SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Ing. Anton Pytel

Autoreferát dizertačnej práce

PREDIKTÍVNE METÓDY RIADENIA V IOT PROSTREDÍ

na získanie akademickej hodnosti doktor (philosophiae doctor, PhD.)

v doktorandskom študijnom programe:
mechatronické systémy

Miesto a dátum: Bratislava November 2016

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia

Na Oddelení informačných, komunikačných a riadiacich systémov, Ústave automobilovej mechatroniky Fakulty elektrotechniky a informatiky STU v Bratislave.

Predkladateľ: Ing. Anton Pytel
Ústav automobilovej mechatroniky, OIKR
FEI STU BA, Ilkovičova 3, 812 19, Bratislava

Školiteľ: prof. Ing. Štefan Kozák, PhD.
Ústav automobilovej mechatroniky, OIKR
FEI STU BA, Ilkovičova 3, 812 19, Bratislava

Oponenti: prof. Ing. Boris Rohal'-Ilkiv, PhD.
Ústav automatizácie, merania a aplikovanej informatiky
SJF STU BA, Námestie slobody 17, 812 31, Bratislava

doc. Ing. Tibor Krajčovič, PhD.
Ústav počítačových systémov a sietí
FIIT STU BA, Ilkovičova 2, 842 16, Bratislava 4

Autoreferát bol rozoslaný: .....

Obhajoba dizertačnej práce sa koná: ..... o ..... h

Na Fakulte elektrotechniky a informatiky STU v Bratislave.
Ilkovičova 3, 812 19, Bratislava

prof. Dr. Ing. Miloš Oravec
dekan FEI STU v Bratislave

ANOTÁCIA

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	mechatronické systémy
Autor:	Ing. Anton Pytel
Dizertačná práca:	Prediktívne metódy riadenia v prostredí IoT
Vedúci záverečnej práce:	prof. Ing. Štefan Kozák, PhD.
Miesto a rok predloženia práce:	Bratislava 2016

Hlavná myšlienka dizertačnej práce je modifikácia moderných, existujúcich metód automatického riadenia, tak aby boli implementovateľné do nových foriem realizácie, ktoré využívajú IoT prostredie. Dizertačná práca je zložená z dvoch základných častí a to teoretickej a praktickej. Teoretická práca sa opiera o opis moderných metód riadenia, konkrétne prediktívneho riadenia a ich modifikácií, tak aby boli aplikovateľné do reality v integrácií s modernými informačnými technológiami založenými na IoT architektúre. Praktická časť dizertačnej práce je založená na vývoji všeobecného programového systému, ktorý umožňuje modelovať, simulovať a konfigurovať prediktívne algoritmy riadenia. Významnou súčasťou dizertačnej práce je návrh a realizácia softvérového systému, ktorého nosnou časťou je IoT architektúra. V rámci tejto architektúry vzniká myšlienka prevádzkovať regulátor ako službu (Controller as a service - CaaS). Takto navrhovaná metodika riadenia je overená v praxi na IoT systéme, ktorý predstavuje inteligentnú domácnosť. V dizertačnej práci je porovnaná nová metodika s klasickým prístupom k informačnému systému pre riadenie.

Kľúčové slová: MPC, REST, IoT, CaaS

Obsah

Úvod

V posledných rokoch môžeme sledovať veľký rozmach pripájania do siete internet, či už mobilných zariadení, počítačov nehovoriac a sú prípady pripájania už aj chladničiek. Rovnako môžeme sledovať zvýšenie dostupnosti elektronických súčiastok a lacných zariadení typu Raspberry Pi. Okrem toho vidíme kontinuálny rozvoj v oblasti návrhu a vývoja softvéru nové jazyky, nové štandardy, nové spôsoby integrácie. Všeobecne môžeme povedať, že informačné technológie idú dopredu rýchlym tempom. Výsledkom rozvoja informačných technológii a spomínaných oblastí je vznik nových typov informačných systémov s názvom internet vecí - IoT (internet of things) systémy. Aktuálne je pojem IoT čoraz častejšie skloňovaný na konferenciách akademickej a rovnako aj komerčnej sféry. Rôzne popredné spoločnosti ako IDC, Gartner ai. zaoberajúce sa výskumnými a poradnými činnosťami v oblasti informačných a komunikačných technológií robia odhady využitia. Tvrdenie portálu www.crn.com: „Júnový prieskum trhu spoločnosti IDC predikoval, že výdavky na IoT dosiahnú v roku 2020 sumu vo výške 1,7 biliónov dolárov, zatiaľ čo Gartner predpovedal, že v tom istom roku bude pripojených 21 miliárd zariadení.“[1] V oblasti regulátorov určite v percentuálnom nasadení stále prevládajú konvenčné metódy typu PID, avšak dostávajú sa do praxe aj moderné metódy ako prediktívne alebo adaptívne riadenie. Zvýšiť mieru nasadenia moderných metód riadenia je možné práve overením ich funkčnosti a potvrdením ich lepších vlastností v praxi a následnou propagáciou týchto výsledkov.

Cieľom práce je spojenie novovznikajúcich IoT systémov s aplikovaním prediktívnej metódy riadenia (MPC - model predictive controller) do nich. Ak chceme hlavný cieľ prace špecifikovať do detailu, tak ide o overenie zavedenia myšlienky regulátor ako služba (CaaS - controller as a service) do praxe. V práci sa overuje konkrétne online forma prediktívneho riadenia s obmedzeniami a sledovaním referenčnej hodnoty, čo je vysvetlené v prvej časti práce. Myšlienka regulátor ako služba predstavuje koncept, v ktorom je celá zložitosť výpočtu akčného zásahu regulátora, ktorú MPC prináša, na serveri a akčný člen len vykonáva poskytnutý akčný zásah riadenej veličiny. V práci sú popísané detaily tejto myšlienky, implementácie, výhody a nevýhody tohoto prístupu. Regulátor sa v práci implementuje do reálneho IoT systému inteligentnej domácnosti. Prechod od opisu prediktívneho regulátora ku jeho overeniu tvorí časť popisujúca trendy softvérových architektúr. V tejto časti sú vysvetlené architektúry a architektonické princípy, ktoré sú význačné pre IoT systémy. V tejto časti je aj detailne vysvetlený pojem IoT. Keďže myšlienka CaaS by bez IoT systému nebola realizovateľná, rovnaké architektonické princípy sú použité pri

implementácii riešenia CaaS. Po opise implementácie prediktívneho regulátora do IoT systému je najdôležitejšia časť v práci a to overenie na konkrétnych fyzických zariadeniach. Overenie je vykonané na systéme udržiavania požadovanej hladiny intenzity osvetlenia. Na konci práce je posledná časť venujúca sa rozdielom medzi klasickým softvérovým systémom a IoT systémom z viacerých uhlov pohľadu, vychádzajúc z praktických skúsenosti návrhu, vývoja a prevádzkovania IoT systému.

Ciele dizertačnej práce

Ciele dizertačnej práce sú aplikovať prediktívny regulátor do IoT systému s využitím znalostí softvérovej architektúry a definovať myšlienku regulátor ako služba (CaaS) aj s jej overením. Na uskutočnenie hlavných cieľov sú potrebné nižšie definované kroky.

- Vybrať vhodný typ softvérovej architektúry.
- Navrhnúť a vytvoriť IoT systém.
- Implementovať prediktívny regulátor na IoT backend ako službu.
- Implementovať klientskú časť regulátora na IoT bráne.
- Regulovať proces udržiavania intenzity osvetlenia pomocou prediktívneho regulátora v zapojení cez servisnú aplikáciu princípom CaaS.

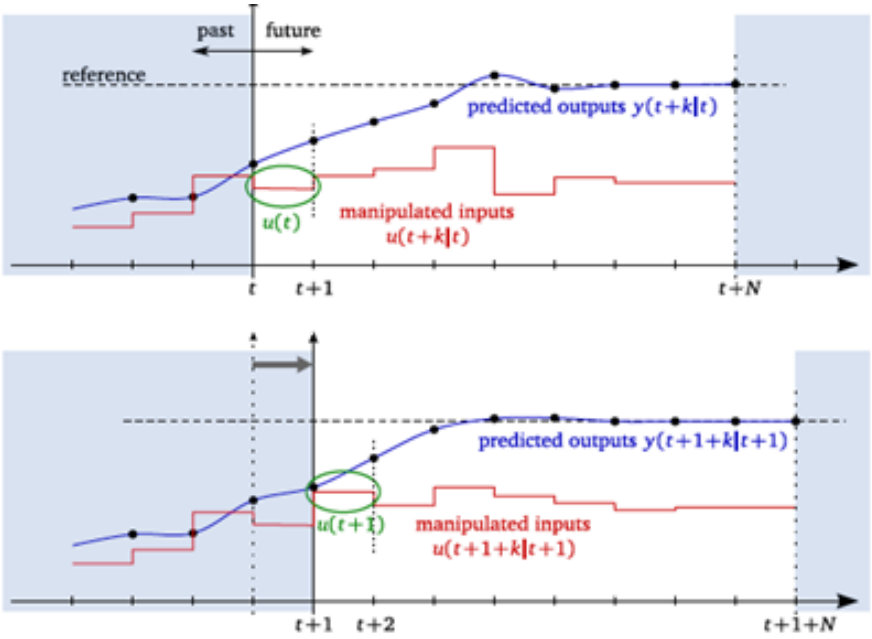
1 Prediktívne metódy riadenia

1.1 Teoretický základ MPC

V tejto časti sa vysvetlí matematický základ a variácie MPC regulátora.

1.1.1 On-line MPC

Prediktívny regulátor je, na najnižšej úrovni, metóda riadenia dynamických systémov, ktorá využíva nástroje matematickej optimalizácie. Spoločné črty všetkých prístupov riešenia problému prediktívnych regulátorov je vypočítať on-line, v každej časovej vzorke, optimálny riadiaci zásah v konečnom horizonte predikcie pre dynamický model systému, kde aktuálny stav je počiatočný stav. Iba prvý element z vypočítanej sekvencie predikovaných riadiacich zásahov je potom aplikovaný na systém. V ďalšom okamihu vzorkovania je horizont predikcie posunutý a výpočet optimálneho riadiaceho zásahu je vykonávaný znovu pre novonadobudnutý stav. Popisovaný princíp je znázornený na obrázku 1. Nech existuje lineárny dynamický model s časovo invariantnými parametrami,



Obrázok 1: Postupujúci horizont riadenia

vyjadrený diskrétnou stavovou rovnicou:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \\ x(t) \in R^n, y(t) &\in R^P, u(t) \in R^m, \\ A \in R^{(n\times n)}, B &\in R^{(n\times m)}, C \in R^{(p\times n)} \end{aligned} \tag{1}$$

x, y, u sú stavy systému, výstupy systému a riadiace zásahy v uvedenom poradí v čase alebo lepšie povedané vo vzorke t.
n – predstavuje počet stavov systému
p – predstavuje počet výstupov
m – predstavuje počet vstupov
Matice A, B, C sú systémová matica, vstupná matica a výstupná matica v uvedenom poradí. System 1 musí spĺňať nasledujúce obmedzenia na stav a vstup:

$$\begin{aligned} x(t) \in X, u(t) \in U \\ U \subset R^m \\ X \subset R^n \end{aligned} \tag{2}$$

kde obmedzujúca množina riadiacich zásahov U je konvexná, kompaktná (uzavretá a ohraničená) a obmedzujúca množina stavov X je konvexná a uzavretá. Pre obidve množiny U a X sa predpokladá, že obsahujú počiatok v ich vnútri.[15] Najskôr je jednoduchšie vyriešiť optimálne riadenie bez obmedzení. Pozornosť bude upriamená na nájdenie takej optimálnej postupností u^*(k), ..., u^*(k+N – 1), ktorá bude minimalizovať zvolené kvadratické kritérium a zároveň rešpektovať dynamiku systému. Čo inými slovami znamená, že tento regulátor sa bude snažiť minimalizovať stav a rovnako riadiaci zásah, ešte lepšie povedané ich kvadrát. Ak by teda nebola špecifikovaná referenčná hodnota, ktorú má výstup regulátora sledovať, tak implicitne výstupná veličina prediktívneho regulátora konverguje k hodnote 0 alebo pri systémoch s viacerými výstupmi k nulovému vektoru.
N – predstavuje horizont predikcie. Ak je teda systém v stave x_0, ktorý poznáme a horizont predikcie je 3, tak do optimalizačnej funkcie vstupujú premenné x_1, x_2, x_3 a u_1, u_2, u_3 kde u_1 zabezpečí prechod do stavu x_1, u_2 do stavu x_2 atď. každá premenná x je odvoditeľná z predošlého stavu a prvý stav x_0 je známy. Preto jedinou „neznámou“ ostáva sekvencia riadiacich zásahov.

