VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2024 Bc. Petr Černocký

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

INTEGRACE SIMULÁTORU VÝMĚNÍKOVÉ STANICE DO AAS

INTEGRATION OF SIMULATOR IN AAS FOR HEAT TRANSFER STATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Petr Černocký

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Jakub Arm, Ph.D.

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Petr Černocký ***ID:*** 220969

Ročník: 2 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Integrace simulátoru výměníkové stanice do AAS

POKYNY PRO VYPRACOVÁNÍ:

Úkolem práce je propojit simulátor (např. pomocí Matlab nebo Unity) výměníkové stanice s technologií AAS. Simulátor bude sloužit jako simulace výrobního zařízení, které obsahuje průmyslové komponenty. AAS by mělo obsahovat základní modely pro popis zařízení a hlavně funkcionality pro kooperaci se simulátorem. Účelem je řídit dané zařízení na základě simulace.

1. Proveďte rešerši v oblasti modelování a simulace průmyslových systémů.
2. Vytvořte simulátor zvoleného zařízení.
3. Navrhněte AAS dle požadavků obsahující simulátor a komunikační prostředky.
4. Realizujte.
5. Vyhodnoťte parametry vytvořeného systému.

DOPORUČENÁ LITERATURA:

Plattform Industrie 4.0. Details of the Asset Administration Shell. Berlin, 2020. Dostupné na: https://www.plattform-i40.de

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 15.5.2024

Vedoucí práce: doc. Ing. Jakub Arm, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

Abstrakt

Tato práce zkoumá dnešní moderní metody řízení výroby, bez kterých by se neobešel celý fenomén průmyslu 4.0. Hlavním zaměřením této práce je Asset Administration Shell a integrace modelů a simulací do něj. Dále je zde provedena rešerše v oblasti modelování a vizualizace výroby a výrobních procesů, která je opatřena i ukázkou komunikace mezi vizualizačním softwarem UNITY a výpočetním softwarem Matlab. Z rozboru je dále vybrán nejvhodnější software, pro konečnou implementaci. Dále je zde proveden návrh řešení integrace simulátoru do AAS a následně jeho praktické provedení. Zároveň je zde zhodnocen celý navržený systém s integrovaným simulátorem.

Klíčová slova

AAS, Unity, Basyx, průmysl 4.0, Modelica, simulace procesů

Abstract

This work examines today's modern methods of production management, without which the entire Industry 4.0 phenomenon would not be possible. The main focus of this work is the Asset Administration Shell and the integration of models and simulations into it. Furthermore, research is carried out in the area of modeling and visualization of production and production processes, which is also provided with an example of communication between the visualization software UNITY and the calculation software Matlab. From the analysis, the most suitable software is also selected for the final implementation. Furthermore, there is a proposal for the integration of the simulator into AAS and then its practical implementation. At the same time, the entire designed system with an integrated simulator is evaluated here.

Keywords

AAS, Unity, Basyx, industry 4.0, Modelica, process simulations

Bibliografická citace

ČERNOCKÝ, Petr. *Integrace simulátoru výměníkové stanice do AAS.* Brno, 2024. Dostupné také z: https://www.vut.cz/studenti/zav-prace/detail/159387. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Jakub Arm.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Petr Černocký
VUT ID studenta:	220969
Typ práce:	Diplomová práce
Akademický rok:	2023/24
Téma závěrečné práce:	Integrace simulátoru výměníkové stanice do AAS

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 10. května 2024 ----------

podpis autora

Poděkování

Rád bych poděkoval svému vedoucímu diplomové práce panu docentu Ing. Jakubovi Armovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

V Brně dne: 10. května 2024 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Úvod

Tato práce zkoumá moderní možnosti řízení výroby a procesů s využitím AAS. V první kapitole jsou popsány a rozebrány technologie, bez kterých se neobejde průmysl 4.0. A dále obsahuje informace o regulačních obvodech, které jsou dále využity v této práci.

Ve druhé kapitole jsou představeny a krátce rozebrány dostupné simulační a modelovací aplikace. Kapitola se dále zaměřuje na jejich srovnání. Je zde proveden pokus o spojení dvou simulačních nástrojů, a to Unity3D a Matlabu. Toto spojení je dále vyhodnoceno na základě měření doby odezvy. Z naměřených dat jsou pak za pomocí aplikace LabView vypočteny nejistoty měření. V poslední části této kapitoly je uvedena ukázka simulačního nástroje OpenModelica a shrnutí dostupných nástrojů a jejich vyhodnocení.

Třetí kapitola se věnuje návrhu řešení integrace simulátoru do AAS. Je představen reálný systém výměníkové stanice, který bude řízen za pomocí AAS. Je zde jednoduchý popis funkčnosti a zároveň odkazy na přiložené procesní schéma celého systému včetně popisu jednotlivých komponent. Dále jsou zde popsány jednotlivé prvky, které byly využity v návrhu.

V poslední kapitole je popsána realizace systému a obsah jednotlivých submodelů. Zároveň je zde uvedeno srovnání reálného systému s jeho modelovým ekvivalentem. Dále je zde provedeno vyhodnocení celého systému a pohled na jeho možnosti a limity. Nakonec jsou zde diskutovány i důležité věci ze strany bezpečnosti.

Cílem celé práce je vytvořit funkční systém s integrovaným simulátorem uvnitř AAS, zhodnotit jeho možnosti využití a odhalit možné limity systému a navrhnout jejich možné řešení.

1. TEORETICKÁ REŠERŠE

Tato kapitola pojednává o základních technologií a procesech, využívaných v moderním řízení výroby. Tyto technologie mají sloužit především k zefektivnění a zjednodušení výroby, případně ke komplexnějšímu návrhu životního cyklu komponenty a systému a objevení jejich slabin. Dále jsou zde rozebrány témata z oblasti vyhodnocování regulačního děje a malá rešerše rozvětvených regulačních obvodů.

1.1 Průmysl 4.0

Fenoménem dneška je propojování internetu věcí, služeb a lidí, a s ním související nesmírný objem generovaných dat ať už komunikací stroj – stroj, člověk – stroj nebo člověk – člověk. Nástup nových technologií transformuje celé hodnotové řetězce, otevírá nové možnosti pro obchodní modely, ale také klade důraz na flexibilitu v moderní průmyslové výrobě a zvyšuje nároky na kybernetickou bezpečnost a interdisciplinární přístup. Iniciativa Průmysl 4.0 nepředstavuje pouze snahu o digitalizaci průmyslové výroby, ale je komplexním systémem změn, který zahrnuje mnoho lidských aktivit, a to nejen v průmyslovém odvětví.[1]

Na Hannoverském veletrhu byla v roce 2011 představena první vize německé vlády ohledně budoucího vývoje průmyslu. Oficiálně byla platforma "Industrie 4.0" spuštěna na stejném místě o dva roky později. [1]

1.1.1 Hlavní znaky

Průmysl 4.0 přináší transformaci výroby z izolovaných automatizovaných jednotek na plně integrovaná, automatizovaná a neustále optimalizovaná výrobní prostředí. Vznikají nové globální sítě založené na propojení výrobních zařízení do kyberneticko-fyzických systémů. Tyto systémy se stávají základním stavebním kamenem "inteligentních továren", které jsou schopny autonomního sdílení informací, spouštění potřebných akcí v reakci na aktuální podmínky a vzájemné nezávislé kontroly. Senzory, stroje, součástky a IT systémy jsou propojeny v rámci hodnotového řetězce, překračujícího hranice jednotlivých firem. Tyto propojené CPS pak vzájemně reagují a analyzují data pomocí standardních komunikačních protokolů založených na internetu. To jim umožňuje předvídat potenciální chyby či poruchy, konfigurovat se a dynamicky se přizpůsobovat změnám podmínek v reálném čase. [1]

Základní charakteristiky inteligentních továren, které odpovídají konceptu Průmysl 4.0, lze shrnout dle [1] následovně:

- Výrobní procesy jsou optimalizovány v celém hodnotovém řetězci díky vertikálně i horizontálně integrovaným IT systémům.
- Izolované výrobní jednotky jsou nahrazeny plně automatizovanými a vzájemně propojenými výrobními linkami.

- Fyzické prototypy jsou nahrazeny virtuálními návrhy výrobků, výrobních prostředků a procesů, a jejich uvedení do provozu probíhá v rámci integrovaného procesu zapojujícího výrobce i jeho dodavatele.
- Flexibilní výrobní procesy umožňují efektivní výrobu i malých dávek přizpůsobených individuálním požadavkům zákazníků.
- Vzájemně komunikující roboti, výrobní zařízení a výrobky částečně autonomně rozhodují v reálném čase, čímž zvyšují flexibilitu a efektivitu výrobního procesu.
- Výrobní zařízení se samo optimalizuje a konfiguruje podle parametrů zpracovávaného produktu.
- Automatizované logistické zázemí využívá autonomních vozíků a robotů, které se automaticky přizpůsobují potřebám výroby. Logistické zázemí zahrnuje více subjektů v kooperaci, které nemusí být lokalizovány na jednom místě, což zahrnuje i koordinaci dopravního spojení mezi výrobními subjekty a distribučnímu procesu výrobku.

1.2 AAS – Asset Administration Shell

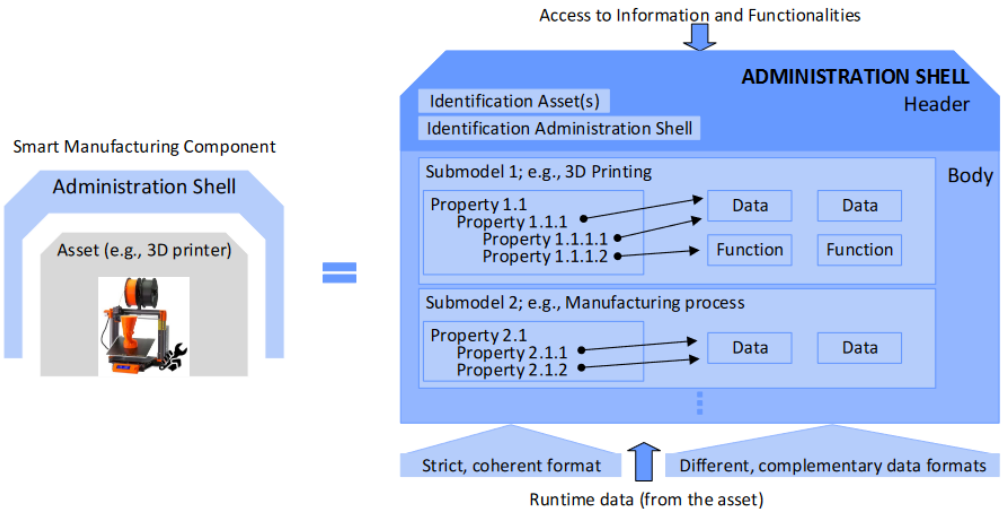
Asset Administration Shell (AAS) je konceptuální model a architektonický framework, který vznikl v rámci iniciativy Industry 4.0, která se zabývá digitalizací a automatizací průmyslových procesů. AAS slouží k reprezentaci, správě a sdílení informací o průmyslových aktivách a zařízeních v digitální podobě. AAS můžeme taktéž považovat za specifickou verzi digitálního dvojčete.

AAS lze též považovat za klíčovou jednotku v oblasti myšlenky průmyslu 4.0 a jeho snahu o decentralizované řízení výroby – orientaci na služby (Service Oriented Arhitecture).

Digitální obálka – tak se někdy v překladu AAS říká, obsahuje veškeré informace o svém assetu – fyzické či virtuální věci. AAS může existovat jak pro produkt, zařízení ale i celý proces, což je i případ téhle práce.

1.2.1 Struktura AAS

Obálku (Administration Shell) tvoří dvě části. Hlavička (Header), která obsahuje základní unikátní identifikátory jak Assetu, tak digitální obálky. A tělo (Body), v něm se nacházejí další informace – submodely, např. pro komunikaci nebo o aktuálním stavu produktu. [2]



Obrázek 1.1 Základní struktura AAS

1.2.2 Požadavky na AAS

Při tvorbě AAS je potřeba dodržovat strukturu, vlastnosti, obsah submodelů. Vlastnosti AAS byly specifikovány pracovními skupinami ZVEI, VDI/VDE, GMA, IDTA. Tyto požadavky jsou uvedeny v tabulce 1.1.

Tabulka 1.1 Přehled požadavků AAS – tabulka vychází z literatury [3]

Číslo	Popis
1	AAS přijímá vlastnosti z různých technických oblastí vzájemně odlišných submodelech, které jsou udržovány nezávisle na sobě
2	AAS by měl být schopen zahrnout vlastnosti z široké škály technických domén a identifikovat odkud jsou odvozeny
3	Pro nalezení definic v každé příslušné technické oblasti by měly být povoleny různé procedurální modely, které splňují požadavky norem, specifikací konsorcia a souborů specifikací výrobce
4	Různé přidružené systémy ve vztahu k aktivu (Assetu) musí být schopny se navzájem odkazovat. Prvky přidruženého systému by měly být schopny plnit roli kopie odpovídajících prvků z jiného AAS
5	Jednotlivé AAS by měly při zachování své struktury být schopny sloučeny do celkového AAS
6	Identifikace AAS, jejich vlastností a vztahů musí být dosaženo za pomocí omezené sady identifikátorů, které poskytují co největší jedinečnost
7	AAS by mělo umožňovat zpětné získávání alternativních identifikátorů, jako náhradu za identifikátor Assetu
8	Asset Administration Shell obsahuje hlavičku a tělo
9	Hlavička obsahuje informace o identifikaci, dále obsahuje také identifikaci jednoho nebo více aktiv, která jsou popsané v AAS
10	Tělo obsahuje informace o příslušném assetu
11	Informace a funkce v AAS jsou přístupné prostřednictvím standardizovaného aplikačního rozhraní (API)

12	AAS má unikátní ID
13	Asset má unikátní ID
14	Aktivem může být také průmyslové zařízení, koncepce AAS musí být použitelná na všech úrovních hierarchie (od ovládacího prvku, až po celý závod)
15	Typy a instance musí být také identifikovány
16	AAS může obsahovat odkazy na jiná AAS, nebo informace ke SmartManufacturing
17	V každém AAS musí být možné další vlastnosti, např. specifika výrobce
18	V každé AAS musí být definováno nutné minimum informací.
19	Vlastnosti a další prvky informací v AAS musí být vhodná pro všechny typy a instance.
20	Musí existovat možnost hierarchického strukturování vlastností
21	Vlastnosti musí být schopny odkazovat na jiné vlastnosti i v jiných AAS.
22	Vlastnosti musí být schopny odkazovat na funkce a informace.

1.2.3 Identifikátory AAS

Identifikátory jsou klíčovým a účinným prostředkem pro jednoznačnou identifikaci prvků v rámci Průmyslu 4.0. Identifikace je nezbytná dle [4] zejména pro:

- Asset Administration Shell (administrační obálku prvků)
- Asset (identifikátor obsažený v Asset Administration Shell)
- Submodel (platí jak pro šablonu, tak pro instanci)

Existují tři hlavní druhy identifikátorů, které jsou k dispozici:

- Identifikátor údajů o mezinárodní registraci (IRDI), definovaný normou ISO 29002-5, ISO/IEC 6523 a ISO/IEC 11179-6 [4]
- Internacionalizovaný identifikátor zdrojů (IRI), definovaný podle RFC 3986 [4]
- Vlastní identifikátory, jako například univerzálně jedinečný identifikátor (UUID), které si výrobce definuje podle svých interních potřeb.[4]

Identifikace nejen umožňuje jednoznačné rozlišení prvků v Asset Administration Shell, ale také vytváří vazby (odkazy) mezi šablonami a instancemi submodelů, metamodely a prvky Asset Administration Shell.

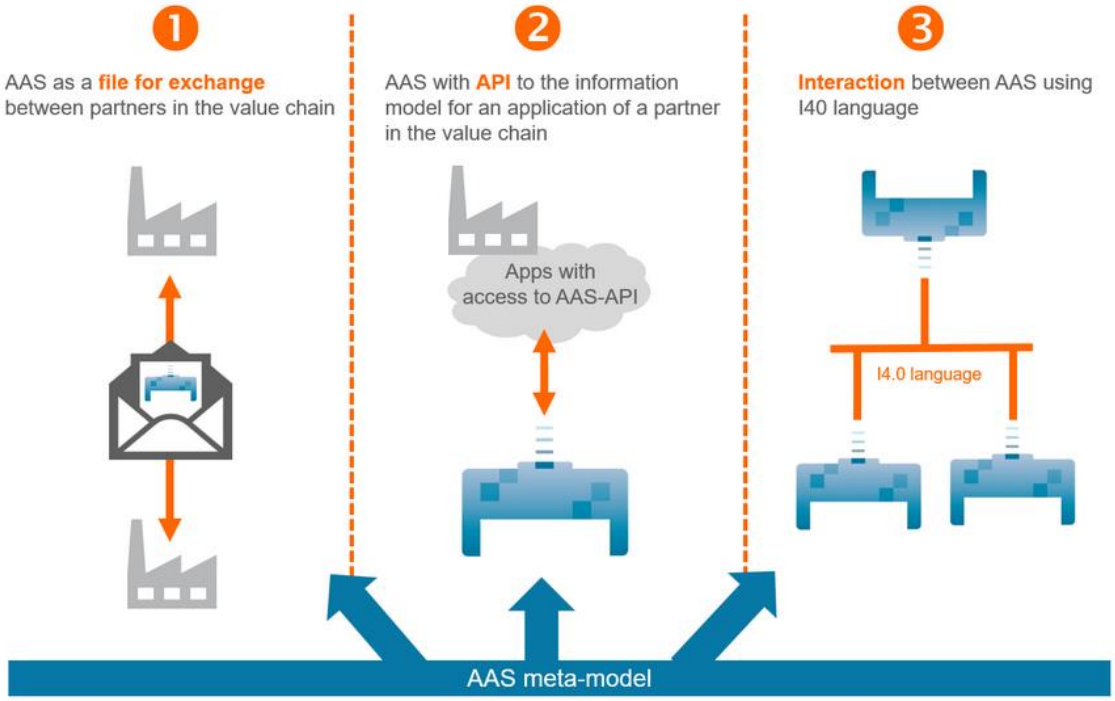
1.2.4 Dělba AAS

Asset Administration Shel se v základě dělí na 3 typy:

- **Pasivní AAS**

  Tento typ AAS se zaměřuje na pasivní monitorování a sledování průmyslových zařízení. Pasivní AAS pouze sbírá data, nijak jinak nezasahuje do běhu zařízení. Tento typ se využívá v situacích, kdy není potřeba nijak ovládat zařízení, ale pouze monitorovat a získávat informace pro účely diagnostiky, údržby a plánování výrobního procesu.[5]

- **Reaktivní AAS**
  Druhý typ AAS již využívá aktivní řízení a reakci na změny nebo události týkající se průmyslových aktiv a zařízení. Reaktivní AAS je navržen tak, aby mohl interagovat s prostředím a provádět akce na základě různých podnětů a scénářů. Tím se zvyšuje schopnost AAS efektivně řídit průmyslové procesy a optimalizovat výkonnost zařízení.[5]
- **Aktivní AAS**
  Jedná se o třetí a poslední typ AAS. Tento typ má v sobě určitou inteligenci, proto může sám rozhodovat. Aktivní AAS by měl být navržen tak, aby cíleně ovlivňoval průmyslové procesy za účelem zefektivnění výroby. Tato vlastnost je účinně využívána právě v myšlenkách průmyslu 4.0, kde hraje klíčovou roli autonomie rozhodování zařízení a eliminace lidské chyby. [5]



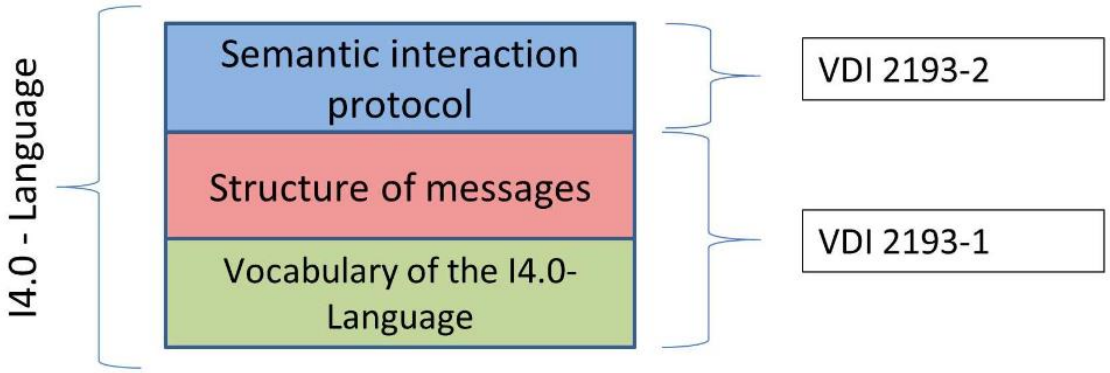
Obrázek 1.2 Typy AAS [5]

1.2.5 Komunikace

Asset administration shell vzájemně komunikují v průmyslovém prostředí pomocí standardizovaných protokolů a rozhraní. Tato komunikace umožňuje průmyslovým aktivům a zařízením sdílet informace, spolupracovat a efektivně se zapojovat do průmyslových procesů.

Jednotlivé komunikační rozhraní definuje standard VDI2193-1/2. Tato norma je součástí souboru pravidel a norem, které se týkají automatizace průmyslových procesů a průmyslového inženýrství. Celý jazyk průmyslu 4.0 se skládá ze tří částí. První část

definuje VDI 2193-1, což je tzv. slovník jazyku 4.0. Druhou a třetí část popisuje druhá část VDI2193-2. [6]



Obrázek 1.3 Schéma jazyku průmyslu 4.0 [6]

Celý princip komunikace spočívá v rozdělení zařízení na žadatele a poskytovatele služeb. Této architektuře se taky někdy říká SOA – Service orientering architecture (architektura orientovaná na služby). Fungování této architektury si lze představit jako výrobek, který žádá po okolních strojích – poskytovatelích (nemusí se nutně jednat o stroje). Z poskytovatele se v průběhu může stát žadatel např. stroj vyžadující opravu. Žadatel se naopak nemůže stát poskytovatel. Dále je pak potřeba, aby žadatel byl schopen rozhodnout, která nabídka, od kterého poskytovatele je pro něj nejlepší. [6]

1.2.6 Shrnutí AAS

Asset administration shell je konceptuální model, který se snaží pomocí určité standardizace nastavit správný pohled na myšlenku průmyslu 4.0. Skupina IDTA se nesnaží pouze o vytváření standardů, ale dále pomocí zapojených společností vytváří příklady jednotlivých submodelů, které jsou již odladěny a připraveny pro přímé použití v průmyslu.

