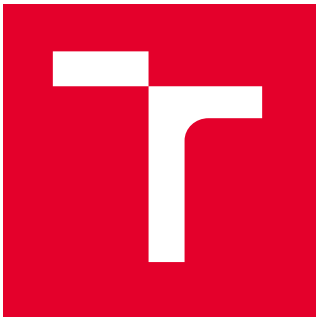
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2023 Tomáš Pečinka



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

DIGITÁLNÍ OBÁLKA PRVKŮ VÝROBNÍ LINKY

AN ASSET ADMINISTRATION SHELL OF INDUSTRY 4.0 COMPONENTS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Pečinka**

AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Marcoň, Ph.D.**

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Tomáš Pečinka ***ID:*** 230154

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Digitální obálka prvků výrobní linky

POKYNY PRO VYPRACOVÁNÍ:

Mnoho výrobních společností chce investovat do digitálních řešení a vyvíjet nové obchodní modely. Otázkou však je, do jakých řešení investovat, aby byla kompatibilní se systémy, které používají jeho zákazníci, partneři a dodavatelé. Proto byla definována v oblasti Průmyslu 4.0 tzv. digitální obálka „Asset Administration Shell“ (AAS). Ta poskytuje základ pro vývoj a používání jednotných a otevřených standardů Průmyslu 4.0. Cílem projektu je vytvořit AAS vybraných prvků výrobní linky.

1) Nastudujte principy Průmyslu 4.0, RAMI model a teoretický základ AAS.
2) Seznamte se se strukturou AAS a s možnostmi jeho vytvoření. Zaměřte se na AAS package explorer, který používá UML, případně na jinou softwarovou podporu pro vytvoření AAS.
3) Navrhněte demonstrační úlohu, kde vybrané prvky výrobní linky budou mít svůj AAS a komunikace mezi nimi bude probíhat pomocí vhodného komunikačního protokolu.
4) Realizujte příklad demonstrátoru části výrobní linky s AAS.
5) Testujte navržené řešení, popište dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] ARM, Jakub, Tomas BENESL, Petr MARCON, et al. Automated Design and Integration of Asset Administration Shells in Components of Industry 4.0. Sensors [online]. 2021, 21(6), 2004. Dostupné z: doi:10.3390/s21062004

[2] Details of the Asset Administration Shell - Part 1 (Version 3.0RC01). Federal Ministry for Economic Affairs and Energy, listopad 2020.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 22.5.2023

Vedoucí práce: doc. Ing. Petr Marcoň, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce se zaměřuje na decentralizované průmyslové řízení s využitím Asset Administration Shell (AAS). Teoretická část práce seznámí čtenáře s pojmy z oblasti Průmyslu 4.0 a přiblíží čtenáři strukturu a způsob vytváření AAS. Taktéž je popsána rešerše dostupných vývojových prostředí pro tvorbu AAS a popis komunikace mezi AAS. Druhá polovina práce je věnována návrhu demonstrační úlohy, která slouží k názorné ukázce decentralizovaného řízení v praxi. Je zde popsán návrh hardwarové i softwarové části úlohy a přístup k vytváření jednotlivých AAS. Výsledná demonstrační úloha se skládá ze dvou 3D tiskáren, objednávkového e-shopu a chytrého skladu. AAS pro tyto komponenty jsou vytvořeny v programovacím jazyce Python.

KLÍČOVÁ SLOVA

AAS, Průmysl 4.0, Asset Administration Shell, MQTT, Python, aktivní část AAS

ABSTRACT

This bachelor thesis focuses on decentralized industrial control using Asset Administration Shell (AAS). The theoretical part of the thesis introduces the reader to the concepts of Industry 4.0 and explains the structure and methods of creating AAS. It also describes a research of available software platforms for AAS development and a description of communication between AAS. The second half of the paper is devoted to the design of a practical demonstrator that serves to illustrate decentralized control in practice. The design of the hardware and software parts of the demonstrator and the approach to creating individual AASs are described. The resulting demonstrator consists of two 3D printers, an customer e-shop and a smart warehouse. AAS for these components are created in programming language Python.

KEYWORDS

AAS, Industry 4.0, Asset Administration Shell, MQTT, Python, active part of AAS

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

PEČINKA, Tomáš. *Digitální obálka prvků výrobní linky*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2023, 55 s. Bakalářská práce. Vedoucí práce: doc. Ing. Petr Marcoň, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Tomáš Pečinka
VUT ID autora:	230154
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Digitální obálka prvků výrobní linky

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/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu doc. Ing. Petru Marcoňovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

V posledních letech investuje stále více firem do modernizace výroby v souladu se čtvrtou průmyslovou revolucí. Díky tomu vznikají nové postupy a dochází ke kvalitnější, rychlejší a efektivnější výrobě. Stále více firem také požaduje plně automatizovanou výrobu bez potřeby lidského zásahu. Jednou z možností jak tento požadavek splnit je vytvoření nové, decentralizované sítě, ve které se budou jednotlivé komponenty rozhodovat nezávisle na ostatních a budou komunikovat napříč všemi vrstvami průmyslové pyramidy. K tomu je ale potřeba vhodná „digitální obálka“, která bude obsahovat data o své komponentě a bude zde rozhodovací a komunikační algoritmus. Jednou z možných variant zmíněné digitální obálky je tzv. Asset Administration Shell (AAS), což je varianta digitálního dvojčete, která má přesně definovanou vnitřní strukturu, kde jsou uložena digitalizovaná data.

Tato bakalářská práce se věnuje vytváření již zmíněných AAS. Při návrhu AAS je postupováno dle elektrotechnické asociace ZVEI, která je s vytvořením konceptu AAS úzce spojena.

První kapitola má za cíl uvést čtenáře do problematiky průmyslu 4.0 a vytváření pasivní i aktivní části AAS. Dále jsou popsány typy AAS a možnosti vytvoření AAS s využitím několika počítačových programů. Ve druhé kapitole je popsána komunikace mezi AAS a komunikace mezi AAS a assetem.

Třetí kapitola se zabývá návrhem fyzické i programové části demonstrační linky, která má sloužit k názorné ukázce možného využití AAS v praxi. Cílem této práce je vytvořit demonstrační úlohu, která přiblíží výhody a nevýhody decentralizovaného řízení v průmyslové praxi. K tomu je potřeba vytvořit několik vzorových AAS, které budou vhodně komunikovat. Návrh je prováděn co nejvíce obecně a modulárně z důvodu možné budoucí znovupoužitelnosti i na jiné průmyslové linky.

Navržený software je otestován pomocí testovacích scénářů. Ve čtvrté kapitole jsou tyto scénáře popsány a jsou zde uvedeny výsledky testování. Následuje zhodnocení navrženého řešení a souhrn navržených vylepšení práce.

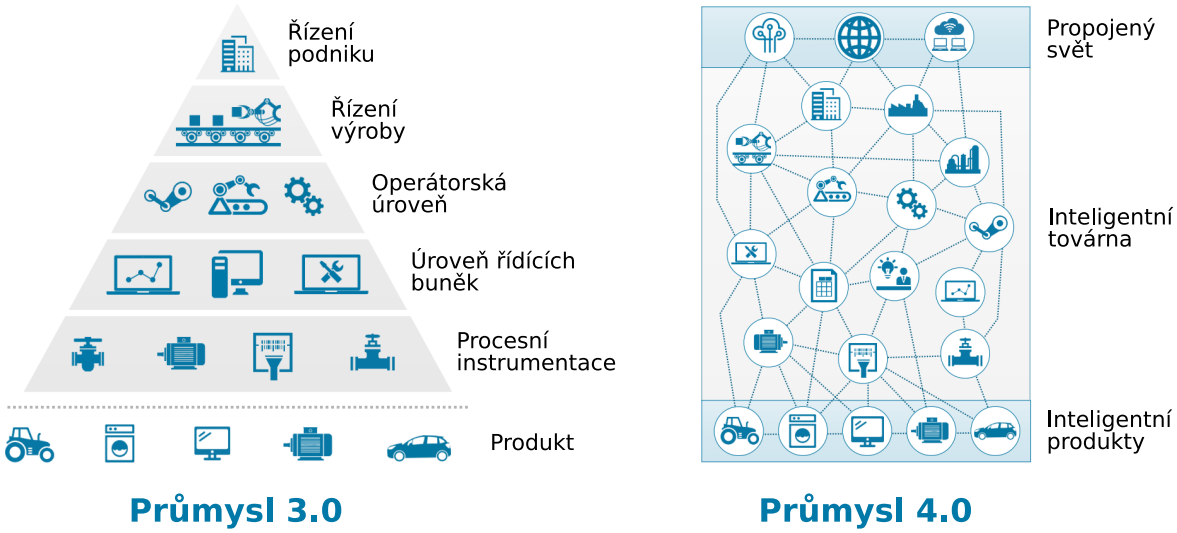
1 Úvod do Průmyslu 4.0 a AAS

1.1 Průmysl 4.0

Průmysl 4.0 definuje inteligentní propojení průmyslových strojů, řídících procesů a technologií pomocí informačních a komunikačních systémů. První vize o vývoji průmyslu tímto směrem byla představena německou vládou na Hannoverském veletrhu již v roce 2011. Oficiálně však byla zapsána až v roce 2013 jako „Industrie 4.0“. [1]

V současné době se průmysl nachází na přelomu třetí a čtvrté průmyslové revoluce. Třetí průmyslová revoluce (Průmysl 3.0) se vyznačuje zavedením částečné automatizace s využitím programovatelných automatů a řídících prvků. Rozdílný přístup Průmyslu 3.0 a Průmyslu 4.0 k řízení a uspořádání prvků průmyslové výroby je zobrazen na obrázku 1.1. [1]

Průmysl 4.0 prezentuje nový přístup k průmyslové automatizaci, jehož cílem je dosáhnout automatizované výroby bez potřeby zásahu operátora. K tomu by mimo jiné měly sloužit i tzv. *inteligentní továrny*, které mohou být vytvořeny a znovu rozebrány z modulárních prvků, což přispěje k flexibilitě výroby. V takovýchto továrnách budou vznikat *inteligentní produkty*, které budou identifikovatelné, lokalizovatelné a budou obsahovat svoji výrobní historii. Vzájemným propojením inteligentních továren v rámci jedné globální sítě vzniknou nové vztahy mezi výrobci, dodavateli a zákazníky. Tyto změny přispějí mimo jiné i k šetrnějšímu nakládání s energiemi a drahými surovinami, efektivnějšímu využití logistiky a snížení ekonomických nákladů na výrobu. [1]



Obr. 1.1: Porovnání struktur Průmyslu 3.0 a 4.0. (inspirace z literatury[4])

1.1.1 Charakteristiky Průmyslu 4.0

Průmysl 4.0 může být definován následujícími principy:

- Výrobní procesy jsou v rámci celé podnikové pyramidy propojeny pomocí vertikálně a horizontálně integrovaných informačních systémů. [1]
- Řízení a interakce s průmyslovými prvky na dálku pomocí internetového připojení, vzájemná konektivita mezi zařízeními. [1]
- Samostatné výrobní jednotky jsou nahrazeny automatizovanými výrobními linkami.
- Virtuální prototypy a návrhy jsou upřednostněny před fyzickými. [1]
- Flexibilní výroba, při které se do procesu vývoje produktu zapojuje několik společností. [2]
- Stroje, činící autonomní rozhodnutí v reálném čase na základě dat ze senzorů a požadavků od řídícího systému, zvyšující efektivitu výroby. [1]
- Optimalizace logistiky pomocí algoritmů, které sestaví efektivní trasu a obstarají potřebný materiál. [2]
- Efektivní využití přírodních zdrojů a recyklace použitých materiálů lze zvážit již při návrhu výrobku a v případě potřeby lze upravit ještě před zahájením výroby. [2]

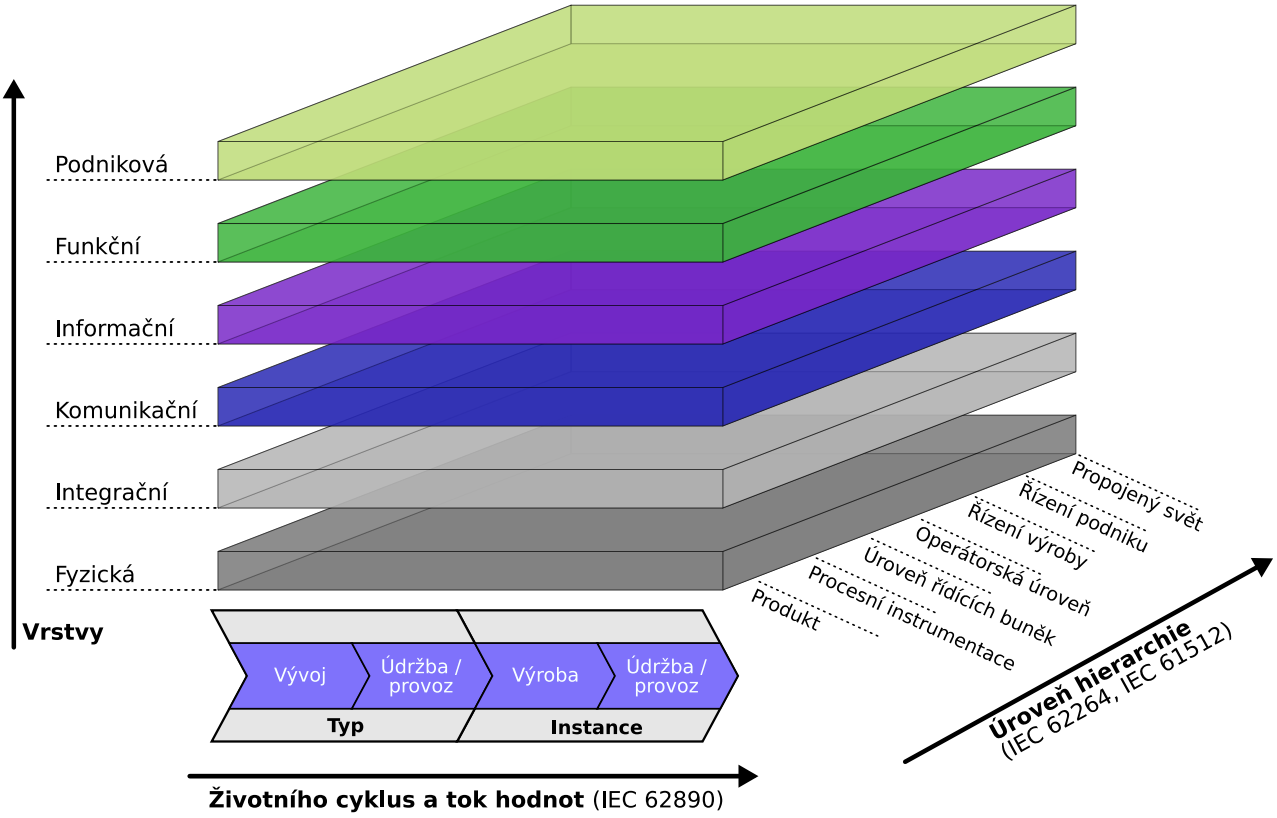
1.2 Referenční architektonický model pro Průmysl 4.0 (RAMI 4.0)