Pre porozumenie ďalších vzťahov si je potrebné uvedomiť, že: $x(t+k) = x_{(t+k)}$

- Pre vzorku k = 0 (aktuálny stav systému): x(t)= x_t,
- pre vzorku k = 1 x(t+1)=$x_{(t+1)}$,
- atď.

Kvadratické kritérium pre konečný horizont predikcie dĺžky N je:

$$J(x(t), u(t), \ldots, u(N-1)) = \frac{1}{2}\sum_{k=0}^{N-1}[x_k^T Q x_k + u_k^T R u_k] + \frac{1}{2}x_N^T Q_N x_N \tag{3}$$

kde:

$$\begin{aligned} x_(k+1) &= Ax_k + Bu_k \\ x_0 &= x(t), \\ Q = Q^T \succcurlyeq 0, Q_N = Q_N^T \succcurlyeq 0, R = R^T &\succ 0 \\ Q_N &\in R^{n\times n} \\ R &\in R^{m\times m} \end{aligned} \tag{4}$$

Matice, Q, Q_N a R sú nazývané váhové matice a spolu s horizontom predikcie N sú nazývané parametrami na ladenie prediktívneho regulátora. Nájdenie optimálneho riadenia, ktoré by minimalizovalo kvadratické kritérium predstavuje dynamickú optimalizáciu a má v tomto prípade aj analytické riešenie: [16]

$$\begin{aligned} x_{t+k} &= A^k x_0 + \sum_{i=0}^{k-1} A^i B u_{k-1-i} \\ u_{t,N} &= [u_t^T, \ldots, u_{t+N-1}^T] \end{aligned} \tag{5}$$

Vyjadrenie predikovaného stavu je potom:

$$[x_{t+1}^T, \ldots, x_{t+N}^T]^T = Vx_0 + Tu_{t,N} \tag{6}$$

Táto rovnica je jedna z najdôležitejších pri pochopení fungovania on-line prediktívneho algoritmu. Dôležitý krok pri hľadaní optimálneho riadenia je zo vzťahu 3 „odstrániť“ sumu. Finálny vzťah vyzerá:

$$J\left(x\left(t\right), U_t\right) = \frac{1}{2}{u_{t,N}}^T H u_{t,N} + x^T\left(t\right) F U_t + \frac{1}{2}x^T\left(t\right) \mathrm{Yx}\left(t\right), \tag{7}$$

Matica $H \in R^{(m\bullet N \times m\bullet N)}$, $F \in R^{(n\times m\bullet N)}$, $Y \in R^{(n\times n)}$. Ak sa symbol $\bigotimes$ označí ako kroneckerovo násobenie matíc a $I_j \in R^{(j\times j)}$, jednotková matica s príslušnou dimenziou, tak platí:

$$\begin{aligned} H = T^T\tilde{Q}T + I_N \bigotimes R, \quad F = V^T\tilde{Q}T, \quad Y = V^T\tilde{Q}V \\ \tilde{Q} = \begin{bmatrix} I_{N-1} \bigotimes Q & 0 \\ 0 & Q_N \end{bmatrix} \end{aligned} \tag{8}$$

Optimálnu riadiacu sekvenciu je možné dostať minimalizáciou kvadratickej formy 7:

$$u^*_{t,N}\left(x\left(t\right)\right) = \arg\min_{u_t,N}\left\{J(x\left(t\right), u_{t,N}\ \right\} = -H^{-1}F^T x(t) \tag{9}$$

Optimálna hodnota kritéria je:

$$J^*\left(x\left(t\right)\right) = \min_{u_t,N}\left\{J(x\left(t\right), u_{t,N}\ \right\} = \frac{1}{2}x^T(t)(Y - FH^{-1}F^T)x(t) \tag{10}$$

Na to aby sa zabezpečila spätná väzba, je potrebné v každom kroku použiť iba prvú hodnotu zo sekvencie riadiacich zásahov a potom zmerať stav a na základe tej hodnoty znovu vypočítať optimálnu sekvenciu riadiacich zásahov. Bez merania stavu, by to bola regulácia v otvorenom regulačnom obvode. Meranie stavu a jeho použitie pri výpočte nasledujúceho riadiaceho zásahu v každom kroku zabezpečí reguláciu v uzavretom regulačnom obvode.

1.1.2 On-line MPC s obmedzením

Doteraz sa reguloval systém, v ktorom nebolo žiadne obmedzenie. V realite ich však býva mnoho. Pokiaľ sa pri návrhu regulátora začnú brať do úvahy rovnice 2, bude ich treba zapracovať do výpočtu. Pri návrhu prediktívneho regulátora s obmedzením sa využíva kvadratické programovanie. Úloha kvadratického programovania je úlohou nelineárneho programovania, v ktorej sústava ohraničení je lineárna a účelová funkcia je kvadratická. Všeobecná formulácia kvadratickej úlohy je:

$$\begin{aligned} f\left(x\right) &= \min_x \left\{x^T H x + F^T \mathrm{x}\ \right\} \\ x \in D &= \{\mathrm{x}|\quad Lx \leq m_c,\ x \geq 0\} \end{aligned} \tag{11}$$

Pre návrh regulátora je premenná, ktorej minimum sa hľadá, sekvencia riadiacich zásahov $u^*_{t,N}$. Rovnice pre prediktívny regulátor s obmedzením následne vyzerajú:

$$J\left(x\left(t\right),\ u\left(t\right), \ldots,\ u\left(N-1\right)\right) = \frac{1}{2}\sum_{k=0}^{N-1}\left[x_k^T Q x_k + u_k^T R u_k\right] +\ \frac{1}{2}x_N^T Q_N x_N \tag{12}$$

kde:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \\ x_0 &= x\left(t\right), \\ E_c x_k + G_c u_k &\leq m_c, \quad k = 1 \ldots N \\ Q = Q^T \succcurlyeq 0,\ Q_N = Q_N^T \succcurlyeq 0,\ R = R^T &\succ 0 \end{aligned} \tag{13}$$

Aby bolo možné sústavu rovníc 12 a 13 zapracovať do kvadratického programovania, previesť do maticového tvaru. Prvá časť rovnice ostáva nezmenená, pridá sa k nej druhá časť:

$$J\left(x\left(t\right),U_t\right) = \frac{1}{2}{u_{t,N}}^T H u_{t,N} + x^T\left(t\right) F U_t + \frac{1}{2} x^T\left(t\right) \mathrm{Yx}\left(t\right), \quad G u_{t,N} \leq w + E x(t) \tag{14}$$

Kde matice H, F, Y sú určené rovnako ako vo vzťahu 8 a matice G, E, a vektor w majú tvar:

$$G = \begin{bmatrix} -I_N \\ I_N \\ -(I_N \bigotimes CT) \\ (I_N \bigotimes CT) \\ \vdots \end{bmatrix},\ E = \begin{bmatrix} 0 \\ 0 \\ -(I_N \bigotimes CV) \\ (I_N \bigotimes CV) \\ \vdots \end{bmatrix},\ w = \begin{bmatrix} -(1_N \bigotimes u_{\min}) \\ (1_N \bigotimes u_{\max}) \\ -(1_N \bigotimes y_{\min}) \\ (1_N \bigotimes y_{\max}) \\ \vdots \end{bmatrix} \tag{15}$$

V rovnici 15 sú znázornené základné systémové obmedzenia. Môže existovať ďaleko viac. 1_N predstavuje jednotkový vektor o veľkosti N. Prvé dva riadky matice G, $G_{1,2} \in R^{N\times N}$, E, $E_{1,2} \in R^{N\times n}$ a vektor w, $w_{1,2} \in R^N$ v (19) predstavujú obmedzenia vstupu. Druhé dva riadky matice G, $G_{3,4} \in R^{N\times N}$, E, $E_{3,4} \in R^{N\times n}$ a vektor w, $w_{3,4} \in R^N$ v 15 predstavujú obmedzenia výstupu. Ďalšie obmedzenia, by museli nadobúdať rovnaké rozmery ako predošlé riadky príslušných matíc a vektora.

1.1.3 On-line MPC so sledovaním referenčného signálu

Doteraz bol riešený regulátor, ktorý reguloval hodnotu k „počiatku“ k hodnote 0 alebo nulovému vektoru. Táto kapitola obsahuje návrh MPC regulátora, ktorý bude sledovať po častiach konštantný referenčný signál r(t). Stále sa berie do úvahy systém 1. Najprv je potrebné nahradiť člen $x_k^T Q x_k$ v pôvodnom kritériu 13 odchýlkou

$$(z_k - r_k)^T Q_y (z_k - r_k) \tag{16}$$

kde

$$z\left(t\right) = Z y\left(t\right),\ z\left(t\right) \in R^{p_z},\ p_z \leq p,\ r \leq m \tag{17}$$

Aby to bolo možné, je potrebné poznať predikovanú hodnotu referenčného signálu. Ta môže byť definovaná rôznymi spôsobmi jedna z možností: r(t+1)=r(t).

V ustálenom stave je potrebné, aby platila rovnosť $z_u = r_u$. Takže:

$$\begin{bmatrix} I_n - A & -B \\ \mathrm{ZC} & 0 \end{bmatrix} \begin{bmatrix} x_u \\ u_u \end{bmatrix} = \begin{bmatrix} 0 \\ r_u \end{bmatrix} \tag{18}$$

Ak ma predchádzajúca matica plnú riadkovú hodnosť, existuje riešenie u_u a x_u.. V dôsledku nenulového ustáleného vstupu sa v praxi často používa odchýlka $u_k = u_k - u_{k-1}$. Ak sústava obsahuje integrátor, je lepšie použiť hodnotu u. [16]

Kritérium pre sledovanie referencie má tvar:

$$J\left(x\left(t\right),\ u\left(t\right),\ldots,\ u\left(N-1\right)\right) = \frac{1}{2}\sum_{k=0}^{N-1}\left[e_k^T Q_e e_k + u_k^T R u_k\right] \tag{19}$$

kde

$$e_k = y_k - r_k \tag{20}$$

je regulačná odchýlka a

$$\Delta u_k = u_k - u_{k-1} \tag{21}$$

je optimalizovaná premenná. Systém rozšírený o históriu riadiaceho zásahu u_k a generátor referencie r_k je formulovaný:

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ u_k \\ r_{k+1} \end{bmatrix} = \begin{bmatrix} A & B & 0 \\ 0 & I_m & 0 \\ 0 & 0 & I_{n_y} \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \\ r_k \end{bmatrix} + \begin{bmatrix} B \\ I_m \\ 0 \end{bmatrix} u_k = \ A_m \hat{x_k} + B_m u_k \\ e_k = \begin{bmatrix} C & 0 & -I_{n_y} \end{bmatrix} \hat{x_k} = C_m \hat{x_k} \end{aligned} \tag{22}$$

Po upravení algoritmu na sledovanie po častiach spojitého referenčného signálu už MPC regulátor neriadi výstupnú veličinu k 0 hodnote alebo nulovému vektoru, ale ku aktuálnej hodnote prípadne vektoru referenčného signálu v kroku k, ..., k + N - 1, kde N predstavuje horizont predikcie. Preto referenčný signál vstupujúci do výpočtu je je

buď vektor pre jednorozmerné systémy (SISO) alebo matica pre viacrozmerné systémy (MIMO). V dizertačnej práci je spomenutá možnosť offline riešenia MPC. Táto však v implementácii nie je využitá a preto nie je spomenutá ani v autoreferáte. V online riešení do každého kroku vstupuje matematický model systému a teda je možné matematický model dynamicky adaptovať, aby čo najviac zodpovedal reálnemu systému. Pri voľbe spôsobu implementácie praktickej časti pre výučbu aj neskôr v IoT prostredí, tento fakt zavážil a preto sa zvolila online metóda implementácie.

2 Softvérové architektúry pre pokročilé metódy riadenia

V súčasnej dobe dochádza k užšej spätosti IT a OT, preto je nevyhnutné pri návrhu regulátora mať aj znalosti o prostredí, v ktorom regulátor má spoľahlivo fungovať. Práca sa preto v druhej časti zameriava na oblasť softvérovej architektúry. A hlavne tých oblastí, ktoré vedú do problematiky internetu vecí a zároveň súvisia s aplikáciou pokročilých metód riadenia v praxi.