Jedním z cílů této práce je ukázat, že koncept AAS lze využít i v jiné než diskrétní výrobě, a to při návrhu a simulaci různých procesů a využít možnosti integrace těchto simulátorů do AAS. Dále je v této práci využití jak vlastností pasivního, tak i reaktivního typu AAS.

1.3 Integrální kritéria kvality regulace

Kritéria kvality zjišťují kvalitu nastavení parametrů regulátoru v časové oblasti. Vychází se z průběhu regulační odchylky, která je získána z odezvy regulované soustavy na skokovou změnu žádané hodnoty.

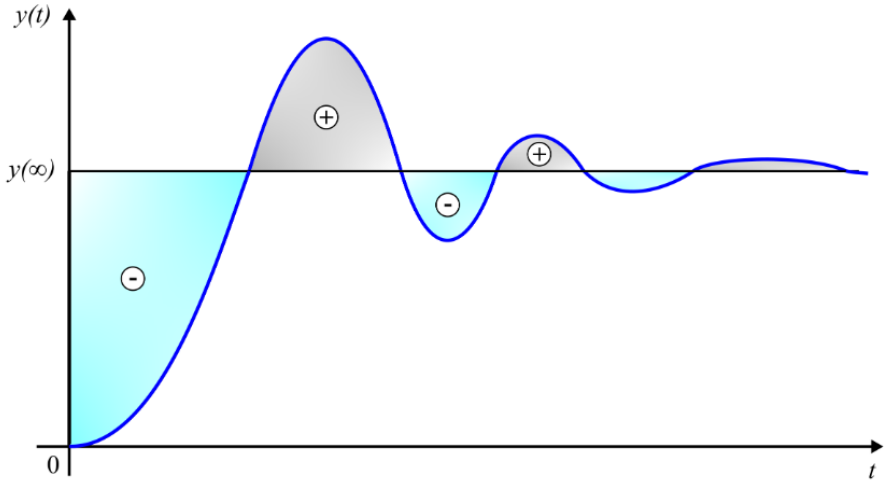
1.3.1 Lineární kritérium

Lineární integrální kritérium slouží k výpočtu oblasti mezi křivkou průběhu regulační

odchylky e(t) a konstantní odchylkou e(∞). Tato oblast je známá jako lineární regulační oblast. V matematickém smyslu je tato oblast, která je omezena určitou křivkou, vyjádřena pomocí integrálu:

$$J_L = \int_0^{\infty} [e\,(t) - e(\infty)]dt\,, \qquad (1.1)$$

Pro účely výpočtu daného integrálu je nezbytné, aby systém byl aperiodický. V případě, že by systém byl periodický, oblasti pod úrovní konstantní odchylky e(∞) by byly od sebe odečítány. To by vedlo k nesprávnému snížení hodnoty kritéria a výsledek by tak byl zkreslený. Tento problém je znázorněn na obrázku 1.4 Problém lineárního integračního kritéria. Řešením tohoto problému je využití modifikovaných kritérií usměrněné lineární plochy např. využitím absolutní hodnoty z rozdílu hodnot. [7]



Obrázek 1.4 Problém lineárního integračního kritéria [8]

1.3.2 Kvadratické integrální kritérium

Toto kritérium vyjadřuje kvadrát regulační plochy a je definováno integrálem:

$$J_L = \int_0^{\infty} [e\,(t) - e(\infty)]^2 dt\,, \qquad (1.2)$$

Díky tomuto kritérium není potřeba řešit záporné odchylky, neboť kvadrát dané hodnoty zaručuje vždy kladný výsledek. Dále toto kritérium dává velkou váhu velkým regulačním odchylkám většinou z počátku regulačního děje. Jednou z dalších výhod tohoto kritéria je jednoduchost jeho výpočtu.

Při snaze o minimalizaci kvadratického kritéria dochází k situaci, kdy se systém pokouší odstranit odchylky zejména v počáteční fázi co nejefektivněji. Tento proces typicky vyvolává poměrně vysoký překmit a oscilace odchylky, což je často považováno za nevýhodu kvadratického kritéria. [7]

1.3.3 ITAE kritérium

Kvůli výše zmíněné nevýhodě kvadratického kritéria bylo vytvořeno další hodnotící

kritérium a to ITAE (Integral of Time Multiple by Absolute value of Error). Toto kritérium je definováno vztahem:

$$J_L = \int_0^{\infty} |e\,(t) - e(\infty)| t\, dt\,, \tag{1.3}$$

Kde *e(t)* je časový průběh regulační odchylky, *e(∞)* je trvalá ustálená odchylka a *t* je čas. Jedná se tedy o váhové kritérium, kde váhou je čas dané regulační odchylky. Tímto se řeší problém velké odchylky při začátku regulačního děje, jelikož váhová proměnná času nabývá v té chvíli nejmenšího hodnot. [7]

1.4 Návrh regulátoru metodou Ziegler-Nicholse

Jde o jednoduchý přístup, který nepotřebuje hluboké znalosti z oblasti dynamického systémového řízení. V praxi je oblíben a často používán právě díky své jednoduchosti. Postup vyžaduje měření na skutečném objektu, model systému nebo případně data získaná simulací modelu.

Princip této metody spočívá v získání kritických parametrů systému a z těchto parametrů lze následně určit konstanty daného regulátoru dle tabulky 1.2**.** Pro nalezení optimálních hodnot nastavení regulátoru je nutno přivést systém s regulátorem na hranici stability. Tato hranice je nazývána kritický stav, což znamená, že regulátor je nastaven tak, aby byla aktivní pouze jeho proporcionální složka. Integrační a derivativní složky jsou v této fázi nulové. Díky proporcionální složce se zvyšuje zesílení regulátoru až do doby, než regulační obvod začne kmitat s konstantní amplitudou. [7]

Takový pokus není přijatelný pro všechny systémy kvůli technologickým a bezpečnostním důvodům. Postup pro určení kritických parametrů z výsledků simulace se neliší od zjišťování těchto hodnot na reálném systému. Existují čtyři možnosti pro takové systémy:

- identifikovat kritické parametry z přechodové charakteristiky,
- použít model systému a určit kritické parametry z výsledků simulace,
- použít model systému a určit kritické parametry výpočtem.
- použít relé bez hystereze.

1.4.1 Určení kritických parametrů z přechodové charakteristiky

V této práci bude využita tato možnost zjištění parametrů. Pokud máme k dispozici přechodovou charakteristiku, můžeme určit kritické parametry odečtením doby průtahu T_u a doby náběhu T_n. Pro kritické parametry platí přibližné vztahy:

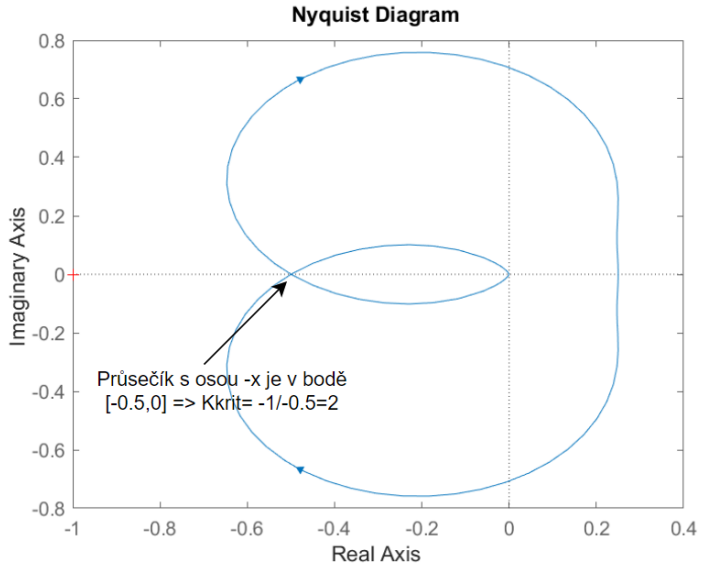
$$K_{krit} \doteq \frac{\pi}{2}\frac{T_n}{T_u} + 1\,, \tag{1.4}$$

$$T_{krit} \doteq 4T_u\,, \tag{1.5}$$

Z těchto parametrů lze následně dopočítat parametry pro PID regulátoru, případně jejich částí. [7]

1.4.2 Určení kritických parametrů výpočtem ze známého modelu

Pokud máme k dispozici model systému, zajímá nás zesílení, které přivede systém na hranici stability. K tomu můžeme využít různých postupů. Podle Nyquistova kritéria stability je uzavřený obvod na hranici stability, pokud frekvenční charakteristika otevřené smyčky prochází bodem (-1, 0). Na základě tohoto kritéria stačí určit průsečík frekvenční charakteristiky $F_0(j\omega)$ se zápornou částí reálné osy (-x, 0). Kritické zesílení je potom $K_{krit}=1/x$. Určení tohoto průsečíku není pro systémy vyšších řádů jednoduché, proto je lepší využít pro určení kritického zesílení některé z algebraických kritérií stability (Routh-Schurovo nebo Hurwitzovo). Kritická perioda se spočítá z podmínky pro nulovou imaginární část. [7]



Obrázek 1.5 Určení K_{krit} pomocí Nyquistova kritéria

1.4.3 Rozkmitávání pomocí relé bez hystereze

Někdy se pro zjištění kritických parametrů systému používá ideální relé bez hystereze místo proporcionálního regulátoru. Výhodou je, že kmitání jsou poté řízené (amplituda kmitů závisí na amplitudě relé) a nehrozí proto, že by se systém nekontrolovaně rozkmital.

1.4.4 Závěr

Metoda Ziegler-Nicholse je vhodná v případě, kdy nemáme k dispozici přenosovou funkci soustavy, ale můžeme na ni provádět experiment spočívající v přivedení soustavy na hranici stability. Toto může být v některých případech nebezpečné, proto je lepší využít rozkmitávání pomocí relé bez hystereze, kdy je amplituda kmitů řízena výstupní amplitudou relé. Z kritických parametrů lze následně určit konstanty parametrů navrhovaných regulátorů.

Tabulka 1.2 Tabulka pro návrh konstant regulátorů [7]

Typ regulátoru	K_R	T_I	T_D
P	$K_R=0,5K_{krit}$	-	-
PI	$K_R=0,45K_{krit}$	$T_I=0,85T_{krit}$	-
PD	Nutno doladit	-	$T_D=0,12T_{krit}$
PID	$K_R=0,6K_{krit}$	$T_I=0,5T_{krit}$	$T_D=0,125T_{krit}$

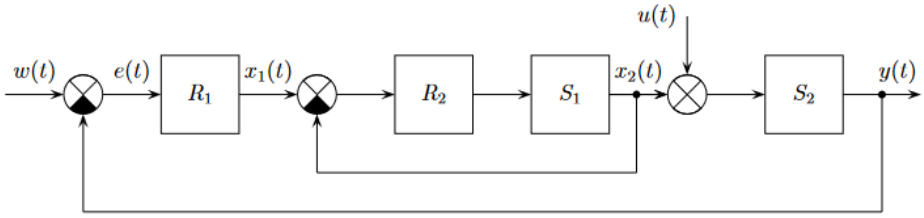
1.5 Rozvětvené regulační obvody

Jedná se o složitější soustavy s více než jednou zpětnou vazbou. Dále se vyznačují tím, že bývá od obvodu vyžadována větší kvalita regulace, případně je požadováno optimální uspořádání regulačního obvodu. Ke zlepšení kvality regulace je využíváno různých pomocných veličin. O těchto typech pojednávají další podkapitoly.

1.5.1 Regulační obvod s pomocnou regulovanou veličinou

Základní blokové schéma systému je znázorněno na obrázku 1.6. V řízeném systému, který obsahuje bloky S_1 a S_2 spojené sériově, je měřená veličina $x_2(t)$, která je regulována regulátorem R_2 podle požadované hodnoty $x_1(t)$ (hlavní řídicí veličina). Hlavním regulátorem je zde blok R_1. [7]

Pomocná regulovaná veličina je často využívána při řízení teploty a polohových servomechanismech. Při regulaci teploty je systém obvykle složen z více setrvačných článků propojených sériově. Zavedením pomocné regulované veličiny měřené v blízkosti vstupu do systému získáváme schopnost rychlejší reakce na vzniklou poruchu a tím lepší potlačení této poruchy.[8]



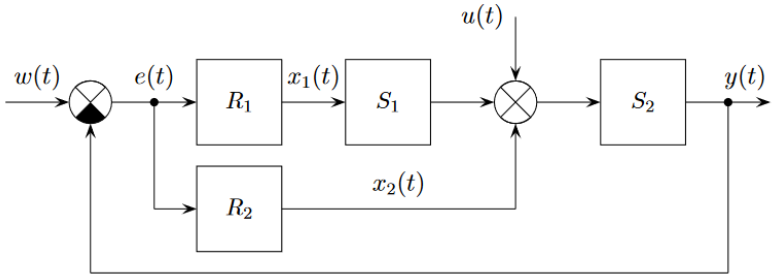
Obrázek 1.6 Schéma regulačního obvodu s pomocnou regulovanou veličinou[7]

1.5.2 Regulační obvod s pomocnou akční veličinou

Základní blokové schéma systému je znázorněno na obrázku 1.7. Systém se skládá ze dvou bloků S_1 a S_2 zapojených sériově. Hlavní regulátor R_1 ovlivňuje systém pomocí akční veličiny x_1, zatímco vedlejší regulátor R_2 řídí pomocnou akční veličinu x_2. Změny této pomocné akční veličiny jsou přenášeny na výstup systému y rychleji než změny hlavní akční veličiny x_1. [7]

Pro zavedení této pomocné vazby je nezbytné mít možnost ovlivňovat systém minimálně dvěma akčními veličinami. Přitom přenosová charakteristika akčních veličin na výstup musí být různá, nebo se alespoň musí lišit časové konstanty obou přenosů. To je důležité s ohledem na to, že změny v jedné akční veličině musí být přenášeny na regulovanou veličinu rychleji než změny v druhé akční veličině. [8]

Hlavní regulátory v těchto rozvětvených obvodech jsou voleny jako typ I nebo PI, aby byly minimalizovány ustálené odchylky. Pomocné regulátory jsou obvykle typu PD s ohledem na co nejrychlejší průběh přechodného děje v pomocné regulační smyčce. Tento typ rozvětveného regulačního obvodu je velmi často používán v regulaci chemických destilačních procesů. [7]

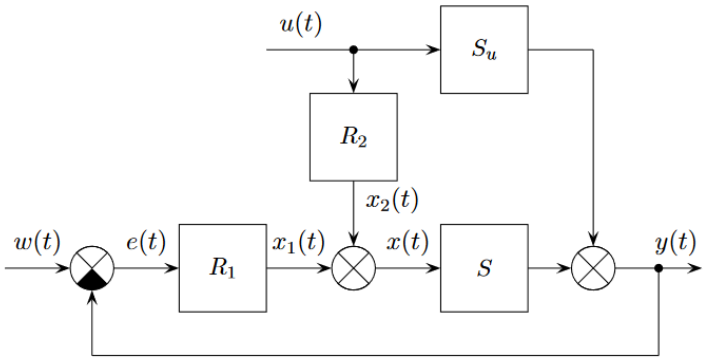


Obrázek 1.7 Schéma regulačního obvodu s pomocnou akční veličinou[7]

1.5.3 Regulační obvod s měřením poruchy

Základní blokové schéma systému je znázorněno na obrázku 1.8. Porucha u(t) projde blokem s přenosem $S_u(p)$ a přičte se k výstupu řízené soustavy. Současně tuto poruchu měříme a přes regulátor R_2 ji přičteme k akční veličině $x_1(t)$. Přenos řízení zůstává touto dodatečnou vazbou nezměněný. [7]

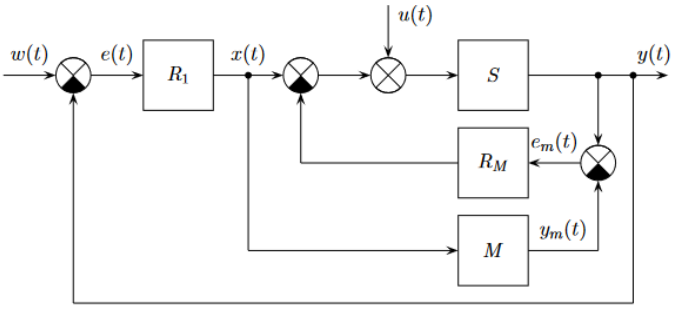
Systémy s měřením poruch jsou často používány při regulaci teploty ve velkých objemech. Například při vytápění budov lze výrazně zlepšit regulaci měřením venkovní teploty, která je hlavním zdrojem poruchy. Podobně měření napájecího napětí, teploty a tlaku vyhřívacího média nebo kvality topného materiálu obvykle zvyšuje kvalitu regulace. [7]



Obrázek 1.8 Schéma regulačního obvodu s měřením poruchy[7]

1.5.4 Regulační obvod s modelem regulované soustavy

Tyto systémy se využívají převážně v adaptivních obvodech, lze je ovšem využít i v jednoduchých regulačních obvodech a zlepšit tím jejich kvalitu. Speciálním případem tohoto typu rozvětveného regulačního obvodu je obvod, pro kompenzaci dopravního zpoždění, přítomného v regulované soustavě. Praktická realizace modelu a dopravního zpoždění ve spojité variantě by byla velmi nákladná. Z toho důvodu se při realizaci používá diskrétní model s diskrétním regulátorem. [7]



Obrázek 1.9 Schéma regulačního obvodu s modelem regulované soustavy[7]

2. SIMULACE VÝROBNÍCH PROCESŮ

Simulátory výrobních procesů jsou nástroje nebo software, které umožňují modelovat, simulovat a analyzovat různé aspekty výrobního procesu. Tyto simulace jsou navrženy tak, aby pomohly firmám, vývojářům a inženýrům lépe porozumět procesům výroby a optimalizovat je. Na trhu se nachází nepřeberné množství takovýchto nástrojů ať už se jedná o placené (Siemens), tak volně stažitelné (Modelica). Tato kapitola obsahuje jejich řešerši a dále zde byl proveden pokus o propojení modelovacích nástrojů Unity3D a matlabu a změření doby jejich odezvy včetně výpočtu nejistot měření.

2.1 Motivace k zavedení simulátorů výroby

Důvodů pro zavádění simulace a modelování do výrobního podniku je mnoho. Mezi hlavní patří možnost modelovat budoucí výrobu nebo identifikovat příčiny problémů ve stávající výrobě. To umožňuje získat přehled o slabých místech ve výrobních procesech, zatížení jednotlivých strojů a pracovníků.

Dále také může také sloužit k školení zaměstnanců a fungovat jako trenažér. Existují případy, kdy na základě správně navrženého modelu lze vytvořit další komponenty, jako je například řídicí systém. Modelovací nástroj zde pak slouží jako emulátor. Důležitým faktorem je časový aspekt. Modelování může být provedeno relativně rychle, přičemž doba závisí především na uživatelových zkušenostech a na složitosti modelovaného systému. Čím přesnější model chceme vytvořit, tím více času musíme věnovat jeho návrhu. [9]

V neposlední řadě je určitě potřeba zmínit ekonomickou motivaci k zavedení do výroby. Podniky investují nemalé částky do těchto simulačních nástrojů, jelikož si díky nim dokážou správně navrhnout pracoviště a odhalit případné úskalí svých řešení, což díky tzv. pravidlu deseti je stojí 10krát méně, než kdyby na tento problém narazili při konečné implementaci řešení. [10]

2.2 UNITY 3D

Unity 3D je vývojové prostředí a herní engine, který umožňuje vytvářet 2D a 3D hry, simulace, vizualizace a interaktivní aplikace. Tento engine ovšem nemusí sloužit pouze k vývoji her. Základní aplikace umožňuje tvorbu 3D modelů a jejich následné rozpohybování za pomocí C# scriptů, které lze přidružit k libovolnému objektu. Dále lze k jednotlivým objektům přiřadit jejich fyzikální vlastnosti. K těmto scriptům můžeme přidat i jednoduchý script pro komunikaci s externí aplikací či nástrojem (v případě této práce probíhala komunikace s výpočetní aplikací Matlab). Těchto vlastností lze využít i v průmyslu. Pro toto odvětví představila společnost Unity tzv. Unity Industry speciálně pro průmyslové aplikace.

Unity Industry umožňuje vývojářům a inženýrům napříč průmyslovými odvětvími, vytvářet a poskytovat vlastní 3D zážitky v reálném čase pro rozšířenou realitu (AR), virtuální realitu (VR), mobilní zařízení, stolní počítače a web. Obsahuje i pluginy speciálně určené pro modelování digitálních dvojčat. Jedná se ovšem o placenou verzi. [11]

Mezi obecnou nevýhodu lze zařadit nemožnost vytvářet vlastní komplexní modely, nástroj obsahuje pouze jednoduché modelovací prostředí, které obsahuje modely základních tvarů, ze kterých lze sestavit jednoduché objekty. Placená verze ovšem i tyto nedostatky maže.

2.3 ABB Robot Studio

Společnost ABB přinesla na trh software nazvaný RobotStudio, který nabízí širokou škálu nástrojů pro uživatele průmyslových robotů. Tento program umožňuje vytvářet detailní simulace v 3D prostředí. Firmy, které již vlastní zařízení od ABB, ocení přidanou hodnotu tohoto nástroje. Díky knihovnám s roboty této společnosti, které jsou součástí RobotStudio, je proces vytváření digitálních dvojčat značně zjednodušen.

Mezi hlavní funkce a vlastnosti ABB RobotStudio dle [12] patří:

- Simulace robotických operací: Umožňuje uživatelům simulovat pohyby a operace průmyslových robotů v 3D prostředí. To umožňuje přesné plánování a optimalizaci výrobních procesů.
- Programování: RobotStudio poskytuje prostředí pro programování robotů pomocí různých programovacích jazyků nebo grafických uživatelských rozhraní.
- Vytváření digitálních dvojčat: Umožňuje vytvářet digitální modely reálných zařízení a výrobních linek, což umožňuje simulovat a testovat robotické operace v přesném prostředí.
- Optimalizace procesů: Pomocí RobotStudio mohou uživatelé optimalizovat výrobní procesy a pracovní postupy před jejich nasazením v reálném provozu.
- Integrace s ABB roboty: Software je plně integrován s průmyslovými roboty od společnosti ABB, což umožňuje snadnou synchronizaci mezi digitálními simulacemi a reálnými robotickými operacemi.

