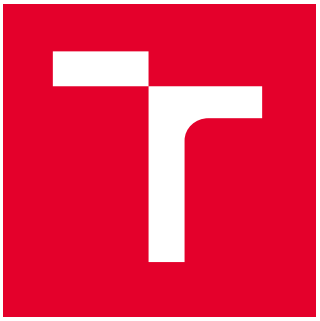


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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

AAS INTERPRETER PRO TESTBED I4.0

AAS INTERPRETER FOR TESTBED I4.0

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Luboš Chmelař**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Ondřej Baštán**

SUPERVISOR

BRNO 2024

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: Luboš Chmelař ***ID:*** 240355

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

AAS interpreter pro testbed I4.0

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a zprovoznit embedded systém sloužící jako interpreter asset administration shellu výrobku pro I4.0 testbed, který zprostředkuje přístup k jeho informačnímu modelu autonomní výrobní buňce.

1. Seznamte se se stávajícím řešením univerzální NFC čtečky.
2. Seznamte se s principy AAS
3. Navrhněte vhodný model pro AAS pro i4.0 testbed.
4. Navrhněte a implementujte FW.
5. Otestujte funkčnost a zhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

Pavel Herout: Učebnice jazyka C, KOPP, 2004, IV. přepracované vydání, ISBN 80-7232-220-6

Dle pokynů vedoucího práce.

ZVEI Details of the Asset Administration Shell. [(accessed on 20 August 2021)]; Available online: https://www.zvei.org/en/press-media/publications/details-of-the-asset-administration-shell/

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 22.5.2024

Vedoucí práce: Ing. Ondřej Baštán

Ing. Miroslav Jirgl, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce představuje inovativní přístup k integraci modelu Asset Administration Shell (AAS) do vestavěného systému, který je klíčovou součástí testbedu Průmyslu 4.0 - Barman. V tomto kontextu vystupuje AAS interpreter jako agent výrobku. Práce začíná podrobným popisem konceptů a terminologie spojené s tímto tématem. Následuje detailní analýza stávající a nové verze hardwaru univerzální NFC čtečky. Hlavní část práce se zaměřuje na návrh AAS modelu, který je následně implementován do firmwaru mikroprocesoru. Součástí tohoto procesu bylo také vytvoření OPC Unified Architecture (OPC UA) serveru, který sloužil k důkladnému testování celkového systému. Výsledky testování potvrdily úspěšnou implementaci a funkčnost systému. Všechny teoretické scénáře proběhly dle navrženého modelu AAS, což potvrzuje úspěch tohoto přístupu. Tato práce tak představuje významný přínos v oblasti návrhu vestavěných systémů v rámci Průmyslu 4.0.

KLÍČOVÁ SLOVA

AAS, Testbed, Robotický barman, Průmysl 4.0, OPC UA, MIFARE, ESP32, PN532, NFC

ABSTRACT

This bachelor thesis presents an innovative approach to integrating the Asset Administration Shell (AAS) model into an embedded system that is a key part of the Industry 4.0 testbed - Barman. In this context, the AAS interpreter acts as a product agent. The paper begins with a detailed description of the concepts and terminology associated with this topic. This is followed by a detailed analysis of existing and new versions of the universal NFC reader hardware. The main part of the thesis focuses on the design of the AAS model, which is then implemented in the firmware of the microcontroller. This process also included the creation of an OPC Unified Architecture (OPC UA) server, which was used to thoroughly test the overall system. The results of the testing confirmed the successful implementation and functionality of the system. All theoretical scenarios were run according to the proposed AAS model, confirming the success of this approach. Thus, this work represents a significant contribution in the field of embedded system design in the context of Industry 4.0.

KEYWORDS

AAS, Testbed, Robotic bartender, Industry 4.0, OPC UA, MIFARE, ESP32, PN532, NFC

CHMELAŘ, Luboš. *AAS interpreter pro testbed I4.0*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2024. Vedoucí práce: Ing. Ondřej Baštán

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Luboš Chmelař
VUT ID autora:	240355
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	AAS interpreter pro testbed I4.0

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

S hlubokým uznáním a vděčností bych rád vyjádřil své poděkování vedoucímu mé bakalářské práce, Ing. Ondřeji Baštánovi. Jeho odborné vedení, konstruktivní konzultace a inspirativní rady byly klíčové pro úspěšné dokončení této práce.
Mé upřímné díky patří také mé rodině a všem přátelům, jejichž neochvějná podpora a pomoc byly pro mě během tohoto procesu neocenitelné.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V současné době se svět průmyslu nachází na prahu čtvrté průmyslové revoluce, známé také jako Průmysl 4.0. Tato revoluce přináší nové možnosti a výzvy, které vyžadují inovativní řešení. Jedním z klíčových aspektů Průmyslu 4.0 je koncept Asset Administration Shell (AAS), který slouží jako digitální reprezentace fyzického zařízení.

Tato bakalářská práce se zaměřuje na návrh a SW implementaci embedded systému, který slouží jako AAS interpreter pro testbed I4.0 - Barman. Cílem tohoto systému je zprostředkovat možnost komunikace a spolupráce mezi výrobkem (sklenicí) a autonomními buňkami.

Práce je rozdělena do několika kapitol, které budou pokrývat různé aspekty tohoto tématu. První kapitola se bude věnovat popisu Průmyslu 4.0 a jeho základních myšlenek. Dále se bude zabývat charakteristikami Průmyslu 4.0. Konceptu AAS a následně popisu testbedu I4.0 - Barman, kde bude také charakterizována funkce AAS interpreteru.

Druhá kapitola se bude věnovat OPC Unified Architecture (OPC UA), což je komunikační protokol používaný v Průmyslu 4.0. Tato kapitola bude popisovat historii a cíle OPC UA a také se bude zabývat aktuálním stavem tohoto protokolu, kde bude popsán jakým způsobem je prostor adresován v uzlech. Zároveň budou popsány normou definované druhy uzlů. Následně budou popsány sady služeb, které budou využívány v komunikaci mezi OPC UA klientem a OPC UA serverem.

Třetí kapitola se bude zaměřovat na technologii Near field communication (NFC), která bude v této práci využita jako způsob komunikace s pamětí na výrobku. Tato kapitola obsahuje popis Radio frequency identification (RFID) a NFC, klasifikaci zařízení a následně výpisem populárních NFC tagu.

Čtvrtá kapitola se bude věnovat hardwarové části práce. Zde bude proveden rozbor stávající verze NFC čtečky a stanoveny problémy návrhu. Následně bude proveden rozbor nové verze hardwaru, kde bude popsán blokový diagram návrhu, rozmístění prvků na již hotovém zařízení. Následně budou stanoveny možnosti vylepšení návrhu do následující verze.

Pátá kapitola se bude zabývá návrhem modelu AAS pro testbed I4.0 - Barman. Zde budou určeny požadavky na tento model a následně bude navržena datová struktura úložiště NFC a způsob interpretace a s tím spojené metody protokolu OPC UA pro komunikaci.

Šestá kapitola se bude věnovat softwarové implementaci. Budou zde popsány hlavní součásti programu a následně funkcionality, které jsou v programu obsaženy.

Sedmá se bude věnovat popisu testovacího stanoviště, následně testovacích metod a z nich vyplývajících výsledků.

1 Průmysl 4.0

První teoretická kapitola se bude zabývat popisem I4.0 (Průmysl 4.0) a jeho praktickým nasazením na testbedu I4.0 - Barman.

Průmysl 4.0 znamená digitální transformaci pro výrobce a producenty, integrující trendy jako bezpečnostní systémy, internet věcí a chytré továrny. Vznikl v roce 2011 s cílem inovovat německou výrobu prostřednictvím nových technologií. Zatím ho lze označit za iniciativu.

Tento koncept nejen investuje do efektivity výroby, ale také mění fungování firem. Celý hodnotový řetězec je ovlivněn, od surovin až po zpracování, s cílem vytvořit inteligentní továrny propojující fyzický, digitální a biologický svět.

Průmysl 4.0 se zaměřuje na urychlení výroby, zefektivnění procesů a orientaci na zákazníka, přičemž překračuje hranice automatizace a optimalizace.[1]

1.1 Základní principy

Zde jsou vypsány základní principy, které by mělo zařízení v I4.0 splňovat:

1. Interoperabilita - Jedná se o schopnost prvků I4.0 mezi sebou komunikovat a spolupracovat.
2. Virtualizace - Lze vytvořit virtuální model chytré továrny a aplikovat na něj reálně získaná data.
3. Decentralizace - Rozhodovací schopnost se přesune z hlavního prvku na úroveň jednotlivých strojů.
4. Operace v reálném čase - Je nutnost reagovat na požadavky okamžitě s nulovou odezvou systému.
5. Orientace na služby - V I4.0 se musí podnik orientovat na poskytování služeb (např. servis), tak i na nákup služeb (např. cloudové služby)
6. Modularita - Celkový systém v I4.0 se musí umět adaptovat na nové požadavky a změny. Modularita nám slouží k jednoduchému přidávání nových technologií.[3]

1.2 Charakteristiky

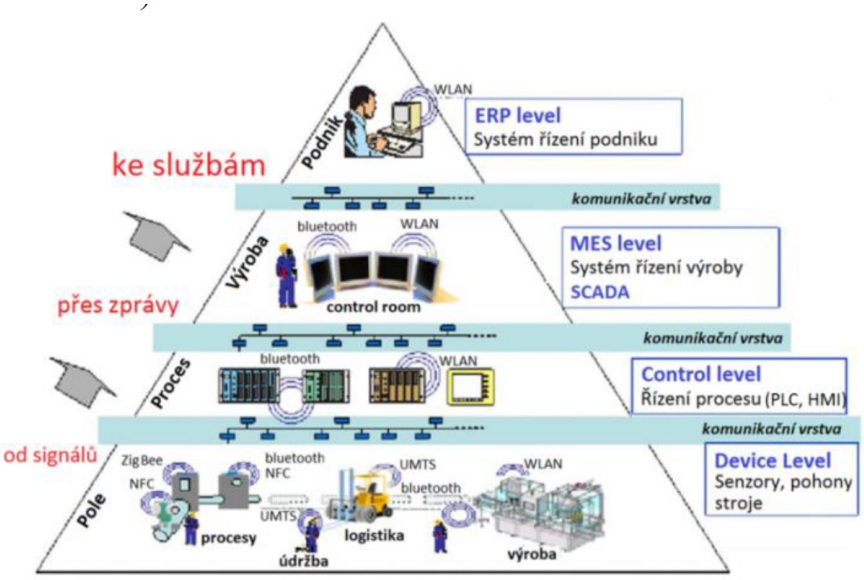
Koncept I4.0 lze rozdělit na tři základní pilíře, kde se vyskytuje:

- Vertikální propojení
- Horizontální propojení
- Průnikové inženýrství

Tyto pojmy jsou vysvětleny v následujícím textu.

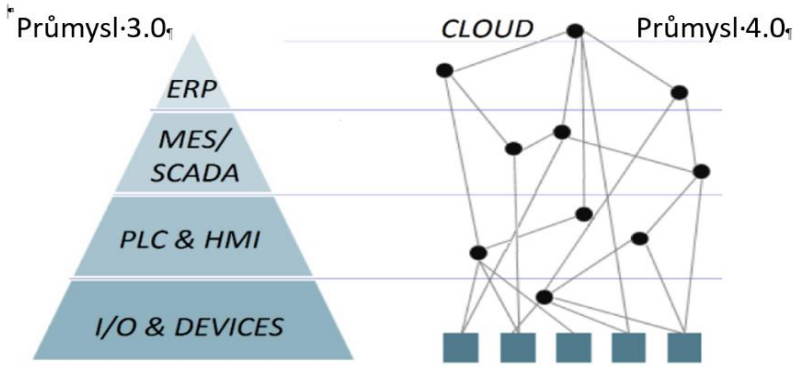
1.2.1 Vertikální propojení

Jedná se o rozšíření pyramidy, která znázorňuje 4 vrstvy automatizačních funkcí výroby (obrázek 1.1). Někdy je uváděno 5 vrstev. V takovém případě je vrstva MES a SCADA rozdělena.



Obr. 1.1: Znázornění automatizační pyramidy průmyslu 3.0[2]

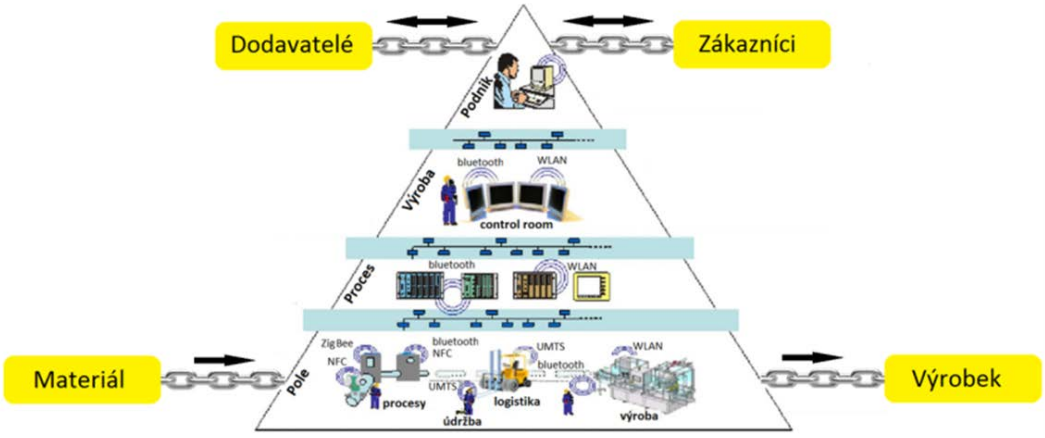
Tahle pyramida se neustále rozvíjí ve směru toku dat mezi vrstvami a v ohledu aktivit na jednotlivých úrovních. Zde celé fungování pyramidy řídí centrální MES. U konceptu I4.0 jsou dva názory na existenci MES. Jeden říká, že by měl být zachován a jeho nová funkce bude spojování nových dat a tvořit mezi nimi souvislosti. Druhý názor s ním nepočítá. Tedy všechny výrobní prostředky a výrobky komunikují v určené síti mezi sebou (znázorněno na obrázku 1.2).[2]



Obr. 1.2: Vertikální propojení prvků výroby průmyslu 4.0 v porovnání s průmyslem 3.0[2]

1.2.2 Horizontální propojení

Pojem horizontální integrace lze chápat jako propojení obchodních řetězců. Tedy způsob jakým se materiál dostane od dodavatele k procesu, kterým jej zpracováváme, a následná distribuce výrobku k zákazníkovi (znázorněno na obrázku 1.3).