2.1 Základne typy architektúr

Jedno z kľúčových rozhodnutí pri návrhu systému je voľba typu architektúry. Tie najpoužívanejšie sú vymenované nižšie podľa článku [3] napísanom na základe knihy[17], ktorú napísal Mark Richards - softvér architekt s 30 ročnou skúsenosťou. V článku sa tiež uvádza, že v jednom systéme môže byť použitých viacero typov, čo je dôležitá informácia pri návrhu.

- **N-vrstvová architektúra.**
- **Architektúra riadená udalosťami.**
- **„Microkernel“ architektúra.**
- **„Microservice“ architektúra.**
- **„Space-based“ architektúra.**

2.2 Základné typy aplikácií

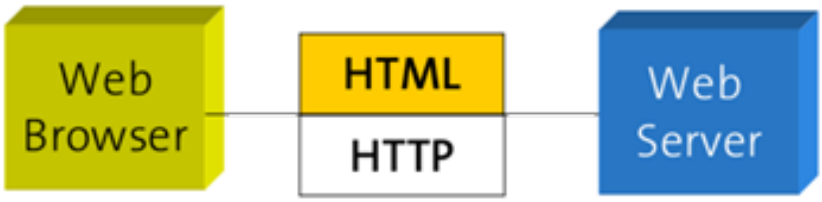
Ďalšie z kľúčových rozhodnutí je voľba typu aplikácie. Na výber podľa článku [2] sú:

- **Mobilná aplikácia.**
- **Tučná klientská aplikácia.**
- **Tučná internetová aplikácia.**
- **Webová aplikácia obrázok.**
- **Servisná aplikácia obrázok.**

2.2.1 Servisná aplikácia

Servisným aplikáciam je v práci venovaná samostatná kapitola, pretože je to najvhodnejší spôsob akým moderné metódy riadenia, či už jeden alebo viac regulátorov integrovať do softvérového systému. Pomocou servisnej aplikácie je možné implementovať mikroservice architektúru, ktorá má pomenovanie aj SOA (service oriented architecture). Existuje viacero spôsobov ako SOA implementovať.

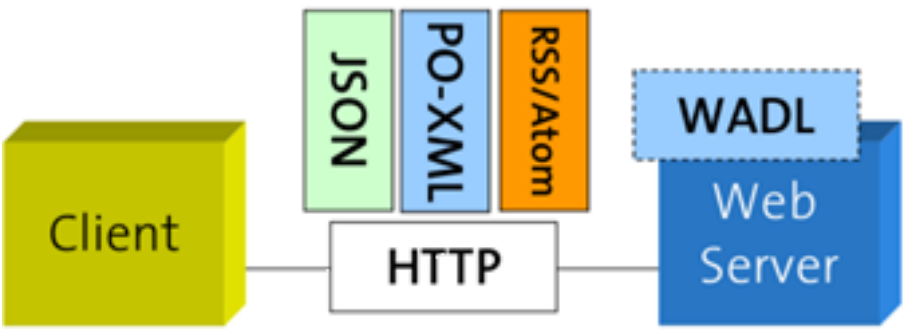
- Najznámejší spôsob implementácie SOA, ktorí dnes používa väčšina užívateľov informačných technológií je implementácia protokolu WWW - **World Wide Web** v skratke často označovaný len WEB. Obrázok 2 znázorňuje princíp fungovania.



Obrázok 2: WWW [4]

Prehliadač (web browser) je v pozícii klienta, konzumera služby, ktorú poskytuje Web Server. Web Server je označenie takého počítača, na ktorom beží servisná aplikácia, ktorá vie na základe definovaných pravidiel poskytnúť požadovaný obsah.

- Iná implementácia SOA architektúry uvádzaná v tejto práci je **architektonický štýl REST**. Možností samozrejme existuje viacero, ale tento spôsob je najviac využívaný vo svete a aj v rozvíjajúcej sa oblasti IoT. REST je skratka od REpresentational State Transfer. Ako je možné vidieť na obrázku 3 tu vystupuje generický klient a generický server.



Obrázok 3: REST [4]

2.3 Internet vecí - IoT

Organizácia IEEE vydala celý dokument s názvom „Smerom k definícii Internetu vecí“. Cieľom dokumentu je podať plnohodnotnú definíciu IoT v rozmedzí od malých lokálnych systémov, obmedzených na konkrétnu lokalitu, až po globálny systém, ktorý je distribuovaný a poskladaný z komplexných systémov. V dokumente je možnosť nájsť prehľad základných požiadaviek na architektúru IoT [10]. Podľa spoločnosti Gartner, ktorá je svetový líder v oblasti výskumu informačných technológií je IoT sieť fyzických objektov, ktoré majú vstavanú technológiu na komunikovanie a snímanie alebo interakciu s ich vnútornými stavmi alebo vonkajším prostredím [5]. Ešte definícia podľa stránky Techopedia: IoT je koncept, ktorý popisuje budúcnosť, kde každodenné fyzické objekty budú pripojené do internetu a budú sa vedieť sami identifikovať iným zariadeniam. Pojem je úzko spojený s RFID technológiou ako spôsobom komunikácie, aj keď to môže zahŕňať iné technológie na snímanie, bezdrôtové technológie alebo QR kódy.

IoT je významné, pretože ak objekt sa vie sám digitálne reprezentovať stáva sa z neho niečo viac ako objekt samotný. Objekt sa už nevzťahuje len na nás, ale je spojený s okolitými objektami a dátami v databázach. Keď veľa objektov spolupracuje, je možné to nazvať ako inteligencia okolitého prostredia „ambient intelligence“ [13]. Viacero ďalších technologických spoločností majú ich definíciu. Cieľom tejto práce nie je vytvoriť ďalšiu definíciu, ale identifikovať spoločné črty väčšiny definícií, ale hlavne načrtnúť možnosti zavedenia moderných metód riadenia do praxe.

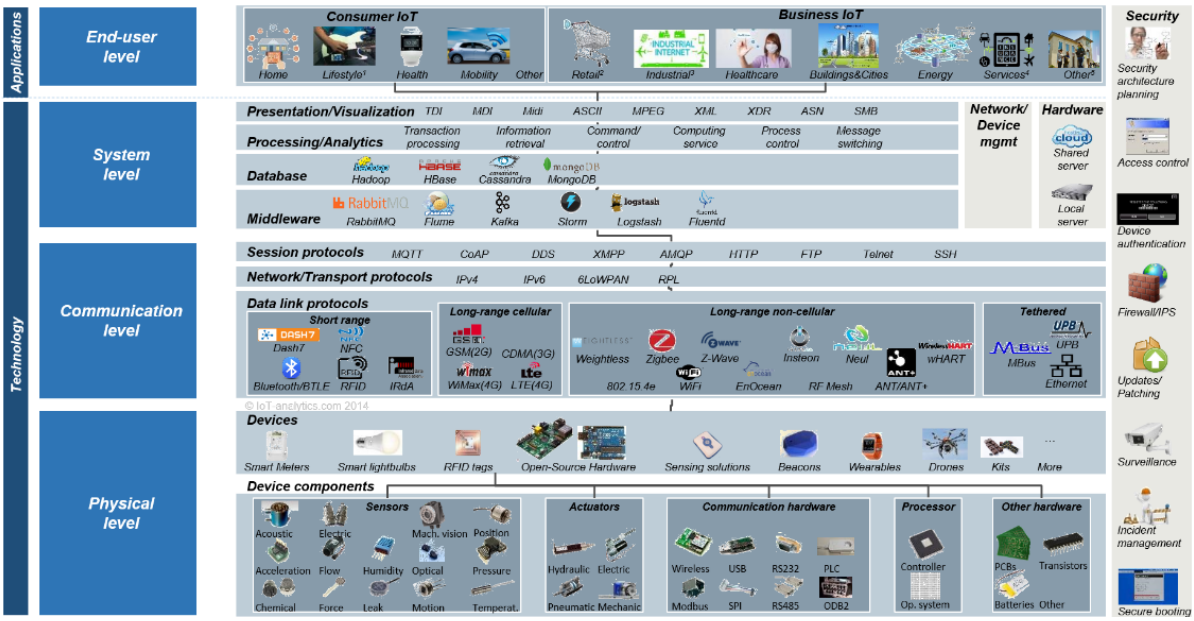
- Prvá základná črta definícii je, že v nej vystupujú objekty, ktoré môžu **snímať a ovplyvňovať okolie**. Na dosiahnutie tejto črty je potrebné, aby objekty mali senzory a akčné členy, čo je jeden z hlavných záujmov odboru automatizácia, pre ktorý to nie je nič nové. Dostupnosť a klesajúca cena snímačov a akčných členov umožňuje ich umiestňovanie aj na objekty mimo priemyslu, čo otvára priestor pre Internet vecí.

- Druhá základná črta definícii je **prepojenie**. Opäť v automatizácii a priemysle to nie je nič nové, veď koľko priemyselných štandardov rieši prepojenie senzorov a akčných členov na výrobných linkách po celom svete. Hnacou silou rozvíjajúceho sa Internetu vecí sú nové možnosti drôtového a bezdrôtového pripojenia dostupné pre koncových používateľov, klesajúce náklady na prevádzku sietí a klesajúca cena zariadení nazývaných - edge device, gateway alebo agregátor, ktoré umožňujú zber a posielanie dát do dátových centier.

- Ďalšia spoločná črta je vyústením dvoch predošlých a to **vytváranie inteligent-**

ného prostredia. Keď bežné objekty vedia snímať rôzne fyzikálne veličiny, zosnímané hodnoty prostredníctvom pripojenia môžu poslať prostredníctvom agregátora do dátového centra, v dátovom centre sú hodnoty zoskupené a pripravené na urobenie analýz a štatistík, na základe ktorých, ak je možné robiť rozhodnutia v reálnom čase alebo kvázi reálnom čase, tak je spätne možnosť ovplyvniť akčný člen, čo vo veľkom môže znamenať vytvorenie inteligentného prostredia. Zabezpečenie tejto črty je predmetom oblasti ukladania a spracovania dát. V tomto bode sa IoT často spája s pojmom Big Data. Definícia tohto pojmu má viacero verzií podobne ako definícia IoT. Zvolila sa aspoň jedna od spoločnosti Gartner pre ilustráciu. Big data sú informácie veľkého objemu, veľkej rýchlosti a/alebo veľkej variability, ktoré si vyžadujú rentabilné, inovatívne formy spracovania informácii, ktoré umožňujú väčší náhľad, napomáhať rozhodovaniu a automatizácii procesov [12]. V implementačnej časti je jedna časť venovaná problematike Big data.

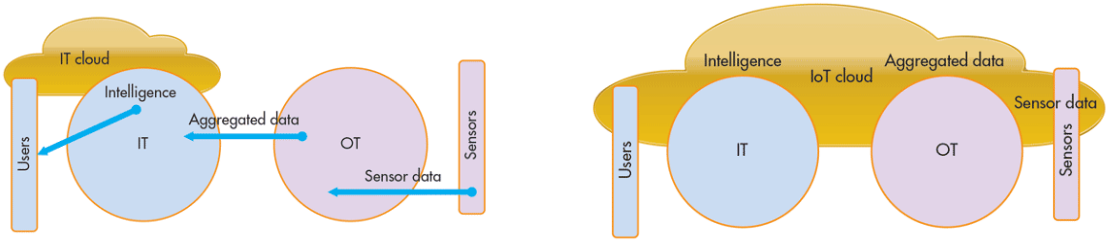
Množina technológii, určite nie všetkých, ktoré môže IoT zhŕňať je ďalej na obrázku 4. Na obrázku je možné vidieť súčasti, ktoré sme uviedli ako spoločné črty IoT definícií.