2.3.1 Shrnutí

RobotStudio patří mezi velmi propracované nástroje, ovšem je velmi úzce zaměřený na roboty společnosti ABB a jejich programování. Do scény sice lze exportovat CAD/CAM objekty, ale tím můžeme pouze dotvořit scénu kolem robotů. Mezi výhodu lze uvést komunikaci přes OPC UA, což je komunikační standard, který podporuje myšlenky průmyslu 4.0.

2.4 SIEMENS Simatic Simit

Siemens SIMATIC SIMIT je simulační platforma vyvinutá společností Siemens pro testování a ověřování automatizačních systémů a řídicích procesů. Tato platforma je určena pro průmyslové aplikace a umožňuje inženýrům a technikům simulovat a testovat chování jejich řídicích systémů a automatizačních procesů před jejich nasazením v reálném provozu.

Jednou z možností využití Simit Simulation Platform jsou aplikace pro virtuální uvádění do provozu a školení operátorů. Simit umožňuje simulace chování různých vstupů či výstupů, až po simulace složitých procesů. Simit umožňuje testovat automatizační software pomocí skutečného hardwaru (tzv. HIL – Hardware in loop), nebo pomocí emulovaného softwaru. (tzv. SIL – Software in loop). [13]

Další výhodou této aplikace je spojení s dalšími možnými Siemens softwarem jako jsou například Siemens Mechatronic Concept Designer, ve kterém lze jak modelovat jednoduchý model, tak tvořit celou sestavu a modelovat její fyzikální vlastnosti.

2.5 MATLAB

MATLAB je softwarový produkt vyvinutý firmou MathWorks, který poskytuje výkonné nástroje pro matematické výpočty, modelování, simulaci, analýzu dat a vizualizaci. Název MATLAB pochází z anglického "MATrix LABoratory" (matematická laboratoř), což odráží jeho původní zaměření na práci s maticemi a lineární algebrou.

Matlab by byl v této aplikaci využit jako emulátor systému, který bude zastřešen pomocí AAS.

2.5.1 Způsoby komunikace

Jelikož Matlab patří mezi hojně rozšířené výpočetní aplikace je k dispozici spoustu knihoven pro různé komunikační protokoly. V této práci se bude využívat MQTT komunikace. Tento způsob komunikace obsahuje Indistrial Communication Toolbox. Obsahuje mimo jiné i Modbus komunikaci, či OPC standard.

2.6 MĚŘENÍ DOBY ODEZVY MEZI MATLABEM A UNITY 3D

Tato kapitola popisuje způsob komunikace a způsob měření doby komunikace. Dále jsou zde uvedeny výpočty a grafické znázornění výsledků.

2.6.1 Komunikace

Unity 3D nabízí možnost komunikace s matlabem přes TCP/IP protokol. Jedná se o typ komunikace Klient-Server. Kde jako server vystupuje Unity. Matlab jakožto klient se

k němu připojuje. Pro vytvoření serveru je v Unity nutné vytvořit script v programovacím jazyce C#. Využijí se již existující knihovny System.net.socket. Tento script je obsažen v příloze pod názvem ServerUnity.cs.

V Matlabu, který vystupuje jakožto klient a připojuje se na server, byl napsán jednoduchý script pro připojení a posílání zpráv. Opět bylo využito již existující rozšíření matlabu o TCP/IP komunikaci. Dále zde bylo využito funkce tic toc, za pomocí které byl změřen čas mezi dotazem a jeho odpovědí. Pro spolehlivější komunikaci, bylo nutné přidat při čekání na odpověď 10 ms pauzu. Tato skutečnost nám samozřejmě značně zasahuje do výsledků měření a zpomaluje celou komunikaci. Matlab script je uveden v příloze pod názvem ClientMatlab.m.

Stanice na které bylo provedeno měření

Měření bylo provedeno na notebooku Dell Inspiron 15 5000. Níže jsou uvedeny podrobnější specifikace zařízení.

CPU: Intel Core i5 1,6GHz, 8. generace
RAM: 8GB
GPU: NVIDIA GeForce MX130 + interní Intel UHD Graphics 620
OS: 64 bit Windows 10 22H2

2.6.2 Výsledky měření

Měření rychlosti komunikace mezi prostředím Unity a Matlab bylo provedeno ve třech variantách a to pro 1 000, 10 000 a 20 000 pokusů. Počet pokusů lze měnit ve *For* smyčce znázorněné ve scriptu výše. Výsledky byly z matlabu exportovány do *.csv* dokumentu. S tímto dokumentem se dále prováděli výpočty za pomocí nástroje LabView.

Nejdříve byl spočten prvotní odhad (průměr) a poté byla za jeho pomocí spočtena nejistota typu A. Vzorce, které byly využity pro výpočet jsou uvedeny níže.

Výpočet prvotního odhadu:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \,, \tag{2.1}$$

kde n je počet provedených měření.

Výpočet nejistoty typu A:

$$u_a = s_{\bar{x}} = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n(n-1)}}\,, \tag{2.2}$$

kde n je počet provedených měření a $\bar{x}$ je prvotní odhad, vypočtený ze vzorce (2.1). Jelikož se jedná o vyšší počet měření, než 9, nemusíme nejistotu korigovat o koeficient ks. Výpočet nejistoty typu B se v tomto případě nebude uplatňovat. Jako měřící přístroj byl použit software Matlab.

Výpočet rozšířené standardní nejistoty:

Jelikož je nejistota typu B nulová, celková kombinovaná nejistota je proto rovna nejistotě typu A.

$$u_a = u_c \quad (2.3)$$

Rozšířená standardní nejistota U s rozšiřovacím koeficientem rozšíření k_r, je pravděpodobnost, že v daném intervalu se nachází výsledek měření. Koeficient k_r se volí dle tabulky 2.1. Ve všech provedených výpočtech je počítáno s koeficientem rozšíření $k_r = 2$. Tedy že daný výsledek se nachází z 95 % v intervalu. Vzorec pro výpočet standardní rozšířené nejistoty odpovídá rovnici (2.4).

$$U = u_c * k_r \quad (2.4)$$

Posledním krokem byl výpočet úspěšnosti komunikace. Tento výsledek je určen jako podíl počtu úspěšných pokusů ku počtu pokusů. Výsledek je poté udáván v procentech.

$$V = \frac{n_V}{p} * 100\ [\%]\ , \quad (2.5)$$

Kde p je celkový počet pokusů o spojení a n_v je počet úspěšných pokusů.

Tabulka 2.1 Volba koeficientu k_r na základě pravděpodobnosti

k_r	1	2	3	4
p [%]	68	95	99	99,7

2.6.3 Výsledky měření pro 1 000 pokusů

Pro přehlednost jsou zde uvedeny pouze grafické výsledky. Pro úplnost jsou zde i uvedeny hodnoty nejistoty typu A, či procentuální úspěšnost komunikace. Všechny tyto výpočty jsou v příloze v LabView projektu s názvem Vyhodnoceni.

Nejistota typu A byla vypočtena, dle vzorců (2.1) a (2.2), byla rovna u_a= 0,000525771 s. Rozšířená standardní nejistota vypočtena dle vzorce (2.4) byla rovna U = 0,001052 s. Průměr byl určen dle vzorce (2.1) a je roven $\bar{x}$ = 0,03448 s. Výsledky je samozřejmě nutno zaokrouhlit. Grafické výsledky jsou znázorněny na obrázku v příloze A.1.

Úspěšnost komunikace byla určena dle vzorce (2.5). Výsledek pro 1 000 pokusů byla V_{1000} = 93,7 %. Všechny vypočtené a zaokrouhlené hodnoty jsou pro přehlednost uvedeny v tabulce 2.2.

2.6.4 Výsledky měření pro 10 000 pokusů

Nejistota byla vypočtena, dle vzorců (2.1) a (2.2), byla rovna u_a= 0,0000467011 s. Rozšířená standardní nejistota vypočtena dle vzorce (2.4) byla rovna U = 0,000093 s. Průměr byl určen dle vzorce (2.1) a je roven $\bar{x}$ = 0,02825 s. Výsledky je samozřejmě nutno zaokrouhlit. Grafické výsledky jsou znázorněny na obrázku v příloze A.2. Úspěšnost komunikace byla určena dle vzorce (2.5). Výsledek pro 10 000 pokusů byla V_{10000} = 98,21 %. Všechny vypočtené a zaokrouhlené hodnoty jsou pro přehlednost uvedeny v tabulce 2.2.

2.6.5 Výsledky měření pro 20 000 pokusů

Nejistota byla vypočtena, dle vzorců (2.1) a (2.2), byla rovna u_a= 0,00010288 s. Rozšířená standardní nejistota vypočtena dle vzorce (2.4) byla rovna U = 0,00021 s. Průměr byl určen dle vzorce (2.1) a je roven $\bar{x}$ = 0,03354 s. Výsledky je samozřejmě nutno zaokrouhlit. Grafické výsledky jsou znázorněny na obrázku v příloze A.3.

Úspěšnost komunikace byla určena dle vzorce (2.5). Výsledek pro 20 000 pokusů byla V_{20000} = 85,9 %. Všechny vypočtené a zaokrouhlené hodnoty jsou pro přehlednost uvedeny v tabulce 2.2.

2.6.6 Shrnutí výsledků měření

Tato kapitola se věnuje způsobu komunikace, mezi programovým prostředí Matlab a Unity. Z grafů znázorněných na obrázcích v Příloha A -je viditelná špička (maximum) ihned při první pokusu o spojení. Tato špička je patrně způsobena připojováním klienta (Matlabu) k serveru (Unity). Po připojení se již doba komunikace ustálí a pohybuje se v rozmezí vypočteného průměru. Celkový průměr všech měření je 32 ms. A celková úspěšnost je 92,6 %. Pro přehlednost jsou všechny hodnoty uvedeny v tabulce 2.2.

Tabulka 2.2 Shrnuté výsledky měření

Počet pokusů	Úspěšnost [%]	Průměr [s]	Nejistota A [s]	Rozšířená nejistota A [s]	Zaokrouhlení [s]	Konečný výsledek [s]
1000	93,7	0,0345	0,000525771	0,001052	0,0011	(0,0345±0,0011)
10000	98,21	0,028253	0,00004670	0,000093	0,000093	(0,028253±0,000093)
20000	85,9	0,03354	0,00010288	0,000206	0,00021	(0,03354±0,00021)
Průměr	92,60	0,032				

2.7 MODELICA a OpenModelica

Modelica je jazyk pro modelování kyberneticko-fyzikálních systémů, který podporuje akauzální spojení komponent řízených matematickými rovnicemi pro usnadnění modelování od prvních principů. Poskytuje objektově orientované konstrukce, které usnadňují opětovné použití modelů, a lze je pohodlně použít pro modelování složitých systémů obsahujících např. mechanické, elektrické, elektronické, magnetické, hydraulické, tepelné, řídicí, elektrické nebo procesně orientované dílčí komponenty.[14]

Modelica se oproti jiným modelovacím nástrojům vyznačuje těmito vlastnostmi:

- **Objektově orientované modelování** umožňuje vytvářet komponenty modelu, které jsou fyzicky relevantní a snadno použitelné. Tyto komponenty podporují hierarchické strukturování, opětovné použití a vývoj rozsáhlých a komplexních modelů, které pokrývají více technologických oblastí. [15]

- **Akauzální modelování** nahrazuje tradiční blokovou abstrakci příkazy přiřazení rovnicemi tzn. rovnice neznamenají přiřazení (tj. uložení výsledku výpočtu přiřazovaného příkazu do dané proměnné), ale definici vztahů mezi proměnnými. To umožňuje přímé použití rovnic a opakované použití komponent modelu, protože se přizpůsobují kontextu toku dat, ve kterém jsou využívány. Tato metoda zjednodušuje modelování a zlepšuje efektivitu simulace. [15]
- **Fyzické modelování** více domén umožňuje komponentám modelu odpovídat fyzickým objektům v reálném světě, což eliminuje potřebu konverze na signálové bloky. Pro aplikace v inženýrství je tak snadné kombinovat takové "fyzické" komponenty do simulací pomocí grafických editorů. [15]
- **Hybridní modelování** podporuje integrovaný způsob modelování spojitých i diskrétních aspektů systémů. Od verze Modelica 3.3 je také dostupná podpora pro modelování v diskrétním čase, což zvyšuje přesnost modelování a výkon simulace. [15]

2.7.1 Open Modelica

OpemModelica je open-source modelovací a simulační prostředí založené na jazyku Modelica určené pro průmyslové a akademické použití. Jeho dlouhodobý rozvoj podporuje nezisková organizace – Open Source Modelica Consortium (OSMC). Cílem projektu je vytvořit kompletní modelovací, kompilační a simulační prostředí Modelica založené na svobodném softwaru distribuovaném ve zdrojovém kódu a spustitelné podobě určené pro výzkum, výuku a průmyslové využití. OpenModelica je volně k dispozici a lze jej kombinovat s dalším softwarem. Aktuální verze OpenModelica podporuje většinu jazyka Modelica, včetně rovnic, algoritmů, zpracování událostí, funkcí a balíčků. [15]

2.7.2 Dostupné Modelica knihovny

Většina Modelica knihoven je volně dostupných na internetu a zdarma stažitelných. Ať už se jedná o knihovny, které jsou schopny komunikovat s vnějšími zařízeními, např. přes MQTT protokol, či knihovny do kterých spadají práce se soubory, nebo pro jiné speciální průmyslové odvětví. Jako nevýhodu volně dostupných knihoven se třeba uvést, jejich možnou nekompatibilitu s jinými verzemi OpenModelica, či jinými operačními systémy.

Existují ovšem i komerční knihovny, jako například od společnosti LTX Simulation nebo od společnosti Modelon. Tyto knihovny jsou určené pro speciální účely jako například pro letecký či chemický průmysl.

2.7.3 3D Vizualizace v OpenModelica

Software OpenModelica podporuje taktéž tvorbu vizualizace a 3D vizualizace simulovaných systémů. Tento nástroj je dostupný od verze 1.11 a nahradil tím pak dříve využívanou knihovnu Modelica3D. Vizualizace v OpenModelica je založena na nástroji OpenSceneGraph, což je opět open source řešení pro tvorbu grafických aplikací a vizualizacích. [16]

Standard jazyka Modelica obsahuje definice standardizovaných grafických anotací, které umožňují definovat 3D tvary fyzických objektů. Existují standardní anotace pro množství tvarů, jako jsou válce, tyče atd. Animace a vizualizace v OpenModelica jsou založeny na 3D tvarech definovaných knihovnou Modelica Multi-Body. Tato knihovna poskytuje vizualizaci výsledků simulace a animace geometrických primitiv a souborů CAD. [15]

2.7.4 Modelica FMI

FMI je nový standard pro výměnu modelů a společnou simulaci pomocí Functional Mockup Interface (FMI) je důležitým nástrojem v oblasti modelování a simulace. FMI umožňuje export modelů ve formě předkompilovaného kódu, jako například v C kódu nebo binárním kódu, z jednoho simulačního nástroje, aby mohl být importován do jiného nástroje a použit v různých simulačních prostředích. To znamená, že modely vytvořené v jednom nástroji mohou být snadno použity v jiných simulačních prostředích, což zvyšuje interoperabilitu a flexibilitu vývoje simulačních modelů. [16]

OpenModelica je jeden z těch simulačních nástrojů, který podporuje standard FMI. Podpora pro verze FMI 1.0 i 2.0 umožňuje uživatelům OpenModelica snadno importovat a exportovat modely do a z jiných simulačních prostředí, a tím rozšiřovat možnosti vývoje a využití simulačních modelů v různých projektech a aplikacích. [16]

2.7.5 Jednoduchá ukázka simulace v OpenModelicaEditoru

Pro ukázku byl zvolen jednoduchý model dvou propojených nádrží s různým objemem kapaliny a zároveň s rozdílnou teplotou kapaliny. Tato ukázka má sloužit hlavně k demonstraci toho, že OpenModelica zvládne odsimulovat zároveň více sledovaných proměnných.

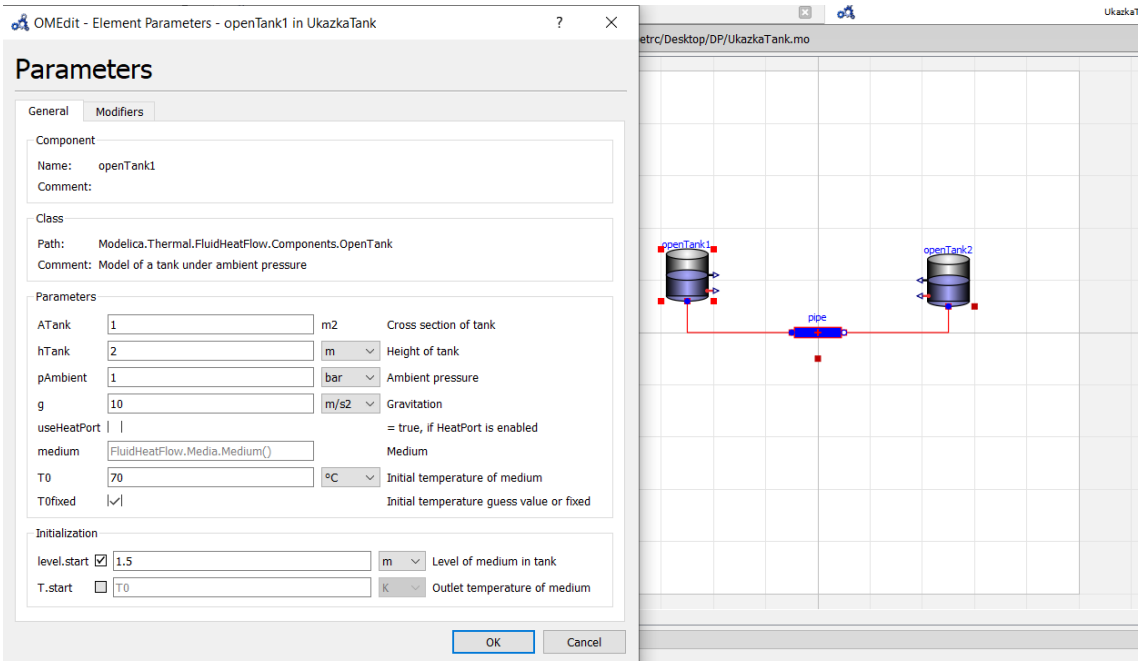
Na obrázku č. 2.1 je znázorněno grafické rozhraní tohoto simulačního editoru. Tanky lze jednoduše propojit. Dále je pak nutné zadat důležité parametry tanků jako jejich objem, množství media, které obsahuje, samotný druh media a v neposlední řadě teplotu media. Jelikož tato simulace bude využívat pouze samočinného pohybu kapaliny, je nutno zadat i gravitační zrychlení. Pro větší zpřesnění lze zadat ještě simulátor potrubí, tím se omezí rychlost proudění kapaliny. V této ukázce ovšem není využita.

Po zadání potřebných parametrů lze spustit simulaci. V základním nastavení simulace lze volit:

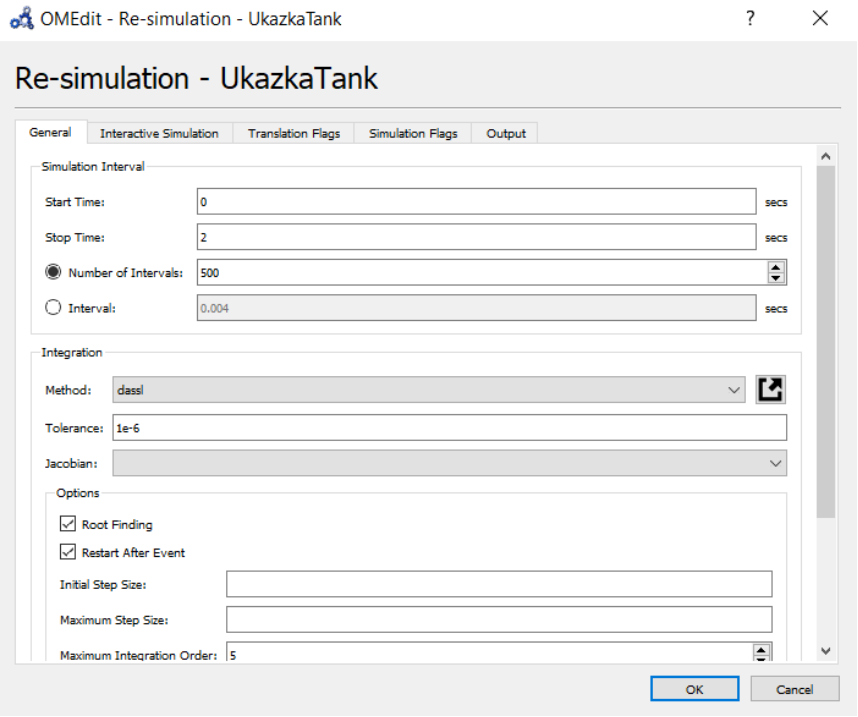
- čas simulace,
- vzorkování,
- toleranci,
- výpočetní metodu,
- interaktivní simulaci, která by byla schopna reagovat i na vnější události,
- a další parametry.

Panel nastavení simulace je znázorněn na obrázku 2.2.

Mezi nevýhody OpenModelica lze uvést nekompatibilitu některých knihoven s prostředím Windows. Například se problém vyskytl při práci s knihovnami pro MQTT a TCPIP. Celý vývoj dle informací od tvůrců probíhá v prostředí Linux, proto je potřeba na tyto informace dávat pozor.



Obrázek 2.1 Grafické rozhraní OpenModelicaEditoru



Obrázek 2.2 Ukázka možností simulace

2.8 Shrnutí

Na trhu se nachází velké množství modelovacích a simulačních nástrojů. Výhoda Unity je bezesporu v jeho grafickém prostředí, to ovšem v této konkrétní práci nebude potřeba, jelikož se jako model po konzultaci s vedoucím práce zvolila výměníková stanice. V tomto případě by nebylo využito nic z potenciálu Unity3D ani RobotStudia, které je spíše zaměřeno čistě na průmyslové aplikace od společnosti ABB.