Obr. 1.3: Znázornění horizontální integrace v průmyslu 3.0[2]

V I4.0 přesahuje horizontální integrace vnitřní operace podniku. Lze tedy, kromě zajištění materiálu od dodavatele a převzetí výrobku zákazníkem, spravovat více služeb. Lze propojit všechny články od dodavatelů (řídit více dodavatelů mezi sebou, reagovat na výpadky) až po odběratele (zákazníka) a zajistit následný servis. Lze díky tomu reagovat v reálném čase na požadavky odběratele a specifická přání.[2]

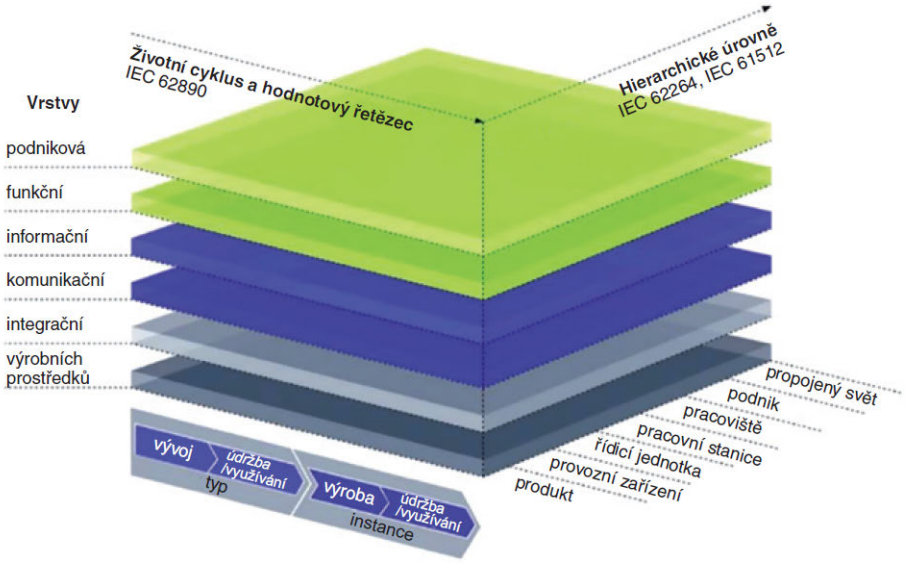
1.2.3 Průnikové inženýrství

Jedná se o kombinaci vertikální a horizontální integrace ve významu rámce celého životního cyklu výrobku (služby). Lze díky tomu poskytnout okamžitou zpětnou vazbu konstruktérům v případě problému s výrobním procesem výrobku. Dříve byl velký problém v paralelizaci inženýrských disciplín. Nyní díky novým nástrojům (např. program COMOS) lze tyhle procesy provádět paralelně.

Další velkou výhodou průnikového inženýrství I4.0 je možnost simulace. Simulace slouží k tomu, abychom mohli v digitální podobě ověřit různé výrobní postupy a simulovat krajní situace.[2]

1.3 Referenční model RAMI 4.0

Trojrozměrný model popisuje základní aspekty I4.0. Všechny prvky jsou rozděleny do menších a jednodušších struktur. Díky modelu můžeme specifikovat určité požadavky na jednotlivé prvky.[4]



Obr. 1.4: Model RAMI 4.0[4]

- Pravá horizontální osa - hierarchické vrstvy dle normy IEC 62264 - *Integrovaný systém podnikového řízení*. Vrstvy odpovídají jednotlivým funkcím v podniku.
- Levá horizontální osa - životní cyklus zařízení dle IEC 62890 - *Správa životního cyklu produktů a systémů používaných pro měření, řízení a automatizaci v procesním průmyslu*. Osa je rozdělená na dvě podskupiny: typ a instance. Typ se přemění na instanci, když se přejde z testování prototypu na sériovou výrobu.
- Vertikální osa - vlastnosti zařízení. Šest vrstev určuje vlastnosti zařízení rozdělené do podskupin. Takový způsob ztvárnění je inspirovaný dle informační techniky.[4]

Model RAMI 4.0 je základ pro další kroky k realizaci iniciativy I4.0. Slouží k realizaci následujících vlastností:

- Identifikace - Představuje klíčový prvek pro vzájemnou spolupráci zařízení. Díky identifikaci zařízení dokáží autonomně vyhledávat další zařízení a spolupracovat s nimi.
- Sémantika - Komunikace mezi systémy různých výrobců vyžaduje mít jednotnou sémantiku a společnou syntaxi pro data.
- Kvalita služeb - Je potřeba definovat kritické služby u kterých je nutné zajistit kvalitu QoS.[4]

1.4 Asset Administration Shell

Spojení digitálního a skutečného světa zajišťuje obálka AAS. Obálka nám určí vlastnosti a charakteristiky modelů I4.0.[6]

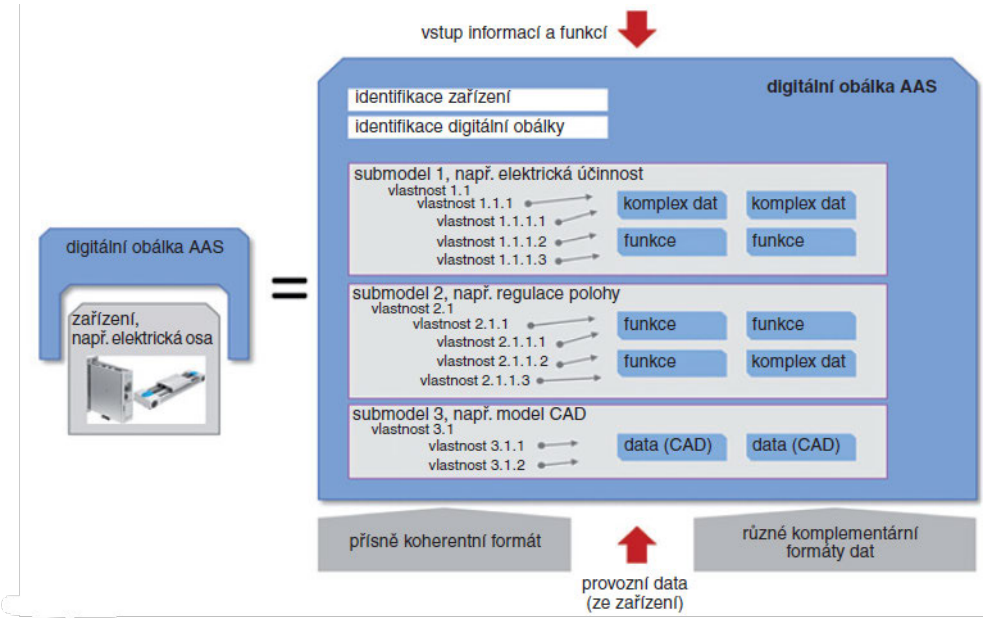
1.4.1 Model Industry 4.0 component

Jedná se model iniciativy I4.0 pro pomoc výrobcům a implementátorům navrhovat HW a SW komponenty pro I4.0. Vychází z RAMI 4.0. Model slouží k lepšímu popisu kyberneticko-fyzikálních prvků a umožňuje popis komunikace mezi virtuálními a kyberneticko-fyzikálními objekty a procesy a to vše v reálném čase.

Realizace spočívá v tom, že každá součást systému I4.0 má v celém životním cyklu obal dat, která jsou dostupná všem prvkům výrobního řetězce. Pro I4.0 je nutnost, aby každý objekt byl implementován jako součást I4.0. Objekt se často implementuje bez vlastností I4.0 a obklopí se datovou schránkou (Asset Administration Shell) a teprve tehdy se stane součástí I4.0.[5]

1.4.2 Definice AAS

Digitální obálka (AAS) vychází z konceptu modelu Industry 4.0 component. Obálka popisuje objekt v různých submodelech (identifikace, komunikace, bezpečnost, energetická účinnost, atd.). Každý submodel je definován unikátním identifikátorem a obsahuje množství jednoznačných charakteristik. Všechny vlastnosti a jejich hodnoty musí brát v úvahu platné normy IEC a ISO.[6]



Obr. 1.5: Struktura metamodelu AAS[6]

1.4.3 Typy AAS dle přístupu k datům

Dělení AAS dle přístupu jakým se dostaneme k datům můžeme rozdělit na následující kategorie:

Pasivní AAS

Pasivní AAS pouze nabízí informace a popis zařízení bez možnosti inicializovat vlastní akce. Přístup k datům si řídí pouze externí inicializátor žádosti o data.[7]

Pasivní AAS s API přístupem

Jedná se o rozšíření pasivní AAS, kde pro vnější svět není viděna vnitřní struktura dat, tak jak je uložena v paměti, ale je viditelná přes API, lze tak zabezpečit operace nad daty inicializované externím žadatelem o data.[7]

Aktivní AAS

Aktivní AAS má uvnitř své struktury pasivní AAS, ovšem nad daty je postaven systém na zpracování a inteligentní komunikační rozhraní. Zařízení komunikuje pomocí zpráv, které jsou vytvořeny dle jazyka I4.0.[7]

1.5 Praktický příklad I4.0 - Testbed I4.0 - Barman

Projekt Testbed I4.0 - Barman byl vytvořen na VUT FEKT, aby se studenti automatizace připravili již během studia na běžnou práci v automatizační oblasti. Zde na testbedu si vyzkouší praktické a teoretické pochopení průmyslu 4.0.[8]

1.5.1 Popis testbedu

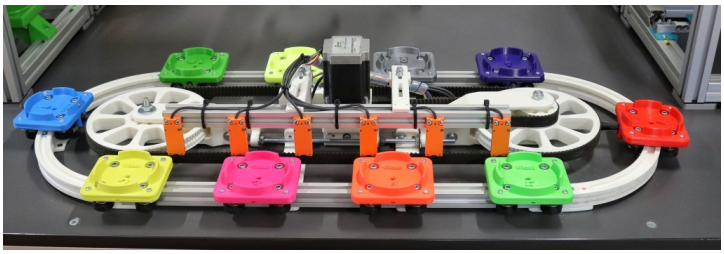
Celý zkušební stůl je postavený na pracovní ploše o rozměrech 2000 x 1000 mm uzavřené v kleci. Pod tímhle prostorem se nachází stejně velká oblast, která poskytuje podpůrné prvky IT technologií.[8] Zde se nachází několik autonomních jednotek (buněk). Buňky se skládají z hliníkových rámů a vnitřních systémů. Zároveň na vstupu každé buňky se nachází čtečka NFC čipů. Ta bude sloužit ke čtení čipů na spodní straně skleničky. Touto funkcionalitou se bude zabývat má následující bakalářská práce.

Obr. 1.6: Testbed I4.0 - Barman

Testbed I4.0 se skládá z několika částí. V následujících bodech si je popíšeme a určíme některé jejich vlastnosti.