Referenční architektonický 3D model pro Průmysl 4.0 (RAMI 4.0) je jeden ze základních prvků znázorňující nejdůležitější aspekty Průmyslu 4.0. Model RAMI 4.0 definuje architekturu, tedy způsob, jakým se jednotlivé prvky propojují, jak spolu komunikují a jakým způsobem se integrují. Zároveň také určuje způsob, jakým se budou data přenášet mezi jednotlivými úrovněmi a komponentami v průmyslových procesech. Funkcí RAMI modelu je poskytnout standardizovaný referenční rámec pro architekturu a integraci průmyslových systémů. To znamená, že všichni výrobci a uživatelé mohou používat stejný model pro vytváření a propojování průmyslových zařízení. Díky tomu je snadnější integrovat a propojovat různé technologie, což zvyšuje efektivitu, rychlost a kvalitu průmyslové výroby.[3]

Z obrázku 1.2 vidíme, že model RAMI 4.0 je tvořen třemi osami. Osa hierarchické úrovně vychází z norem IEC 62664 a IEC 61512 a znázorňuje jednotlivé úrovně tak, jak je známe z průmyslové pyramidy na obrázku 1.1. Aby model vyhověl požadavkům Průmyslu 4.0, byla osa hierarchické úrovně rozšířena o nové úrovně „produkt“ a „propojený svět“. Osa životního cyklu a toku hodnot vychází z normy IEC 62890

(řízení životního cyklu). Dále rozlišujeme, zda se jedná o návrh, nebo o výrobu. Návrhový druh *Typ* se používá při návrhu výrobku a při vytváření prototypů. V okamžiku, kdy produkt přejde z prototypování do výroby, se z *Typu* stává *Instance*. Svislá osa popisuje rozdělení fyzického zařízení na šest vrstev, které jsou později digitalizovány (tzv. virtuální mapování zařízení). [3]

Pomocí zmíněných tří os je umožněno klasifikovat veškeré objekty v Průmyslu 4.0, jako např. průmyslové výrobní zařízení, podle předepsaného a jednotného modelu. Referenční architektonický model tak umožňuje postupnou migraci ze současného stavu průmyslové automatizace do světa Průmyslu 4.0. [3]



Obr. 1.2: (inspirace z literatury [3])

1.3 Asset

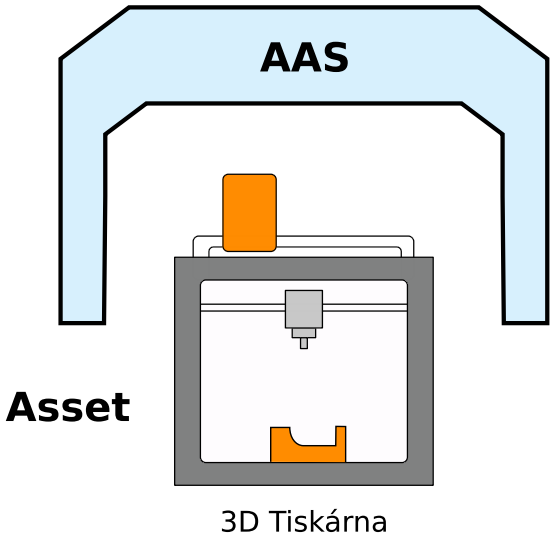
Pojmem asset jsou označeny veškeré fyzické i digitální prvky, které komunikují v rámci Průmyslu 4.0 (stroje a jejich komponenty, materiály, produkty, výkresy, schémata zapojení, smlouvy, objednávky, lidský operátor, služby). [5]

Snahou Průmyslu 4.0 je dosáhnout interoperability dvou a více assetů. Pojmem interoperabilita rozumíme vzájemnou výměnu informací a následné kooperativní vykonání činnosti na základě takto získaných informací. [6]

1.4 Asset Administration Shell (AAS)

Asset Administration Shell (AAS) je jedna z klíčových komponent Průmyslu 4.0, která poskytuje standardizovaný popis digitálního nebo fyzického assetu. Jedná se o popis, který je definován podle striktních pravidel a není závislý na technologii, ani na daném zařízení. Motivací pro vyvinutí AAS byla absence standardu v oblasti průmyslové automatizace, který by byl nezávislý jak na výrobci, tak na technologii. [8]. Příklad AAS s assetem v podobě 3D tiskárny je uveden na obrázku 1.3.

AAS musí obsahovat pasivní část, ve které jsou uloženy informace o assetu a AAS. Pasivní část může být rozšířena o aktivní část umožňující komunikaci s ostatními AAS. [11]



Obr. 1.3: Příklad AAS s assetem

1.4.1 Identifikátory

Identifikátory jsou nezbytným a efektivním nástrojem k unikátní identifikaci prvků Průmyslu 4.0. Identifikace je vyžadována přinejmenším pro [11]:

- Asset Administration Shell
- Asset (identifikátor uveden v zapouzdřujícím AAS)
- Submodel (platí pro šablonu i instance)

Pro identifikaci máme k dispozici tři druhy identifikátorů, a to identifikátor údajů o mezinárodní registraci (IRDI), definován normou ISO29002-5, ISO IEC 6523 a ISO IEC 11179-6, internacionalizovaný identifikátor zdrojů (IRI), definován podle RFC 39862 a vlastní identifikátory, jako např. univerzálně jedinečný identifikátor (UUID), které si definuje výrobce podle svých interních potřeb. Identifikace neslouží pouze jako prostředek pro jednoznačné rozlišení prvků v AAS, ale také k vytváření vazeb (odkazů) mezi šablonami a instancemi submodelů, metamodelů a elementů AAS. [11]

1.4.2 Typy AAS

V současné době rozdělujeme AAS do tří kategorií podle aktivity, se kterou komunikují v síti Průmyslu 4.0. Rozdíl mezi jednotlivými typy AAS lze ilustrovat jejich zobrazením v modelu RAMI 4.0 na obrázku 1.4.

Typ 1 AAS (pasivní)

AAS typu 1, někdy označované AAS jako soubor, obsahuje pouze pasivní část, kterou tvoří serializované soubory typu JSON nebo XML obsahující statické informace o assetu, který zapouzdřují. Tyto soubory si jednotlivé firmy sdílí mezi sebou a vyměňují si tak informace o AAS. [8]

Typ 2 AAS (reaktivní)

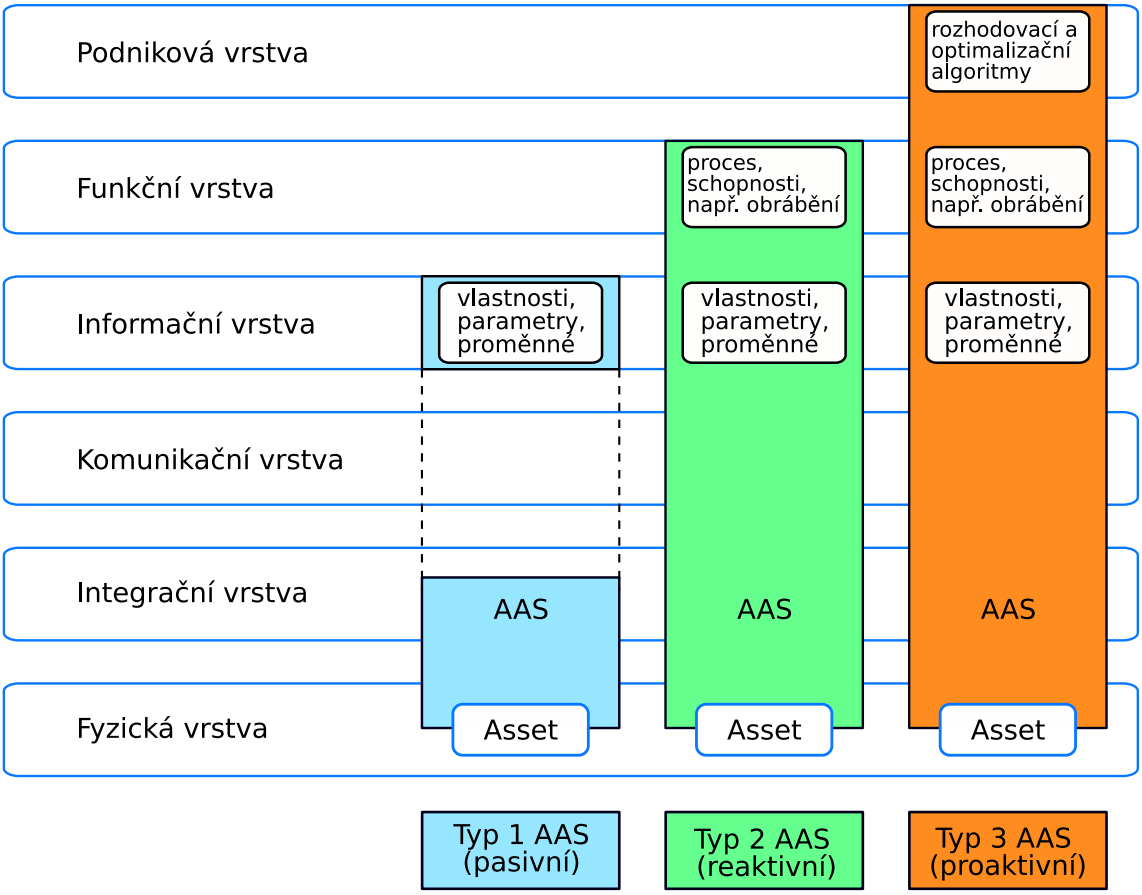
Reaktivní AAS je AAS obsahující pasivní část, která je umístěna na serveru jako tzv. runtime instance obsahující rozhraní pro programování aplikací (API). Reaktivní AAS obsahuje statická data obsažená v pasivní části AAS, ale oproti pasivnímu AAS je schopné interakce s ostatními reaktivními a proaktivními AAS. Pomocí reaktivních AAS je možné v aktuálním čase zobrazit data ze senzorů, dostupnost výrobků, efektivitu výroby apod. [8]

Typ 3 AAS (proaktivní)

Proaktivní AAS, v některé literatuře označeno pouze jako aktivní, rozšiřuje pasivní AAS o aktivní chování, které spočívá ve schopnosti samostatné komunikace, rozhodování a vyjednávání. Aktivita AAS je zajištěna na základě znalosti statických

informací o daném assetu, které poskytuje pasivní část AAS. [8][9]

Rozdíl mezi AAS lze ilustrovat vložením do systému RAMI 4.0. Typ 1 a Typ 2 AAS obsahují pouze popis vlastností, parametrů a proměnných ve formě submodelů. K těmto hodnotám lze přistupovat, číst a manipulovat s nimi. AAS ale nemůže převzít iniciativu a rozhodovat se na základě vnějších požadavků. Proaktivní AAS, tedy Typ 3, je schopné autonomně interagovat s ostatními AAS a má mechanismy pro rozhodování a vyjednávání. [9]



Obr. 1.4: Typy AAS zobrazené ve vrstvové ose RAMI 4.0 (inspirace z literatury [9])

1.4.3 Pasivní část AAS

Pasivní část AAS obsahuje veškeré informace o assetu, který zapouzdřuje. Jedná se o informace od výrobce assetu, informace od provozovatele, technickou dokumentaci, servisní manuály aj. V tabulce 1.1 jsou shrnuty požadavky na tvorbu pasivní části AAS. [7]

Vnitřní struktura pasivní části AAS je tvořena hlavičkou (header) a tělem (body). Hlavička obsahuje identifikátor samotného AAS, ale stejně tak i assetu, který zapouzdřuje. Dle německé asociace ZVEI by hlavička měla obsahovat požadovanou úroveň zabezpečení a odkaz na tělo AAS, kde jsou uvedeny další bezpečnostní atributy (bezpečností klíče, ověřování, atd.). Pomocí identifikátorů uvedených v hlavičce AAS je umožněno ostatním entitám přistupovat k informacím o typu a instanci AAS. Tělo pasivní části AAS je tvořeno ze submodelů. Submodely jsou strukturované bloky, které popisují různé aspekty AAS. [10]

Každý submodel se vztahuje k přesně definované doméně. Nejčastější jsou submodely popisující technologii, kterou asset disponuje, bezpečnost, energetickou účinnost, identifikaci AAS i assetu a monitorování podmínek. Submodely se mohou standardizovat a stát se tzv. submodel typem pro vytváření nových AAS.[6]

Submodel je dále rozčleněn na jednotlivé elementy, např. elektrický motor se submodelem s názvem „operační parametry“ bude obsahovat elementy „rychlost otáčení“ a „kroutící moment“. Jak již bylo řečeno, každý submodel musí obsahovat svůj vlastní identifikátor. Používání identifikátorů u submodel elementů je doporučené. [11]

1.4.4 Aktivní část AAS

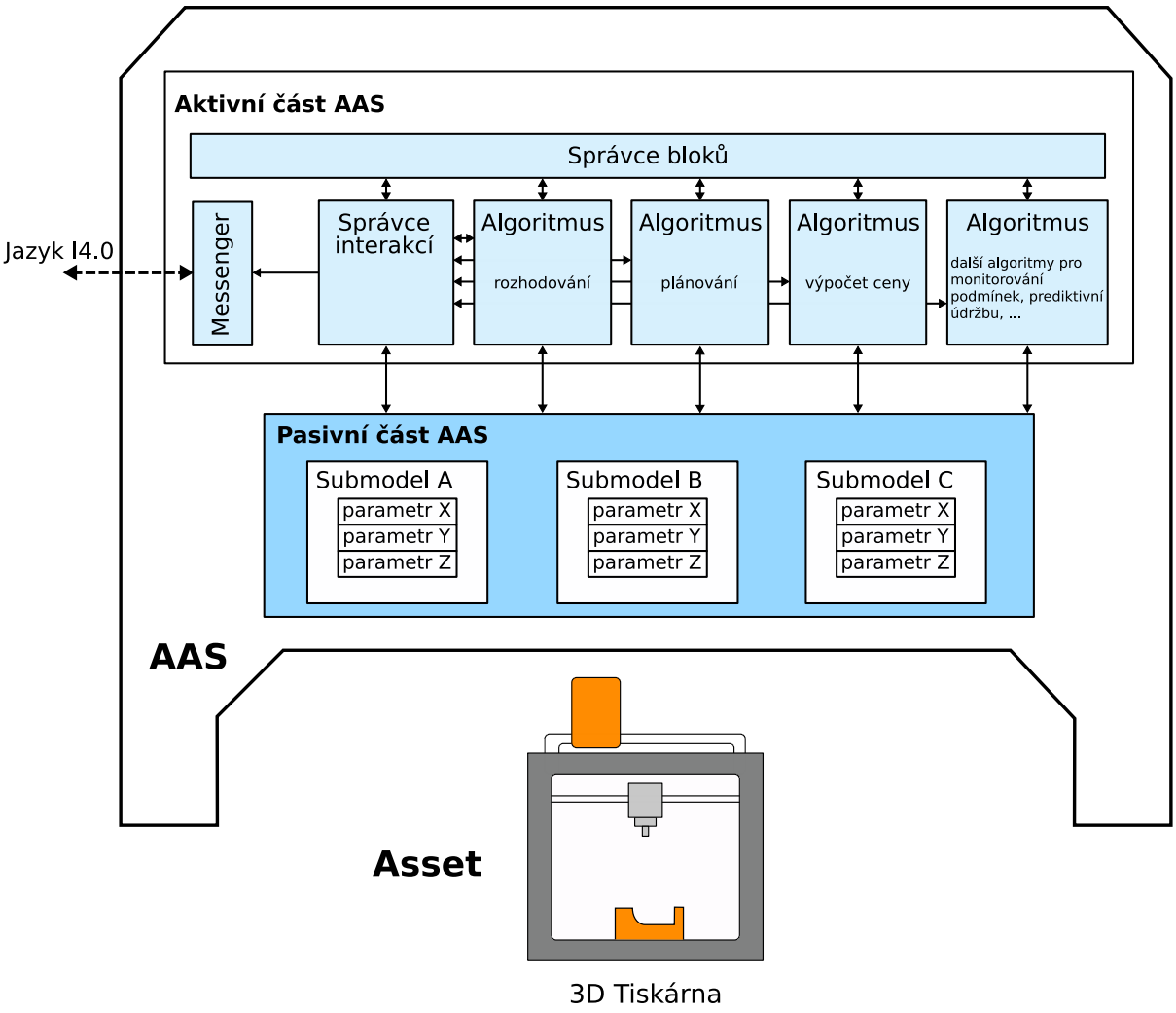
Aktivní část AAS přináší možnost komunikace s ostatními AAS, rozhodování, vyjednávání, plánování, prediktivní údržbu a další aktivitu, díky které je AAS schopné reagovat v reálném čase na změny a zefektivnit tak výrobu. Aktivní část dělí AAS na dva základní typy, žadatele služeb a poskytovatele služeb. Žadatel služeb může být například polotovar, který požaduje po strojích informace o tom, zda splňují určité parametry pro jeho zpracování. Také to ale může být samotný stroj, který po operátorovi požaduje údržbu, nebo výměnu pracovního nástroje. Poskytovatel služeb může být ten samý stroj, jenž poskytuje hotové výrobky jiným strojům. Z uvedeného příkladu je vidět, že dělení na poskytovatele a žadatele služeb se může měnit podle toho, s kým AAS zrovna komunikuje, není tedy předem pevně definované. [9]

Aktivní část AAS obsahuje bloky správce interakcí, messenger, správce bloků, vnitřní algoritmy pro rozhodování, plánování, výpočet ceny aj.