Z výše uvedeného důvodu zůstaly v úvahu hlavně aplikace Simit, Matlab a OpenModelicaEditor. Jelikož Simit nepatří mezi open source aplikace, byl z rozhodování taktéž vyřazen.

Nakonec bylo rozhodnuto využít aplikaci OpenModelicaEditor, hlavně pro její volnou dostupnost, grafické rozhraní, přívětivé uživatelské rozhraní a dostupnost velkého množství knihoven pro různé instrumentace jak už v průmyslových oborech, tak i v dalších vědeckých disciplínách.

3. NÁVRH ŘEŠENÍ

V této kapitole je popsán vybraný demonstrační systém a následně vytvořen nástin řešení za pomocí blokového schématu. V poslední části kapitola obsahuje popis jednotlivých bloků řešení.

Jakožto demonstrační systém pro řízení za pomocí AAS byl zvolen demonstrační systém výměníkové stanice, který se nachází v laboratoři UAMT FEKT. Řízení této stanice se aktuálně provádí přes PLC Siemens. Výměníkové stanice se v průmyslu využívají pro přenos tepelné energie mezi různými medii. K tomuto účelu využívají výměníky tepla.

Ústřední komponentou této stanice je výměník tepla Alfa Laval CB14. Dále tento systém obsahuje dvě oběhová čerpadla Grundfos Alpha+ 25-40, tepelná a tlaková čidla neposlední řadě zdroj tepla bojler Ariston. Pro možnosti regulace jsou nejdůležitější třícestné ventily RV111 se servořízením. Pomocí nich dochází k regulaci teploty vody v primárním i sekundárním okruhu.

Obrázek 3.1 Výměníková stanice

3.1 Popis jednotlivých komponentů

V této podkapitole je uveden soupis a detailnější popis jednotlivých komponentů demonstračního systému (obr 3.1). Soupis není kompletní, jelikož jsou zde vynechány analogové snímače, jejichž hodnoty nelze převést do řídícího PLC.

3.1.1 Výměník tepla ALFA LAVAL CB14

Výměník tepla slouží obecně k předávání tepla mezi dvěma různými médii bez jejich vzájemného smíšení. To znamená, že teplonosná média mohou být oddělena fyzikální bariérou, ale přesto dochází k přenosu tepla z jednoho média do druhého. [17]

V demonstračním systému byl použit deskový výměník, to znamená, že je tvořen několika tenkými kovovými deskami, které jsou k sobě nerozebíratelně připájeny. Pájené výměníky tepla Alfa Laval CB přinášejí několik výhod ve srovnání s tradičními výměníky tepla, které jsou běžně používány v průmyslových aplikacích pro vytápění, chlazení a v chladírenství. Jejich kompaktní konstrukce a vyšší termická účinnost umožňují jejich využití i v prostorově omezených podmínkách. Tyto výměníky jsou schopny dosáhnout vysokých výkonů, což je ideální pro aplikace, kde je zapotřebí efektivní tepelné výměny. [18]

3.1.2 Ohřívač vody – Bojler Ariston Ti Shape 10UR

Jako zdroj tepla je v demonstrační stanici zvolen bojler Ariston Ti Shape 10UR. Tento bojler má výkon 2kW a objem 10 litrů. Dále je vybaven vlastním termostatem a přetlakovým ventilem. Maximální pracovní tlak je v katalogovém listu uveden 8 barů.

3.1.3 Oběhové čerpadlo Grundfos ALPHA+ 25-40 180

Pro zajištění cirkulace kapaliny systém obsahuje dvě oběhová čerpadla. Tato čerpadla jsou poháněna dvoupólovým asynchronním motorem s kotvou nakrátko a s odrušovacím filtrem dle normy VDE0875. Motor obsahuje i impedanční ochranu a nevyžaduje proto žádnou další externí motorovou ochranu. [19]

Čerpadlo může pracovat dle různých nastavení v různých pracovních křivkách. Přepínání mezi těmito režimy v této práci není využito, protože pro zjednodušení uvažujeme pouze konstantní rychlost proudění media.

3.1.4 Třícestný ventil

Nejdůležitějšími komponentami pro regulaci jsou třícestné ventily LBM RV 111 R. Tyto ventily jsou dále vybaveny servopohony Siemens SSC61 pro regulaci za pomocí PLC analogovým napěťovým výstupem 0-10 V.

3.1.5 Průtokoměr TCM 142/99-3047

Soustava je dále vybavena dvěma průtokoměry pro měření množství kapaliny, která proteče skrz výměník. Tyto průtokoměry jsou dále vybaveny impulsními výstupy, které lze odečítat a vyhodnocovat v PLC. Hodnoty průtoku jsou identifikovány na impulsy a přepočet je udáván na 10 litrů na impuls.

3.1.6 Odporový snímač teploty PTP05 s převodníkem

Tento odporový snímač určen pro měření teploty na povrchu potrubí. Pro měření se používá měděný výlisek, na kterém se měří teplota za pomocí odporového snímače Pt100. Tento typ má více možností pro rozhraní, ať už se jedná o klasické 4-20 mA, tak je zde možnost komunikace i například RS485 protokol ModBus RTU. V této práci je ovšem využito komunikace přes standardní rozhraní 4-20 mA.

3.1.7 Odporový snímač teploty PTP55 s převodníkem

Tento snímač je určen pro usazení do jímek a návarků pro kontaktní měření teploty. Tento teploměr obsahuje převodník, který převádí teplotu na standardní rozhraní 4–20 mA.

3.1.8 Expanzní nádoby

Demonstrační systém obsahuje také expanzní nádoby, které slouží k vyrovnání změny tlaků při měnící se teplotě média. V tomto konkrétním případě se jedná o expanzní nádoby CIMM ACS, které mají objem 5 litrů a maximální provozní tlak je udáván dle výrobce na 10 barů.

3.1.9 Snímač tlakové diference DMD331

Systém je dále vybaven snímačem tlakové diference BD Sensors DMD331. Základem snímače je piezorezistivní senzor z nerezové oceli, který může být oboustranně vystaven tlakům kapalin a plynů. Převádí rozdíl tlaků mezi pozitivním a negativním vstupem na standardní analogový výstupní signál 4-20 mA.

3.1.10 Řídící PLC Siemens

K řízení modelu výměníkové stanice byl využit panel s PLC Siemens Simatic S7-1500 typ 1512C-1 PN s dotykovým panelem Siemens TP 700, který slouží pro ovládání a zobrazení aktuálního stavu výměníkové stanice.

3.2 Popis principu fungování výměníkové stanice – aktuální stav

Primární okruh je navržen jako aktivní – obsahuje bojler jakožto zdroj ohřevu vody. Tu pak dále za pomocí PLC a ovládání třícestného ventilu regulujeme na požadovanou teplotu. Naopak sekundární okruh je navržen jakožto pasivní z pohledu tepelné energie.

Obsahuje totiž chladič, jímž je odebírána tepelná energie ze soustavy do okolního prostředí. Pro pochopení principu nejlépe vypovídá procesní schéma, které je uvedeno v příloze B.1**.** Příloha B -dále obsahuje zjednodušený popis využité instrumentace výměníkové stanice a účel každého prvku. Pro podrobnější technický popis a případná elektrická schémata lze využít literaturu [20].

Aktuálně lze model řídit čistě za pomocí PLC Siemens. Hlavní nevýhodou tohoto systému je to, že systém může nastavit pouze jednorázově. Případně lze využít dostupného HMI panelu. Nemůžeme v tomto systému využít např. dopředného zapnutí, jelikož nemáme informace, jak dlouho trvá daný děj, než se například podaří výstup nahřát na teplotu 50 °C.

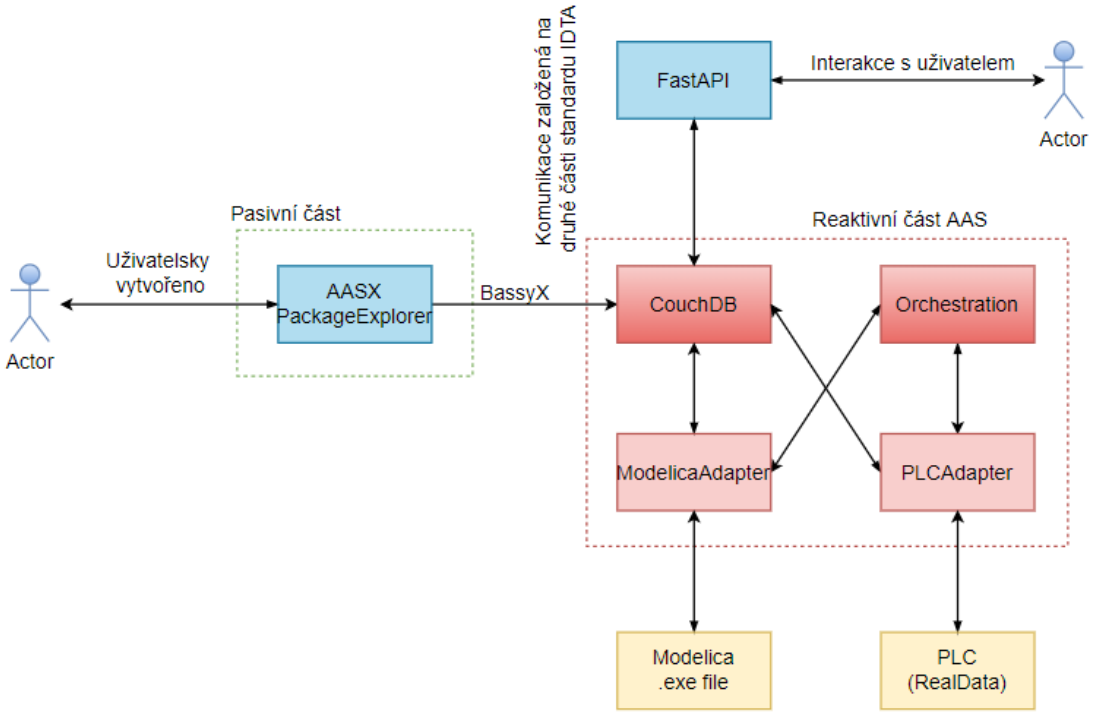
Hlavní myšlenkou tohoto demonstračního systému, řízeného za pomocí AAS je požadavek výroby (technologie) na dodaný tepelný výkon a tento výkon pak nějakým způsobem regulovat, aby zůstal konstantní i po připojení zátěže.

3.3 Blokové schéma řízení za pomocí AAS

Způsob řešení nastiňuje blokové schéma na obr. č. 3.2. V prvním kroku je vytvořena pasivní část AAS za pomocí aplikace AAS PackageExplorer. Pasivní část je poté uložena do AASX souboru. S tímto souborem je dále pracováno za pomocí adaptéru BasyxPythonSDK verze 0.2.2. Nejnovější verze bohužel stále není plně kompatibilní s nejnovějším AAS PackageExplorerem. Proto je nutno pracovat i s PackageExplorerem ve starší verzi, konkrétně verze 2021-08-18. Nekompatibilita některých verzí AASX souborů je poměrně častý problém, se kterým je možné se potýkat.

Uživatel si může vytvořit pasivní část AAS přes uživatelské rozhraní programu AASX PackageExplorer. Například lépe popsat dané komponenty výměníkové stanice. Poté, co uživatel vytvoří AASX soubor, se tento soubor za pomocí BasyxPython SDK nahraje do databáze CouchDB. Do této databáze lze přistupovat přes uživatelské rozhraní FastAPI. Příkazy pro komunikaci pak budou nasazeny dle standardu skupiny IDTA, konkrétně druhé části [21]. Po tom, co uživatel zadá příkaz (zvolená teplota, požadovaný výkon) odstartuje se cyklus, který je znázorněn na obrázku 3.3. A popsán v další podkapitole.

Pro získání reálných dat a zápisu do AAS slouží modul PLCAdapter. Pro zápis a čtení dat získaných ze simulace je využit blok ModelicaAdapter. Celou komunikaci mezi jednotlivými bloky a vyhodnocení výsledků simulace má pak na starost orchestrátor.

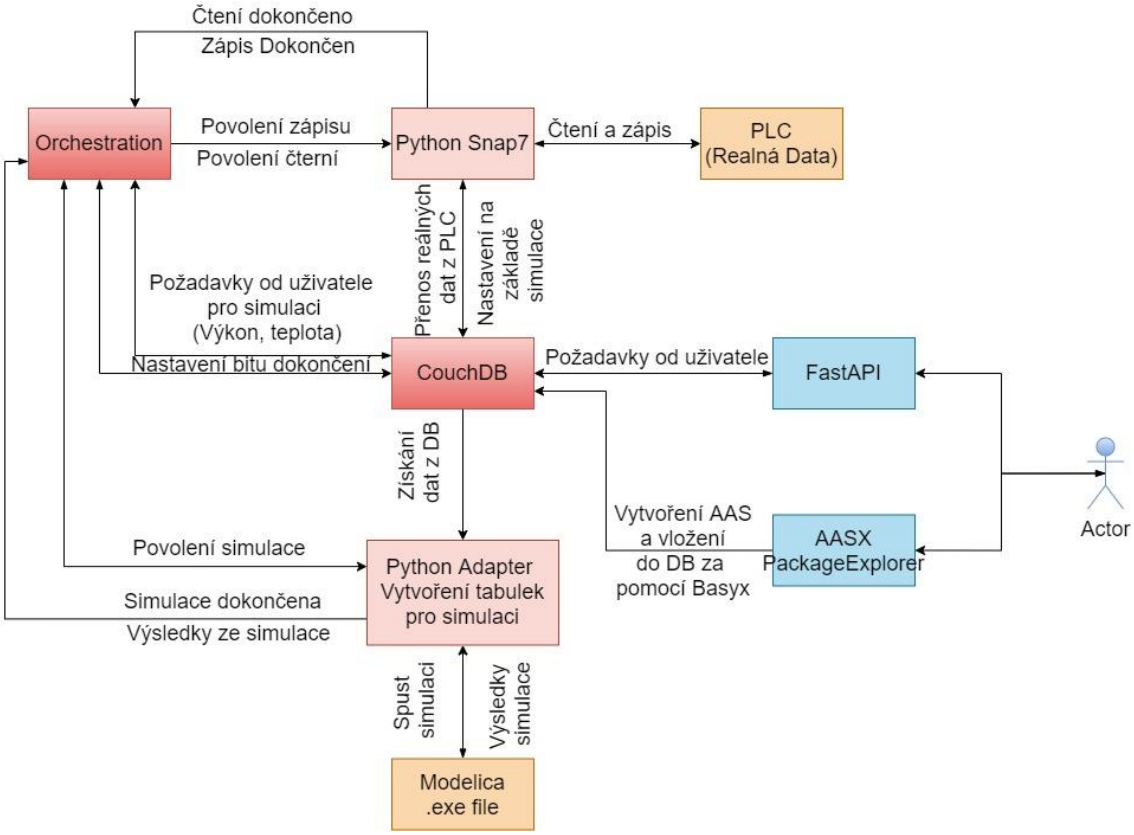


Obrázek 3.2 Blokové schéma řešení

3.3.1 Cyklus řízení

Cyklus začíná zadáním požadavku uživatele přes nástroj FastAPI. Další postup se už obejde bez kontaktu s uživatelem a je řízen orchestrátorem. Ten přijmutím požadavku od uživatele, dá pokyn pro načtení reálných dat a jejich následné uložení do databáze CouchDB. Poté, co je tento přenos dokončen, umožní získání dat z databáze adaptéru pro nastavení simulace. Tento adaptér ukládá průběžně výsledky jednotlivých scénářů simulací s různými nastavení akčních členů regulačního obvodu.

Orchestrátor dále vyhodnocuje nejlepší výsledky ze simulací, ty jsou pak uloženy do databáze AAS a následně je umožněno jejich načtení a zápis hodnot do PLC pro ovládání reálného systému. Celý cyklus je zobrazen na obrázku 3.3



Obrázek 3.3 Blokové schéma interakce bloků mezi sebou

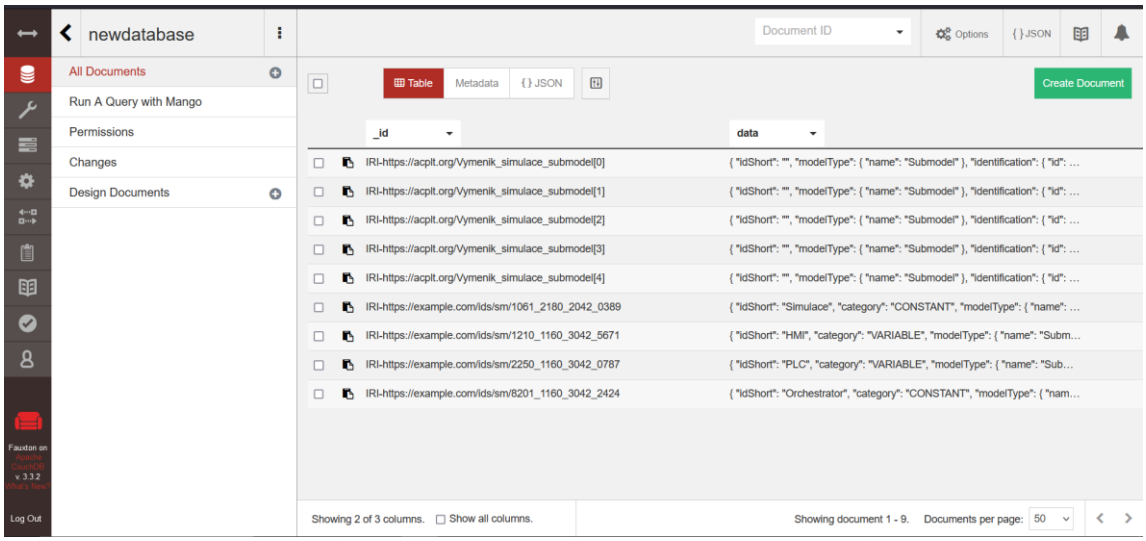
3.4 Databáze a AAS

Ústřední komponentou AAS je databáze. V tomto případě se jedná o CouchDB databázi. Tato databáze byla vybrána z důvodu implementace s Basyx knihovnou. S databází musí spolupracovat všechny části AAS.

Apache CouchDB je opensource NoSQL dokumentově orientovaný databázový systém. Díky tomu, že se jedná o opensource řešení, má správce nad softwarem větší kontrolu a zároveň poskytuje flexibilitu, která je nutná pro vytváření odolných a spolehlivých řešeních. CouchDB dále využívá REST API pro přístup k databázi odkudkoli s plnou flexibilitou operací CRUD (create, read, update, delete). Dále je možné tento nástroj instalovat nezávisle na operační systému PC. [22]

Mezi další výhodu CouchDB lze uvést její uživatelské rozhraní, které je dostupné přes webový prohlížeč. Při prvotním spuštění je nutné databázi nakonfigurovat, zvolit přihlašovací údaje. Role pro další uživatele už lze následně nastavovat ve webovém rozhraní. Těmito zásahy můžeme omezit přístup pro některé uživatele AAS a tím pádem zvýšit robustnost a bezpečnost celého systému.

Do databáze se ukládají všechna data z výsledků a tyto data jsou následně dále zpracovávána. Dále databáze slouží k interakci s koncovým uživatelem. S uživatelem je propojena přes API.



Obrázek 3.4 Ukázka uživatelského rozhraní CouchDB

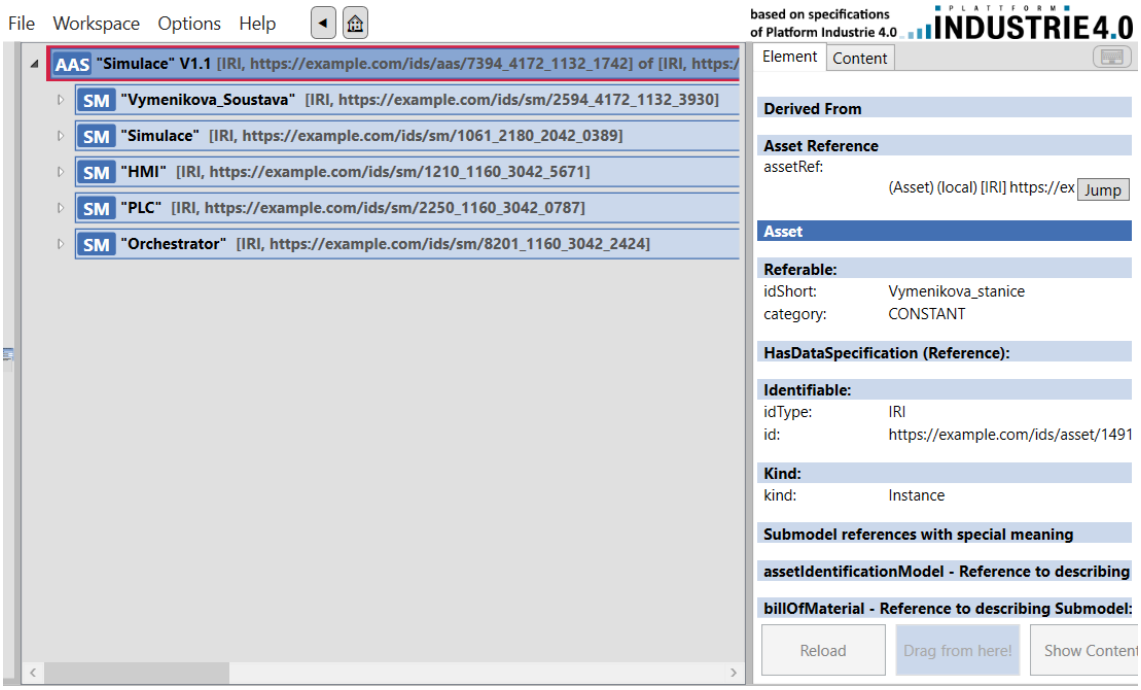
3.5 AASX Package Explorer

K práci s AAS byl využit nástroj AASX PackageExplorer. Tento nástroj slouží k prohlížení existujících AAS (například pro různá zařízení), ale také umožňuje vytvářet vlastní AAS. Další výhodou tohoto programu je možnost vytvoření OPC UA serveru z prohlíženého AAS. Ačkoliv to není jediný podporovaný komunikační standard, umožňuje také vytvoření MQTT serveru a dalších. Tento nástroj rovněž umožňuje exportovat formát *.aasx* do jiných formátů, jako jsou například *.json* soubory.