Dopravník

Jedná se o zařízení k přepravě skleniček od manipulátoru směrem k zákazníkovi a zpět. Je to způsob, jak zabezpečit bezpečnost demonstrátoru (zákazníci nemusí sahat do prostoru manipulátoru). Lze jej využít jako úložiště rozpracovaných nápojů. Na dopravník lze uložit až deset sklenic.[10]



Obr. 1.7: Dopravník Testbedu I4.0 - Barman[9]

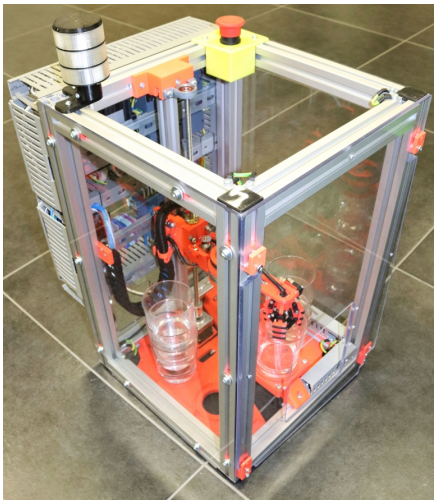
Manipulátor

Manipulátor se stará o transport sklenic mezi výrobními buňkami a dopravníkem. Zařízení operuje jen v oblasti, kde nemají přístup zákazníci z důvodu bezpečnosti. Jedná se o SCARA robota EPSON H544BN. Jako efektor má uchopovací mechanizmus na sklenice.[10]

Obr. 1.8: Manipulátor Testbedu I4.0 - Barman[9]

Sklad sklenic

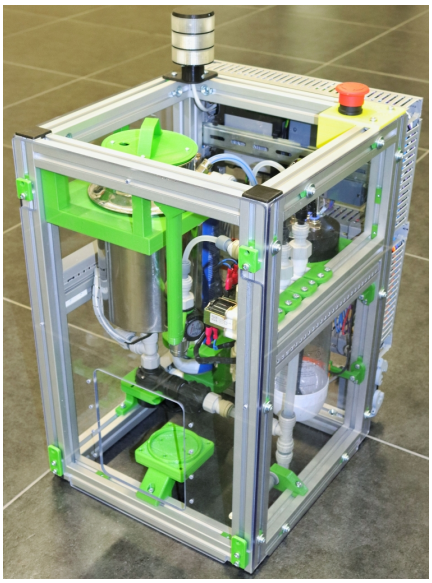
Skladovací buňka sklenic má za funkci vydávat, přijímat a třídit čisté a špinavé sklenice. Dvouosý manipulátor může manipulovat přes všechny čtyři zásobníky sklenic. Celková kapacita skladu sklenic je 24 skleniček.[10]



Obr. 1.9: Sklad sklenic Testbedu I4.0 - Barman[9]

Sodovač

Výrobní buňka sodovače se stará o výrobu a skladování perlivé vody, přičemž dokáže naplnit určenou skleničku. Perlivá voda se zde vyrábí pomocí CO2. Voda se vyrábí v dávkách, které odpovídají velikosti láhve od firmy SodaStream.[9][10]



Obr. 1.10: Sodovač Testbedu I4.0 - Barman[9]

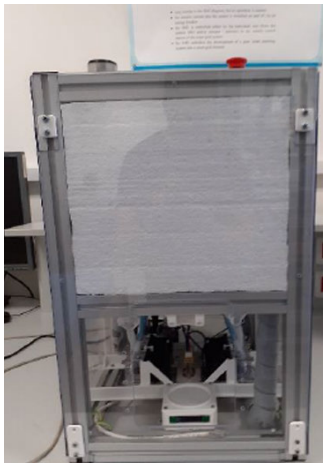
Shaker

Pro potřeby protřepání a promíchání obsahu sklenice je určena buňka shakeru. Jedná se o zařízení poháněné stejnosměrným motorem, kde výstup motoru je transformován pomocí klikové hřídele z rotačního na lineární pohyb. Shaker nabízí funkci opláchnutí horního víka, aby se nenamíchaly zbytky tekutiny do dalšího nápoje.[9][10]

Obr. 1.11: Shaker Testbedu I4.0 - Barman[9]

Zásobník nealkoholických nápojů

Jako sklad tekutin pro míchání nápojů je zde sestaven zásobník nealkoholických nápojů. Ten obsahuje čtyři nádržky na tekutiny, které jsou chlazeny pomocí chladicího okruhu. Dávkovač poskytuje větší množství (1 až 3 dl) tekutiny.[8][10]



Obr. 1.12: Zásobník nealkoholických nápojů Testbedu I4.0 - Barman[10]

Zásobník alkoholických nápojů

Buňka pro výdej alkoholických nápojů je složena ze skladu lihovin a sirupů. Buňka vydává tekutiny v malém množství (2 až 5 cl) do sklenice. Dále obsahuje dvoupatrový otočným rám, který nese láhve s tekutinami.[8]

Obr. 1.13: Zásobník alkoholických nápojů Testbedu I4.0 - Barman[9]

1.5.2 Interpreter

Navrhovaný AAS interpreter bude sloužit v rámci Testbedu I4.0 jako čtečka NFC tagů nalepených na dně sklenice. Lze tedy konstatovat, že AAS interpreter bude vystupovat jako agent výrobku (sklenice). AAS Interpreter se bude nacházet na vstupu u každé buňky. Důvod ke vzniku intepreteru jako embedded zařízení je přechod z Průmyslu 3.0 na Průmysl 4.0. Čtečka byla zvolena z cenového důvodu, kdy je mnohem levnější na sklenici umístit datové úložiště ve formě NFC tagu, místo drahého výpočetního zařízení na každé sklenici. Zároveň AAS interpreter musí splňovat všechny principy (popsány v kapitole 1.1 a v praktickém pojetí v kapitole 5.1) přesunu Testbedu I4.0 do Průmyslu 4.0.

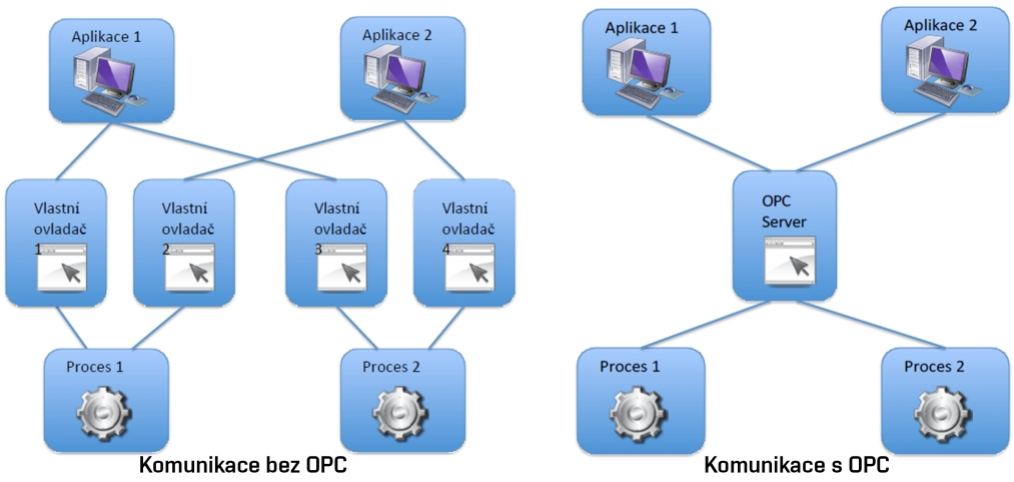
2 OPC Unified Architecture

Historie a důvody vzniku iniciativy OPC a následný stupeň vývoje do OPC UA je popisován v následující kapitole. U OPC UA jsou popsány výhody oproti původním verzím.

2.1 Založení OPC(OLE for process)

Koncem 80. let 20. století se začaly rozšiřovat sítě, protokoly a sběrnice. Spoustu z nich se stalo univerzálními komunikačními protokoly v průmyslu, ale některé se využívaly jen ve specifických oblastech. Bylo potřeba vyměňovat data pro SCADA, která zpracovává data a to z různých protokolů a sběrnic. Musel se zajistit pro každé nové zařízení ovladač, který by zařízení k SCADA připojil. Problém nastal při každé změně ve specifikaci protokolu, která vedla k nefunkčnosti komunikace, a kvůli tomu se musely provést úpravy v ovladači.

V roce 1995 se vytvořila pracovní skupina (poté nadace) OPC, která měla za úkol vytvořit standard, který popisuje architekturu server-klient. Standard měl za úkol sběr reálných dat ze zařízení a předávat je do nadřazeného řídícího systému (např. SCADA). První specifikace se vytvořila v roce 1996 a dostala název OPC-DA (Data Access).[12]



Obr. 2.1: Rozdíl komunikací bez OPC (vlevo) a komunikace s OPC (vpravo)[12]

2.2 Historie OPC UA(OPC Unified Architecture)

Ačkoli se využívaly různé varianty OPC klient-server, obzvláště DA a A&E (Alarm & Event), tak přesto vznikla v roce 2006 nová koncepce, která spojila různé architektury OPC. Nebylo již nutné kvůli téhle koncepci mít pro každou architekturu jiný adresní prostor. Tvůrci zaintegrovali také nové bezpečnostní prvky a vylepšili klíčové vlastnosti (odolnost proti chybám, redundance, interoperabilita a webové technologie), které odpovídají pozdější iniciativě I4.0. Zároveň se vylepšila schopnost pracovat v reálném čase tím, že protokol umožňuje odesílání zpráv point to multipoint.

Nadace OPC se rozhodla všechny předchozí verze OPC přejmenovat na *Classic OPC*. Ačkoliv verze *Classic* OPC byly hojně využívány v průmyslu, tak nebyly schváleny jako norma IEC. Verze OPC UA se postupně přebírala jako norma IEC a nakonec se stala normou IEC 62541. Taky se začala měnit zkratka OPC z OLE for process na *Openness, Productivity Collaboration.*[12][15]

2.3 Cíle

Aby bylo možné začít definovat parametry OPC UA, muselo být stanoveno množství určitých cílů:

2.3.1 Sjednocení starších architektur

Jak již bylo výše zmíněno, OPC UA si dala za cíle sjednotit různé architektury, tedy i adresní prostory. Norma specifikuje objekt, který obsahuje proměnné (OPC-DA/OPC-HDA), metody (OPC-A&E) a události (OPC-Commands).[12]

Obr. 2.2: OPC UA model objektu[12]

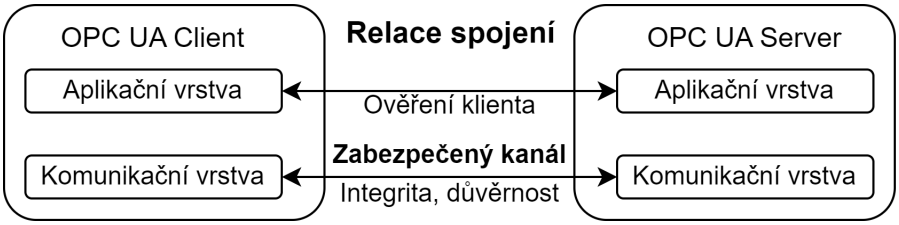
2.3.2 Zabezpečení

Bezpečnost byla v minulosti velmi podceňována. Sice se bezpečnostní specifikace oznámila, ale byla oznámena příliš pozdě na to, aby se aplikovala do průmyslu na

existující řešení. Jedno z řešení, které se dříve nabízelo, bylo vytvořit bezpečný datový tunel mezi klientem a serverem.

Zabezpečení u OPC-UA je aplikováno v dvou různých fázích komunikace (znázorněno na obrázku 2.3):

- Aplikační vrstva - slouží k ověření klienta, který se připojuje k serveru. Ověření klienta probíhá např. pomocí hesla, certifikátu, atd. Díky autorizaci uživatelů můžeme definovat specifická práva pro každého klienta.
- Komunikační vrstva - zde se vytvoří bezpečný tunel dat, čím můžeme zajistit důvěrnost, integritu a autorizaci aplikace u klienta. Důvěrnost nám slouží k zajištění zašifrování dat, aby se k nim dostali pouze autorizovaní klienti a ostatní nemohli přečíst odeslaná data. Integrita nám zajišťuje, aby se zpráva nezměnila během přenosu. Zde se můžou aplikovat různé metody ochrany a obnovy dat (např. kontrolní součty, podpisy, atd.). [12]

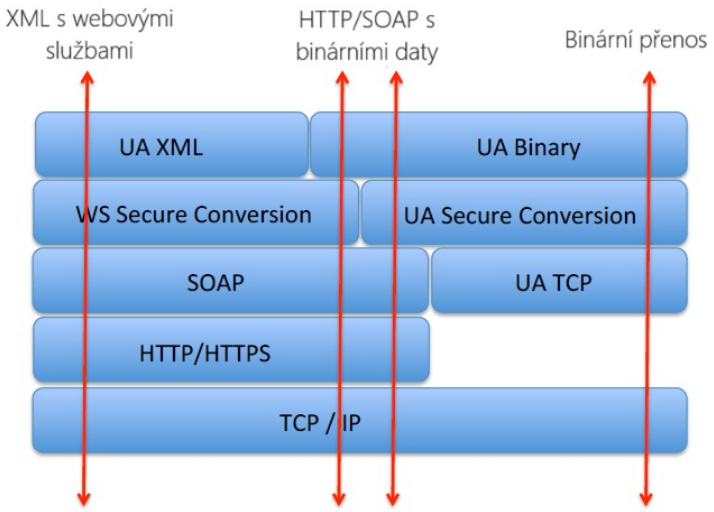


Obr. 2.3: Zabezpečení relace OPC UA [12]

2.3.3 Způsoby přenosu dat

Aby byla zajištěna interoperabilita, jsou dostupné různé způsoby přenosu dat, aby nezáleželo na konkrétním přenosovém protokolu nebo operačním systému. Data se můžou přenášet pomocí následujících způsobů:

- Binárně
- XML pomocí webových služeb
- HTTP/SOAP s binárními daty [12]



Obr. 2.4: Přenos dat v OPC UA[12]

2.4 Aktuální stav

Nyní je specifikace OPC UA komunikace mezi OPC UA serverem a OPC UA klientem provedena jako architektura orientovaná na služby (SOA - Service Oriented Architecture), tedy může využít definované služby, které mu server nabízí a získat zpět na každý dotaz odpověď. Tímhle způsobem je vytvořen abstraktní komunikační model. [13]

Aby byla implementace protokolu OPC UA co nejjednodušší, není potřeba implementovat do každé aplikace všechny funkce, které jsou specifikovány. [13]

2.4.1 Adresování prostoru - uzlů

Aby bylo předem definováno adresování dat dostupných na OPC UA serveru bylo stanoveno, že adresní prostor, který tvoří sdílená data, je tvořen uzly (Nodes).