Tab. 1.1: Přehled požadavků na tvorbu pasivní části AAS (inspirace z literatury [6])

Index	Požadavek
1	AAS by měla přijímat vlastnosti z různých technických domén ve vzájemně se doplňujících odlišných submodelech, které lze udržovat nezávisle na sobě.
2	AAS by měla být schopna začlenit parametry z široké škály technických oblastí a identifikovat, ze které domény pocházejí.
3	Pro nalezení definic v rámci každé příslušné technické oblasti by měly být použity různé procesní modely, které splňují požadavky norem, specifikací konsorcií a specifikací výrobců.
4	Různé AAS musí být schopny vzájemně se odkazovat. Zejména prvky AAS by měly být schopny splnit roli "kopie" odpovídajících prvků z jiného AAS.
5	Jednotlivé AAS by měly být při zachování své struktury sloučeny do celkového AAS.
6	Identifikace assetů, AAS, parametrů a vzájemných vazeb se provádí pomocí omezenou sadu identifikátorů (IRDI, URI a GUID), které v co největší míře nabízejí globální jedinečnost.
7	Asset Administration Shell by měla umožnit získávání alternativních identifikátorů, jako jsou GS1 a GTIN identifikátory, jako náhrada za identifikátor assetu (dereference).
8	Pasivní část AAS se skládá z hlavičky a těla.
9	Hlavička obsahuje informace o identifikaci.
10	Tělo obsahuje informace o příslušném assetu (assetech).
11	Informace a funkcionalita AAS jsou přístupné prostřednictvím standardizovaného API.
12	AAS musí mít unikátní ID.
13	Asset musí mít unikátní ID.
14	Průmyslové zařízení je také asset, má AAS a je dohledatelné pomocí ID.
15	Typy a instance musí být identifikovatelné.
16	AAS může obsahovat reference na jiné AAS.
17	Doplňkové parametry, např. údaje specifické pro výrobce, musí být povoleny.
18	Pro každou AAS musí být definován spolehlivý minimální počet parametrů.
19	Parametry a další prvky informací v AAS musí být vhodné pro typy a instance.
20	Musí existovat možnost hierarchického a počitatelného členění parametrů.
21	Parametry AAS musí být schopny odkazovat na jiné parametry, a to i v jiných AAS.
22	Parametry AAS musí být schopny odkazovat na informace a funkce uvnitř AAS.

Struktura aktivní části AAS je zobrazena na obrázku 1.5. Správce interakcí obsahuje předdefinované sady interakcí, z nichž nejznámější je tzv. nabídkový proces (bidding process). Správce interakcí komunikuje s messengerem pomocí serializovaných zpráv ve tvaru JSON. Messenger tvoří komunikační rozhraní a zajišťuje komunikaci pomocí vhodného protokolu. Správce bloků organizuje bloky a přiřazuje jim roli žadatele služeb nebo poskytovatele služeb. [9]



Obr. 1.5: Ukázka AAS popisujícího 3D tiskárnu (inspirace z literatury [9])

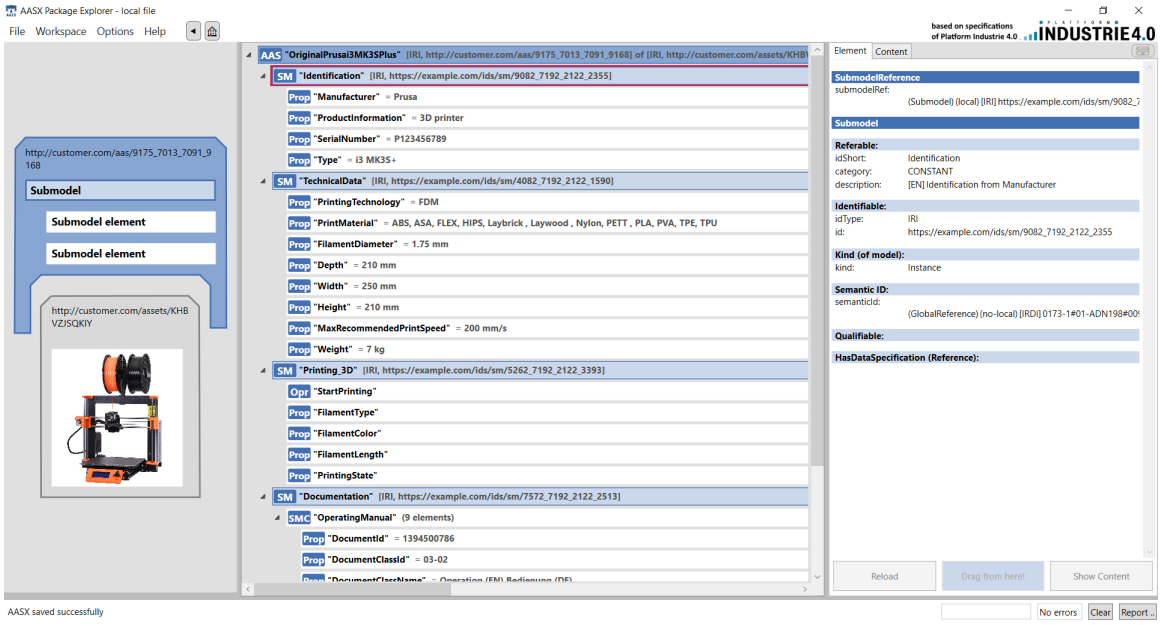
1.5 Možnosti implementace AAS

V současné době je k dispozici několik softwarových nástrojů, které usnadňují vytváření nových AAS. Vybrané nástroje jsou popsány v podkapitolách níže.

1.5.1 AASX Package Explorer

AASX Package Explorer je open-source nástroj s grafickým uživatelským rozhraním určený pro implementaci a názornou demonstraci prvků AAS. AASX Package Explorer již ve své architektuře obsahuje server pro přenos reprezentativního stavu (REST) i server pro jednotnou architekturu řízení objektových procesů (OPC UA), do kterého může uživatel nahrát vlastní soubor s příponou .aasx. V případě požadavku na současný běh několika AASX entit je nutné použít doplňkový software AASX Server.[12]

Ukázka vytvoření AAS pro 3D tiskárnu včetně submodelů popisujících výrobní a technické parametry tiskárny je na obrázku 1.6.

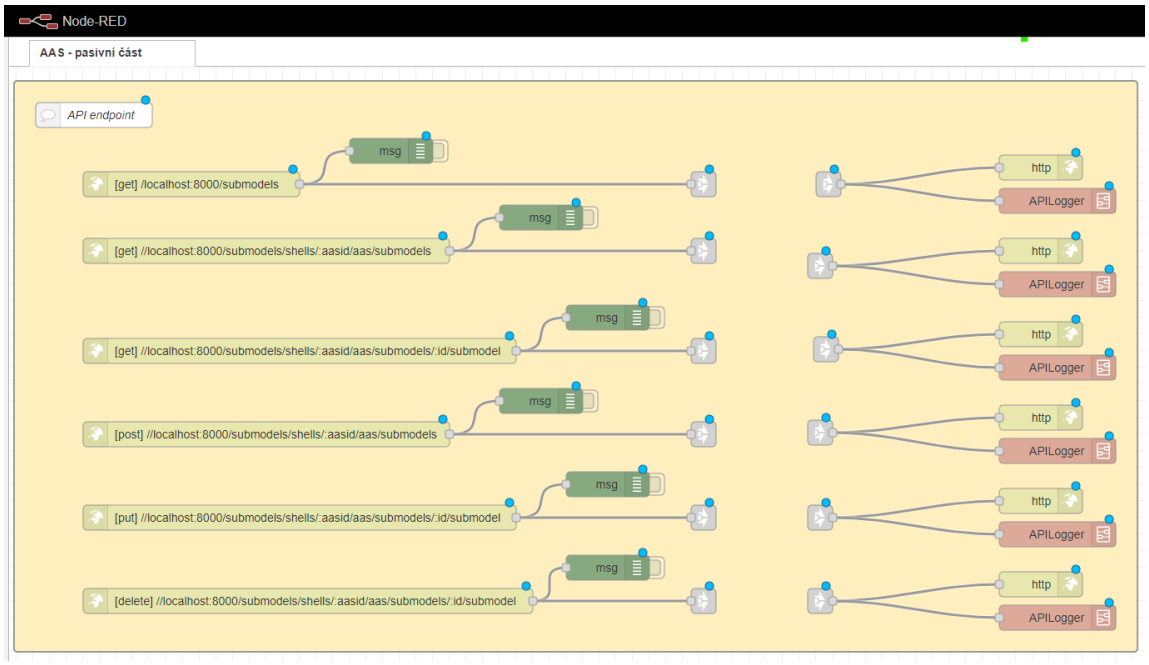


Obr. 1.6: Vývojové prostředí AASX Package Explorer

1.5.2 NOVA Asset Administration Shell (NOVAAS)

NOVA Asset Administration Shell (NOVAAS) je open-source software pro implementaci prvků AAS vyvinutý organizací NOVA School of Science and Technology. Uživatel vytváří AAS pomocí vizuálního programovacího jazyku Node-RED, jak

je zobrazeno na obrázku 1.7. Programovací jazyk Node-RED je založen na principu programování známém jako „tok dat“. [13]



Obr. 1.7: Vývojové prostředí NOVAAS

1.5.3 Eclipse BaSyx Python SDK

Eclipse BaSyx Python SDK, dříve známý pod názvem PyI40AAS, je knihovna implementovaná do programovacího jazyka Python, která se zaměřuje na implementaci AAS v souladu s metamodelem a rozhraním definovaném v dokumentu „Details of the Asset Administration Shell“. V rámci knihovny je uživateli umožněno vytváření prvků AAS, čtení a zapisování do souborů s příponou *.aasx*, serializace vytvořených AAS na soubory JSON a XML (včetně zpětné deserializace), ukládání AAS do databáze a ověření, zda je vytvořené AAS v souladu s JSON/XML AAS předpisem.[14]

2 Komunikace

Průmysl 4.0 potřebuje při zpracovávání požadavků rychlou a robustní komunikaci, ať už se jedná o komunikaci mezi jednotlivými AAS nebo mezi AAS a assetem. Následující kapitola popisuje dostupné komunikační protokoly a proces vyjednávání mezi AAS v souladu s Jazykem I4.0

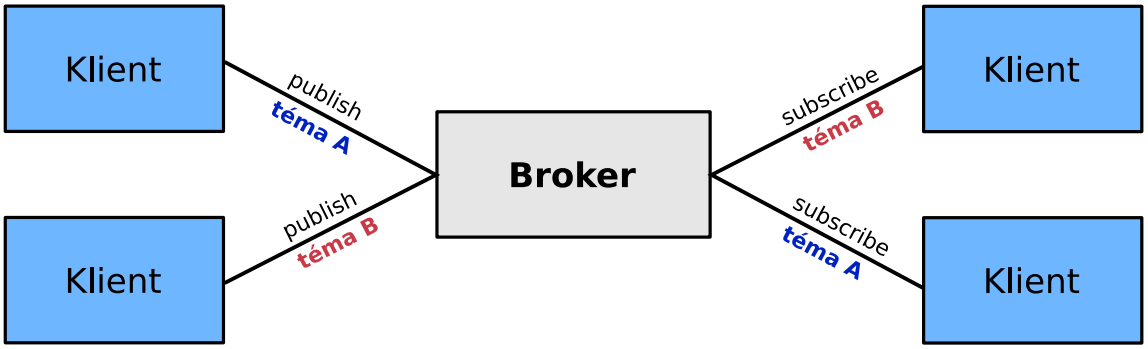
2.1 Komunikace mezi AAS

Komunikace mezi AAS probíhá nejčastěji pomocí OPC UA nebo MQ Telemtry Transport (MQTT). V následujících dvou podkapitolách budou popsány jejich výhody a nevýhody.

2.1.1 MQTT

MQ Telemetry Transport (MQTT) byl vynalezen v roce 1999 jako protokol s minimálními nároky na šířku pásma a jednoduchou implementací. Protokol byl původně používán v proprietárních vestavných systémech, nyní je často využíván pro komunikaci v oblasti internetu věcí (IoT). Protokol byl od roku 1999 několikrát modernizován, v současné době je k dispozici verze MQTT 5 standardizovaná dle ISO/IEC 20922:2016. Protokol MQTT je založen na principu publisher-subscriber, tedy zveřejňování a odebírání dat pomocí centrálního prvku, nazvaného broker. Diagram komunikace typu publisher-subscriber je zobrazen na obrázku 2.1. Zařízení v komunikační síti může zároveň sdílet a odebírat neomezené množství zpráv.[18]

Autoři experimentu popsaném v literatuře [16] zjistili, že při používání MQTT v komunikační síti s více AAS nebyl provoz bezproblémový, objevovaly se problémy ve vyjednávání více služeb v jednom časovém okamžiku a následné zahlcení domény. Neuspokojivá byla také časová odezva, která se pohybovala v řádu sekund.[16].