Pro práci byla použita verze 2021-08-18, Jelikož se jednalo o poslední verzi, která byla kompatibilní s Basyx knihovnou.

3.5.1 Vytvoření AAS za pomocí PackageExploreru

Pro vytvoření kostry celého systému AAS byl využit výše zmíněný nástroj. Prvním krokem je nutno vytvořit AAS, následně přidáme Asset. Jakmile jsou tyto operace dokončeny, lze již tvořit jednotlivé submodely se jejich vlastnostmi. Tyto jednotlivé submodely jsou následně za pomocí Basyx knihovny implementovány do databáze.



Obrázek 3.5 Ukázka uživatelského rozhraní AAS Package exploreru

3.6 Eclipse BaSyx Python SDK

Jedná se o sadu nástrojů, knihoven a frameworků vyvinutých v jazyce Python pro implementaci konceptu Asset Administration Shell v rámci projektu Eclipse BaSyx. Tato SDK umožňuje vývojářům pracovat s AAS v Pythonu a integrovat je do svých průmyslových aplikací a prostředí. V této práci byla využita verze 0.2.2, jelikož novější verze nebyla v době práce plně kompatibilní s AASX soubory.

BaSyx poskytuje nezbytnou infrastrukturu pro vytváření AAS konkrétněji:

- Knihovny pro práci s AAS: SDK poskytuje sadu knihoven pro práci s různými částmi AAS, jako jsou vlastnosti, funkce, vztahy a metadata.[23]
- Framework pro vytváření AAS: SDK obsahuje framework pro vytváření a správu AAS v rámci Python aplikací. Tento framework umožňuje vytváření, modifikaci a správu digitálních modelů objektů a procesů. [23]
- Integrace do průmyslových systémů: SDK poskytuje rozhraní pro integraci AAS do existujících průmyslových systémů a prostředí, což umožňuje propojení a interoperabilitu různých průmyslových zařízení a aplikací. [23]

3.7 Snap7

Snap7 Python knihovna je rozhraní pro programovací jazyk Python, které umožňuje komunikaci s průmyslovými zařízeními, zejména s PLC (programovatelnými logickými kontroléry) využívajícími protokol Siemens S7.

Mezi hlavní vlastnosti patří především:

- Komunikace s PLC: Knihovna umožňuje navázání spojení s PLC a komunikaci s nimi pomocí protokolu S7. To zahrnuje čtení a zápis dat do paměti PLC, získávání informací o zařízení a řízení provozu PLC. [24]
- Snadné použití: Díky svému designu pro jazyk Python je knihovna snadno integrovatelná do existujících Python aplikací a nabízí uživatelsky přívětivé rozhraní pro manipulaci s PLC a daty. [24]
- Multiplatformní podpora: Knihovna je navržena tak, aby byla kompatibilní s různými operačními systémy, včetně Windows, Linux a macOS. [24]
- Flexibilita: Snap7 Python knihovna poskytuje vývojářům flexibilitu při práci s daty v PLC a umožňuje jim je integrovat do svých aplikací podle potřeb. [24]

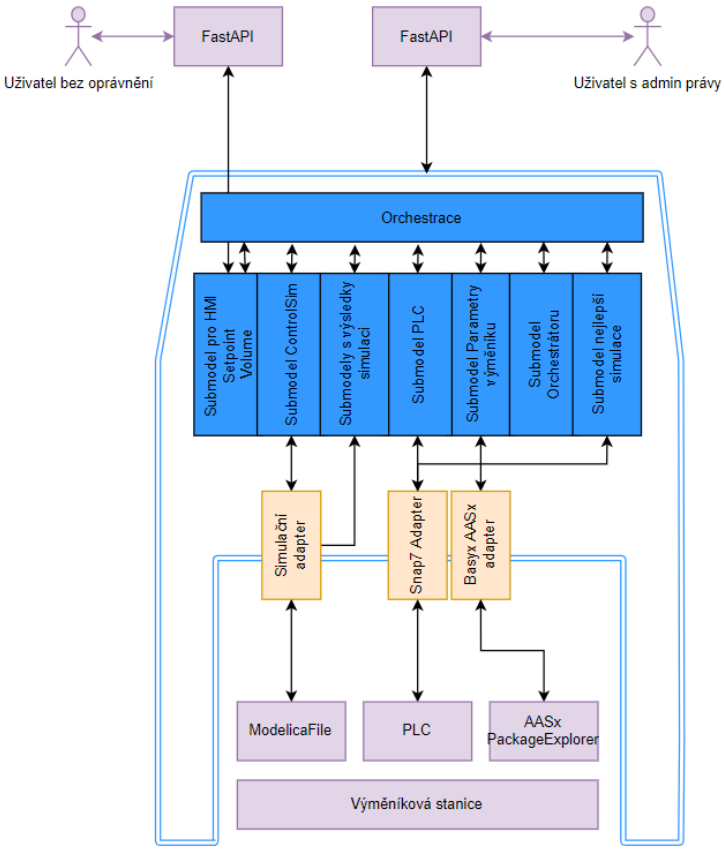
3.8 Shrnutí návrhu

Tato kapitola obsahuje popis zvoleného demonstračního systému se základními technickými specifikacemi komponentů, jež jsou součástí výměníkové stanice. Dále pro lepší pochopení principu celé stanice a pro lepší popis, bylo vytvořeno procesní schéma, které je obsaženo v příloze B.1. Příloha dále obsahuje soupis všech prvků, které obsahuje výměníková stanice.

Dále zde bylo vytvořeno blokové schéma s možným řešením systému s využitím AAS a následný popis cyklu řízení. V neposlední řadě jsou zde popsány jednotlivé části systému, které budou využity při realizaci této práce.

4. Realizace Systému

Tato kapitola popisuje realizaci jednotlivých kroků pro vytvoření jednotlivých submodelů, které jsou následně použity v AAS. Celá realizace se řídí dle návrhu, který je znázorněn na obrázku 4.1.



Obrázek 4.1 Struktura AAS

Každému znázorněnému adaptéru odpovídá jeden soubor s Python kódem – třídou, který zajišťuje komunikaci mezi submodelem a assetem. V tomto případě se AAS zastřešuje více než jednu položku a celý asset zde tvoří několik prvků a to:

- Výměníková stanice,
- PLC pro komunikaci se stanicí,
- Soubor obsahující model reálného systému.

Pro interakci se systém by mělo sloužit API, přes které lze komunikovat s vnitřní strukturou AAS. Pro interakci s uživatelem, který nemá žádné oprávnění by dle návrhu měl sloužit submodel s názvem HMI. V tomto submodelu jsou obsaženy údaje, které nemusí být nijak více zabezpečeny.

Pro interakci s celým AAS by měl mít přístup pouze oprávněný uživatel s administrátorskými právy. Tento uživatel by měl přístup do všech submodelů ve vnitřní struktuře AAS.

4.1 Databáze CouchDB

Před popisem jednotlivých submodelů, je třeba zmínit, že hlavní složkou celého AAS je jeho databáze a komunikace – ukládání dat do ní. Je potřeba v každém submodelu vyhradit jeho databázový prostor, tzn. zvolit správné přihlašovací údaje pro připojení. V případě, že by bylo potřeba AAS doplnit o další submodel, který by zajišťoval další funkcionality je potřeba se zamyslet, zdali musí mít tyto funkcionality přístup do celé databáze, nebo jen do nějaké její části. Tímto způsobem lze zajistit škálovatelnost a větší robustnost systému.

V této práci pro demonstraci jsou všechny submodely ukládány v jedné části databáze a všechny mají administrátorský přístup. Tudíž jedním přihlášením lze přistoupit do všech submodelů. Pro konfiguraci připojení přes python je nutno nainstalovat knihovnu *couchdb*. Následně lze nakonfigurovat přístupové údaje pro připojení do konkrétní databáze, jak ukazuje obrázek níže.

```
couchserver = couchdb.Server("http://localhost:5984/")
user = "admin"
password = "admin"
couchserver = couchdb.Server("http://%s:%s@localhost:5984/" % (user, password))

dbname = "newdatabase"
```

Obrázek 4.2 Konfigurace připojení ke CouchDB

4.1.1 Převod submodelů z PackageExploreru do databáze

Pro převod jednotlivých submodelů do databáze slouží soubor *DBconf.py*. V tomto souboru je nastavena konfigurace databáze. Provedení načtení výstupního souboru z AASX PackageExploreru (přípona *.aasx*) za pomocí BasyX knihovny. Je třeba opět zdůraznit, že tato funkce funguje pouze ve verzi v2021-08-18 PackageExploreru, jelikož je potřeba využít IRI identifikátoru a ty v novější verzi nejsou editovatelné.

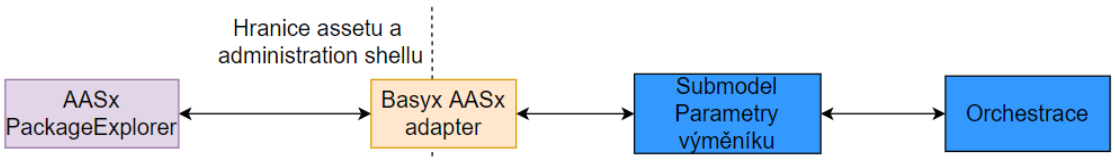
V tomto souboru je buď nutno postupně měnit identifikátory všech submodelů, aby se postupně nahrály do DB, nebo vytvořit nějaký cyklus pro nahrání všech zároveň. Pro testovací účely byl nahráván každý submodel zvlášť, jelikož se vždy při testování provedla nepatrná změna a bylo nutno následně soubor opět nahrát do databáze. BasyX z *aasx* souboru vytvoří dokument a ten je následně nahrán do databáze. Dále už se pracuje a komunikuje pouze s databází a dokumenty v ní. Tento převod lze tedy považovat za inicializační krok celé struktury AAS, než je s ní dále pracováno.

4.2 Submodel výměníkové stanice

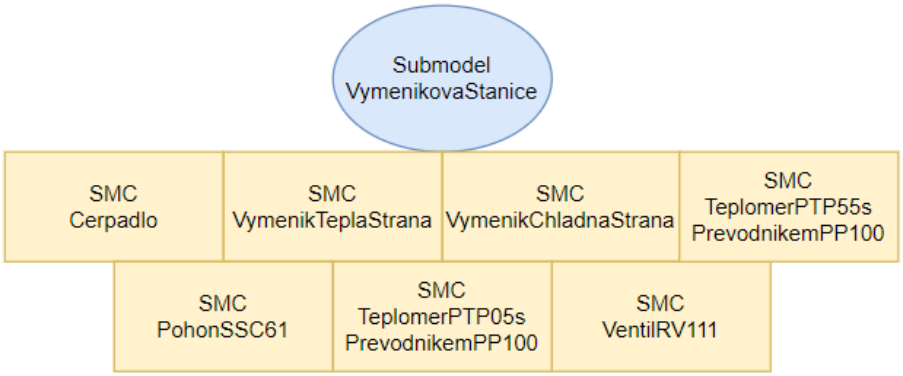
Tento submodel byl vytvořen v programu AASX Package Explorer ve verzi v2021-08-18. Tato informace je důležitá. V novějších verzích není v tomto programu možno editovat identifikátory IRI a IRDI. Na základě těchto identifikátorů se poté se submodely pracuje pomocí Basyx Python SDK. Pomocí tohoto rozhraní je submodel

následně nahrán do databáze CouchDB.

Submodel obsahuje základní informace o výměníkové stanici. Každá komponenta je zde popsána v jednotlivých podsubmodelech. Jsou zde vlastnosti od popisu po provozní podmínky. Tyto údaje byly získány z katalogových listů jednotlivých komponent výměníkové stanice. Mezi hlavní zdroj těchto informací lze uvést shrnutí v předchozí práci, která se zabývala touto výměníkovou stanicí ([20]). Tento submodel lze považovat za pasivní část AAS, jelikož slouží čistě k popisu demonstračního systému. Jednotlivé podsubmodely jsou uvedeny níže na obr 4.4**.**



Obrázek 4.3 Struktura interakce AAS s jednotlivými bloky a *.aasx* souborem



Obrázek 4.4 Struktura submodelu Výměníkové stanice

4.2.1 Shrnutí

Submodel výměníková stanice slouží výhradně pro popis komponentů, které obsahuje. Tento submodel si můžeme představit, jako katalogový list s danými komponentami obsahující informace a provozní veličiny. Celkem se stanice sestává z více než těchto sedmi komponentů. Pro úplnost by bylo možné doplnit například o parametry potrubí.

4.3 Submodel PLC

Tato podkapitola popisuje tvorbu submodelu pro komunikaci s PLC a jeho strukturu a dále popisuje měření na reálném systému.

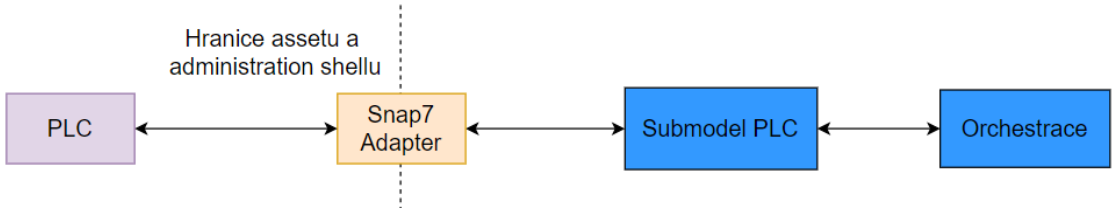
4.3.1 Měření na reálném systému

Pro porovnání dynamiky reálného systému a modelu v OpenModelica bylo provedeno měření na fyzické výměníkové stanici. Toto měření bylo provedeno za pomocí PLC Siemens Simatic S7-1500 typ 1512C-1 PN. Tento systém je popsán v předcházející podkapitole.

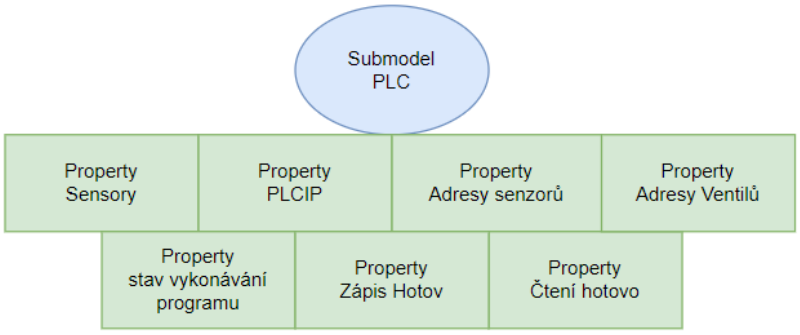
Pro měření bylo využito modulu traces v prostředí TIA Portal V17. Byl sestaven jednoduchý program, který nastavil polohu třícestných ventilů na 100% otevření. A dále vyčtení analogových hodnot na dostupných teplotních snímačích. Tato data byla následně za pomocí bloku Scale přepočtena na inženýrské jednotky (°C). Následně byly zaznamenány v modulu Trace a dále vyhodnoceny a srovnány s časovými průběhy modelu stanice. Perioda zaznamenávání dat byla nastavena na 1 sekundu. Výsledky měření jsou zaznamenány na obrázku č. 4.9.

4.3.2 Struktura submodelu

Tento submodel slouží pro komunikaci s reálným světem – systémem. Obsahuje základní procesní data, jako jsou teploty, současné nastavení ventilů a adresy prvků v PLC a dále samotné specifikace PLC, jako je například jeho IP adresa. Tato data jsou pak využívána pro nastavení simulace.



Obrázek 4.5 Struktura interakce s assetem PLC

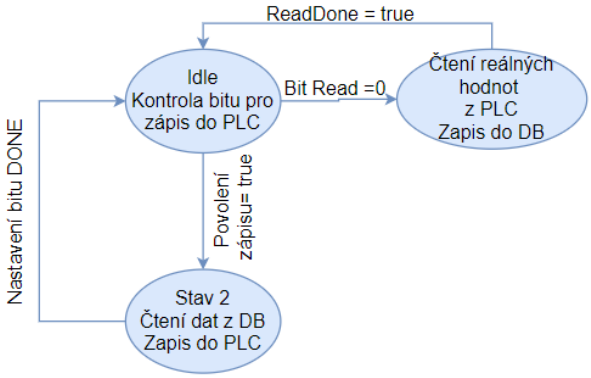


Obrázek 4.6 Struktura submodelu PLC

4.3.3 Adaptér pro komunikaci

Tento adaptér slouží k interakci se submodelem (databází) a reálným PLC s daty o aktuálním stavu stanice. Jedná se o python třídu s názvem *PLCAdapter*. Tato třída využívá knihoven python snap7 a obsahuje stavový automat dle diagramu na obrázku 4.7. Dále jsou zde vytvořeny funkce pro zápis a čtení dat z PLC Siemens na základě

jeho IP adresy a adres daných vstupů respektive výstupů. Submodel taktéž obsahuje normalizaci integer hodnot na inženýrské. Tohoto je využito při získávání dat z teplotních čidel a následné nastavení třícestných ventilů. Informace získané za pomocí tohoto submodelu jsou dále zrcadleny do submodelu pro interakci s uživatelem s názvem „HMI“.



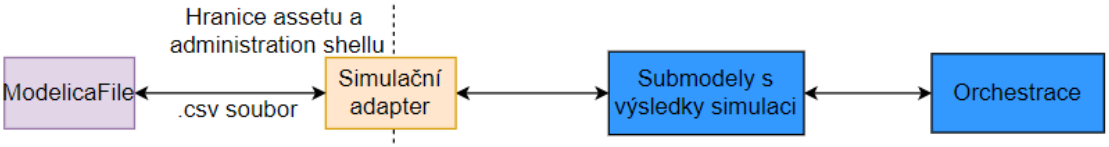
Obrázek 4.7 Struktura stavového automatu třídy PLCAdapter

4.4 Submodel vytvořený na základě výsledků simulace

Submodel s výsledky simulace je tvořen automaticky čistě za pomocí BasyX knihovny po ukončení simulace z *.csv* souboru, který obsahuje výsledky simulace. Celkem může proběhnout až 16 simulací, je tedy vytvořeno až 16 submodelů s výsledky. Tyto submodely mají stejné jméno, rozlišují se čísly 0-15. V případě, že se jedná pouze o simulaci pro druhý scénář, který je vysvětlen níže, je submodel označen číslem nula. Tyto parametry lze samozřejmě modifikovat různými způsoby.

Submodel obsahuje všechny proměnné, které jsou obsaženy v modelu. Názvy proměnných jsou určeny v OpenModelice, ale musí být upraveny z důvodu, že dle standardu skupiny IDTA nesmí ShortId obsahovat žádné speciální znaky ani mezery. Kvůli této podmínce je ve simulačním adaptéru vytvořeno zpracování jednotlivých názvů a jejich upravení, aby odpovídaly standardu dle IDTA. Hodnoty těchto proměnných odpovídají poslední hodnotě zaznamenané v simulaci. Tyto hodnoty se mění v závislosti na výsledku simulace. Bohužel se nepodařilo vložit přes knihovnu BasyX pole všech hodnot, a tudíž průběh simulace.

Do tohoto submodelu jsou následně ukládána i data s vypočtenými kriteriálními funkcemi pro zhodnocení výsledků simulace.



Obrázek 4.8 Struktura interakce AAS s výsledným souborem

4.5 Submodel simulace

Tato podkapitola popisuje simulační scénáře, nastavení a spuštění simulace v třídě ModelicaAdapter. Dále je zde vysvětleno vytvoření submodelu simulace a jeho proměnné.

4.5.1 Simulace – model

Pro vytvoření modelu a provádění simulací byl zvolen open source nástroj OpenModelica, ve kterém byl vytvořen a odladěn model výměníkové stanice, který téměř koresponduje s reálnými výsledky. Pomocí OpenModelica je pak dále Modelica soubor zkomprimován na spustitelný soubor s příponou *.exe*. Tento soubor je následně volán za pomocí příkazové řádky. Pro účely volání byl vytvořen dávkový soubor se všemi specifikacemi simulace.

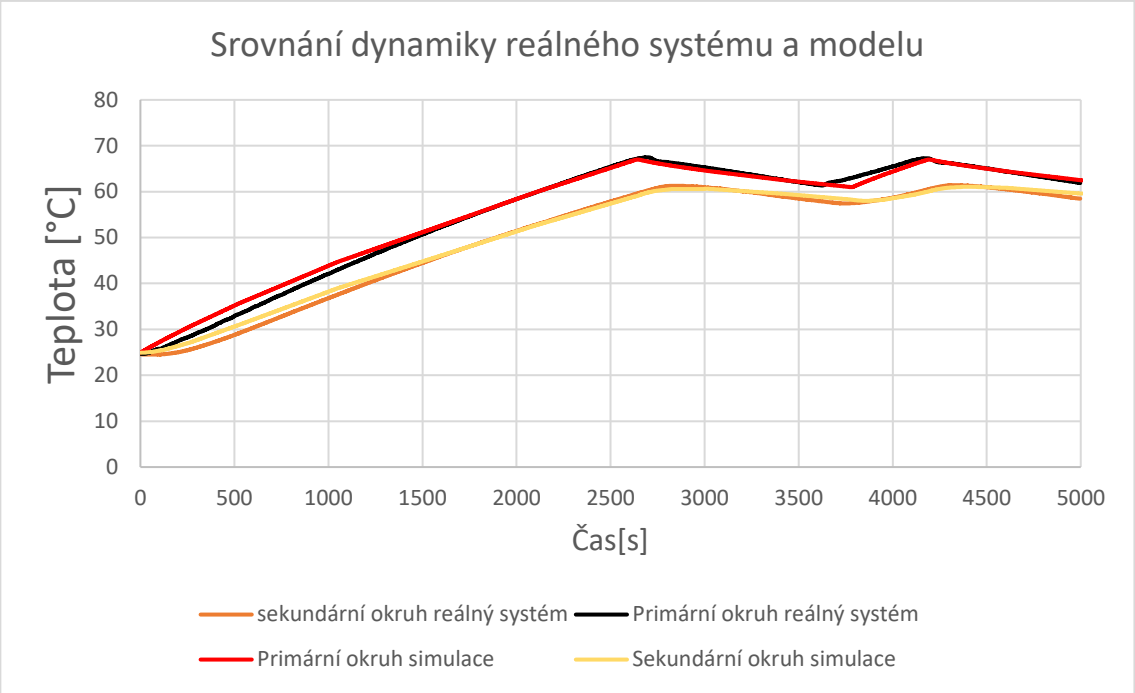
Hodnoty pro nastavení simulace byly získány z katalogových údajů komponent, kterými je demonstrační systém vybaven. A dále byl model doladěn na základě naměřených hodnot z reálného systému, aby co nejvěrohodněji odpovídal tomuto systému.