Každý uzel musí mít vlastní identifikátor (NodeId). Ten se skládá z následujících parametrů:

1. **Jmenný prostor (Namespace)** - Označuje kontext uzlu pomocí URI. Lze využít stejný identifikátor i na jiném serveru. ()
2. **Identifikátor (Identifier)** - Hodnota identifikátoru může být více druhů:
   (a) **Číselně** - Kladná číselná hodnota identifikátoru pomocí proměnné typu *UInt32* (např. 5)
   (b) **Textový řetězec** - identifikátor je vyjádřen pomocí proměnné typu *String*. Je zde nutnost myslet na to, že identifikátor ve formě textového

řetězce je case sensitive[1].(např. "MujUzel1")

(c) **GUID (Globally Unique Identifier)** - Unikátní identifikátor ve formě struktury, která obsahuje 4 datové proměnné. Struktura obsahuje proměnné Data1 (*UInt32*), Data2 (*UInt16*), Data3 (*UInt16*) a Data4 (*Byte[8]*). Data můžou být zapsána ve formátu:
<Data1>-<Data2>-<Data3>-<Data4[0:1]>-<Data4[2:7]>
(např: $C496578A - 0DFE - 4B8F - 870A - 745238C6AEAE_{16}$)

(d) **Opaque** - Jedná se o speciální typ identifikátoru, kdy nemáme předem stanovený datový typ, kvůli tomu nemusí být hodnota lidsky čitelná. Jedná se o datový typ *ByteString*. Datový typ *ByteString* je strukturálně stejný datový typ jako jednorozměrné pole datového typu *Byte*.
(např. $\{123, 21\}_{10}$) [14]

2.4.2 Třídy uzlů - NodeClass

NodeClass je skupina parametrů pro specifické množiny uzlů. Každý uzel náleží některé třídě uzlů. Třídy definují atributy a reference. [14]

Atributy

Jedná se o datové prvky, které popisují vlastnosti uzlu. Atributy se můžou pomocí služeb číst, přepisovat, dotazovat[2], odebírat a monitorovat. Definice atributu se v dokumentaci skládá z:

1. **Identifikátor atributu (Attribute Id)** - Jeden z 27 dostupných identifikátorů, které jsou definovány v dokumentaci (např. 1 - NodeId, 5 - Description, 22 - UserExecutable, ...)
2. **Názvu (name)** - Atribut musí být popsán názvem, který by měl popisovat jeho funkci nebo účel.
3. **Popisu (description)** - Zde je popsán atribut.
4. **Datový typ (data type)** - Datový typ hodnoty atributu
5. **Povinnost atributu (mandatory/optional indicator)** - Určuje jestli je atribut povinný. [14]

Reference

K propojení a odkázání jednoho uzlu na druhý se využívají reference. Na rozdíl od atributu jsou reference definovány jako uzel speciálního typu *ReferenceType*. [14]

[1]Řetězec, kde záleží na velikosti textových znaků("text"a "Text"jsou rozdílné textové řetězce).

[2]Dotazování (Query) - Klient může přistupovat k datům na serveru bez znalosti schématu pro schéma interního ukládání dat.[14]

2.4.3 Standardizované třídy uzlů

V adresním prostoru se nacházejí standardizované třídy uzlů. Třídy, které využívají objekty, lze rozdělit do 3 kategorií:

1. **Definující instanci** - Jsou využity ke konkrétní definici objektu (např. máme typ instance *"Převodovka"* a tedy instance je *"Převodka1"*).
2. **Definující typy instancí** - Jedná se o typ, který obecně popisuje vlastnosti instancí určitého typu (např. určují obecné vlastnosti všech instancí typu *"Převodovka"*).
3. **Definující datových typů** - K určení vlastnosti určitých typů instancí využijeme právě tenhle typ tříd. (např. můžeme mít třídu *"PomerPrevodu"*, která určuje jaký má poměr převodů typ instance *"Převodovka"*).[14]

Dále budou popsány základní typy tříd, které jsou popsány v dokumentaci OPC UA:

1. **Základní třída (Base NodeClass)** - Jedná se o základní třídu, ze které vycházejí všechny ostatní typy tříd. Definuje základní povinné a nepovinné atributy. Neobsahuje žádné reference.
2. **Třída typu reference (ReferenceType NodeClass)** - Určuje uzly reference, které jsou viditelné v adresovém prostoru.
3. **Zobrazovací třída (View NodeClass)** - Slouží k vyfiltrování dat z velkého objemu dat. Celý adresní prostor je základní zobrazení. Všechny uzly, které jsou obsaženy v zobrazovacím uzlu musí být přístupné přímo z určitého uzlu zobrazení.
4. **Objektová třída (Object NodeClass)** - Objekty slouží k reprezentování systému, částí systémů, objektů z reálného světa a SW objektů.
5. **Třída typu proměnné (VariableType NodeClass)** - Slouží k definici datového typu.
6. **Třída proměnné (Variable NodeClass)** - Uzel reprezentuje určitou hodnotu určitého datového typu.
7. **Třída datové typu (DataType NodeClass)** - Popisuje spojení uzlu typu proměnné s uzlem proměnné.
8. **Třída metody (Method NodeClass)** - Slouží k vytvoření metody, kterou lze volat pomocí služby Call.[14]

2.4.4 Sady služeb

Protože protokol OPC UA je Service Oriented Architecture, je nutné mít definovány služby, které bude server nabízet. Každý požadavek služby musí mít hlavičku požadavku (RequestHeader) a hlavičku odpovědí (ResponseHeader). V hlavičce odpovědi se nachází výsledek služby, který je definován v datové proměnné

typu *StatusCode*.[14]

Mohou nabývat tří hodnot:

1. **Bad** - Pokud proces služby byl neúspěšný bude navrácena tahle hodnota. Server zároveň odešle *ServiceFault* parametr, který popisuje chybu, dle nastaveného parametru textitreturnDiagnostics.
2. **Uncertain** - Když část služeb bude úspěšná, bude navrácena tahle hodnota.
3. **Good** - Když všechny části služeb budou úspěšné, bude navrácena tahle hodnota. [14]

OPC UA klient by měl pokaždé zkontrolovat návratovou hodnotu, kterou dostane v hlavičce odpovědi od OPC UA serveru. Dále jsou definovány následující sady služeb:

1. **Průzkumná sada služeb (Discovery Service Set)** - Slouží k zjišťování údajů o samotném OPC UA serveru. Zároveň získáme i možnosti připojení. Díky téhle sadě hledáme *Discovery Endpoints*, které by měl mít každý server. Zároveň se můžou OPC UA servery registrovat u předem známého Discovery Serveru. Později díky tomu může klient nalézt dostupné servery pomocí služby *FindServers*.
2. **Sada služeb pro zabezpečený kanál (SecureChannel Service Set)** - Jedná se o sadu služeb ke správě komunikačního kanálu mezi OPC UA klientem a OPC UA serverem. Komunikační kanál zajišťuje důvěrnost a integritu všech přenesených dat. Zabezpečený kanál je nastaven dle *EndpointDescription*, který získá od *Endpoint* OPC UA serveru. Zároveň je ověřen certifikát.
3. **Relační sada služeb (Session Service Set)** - Služba sloužící k správě relací mezi OPC UA klientem a OPC UA serverem. Před vytvořením relace je potřeba navázat komunikační tunel skrze *SecureChannel*. Zároveň se zde ověřuje uživatel dle zadaného hesla.
4. **Sada služeb pro správu uzlů (NodeManagement Service Set)** - Díky téhle službě lze pracovat s uzly a referencemi mezi nimi. Všechny přidané uzly jsou na OPC UA serveru přítomny po dobu konání relace.
5. **Zobrazovací sada služeb (View Service Set)** - Aby OPC UA klient dokázal procházet uzly v adresní prostor nebo zobrazení uzlů části adresního prostoru je využívání téhle sady služeb.
6. **Dotazovací sada služeb (Query Service Set)** - K získání dat, které splňují pouze určité požadavky, slouží tahle služba. Díky téhle sadě se sníží objem dat, který se OPC UA klientu dostane od OPC UA serveru.
7. **Sada služeb pro atributy (Attribute Service Set)** - Přístup k atributům obsažených v uzlech slouží tahle sada služeb. Hodnoty atributů lze jak číst, tak i zapisovat. Pokud máme požadavek na data, aby měli nějaké stáří, může se nastavit parametr *maxAge*, který nastaví maximální dobu stáří, kdy lze data

považovat za platná. Lze přistupovat k historickým hodnotám, díky téhle sadě.

8. **Sada služeb pro metody (Method Service Set)** - Aby se mohli volat služby na OPC UA serveru, je stanovena zmíněná sada služeb. Nachází se zde jen jedna služba, a to *Call*. Metodě jsou předány vstupní argumenty, a jsou navráceny výstupní argumenty. Argumenty jsou definovány v *Properties* uzlu.
9. **Monitorovací sada služeb (MonitoredItem Service Set)** - Jedná se o sadu k přihlášení monitorování uzlů. *Notification* je datová struktura, která popisuje výskyt změn a proběhlé události. Lze nastavit jak má server vzorkovat vstupní data, jakým režimem sledovat uzel, jestli mají být vstupní data filtrována a jaké má odebírací fronta parametry. Pokud OPC UA server vyhodnotí, že se informace má odeslat OPC UA klientu, přepošle je do odebírací fronty.
10. **Odebírací sada služeb (Subscription Service Set)** - Odebírání je mechanizmus k hlášení oznámení od OPC UA serveru k OPC UA klientu. Tahle sada slouží k jeho správě.

    Jedná se o frontu, která pomocí *Publish* odešle data k *OPCUA* klientu. [14]

3 NFC technologie (Near field communication)

Kapitola se zabývá popisem RFID a NFC. Je určeno základní dělení zařízení v oblasti NFC a jsou vysvětleny režimy přenosu. Na konci kapitoly jsou vypsány nejpopulárnější příklady NFC tagů.

Hlavní motivací pro vytvoření NFC bylo integrovat osobní službu jako je platební karta do mobilních telefonů. Z důvodu zabezpečení dat bylo potřeba vytvořit nové rozhraní pro přenos dat na velmi krátké vzdálenosti.[16]

3.1 RFID

RFID(Radio frequency identification) využívá pro komunikaci rádiové vlny, kdy jsou přenášeny data mezi RFID čtečkou a RFID vysílačem. RFID tag (označení pro nosný materiál společně s anténou/cívkou a integrovaným obvodem) je často osazován na předmět, který je potřeba sledovat nebo identifikovat. Tagy obsahují menší objem dat.[16] RFID technologie byla vyvíjena na pokoření problému s čárovými kódy (nutnost viditelnosti kodu, malé datové úložiště, nelze změnit data po vytištění).[17]

3.1.1 Dělení RFID tagů

Zařízení RFID se dělí na následující:

1. Pasivní - neobsahují žádný zdroj elektrické energie. Integrovaný obvod je napájen pomocí příchozího signálu ze čtečky. Vzdálenost, ze které se dají data zapsat nebo načíst v reálných podmínkách, je od 10 cm do pár metrů (velmi záleží na zvolené frekvenci rádiového spojení a na vlastnostech antén).
2. Aktivní - na rozdíl od pasivních RFID zařízeních se v aktivních nachází vlastní zdroj elektrické energie, kdy díky němu může integrovaný obvod, který je obsažen v tagu, sám vysílat bez nutnosti napájení ze čtečky a dokáže komunikovat se čtečkou na větší vzdálenosti.[16]

Radio frequency identification se dělí na 4 frekvenční pásma:

1. Nízké frekvence - 125 - 135 kHz - Značení zvířat, imobilizéry v automobilech
2. Vysoké frekvence - 13,56 MHz - Sledování předmětů, vstupenky
3. Ultra vysoké frekvence - 433 MHz; 860 - 930 MHz - Označení kontejnerů, sledování nákladních vozidel
4. Mikrovlnné frekvence - 2,4 GHz; 5,8 GHz - Průmysl, přístupová práva[17]

3.2 NFC

NFC je speciální kategorie RFID na krátké vzdálenosti, která dovoluje obousměrnou komunikaci mezi zařízeními. Je využívána přenosová frekvence 13,56 MHz. NFC rozšiřuje normu ISO 14443.[16] Díky tomu je zpětně kompatibilní s některými RFID tagy (v pásmu vysokých frekvencí).

Zařízení podporující NFC je možnost využít jako bezkontaktní platební kartu, čipovou kartu ve veřejné dopravě nebo přístupovou kartu. Velkou výhodou zařízení NFC emulující klasickou RFID kartu je možno výměny karty, mít více karet v zařízení a aktualizace karty skrze internet.[18]

3.3 Klasifikace zařízení

Zařízení lze rozdělit do několika kategorií. Rozdělují se z pohledu na jejich napájecí schopnosti a na jejich roli v přenosu dat.

Aktivní/pasivní

V kategorii aktivní/pasivní se zařízení dělí dle jejich schopnosti napájení a poskytování energie:

1. Pasivní - Zařízení je nutno vystavit elektromagnetickému poli, aby bylo napájeno a mohlo začít reagovat na příkazy.
2. Aktivní - Zařízení samo o sobě vysílá elektromagnetické pole a může napájet cílové zařízení.[19]

Iniciátorské/cílové

Při přenosu dat vždy jedno zařízení musí začít přenos. Rozdělují se tedy do následujících kategorií:

1. Iniciátorské zařízení - jedná se o zařízení, které započne přenos dat.
2. Cílové zařízení - cílové zařízení potřebuje k započetí přenosu iniciátorské zařízení, bez něj komunikace nemůže začít.[19]

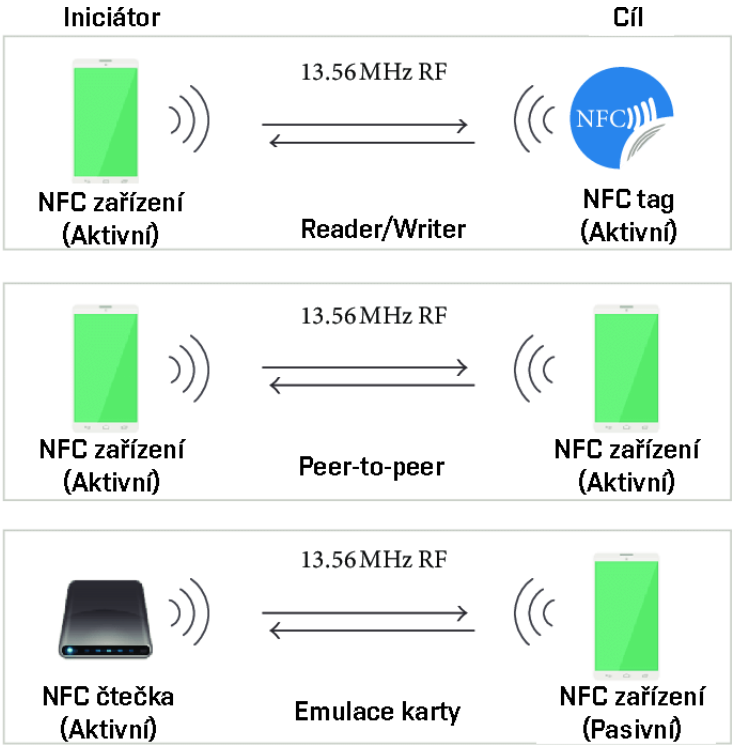
3.4 Režimy přenosu

Zařízení obsahující NFC službu může pracovat v několika režimech. Každý režim má svá speciální benefity pro uživatele.[16]

1. Reader/Writer - Tenhle režim operuje často mezi NFC zařízením a NFC kartou. Cílem zmíněného přenosu je čtení nebo zápis dat do tagu iniciátorským

zařízením. Přenos probíhá vždy jen v jednom módu. Zařízení do tagu jen zapisuje, nebo z něj jen čte.