Obrázok 4: IoT súhrn techológií [14]

Pri pohľade na tento obrázok si je dôležité uvedomiť, koľko rôznych typov softvérov, jazykov a architektúr tu prichádza do interakcie. Začínajúc zo spodu. Senzory a akčné členy podľa zložitosti majú buď len nahratý jednoúčelový firmware, ktorý umožňuje len meranie prípadne vykonávanie akčných zásahov a posielanie dát. Alebo majú určitý druh operačného systému, ktorý umožňuje na tejto úrovni do merania a akčných zásahov robiť

úpravy. Ak sa spomenie posielanie dát, tak opäť je tam softvér zodpovedný za odosielanie a prijímanie dát, pričom ako je možné vidieť na obrázku, tak je viacero úrovni abstrakcie odosielania údajov. Na systémovej úrovni sú to rôzne databázové systémy, systémy na spracovanie správ (message queue), analytické nástroje a vizualizačné rozhrania. Nakoniec až na najvyššej úrovni sú aplikácie, ktoré spojením rôznych pojmov z nižšej vrstvy dokopy majú pridanú hodnotu pre koncového používateľa. Na prvý pohľad by sa dalo povedať, že IoT sa nijako nelíši od existujúcich priemyselných inštalácií, preto je ďalej znázornené, v čom sa IoT líši od tradičnej architektúry priemyselných systémov na obrázkoch 5a a 5b.



(a) Tradičná architektúra [6]

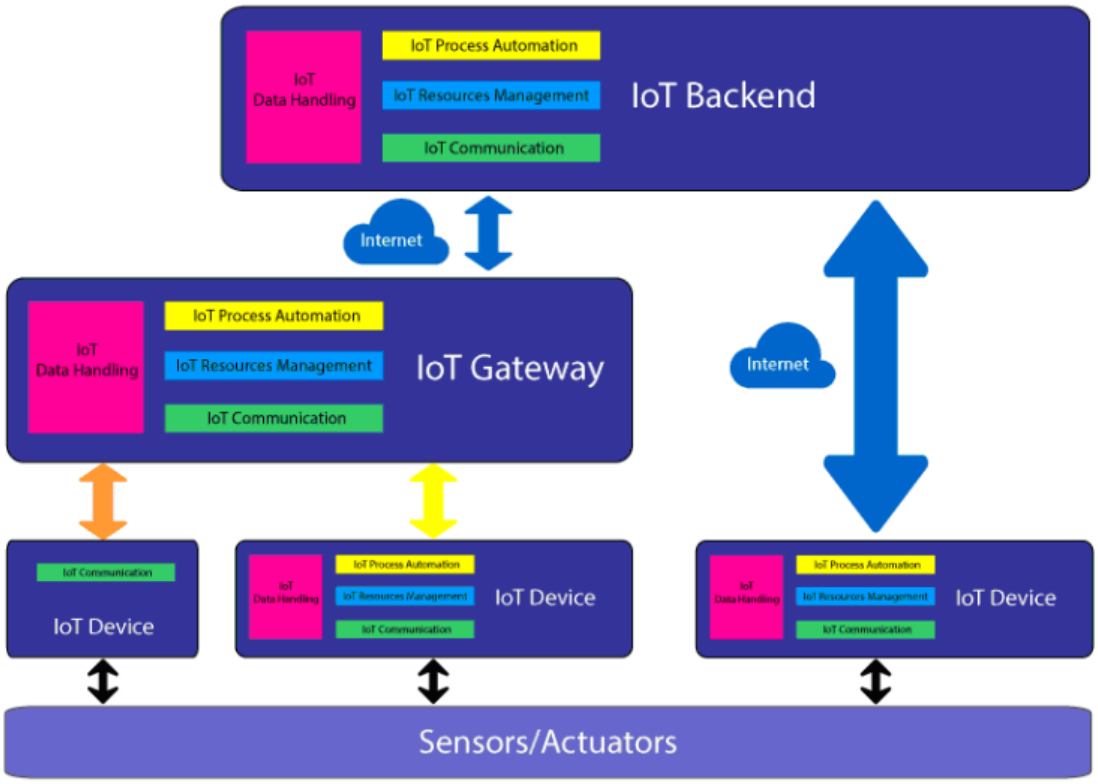
(b) IoT architektúra [6]

Obrázok 5

Tradičná priemyselná architektúra na posunutie dát do platformy informačných technológií (Information Technology - IT), ktorá vie robiť analýzu dát, poskytovať ich používateľovi aj vystavovať do cloud prostredia, používa vstavaný softvér na platforme obslužných technológií (Operational Technology - OT). IoT architektúra práve posúva väčšiu časť inteligencie systému z IT strany do OT strany. Toto umožňujú mikroprocesory a vstavané platformy s jednoduchým prístupom do cloud prostredia a rovnako s jednoduchým prístupom ku autorizovaným zariadeniam a používateľom. [6]. Táto definícia potvrdzuje to, čo sme už vyslovili, že ide o užšie prepojenie IT a OT.

Základné ciele IoT architektúr je teda zosnímané dáta čo najskôr, najbezpečnejšie a najspoľahlivejšie poslať ďalej na miesto, kde môžu prejsť analýzou, či už je to cloud alebo dosť často to býva aj samotný agregátor, ktorý má dostatočný výkon na určitý druh analýz a na základe analýzy ovplyvniť akčné členy, aby sa optimalizoval proces, náklady, zdroje atď. Aktuálny trend vo svete je, že veľké technologické firmy sa snažia spraviť univerzálnu IoT platformu a podchytiť si tak čo najviac zákazníkov. Základné komponenty IoT systémov sú naznačené na obrázku 6

V tomto bode máme zadefinované všetky pojmy a princípy, ktoré sú potrebné pri návrhu reálnej IoT aplikácie s modernými metódami riadenia a teda spojenia oblasti automatizácie a softvérovéj architektúry. Pre návrh a implementáciu bol zvolený IoT



Obrázok 6: IoT súčasti - všeobecne[11]

systém inteligentnej domácnosti, ktorého špecifiká sú v práci priebežne menované. Týmto končí časť definícii a nasleduje popis technických krokov návrhu a vývoja IoT systému a aplikovania MPC do tohto systému.

3 Návrh a implementácia IoT systému s modernými metódami riadenia

Táto časť popisuje postup návrhu a implementácie aplikácie moderných metód riadenia do IoT prostredia. Pod modernými metódami riadenia sa rozumie zavedenie nového pojmu controller as a service (CaaS) - regulátor ako služba a pod IoT prostredím sa tu rozumie inteligentná domácnosť. Myšlienka CaaS je výsledkom sledovania IoT trendu a návrh je inšpirovaný IoT architektúrami v dvoch bodoch. Prvý bod je využitie potenciálu výpočtového výkonu serverov. Vstavané zariadenia sa ukázali ako nedostatočné pre vykonávanie online MPC výpočtu. Druhý bod je užšia spolupráca serveru a akčného člena. Návrh konkrétnej realizácie je založený na SOA architektúre s REST princípmi. Viac je popísane v časti 3.2. Teraz nasleduje popis IoT prostredia s odôvodnením voľby kľúčových elementov s ohľadom na finálnu architektúru.

3.1 Popis a voľba experimentálneho IoT prostredia

Cieľová architektúra vychádza z obrázkov 4 a 6 a jej konkrétna implementácia je znázornená na obrázku 7 Fyzická vrstva sa skladá z práci vymenovaných senzorov a akčných členov, z ktorých väčšina je pripojená do systému pomocou protokolu Z-Wave. Hlavný uzol siete Z-Wave zohráva zariadenie VeraLite. Pomocou tohto zariadenia sú jednotlivé fyzické zariadenia pridávané do siete. Dôležitou vlastnosťou je, že okrem fyzických zariadení vie toto zariadenie pridávať aj **zariadenia softvérové**, čo súvisí s možnosťou programovateľnosti. Softvérovým zariadeniam sa bude venovať neskôr. Zariadenie VeraLite v našom IoT systéme, ako je možné vidieť na obrázku 7 zohráva úlohu brány.

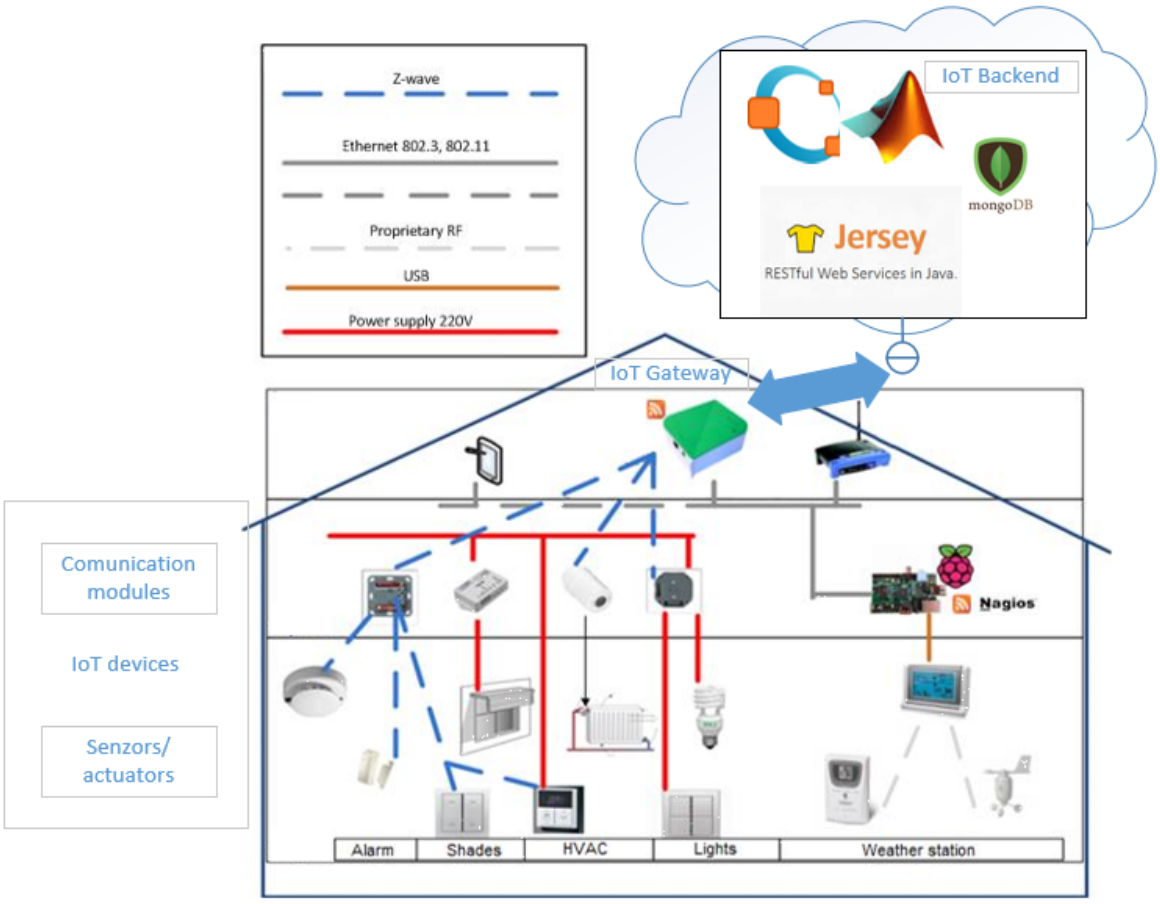
Ďalšie zariadenie pripojené v IoT systéme je meteostanica a to prostredníctvom USB káblu na zariadenie Raspberry Pi model A.

Ďalší spôsob pripojenia do IoT systému tvorí fotorezistor s PCF8591 čipom cez GPIO rozhranie pripojené do zariadenia Raspberry pi 2 Model B+. Fyzikálna veličina odporu je takto skonvertovaná do digitálnej podoby a takto prístupná takému zariadeniu, ktoré vie vytvoriť HTTP požiadavku a prečítať jej odpoveď.

3.1.1 Detaily IoT brány

Táto kapitola sa venuje

- popisu VeraLite zariadenia.
- Spôsobu komunikácie zariadení Raspberry Pi s VeraLite (IoT bránou).
- Spôsobu programovania zariadenia VeraLite a vytvárania softvérových zariadení v



Obrázok 7: Experimentálne prostredie

ňom.

Framework VeraLite zariadenia je postavený na protokole UPnP - priemyselný štandard na ovládanie zariadení, a skriptovacom jazyku Lua. Framework sa volá *Luup* [9].
Na základe generickosti UPnP protokolu sa v práci definovalo šesť nových typov zariadení, štyri pomocné a dve zariadenia nevyhnutné pri implementácii CaaS. V autoreferáte spomenieme len tie najdôležitejšie.