Pro realizaci simulace bylo využito volně dostupných Modelica knihoven a to konkrétněji:

- Modelica – základní knihovna dostupná přímo v OpenModelica,
- Buildings – rozšířená knihovna pro modelování modelů s kapalinami,
- ExternData – knihovna pro práci s externími soubory pro jejich načtení a zpracování.

4.5.2 Nastavení modelu

Základní nastavení bylo provedeno na základě nastavení hodnot dle katalogového listu a dále za pomocí změřených hodnot na reálném systému. Další parametry simulace, jako například výkon zátěže, či hmotnost vody v systému, byly doladěny tak, aby dynamika systému co nejvíce odpovídala reálnému systému. Na obrázku 4.9 je uvedeno srovnání dynamiky reálného systému se systémem modelovaným v OpenModelica. Cílem práce není vytvořit naprosto dokonalý model, ale integrovat tento model do AAS.



Obrázek 4.9 Srovnání dynamiky reálného systému s modelem

4.5.3 Spuštění simulace

Spouštění simulace, respektive *.exe* souboru modelu se provádí za pomocí dávkového souboru s příkazem pro spuštění a dalšími parametry pro nastavení simulace. Jednotlivé parametry a kroky, které soubor dělá jsou popsány níže.

1. Nejprve je potřeba nastavit cestu, kde je nainstalována OpenModelica
2. Následně je třeba najít cestu k souboru s modelem
3. Spuštění simulace:
    - startTime – čas spuštění simulace
    - stopTime – čas zastavení simulace
    - stepSize – perioda vzorkování
    - tolerance – tolerance výsledků
    - solver – metoda řešení a výpočtu lze volit např. Euler, dassl atd.
    - outputFormat – formát výsledků, lze zvolit buď *.csv* nebo.*mat*
    - variableFilter – filtr proměnných, lze si vyfiltrovat pouze jednu proměnnou a ta bude uložena ve výsledném souboru
    - -r/-w – příkazy pro přepsání předchozích výsledků
    - -lv – příkaz pro různé stavové informace např. LOG_STATS vypíše na konzoly informace o průběhu simulace

Byly vytvořeny 2 dávkové soubory, obsahující tyto základní parametry. Jedná se o soubory *Vymenik.bat* a *VymenikPI.bat.* Každý z nich spouští jiný model a tím pádem

i jinou simulaci. Volání těchto souborů probíhá v submodelu Simulace, tedy ve třídě *ModelicaAdapter.py* na základě získaných parametrů z databáze.

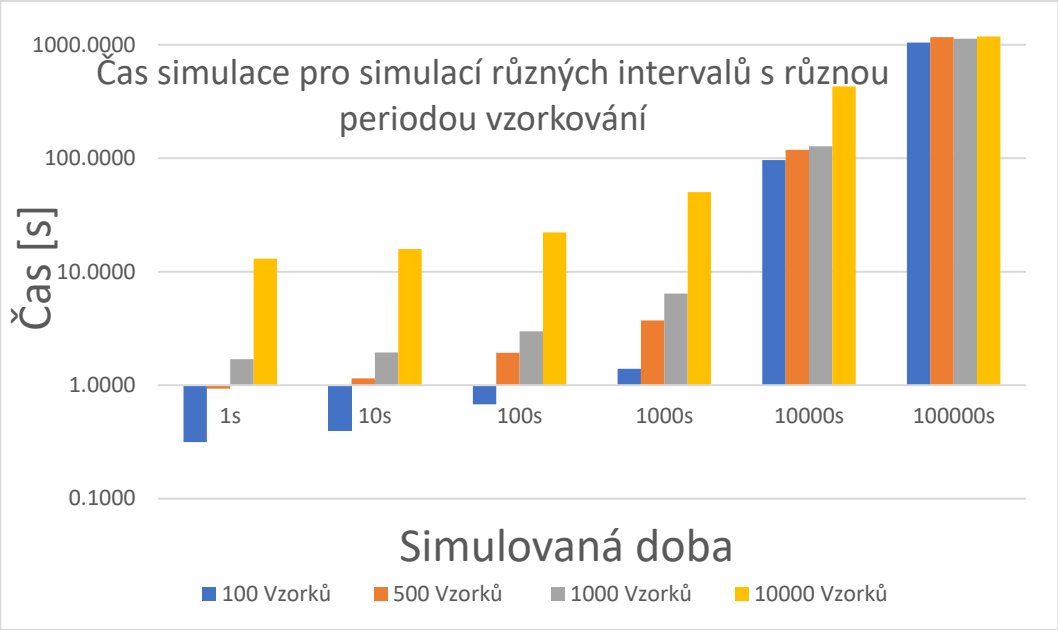
Původní myšlenka počítala s tím, že všechna data se načtou do *xlsx* nebo *csv* souboru a s tímto souborem bude následně pracovat OpenModelica. Ovšem ukázalo se, že pro inicializaci některých startovacích hodnot (např. počáteční teplota kapaliny v okruhu) nelze tento přístup využít a model nelze zkompilovat do spustitelného souboru. Tento problém se podařilo vyřešit tím, že pro inicializační hodnoty byly vytvořeny proměnné v kódu modelica s počátečními hodnotami, které dovolují kompilaci souboru. Je nutno zdůraznit, že je potřeba všechny hodnoty přepočítat na jednotky SI, to znamená, že teplota se do openModelica nepředává ve Celsiově stupnici, nýbrž ve stupnici Kelvinově. OpenModelica po té model zkompiluje bez chyby a zároveň vytvoří *.init* soubor. Pro tento soubor byl následně vytvořen parser, který najde konkrétní proměnou a přepíše jí před spuštěním simulace. Díky tomu bylo dosaženo možnosti spouštění simulace s různými počátečními podmínky. V této práci je zde možnost pouze nastavení různých počátečních teplot. Nicméně lze poměrně snadno zajistit práci i s dalšími parametry, jako je například tlak.

4.5.4 Doba simulace

Pro vytvoření co nejlepšího systému, je zapotřebí, aby simulace probíhala v co nejkratším čase. Jelikož simulování ohřevu kapaliny je poměrně zdlouhavý proces, proto je zapotřebí simulovat dobu několika desítek vteřin v jednotkách vteřin.

Na základě tohoto požadavku byla provedena měření doby simulace s různou délkou simulace a různou vzorkovací periodou. Průměrnou dobu simulace pro různé časové úseky s různou vzorkovací periodu jsou znázorněny na obrázku 4.10. Je třeba zdůraznit, že se jednalo o simulaci, při které byly oba třícestné ventily nastaveny na 100% otevření. V jiných konfiguracích ventilů může simulace trvat déle, a to z důvodu toho, že může v některých případech docházet například k dělení nulou a hledání jiného nejbližšího možného výsledku. Tato operace samozřejmě čas simulace prodlužuje.

Porovnání různých časových průběhů simulace lze vidět v tabulce 4.1, kde je znázorněno 16 simulací s různými parametry otevření ventilů



Obrázek 4.10 Grafické znázornění časů simulace

4.5.5 Simulační scénáře

Pro ověření funkčnosti implementace byli vytvořeny dva simulační scénáře, z nichž si uživatel může vybrat podle kterého bude simulace řízena a případně vyhodnocena. Pro každý scénář byla vytvořena simulace (model), obsahující vhodné prvky. Jednotlivé parametry scénářů jsou dále popsány níže.

- **Demo scénář na základě kombinací otevření ventilů**
  Tento scénář simuluje různé možnosti otevření ventilů a následně je vyhodnocováno zvoleným kritériem. Kombinace otevření ventilů je 25 %, 50 %, 75 % a 100 %. Celkem se tedy jedná o 16 kombinací a tedy 16 simulací. Tento scénář se chová tak, že po dosažení dané teploty se vypne zdroj tepla, a poté je nutno vyhodnotit simulace na základě vhodného hodnotícího integračního kritéria. Jelikož se jedná pouze o demonstrační scénář, tak vypočtené hodnoty úplně nedávají smysl. Tato funkce je zde implementována, jelikož při jiné simulaci, kdy by už dávalo smysl sledovat tyto kritéria, jsou tyto funkce využívány a stačí pouze změnit simulovaný model.

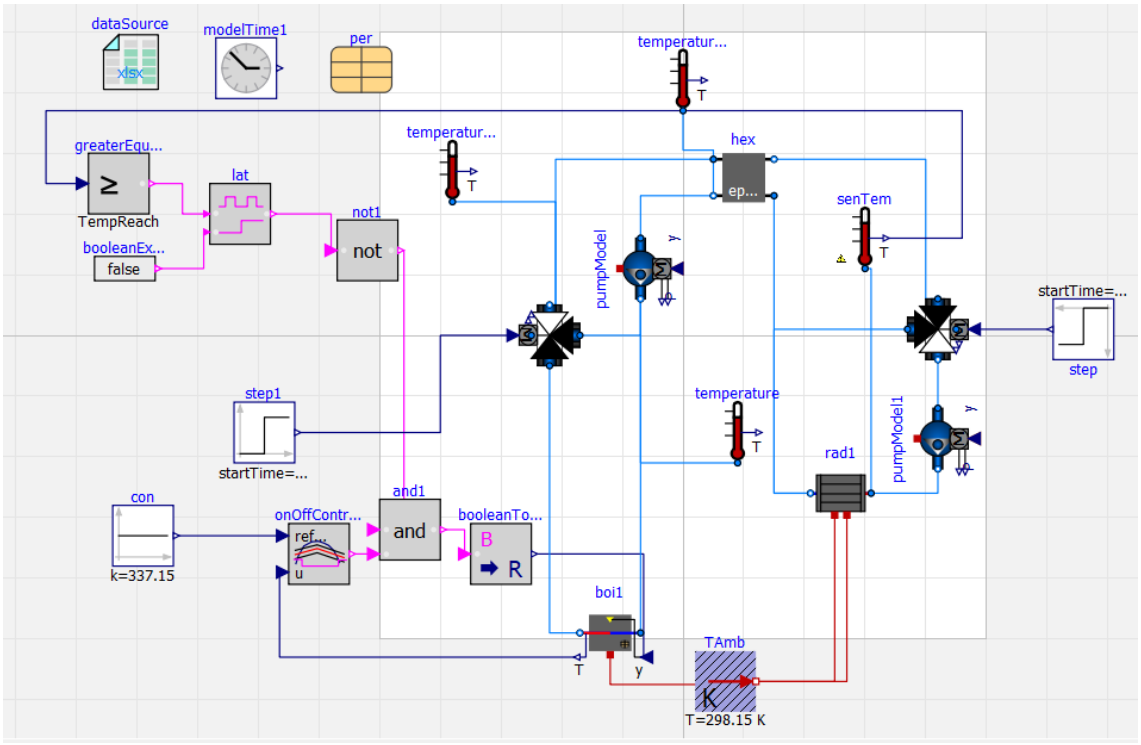
  Vyhodnocení pouze na základě rychlosti dosažení zadané teploty nemusí být vypovídající, jelikož ta nejrychlejší simulace nemusí být nejlepší. Dále je nutno předpokládat setrvačnost děje tzn. po dosažení zadané teploty bude křivka dále nějaký čas stoupat. Proto je vhodnější zvolit jinou metodu vyhodnocování než pouze na základě času dosažení teploty. Na obrázku 4.12 je ukázán průběh ohřevu sekundárního okruhu na 50 °C. V simulaci je nastaveno, že po dosažení zadané teploty se vypne ohřev a díky tomu vidíme překmit, který je způsoben

setrvačností. Délka simulace je nastavena na 10 000 sekund. Ovšem v simulaci je nastavena ukončovací podmínka, takže v realitě simulace netrvá tak dlouho. Tato podmínka je splněna po poklesu teploty pod požadovanou úroveň Ohřev bojleru byl nastaven na 65 °C s hysterezí 7 °C, aby co nejvíce odpovídal reálnému systému. Dále jsou na obrázku vidět různé křivky v závislosti na procentech otevření třícestných ventilů.

Hlavní motivace pro zvolení tohoto scénáře je ukázka možnosti spouštění více simulací a jejich následné uložení do submodelů a vyhodnocení nejlepšího z nich a na základě toho nastavení ventilů PLC systémem AAS. Všechny tyto úkony, kromě zadávání parametrů jsou vykonávány bez dalšího zásahu uživatele, tudíž se systém chová z části jako autonomní.

Simulace pro tuto situaci obsahuje navíc jednoduchou logiku pro vyhodnocení dosažené teploty, kdy se vypne ohřev bojleru po jejím dosažení. A dále jednoduchý ON-OFF kontrolér pro ovládání zdroje tepla – bojleru a dosažení požadované hystereze ohřevu. Na obrázku 4.11 je znázorněn sestavený model v OpenModelica.

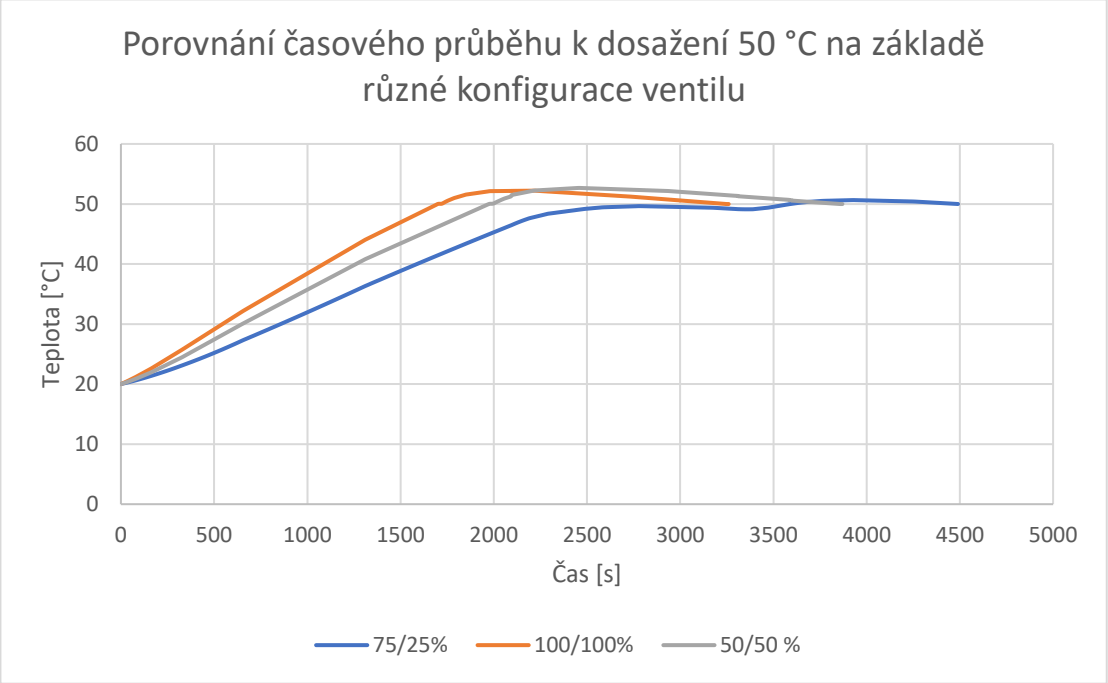
V tabulce 4.1 jsou uvedeny časové průběhy, jak už simulace modelu, tak i doby trvání celé simulace v realitě. Na obrázku 4.12 jsou pak uvedeny průběhy ohřevu pro různé kombinace ventilů.



Obrázek 4.11 Model v OpenModelica pro scénář bez regulace PI regulátorem

Tabulka 4.1 Rychlost simulace v závislosti na nastavení ventilů

Kombinace ventilů [%]	**25/25**	**25/50**	**25/75**	**25/100**	**50/25**	**50/50**	**50/75**	**50/100**
Doba trvání simulace [s]	54.966	42.775	38.437	28.671	95.875	34.173	28.88	18.96
Doba ohřevu [s]	3353.29	4046.72	3529.10	3473.05	4431.13	3814.67	3508.00	3293.41
Kombinace ventilů [%]	**75/25**	**75/50**	**75/75**	**75/100**	**100/25**	**100/50**	**100/75**	**100/100**
Doba trvání simulace [s]	117.49	22	9.01	3.29	81.98	6.11	2.62	2.198
Doba ohřevu [s]	4578.00	3757.57	3453.52	3278.46	3592.81	3712.57	3405.34	3224.1



Obrázek 4.12 Křivka dosažení 50 °C na sekundárním okruhu

- **Scénář pro nastavení ventilů na základě PI regulátoru**
  V tomto scénáři je ke třícestnému ventilu v sekundárním obvodu připojen PI regulátor, který reguluje jeho otevření. Hodnoty regulátorů byly navrženy metodou Ziegler-Nicholse. Kritické parametry pro tuto metodu byly získány z přechodové charakteristiky systému. Trojcestný ventil v primárním okruhu je nastaven jako otevřený na 100 %, jelikož regulovat primární okruh není triviální záležitost, protože obsahuje nelinearitu ve formě reléové regulace teploty vody v bojleru. Tento stav bude považován za konstantu. PI regulátor byl zvolen

z důvodu toho, aby bylo docíleno nulové ustálené regulační odchylky.

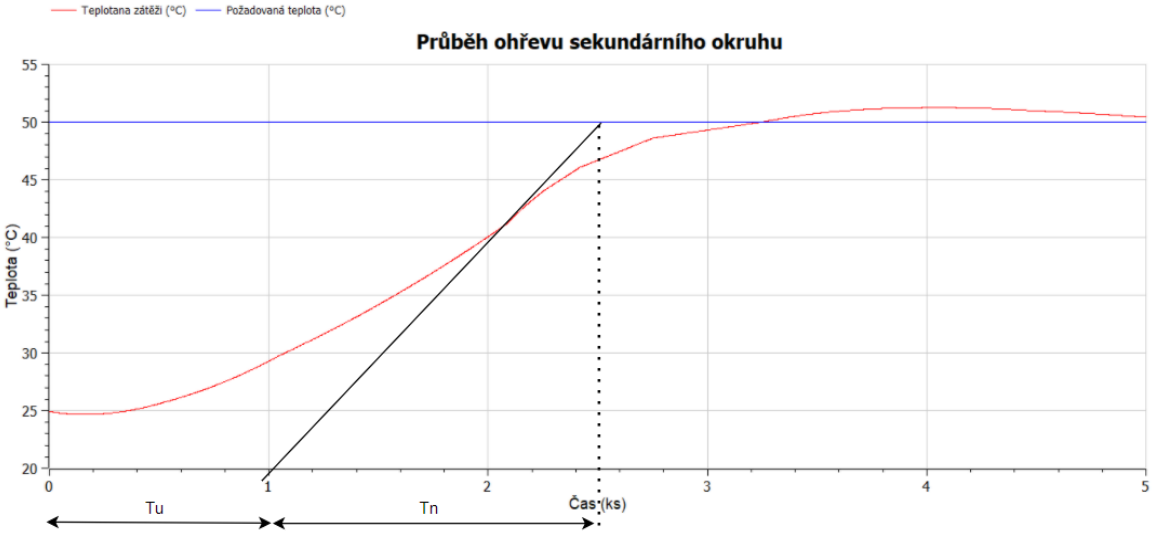
Tento scénář počítá s konstantním nastavením parametrů regulátoru, simulace budou však rozdílné v závislosti na počátečních podmínkách systému. To znamená, jakou teplotu má aktuální stav a z tohoto stavu se za jakou dobu dostaneme do námi požadovaného stavu.

Tuto aplikaci by bylo vhodné používat pro zpožděné, respektive dopředné spouštění systému. Například: Víme, že bude potřeba v 8:00 pro nějakou technologii o určitém výkonu dodávat stálý přísun media o určité teplotě. Tento scénář se odsimuluje a zjistí se, že dosáhnout požadovaného stavu za daných počátečních podmínek a za daného nastavení ventilu lze za 30 minut. Proto můžeme tuto informaci dále využít a celý systém uvést do chodu v 7:30, aby bylo dosaženo zadaných podmínek přesně v 8:00. Výstupem tohoto scénáře simulace je tedy čas, za který se děj ustálí na dané hodnotě v požadované odchylce.

V tomto případě je odchylka nastavena v algoritmické části modelica souboru, kde se porovnává hodnota žádaná s dosaženou na menší než 0,1 °C. Po dosažení této odchylky se simulace vypne, jelikož nemá cenu, aby se simuloval průběh další následující hodiny. Čas, ve kterém se simulace vypne, tj. čas při kterém je dosažena odchylka se bere jako výstup se kterým lze dále pracovat.

Návrh PI regulátoru:

1) Jak už bylo zmíněno výše, pro návrh byla zvolena metod Ziegler-Nicholse. Jeden z principů této metody spočívá k uvedení soustavy kritickým zesílením K_r na mez stability. Tohoto ovšem nelze v OpenModelice dosáhnout, jelikož ta pracuje pouze s reálnými parametry (například s tím, že u teploty vody nelze v realitě a při normálním tlaku dosáhnout vyšší teploty, než je 100 °C). Proto byla k určení parametrů kritického zesílení a integrační konstanty využita přechodová charakteristika systému (obr.4.13). Z ní byla následně určena doba náběhu T_n a doba průtahu T_u. Všechny tyto parametry byly určeny z vymodelovaného systému. Z těchto parametrů lze následujícím výpočtem získat kritické parametry pro metodu Zigler-Nicholse. Jelikož model není přesně odladěn se všemi možnostmi reálného systému, nelze tyto vypočtené údaje použít s jistotou na reálném systému. Z těchto parametrů lze následujícím výpočtem získat kritické parametry pro metodu Zigler-Nicholse.
2) Z grafu byly určeny parametry T_u= 1000 s a T_n= 1500 s. Dále určíme z rovnic (1.4) a (1.5) učíme kritické parametry K_{krit} a T_{krit}. Z těchto parametrů následně z tabulky 1.2. Lze vypočíst konstanty PI regulátoru.