2. Peer-to-peer - V režimu peer-to-peer dvě NFC zařízení komunikují mezi sebou a vyměňují si data. Zde probíhá přenos, kdy může jen jedno zařízení vysílat v jednu chvíli. Druhé zařízení může začít vysílat své data až vysílající zařízení ukončí svůj přenos.
3. Emulace karty - Tahle funkcionalita byla jeden cíl vzniku NFC. Zařízení může emulovat platební kartu. Velkou výhodou oproti fyzické kartě je možnost uchovávání více karet. [16]



Obr. 3.1: Režimy přenosu NFC[19]

3.5 Standardy tagů

NFC tagy se rozdělují do několika skupin. Jeden ze způsobů rozdělení dle odpovídajícího typu normy ISO 14443. [21]

3.5.1 O normě ISO/IEC 14443

Norma byla přejata do češtiny jako ČSN ISO/IEC 14443 - *Karty a bezpečnostní zařízení pro osobní identifikaci*. V normě jsou popsány požadavky na identifikační

karty, ale připouští i jinou formu zařízení než je jen karta.[20] Jsou zde specifikovány parametry dvou typů A a B.[21]

3.5.2 ISO/IEC 14443 Typ A

Typ A se využívá většinou na karty s menším datovým objemem. Využívá se zde upravené Millerovo kódování, kde pro změnu hodnoty z logické 1 na logickou 0 musí být změna signálu z 0 % na 100 %.[21] Je zde využívána ASK modulace. Integrovaný obvod je napájen z iniciátorského zařízení jen při komunikaci. Nejčastěji dosahuje rychlostí přenosu dat 106 kb/s.[27]

NTag2xx

Jedná se o rodinu NFC tagů od firmy NXP. Dělí se na 3 další produkty, které rozlišuje jen uživatelská paměť (144 B, 504 B, 888 B). Tagy mají za cíl levně a masově prosadit technologii NFC. Výrobce doporučuje tagy použít na párování bezdrátových technologií (Wi-Fi, Bluetooth) nebo identifikace skladových položek. Díky velké vstupní kapacitanci můžou mít celkové tagy malou anténu tedy i velikost. Celkovou integritu vnitřní paměti zajišťuje 16bitový CRC. [22]

MIFARE® Classic

Firma NXP vyvinula typ tagů MIFARE® Classic. Tagy jsou určené především pro aplikace ve veřejné dopravě. Velkou výhodou je možnost mít více tagů v elektromagnetickém poli a NFC zařízení si může zvolit, se kterým chce komunikovat. Zároveň se dbalo na nízký čas odezvy. Lze tak dosáhnout odbavení cestujícího za menší čas než 100 ms. Velikost paměti je 1 kB, která ovšem neslouží celá na uživatelské data. Každý 4 block v sektoru je určen k uzamčení sektoru a nastavení přístupových práv. Máme tedy výslednou velikost uživatelské paměti 752 B. Integritu paměti obstarává 16bitový CRC. [23]

MIFARE® Ultralight

Další tag od firmy NXP je MIFARE® Ultralight, který je především určen pro jednorázové lístky do veřejné dopravy, věrnostní karty, nebo denní vstupenky na události. Tagy MIFARE® Ultralight také nabízí možnost pracovat s více tagy v elektromagnetickém poli. Rychlost odbavení lístku je menší jak 35 ms. Tagy se vyrábí ve dvou variantách (48 B, nebo 128 B uživatelské paměti). Zde se taky nachází 16bitový CRC.[24]

ST25TN

Jediným zástupcem od firmy STMicroelectronics je tag ST25TN. Obsahuje paměť 208 B. Tag se dokáže permanentně zablokovat a zabezpečit, aby data, která na něm byla nahrána, nešla přečíst. Tuhle operaci nelze vzít zpět. Jako jediný druh tagů dokáže uchovat data minimálně 40 let.[25]

3.5.3 ISO/IEC 14443 Typ B

Typ B využívá manchester kódování, kde záleží na směru změny logické úrovně pro vyjádření logické hodnoty. Je zde použita ASK modulace, kde je hloubka modulace jen 10 %.[21] Integrovaný obvod je zde napájen po celou dobu přiblížení u iniciátorského zařízení.[27]

MIFARE® DESFire

NFC Tagy MIFARE® DESFire od firmy NXP nabízejí bezpečnostní funkce, které nabízejí i kreditní karty a elektronické pasy. Tagy podporuje, jak je již z názvu jasné, šifrování 3DES. Paměť o velikosti 32 kB může obsahovat až 32 různých souborů. Karta disponuje také rychlým přenosem dat (až 848 kb/s). Data na tagu lze uchovat na minimálně 25 let.[26]

4 Hardware

V téhle kapitole se zaměřuji na požadavky, se kterými stávající verze univerzální NFC čtečky vznikala. Dále byly vypsány chyby stávající verze, kvůli kterým vznikla nová verze HW NFC čtečky. Novou verzi NFC čtečky jsem popsal, jak z pohledu blokového diagramu, tak i dle skutečného rozmístění prvků na DPS. Na závěr kapitoly jsem vypsal možné vylepšení nové verze HW NFC čtečky, které jdou již z návrhu identifikovat, tak i ty, které byly objeveny při testování v kapitole 7.

4.1 Rozbor stávající verze NFC čtečky

Stávající verze HW, který by měl AAS interpreter využívat byla vytvořena jako součást bakalářská práce pana Jiráska v roce 2022. Autor Jirásek zde popisuje požadavky na vytvoření univerzální NFC čtečky. Mezi hlavní požadavky patří požadavek na rozměry, které by se měli vejít do rozměrů 100x100 milimetrů. Zároveň zde popisuje požadavek na PoE napájení. Všechny tyhle požadavky splňuje i nová verze. Poslední požadavek, který je specifikován v práci pana Jiráska je komunikace přes Ethernet pomocí protokolu Modbus TCP[1].[34]

Z důvodu nasazení na řešení I4.0 je v naší verzi nutnost využít protokolu OPC UA. Tohle ovšem není překážka, protože se jedná jen o změnu softwarové implementace.

4.1.1 Blokový diagram

Zařízení bylo koncipováno dle předem určených požadavků. Blokové schéma (obrázek 4.1) znázorňuje nejdůležitější součásti zařízení, které nyní popíšeme.

Jako centrální prvek řídící všechny ostatní součásti zařízení a zajišťující provoz programu byl autorem vybrán mikroprocesor *ESP32-WROOM-32D*.[34]

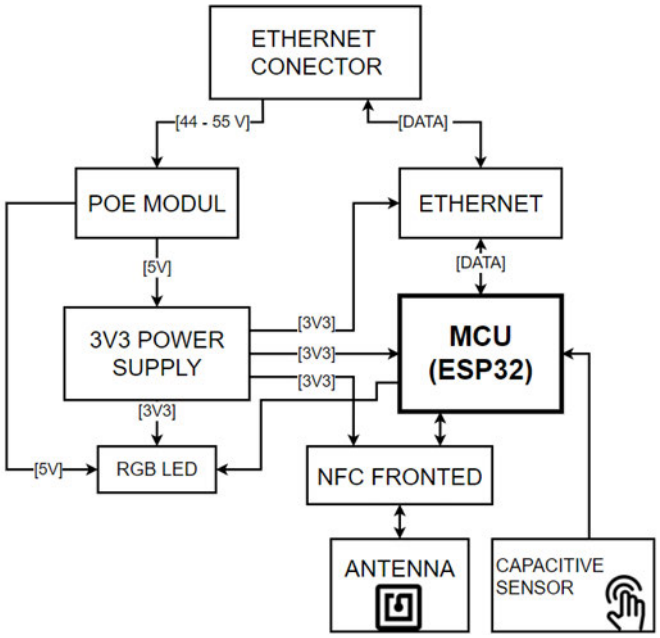
Napájení je zajišťováno pomocí modulu PoE, který převádí vstupní napětí z ethernetového konektoru(44 V až 55 V) na napětí potřebné pro další operace. Hlavní funkci PoE modulu zastává čip TX4139, fungující jako DC měnič, s nastaveným maximálním výstupním proudem 2 A. Další část napájení je řízena regulátorem AMS1117, který konvertuje napětí 5 V na 3.3 V. Přítomnost napájení je indikována LED diodou D2.[34]

Ethernet je taktéž použit jako komunikační kanál. Pro manipulaci s daty je využit integrovaný obvod *W5500*, který je připojen prostřednictvím sběrnice SPI do mikroprocesoru ESP32. Ethernetový kontrolér je schopen komunikovat se rychlostí až 100 Mb/s.[34] Jako konektor je zde zvolen standardní konektor RJ45.

[1]Modbus TCP - Jedná se o možnost využít protokol TCP/IP pro přenos protokolu Modbus.[35]

Pro práci s NFC zařízeními je zde využit čip MFRC522. Jedná se o čip, který je schopen komunikovat s zařízeními standardu ISO/IEC 14443. Zároveň byla navržena anténa jako součást DPS.[34]

Signalizace o stavu programu zde probíhá pomocí adresovatelných LED typu WS2812B. O indikaci přítomnosti sklenice se stará kapacitní senzor.[34]



Obr. 4.1: Blokový diagram HW univerzální NFC čtečky[34]

4.1.2 Problémy návrhu

Jako hlavní problém, na který se při testování původní verze narazilo, byl nízký dosah NFC. Tenhle neduh by se při návrhu AAS interpreteru projevoval.

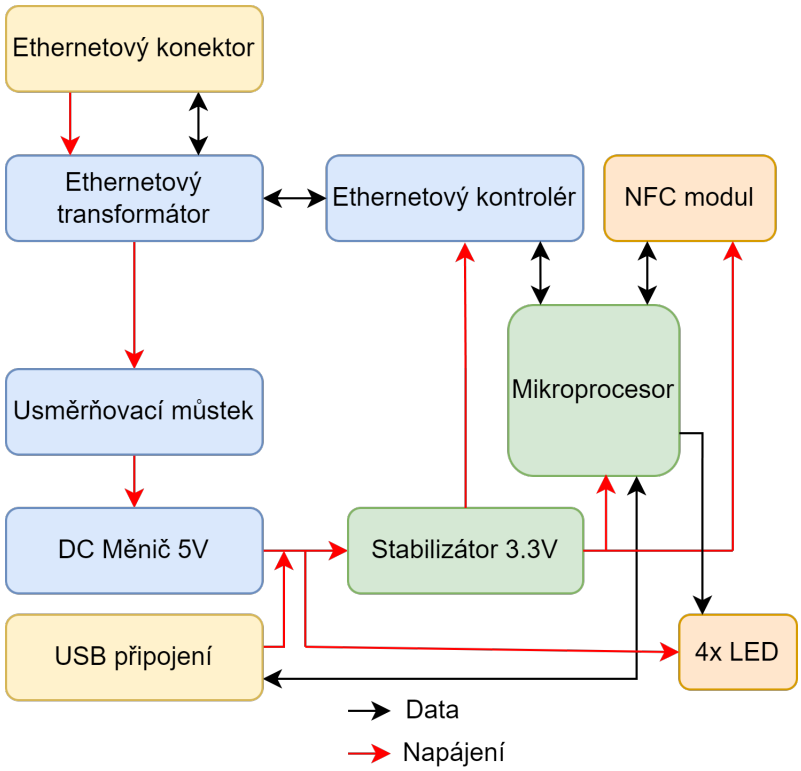
Jako další problém byl již autorem shledán tranzistor, který slouží k proudovému zesílení datového výstupu ESP32 do adresovatelných LED. Zde bylo nutné tranzistor vyřadit a adresovatelné LED napojit přímo na adresovatelný výstup mikroprocesoru.

4.2 Nová verze NFC čtečky

Protože předchozí verze HW disponovala NFC anténou s možností pracovat s NFC zařízením jen na velmi krátkou vzdálenost, bylo nutné navrhnout novou verzi, kde byl obvod pro práci s NFC ze základní desky odstraněn a nahrazen už existujícím modulem. Zároveň byla rozšířena možnost signalizace stavů pomocí adresovatelných LED. Rozměry DPS zůstaly stejné. Návrh a realizaci provedl můj vedoucí práce, Ondřej Baštán.

4.2.1 Blokový diagram

Nové verze HW obsahuje některé změny, proto jsem vytvořil nový blokový diagram, který znázorňuje funkční bloky a lze díky tomu verze HW mezi sebou porovnat.



