrani. Problem same izvedbe je v dejstvu, da navkljub splošni iskanosti, manjka standardizirana podpora tovrstnim zavihkom v obstoječih brskalnikih.



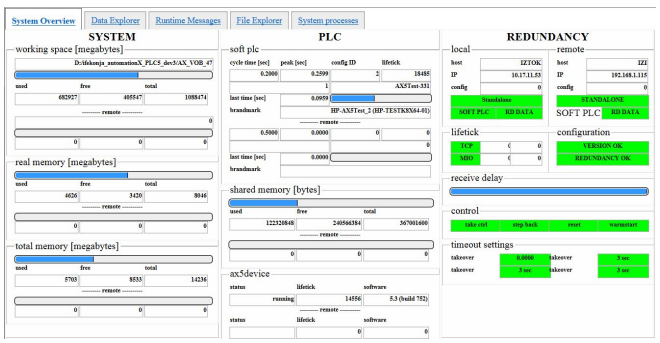
Slika 5: Povezovalni sistemski meni z zavihki.

V osnovnem meniju vidimo sledeče zavihke:

- »System Overview«: celosten pregled stanja, kjer uporabnik dobi vtis o trenutni porabi delovnega in sistemskega pomnilnika obeh krmilnikov ter stanje podatkovne povezave z redundančnim krmilnikom
- »Data Explorer«: pregled trenutnega stanja podatkovnih spremenljivk in možnost nastavljanja želenih vrednosti.
- »Runtime Messages«: prikazuje sistemska sledilna sporočila in s tem omogoča sledenje dogajanju v sistemskih programskih modulih, ter posledično sklepanje o morebitnih vzrokih nastalih težav.
- »File Explorer« : omogoča pregled in dostop do datotečnega sistema in nadzor nad datotekami
- »System Processes«: pregled in nadzor trenutnega stanja sistemskih procesov

2.2 *Osnovni sistemski pregledni meni*

Programski modul »System Overview« mogoča vpogled v stanje sistema, kot to lahko vidimo na sliki 6:



Slika 6: Sistemski pregled stanja krmilja.

Uporaba grafičnih stolpcev omogoča strnjen in hipen pregled nad trenutnim stanjem sistema.

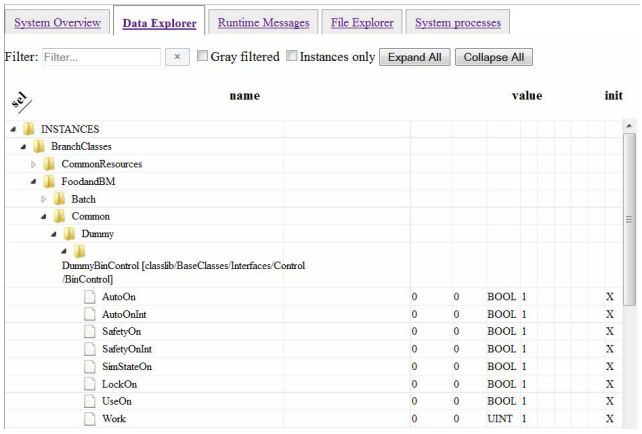
Na desni strani so vidne kontrolne tipke za neposredno upravljanje krmilja:

- »Take ctrl«: prehod v MASTER stanje, kjer trenutni krmilnik prevzame vodenje redundančnega sistema in prisili drug krmilnik v stanje rezervnega načina.
- »Step back«: prehod v STANDBY stanje, uporabnik zavestno prepusti nadzor redundančnemu krmilniku in trenutni krmilnik preide v rezervni način.
- »Reset«: prekine morebitno nedefinirano stanje, kjer oba krmilnika pričneta s ponovnim protokolom pogajanja, kjer se določi glavni/rezervni krmilnik
- »Warmstart«: mehki ponovni zagon sistema, kjer se ponovno naloži nameščen projekt s trenutnimi stanji podatkovnih spremenljivk, ter s tem prekine morebitno mrtvo zanko programskih gonilnikov

Sistemski pregledovalnik za svoje delovanje ves čas intenzivno uporablja preko 100 sistemskih podatkovnih spremenljivk, zato je bil uporabljen optimiran in prilagodljiv algoritem, ki omogoča majhno porabo procesorske moči na obeh straneh, uporabniški in strežniški.