- Na komunikáciu VeraLite s Raspberry PI model A a čítanie hodnôt z meteostanice sa vo VeraLite vytvorilo všeobecné softvérové zariadenie typu RSS Reader s ID zariadenia *urn:demo-micasaverde-com:device:RssReader:1.*

- Na komunikáciu VeraLite s Raspberry PI model B a čítanie hodnôt z fotorezistora sme vytvorili všeobecné zariadenie Rest Handler, všeobecne na čítanie akejkoľvek rest služby s ID *urn:demo-micasaverde-com:device:RestHandler:1.*

Rest Handler softvérové zariadeniemá len jednu službu s menom HandleRest, ktorej ID je *(urn:demo-micasaverde-com:serviceId:HandleRest1)* - spracuj REST službu, čo znamená, že program pravidelne oslovuje vystavený REST interface, ktorý poskytne dáta, v našom prípade informáciu z fotorezistora. Ohľadne implementácie, je potrebné použiť modul na vytvorenie HTTP klienta a v tomto prípade je formát správ v JSON notácii, takže miesto XML parsera je tu použitá knižnica [7] na konverziu JSON objektov. Základne operácie definovane v službe sú nasledovné.

- setRestUrl - operácia na zadefinovanie URL, na ktorú sa ma HTTP klient dopytovať.

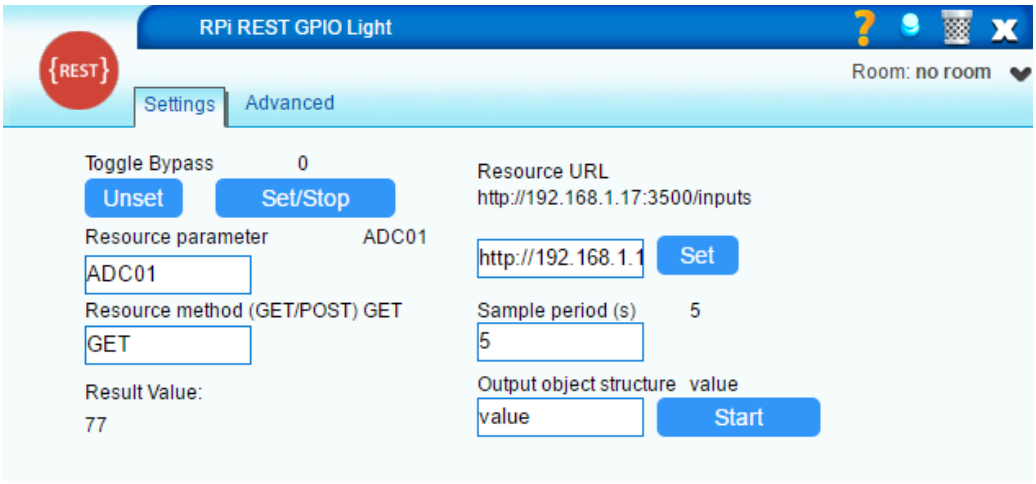
- handleRest - operácia na samotné spustenie programu na pravidelné načítavanie hodnôt. Je potrebné definovať REST zdroj, HTTP metódu, ktorá sa ma použiť, perióda s akou ma byť REST zdroj volaný a objekt, z ktorého sa ma čítať výstupná hodnota.

- (un)setBypass - operácie na zastavenie načítavania hodnôt.

Rozhranie na zadanie hodnôt a spustenie je zobrazené na obrázku 8.

3.2 CaaS - Regulátor ako služba

Táto kapitola sa zameriava na vysvetlenie, motiváciu a popis implementácie myšlienky regulátor ako služba (Controller as a Service). Ako už bolo naznačené ide o vystavovanie služby regulovania procesu do cloud prostredia. Kedy na servery je uložený matematický



Obrázok 8: Rozhranie na zadávanie parametrov REST klienta.

model regulátora a na požiadavku vie server vrátiť akčný zásah pre daný regulátor v danom stave. Od klienta sa očakáva, že vie obsluhovať HTTP komunikáciu a základnú prácu s reťazcami, žiadne výpočty nevykonáva, len sa pravidelne spýta na akčný zásah. Pre regulátory s nízkymi nárokmi na výpočtovú kapacitu tento koncept nepridáva hodnotu výpočtového výkonu serveru, iba možnosť uloženia veľa inštancií, takže sa môže využiť iba pamäťová kapacita servera. Tento princíp dáva zmysel hlavne pre regulátory s veľkými požiadavkami na výpočtový výkon. Medzi takéto patrí práve online metóda MPC algoritmu, ktorý si vyžaduje, ako je v prvých kapitolách naznačené, prácu s veľkými maticami ich násobenie a inverzia. Až niekoľko neúspešných pokusov v postupe práce priviedli k tomuto konceptu. CaaS je teda proces riadený pomocou regulátora, uloženého a výpočty vykonávajúceho na vzdialenom servery. Implementovaná aplikácia služby regulátora je pripravená na to, aby sa matematický model menil aj v každom kroku.

3.2.1 Implementácia služby regulátora

Súčasti systému sú

- Matlab/Octave ako jadro výpočtov
- MongoDB ako úložisko regulátorov
- Java Jersey ako nástroj na vystavenie REST rozhraní.
  - POST c-a-a-s/rest/mpccontrollers
    Toto rozhranie slúži na zadanie regulátora do aplikácie
  - GET c-a-a-s/rest/mpccontrollers/{id}
    Ak už klient pozná ID regulátora, tak pomocou tohto rozhrania je možné si

pozrieť detail uloženého objektu.

- POST c-a-a-s/rest/mpccontrollers/{id}/steps Pomocou tohto rozhrania vie klient požiadať o ďalší riadiaci zásah.
- GET c-a-a-s/rest/mpccontrollers/{id}/steps/{id} Ak je potrebné pozrieť kroky v minulosti, tak je to možné pomocou tohto rozhrania.

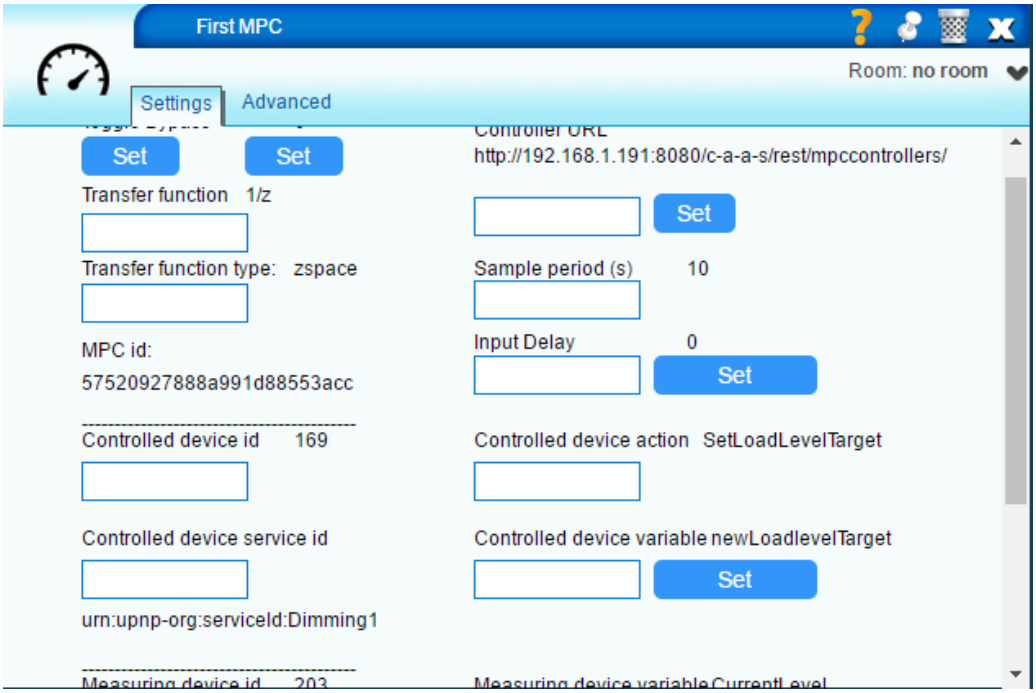
Funkcionalita, ktorú musí rozhranie na servery vykonať pri vytváraní inštancie regulátora a rovnako aj pri výpočte akčného zásahu pozostáva z týchto krokov:

1. Inicializácia nenastavených default premenných.
2. Uloženie do Mongo databázy, ktorá vygeneruje ID záznamu, ktoré figuruje ako ID regulátora.
3. Pred výpočtom sa uloží JSON do súboru.
4. Nasleduje volanie výpočtového systému Matlab alebo Octave, ktorý načíta uložený súbor, pomocou pluginu Jsonlab [8], ktorý vie z JSON objektu spraviť Matlab/Octave dátové typy a naopak.
5. Nasleduje krok v Matlab/Octave nástroji, kde prebehne výpočet podľa kapitoly **??**, či už inicializácia systému alebo výpočet akčného zásahu. Na konci výpočtu sa dátové typy uložia do súboru pomocou Jsonlab vo forme JSON objektu.
6. Späť v Java aplikácii sa súbor načíta a spraví z neho inštancia príslušnej triedy.
7. Upravený objekt sa uloží do databázy.
8. Ten istý objekt sa pošle používateľovi na výstup.

Pri implementácii boli komplikácie s načítavaním objektu zo súboru, ktorý ukladal Jsonlab. Chýbala typová striktnosť, takže sa stalo, že z premennej typu vektor v Java aplikácii sa v Matlab/Octave stalo obyčajné číslo, ak vektor pozostával z jedného prvku. Následne pri ukladaní výsledkov výpočtov do súboru sa vynechala vektorová notácia a keď typovo striktný Java interface vytváral objekt, ktorý poskytne klientovi, tak vektorová notácia bola očakávaná. Na vyriešenie týchto komplikácii, boli potrebné úpravy do jadra Java objektov poskytujúcich transformáciu textu na JSON objekty.

3.2.2 Integrácia MPC pomocou softvérového zariadenia

Pre službu regulátora popísanú v predošlej časti a vystavenú v časti IoT Backend - obrázok 7 je potrebné vytvoriť klienta. V Luup frameworku je definované softvérové zariadenie MpcClient s ID *urn:demo-micasaverde-com:device:MpcClient:1* so službou, ktorá ma ID definované ako *urn:demo-micasaverde-com:serviceId:MpcControl1*. Rozhranie na zadávanie parametrov a spustenie riadenia je na obrázku 9. Nachádza sa tam



Obrázok 9: Rozhranie na zadávanie parametrov MPC regulátora.

- vstup pre prenosovú funkciu.
- Vstup pre zadanie URL služby, kde je aplikácia CaaS vystavená.
- Zobrazenie ID regulátora.
- Tlačítko na vypnutie/povolenie riadenia.
- Vstup pre zadanie periódy vzorkovania, ktorá zároveň identifikuje ako často sa bude volať služba na načítanie akčného zásahu.
- Vstupy na definovanie zariadenia, ktoré bude regulátorom ovládané.
  - ID inštancie zariadenia.
  - ID služby.

  - ID operácie.
  - Meno vstupnej premennej do operácie.

- Vstupy na definovanie zariadenia, ktoré bude slúžiť na meranie výstupnej veličiny
  - ID inštancie zariadenia.
  - ID služby.
  - Meno meranej stavovej premennej.

Princíp fungovania je taký, že na začiatku sa inicializuje regulátor zadaním prenosovej funkcie, snímacieho zariadenia, vykonávacieho zariadenia a periódy vzorkovania. Regulátor sa uloží do IoT Backend a na klientovi sa vždy na konci výpočtu načasuje spustenie nového výpočtu, ktorý sa spustí v čase periódy vzorkovania. Takto sa pravidelne zavolá IoT Backend, aby vrátil vypočítaný akčný zásah. Na vstupe je vždy nameraná hodnota zo snímacieho zariadenia, aby sa zabezpečila spätná väzba. Akčný zásah na výstupe je aplikovaný do akčného členu.

Pri identifikácii systému, ktorá je popísaná neskôr, je identifikovaný nelineárny systém. Tento fakt spôsobuje, že na jeden proces sú potrebné viaceré regulátory pre každý pracovný bod lineárnej časti jeden. Na to aby bolo možné prepínať, ktorý regulátor ma riadiť danú časť priebehu je v práci vytvorené posledné šieste softvérové zariadenie s názvom MpcMaster. UPnP ID zariadenia je *urn:demo-micasaverde-com:device:MpcMaster:1* a má jednu službu s UPnP ID *urn:demo-micasaverde-com:serviceId:**MpcMultipleControl1***. Operácie, definované v tejto službe sú:

- setSetpoint - slúži na stavenie žiadanej hodnoty
- setMpcs - slúži na zadefinovanie ID softvérových zariadení MPC regulátorov (MpcClient), ktoré sú zodpovedné za riadenie jednotlivých priebehov, ďalej sa tu zadefinuje hranica, podľa ktorej sa rozhodne, ktorý regulátor bude vypočítavať akčný zásah a nakoniec slúži na samotné spustenie riadenia.
- (un)setBypass - slúži na zastavenie riadenia.