Obr. 4.2: Blokový diagram nové verze HW NFC čtečky

Jedna z hlavních změn proběhla u mikroprocesoru. Byl změněn původní na *ESP32-S3-WROOM-1*, který disponuje už přímo USB převodníkem jak na JTAG, tak i na UART, byl tedy přidán konektor pro připojení USB. Lze i přes tenhle konektor celé zařízení napájet. Ladění SW přes JTAG je popsáno v kapitole 6.1.2.[38]

Další podstatná změna proběhla ve výměně NFC modulu na DPS za již hotový NFC modul. Jedná se o modul s označením PN532, který jsme otestovali a zjistili, že pro naši aplikaci bude ideální. Název modulu je odvozen od použitého čipu PN532 od firmy NXP. Vzdálenost čtení dle popisu je 5 cm až 7 cm, což je dostatečné v našem případě.[29] Dále modul podporuje připojení skrze sběrnici I^2C, SPI a HSU.[30] Pro připojení NFC modulu byla zvolena sběrnice SPI. Tento výběr byl motivován specifickými napěťovými požadavky na vstupní/výstupní piny (GPIO). Mikroprocesor ESP32 S3 poskytuje výstupní napětí 3.3 V, což je napětí, které je kompatibilní se sběrnicí SPI.[28]

Počet signalizačních LED byl změněn na 4. Lze tedy signalizovat více stavů zařízení. Stále se jedná o adresovatelné LED typu WS2812B.

Zbytek HW je stejný, jako předchozí verze, popsána v kapitole 4.1.1, kde jsou zbývající části diagramu popsány.

4.2.2 Rozmístění prvků

V následující kapitole popíšeme pozice prvků rozmístěných na skutečném HW. DPS je v provedení dvouvrstvé desky, kdy na spodní straně (obrázek 4.3) jsou umístěny všechny součástky a na horní straně (obrázek 4.4) se nachází signalizační LED a NFC modul. Rozměry DPS jsou 80x80 milimetrů.

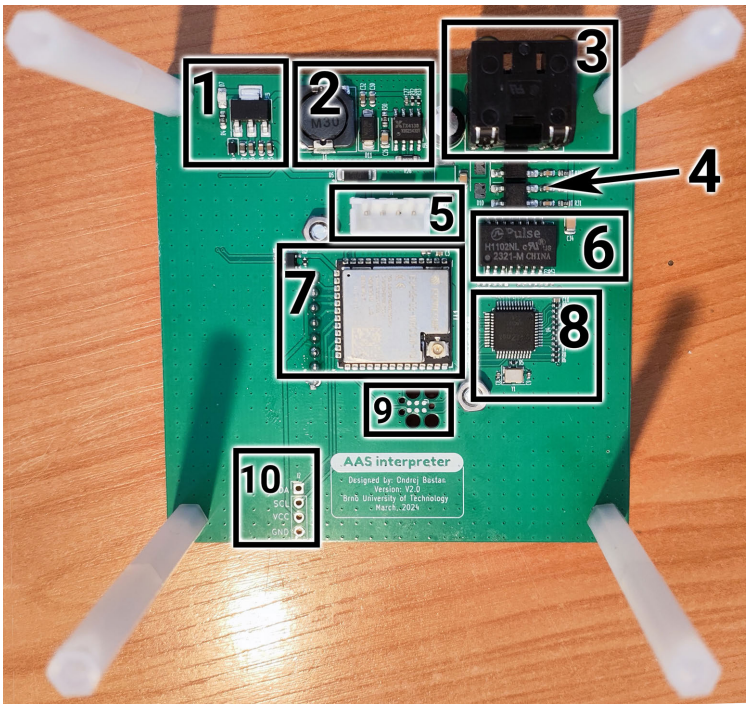
Jako první rozepíši popis částí DPS, které se nacházejí na spodní straně (obrázek 4.3):

1. Stabilizátor 3.3 V - Jedná se o stabilizátor napětí AMS1117 a přidružené součástky pro mikroprocesor ESP32, ethernetový modul *W5500* a NFC modul.
2. DC Měnič 5 V - Součást napájení PoE , která pomocí čipu TX4139 a dalších součástek slouží k převodu vstupního napětí PoE na 5 V, které je dále upravováno.
3. Ethernetový konektor - Jako připojovací konektor ethernetu je zde využit standardní konektor RJ45. jedná se konkrétně o konektor *RJHSE-5081* zapojen v režimu 10/100 Base-T. Na konektoru se nachází dvě LED. Jedna slouží k indikaci Link statusu[2] a druhá k indikaci Active statusu[3]. [36]
4. Usměrňovací můstek - Pro správnou polaritu napětí přivedenou na PoE DC měnič 5V (bod 2) je využit pár usměrňovacích můstků *MB10S*. Na vstup prvního usměrňovacího můstku je přivedeno PoE napětí při zapojení v IEEE 802.3af režim A, kdy jsou využity střední vývody ethernetového transformátoru k napájení, tedy oba páry datových vodičů obsahují i napájecí napětí. Druhý slouží k usměrnění napětí v zapojení IEEE 802.3af režim B, kdy jsou využity přímo 4 volné vodiče při zapojení 10/100 Base-T.[33]
5. USB konektor - Pro připojení ladícího USB byl do návrhu zahrnut konektor *JST XH B4B*.
6. Ethernetový transformátor - Aby byl ethernetový obvod *W5500* galvanicky oddělen od přenosu dat, je potřeba oddělovací ethernetový transformátor. V našem případě se na DPS nachází transformátor *H1102NLT* od firmy Pulse.
7. Mikroprocesor - Jak již bylo v předchozí kapitole 4.2.1, změna proběhla v mikroprocesoru. Nyní je využit mikroprocesor *ESP32-S3-WROOM-1*, který zjednodušil návrh DPS a následné ladění SW.

[2]Link status - Aktivní při správném zapojení ethernetového kabelu na obou jeho stranách.[36]

[3]Active status - Aktivní při správném nastavení přenosu dat a zároveň znázorňuje aktivitu jak odesílání dat, tak i přijímaní dat.[36]

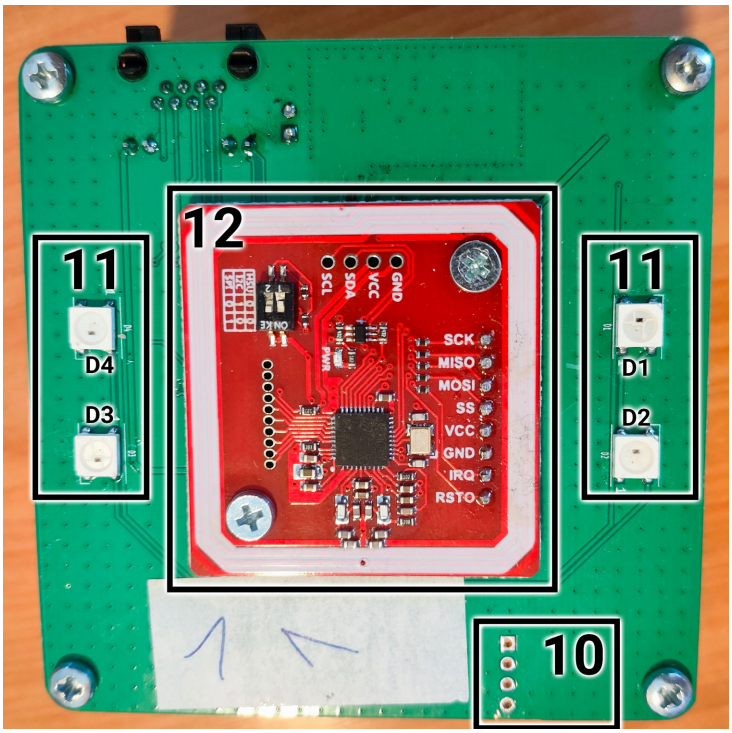
8. Ethernetový kontrolér - Pro práci s ethernetem byl stejně jak v předchozí verzi HW využit čip *W5500* od firmy WIZnet.
9. Programovací port - Pokud bychom chtěli mikroprocesor programovat jinou možností než přímo skrze USB, lze využít kabel s konektorem typu *TC2030-IDC*. Kabel se pomocí zámku uchytí na DPS a konektor se spojí s kabelem.
10. I^2C konektor - Do budoucna se počítá s možnosti připojit externí modul pomocí sběrnice I^2C do NFC čtečky. Aby bylo připojení modulu co nejjednodušší, jsou zde vyvedeny vodiče pro připojení propojovacího kabelu.



Obr. 4.3: Spodní strana HW NFC čtečky

V následující části popíši horní stranu DPS (obrázek 4.4):

11. Adresovatelné RGB LED - Pro získání stavu, ve kterém se stavový automat nachází a jaká operace probíhá, nám slouží 4 adresovatelné RGB LED typu *WS2812B*. Díky jejich adresaci a možnosti sériového spojení samotných LED můžeme využít jen jeden GPIO výstup. Jejich pořadí je vyznačeno na obrázku 4.4.
12. NFC čtečka - Pro získání informací z NFC tagu byl využit modul NFC čtečky *PN532*. Jedná se o samostatný modul pro práci s NFC zařízeními. Modul je uchycen k základní DPS pomocí 2 šroubů a 4 matic, kdy jeden pár je využit jako distanční oddělení obou desek a druhý pár je využit pro přitažení šroubů.



Obr. 4.4: Horní strana HW NFC čtečky

Zatím není navržen vnější obal, ve kterém by NFC čtečka byla. Bylo tedy nutné pro možnost testování přidat distanční šrouby, které nám dovolí připojení kabelu pro USB. Zároveň má zařízení mnohem větší stabilitu.

4.2.3 Možné vylepšení

Hned z návrhu nové verze vyplývá jedno možné vylepšení, které by posunulo bezpečnost zařízení a celkového systému Testbedu I4.0 na vyšší úroveň. Jedná se o možnost využít aktivní PoE. Jedná se o možnost napájení skrze ethernet dle normy IEEE 802.3. Nasazení PoE napájení, které je využito na HW PoE čtečce, je provedeno jako pasivní. Rozdíl mezi aktivním a pasivním PoE je ve způsobu komunikace mezi napájecím zařízením (PSE) a napájeným zařízením (PD). U pasivní PoE neprobíhá žádná komunikace a PSE dodává napájení přímo. Navíc pasivní PoE není žádným způsobem standardizováno. Oproti tomu u aktivní PoE probíhá komunikace mezi PSE a PD, kdy jako první PSE je ve fázi detekce zařízení a až poté PSE dodá napájecí napětí PD.[31] [32]

Jako druhou chybu, kterou jsem objevil a přede mnou i autor Jirásek, je tranzistor pro proudové zesílení signálu pro adresovatelné RGB LED. Zde jsem využil stejné opravy jako autor Jirásek. Bylo potřeba odstranit tranzistor Q1 a propojit výstup z mikroprocesoru přímo na vstup dat do adresovatelné RGB LED.

5 Navržený model AAS

První musíme stanovit požadavky na AAS interpreter, které by měl splňovat. Následně jsem definoval datovou strukturu, která bude uložena na NFC tagu. Tahle datová struktura bude interpretována.

Celkový pohled na způsob interpretace je popsán v následující podkapitole z pohledu obecného stavového digramu a následně jsou jednotlivé stavy rozepsány do vývojových diagramů.

V poslední podkapitole definuji metody, které budou v rámci modelu AAS využívány při komunikaci mezi AAS interpreterem a OPC UA serverem.

5.1 Určení požadavků

Abychom mohli začít navrhovat model AAS je potřeba si uvědomit všechny požadavky, které jsou od AAS interpreteru očekávány. Jednu sadu požadavků nám určila kapitola 1.1, která se zaobírá hlavními principy I4.0. Pokud bychom je měli specifikovat konkrétně na naši verzi AAS interpreteru zněly by následovně:

1. Interoperabilita - AAS interpreter musí umožňovat začlenit do komunikačního procesu i výrobky.
2. Virtualizace - AAS interpreter v případě obsazení výrobkem musí virtualizovat jeho instanci.
3. Decentralizace - Díky AAS interpreteru je řízení prováděno přímo výrobkem v místě, kde se výrobek nachází.
4. Operace v reálném čase - Výrobek musí skrze AAS interpreter prosazovat své zájmy v reálném čase.
5. Orientace na služby - AAS interpreter musí umožnit výrobku poptávat služby od autonomních buněk.
6. Modularita - AAS interpreter musí být generické zařízení, které musí umožnit propojit jednotlivé účastníky standardním způsobem a tím zajistit modularity celého systému.