Obrázek 4.13 Graf pro určení doby náběhu a doby průtahu pro metodu Ziegler Nichols

3) Výpočet parametrů PI regulátoru:

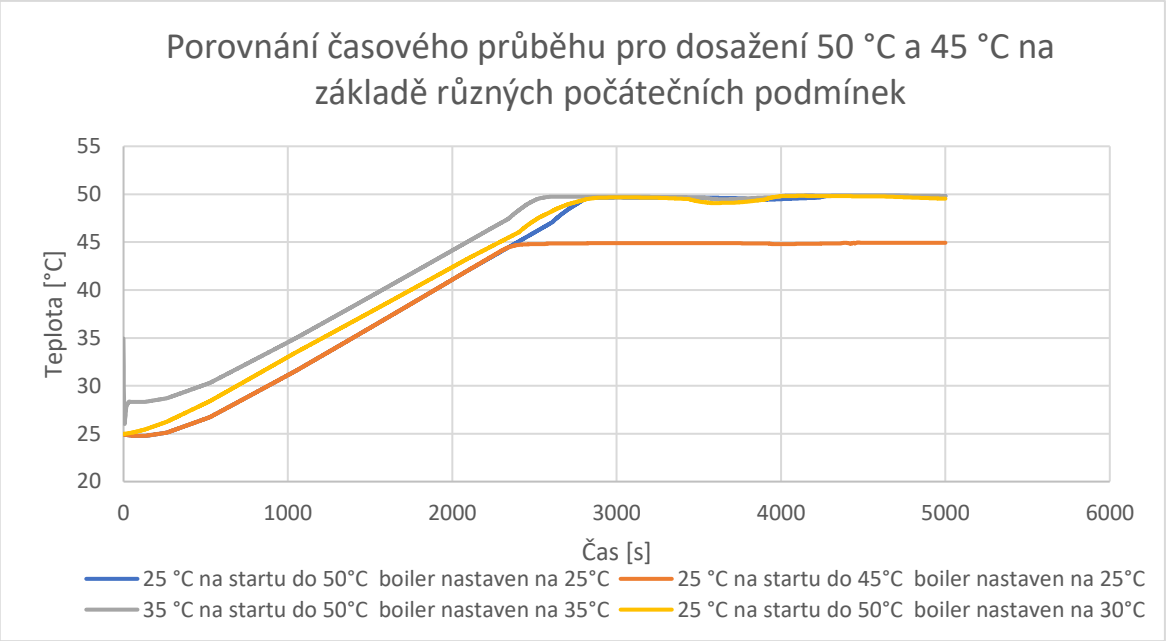
$$K_{krit} \doteq \frac{\pi}{2}\frac{T_n}{T_u} + 1 \doteq \frac{\pi}{2} * \frac{1500}{1000} + 1 \doteq 3{,}356 \quad (4.1)$$

$$T_{krit} \doteq 4T_u \doteq 4 * 1000 \doteq 4000\ s \quad (4.2)$$

$$K_{PI} = 0{,}45 * K_{krit} = 0{,}45 * 3{,}356 = 1{,}51029 \quad (4.3)$$

$$T_I = 0{,}85 * T_{krit} = 0{,}85 * 4000 = 3400\ s \quad (4.4)$$

4) Výsledná regulace při nastavení regulátoru na vypočtené hodnoty:
Na obrázku 4.14 je znázorněn časový průběh ohřevu pro různé počáteční a koncové podmínky. Výsledky odpovídají předpokladům, čím vyšší teplota na vstupu, tím rychleji dojde k vyregulování na žádanou teplotu. Výstupem z této simulace je rychlost, za jakou se podaří dosáhnout žádané teploty a zároveň lze zjistit i kvalitu regulace a tu případně porovnat v případě přenastavení jiných parametrů pro PI regulátor. Tato funkce je implementována pouze tak, že uživatel si může zadat pouze parametry přes submodel, který slouží pro interakci s uživatelem. Ukončovací podmínka pro simulaci je nastavena dle odchylky od žádané hodnoty. Tento parametr by bylo možné taktéž v případě potřeby editovat. Pro ladění parametrů PI regulátoru za chodu aplikace sloužil scénář, který je popsán v další podkapitole.



Obrázek 4.14 Graf porovnání časového průběhu pro dosažení 50 °C a 45 °C na základě různých počátečních podmínek

- **Scénář pro nastavení ventilů na základě PI regulátoru II.**
  Tento scénář by byl modifikací scénáře z předcházející kapitole. Hlavní rozdíl by byl v tom, že parametry PI regulátoru by nebyly navrženy jako konstanty, nýbrž by byl do AAS přidán další blok pro optimalizaci regulátoru.

  Tento doplněk není jednoduché implementovat a jeho implementace je nad rámec této práce. Stojí ovšem za zmínění, že v AAS je na tuto doplňkovou funkci nachystáno a zdokumentováno místo pro implementaci ve zdrojovém kódu i v proměnných AAS.
- **Další možnosti scénářů**
  Mezi další možnosti by se dalo navrhnout regulaci nelineárního primárního okruhu na konstantní teplotu, tím by bylo zjednodušena regulace v okruhu sekundárním a dalo by se rychleji reagovat na poruchy ve formě kolísající teploty při odebíraném teplu. V teoretické rešerši jsou rozebrány rozvětvené regulační obvody. Do těchto obvodů patří i tento systém. Konkrétněji se jedná o rozvětvený regulační obvod s pomocnou akční veličinou. Kde odpovídá druhému akčnímu zásahu třícestný ventil v primárním okruhu. Bohužel bojler není zdrojem konstantní teploty, nýbrž se chová jak relé s hysterezí. Proto návrh takového regulátoru není úplně triviální.

  Další možností vytvoření scénáře je regulovat spínání bojleru. Mohli bychom použít klasický P regulátor. Jelikož v reálném systému v sobě má bojler integrovanou reléovou regulaci s pevnou hysterezí (při měření odpovídala cca 6 až 7 °C) nemuseli bychom se bát kmitání ze stran P regulátoru. Dále třícestný ventil v primárním okruhu by byl nastaven na plně otevřen. V modelu s tímto

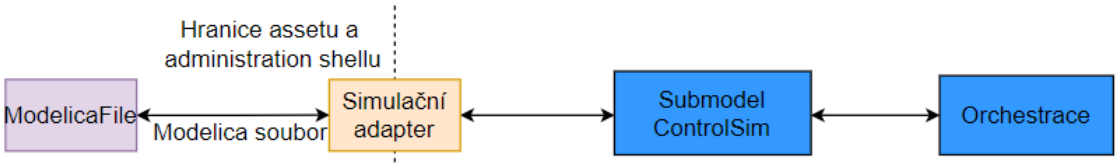
přístupem se daří vytvořit poměrně ustálenou regulační odchylku. Je ovšem třeba mít v paměti to, že tento model není ideální z hlediska přesnosti a jeho dynamických vlastností, jelikož neproběhlo hlubší zkoumání jeho reálných dynamických charakteristik.

Dále by se jistě dalo navrhnout regulaci na základě nelineárního odběru tepelného výkonu, např. pokud by se do soustavy dostala další porucha. Možností, jak vylepšovat AAS se najde celá řada. Tato práce má za cíl ukázat jakým způsobem se dají některé vybrané funkce implementovat a pokusit se aspoň z části přichystat možnosti pro implementaci výše zmíněných scénářů. A dále vytvořit vzor, podle kterého lze implementovat další modely do AAS.

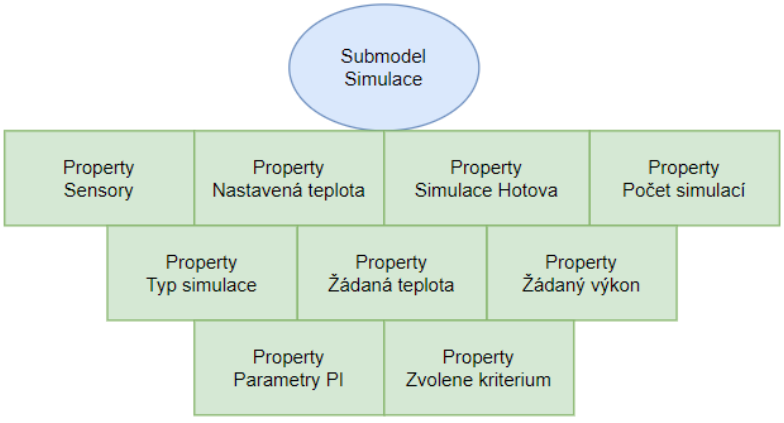
4.5.6 Tvorba submodelu

Tento submodel slouží pro interakci se simulací, informuje o jejím aktuálním průběhu. Obsahuje proměnné pro inicializaci simulace, a to jsou teploty vyčtené z reálného systému a dále žádané parametry, které si uživatel volí v submodelu HMI jako například typ simulace anebo požadovaná teplota na výstupu. A v neposlední řadě obsahuje proměnnou, pro kontrolu stavu simulace.

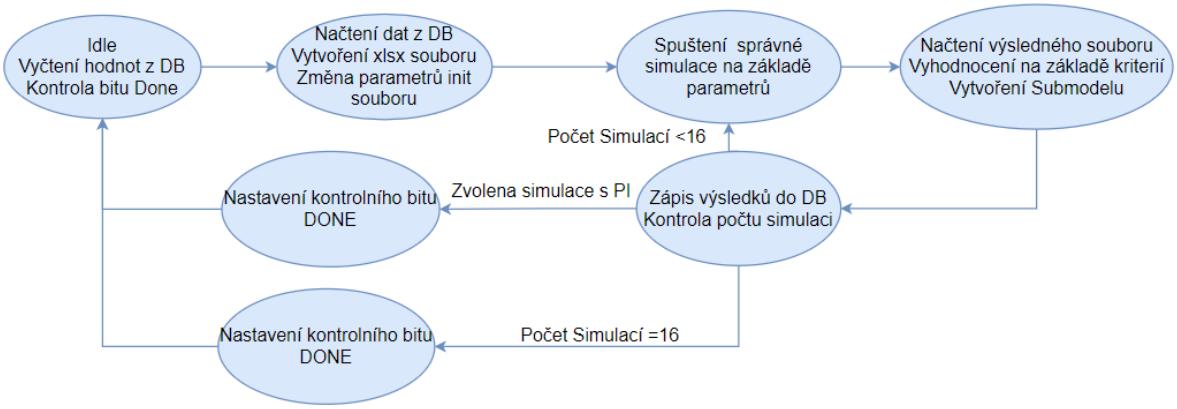
Jak už bylo zmíněno výše, simulační adaptér interaguje se zkompilovaným modelica souborem přes volání dávkového souboru, ve kterém jsou nastaveny základní parametry pro běh simulace.



Obrázek 4.15 Struktura interakce se spustitelným Modelica souborem



Obrázek 4.16 Struktura Submodelu výsledku simulace



Obrázek 4.17 Stavový automat třídy ModelicaAdapter

4.6 Submodel pro interakci s uživatelem - „HMI“

Tento submodel slouží k interakci s uživatelem a pomocí FastAPI by měl vytvářet jakési uživatelské rozhraní s vnitřní části AAS. Tento submodel slouží jako rozhraní pro koncové uživatele, který může zadávat požadovanou teplotu.

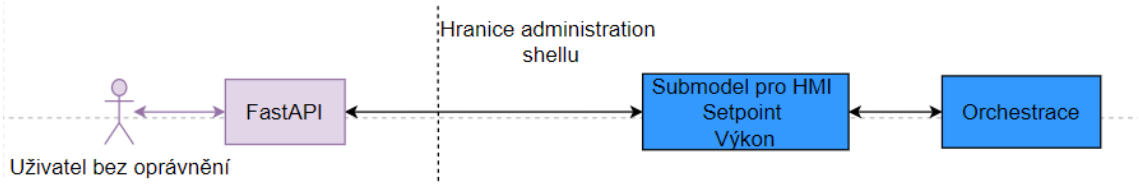
Pro účely testování nebylo vytvářeno API pro uživatele dle [21]. V této testovací fázi se zadané parametry a hodnoty zadávali pouze přes uživatelské rozhraní CouchDB. Pro reálné nasazení je zapotřebí tento submodel důsledně oddělit od vnitřku AAS pro zajištění větší bezpečnosti a robustnosti systému. K tomu by mohly být využity možnosti CouchDB, jelikož podporuje různé možnosti uživatelských práv. Pokud by byly nastaveny práva a jiné uživatelské údaje pro tento submodel.

Struktura submodelu, která je znázorněna na obrázku 4.19 obsahuje základní údaje o aktuální teplotě měřenou senzory a následně vlastnosti pro vyplnění od uživatele, jako jsou žádaná teplota, na kterou chceme simulovat, žádaný výkon, který by představoval zátěž výměníkové stanice. V poslední řadě lze zvolit jaký typ scénáře chceme zvolit. Pro druhý scénář zapíšeme do proměnné „PI“ pro první scénář zapíšeme „bezPI“. Následně lze ještě zvolit na základě kterého kritéria si uživatel přeje simulaci vyhodnotit. Možnosti jsou:

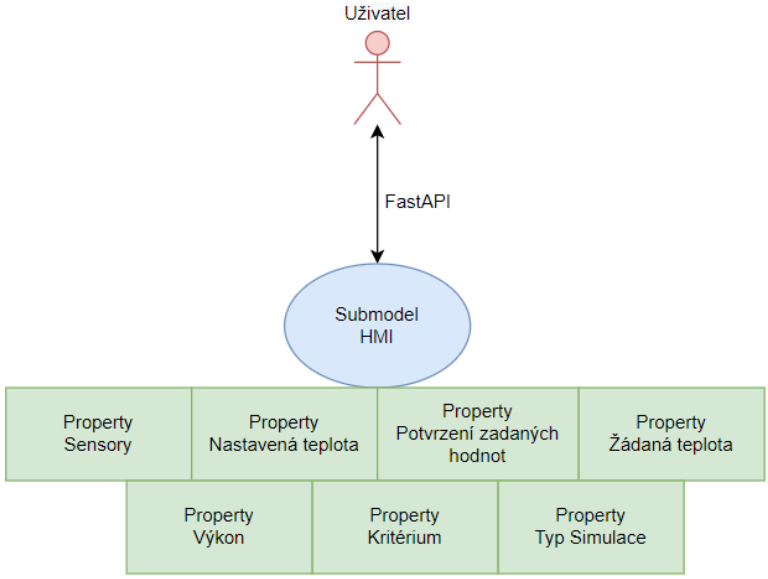
- „Linear“ – pro lineární integrální kritérium,
- „ITAE“ – pro ITAE kritérium
- „ABS“ – absolutní integrální kritérium,
- „Kvadra“ – pro kvadratické integrální kritérium.

Jednotlivá kritéria a jejich funkčnosti jsou popsány v teoretické rešerši konkrétněji v kapitole 1.3. V posledním kroku uživatel potvrdí zadaná data tím, že nastaví hodnotu potvrzovací proměnné na „True“.

Tento submodel nemá žádnou svou vlastní třídu, slouží pouze jako reaktivní část AAS a pro interakci s uživatelem. Hodnoty, které uživatel zadá hlídá submodel orchestrátoru a ten je i dále distribuuje do ostatních submodelů.



Obrázek 4.18 Struktura interakce AAS s uživatelem

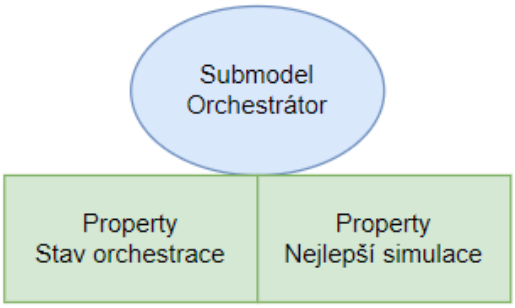


Obrázek 4.19 Struktura Submodelu „HMI“

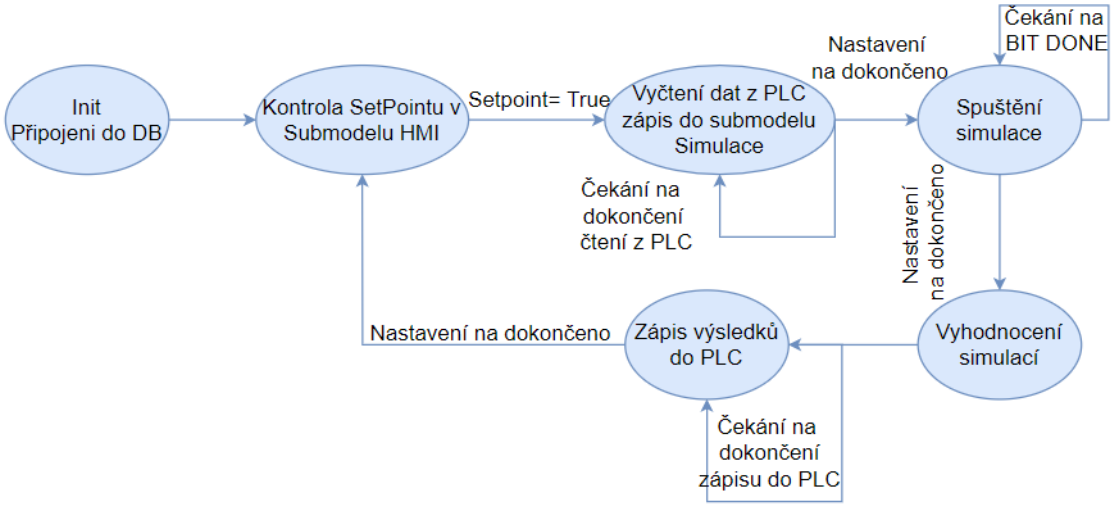
4.7 Submodel orchestrátor

Submodel orchestrátoru slouží pro ovládání celého AAS. Je opět realizován ve formě třídy s názvem *Orchestrator.py*. Orchestrace probíhá ve *While* smyčce a vždy čeká na potvrzení zadaných uživatelských dat v submodelu HMI. Po tomto potvrzení se rozběhne cyklus znázorněný na stavovém diagramu na 4.21. Koordinuje spolupráci mezi ostatními submodely. Dále zde probíhá zrcadlení proměnných do ostatních submodelů jako například při vyčtení reálných parametrů z PLC za pomocí submodelu PLC, orchestrátor vezme tyto informace a zároveň je vloží do uživatelského submodelu HMI a následně do submodelu pro simulaci, kde jsou hodnoty využity pro nastavení simulace a její realizaci. Submodel dále řídí spouštění simulací, povolování zápisu hodnot do PLC a v neposlední řadě výběr té nejlepší simulace. Jednotlivé stavy této orchestrace jsou znázorněny na obr. 4.21.

Struktura tohoto submodelu obsahuje pouze základní informace o aktuálním stavu orchestrátoru a informaci, která ze simulací ze druhého simulačního scénáře je ta nejlepší. Strukturu lze samozřejmě dále vylepšovat o další ovládací prvky.



Obrázek 4.20 Struktura Submodelu orchestrátoru



Obrázek 4.21 Stavový diagram Orchestrátoru

4.8 Vyhodnocení

V této kapitole byla popsána realizace integrace simulátoru do AAS. Jednotlivé ovládací bloky byly realizovány ve formě tříd v programovacím jazyce Python. Pro vytvoření skeletu AAS byly jednotlivé submodely nejprve vytvořeny v AASX PackageExploreru a uloženy jako soubor *.aasx*. Tento soubor Byl poté za pomocí BasyX knihoven nahrán do databáze.

Následně byly vytvořeny dva modely systému v OpenModelica, pro každý navržený scénář zvlášť. Cílem nebylo vytvořit dokonalý model systému, ale vymyslet způsob, jakým lze tento simulátor zaintegrovat do AAS.

První navržený scénář slouží hlavně jako demonstrace možností využití AAS. Jeho hlavním výstupem je to, že můžeme do AAS lze zakomponovat jak virtuální model reálné soustavy, tak lze s tímto modelem kooperovat na základě vyčtených reálných parametrů pomocí PLC. A dále tuto reálnou soustavu pomocí AAS prostřednictvím PLC řídit na základě výsledků simulace modelu.

Druhý scénář má už rysy reálnějšího využití byl zde navržen PI regulátor metodou Ziegler-Nicholse a odladěn na modelu. K porovnání s reálným systémem bohužel vzhledem k nedostatku času nedošlo. Model by bylo nutno vyzkoušet s více scénáři na reálném systému. Cílem této práce byla hlavně integrace těchto modelů.

Tento scénář simuluje na základě různých počátečních podmínek získaných z reálného systému. Díky tomu můžeme určit dobu trvání děje a vyhodnotit regulaci na základě integračních kritérií. Tyto informace pak lze využít k dopřednému zapnutí systému a tím pádem lze predikovat, kdy bude systém zahřátý na zadané podmínky.

V systému bylo vytvořeno místo pro možnou integraci optimalizátoru PI regulátorů a v kódu bylo náležitě zadokumentováno. Systém se může dále rozšiřovat o další funkcionality, tyto věci jsou už ovšem nad rámec této práce.

Pro interakci s AAS by mělo být vytvořeno API dle[21], což je další možnost pokračování v práci. V tomto API by se mělo brát ohled na bezpečnost a robustnost systému. Z hlediska možné bezpečnosti byl zvolen pro interakci s uživatelem samostatný submodel, do kterého se za pomocí orchestrátoru zrcadlí pouze vybrané proměnné a jen s těmito může uživatel editovat.

V případě reálného nasazení by bylo potřeba daný model více doladit a ověřit jeho dynamické vlastnosti pro více scénářů, například pro nahřátí primárního okruhu a následné otevření. Pro tyto možnosti lze v modelice nastavit spoustu dalších parametrů a je zde tudíž spousta možností, jakými ladit tento model. Je nutno vytvořit více scénářů na reálném systému a tyto scénáře následně porovnat s dostupným modelem a případně tento model na jejich základě více doladit. Jak již bylo zmíněno Modelica umožňuje nastavení velkého množství parametrů, pro představu lze počítat i s hmotností látky v potrubí, dále lze model doplnit o potrubí a jeho tepelné vlastnosti s prostupem tepla a jistě by se daly uvést další příklady.