2.3 *Pregled podatkovnih spremenljivk*

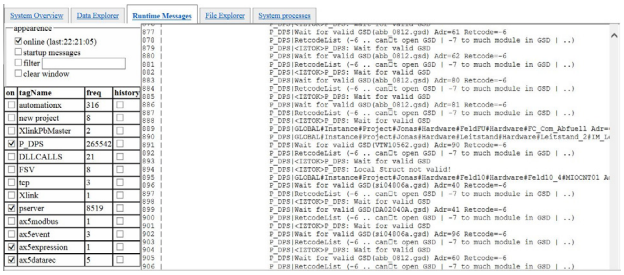
Vmesnik omogoča drevesni pregled vseh podatkovnih spremenljivk s trenutnimi ter začetnimi vrednostmi, kot je razvidno na sliki 7:



Slika 7: Pregledovalnik spremenljivk.

2.4 *Pregled sistemskih sledilnih sporočil*

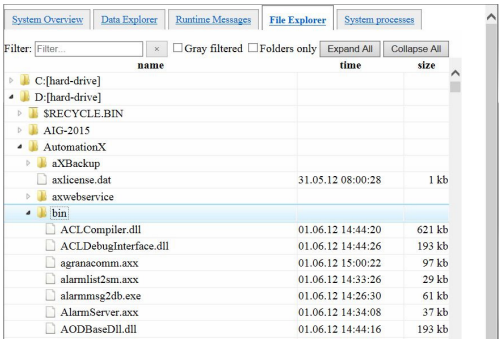
Vgrajeni krmilni programski moduli imajo možnost pošiljanja informativnih sporočil, na osnovi katerih lahko sistemski tehnik spremlja dogajanje v izvajanju tega modula. Ta del orodja je v bistvu najbolj pomemben del vmesnika, saj lahko procesnemu tehniku pomaga k hitri razjasnitvi zapletov pri nastavitvi parametrov vhodno izhodnih komunikacijskih protokolov. Vmesnik je izveden z kombinacijo klasičnih HTML objektov »table« in »textarea«, saj je bilo potrebno izpolnit zahtevo po hitrosti osveževanja in preprostosti prikaza množičnih sporočil, kot je razvidno na sliki 8:



Slika 8: Pregledovalnik sistemskih sporočil.

2.5 *Pregledovalnik datotek*

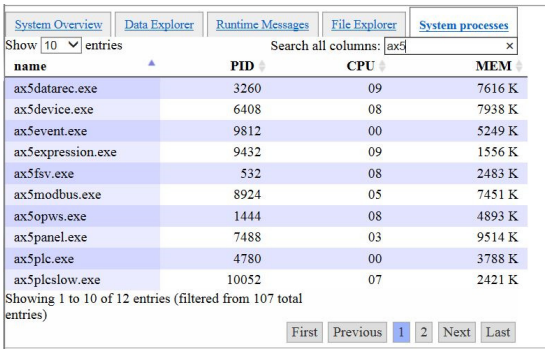
Vmesnik služi za pregled stanja datotek in upravljanja z njimi. Njegova izvedba je zelo podobna zgradbi podatkovnega pregledovalnika in je prav tako izvedena na osnovi knjižnice »fancyTree« [14]. Na sliki 9 je prikaz tega okna:



Slika 9: Pregledovalnik datotek.