Obr. 2.1: Architektura publisher-subscriber

2.1.2 OPC UA

OPC Unified Architecture (OPC UA) je komunikační architektura používaná v průmyslové automatizaci od vestavěných systémů až po výrobní informační systémy (MES) a systém pro plánování podnikových zdrojů (ERP). OPC UA je založena na principu server-klient. Neobsahuje tedy žádný centrální prvek, který by zajišťoval přenos zpráv. Absence centrálního prvku přispívá k použití OPC UA v Průmyslu 4.0, kde je klíčovým prvkem decentralizace. Dle experimentu popsaném v literatuře [16] se v praxi při použití OPC UA v síti s více zařízeními projevila vyšší latence. Tato nevýhoda byla do určité části eliminována s využitím lokálních serverů pro vyhledávání (Local Discovery Servers, LDS) a rozšířením pro vícesměrové rozesílání zpráv (Multicast Extension, ME).[15][16]

Verze OPC v1.04 přináší rozšíření PubSub, které umožňuje komunikaci na principu publisher-subscriber. S rozšířením PubSub jsou zařízení schopna zveřejňovat a odebírat data pomocí softwarové infrastruktury, která výměnu zpráv zprostředkuje. Tato infrastruktura může a nemusí obsahovat centrální prvek (broker). V případě použití centrálního prvku je princip výměny zpráv podobný jako u MQTT. Zpráva, která je sdílena, dorazí k centrálnímu prvku, brokerovi, a ten ji pošle všem účastníkům komunikace, kteří ji odebírají. U varianty bez použití brokera je centrální prvek nahrazen síťovými zařízeními (směrovače, přepínače, mosty) a vhodným peer-to-peer protokolem, nejčastěji UDP. Výhodou této softwarové infrastruktury je snížení latence a režie, použití pouze standardního síťového vybavení bez dalších softwarových komponent jako je broker a možnost vícesměrového adresování (při použití protokolu UDP).[17]

2.2 Komunikace mezi AAS a assetem

Komunikace mezi AAS a assetem je závislá na konkrétním typu assetu a jeho komunikačních možnostech. V případě, že AAS není implementováno přímo v assetu, je nutné zvolit typ komunikace vhodný jak pro AAS, tak pro asset. Může se jednat například o průmyslový ethernet, ale i o bezdrátovou komunikaci v podobě MQTT či Hypertextového přenosového protokolu (HTTP).

2.3 Jazyk I4.0

Industrie 4.0 language (Jazyk I4.0) je definován v normě VDI/VDE 2193. Norma se skládá ze dvou celků. První část zmíněné normy, VDI/VDE 2193-1, definuje strukturu přenášených zpráv mezi AAS. Sestavování zpráv dle normy probíhá v aktivní části AAS, konkrétně v bloku správce interakcí. Správce interakcí předá vytvořenou zprávu bloku s názvem messenger, který ji příslušným komunikačním protokolem rozešle ostatním AAS. Druhá část normy definuje způsob vyjednávání mezi AAS, který je popsán pomocí vyjednávacího protokolu. [9]

2.3.1 Struktura zpráv

Struktura zpráv je dle normy VDI/VDE 2193-1 definována jako soubor JSON, který je tvořen jednotlivými parametry. Zpráva se dělí na rámec a interakční elementy. Rámec zprávy obsahuje několik parametrů, které jsou zobrazeny v tabulce 2.1. Interakční elementy tvoří tělo zprávy, ve kterém jsou uvedeny podrobné informace v závislosti na typu zprávy. [9]

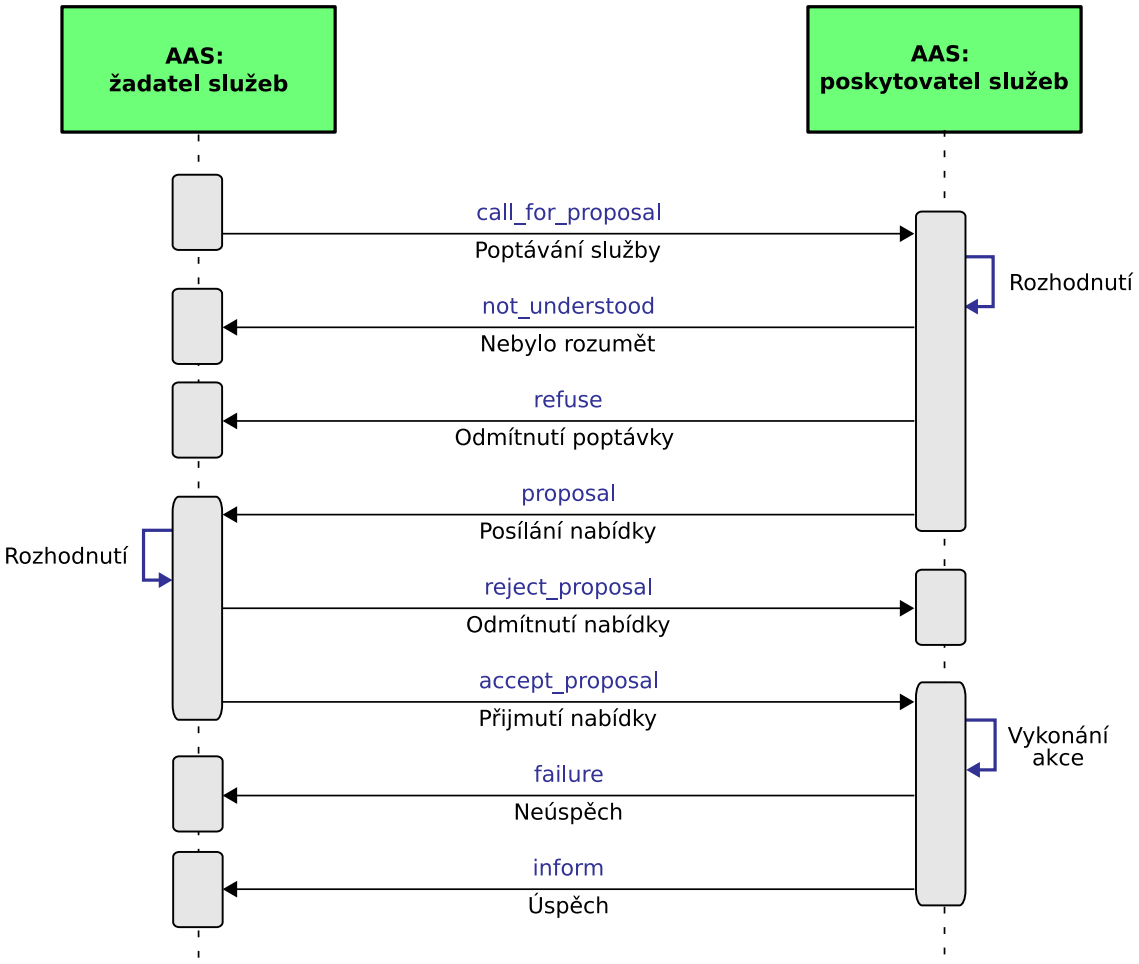
Tab. 2.1: Struktura zprávy dle Jazyka I4.0 (inspirace z literatury [9])

Umístění	**Název použitý v programu**	**Popis**	**Použití**
Interakční elementy	interactionElements	Tělo zprávy	Volitelné
Rámec	type	Typ zprávy	Povinné
Rámec	receiver	Příjemce zprávy	Povinné
Rámec	sender	Odesílatel zprávy	Volitelné
Rámec	conversationId	Identifikátor konverzace	Volitelné
Rámec	messageId	Identifikátor zprávy	Povinné
Rámec	inReplyTo	Odkaz na zprávu, která předchází aktuální zprávě	Volitelné
Rámec	replyBy	Čas, do kterého je nutné odpovědět na zprávu	Volitelné

2.3.2 Vyjednávací protokol

Komunikační protokoly vyjednávání v Průmyslu 4.0 jsou standardizované a obsahují jednoznačnou sémantiku, tedy strukturu komunikace a obsah přenášených zpráv. Sémantika vyjednávání vychází z Jazyka I4.0, přesněji z normy VDI/VDE 2193-2. Vyjednávání mezi poskytovatelem služeb (SP - z angl. service provider) a žadatelem

služeb (SR - z angl. service requester) probíhá pomocí vybrané interakční sady, která je obsažena ve správci interakcí a je známá oběma AAS, které spolu vyjednávají. Nejčastěji se jedná o vyjednávání pomocí nabídkového procesu. Typ a obsah zpráv, které si účastníci v rámci nabídkového procesu vyměňují, je předem definovaný podle toho, zda se jedná o poskytovatele nebo žadatele služeb. Na obrázku 2.2 je zobrazen typický průběh komunikace mezi žadatelem a poskytovatelem služeb. Komunikaci zahajuje žadatel služeb rozesláním poptávky na službu. Pokud je zpráva v korektním formátu, musí se poskytovatel rozhodnout, zda nabídku přijme, nebo odmítne a poslat svojí odpověď zpět žadateli. Žadatel vyhodnotí nabídku a pošle poskytovateli služeb zprávu s rozhodnutím, zda ji přijme či odmítne. V případě kladné odpovědi zahájí poskytovatel služeb požadovanou akci a informuje žadatele o její úspěšnosti. V případě odmítnutí nabídky proces vyjednávání končí a poskytovatel služeb přijímá nové poptávky na službu od žadatelů. [9]



Obr. 2.2: Interakce mezi dvěma AAS podle nabídkového procesu (inspirace z literatury [9])

3 Realizace demonstrační úlohy

Tato kapitola se zabývá návrhem softwarové i hardwarové části demonstrační úlohy s využitím AAS. Návrh AAS vychází z teoretických principů, které byly uvedeny v předchozích kapitolách. V následujících podkapitolách je popsána demonstrační úloha včetně jejích částí, návrh AAS pro demonstrační úlohu a návrh praktického provedení.

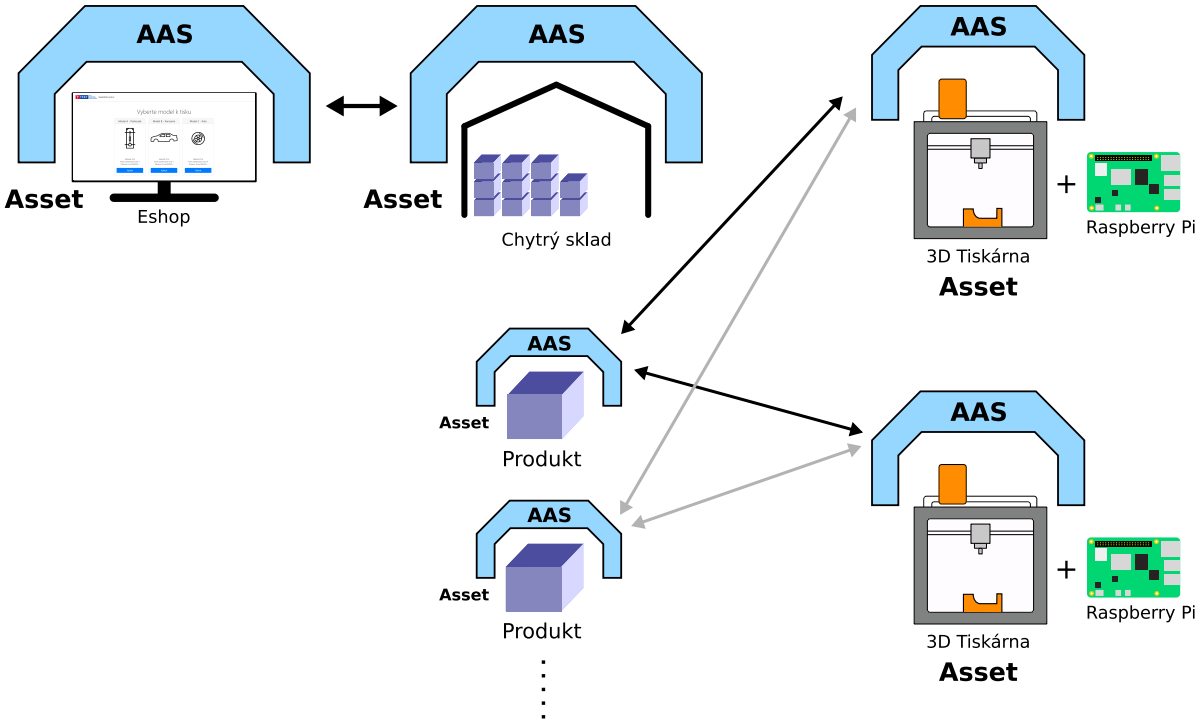
3.1 Cíl demonstrační úlohy

Cílem demonstrační úlohy je vytvořit několik vzorových AAS, které budou popisovat fyzické či digitální prvky a budou mezi sebou vhodně interagovat. Takto vytvořené AAS vytvoří univerzální modulární výrobní jednotku, která bude demonstrovat decentralizované řízení v Průmyslu 4.0. Důraz je kladen především na aktivní část AAS a algoritmy, které obsahuje. Vyjednávací protokol i zprávy, které jsou předávány mezi AAS, jsou implementovány dle normy VDI/VDE 2193.

3.2 Popis demonstrační úlohy

Jako vhodný příklad demonstrační úlohy byl zvolen 3D tisk na zakázku. Jedná se o sestavu 3D tiskáren, objednávkového e-shopu a skladu, ve kterém jsou výtisky uskladněny. Návrh úlohy je zobrazen na obrázku 3.1. V demonstrační úloze je omezen počet tiskáren, jelikož praktická realizace proběhla v laboratoři, kde jsou k dispozici pouze dvě 3D tiskárny. Úloha je ale zcela modulární a přidání dalších 3D tiskáren proběhne bez problémů.

Výrobní linka popsaná v demonstrační úloze má za úkol vytisknout produkt, který si uživatel objednal na webové stránce e-shopu. Ke splnění zmíněného úkolu je zapotřebí několik kroků, jejichž logický sled bude nyní popsán. Prvním krokem je vytvoření objednávky od uživatele. Po objednání tisku zvoleného modelu pošle administrativní obálka e-shopu zprávu administrativní obálce chytrého skladu, který vytvoří novou administrativní obálku pro produkt. Nově vytvořený typ produktu, který je ovšem stále ve fázi digitálního modelu, je nyní schopen vyjednávat s dostupnými 3D tiskárnami. Po dokončení vyjednání je produkt vytisknut na 3D tiskárně, která nabídla nejnižší cenu za tisk a výroba produktu je ukončena.



Obr. 3.1: Návrh demonstrační úlohy

3.3 Životní cyklus komponenty Průmyslu 4.0

Při pohledu na osu životního cyklu v RAMI modelu vidíme, že AAS vzniká již při návrhu komponenty Průmyslu 4.0 a mapuje její celý životní cyklus. Pro účely demonstrační úlohy je životní cyklus detailně řešen pouze pro dynamicky vytvářená AAS. Jedná se o AAS produktů, neboli modelů, které uživatel požaduje vytisknout. Po obdržení potvrzení uživatelské objednávky z e-shopu vytvoří chytrý sklad nové AAS, jehož assetem je pouze digitální model produktu. Tento digitální model produktu je díky svému AAS schopný vyjednat si tisk s dostupnými 3D tiskárnami. Po dokončení tisku je výtisk přiřazen jako nový asset stávajícímu AAS. AAS produktu obsahuje kompletní výrobní historii, která může být využita například pro simulaci, nebo při zlepšování detekce defektů ve výrobě.

3.4 Přístup k AAS

Důležitým prvkem při návrhu demonstrační linky je zvolit správnou strategii přístupu k AAS. Jedna ze zásadních otázek je hostitel AAS, respektive místo, kde budou AAS uložena a spuštěna.