Riadenie pozostáva z jednoduchých krokov.

1. Načítanie žiadanej hodnoty.
2. Rozhodnutie, ktorý MpcClient má byť spustený,

3. Spustenie správneho MpcClienta, ktorý následne zavolá výpočet akčného zásahu a aj zabezpečí vykonanie akčného zásahu.

4. Vypnutie MpcClienta, aby nezačal samostatne riadiť proces.

5. Nastavenie časovača, podľa periódy vzorkovania, kedy sa má riadiaca slučka znovu spustiť.

3.2.3 Experiment riadenia prostredníctvom MPC

V tejto kapitole je vykonané overenie riadenia pomocou prediktívnej metódy v IoT prostredí. Samozrejme IoT prostredie je systém popísaný v kapitole 3.1. Na experiment overenia riadenia sú využité tieto súčasti.

- Z fyzickej vrstvy sú využité zariadenia - stmievač FIBARO FGD 211 a fotorezistor pripojený na Raspberry Pi model B.

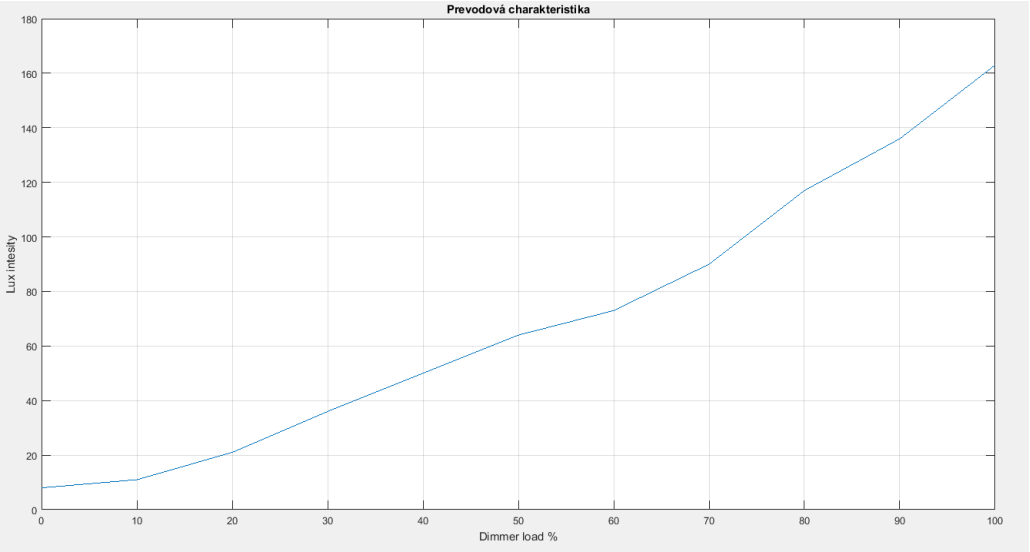
- Na IoT bráne sú použité súčasti - softvérové zariadenia - 1x RestHandler na načítanie intenzity žiarenia, 2x MpcClient na získavanie hodnôt z IoT Backend časti, potrebných na riadenie a 1x MpcMaster na riadenie MpcClientov.

- Nakoniec je využitý IoT Backend popísaný v kapitole 3.2.1 na vykonávanie výpočtov prediktívneho regulátora.

Na overenie riadenia je zvolený proces udržiavania žiadanej intenzity svetla pomocou svetelného zdroja, ktorý je ovplyvňovaný univerzálnym stmievačom FIBARO FGD 211 a pomocou fotorezistora, ktorý sníma intenzitu svetla.

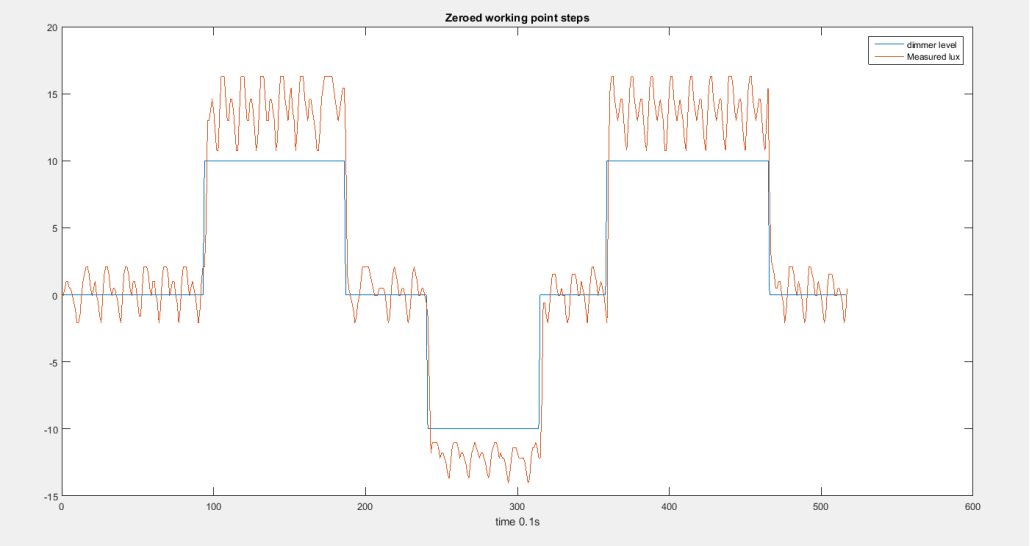
V práci bol potrebný prepočet informácie z fotorezistora, ktorá je v jednotkách odporu (Ohm) na jednotku intenzity svetla (Lux). V testovacej miestnosti sa intenzita pohybovala do 200 luxov (v závislosti od polohy merača), kde od hodnoty 140 luxov meranie začína kolísať v rozmedzí 40 luxov, čo uvidíme neskôr pri identifikácii. Tento fakt neskôr zohľadníme pri vyhodnocovaní kvality riadenia.

Prejdeme k samotnej identifikácii systému. Na začiatku je vykonaná prevodová charakteristika zobrazená v grafe 10. Na osi y je intenzita osvetlenia v luxoch a na osi x je miera zotmenia stmievača v %. Z grafu je možné vidieť, že systém nie je lineárny. V prvej časti grafu od 20% do 60% na osi x je možné vidieť lineárnu oblasť, kde na zmenu o 40% intenzita narástla o necelých 25 luxov na osi y. Potom od 60% do 100% na osi x je druhá lineárna oblasť na zmneu 40% intenzita narástla o viac ako 40 luxov na osi y. Na základe týchto dvoch oblastí sú zvolené dva pracovné body: miera zotmenia 30% a 80% na osi x.



Obrázok 10: Prevodová charakteristika závislosť intenzity osvetlenia od percenta zotmenia

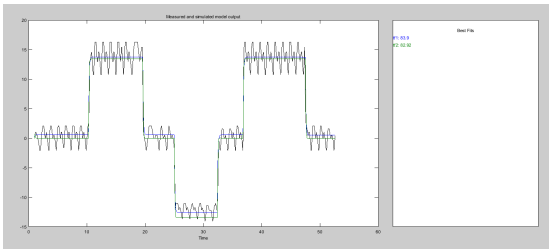
Hranica, ktorá tieto dve oblasti oddeľuje je približne 75 luxov na osi y. Po zvolení pracovných bodov je potrebné spraviť identifikáciu systému pre tieto dva body. Pri identifikácii bola perióda vzorkovania čítania hodnoty intenzity osvetlenia nastavená na 0.1 sekundy pri riadení je to nastavené už len na každých 5 sekúnd. Dáta namerané pre identifikáciu v pracovnom bode 30 sú zobrazené v grafe 11. Dáta v tomto grafe sú už posunuté do bodu 0 vstup v percentách o 30% a výstup systému o strednú hodnotu ustáleného skoku, čo je 48 luxov. Následne namerané a posunuté dáta sa použijú na identifikáciu systému. Tá



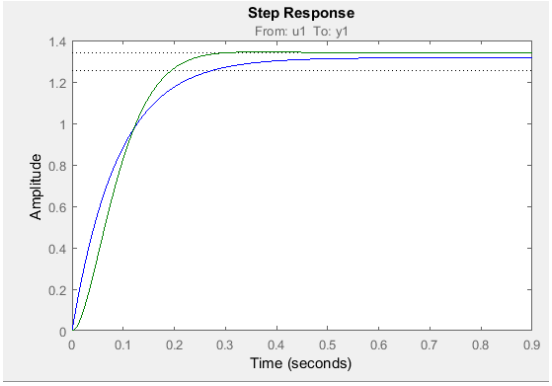
Obrázok 11: Odozva systému v pracovnom bode 30, po posunutí hodnôt do bodu 0

je vykonávaná prostredníctvom funkcionality *ident* nástroja Matlab. Nástroj poskytuje

viacero metód identifikácie, v experimente je použitá identifikácia prostredníctvom modelov prenosových funkcií, kde je zvolená identifikácia na systém 2. rádu. a pre porovnanie aj systém 3. rádu. Výstupné grafy identifikácie sú znázornené na obrázku 12a a odozvy identifikovaných funkcií sú na obrázku 12b.



(a) Odozva na vstup pre bod 30.



(b) Prenosové funkcie pre bod 30.

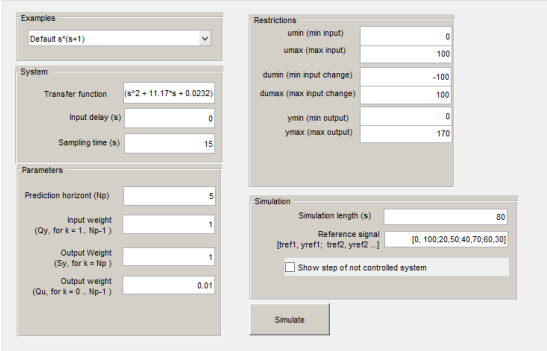
Obrázok 12

Na obrázkoch sú znázornené čiernou čiarou nameraný výstup, modrou farbou prechodová funkcia 2. rádu a zelenou farbou prechodová funkcia 3. rádu. Keďže funkcia 2. rádu presnejšie opisuje existujúci systém táto prechodová funkcia je vybratá na použitie do riadenia pomocou MPC, ktoré potrebuje matematický model. Táto prenosová funkcia má tvar:

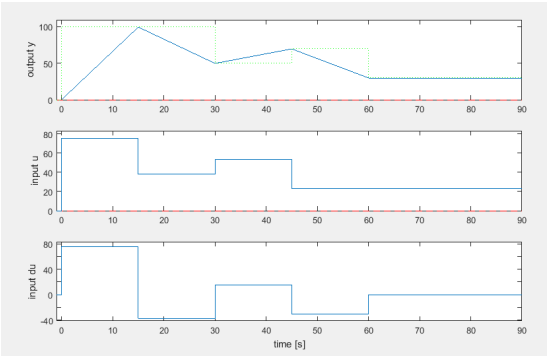
$$G_1 = \frac{14.71 * s + 0.0291}{s^2 + 11.17 * s + 0.0232} \tag{23}$$

Pred aplikovaním funkcie do procesu je vyskúšaná v simulačnom nástroji opisovanom na konci prvej časti práce - kapitola **??**. Parametre a výstup simulácie sú na obrázkoch 13a a 13b. Dôležité informácie na týchto obrázkoch sú voľba periódy vzorkovania až na 15 sekúnd. Táto hodnota je zvolená kvôli tomu, aby sme týmto jedným softvérovýhm zariadením a riadiacim procesom nevyťažili zdroje VeraLite brány natoľko, že by ostatné funkcie fungovali bez oneskorenia. Ďalej sú tu nastavené obmedzenia procesu tak, aby korešpondovali s reálnymi možnosťami zariadení. Obmedzenie na vstup je od 0 po 100 v našom prípade % a zmena môže byť či už z 0 na 100 alebo zo 100 na 0, takže obmedzenia na zmenu vstupu sú na hodnotách -100 a 100. Nakoniec obmedzenia na výstup sú od 0 po 170 luxov. Váhy prediktívneho regulátora nie je potrebné meniť, pretože výstupná veličina sleduje žiadanú hodnotu s požadovanou kvalitou riadenia.