2.6 *Pregled stanja procesov*

Nudi pregled trenutnega stanja sistemskih procesov in ponovni zagon katerega od njih v primeru, da uporabnik pride do analitičnega sklepa o smiselnosti ponovnega zagona tega procesa (recimo ob nenormalni porabi procesne moči ali pomnilnika. Vmesnik je izveden na osnovi jQuery knjižnice »DataTables« [13], ter ga lahko vidimo na sliki 10:



Slika 10: Pregledovalnik sistemskih procesov.

3 Povzetek

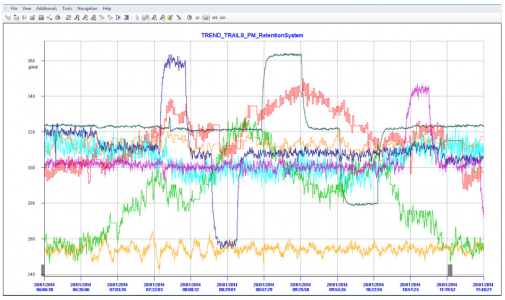
Dosedanja testiranja so potrdila pravilnost zasnove idejnega koncepta spletnega vmesnika, saj izpolnjuje vse zahteve in pričakovanja glede prenosljivosti in funkcionalne uporabnosti.

Nadaljnja nadgradnja vmesnika bo potekala v smeri vizualizacijskega vmesnika analize trend podatkov in prikaza funkcijske logike, kar bo omogočalo lažji pregled nad trenutnim in posnetim stanjem izbranih podatkovnih spremenljivk in krmilnih tokokrogov.

Sledeče slike prikazujejo zaslonski zajem obstoječih programskih modulov, ki bodo služili kot idejna predloga za izvedbo v JavaScript kodi

3.1 *Grafični prikaz trend podatkov*

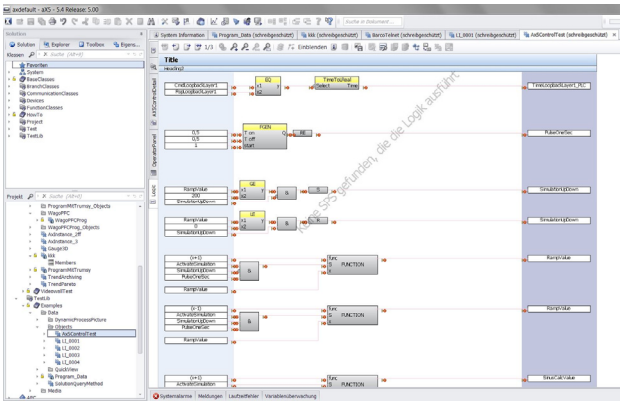
Za ugotavljanje vzročnosti nekega stanja podatkovne spremenljivke je najbolje uporabiti grafikon sprememb vzročnih spremenljivk, kot vidimo na sliki 11:



Slika 11: Grafični prikaz trend podatkov.

3.2 Vizualizacija logičnega krmilja

Koncept vizualizacijskega vmesnika bo sledil že uveljavljenemu hišnemu konceptu in bo izgledal približno tako, kot je vidno na sliki 12:



Slika 12: Vizualizacija logičnega krmilja.

4 Literatura

[1] AutomationX: http://www.automationx.com
[2] GAW: http://www.gaw.at
[3] ARM A9: http://www.arm.com/cortex-a9.php
[4] NGSM: http://www.sperrymarine.com
[5] PHP: http://php.net/
[6] CGI: http://www.w3.org/CGI
[7] FastCGI: http://www.fastcgi.com/drupal
[8] HTTP: http://www.w3.org/Protocols
[9] JavaScript: http://sl.wikipedia.org/wiki/JavaScript
[10] jQuery: http://jquery.com/
[11] JSON: http://www.json.org/
[12] AJAX: http://learn.jquery.com/ajax/
[13] DataTables: http://www.datatables.net
[14] FancyTree: http://wwwendt.de/tech/fancytree
[15] EasyTabs: http://os.alfajango.com/easytabs
[16] Apache: http://www.apache.org/
[17] X11: http://www.x.org/wiki/
[18] .NET: http://www.microsoft.com/net