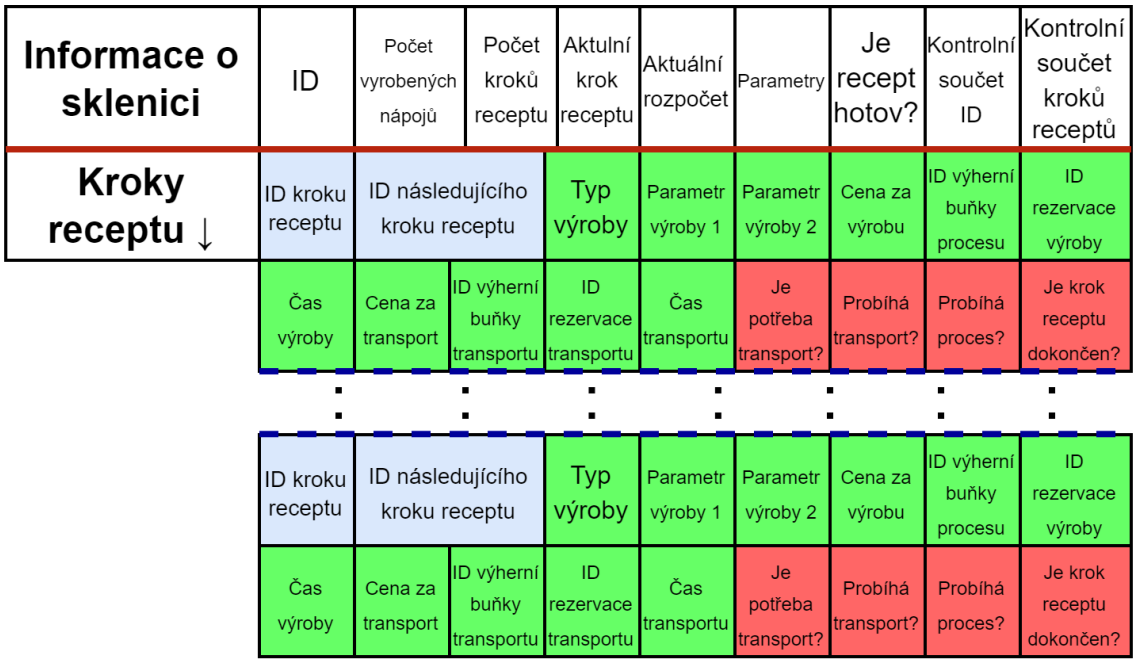
Je taky nezbytné splnit požadavky, které plynou po provedení rešerše NFC tagů (kapitola 3.5), a to je nutnost minimalizovat počet zápisů na NFC tag z důvodu omezené životnosti přepisovatelné paměti a nutnost minimalizovat velikost uložených dat. Jako ideální NFC tag se nám jeví z důvodu dostupnosti a parametrů MIFARE® Classic 1k. Zde nám jeho velikost přepisovatelné paměti bude vystačovat. Ovšem je potřeba myslet na využití jiného typu NFC tagu a vytvořit knihovnu co nejvíce univerzální. Abychom šetřili počet zápisů do NFC tagu je potřeba při návrhu algoritmu stavového automatu interpretace minimalizovat počet zápisu. K tomu nám

pomůže knihovna, která se bude starat o udržení integrity vnitřní paměti. Ta zajišťuje správné načtení dat do paměti mikroprocesoru a zároveň vnitřní mezipaměť pro data, která se poté zapíší v jednom kroku. Tím bude ušetřený počet zápisů.

Z důvodu zvoleného způsobu uložení dat (data budou uložena na NFC tagu, který je na sklenici, a ta se nenachází stále na AAS interpreteru) je vhodné zvolit metodu komunikace přes protokol OPC UA ve formě volání metod. Z toho nám plyne, že využijeme architekturu server-klient, kdy server bude procesní buňka a klient bude AAS interpreter. Zvolené řešení je unikátní tím, že dovolí přenést výpočetní část na procesní buňky, kdy každá může mít využitý jiný algoritmus odpovědí na zavolané metody.

5.2 Datová struktura úložiště NFC

Vzhledem k omezené kapacitě úložiště dat je nezbytné vytvořit datovou strukturu, která bude efektivně uložena na NFC tagu. Tento přístup nám omezuje možnost sledování detailních statistik o provedených operacích. Nicméně, v budoucnosti můžeme zvážit implementaci dalšího typu buňky, která by se zaměřovala na analýzu pomocí volání metod protokolu OPC UA AAS interpreterem, což by nám umožnilo získat podrobnější statistiky.



Obr. 5.1: Datová struktura ve vnitřní paměti NFC tagu

Na obrázku 5.1 lze vidět rozdělení paměti do dvou kategorií:

1. **Informace o sklenici** - V úvodní části paměti jsou uloženy data, která se vztahují ke sklenici a celému receptu. Jako první vlastnost, která je zde uložena, je identifikační číslo sklenice. Dále se nachází jako jediný statistický údaj a to počet vyrobených nápojů. Tento údaj jsem zde integroval z důvodu jednoduché analýzy při testování.
   Následují údaje, které se týkají receptu. Jako první je zde údaj o počtu kroků receptu. Tento údaj je pro nás důležitý, abychom věděli, jak velkou paměť NFC tagu musíme načíst, abychom měli recept kompletní. Následuje informace o aktuálním kroku. Rozpočet, se kterým sklenice může ještě disponovat, je na následujícím místě. Pokud by bylo potřeba u sklenice určit nějaký parametr (jiný objem, atd.), využijeme k tomu následující proměnnou. Dále se nachází v paměti příznak, že je recept hotový.
   Jako poslední část informací o sklenici se nachází kontrolní součet k ID sklenice, který musí vždy sedět v rovnici:

$$\mathrm{ID} + \mathrm{ID}_{Kontrolni\ soucet} = 0 \tag{5.1}$$

   Využívá se zde skutečnost přetečení datové typu. V případě, že tomu tak není, můžeme recept považovat za neplatný, a AAS interpreter se podle toho zachová. Zároveň se zde nachází kontrolní součet kroků receptu, který nám slouží k identifikaci změny uložených dat do kroků receptů. Pokud je kontrolní součet rozdílný od načtených dat v mikroprocesoru můžeme tvrdit, že data se změnila a jsou potřeba znovu načíst.
2. **Kroky receptu** - Jedná se o pole struktur kroků receptu. Můžeme jej rozdělit ještě do podkategorií:
   (a) **Informace o pohybu v krocích receptu** - Je zde určeno identifikační číslo receptu, které odpovídá obvykle pozici v poli. Jako druhá proměnná je tu ukazatel na následující krok receptu. Tenhle způsob lineárního seznamu[1] je skvělý, pokud víme, že budeme přidávat nebo ubírat prvky v seznamu. Při přidávání nového kroku do receptu je nejprve nutné zvětšit alokovanou paměť v mikroprocesoru, která je určena pro ukládání kroků receptu. Toto zvětšení paměti nám umožní přidat nový krok na konec seznamu kroků receptu. Nově přidaný krok receptu pak musí obsahovat identifikátor (ID), který odkazuje na následující krok v seznamu. Toto ID by mělo být stejné jako ID, které bylo původně nastaveno jako následující krok receptu u kroku, za který chceme nový krok přidat. Nakonec je

[1] Lineární seznam je dynamická datová struktura, která není provázána svým pořadím, ale ukazateli na následující prvek. [37]

třeba aktualizovat krok receptu, za který jsme nový krok přidali. U tohoto kroku změníme ID následujícího kroku na ID nově přidaného kroku. Tímto způsobem se nový krok stane součástí lineárního seznamu kroků receptu a bude správně odkazovat na další krok v seznamu.

Aby mikroprocesor poznal, kdy nastal konec receptu, nastavíme hodnotu ID následující kroku na totožný krok. Tedy za předpokladu, že se ID aktuálního kroku a ID následujícího kroku rovnají, mikroprocesor vyhodnotí recept jako hotový.

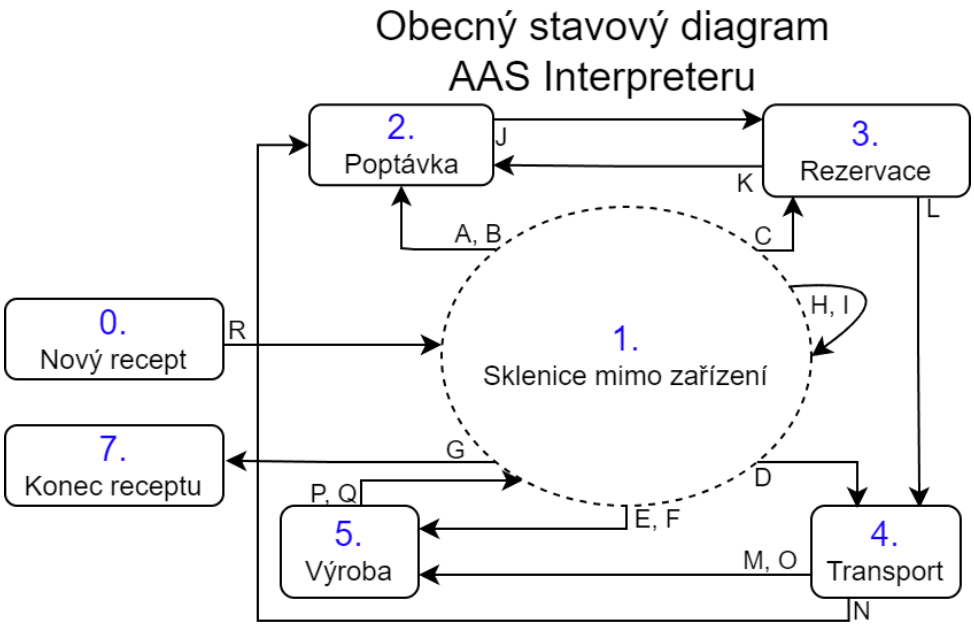
(b) **Informace o procesu výroby a transportu** - Velká část paměti kroku receptu je vyhrazena informacím o procesu výroby a transportu, jsou zde uloženy také informace o typu výroby a její parametrech. Zároveň s tím se zde ukládají parametry domluvených procesů, jako jsou ID buněk, které nabídli sklenici nejlepší parametry potřebných operací.

(c) **Příznaky kroku receptu** - Závěrečná část kroku receptu je určena příznakům, dle kterých se stavový automat rozhoduje, do kterého stavu má přejít po objevení sklenice na AAS interpreteru.

Navržená datová struktura nám poskytne možnost efektivně pracovat s daty a ukládat aktuální stav kroku receptu. V případě ztráty spojení mezi sklenicí a interpretem AAS jsme schopni určit stav, do kterého přejde stavový automat.

5.3 Způsob interpretace

Celý proces intepretace dat je proveden pomocí konečného stavového automatu. Celý proces interpretace je rozdělen do obecných stavů znázorněných na obrázku 5.2. Přechody mezi stavy jsou popsány velkými písmeny, které jsou znázorněny v jednotlivých vývojových digramech částí obecných stavů.



Obr. 5.2: Obecný stavový diagram AAS interpreteru pro interpretaci dat

5.3.1 Popis obecných stavů

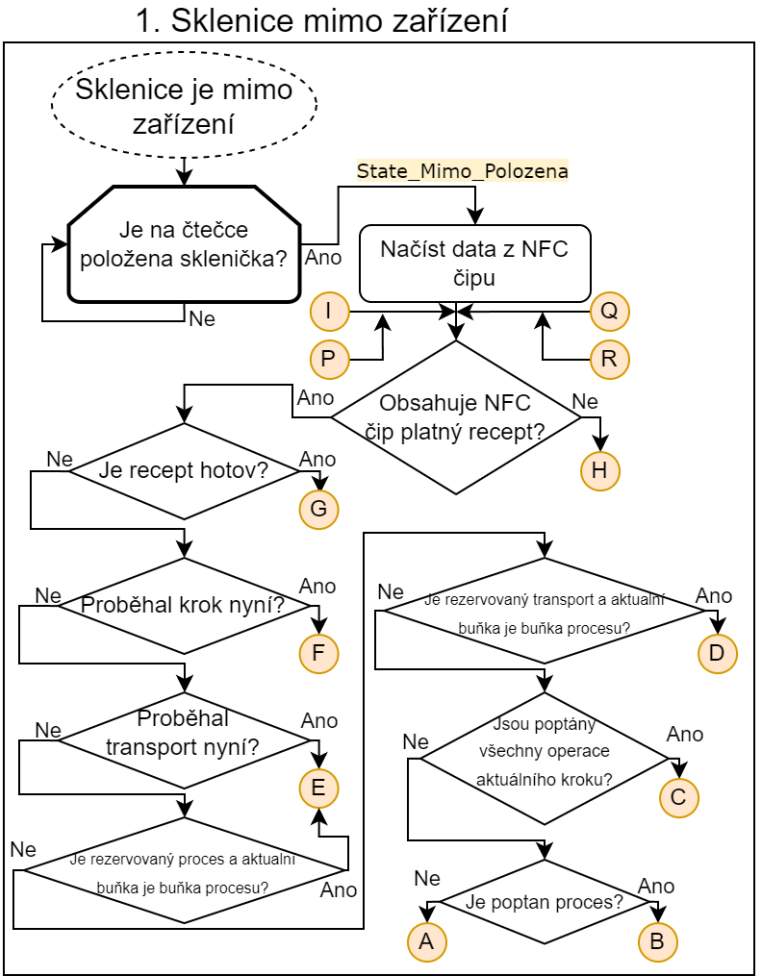
V následujícím textu popíši jednotlivé stavy a znázorním vývojovými diagramy. Text s oranžovým pozadím na vývojových diagramech znázorňuje stav interpretace. Některé stavové diagramy jsou rozděleny na více stavů pro lepší orientaci a jednoduší interpretaci.

0. Nový recept

Jedná se o stav, do kterého se stavový automat dostane při nahrávání nového receptu. Je zde upravena paměť *Informace o sklenici*, kdy je nutnost zachovat některé informace (ID, počet vyrobených nápojů) a ostatní přepsat dle aktuálního receptu. Kroky receptu jsou celé přepsány dle aktuálního receptu.

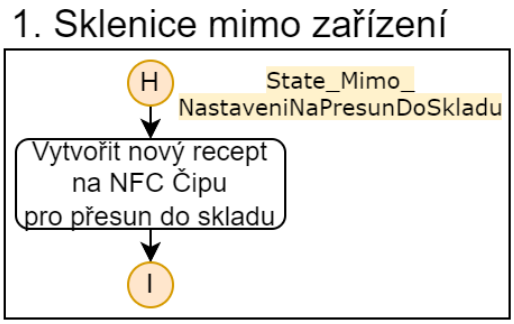
1. Sklenice mimo zařízení

Celou vstupní část stavového automatu zajišťuje právě tenhle blok znázorněný na obrázku 5.3.