Keď sme predchádzajúcou prenosovou funkciou dali riadiť systém v celom rozsahu, tak od hodnoty 100 luxov už dochádzalo k trvalým regulačným odchýlkam. Preto je



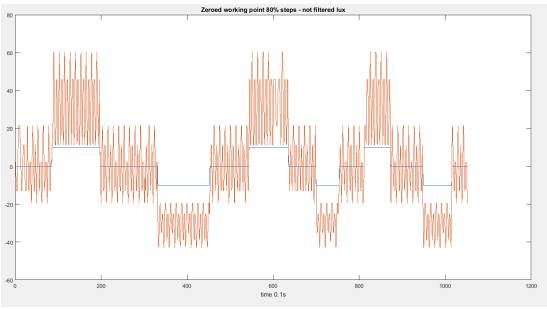
(a) Parametre simulácie pre prvú prenosovú funkciu



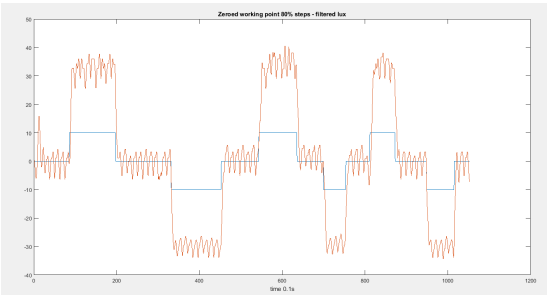
(b) Výstup simulácie pre prvú prenosovú funkciu

Obrázok 13

nevyhnutné robiť identifikáciu pre druhý pracovný bod 80%. Rovnaký postup ako pre bod 30 sa namerali údaje so skokmi okolo pracovného bodu. Namerané údaje je možné vidieť na obrázkoch 14a a 14b, opäť sú namerané dáta posunuté do bodu 0. V tomto prípade bol signál natoľko zašumený, že sme aplikovali filter pomocou pohyblivého priemeru (moving average) po 7 vzorkách takže za čas 0.7 sekundy. Na obrázkoch je vidieť rozdiel. Pre účely identifikácie bol zvolený odfiltrovaný signál, pretože identifikačný nástroj pre nefiltrovaný signál nedával vhodné modely.



(a) nefiltrovaná

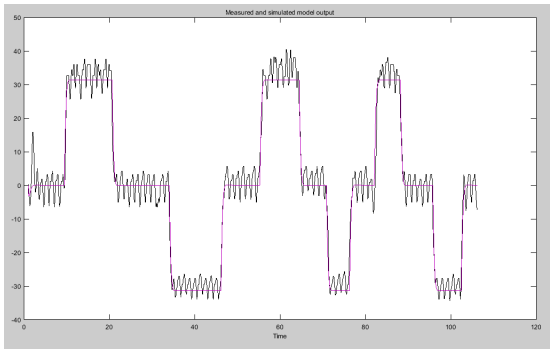


(b) filtrovaná

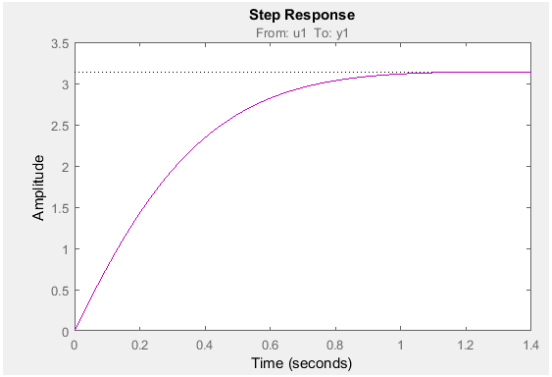
Obrázok 14: Závislosť intenzity osvetlenia od percenta zotmenia v pracovnom bode 80%.

Po tom ako je signál odfiltrovaný ident funkcionalita vráti kvalitnejšie modely. Percento zhody je síce nižšie ako pri menej zašumenom signály v predchádzajúcom pracovnom bode, ale nemá to vplyv na kvalitu regulácie. Odozvy identifikovaného systému sú znázornené na obrázku 15a, prechodová funkcia je na obrázku 15b. Matematický tvar prenosovej funkcie v pracovnom bode 80 je:

$$G_2 = \frac{8.074 * s + 57.6}{s^2 + 7.645 * s + 18.4} \tag{24}$$



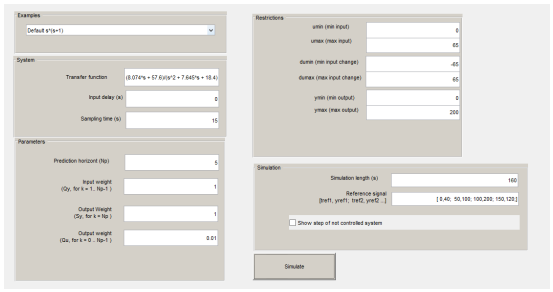
(a) Odozvy systému v pracovnom bode 80.



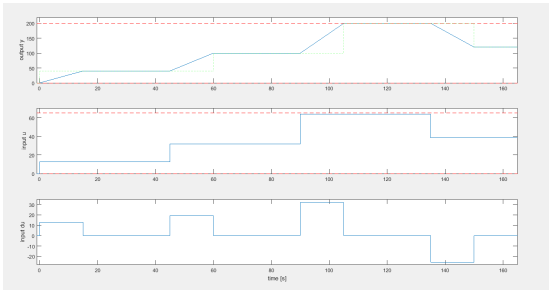
(b) Prenosová funkcia systému v pracovnom bode 80.

Obrázok 15

Pre vypočítanú prenosovú funkciu sú vykonané simulácie, kde parametre a výstup sú znázornené na obrázkoch 16a a 16b. V tomto prípade treba upozorniť na obmedzene na



(a) Parametre simulácie pre druhú prenosovú funkciu



(b) Výstup simulácie pre druhú prenosovú funkciu

Obrázok 16

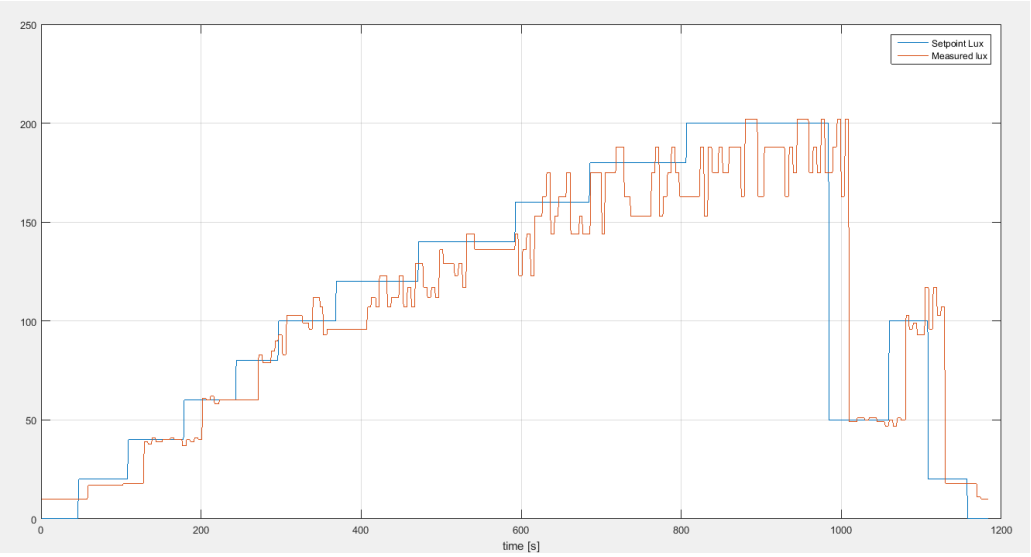
vstup, ktoré je nastavené od 0 po 65%. Ako je možné vidieť v strednom grafe obrázku 16b táto hodnota nie je dosiahnutá ani pri maximálnej intenzite osvetlenia, ktorá bola v miestnosti dosiahnutá - 200 luxov. V tejto simulácii je pracovný rozsah stmievača od 0 po 65%. Tým, že merané hodnoty boli pri identifikácii posúvané k nule, ale hlavne tým, že systém nie je lineárny a dolnú časť má na starosti iný regulátor k akčnému zásahu regulátora pre hornú oblasť je potrebné vždy pridať 100-65 = 35%. Takto identifikované systémy a parametre sú aplikované do pripraveného IoT prostredia.

1. Prenosová funkcia G_1 (23) je aplikovaná do jednej inštancie MpcClient zariadenia.

2. Prenosová funkcia G_2 (24) je aplikovaná do druhej inštancie MpcClient zariadenia a offset akčného zásahu je nastavený na 35%.

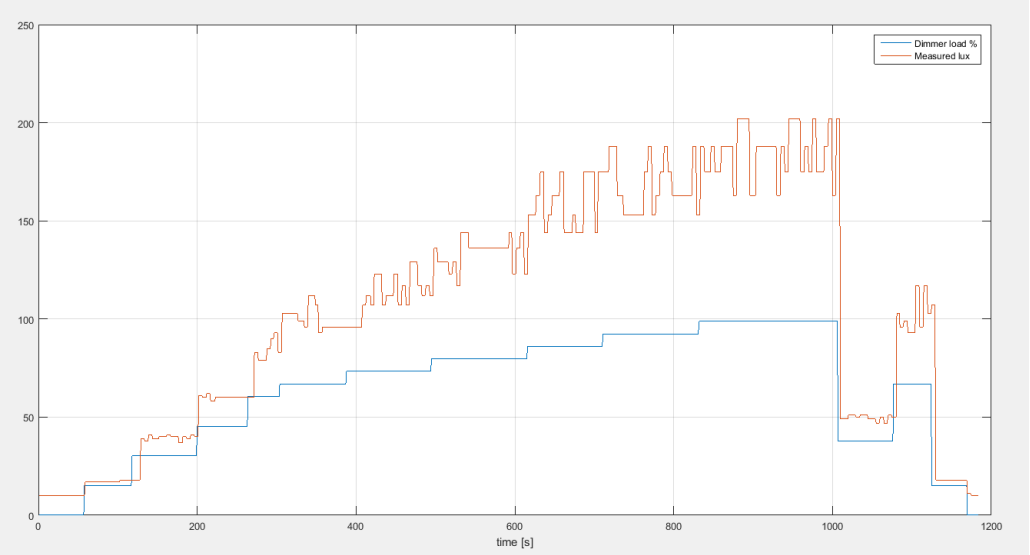
3. Na správu dvoch MpcClient zariadení je vytvorená jedna inštancia MpcMaster zariadenia, ktorá má na vstupe identifikované dve predošlé inštancie a hranicu 75 luxov žiadanej hodnoty ako rozhodujúcu. Ak je žiadaná hodnota pod touto hodnotou, tak MpcMaster volá prvú inštanciu regulátora, aby poskytla údaje o akčnom zásahu zo servera, ak je žiadaná hodnota vyššia, tak volá druhú inštanciu regulátora, ktorá je zodpovedná za získanie akčného zásahu zo servera a jeho uskutočnenie.

S pripravenými zariadeniami sa spúšťa proces riadenia pomocou tlačítka v MpcMaster zariadení, v ktorom sa rovnako postupne mení žiadaná hodnota. Namerané dáta experimentu v Matlabe sú na obrázkoch 17 a 18.



Obrázok 17: Sledovanie žiadanej hodnoty výstupnou hodnotu intenzity osvetlenia.