V případě digitálních assetů typu e-shop a chytrý sklad je vhodné, aby byla AAS integrována přímo do softwarové části assetů, tedy vložena jako část kódu do již existujícího programu. Tento způsob nevyžaduje další komunikaci mezi AAS a assetem, jelikož se informace předávají přímo pomocí programových proměnných.

Tento způsob propojení AAS a assetu nelze využít pro fyzické assety, tedy produkty. Jedno z možných řešení je vytvoření nové komponenty, která má na starost vytváření AAS produktů a zároveň je jejich hostitelem. Takovou komponentou je chytrý sklad, který může být spuštěn například na osobním počítači nebo firemním serveru.

Obdobným způsobem je řešeno propojení AAS a 3D tiskáren. Každá 3D tiskárna je připojena k mikrokontroléru, na kterém je spuštěn obslužný operační systém pro tisk. Prvotní vizí bylo společná integrace zmíněného operačního systému a AAS na jednom mikrokontroléru. Tato vize se ovšem ukázala jako nemožná, protože operační systém pro 3D tiskárnu neumožňuje paralelní vykonávání programu s AAS. Jako možné řešení se ukázalo spouštět AAS pro 3D tiskárny na osobním počítači a komunikovat s 3D tiskárnami pomocí API.

3.5 Výběr vhodného hardwaru

Vybraný hardware je závislý na vybrané demonstrační úloze. Při výběru 3D tiskárny je zohledněna jejich dostupnost v laboratořích a proto je výběr omezen.

3.5.1 3D tiskárna

Na trhu existuje několik druhů 3D tiskáren od různých výrobců, nicméně pro demonstrační úlohu byly zvoleny tiskárny typu i3 MK3S+ od výrobce Prusa Research. Tiskárny i3 MK3S+ se vyznačují kvalitními komponenty, podporou širokého spektra materiálů a jednoduchým a přehledným nastavením pro uživatele. Tyto tiskárny již byly zprovozněny v laboratoři na FEKT UTEE, což byl hlavní důvod jejich použití pro bakalářskou práci.

3.5.2 Mikrokontrolér

Pro ovládání tiskárny pomocí vzdáleného rozhraní je nutné k tiskárně připojit vhodný mikrokontrolér. Ten musí mít podporu pro připojení wifi modulu s anté-

nou, nebo obsahovat vestavěnou wifi anténu.

V úvahu přichází několik mikrokontrolérů. Výběr je zúžen pouze na mikrokontroléry výrobce Raspberry Pi Foundation, a to: Raspberry Pi A, A+, B, B+, 2B, 3A+, 3B, 3B+, 4B 1/2/4GB, Zero W/2. Toto zúžení výběru je dáno tiskárnou, respektive výčtem mikrokontrolérů, které tiskárna i3 MK3S+ podporuje. Dalším omezením výběru je situace na trhu s mikročipy, kdy je většina zde vypsaných mikrokontrolérů nedostupných a objednávkové lhůty se pohybují okolo několika měsíců.

Po zohlednění zde uvedených omezení ve výběru bylo vybráno cenově dostupné Raspberry Pi Zero W. Tento model má vestavěnou wifi anténu, 512 MB operační paměti a datový microUSB konektor, pomocí kterého je propojen s 3D tiskárnou.

3.6 Výběr vhodného softwaru

Po zvolení hardwaru bylo nutné rozhodnout, na jaké platformě a pomocí jakých nástrojů budou jednotlivé administrativní obálky vytvořeny. Prvním krokem bylo zvolit vhodný nástroj pro vytváření pasivní části AAS, od něj se odvíjela i volba programovacího jazyka, ve kterém byla následně implementována aktivní část AAS a komunikační protokol. Dále bylo nutné zvolit operační systém pro Raspberry Pi Zero W, pomocí kterého bude možné vzdáleně řídit a monitorovat 3D tiskárnu.

3.6.1 Nástroj pro tvorbu pasivní části

Pro vytváření pasivní části byla zvolena knihovna *Eclipse BaSyx Python SDK*. Oproti konkurenčním nástrojům, uvedených v kapitole 1.5, je knihovna napsaná v jazyce Python a poskytuje tak přímou integraci s aktivní částí a automatizaci vytváření AAS. Tato vlastnost je v demonstrační úloze využita při dynamickém vytváření nových AAS produktů. Knihovna také obsahuje rozsáhlou dokumentaci, ve které jsou uvedeny vzorové příklady pro tvorbu, serializaci a deserializaci z formátu JSON nebo XML a ukládání AAS do databáze.

3.6.2 Komunikační protokol

Pro účely demonstrační úlohy byl zvolen protokol MQTT, zejména kvůli dostupnosti vývojových knihoven, nenáročné implementaci a rozdělení zpráv do jednotlivých témat. Právě princip odebírání a zveřejňování zpráv na určité téma poskytuje přehlednost a škálovatelnost komunikace v demonstrační úloze. Navíc lze díky zveřejňování zpráv na určité téma minimalizovat počet přenášených zpráv. Odběratelé

odebírají pouze témata, která jsou pro ně relevantní a nedostávají tak nadbytečné informace.

MQTT knihovna pro jazyk Python

Paho-MQTT 1.6.1 je knihovna pro Python, která umožňuje komunikaci pomocí MQTT. Základní třídy knihovny jsou:

- třída ***Client*** - tato třída umožňuje vytvoření MQTT klienta pro komunikaci s MQTT brokerem. Klient může odesílat a přijímat zprávy.
- třída ***Message*** - tato třída reprezentuje MQTT zprávu, která může být odeslána nebo přijata klientem.
- třída ***MQTTException*** - tato třída reprezentuje výjimku, která se vyvolá, když dojde k chybě při komunikaci s MQTT brokerem.

Knihovna obsahuje také různé funkce pro konfiguraci a ovládání MQTT klienta. Mezi tyto funkce patří:

- ***connect()*** - funkce pro připojení klienta k MQTT brokeru,
- ***disconnect()*** - funkce pro odpojení klienta od MQTT brokeru,
- ***subscribe()*** - funkce pro přihlášení klienta k odběru zpráv daného tématu,
- ***unsubscribe()*** - funkce pro odhlášení klienta z odběru zpráv daného tématu,
- ***publish()*** - funkce pro odeslání zprávy na dané téma.

3.6.3 Operační systém pro 3D tiskárnu

Vhodný operační systém pro obsluhu 3D tiskárny musí splňovat tyto požadavky:

- nemá vysoké nároky na výkon (lze spustit na Raspberry Pi Zero W),
- podporuje 3D tiskárnu i3 MK3S+,
- obsahuje API pomocí kterého lze vzdáleně řídit a monitorovat tisk,
- podporuje auto-connect (uživatel nemusí manuálně konfigurovat operační systém při každém spuštění 3D tiskárny),

Po vyhodnocení uvedených nároků byl jako vhodný operační systém vybrán Octoprint. OctoPrint je operační systém pro mikrokontroléry s otevřeným zdrojovým kódem (open-source). Jeho hlavní funkcí je řízení 3D tiskáren pomocí webového rozhraní. Tento software nabízí uživatelům širokou škálu funkcí a vlastností pro správu tisku, včetně API rozhraní.

API rozhraní OctoPrintu umožňuje uživatelům vytvářet a integrovat vlastní aplikace a nástroje do softwaru pro řízení 3D tiskáren. API je k dispozici v podobě RESTful webového rozhraní, které umožňuje uživatelům komunikovat s OctoPrintem pomocí HTTP požadavků.

Díky API mohou uživatelé vytvářet aplikace, které se integrují s OctoPrintem a poskytují další funkce a vlastnosti pro řízení tisku. API OctoPrintu také umožňuje přístup k datům o tisku, jako jsou stav tisku, informace o teplotách, průběžný postup tisku a další. Tato data mohou být použita pro vytváření statistik a vylepšení procesu tisku.

API OctoPrintu je snadno použitelné a obsahuje rozsáhlou dokumentaci, která je k dispozici na webových stránkách Octoprintu a obsahuje podrobné informace o dostupných funkcích a způsobu jejich použití.

Komunikace s AAS pomocí HTTP protokolu

Komunikace mezi AAS a assetem, tedy 3D tiskárnou, probíhá pomocí HTTP protokolu. Díky Octoprint API je možné pomocí HTTP protokolu odesílat požadavky na příslušnou adresu a port Octoprint serveru. Adresa a port jsou závislé na konkrétní konfiguraci Octoprint serveru. Každá 3D tiskárna v lokální síti má unikátní IP adresu. Protokol je založen na modelu klient-server. Klientem je AAS 3D tiskárny, které požaduje informaci. Server, tedy Octoprint API, poskytuje odpověď v podobě informace ve formátu JSON. V rámci HTTP protokolu jsou definovány různé metody, které umožňují klientovi komunikovat se serverem a manipulovat s daty. Například metoda GET slouží k získání informací z serveru, zatímco metoda POST slouží k odeslání dat na server. [19]
Adresa HTTP požadavku pro zjištění aktuálního stavu tisku je zobrazena na obrázku 3.2. V odpovědi na tento požadavek získáme data ve formátu JSON, které obsahují aktuální stav tiskárny, včetně teploty trysky a podložky, průběhu tisku, aktuálně zvolený souboru k tisku a další data.

http://IP_adresa_octoprint_serveru/api/job

Obr. 3.2: Adresa pro odeslání GET požadavku pro získání informací o tisku

Další důležitou součástí HTTP protokolu je stavový kód, který označuje úspěšnost požadavku. Například kód 200 znamená, že požadavek byl úspěšně vyřízen, zatímco kód 404 označuje, že server nebyl schopen vyhovět požadavku.[19]

3.7 Návrh pasivní části AAS

Při návrhu pasivní části je postupováno dle dostupných materiálů, především se jedná o dokument *Details of the Asset Administration Shell - Part 1* ([11]) od organizace ZVEI. V tomto komplexním dokumentu autoři popisují široké spektrum vlastností, které je možné AAS přidělit. AAS vytvořená pro demonstrační úlohu respektují uvedené pravidla a postupy, ale jsou jednodušší a obsahují pouze základní submodely, které jsou pro demonstrační úlohu dostačující.

V demonstrační úloze se vyskytují celkem čtyři typy assetů, pro které je nutné vytvořit AAS. Jedná se o e-shop, chytrý sklad, produkt a 3D tiskárnu. Jednou vytvořené AAS pro určitý asset mohou být použity pro jiný asset stejného typu opakovaně, jediná změna bude v identifikátorech.

Pasivní část je implementována v souborech s názvem *PassivePart.py*. Při vytváření pasivní části je nutné nejdříve vytvořit objekt assetu, který obsahuje identifikátor v podobě parametru id_short, druh assetu (zda se jedná o instanci nebo o typ) a identifikátor typu IRI. Dále jsou definovány submodely pro jednotlivé výrobní operace. Každý submodel opět obsahuje identifikátor id_short a IRI identifikátor. V každém submodelu je možné definovat submodel elementy, které popisují např. cenu, rozměry, nebo jiné dílčí vlastnosti submodelu. Vytvořené submodel elementy musí být po vytvoření ručně přidány do submodelu. Nakonec je vytvořeno AAS, které mimo standardních identifikátorů id_short a IRI identifikátoru obsahuje také referenci na asset a submodely. V případě nevyužití referencí je nutné definovat AAS včetně jeho dílčích částí (submodely, asset, ...) jako jednu rozsáhlou strukturu. S využitím referencí je programátorovi umožněno vytvořit dílčí části zvlášť a finálnímu AAS přiřadit referenci, tedy odkaz na místo v programu, kde se dílčí části nacházejí. Použitím referencí je zvýšena struktura programu a tím pádem i jeho čitelnost.

Jako příklad je zde uveden postup při vytváření pasivní části AAS pro 3D tiskárnu. Nejdříve je nutné vytvořit objekt assetu, tedy 3D tiskárny, včetně identifikátorů. Následně je přidán submodel pro zajištění služby 3D tisku. Submodel obsahuje několik submodel elementů:

- ***PrintingStatus_property***
  Popis: udává, zda je 3D tiskárna zaneprázdněná (aktuálně probíhá tisk)
  Typ proměnné: Boolean
  Hodnota: true - tiskárna je zaneprázdněná /
  false - tiskárna není zaneprázdněná
- ***FilamentType_property***
  Popis: udává, zda je 3D tiskárna zaneprázdněná (aktuálně probíhající tisk)
  Typ proměnné: Integer

Hodnota: 1 - PLA / 0 - ABS / 2 - PETG / 3 - TPE

- ***ManufacturingQueueStatus_property***
  Popis: udává, zda má 3D tiskárna zaplněnou výrobní frontu
  Typ proměnné: Boolean
  Hodnota: true - fronta je zaplněna / false - fronta není zaplněna
- ***ProductionTime_property***
  Popis: udává, kdy skončí tisk aktuálního produktu
  Typ proměnné: DateTime
  Hodnota: čas ve formátu HH:MM:SS

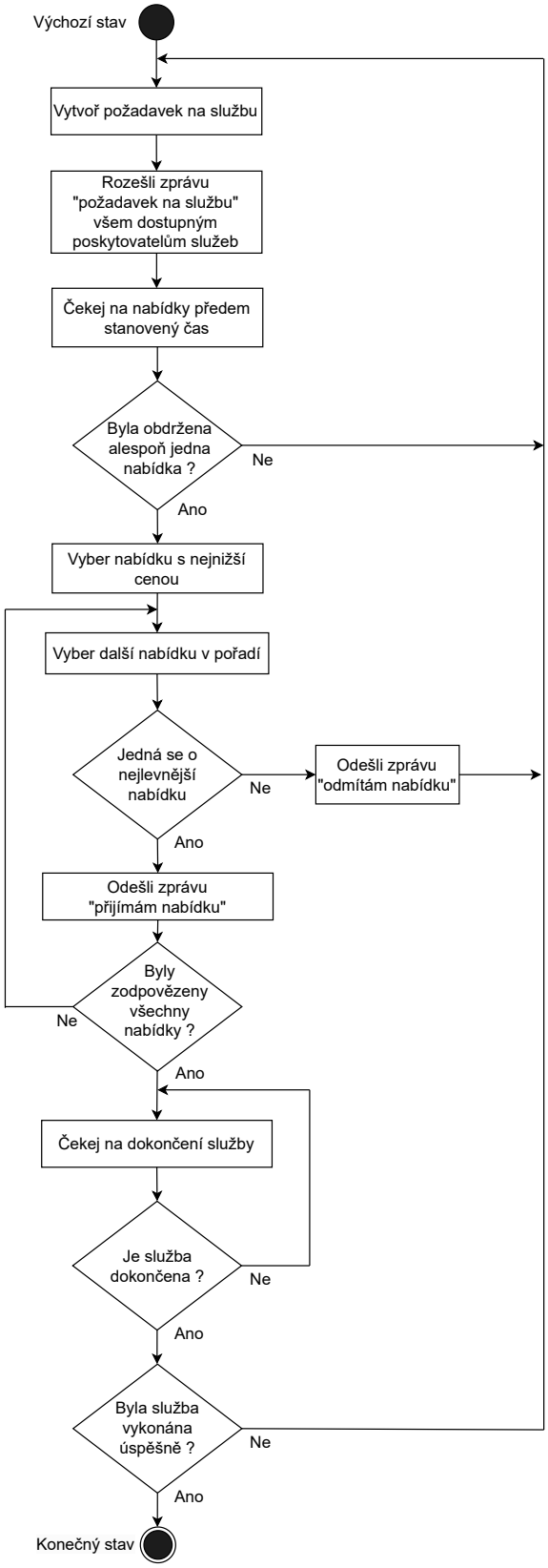
Vytvořené submodel elementy musí být přiřazeny k odpovídajícímu submodelu. Toho je docíleno pomocí metody *submodel_element.add()*. Následně je možné vytvořit AAS, které kromě požadovaných identifikátorů obsahuje referenci na asset a submodel.