Celý systém má samozřejmě i svá omezení. V této fázi prakticky chybí uživatelské rozhraní, testování probíhalo takovým způsobem, že se hodnoty databáze přepisovaly ručně přes CouchDB webový prohlížeč. Toto uživatelské rozhraní by mělo být vytvořeno dle druhé části standardu pro specifikace AAS. Dále pro jinou stanici by bylo nutno celý systém přepracovat a vytvořit submodely na míru tomuto dalšímu řešení. Ovšem lze již postupovat dle tohoto návrhu, kdy máme vytvořeny a zdokumentovány možnosti, jakým způsobem lze komunikovat se modelem a PLC.

5. ZÁVĚR

V první kapitole práce byla vytvořena literární rešerše s informacemi podstatnými pro naplnění cílů této práce. Byla zde rozebrána problematika AAS a následně problematika pro vyhodnocování výsledků regulace a metoda návrhu regulátoru Ziegler-Nichols.

Druhá kapitola obsahuje rozbor několika simulačních a modelujících nástrojů. Tyto nástroje jsou zde srovnány a jsou uvedeny jejich hlavní výhody a nevýhody.

Ve druhé kapitole jsou představeny a krátce rozebrány dostupné simulační a modelovací aplikace. Byl zde proveden pokus pro spojení softwaru Matlab a Unity 3D za pomocí TCP/IP protokolu. Dále byla změřena a vyhodnocena za pomocí LabView rychlost odezvy a úspěšnost komunikace.

Třetí kapitola byla věnována návrhu řešení integrace simulátoru do AAS. Byla zde představena fyzická část systému, dále bylo vytvořeno procesní schéma systému, která je uvedeno v příloze, včetně popisu jednotlivých komponent. Poslední část kapitoly obsahuje blokové schéma navrhovaného systému včetně popisu komunikace mezi jednotlivými bloky.

V poslední kapitole byla popsána realizace systému a obsah jednotlivých submodelů. Bylo zde provedeno srovnání s reálným systémem a popis měření na reálném systému. Dále jsou zde uvedeny možné scénáře pro simulace modelu a jejich možnosti využití. Pro jeden scénář byl proveden návrh regulátoru za pomocí metody Ziegler-Nichols. V poslední části kapitoly bylo provedeno vyhodnocení systému a diskuze možností a limitů celého systému.

Cílem celé práce bylo vytvořit funkční systém AAS s integrovaným simulátorem uvnitř. Dále zhodnotit jeho možnosti využití a odhalit možné limity systému. Lze konstatovat, že cíle práce se podařilo naplnit.

LITERATURA

[1] MINISTERSTVO PRŮMYSLU A OBCHODU ČR. Iniciativa průmysl 4.0. Online. 2016, s. 1-233. Dostupné z: https://www.mpo.cz/assets/dokumenty/53723/64358/658713/priloha001.pdf. [cit. 2024-04-21].

[2] ARM, Jakub; BENESL, Tomas; MARCON, Petr; BRADAC, Zdenek; SCHRÖDER, Tizian et al. Automated Design and Integration of Asset Administration Shells in Components of Industry 4.0. Online. Sensors. 2021, roč. 21, č. 6. ISSN 1424-8220. Dostupné z: https://doi.org/10.3390/s21062004. [cit. 2023-11-25].

[3] BRADAC, Z; MARCON, P; ZEZULKA, F; ARM, J a BENESL, T. Digital Twin and AAS in the Industry 4.0 Framework. Online. IOP Conference Series: Materials Science and Engineering. 2019, roč. 618, č. 1. ISSN 1757-8981. Dostupné z: https://doi.org/10.1088/1757-899X/618/1/012001. [cit. 2023-11-25].

[4] INDUSTRIAL DIGITAL TWIN ASSOCIATION. Specification of the Asset Administration Shell: Part1: Metamodel. Online. In: . Version 3.0. Dostupné z: https://industrialdigitaltwin.org/en/wp-content/uploads/sites/2/2023/06/IDTA-01001-3-0_SpecificationAssetAdministrationShell_Part1_Metamodel.pdf. [cit. 2024-04-21].

[5] Asset Administration Shell. Online. DIEDRICH, Christian. OTTO-VON-GUERICKE- UNIVERSITY MAGDEBURG. I40 - Industrie 4.0. 2020. Dostupné z: https://www.i40.ovgu.de/i40/en/Asset+Administration+Shell.html. [cit. 2023-11-25].

[6] DIEDRICH, Christian a BELYAEV, Alexander. Specification "Demonstrator I4.0-Language" v3.0. Online. 2019, s. 39. Dostupné z: https://www.researchgate.net/publication/334429449. [cit. 2023-11-25].

[7] BLAHA, Petr a VAVŘÍN, Petr. Řízení a regulace I. Online, Elektronické skriptum. Brno: VUT Brno. [cit. 2024-04-21].

[8] ŠVARC, Ivan. *TEORIE AUTOMATICKÉHO ŘÍZENÍ*. Online, Elektronické skriptum. Brno: VUT Brno, 2003. Dostupné z: http://www.fsiforum.cz/upload/soubory/knihy/Automatizace/Teorie.automatickeho.rizeni_Svarc_2003.pdf. [cit. 2024-04-21].

[9] JIRSA, Jan. NÁSTROJE PRO MODELOVÁNÍ A SIMULACE VÝROBNÍCH PROCESŮ. Online. In: . S. 65-70. Dostupné z: http://prog-story.technicalmuseum.cz/images/dokumenty/Programovani-TSW-1975-2014/2004/2004-11.pdf. [cit. 2024-04-21].

[10] KACZMARCZYK, Václav. Digitální továrna. Online. In: *Přednáška z předmětu MPC-AUP*. Dostupné z: https://www.vut.cz/studenti/predmety/detail/269394?apid=269394. [cit. 2023-12-21].

[11] UNITY 3D. Unity Industry. Online. 2022. Dostupné z: https://unity.com/products/unity-industry. [cit. 2023-11-25].

[12] ABB. Product Specification: RobotStudio. Online. In: . Dostupné z: https://search.abb.com/library/Download.aspx?DocumentID=3HAC026932-001&LanguageCode=en&DocumentPartId=&Action=Launch. [cit. 2024-04-21].
[13] SIEMENS. Siemens SIMIT. Online. 2023. Dostupné z: https://www.siemens.com/global/en/products/automation/industry-software/simit.html. [cit. 2023-11-25].
[14] Modelica® – A Unified Object-Oriented Language for Systems Modeling. Language Specification. Version 3.7. Modelica Association, 2023. Dostupné z: https://specification.modelica.org/master/MLS.pdf. [cit. 2023-11-25].
[15] FRITZSON, Peter. The Openmodelica Environment for Building Digital Twins of Sustainable Cyber-Physical Systems. Online. 2021 Winter Simulation Conference (WSC). 2021, s. 1-12. ISBN 978-1-6654-3311-2. Dostupné z: https://doi.org/10.1109/WSC52266.2021.9715443. [cit. 2023-11-25].
[16] OPEN SOURCE MODELICA CONSORTIUM. OpenModelica User's Guide. V1.22.3-1-g24592ab6d0. 2024. Dostupné z: https://openmodelica.org/doc/OpenModelicaUsersGuide/OpenModelicaUsersGuide-1.22.pdf. [cit. 2024-04-30].
[17] BALÁŠEK, Marek. Kotle a výměníky tepla. Čtvrté. Brno: Akademické nakladatelství CERM, 2022. ISBN 978-80-214-6093-5.
[18] ALFA LAVAL. *Pájené deskové výměníky tepla Alfa Laval: Produktová řada pro aplikace vytápění, chlazení a klimatizace*. Online, PDF. 2009. Dostupné z: https://www.kpmark.cz/uploads/katalogy/Pajene_vymeniky_Alfa_Laval.pdf. [cit. 2024-04-30].
[19] TECHNICKÝ KATALOG GRUNDFOS: Oběhová čerpadla série 100. Online, PDF. 1. 2008. Dostupné z: https://product-selection.grundfos.com/cz/products/alpha/alpha-plus/alpha-25-40-180-96288941?pumpsystemid=2306312992&tab=documentation. [cit. 2024-04-30].
[20] ČERNÍK, Michal. MODELOVÁNÍ TEPLOVODNÍ VÝMĚNÍKOVÉ STANICE A JEJÍ REGULACE. Online, Bakalářská práce, vedoucí Ing Ondrej Mihálik. Brno: VUT Brno, 2022. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/145357. [cit. 2023-12-18].
[21] INDUSTRIAL DIGITAL TWIN ASSOCIATION. Specification of the Asset Administration Shell: Part2: Application Programming Interface. Online. In: . Version 3.0. Dostupné z: https://industrialdigitaltwin.org/en/wp-content/uploads/sites/2/2023/06/IDTA-01002-3-0_SpecificationAssetAdministrationShell_Part2_API_.pdf [cit. 2024-04-21].
[22] IBM. *What is CouchDB?* Online. IBM. 2023. Dostupné z: https://www.ibm.com/topics/couchdb. [cit. 2024-04-28].
[23] ECLIPSE FOUNDATION. *Eclipse BaSyx*. Online. 2024. Dostupné z: https://projects.eclipse.org/projects/dt.basyx. [cit. 2024-04-29].
[24] *Python-snap7 documentation*. Online. 2013. Dostupné z: https://python-snap7.readthedocs.io/en/latest/. [cit. 2024-04-29].

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

AAS	Asset Administration Shell
AASX	Asset Administration Shell eXchange
API	Application programming interface
CPS	Cyber-Physical Systems
FEKT	Fakulta elektrotechniky a komunikačních technologií
FMI	Functional Mockup Interface
IDTA	Industrial Digital Twin Association
IP	Interner Protocol
IRDI	International Registration Data Identifier
IRI	Internationalized Resource Identifier
MQTT	Message Queuing Telemetry Transport
OPC UA	Ole for Proces Control Unified Architecture
PLC	Programmable Logic Controller
TCP/IP	Transmission control protocol/Internet protocol
VDE	Verband der Elektrotech.Elektronik Informationstechnik e.V.
VDI	Verein Deutscher Ingenieure e.V.
VDMA	Verband Deutscher Maschinen- und Anlagenbau e.V.
VUT	Vysoké učení technické v Brně

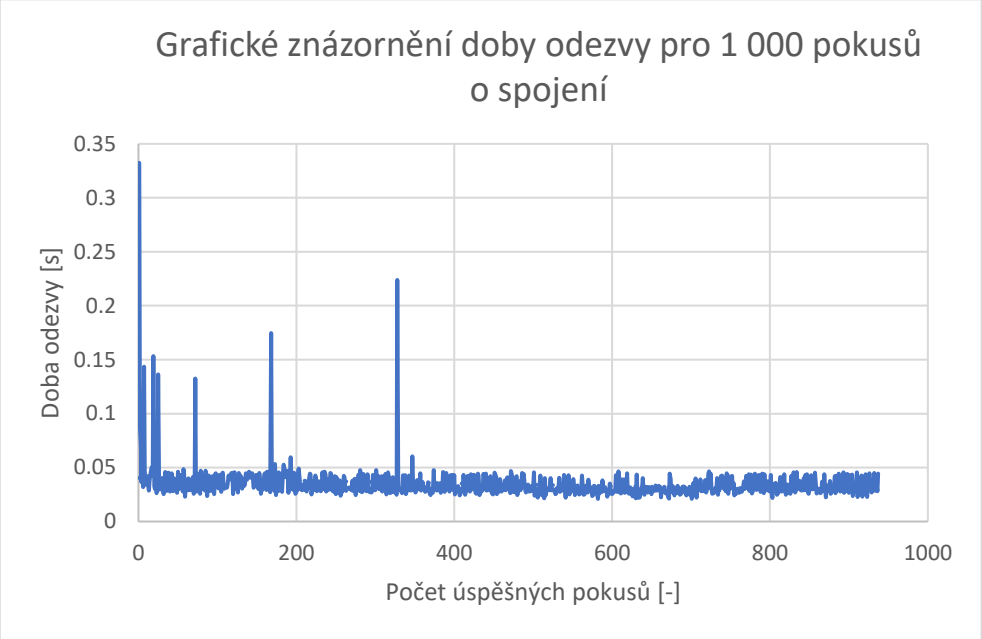
Symboly:

$\bar{x}$	prvotní odhad – průměr	(s)
x_i	i-té měření	(s)
n	počet měření	(-)
u_a	nejistota typu A	(s)
u_c	celková kombinovaná nejistota	(s)
U	rozšířená standardní nejistota	(s)
k_r	koeficient rozšíření	(-)
V	úspěšnost komunikace	(%)
n_v	počet úspěšných pokusů	(-)
p	celkový počet pokusů	(-)

SEZNAM PŘÍLOH

Příloha A - Výsledky měření odezvy mezi Unity3D a Matlabem

A.1 Grafické znázornění doby odezvy pro 1 000 pokusů o spojení

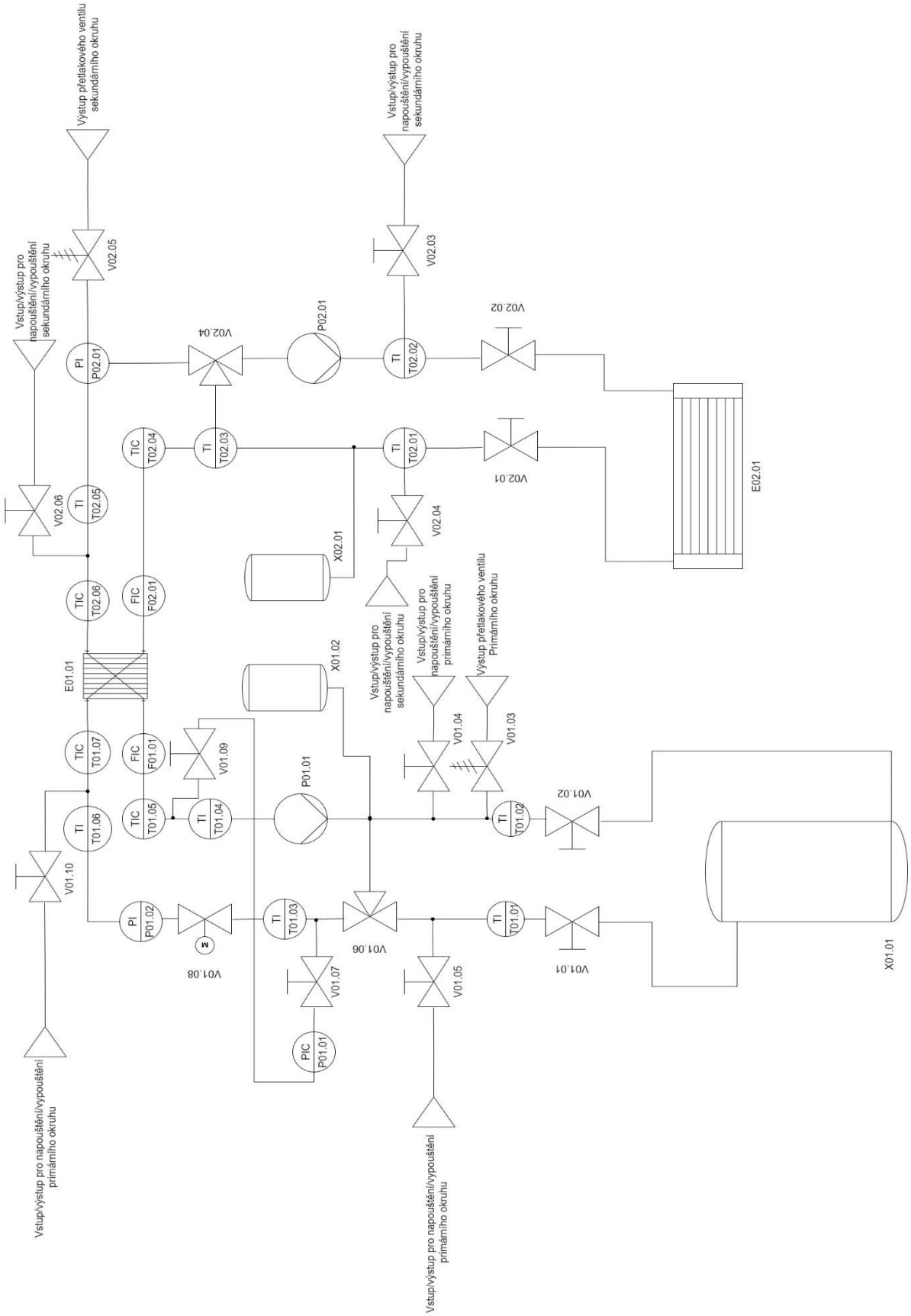


A.2 Grafické znázornění doby odezvy pro 10 000 pokusů

A.3 Grafické znázornění doby odezvy pro 20 000 pokusů o spojení

Příloha B - Schematický popis stanice

B.1 Procesní schéma



B.2 Popis instrumentace z procesního schéma

Označení	Popis	Účel
V01.01	Manuální ventil pro zastavení vody do bojleru	Vhodné pro výměnu bojleru bez vypouštění celého okruhu
V01.02	Manuální ventil pro zastavení vody do bojleru	Vhodné pro výměnu bojleru bez vypouštění celého okruhu
V01.03	Přetlakový ventil primárního okruhu	Ochrana proti přetlaku v okruhu
V01.04	Manuální ventil	Pro napouštění/vypouštění primárního okruhu
V01.05	Manuální ventil	Pro napouštění/vypouštění primárního okruhu
V01.06	Trojcestný ventil RV111 ovládaný servopohonem	Pro regulaci
V01.07	Manuální ventil pro zastavení vody k snímači tlaku	Vhodné pro výměnu snímače
V01.08	Ventil RV111 ovládaný servopohonem	Pro zastavení přívodu teplé vody k výměníku
V01.09	Manuální ventil pro zastavení vody k snímači tlaku	Vhodné pro výměnu snímače
V01.10	Manuální ventil	Pro napouštění/vypouštění/odvzdušnění primárního okruhu
T01.01	Vizuální teplotní senzor	Pro vizuální kontrolu teploty
T01.02	Vizuální teplotní senzor	Pro vizuální kontrolu teploty
T01.03	Vizuální teplotní senzor	Pro vizuální kontrolu teploty
T01.04	Vizuální teplotní senzor	Pro vizuální kontrolu teploty
T01.05	Teplotní sensor PTP05	Pro získání teploty soustavy pro PLC
T01.06	Vizuální teplotní senzor	Pro vizuální kontrolu teploty
T01.07	Teplotní sensor PTP05	Pro získání teploty soustavy pro PLC
P01.01	Tlakový sensor DMD331	Pro získání hodnoty tlaku pro PLC
P01.02	Vizuální snímač tlaku	Pro vizuální kontrolu tlaku [kPa]
X01.01	Bojler Ariston	Pro ohřev kapaliny
X01.02	Expanzní nádoba CIMM ACS5l	Pro vyrovnávání změn objemu kapaliny a udržování přetlaku v soustavě
P01.01	Oběhové čerpadlo Grundfos Alpha+ 25-40	Pro oběh kapaliny
E01.01	Výměník AlphaLaval	K tepelné výměně pro dvě odlišné média
F01.01	Snímač průtoku Enbra EV TCM 142/99-3047	Pro získání hodnoty průtoku pro PLC
V02.01	Manuální ventil pro zastavení vody do topení	Vhodné pro výměnu radiátoru bez vypouštění celého okruhu
V02.02	Manuální ventil pro zastavení vody do topení	Vhodné pro výměnu radiatoru bez vypouštění celého okruhu
V02.03	Manuální ventil	Pro napouštění/vypouštění sekundárního okruhu
V02.04	Manuální ventil	Pro napouštění/vypouštění sekundárního okruhu

V02.05	Trojcestný ventil RV111 ovládaný servopohonem	Pro regulaci
V02.06	Manuální ventil	Pro napouštění/vypouštění/odvzdušnění primárního okruhu
V02.07	Přetlakový ventil sekundárního okruhu	Ochrana proti přetlaku v okruhu
T02.01	Vizuální teplotní senzor	Pro vizuální kontrolu teploty
T02.02	Vizuální teplotní senzor	Pro vizuální kontrolu teploty
T02.03	Vizuální teplotní senzor	Pro vizuální kontrolu teploty
T02.04	Teplotní sensor PTP05	Pro získání teploty soustavy pro PLC
T02.05	Vizuální teplotní senzor	Pro vizuální kontrolu teploty
T02.06	Teplotní sensor PTP05	Pro získání teploty soustavy pro PLC
F01.02	Snímač průtoku Enbra EV TCM 142/99-3047	Pro získání hodnoty průtoku pro PLC
P02.01	Vizuální snímač tlaku	Pro vizuální kontrolu tlaku [kPa]
X02.01	Expanzní nádoba CIMM ACS5l	Pro vyrovnávání změn objemu kapaliny a udržování přetlaku v soustavě
E02.01	Tepelný výměník medium-vzduch	Slouží jako tepelná zátěž

Příloha C - Obsah přiloženého média

V přílohách je obsažen složka se soubory patřící k AAS, složka se soubory pro komunikaci mezi Matlabem a Unity, včetně jejich zpracování v LabView projektu. Nakonec soubor Modelica se simulačními soubory Výměníkové stanice a ukázkový soubor. Posledním souborem je elektronická verze práce s procesním schématem.

- Složka AAS
  - VymenikovaStanice.aasx……AASX soubor – pasivní část AAS
  - DBConf.py…… soubor pro převod AASX souboru do CouchDB
  - ModelicaAdapter.py…….…….třída pro ovládání simulace
  - Orchestrator.py……třída pro orchestraci submodelů AAS
  - PLCAdapter.py…….. třída pro ovládání zápisu a čtení reálných dat
  - SimulationData.xlsx……..... soubor s parametry simulace
  - Vymenik+VymenikPI.bat……..dávkové soubory pro spouštění simulací
  - Složky se simulačními soubory

- Složka komunikace Matlab-Unity
  - ClientMatlab.mat……zdrojový soubor pro komunikaci s UNITY
  - ServerUnity.css……..C# zdrojový kód pro komunikaci
  - ZpracovaniNamerenychDatMatlabUnity.zip…soubor s projektem LabView

- Složka Modelica
  - UkazkaTank.Mod……..……model pro ukázku
  - VymenikovaStaniceDEMO …Simulace výměníkové stanice DEMO scénář
  - VymenikovaStanicePIRegulator....Simulace výměníkové stanice s PI scénář

- Složka Diplomová Práce
  - Diplomová Práce.pdf…….Elektronická verze diplomové práce
  - Procesní schéma.pdf……PDF verze procesního schéma