Na obrázku 17 vidieť modrou farbou žiadanú hodnotu a červenou farbou meranú výstupnú veličinu. Kroky žiadanej hodnoty sú na začiatku po 20 luxov až po 200. Následne je skok na 50 luxov, na 100 luxov a nakoniec na 20. Z tohto grafu nie je vidieť, v ktorom čase riadi akčný zásah iná inštancia MPC regulátor. Keďže pravidlo je pomerne jednoduché. prvé akčné zásahy vykonávala prvá inštancia, až kým žiadaná hodnota neprekročila úroveň 75 luxov. Potom až po 200 luxov vykonávala akčné zásahy druhá inštancia. Následne sa vystriedali a nakoniec posledné akčné zásahy vykonávala prvá inštancia. Ohľadne kvality regulácie môžeme vidieť, že po každej zmene žiadanej hodnoty je dopravné oneskorenie regulátora. To je spôsobene dvoma faktormi. Prvý je zvolená perióda vzorkovania 15s a druhý je čas výpočtu kroku, ktorý momentálne beží na zariadení, ktoré nemá požadovaný výkon. Keďže nejde o kritický proces naše požiadavky takéto dopravné oneskorenie akceptujú. Ďalej je možné pozorovať už spomenutý fakt nepresnosti a zašumenia mera-



Obrázok 18: Závislosť meranej hodnoty od percenta zotmenia.

nia intenzity osvetlenia pri vyššej intenzite. Do žiadanej hodnoty 100 luxov je stredná hodnota takmer zhodná so žiadanou. Následné hodnoty sú už viac zašumené, preto sa kvalita regulácie komplikovanejšie určuje. Avšak pri skokoch na 50 a 100 kedy sa testuje prepínanie regulátorov, je možné vidieť takmer nulovú regulačnú odchýlku. Pri žiadanej hodnote 20 luxov vidieť trvalá regulačnú odchýlku, ktorá je pravdepodobne spôsobená tým, že systém ma od 0 po 10 % ďalšiu nelinearitu, ktorá nie je zohľadnená pri návrhu riadenia. Rovnakým princípom ako sú vyriešené 2 pracovné body, je možné vyriešiť N pracovných bodov. V rozsahu, kde sú stabilné merania, prediktívny regulátor riadi proces s výbornými ukazovateľmi kvality regulácie a princíp CaaS - vypočítania hodnôt akčných zásahov na servery je teda plne funkčný.

Tento princíp je možné využiť napríklad na udržiavanie intenzity svetla so zapadajúcim slnkom. Ďalšie využitie výpočtového výkonu servera a neobmedzený počet regulátorov je na regulovanie teploty pomocou termostatu, ktorý v pripravenom IoT systéme je možné realizovať. Avšak experiment sa dial mimo vykurovaciu sezónu, preto nebolo možné uskutočnenie experimentu aj na tepelnom systéme. Možností využitia princípu CaaS sú týmto spôsobom neobmedzené, keďže pridanie ďalšieho regulátora je len vytvorenie novej inštancie. Stačí si zvoliť proces, ktorý treba riadiť, jeho akčný člen, senzor a identifikovať systém. Nakoniec spustiť riadiacu slučku. Pre lineárne procesy sa spúšťa na zariadení MpcClient pre nelineárne procesy pomocou viacerých MpcClientov a jedného MpcMaster zariadenia.

3.3 Činnosti súvisiace s uvádzaním IoT systému do praxe

Táto kapitola sumarizuje praktické skúsenosti z návrhu, realizácie a udržiavania IoT systému. Zdôrazňuje výhody, ale aj výzvy, ktoré IoT a teda vytváranie užšej väzby medzi IT (informačnými technológiami) a OT (operačnými technológiami) prináša. Kapitola porovnáva činnosti potrebné pre čisto softvérové systémy a činnosti potrebné pre IoT systémy.

3.3.1 Návrh systému

Ak by sme definovali pozíciu architekta IoT systému, jeho znalosti a kompetencie sú v týchto doménach, či technológiách.

- UML a znalosť možností programovacích jazykov potrebných na vytvorenie IoT Backend častí
- znalosť senzorov, akčných členov (pohonov, relé, ventilov, ...)
- znalosť protokolov a možností prepojenia jednotlivých protokolov na vytvorenie vrstvy IoT zariadení a IoT brán
- v určitých prípadoch znalosť návrhu plošných spojov
- bezpečnosť informačných systémov - autentifikácia a autorizácie na základe rolí, ...
- bezpečnosť informačný sietí - šifrovanie komunikácie, VPN, ...
- bezpečnosť zariadení
- právne aspekty domén nasadzovania IoT systémov

3.3.2 Vývoj systému

Pole pôsobenia vývojára IoT systému je tak ako v prípade architekta IoT systému rozšírené. Rozdeliť by sa znalosti dali na skupiny.

- Programovanie IoT zariadení - jazyky blízke hardvéru.
- Programovanie IoT brán - jazyky s nízkymi nárokmi na výkon, ale vysokými na rýchlu odozvu.
- Programovanie IoT backend - zahŕňajúce aktuálne konvenčné technológie.
- Programovanie IoT backend - zahŕňajúce nové rýchlo rozvíjajúce sa technológie na spracovanie a analýzu dát veľkých objemov, rýchlosti a variability.

3.3.3 Prevádzka systému

Administrátorovi IoT systému prislúchajú tieto úlohy.

- Kontrola stavu služieb a serverov resp. brán.
- Kontrola stavu senzorov akčných členov a ich oprava respektíve výmena batérií. oprava

Záver

Hlavným cieľom predloženej dizertačnej práce bolo vypracovanie novej metodiky pre návrh, vývoj a realizáciu prediktívnych regulátorov v súlade s rozvojom moderných informačných technológií využívajúcich nové trendy, ktoré môžeme charakterizovať ako efektívne spojenie metód a algoritmov riadenia s IoT prístupmi a technológiami. Na základe tejto metodiky je v práci navrhovaný nový prístup riadenia s novou myšlienkou, kde regulátor je chápaný ako služba (Caas - Controller as a Service). Vzhľadom na náročnosť riešení úloh prediktívneho riadenia, ktoré vyžadujú zložité numerické algoritmy na optimalizáciu so zahrnutím ohraničení na riadiaci zásah, diferenciu riadiaceho zásahu a výstupnú regulovanú veličinu, bolo nutné zefektívniť takéto riešenia cez IoT systém a CaaS prístup.

V práci je pojem IoT zadefinovaný podloženými literárnymi zdrojmi. IoT systém je v práci podrobne opísaný a architektonické voľby návrhu systému zdôvodnené. Rovnako implementácia IoT systému a problém riadenia sa opiera o architektonické princípy, ktoré sú definované v práci a vychádzajú z najnovších trendov v informačných technológiách. Myšlienka CaaS a jej realizácia je v práci opísaná. Ide tu o vystavenie online metódy MPC regulátora prostredníctvom REST architektonického princípu, čo znamená, že bol vytvorený REST resource (zdroj) regulátor a jeho subresource (podzdroj) krok regulácie daného regulátora. Na základe tohto prístupu je vyvinutá aplikácia riadenia jednoducho aplikovateľná do iných IoT systémov. Jediná požiadavka na umožnenie regulácie procesu je, aby akčný člen vedel vytvoriť HTTP požiadavku a spracovať správu vo formáte JSON. Využitie prediktívneho regulátora a jeho zabudovania do IoT bolo v práci overené na reálnej inteligentnej domácnosti, kde hlavnou úlohou MPC regulátora, bolo zabezpečenie intenzity osvetlenia. Zložité numerické riešenia boli poskytované prostredníctvom IoT Backend súčasti, ktorá je charakteristická disponovaním veľkou výpočtovou kapacitou. V práci bol vytvorený podporný program na simuláciu MPC regulátora, ktorý bol využitý pri realizácii CaaS na porovnávanie výsledkov reálneho systému a simulácie. Pri tomto riešení sa ukázali niektoré nedostatky v ohraničení možnej periódy vzorkovania a v kritických procesoch, pri ktorých je potrebné získanie odozvy v kratšom čase, ako je čas potrebný na sieťovú komunikáciu. Avšak výhody navrhovaného prístupu sú veľmi významné a ukázalo sa, že aj zložité výpočty je možné realizovať takmer v reálnom čase a že je možné vytvoriť ľubovoľný počet inštancií regulátorov a riadiť tak všetky procesy v IoT systéme bez ďalších nákladov, čo potvrdzuje uvedené výhody.

Výsledky dizertačnej práce možno využiť pri riešení nových výskumných úloh zameraných

na moderné metódy automatického riadenia v integrácii s novými informačnými technológiami využívajúce IoT a Big Data.

Zoznam použitej literatúry

[1] 10 iot predictions for 2016 - page: 1 | crn. http://www.crn.com/slide-shows/networking/300079629/10-iot-predictions-for-2016.htm. Accessed on 14-06-2016).

[2] Chapter 20: Choosing an application type. https://msdn.microsoft.com/en-us/library/ee658104.aspx. (Accessed on 16-06-2016).

[3] How to choose the right software architecture: The top 5 patterns. http://techbeacon.com/top-5-software-architecture-patterns-how-make-right-choice. (Accessed on 17-06-2016).

[4] Integracia-soa-rest.pdf. https://prest-tech.appspot.com/docs/Integracia-SOA-REST.pdf. (Accessed on 18-06-2016).

[5] The internet of things - internet of things companies. http://www.gartner.com/it-glossary/internet-of-things/. (Accessed on 19-06-2016).

[6] Iot—the industrial way | iot content from electronic design. http://electronicdesign.com/iot/iot-industrial-way. (Accessed on 20-06-2016).

[7] Jeffrey friedl's blog » simple json encode/decode in pure lua. http://regex.info/blog/lua/json. (Accessed on 22-06-2016).

[8] Jsonlab: a toolbox to encode/decode json files - file exchange - matlab central. https://www.mathworks.com/matlabcentral/fileexchange/33381-jsonlab--a-toolbox-to-encode-decode-json-files. (Accessed on 22-06-2016).

[9] Luup intro - micasaverde. http://wiki.micasaverde.com/index.php/Luup_Intro. (Accessed on 21-06-2016).

[10] Microsoft word - ieee_iot_towards_definition_internet_of_things_revision1_27may15.docx. http://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf. (Accessed on 18-06-2016).

[11] Simple steps to learn iot architecture. https://www.linkedin.com/pulse/simple-steps-learn-iot-architecture-brucelin-kithion. (Accessed on 20-06-2016).

[12] What is big data? - gartner it glossary - big data. http://www.gartner.com/it-glossary/big-data/. (Accessed on 20-06-2016).

[13] What is the internet of things (iot)? - definition from techopedia. https://www.techopedia.com/definition/28247/internet-of-things-iot. (Accessed on 19-06-2016).

[14] Where can i find the best description of the iot architecture and development flow? - quora. https://www.quora.com/Where-can-I-find-the-best-description-of-the-IoT-architecture-and-development- (Accessed on 20-06-2016).

[15] Bemporad, A., and Morari, M. Robust Model Predictive Control: A Survey, 1999.

[16] Řehoř, J. Návrh explicitního prediktivního regulátoru s využitím Multi-Parametric Toolboxu pro Matlab. http://www2.humusoft.cz/www/papers/tcp07/rehor.pdf. Accessed on 12-02-2014).

[17] Richards, M. Software architecture patterns, 2015.

Publikačná činnosť autora

[1] **AFC** PYTEL, Anton - KOZÁK, Štefan. Modelling and effective predictive control of gas turbine process. In Proceedings of the 15th International Carpathian Control Conference [elektronický zdroj] : ICCC 2014; Velké Karlovice, Czech Republic, May 28-30, 2014. [s.l.] : IEEE- Czechoslovakia Section of IEEE, 2014, CD-ROM, p. 469-474. ISBN 978-1-4799-3527-7.

[2] **AFD** KOZÁK, Štefan - PYTEL, Anton - DRAHOŠ, Peter. Application of hybrid predictive control for intelligent buildings. In Process control 2015 : 20th International Conference on Process Control. Štrbské Pleso, Slovak Republic. June 9-12, 2015. 1. vyd. [s.l.] : IEEE, 2015, S. 203-208. ISBN 978-1-4673-6627-4. V databáze: IEEE.

[3] **BEE** PYTEL, Anton - KOZÁK, Štefan - DÚBRAVSKÝ, Jozef. On-line Predictive Controller for Gas Turbine. In 3rd International Conference on Advanced Control Circuits and Systems (ACCS' 013) ; 2nd International Conference on New Paradigms in Electronics & Information Technology (PEIT ' 013) : Luxor, Egypt, 30 November - 3 December 2013. Luxor : Electronics Research Institute, 2013, s.CD-ROM, 5 s.

Iné publikačné aktivity v oblasti výskumu a vývoja

- Článok o myšlienke CaaS poslaný do časopisu IREACO.
- Spoluorganizátor komunity IoT Bratislava.
- Člen komunity IoT Vienna s účasťou na podujatiach a diskusiách organizovaných touto komunitou.
- Prezentácie ohľadne zavádzania REST architektonických princípov do enterprise architektúry v komerčnej sfére.