3.8 Návrh aktivní části AAS

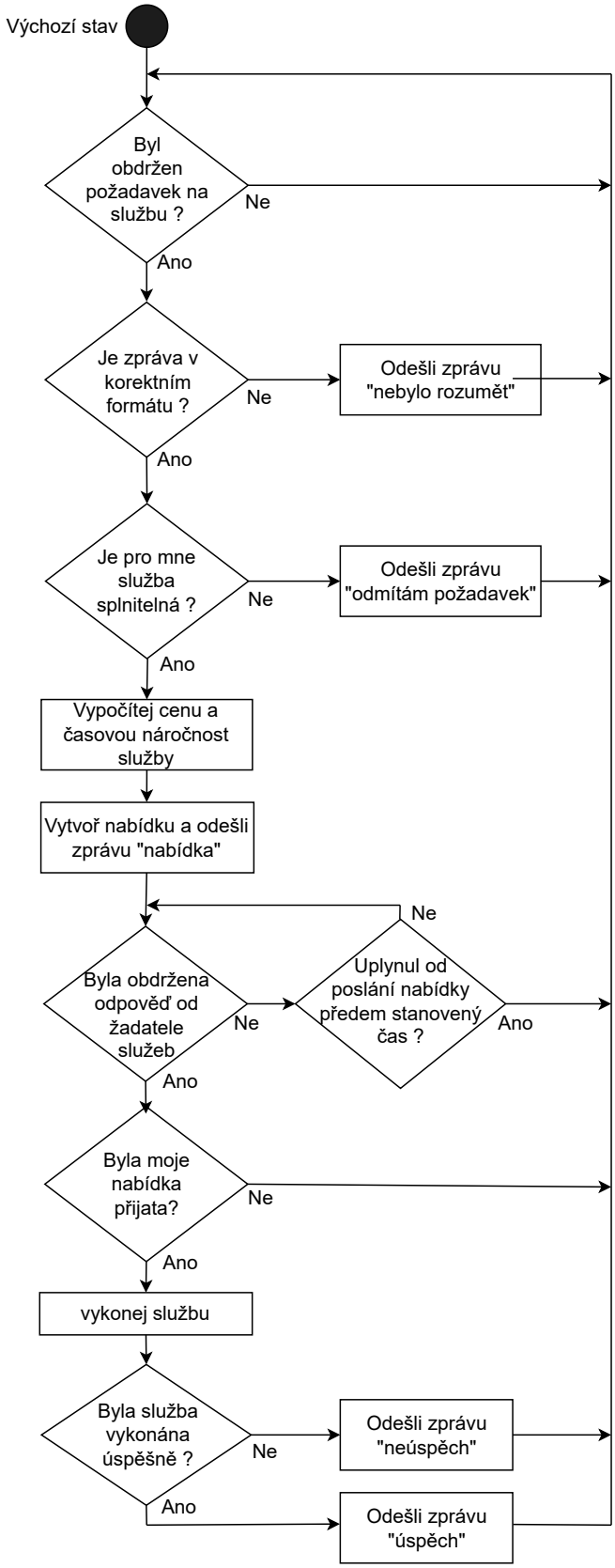
Aktivní část AAS se skládá z funkčních bloků, které umožňují interagovat s ostatními AAS a provádět specifické funkce. Tyto funkční bloky mohou obsahovat řídící, monitorovací, diagnostické a další funkce, které jsou specifické pro daný asset v závislosti na tom, zda se jedná o poskytovatele služeb, nebo o žadatele služeb. Postup vytváření aktivní části AAS je popsán v následujících podkapitolách.

3.8.1 Role AAS

Nejprve je nutné identifikovat roli poskytovatele či žadatele služeb jednotlivých AAS v demonstrační úloze. Primární služba, která se v demonstrační úloze požaduje, je 3D tisk. Poskytovatelem služby jsou tedy 3D tiskárny. Žadatelem služby *"tisk"* jsou jednotlivé produkty, které se vytváří na základě zákaznické objednávky v e-shopu. Jelikož se v demonstrační úloze nachází více produktů, které žádají danou službu od dvou 3D tiskáren, je pro vyjednávání zvolen nabídkový proces. Vývojový diagram programu pro žadatele služeb a poskytovatele služeb je zobrazen na obrázcích 3.3 a 3.4.



Obr. 3.3: Vývojový diagram žadatele služeb (produkt)

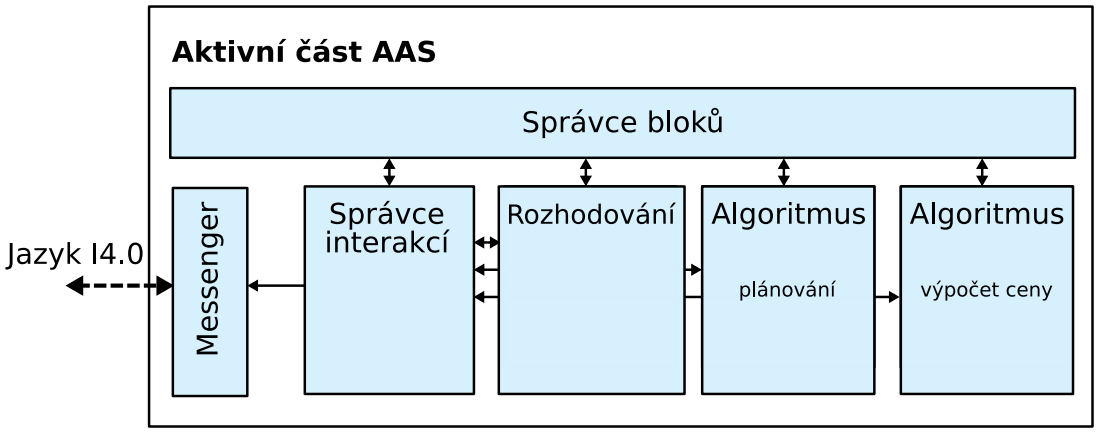


Obr. 3.4: Vývojový diagram poskytovatele služeb (3D tiskárna)

Druhá služba, která je nutná pro demonstrační úlohu, je vytvoření produktu v chytrém skladu. Zde je v roli žadatele služeb e-shop, který požaduje službu *"vytvoření produktu"*. Poskytovatelem služby je komponenta chytrý sklad. Jelikož se v demonstrační úloze nachází pouze jedna komponenta, která žádá tuto službu a jedna komponenta, která ji nabízí, ztrácí vyjednávání pomocí nabídkového procesu smysl a je zjednodušeno pouze na jednu zprávu Jazyka I4.0. Zpráva obsahuje požadavek na službu *"vytvoření produktu"* a je poslána z administrativní obálky e-shopu do administrativní obálky chytrého skladu.

3.8.2 Programová struktura

Jelikož je pasivní část AAS napsána v programovacím jazyce Python, byl tento jazyk zvolen i pro vytváření aktivní části. Aktivní část využívá data uložená v pasivní části pomocí programové techniky zvané dědičnost. Pro čtení a zápis dat do pasivní části jsou vytvořeny speciální metody popsané v kapitole 3.8.4. Struktura programu odpovídá blokovému diagramu aktivní části, který je zobrazen na obrázku 3.5.



Obr. 3.5: Bloková struktura aktivní části AAS

Správce bloků

Správce bloků je implementován v soubory *ComponentManager.py* je hlavním řídícím blokem aktivní části. Je v něm uskutečněna inicializace důležitých programových proměnných. Správce bloků dědí z pasivní části a je mu umožněno číst a zapisovat data zde uložená. Správce bloků také řídí ostatní bloky. Na začátku programu jsou vytvořeny instance tříd jednotlivých bloků, které se nachází v aktivní části. Vytvořené instance tříd, respektive jejich metody, jsou v průběhu programu volány a tím se vykonají požadované akce. Ve správci bloků je implementován hlavní stavový automat, který je vytvořen dle vývojového diagramu zobrazeného na obrázcích 3.4 a 3.3.

Po inicializaci proměnných je stavový automat spouštěn v nekonečné smyčce. Stavy, kterých může automat nabývat jsou uloženy pomocí proměnné typu enumerate v souboru *StateSP.py* (poskytovatel služby) a *StateSR.py* (žadatel služby). Správce bloků je taktéž zodpovědný za implementaci nabídkového procesu.

Messenger

Messenger je implementován pomocí stejnojmenné třídy v souboru *Messenger.py*. Hlavní funkcí messengeru je umožnit výměnu dat s ostatními AAS. Třída Messenger používá knihovnu pro komunikaci pomocí MQTT protokolu. Při vytváření třídy proběhne inicializace MQTT klienta k brokerovi s parametry, které jsou obdrženy od správce bloků. Po inicializaci je Messenger připravený ke komunikaci. K dispozici je několik metod, které může správce bloků k řízení Messengeru použít:

- metoda ***connect_mqtt()***
  parametr: klient
  popis: připojí klienta k MQTT brokerovi
- metoda ***subscribe()***
  parametr: téma
  popis: nastaví odběr na zadané téma
- metoda ***unsubscribe()***
  parametr: -
  popis: zruší odběr u všech odebíraných témat
- metoda ***publish()***
  parametry: zpráva, téma
  popis: zveřejní zprávu na dané téma
- metoda ***on_message()***
  parametr: -
  popis: dekóduje a uloží zprávu, která byla přijata od MQTT brokera

Správce interakcí

Zprávy, které se mezi AAS předávají, jsou vytvořeny ve správci interakcí. Ten je implementován jako třída *InteractionManager* v souboru *InteractionManager.py*. Správce interakcí obsahuje šablonu pro vytváření zpráv v Jazyce I4.0 dle normy VDI/VDE 2193. Tato šablona je využívána třídními metodami, které do šablony doplní vhodné informace získané z pasivní části. Pro každý typ zprávy v nabídkovém procesu existuje jedna metoda, která určí typ zprávy a informace, které je potřeba do zprávy doplnit.

Po vytvoření je zpráva ve formátu JSON předána třídě *Messenger*, který ji dle pokynů správce bloků odešle s příslušným tématem pomocí protokolu MQTT.

Pro převod zpráv do formátu JSON je využita Python knihovna s názvem *json*. Zprávy jsou vytvářeny dynamicky pomocí objektů typu *dictionary*, které jazyk Python nativně obsahuje. Po vytvoření je zpráva převedena do formátu JSON pomocí metody *json.dumps()* a je připravena k předání třídě *Messenger*.

Algoritmus plánování

Plánování je implementováno rozdílným způsobem v závislosti na tom, zda je AAS poskytovatelem či žadatelem služeb. Pro obě role je algoritmus plánování umístěn v souboru *AlgScheduling.py*.

U žadatele služeb se jedná o plánování operací. V plánování operací je seznam všech služeb (operací), které musí žadatel podstoupit. Produkt z demonstrační úlohy má ve svém plánování operací uloženy dvě operace, jedná se o operace *"vytvoření modelu"* a *"vytisknutí"*. Operacím je přiřazeno pořadí, ve kterém se mají provádět. Po vykonání všech operací, které má žadatel služeb ve svém plánu, je vyhodnocen jako hotový výrobek.

U poskytovatele služeb se jedná o plánování výroby. Plánování výroby je vytvořeno ve třídě *ManufacturingScheduler*. Tato třída je implementována pomocí prioritní fronty, do které je možné přidávat a odebírat služby v závislosti na jejich prioritě. Před vykonáváním služby poskytovatel vždy zkontroluje stav své výrobní fronty a vybere tu službu, která je na čele fronty. Tím je zajištěn princip FIFO (First in - first out). Služba, která byla nejdříve vyjednána, je také nejdříve provedena. Pro pohodlné přidávání služeb, odebírání služeb a kontrolu stavu fronty jsou vytvořeny metody:

- metoda ***add_to_schedule()***
  parametr: služba
  popis: vloží službu do fronty dle priority, kterou služba obsahuje
- metoda ***next_to_manufacture()***
  parametr: -
  popis: vrátí službu fronty, která má být vykonána jako další v pořadí. Tato služba je z fronty zároveň odebrána.
- metoda ***size()***
  parametr: -
  popis: vrátí počet služeb ve frontě
- metoda ***is_full()***
  parametr: -
  popis: vrátí hodnotu true, pokud je fronta naplněna

- metoda ***is_empty()***
  parametr: -
  popis: vrátí hodnotu true, pokud je fronta prázdná

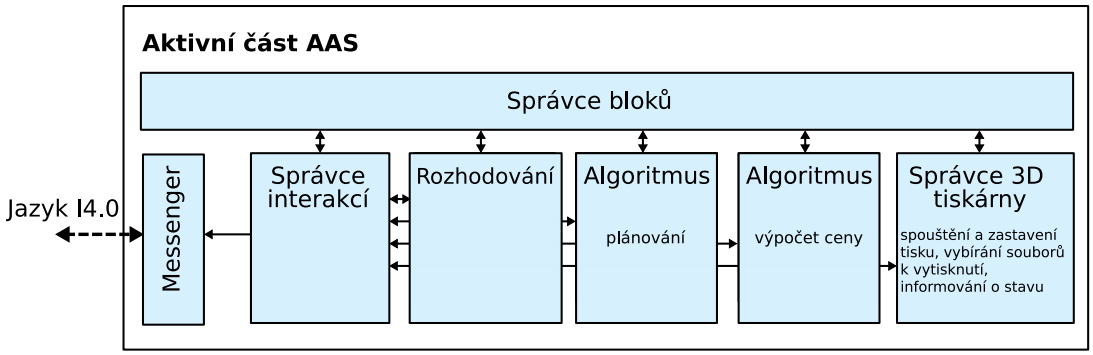
Algoritmus pro výpočet ceny

Žadatel služeb potřebuje pro vytvoření nabídky údaj o finanční nákladnosti dané služby. Ke stanovení ceny za službu je vytvořen algoritmus pro výpočet ceny, který je uložen v souboru *AlgCalculation.py*. Implementace algoritmu se bude lišit v závislosti na daném assetu, který službu poskytuje a na druhu poskytované služby. Nelze tedy stanovit obecný vzorec pro výpočet ceny služby, protože do něj mohou vstupovat parametry, které jsou specifické pro daný asset a službu.

Například u assetu 3D tiskárna a služby tisk se cena odvíjí od použitého typu tiskového materiálu, odhadované doby tisku (záleží na velikosti a složitosti modelu) a odhadované spotřeby materiálu. V případě složitějšího průmyslového zařízení je možné do ceny za službu zahrnout i pořizovací cenu zařízení, náklady na servis nebo počet nástrojů, které bude zařízení potřebovat ke splnění dané služby. Tento algoritmus není součástí AAS, které mají roli žadatele služeb.

3.8.3 Rozšíření aktivní části u assetu 3D tiskárna

Aktivní část assetu 3D tiskárna byla rozšířena o další programový blok s názvem Správce 3D tiskárny. Jak již z názvu vyplývá, tento blok má za úkol komunikaci s 3D tiskárnou. Aktivní část u assetu 3D tiskárna je zobrazena na obrázku 3.6



Obr. 3.6: Bloková struktura aktivní části AAS pro asset 3D tiskárna

Správce 3D tiskárny

Jak bylo popsána v kapitole 3.4, z důvodu omezení operačního systému Octoprint není možné spustit AAS a Octoprint na stejném mikrokontroléru. AAS je tedy umístěno mimo Octoprint, se kterým komunikuje pomocí HTTP protokolu. Integrovat tuto komunikaci do správce bloků by bylo nepřehledné a výrazně by se tím snížila čitelnost kódu, která přispívá ke snadnějším změnám v kódu a údržbě celého programu. Proto byla vytvořena nová třída PrintingManager, která je umístěna v souboru *PrintingManeger.py* a zajišťuje komunikaci s API Octoprintu pomocí HTTP protokolu.

Při vytvoření třídy je nutné nejprve inicializovat adresu Octoprint serveru a unikátní klíč, který slouží k ověření identity uživatele přistupujícího k API. Tento klíč umožňuje API rozpoznat uživatele a poskytnout mu přístup k požadovaným funkcím a datům.

Třída *PrintingManager* obsahuje několik metod, pomocí kterých je umožněno posílání HTTP požadavků a získávání dat o tisku:

- metoda ***send_request()***
  parametry: typ požadavku, cesta, tělo
  popis: Metoda pošle požadavek HTTP <typ> na adresu
  http://<IP_adresa_octoprint_serveru>/<cesta>.
  V případě, že je jako parametr typ zvolen typ POST, k požadavku se přidá i tělo zprávy ve formátu JSON.
  Metoda vrací odpověď na požadavek ve formátu JSON.
- metoda ***update()***
  parametr: -
  popis: Metoda periodicky posílá požadavek GET pomocí metody send_request(), kterou volá s parametry typ = GET, cesta = api/job.
  Odpovědí na požadavek jsou informace o probíhajícím tisku ve formátu JSON.
- metoda ***select_print_file()***
  parametr: název souboru
  popis: Metoda pošle požadavek typu POST s příkazem *select*, který vybere soubor k vytisknutí.
- metoda ***start_print_job()***
  parametr: -
  popis: Metoda pošle požadavek typu POST s příkazem *start*, který spustí tisk vybraného souboru.

- metoda ***get_selected_file_info()***
  parametr: název souboru
  popis: Metoda vrací informace o aktuálně vybraném souboru k tisku a ukládá je do třídy.
- metoda ***get_estimated_print_time()***
  parametr: -
  popis: Metoda vrací odhadovaný čas potřebný k vytisknutí vybraného souboru.
- metoda ***get_estimated_filament_length()***
  parametr: -
  popis: Metoda vrací odhadovanou délku spotřebovaného filamentu, která je potřeba k vytisknutí vybraného souboru.
- metoda ***get_estimated_filament_volume()***
  parametr: -
  popis: Metoda vrací odhadovaný objem spotřebovaného filamentu, který je potřeba k vytisknutí vybraného souboru.
- metoda ***get_completion()***
  parametr: -
  popis: Metoda vrací procentuální postup tisku aktuálního souboru.
- metoda ***get_print_state()***
  parametr: -
  popis: Metoda vrací aktuální stav, ve kterém se 3D tiskárna nachází.

3.8.4 Předávání informací mezi aktivní a pasivní částí

Jelikož aktivní část používá data z pasivní části, bylo nutné najít způsob, jakým se budou data mezi částmi AAS předávat. Optimálním způsobem je přistupovat k pasivní části podobně jako k databázi, do které můžeme data ukládat a také je z ní číst. K tomu je zapotřebí vytvořit metody pro přístup k datům (tzv.settery) a zápis dat (tzv. gettery). V základu je pro každý submodel element, který aktivní část využívá, vytvořena jedna metodu *set()* a jedna metodu *get()* . Metoda *get()* vrací aktuální hodnotu uloženou v submodel elementu a pomocí metody *get()* můžeme tuto hodnotu přepsat. Vytvořené metody jsou vypsány níže.

- **PrintingStatus_property**
  set_printing_status()
  get_printing_status()

- **FilamentType_property**
  set_filament_type()
  get_filament_type()
- **ManufacturingQueueStatus_property**
  set_manufacturing_queue_status()
  get_manufacturing_queue_status()
- **ProductionTime_property**
  set_production_time()
  get_production_time()

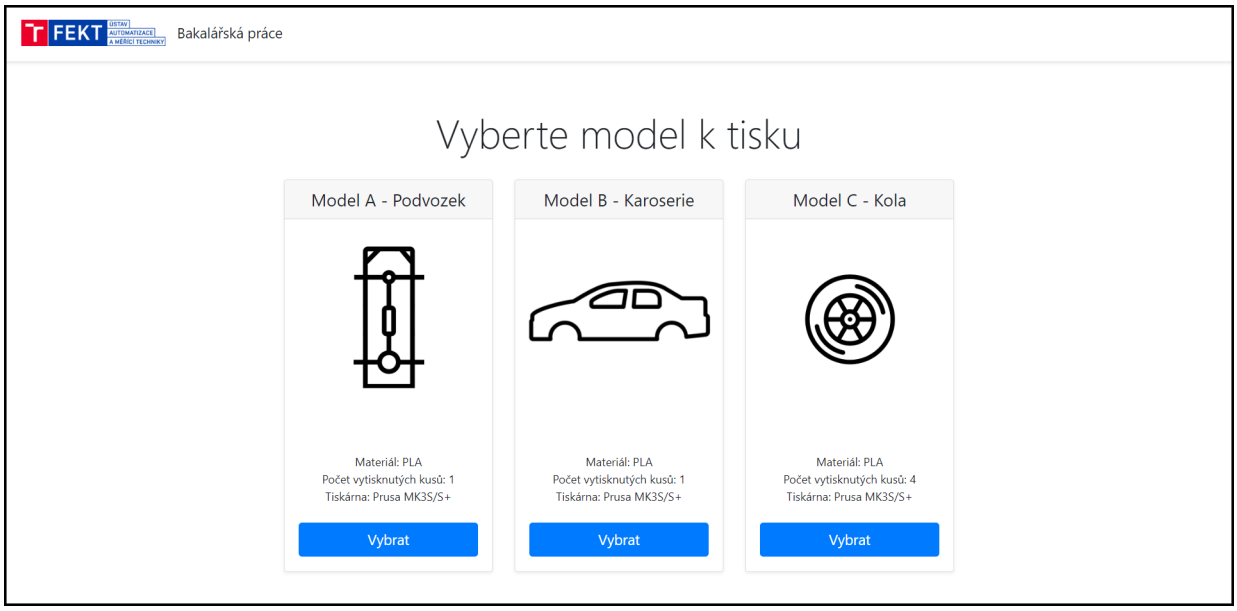
3.9 E-shop

3.9.1 Návrh e-shopu

Na výběr bylo několik způsobů implementace uživatelského rozhraní pro demonstrační úlohu. Mezi kandidáty byla například aplikace pro MS Windows nebo pro operační systém mobilních telefonů Android. Nakonec byla zvolena webová knihovna *Flask*. Knihovna je napsána v programovacím jazyce Python a slouží k vytváření webových stránek s využitím hypertextového značkovacího jazyka (HTML) a programovacího jazyka Javascript. Takto vytvořenou webovou stránku je možné zobrazit pomocí webové prohlížeče. E-shop může být umístěn na firemním serveru, v takovém případě bude dostupný pouze v rámci firemní internetové sítě. Pokud chceme, aby byl e-shop veřejně přístupný, můžeme e-shop nahrát na internet s využitím dostupných webových hostitelů. Pro účely demonstrační úlohy je e-shop spouštěn na notebooku, který zároveň poskytuje lokální bezdrátové připojení ostatním komponentám v úloze. Pokud se do této sítě připojíme s jiným zařízením, například s chytrým telefonem, můžeme i pomocí něj vytvořit na e-shopu objednávku.

3.9.2 Návrh uživatelského rozhraní

Uživatelské rozhraní se skládá ze tří modelů, které jsou k dispozici pro vytisknutí. U každého modelu je uveden jeho popis, který obsahuje údaj o materiálu, ze kterého se model vytiskne, počet vytisknutých kusů a typ tiskárny, která model vytiskne. Součástí modelu je i ilustrační obrázek výtisku. Finální vzhled e-shopu je na obrázku 3.7. Díky moderním webovým technologiím je stránka e-shopu plně responzivní, tedy optimalizovaná pro všechny druhy zařízení. Vzhled e-shopu po připojení z chytrého telefonu je na obrázku 3.8. Modely na stránce jsou nyní umístěny v seznamu pod sebou a uživateli je umožněno prohlížet nabídku modelů pomocí dotyku prstu.

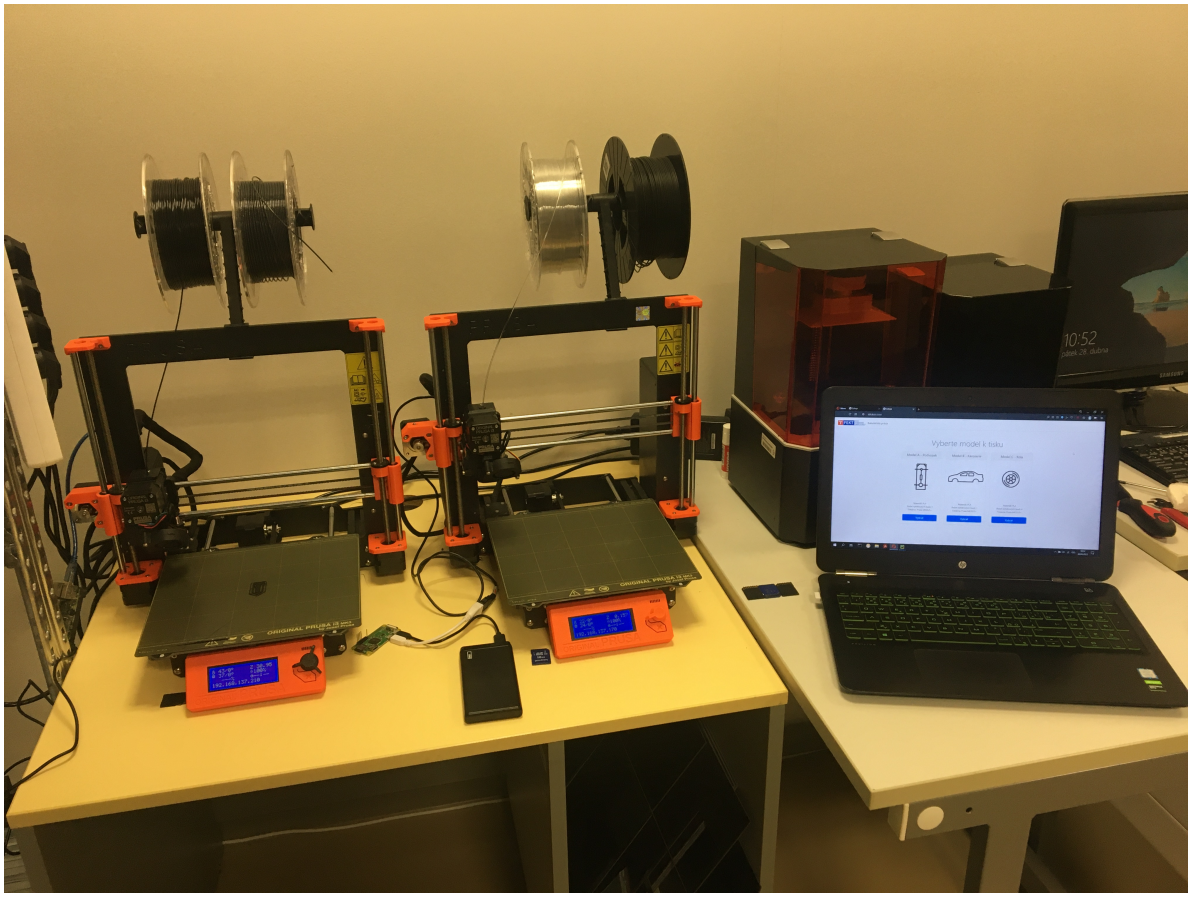


Obr. 3.7: Uživatelské rozhraní e-shopu

Obr. 3.8: Uživatelské rozhraní e-shopu zobrazené na chytrém telefonu

3.10 Realizace v laboratoři

Demonstrační úloha byla nad rámec zadání realizována i s reálnými komponentami. Úloha byla realizována v laboratoři na UTEE FEKT, viz obrázek 3.9. Díky realizaci v laboratoři byly odhaleny některé chyby v softwaru, zejména chyby, které způsobily zdlouhavý proces připojování mikrokontrolérů k lokální wifi síti. Jednalo se o chybně nastavené parametry v Octoprintu, který se díky nim dokázal připojit až na několikátý pokus a také o chybné nastavení notebooku, který sloužil jako poskytovatel lokální sítě. Po správném nastavení již připojení proběhlo bez problémů.



Obr. 3.9: Realizace demonstrační úlohy

4 Testování demonstrační úlohy

Testování demonstrační úlohy probíhalo v laboratoři na FEKT UTEE, na hardwarové sestavě popsané v kapitole 3.5. K testování byly vytvořeny scénáře simulující reálné situace, které mohou nastat při použití AAS v průmyslové výrobě.

4.1 Testovací scénáře

Každý testovací scénář je zahájen uživatelskou objednávkou z e-shopu. Administrativní obálka e-shopu pošle požadavek na službu *"vytvoření produktu"* administrativní obálce chytrého skladu. Chytrý sklad vytvoří nový typ produktu, tedy žadatele služeb. Administrativní obálka produktu je nyní připravena na vyjednávání služby *"tisk"*. Vyjednávání služby pro různé situace je popsáno v následujících scénářích.

Scénář 1: vyjednávání mezi žadatelem a dvěma poskytovateli služeb

Žadatel služeb: produkt
Poskytovatel služeb: 3D tiskárny
Popis:
Komunikaci zahajuje žadatel, který žádá dostupné poskytovatele o službu *"tisk"*. Oba poskytovatelé odpoví na požadavek o službu nabídkou. Žadatel vybere nabídku od nejlevnějšího poskytovatele a pošle mu zprávu o přijetí jeho nabídky. Druhému poskytovateli pošle zprávu o odmítnutí služby. Vybraný poskytovatel služeb zahájí vykonání služby *"tisk"*. Po vytisknutí odešle poskytovatel služby zprávu žadateli o splnění služby a vyjednávání v tomto okamžiku končí.
Výsledek testování: Vyjednávání proběhlo podle předpokladů, výrobek byl úspěšně vytisknut.

Scénář 2: Vyjednávání mezi žadatelem a dvěma poskytovateli služeb - výrobní fronta je zaplněna u poskytovatele s nižší cenou

Žadatel služeb: produkt
Poskytovatel služeb: 3D tiskárny
Popis:
Komunikaci zahajuje žadatel, který žádá dostupné poskytovatele o službu *"tisk"*. Nyní odpoví pouze poskytovatel, který nemá zaplněnou výrobní frontu. Žadatel vybírá pouze z jedné nabídky. Žadatel pošle poskytovateli zprávu o přijmutí jeho nabídky. Poskytovatel služeb zahájí vykonání služby *"tisk"*. Po vytisknutí odešle poskytovatel služby zprávu žadateli o splnění služby a vyjednávání v tomto okamžiku končí.

Výsledek testování: Vyjednávání proběhlo podle předpokladů, výrobek byl úspěšně vytisknut.

Scénář 3: Vyjednávání mezi žadatelem a dvěma poskytovateli služeb - výrobní fronta je zaplněna u poskytovatele s vyšší cenou

Žadatel služeb: produkt
Poskytovatel služeb: 3D tiskárny
Popis:
Objednání produktu i vyjednávání probíhá identicky jako v případě popsaném v druhém scénáři.
Výsledek testování: Vyjednávání proběhlo podle předpokladů, výrobek byl úspěšně vytisknut.

Scénář 4: Vyjednávání mezi žadatelem a dvěma poskytovateli služeb - výrobní fronta je zaplněna u obou poskytovatelů

Žadatel služeb: produkt
Poskytovatel služeb: 3D tiskárny
Popis:
Komunikaci zahájí žadatel služeb rozesláním poptávky po službě dostupným poskytovatelům. Jelikož mají oba poskytovatelé zaplněnou tiskovou frontu, nepřijímají nové poptávky po službě. Žadatel služeb překročí časový limit, během kterého neobdržel žádnou odpověď na svoji poptávku a rozesílá poptávku po službě znovu. Žadatel služeb opakuje tento proces do té doby, než se uvolní výrobní fronta alespoň u jednoho poskytovatele služeb. Poté proběhne standardní vyjednávání a tisk výrobku.
Výsledek testování: Vyjednávání proběhlo podle předpokladů, žadatel služeb setrval ve vyjednávání do té doby, než se uvolnila výrobní fronta alespoň u jednoho poskytovatele služeb. Výrobek byl úspěšně vytisknut.

Scénář 5: Vyjednávání mezi žadatelem a dvěma poskytovateli služeb - žadatel služeb již absolvoval službu, která skončila neúspěchem

Žadatel služeb: produkt
Poskytovatel služeb: 3D tiskárny
Popis:
Žadatel služeb si už jednou vyjednal službu, která ovšem skončila neúspěchem a nyní vstupuje do nového vyjednávání. Aby byl upřednostněn před jinými výrobky, které

také vyjednávají, je mu zvýšena priorita. Vyjednávání žadatele služeb s vyšší prioritou je identické jako u žadatele se standardní prioritou. Priorita se projeví až v okamžiku ukládání do výrobní fronty poskytovatele služeb. Výrobek je vytisknut ihned po skončení probíhajícího tisku a nemusí tak čekat na vytisknutí výrobků se standardní prioritou, které byly vyjednány před ním.
Výsledek testování: Prioritní výrobní fronta se chová tak, aby upřednostnila výrobky se zvýšenou prioritou oproti výrobkům se standardní prioritou. Výrobek se zvýšenou prioritou byl úspěšně vytisknut před výrobky se standardní prioritou.

Scénář 6: Vyjednávání mezi žadatelem a dvěma poskytovateli služeb - poskytovatel služby vykoná službu neúspěšně

Žadatel služeb: produkt
Poskytovatel služeb: 3D tiskárny
Popis:
Vyjednávání probíhá identicky jako v prvním scénáři. Komunikaci zahajuje žadatel, který žádá dostupné poskytovatele o službu tisk. Oba poskytovatelé odpoví na požadavek o službu nabídkou. Žadatel vybere nabídku od nejlevnějšího poskytovatele a pošle mu zprávu o přijetí jeho nabídky. Druhému poskytovateli pošle zprávu o odmítnutí služby. Vybraný poskytovatel služeb zahájí vykonání služby tisk. V průběhu tisku nastane simulovaná porucha (v reálném provozu se může jednat např. o ztrátu napájení, nedostatek filamentu, aj.), která znemožní pokračování tisku. Poskytovatel služby odešle žadateli služby zprávu s informací o neúspěchu při vykonání vyjednané služby. Tím končí proces vyjednávání mezi žadatelem služeb a poskytovatelem služeb, který není schopný vykonat požadovanou službu. Žadatel služeb vstupuje do nového vyjednávání se zvýšenou prioritou tak, jak je popsáno v pátém scénáři.
Výsledek testování: Po neúspěšné službě proběhlo zvýšení priority u výrobku, který je správně vytisknut až po druhém vyjednávání.

5 Budoucí práce

Při tvorbě bakalářské práce se objevily některé problémy, které nebyly vyřešeny zcela optimálně. Jedná se především o způsob, podle kterého se žadatel služeb rozhoduje mezi dostupnými nabídkami. V aktuální verzi práce se žadatel služeb rozhoduje pouze na základě nejnižší ceny za službu, nebere však v potaz jiné parametry, jako je například doba trvání služby. Tento problém řeší pouze částečně omezení maximálního počtu výrobků ve výrobní frontě. Každý poskytovatel služeb má nastavený maximální počet výrobků, při kterém přestane vyjednávat nové služby od žadatelů. V praxi to vypadá tak, že se nejdříve zaplní výrobní fronta u poskytovatele služeb, který nabízí nejnižší cenu za tisk, následuje poskytovatel služeb s druhou nejnižší cenou atd.

Jako další vylepšení připadá v úvahu úprava e-shopu, pomocí kterého uživatel objednává produkty k vytisknutí. Bylo by vhodné, aby měl uživatel možnost nahrát vlastní model a ten si objednat k vytisknutí. Uživatel by dostal zpětnou informaci o tom, jak bude vytisknutí finančně nákladné a jak dlouho bude trvat. Toho by bylo možné dosáhnout úpravou webové stránky a zaintegrováním sliceru, který poskytuje Octoprint.

Různých rozšíření demonstrační úlohy je velké množství, ať už se jedná o stávající prvky, které lze inovovat, nebo o nové prvky, které je možné k úloze přidat. Jedná se například o propojení demonstrační úlohy s databází, ve které by bylo možné ukládat výrobní data, nebo o zautomatizování vykládání výtisků z 3D tiskáren pomocí vhodných robotických manipulátorů.

Závěr

V úvodní kapitole byly vysvětleny pojmy z Průmyslu 4.0, jedná se např. o RAMI 4.0 model, AAS a další. Podrobně byla popsána struktura AAS a požadavky na tvorbu, které jsou shrnuty v tabulce 1.1. V závěru kapitoly byly popsány dostupné nástroje pro vytváření AAS.

Ve druhé kapitole byly rozebrány způsoby komunikace mezi AAS, zejména se práce zaměřila na OPC UA a MQTT, které patří k nejpoužívanějším způsobům komunikace mezi AAS. Dále byl popsán koncept jednotné komunikace mezi AAS, který se označuje názvem Jazyk I4.0. Jako komunikační protokol pro demonstrační úlohu byl zvolen protokol MQTT.

V rámci realizace praktické části byl nejdříve proveden návrh demonstrační úlohy, jejíž cílem je znázornění principů decentralizovaného řízení s využitím AAS. Zvolená úloha znázorňuje 3D tisk na objednávku. Pro implementaci úlohy byl zvolen programovací jazyk Python, ve kterém je napsána pasivní i aktivní část AAS. Ovládání 3D tiskáren zajišťují mikrokontroléry Raspberry Pi Zero W, na kterých je nainstalován operační systém Octoprint. Uživatelské rozhraní webového e-shopu je optimalizováno pro různé rozměry zařízení, na kterých je zobrazen tak, aby byla uživateli usnadněna objednávka tisku.

Navržený software byl otestován na reálném pracovišti. Problémy, které nastaly při připojování k bezdrátové wifi síti, byly v práci popsány a vyřešeny. Testování proběhlo na několika scénářích, které byly popsány a vyhodnoceny ve čtvrté kapitole práce. V páté kapitole byl nastíněn směr, kterým by se práce mohla v budoucnosti ubírat, včetně popsání aktuálních nedostatků a jejich řešení.

Literatura

[1] FEDERAL MINISTRY FOR ECONOMIC AFFAIRS AND CLIMATE ACTION OF GERMANY, 2016. What is Industrie 4.0 ?. Platform Industrie 4.0 [online]. [cit. 2022-03-11]. Dostupné z URL: <https://www.mpo.cz/assets/dokumenty/53723/64358/658713/priloha001.pdf>

[2] FEDERAL MINISTRY FOR ECONOMIC AFFAIRS AND CLIMATE ACTION OF GERMANY, 2016. What is Industrie 4.0 ?. Platform Industrie 4.0 [online]. [cit. 2022-03-11]. Dostupné z URL: <https://www.plattform-i40.de/IP/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html>

[3] HANKEL, Martin a Bosch REXROTH. The Reference Architectural Model Industrie 4.0 (RAMI 4.0) [online]. 1.4.2015 [cit. 2022-11-20]. Dostupné z URL: <https://www.zvei.org/en/press-media/publications/the-reference-architectural-model-industrie-40-rami-40>

[4] SCHWEICHHART, Karsten. Reference Architectural Model Industrie 4.0 (RAMI 4.0): An Introduction [online]. [cit. 2022-11-26]. Dostupné z URL: <https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf>

[5] BRAUNE, Annerose, Christian DIEDRICH, Sten GRÜNER, et al. Discussion Paper: Usage View of the Asset Administration Shell [online]. 13.3.2019, 48 [cit. 2022-11-15].. Dostupné z URL: <https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/2019-usage-view-asset-administration-shell.pdf>

[6] MINISTRY OF ECONOMY AND FINANCES, ALLIANCE INDUSTRIE DU FUTUR, FEDERAL MINISTRY FOR ECONOMIC AFFAIRS AND ENERGY (BMWI), PLATTFORM INDUSTRIE 4.0 a MINISTERO DELLO SVILUPPO ECONOMICO, 2018. Structure of the Administration Shell: Trilateral perspectives from France, Italy and Germany [online]. 30.3.2018, 64 [cit. 2022-11-15]. Dostupné z URL: <https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/hm-2018-trilaterale-coop.html>

[7] ZVEI - ZENTRALVERBAND ELEKTROTECHNIK- UND ELEKTROINDUSTRIE E.V. AUTOMATION DIVISION, 2020. ZVEI Recommendation 2020.01 The Digital Nameplate: Consistent, sustainable, future-proof, networked [online]. 2020, 8 [cit. 2022-11-20]. Dostupné

z URL: <https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2020/November/Das_Digitale_Typenschild_-_ZVEI-Empfehlung/Digital_Nameplate.pdf>

[8] BELYAEV, Alexander, Christian BLOCK, Birgit BOSS, et al. Modelling the Semantics of Data of an Asset Administration Shell with Elements of ECLASS [online]. 29.6.2021, 54 [cit. 2022-11-19]. Dostupné z URL: <https://eclass.eu/fileadmin/Redaktion/pdf-Dateien/Broschueren/2021-06-29_Whitepaper_PlattformI40-ECLASS.pdf>

[9] BELYAEV, Alexander a Christian DIEDRICH. Specification "Demonstrator I4.0-Language"v3.0 [online]. 2019, 38 [cit. 2022-11-18]. Dostupné z URL: <https://www.researchgate.net/publication/334429449_Specification_Demonstrator_I40-Language_v30>

[10] BEDENBENDER, Heinz, Michael HOFFMEISTER, Ulrich EPPLE, et al. Industrie 4.0 Plug-and-Produce for Adaptable Factories: Example Use Case Definition, Models, and Implementation [online]. 2017,69 [cit. 2022-11-20]. Dostupné z URL: <https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2017/Juni/Industrie_4.0_Plug_and_produce/Industrie-4.0-_Plug-and-Produce-zvei.pdf>

[11] PLATTFORM INDUSTRIE 4.0, 2022. Details of the Asset Administration Shell - Part 1: The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC02). Platform Industrie 4.0 [online]. 30.5.2022, 235 [cit. 2022-03-10]. Dostupné z URL: <https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html>

[12] AASX Package Explorer: Github Repository [online]. [cit. 2022-11-20]. Dostupné z URL: <https://github.com/admin-shell-io/aasx-package-explorer>

[13] NOVA Asset Administration Shell: GitLab Repository [online]. [cit. 2022-11-21]. Dostupné z URL: <https://gitlab.com/novaas/catalog/nova-school-of-science-and-technology/novaas>

[14] ECLIPSE FOUNDATION. Eclipse BaSyx Python SDK: GitHub Repository [online]. [cit. 2022-11-21]. Dostupné z URL: <https://github.com/eclipse-basyx/basyx-python-sdk>

[15] UNIFIED AUTOMATION GMBH. OPC Introduction. Unified Automation [online]. [cit. 2022-12-04]. Dostupné z URL: <https://documentation.unified-automation.com/uasdknet/2.5.4/html/L1OpcIntroduction.html>

[16] ARM, Jakub, Tomas BENESL, Petr MARCON, et al. Automated Design and Integration of Asset Administration Shells in Components of Industry 4.0. Sensors [online]. 2021, 21(6), 200 4 [cit. 2022-12-03]. Dostupné z URL: <doi:10.3390/s21062004>

[17] OPC FOUNDATION. PubSub Concept. OPC Foundation [online]. [cit. 2022-12-04]. Dostupné z URL: <https://reference.opcfoundation.org/Core/Part14/v105/docs/5>

[18] MQTT Essentials: MQTT 5 Protocol - MQTT 5 Essentials, 2019. HiveMQ [online]. [cit. 2022-03-10]. Dostupné z URL: <https://www.hivemq.com/mqtt-essentials/>

[19] MDN Web Docs: Hypertext Transfer Protocol (HTTP) [online], 2023. Mozilla Foundation [cit. 2023-04-09]. Dostupné z URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP>

Seznam symbolů a zkratek

AAS	Asset Administration Shell - Digitální obálka
ZVEI	Zentralverband Elektrotechnik- und Elektroindustrie - Asociace německých výrobců elektrických a elektronických produktů
MES	Manufacturing Execution System - Výrobní informační systém
ID	Identification - Identifikace
RAMI	Reference Architectural Model Industry 4.0 – referenční architektonický model Průmyslu 4.0
IRDI	International Registration Data Identifier - Mezinárodní identifikátor registračních dat
URI	Uniform Resource Identifier - Jednotný identifikátor zdrojů
GUID	Globally unique identifier - Globálně jedinečný identifikátor
UUID	Universal unique identifier - Univerzální jedinečný identifikátor
API	Application Programming Interface - Aplikační programové rozhraní
JSON	JavaScript Object Notation - JavaScriptový objektový zápis
XML	Extensible Markup Language - Rozšiřitelný značkovací jazyk
SDK	Software development kit - Sada pro vývoj softwaru
REST	Representational state transfer - Přenos reprezentativního stavu
OPC UA	Object Process Control Unified Architecture – Jednotná architektura řízení objektových procesů
MQTT	MQ Telemetry Transport – Telemetrický transport MQ
ERP	Enterprise Resource Planning – Plánování podnikových zdrojů
UDP	User Datagram Protocol – Uživatelský datagramový protokol
HTML	Hypertext Markup Language – Hypertextový značkovací jazyka

A Obsah elektronické přílohy

/ ..... kořenový adresář přiloženého archivu
- src ..... adresář se zdrojovými soubory
  - SmartWarehouse.zip
  - ServiceProvider.zip
  - WebEshop.zip