Obr. 5.3: Stav interpretace - 1. Sklenice mimo zařízení

Po objevení sklenice na AAS interpreteru jsou načteny data. Dále je určeno, jestli načtený recept je platný. Pokud ano, je určen stav, do kterého se stavový automat má dostat. Zároveň součástí zmíněného obecného stavu je stav *State_Mimo_NastaveniNaPresunDoSkladu*, kdy se na AAS interpreteru objeví NFC tag, který neobsahuje platný recept. Stav je zobrazen na obrázku 5.4. V tomhle případě je nutné vytvořit nový recept *NavratDoSkladu*, který slouží pouze k transportu sklenice do skladu.



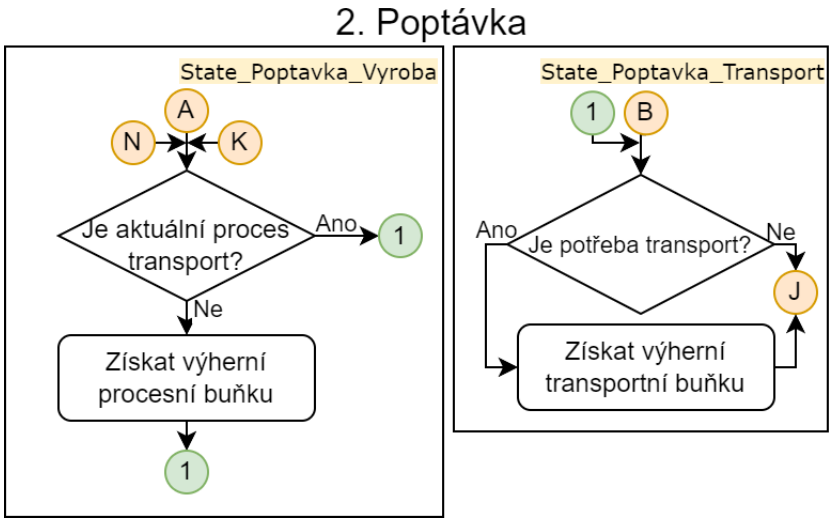
Obr. 5.4: Stav interpretace - 1. Sklenice mimo zařízení - Neplatný recept

2. Poptávka

První, co je potřeba zajistit, je poptání buněk, které provedou výrobní proces a případný transport do výherní výrobní buňky. Na obrázku 5.5 je vývojový diagram znázorňující popisovaný obecný stav. Pro přehlednost byl stav rozdělen do dvou stavů, kdy v jednom stavu je poptán výrobní proces a v druhém je poptán transport.

Od procesních buněk, které nabízejí druh výrobního procesu, který je v aktuálním kroku receptu, jsou poptány parametry výrobního procesu (cena a čas, kdy může nastat výrobní proces). Algoritmus vybere dle vnitřních požadavků na recept výherní buňku. V naší implementaci bereme fakt, že požadavek na recept žádný není, tedy je volena nejlevnější výrobní buňka. Může být ovšem požadavek na rychlost výroby. V tu chvíli je potřeba volit výrobní buňky, které nabídnou výrobní proces co nejdříve. Obdobně je poptávka provedena u transportu.

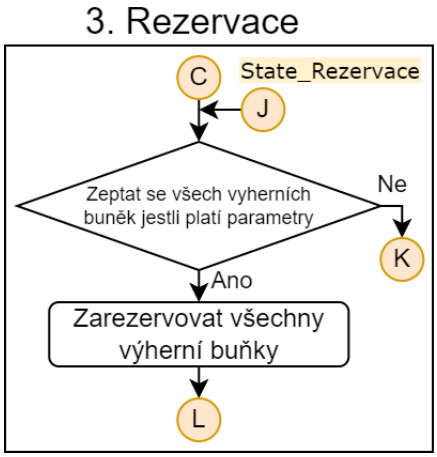
Může nastat situace, kdy je jako výrobní proces zvolen transport (je potřeba provést pouze transport do určité buňky), tedy je poptána jen transportní buňka. Zároveň může nastat situace, kdy není potřeba transport (aktuální buňka je výherní procesní buňka).



Obr. 5.5: Stav interpretace - 2. Poptávka

3. Rezervace

Po objevení v obecném stavu *3. Rezervace*, je jako první ověřeno od výherních buněk, jestli platí parametry, které byly poptány. Parametry můžou přestat být aktuální, protože některý jiný AAS interpreter poptal proces prioritně. V tom případě musí opět proběhnout poptávka. Po kladné odpovědi od všech výherních buněk zarezervuje AAS interpreter poptané procesy.



Obr. 5.6: Stav interpretace - 3. Rezervace

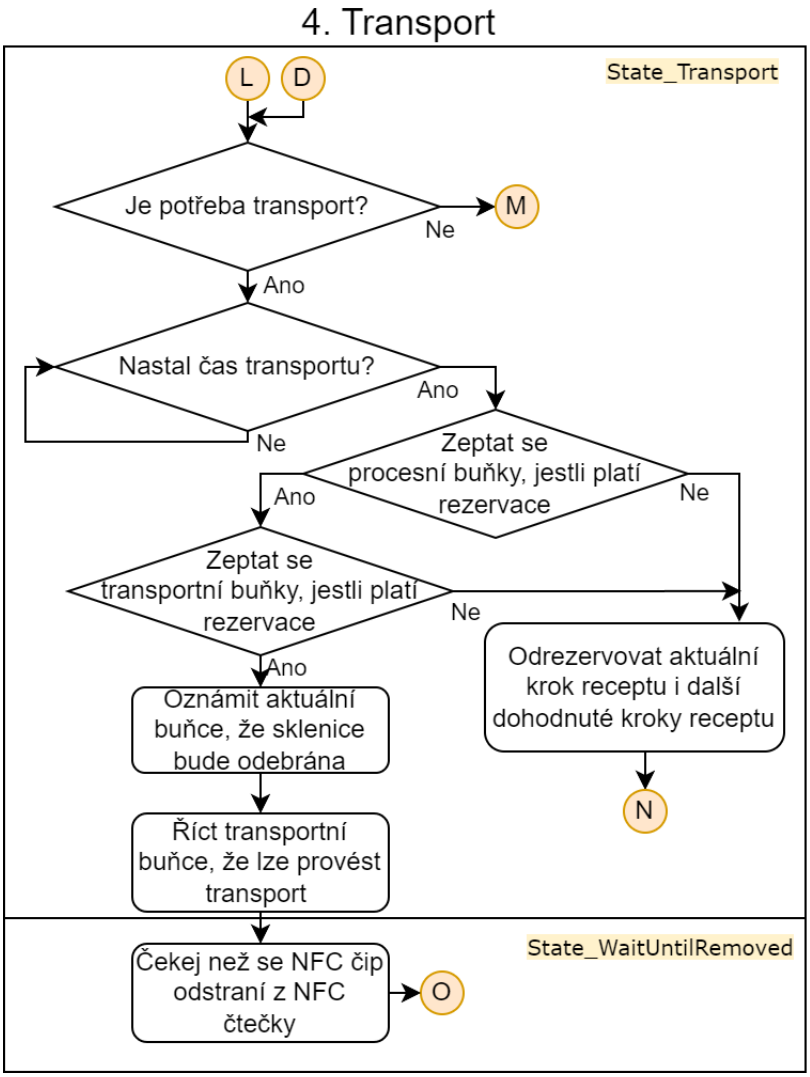
4. Transport

Procesní úkon transport je prováděn v tomhle kroku znázorněném ve vývojovém diagramu na obrázku 5.7. První je potřeba ověřit, jestli je v aktuálním kroku receptu

potřeba transport. Po ověření a kladné odpovědi se ověřuje jestli nastal čas transportu. V čase transportu je ověřeno jestli platí předem domluvená rezervace. Tahle situace může nastat, pokud někdo zažádal o proces prioritně, nebo pokud je buňka obsazena. Pokud neplatí, je potřeba odregistrovat všechny zaregistrované operace a opět provést poptávku.

Pokud všechny rezervace platí, oznámí se aktuální buňce, že sklenice bude odebrána a bude volná. Poté je oznámeno transportní buňce, že může proběhnout zarezervovaný transportní proces.

Nyní AAS interpreter čeká v následujícím stavu *State_WaitUntilRemoved*, dokud nebude sklenice odebrána z NFC čtečky.



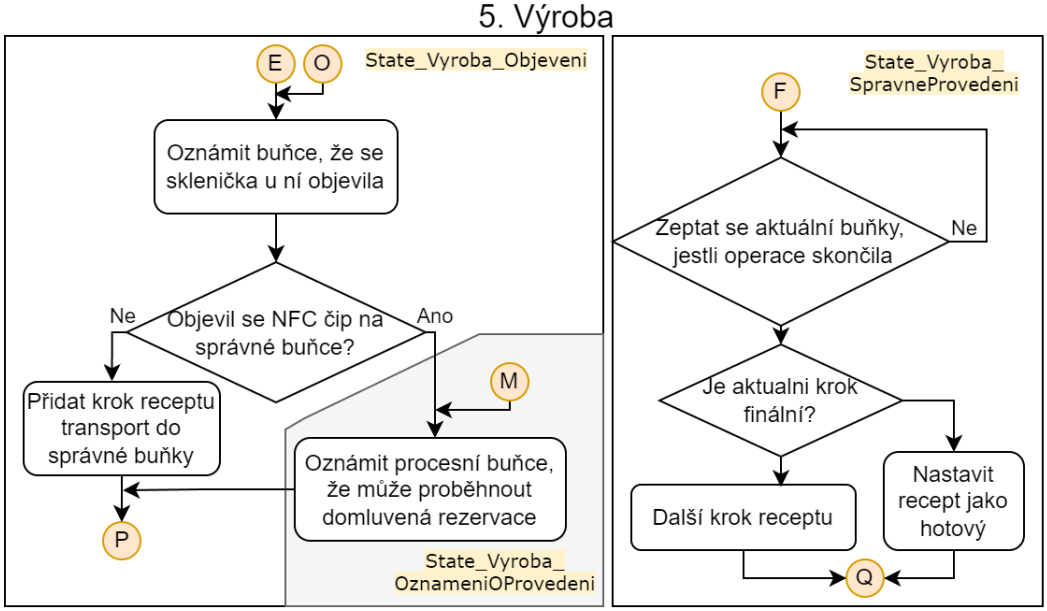
Obr. 5.7: Stav interpretace - 4. Transport

5. Výroba

Výrobní procesní operace proběhne v kroku *5. Výroba* zobrazeném na obrázku 5.8. Stav je rozdělen do 3 dílčích stavů.

První se oznámí aktuální buňce, že je obsazena a následně se musí ověřit, jestli jsme se ověřili u správné buňky. Pokud ne, je potřeba přidat transportní krok receptu do správné buňky. Pokud se sklenice nachází ve správné výrobní buňce, oznámí se buňce, že může proběhnout rezervace. Zde přecházíme do obecného stavu *1. Sklenice mimo zařízení*, z důvodu nejistoty, jestli bude sklenice z NFC čtečky odebrána.

Po objevení sklenice na NFC čtečce je ověřováno od procesní buňky, jestli již výrobní proces skončil. Po skončení výrobního procesu je ověřeno jestli je aktuální krok receptu finální (ověřuje se, jestli krok receptu odkazuje sám na sebe). Pokud není aktuální krok receptu finální, je tento krok označen jako dokončený a aktuální krok receptu se přesune na následující krok, na který se odkazuje nynější krok receptu. V opačném případě je krok receptu také označen jako dokončený a následně je celý recept označen jako dokončený.



Obr. 5.8: Stav interpretace - 5. Výroba

6. Konec receptu

Jedná se o finální stav, do kterého stavový automat přejde po dokončení receptu. Zde bude záležet na finální implementaci, co se bude dít. V testovací fázi algoritmu se nastaví recept na přesun do skladu, podobně jako je zobrazeno na obrázku 5.4.

5.3.2 Možné rozšíření interpretace

Další funkcionalita, kterou by interpretace mohla získat, je možnost přesunout sklenici do bufferu[2]. Tohle vylepšení by dokázalo odstranit blokování buněk sklenicí, která čeká na další výrobní proces. Nyní v testbedu I4.0 přímo takový buffer neexistuje, tedy v návrhu využít tuhle možnost se nepočítalo.

5.4 Navržené metody OPC UA

Aby mohla probíhat komunikace mezi OPC UA klientem a OPC UA serverem je potřeba navrhnout sadu metod. V tabulkách jsou popsány vlastnosti metody. Všechny metody jsou nastaveny jako *NodeId* v datovém typu *Textový řetězec - String* (popsáno v kapitole 2.4.1). Dále jsou vypsány vstupní a výstupní argumenty metody. U argumentů je popsáno jestli se jedná o Vstupní/Výstupní argument, ID argumentu, název argumentu, datový typ a rozměr argumentu.

Inquire - Poptávka procesu

Aby se oslovila buňka na parametry procesu slouží tahle metoda. Vlastnosti metody jsou popsány v tabulce 5.1. Metoda má 5 vstupních argumentů, kdy význam argumentu *Parameter1* a *Parameter2* slouží ke sdělení vlastností procesu. Výstupní argumenty slouží k získání parametrů procesu. *IDOfReservation* slouží jako identifikátor poptávky, a poté i rezervace.

Zmíněná metoda se využívá ve stavu *2. Poptávka*, který je popsán na obrázku 5.5.

[2]Buffer - Mezisklad určený pro dočasné uskladnění sklenice s obsahem.

Tab. 5.1: OPC UA metoda pro poptávku - Inquire

NodeId(String)	Inquire	**Zobrazované jméno**	Inquire	
Druh argumentu	**ID**	**Název**	**Datový typ**	**Rozměr**
Vstupní	0	InterpreterID	UInt16	Skalár
Vstupní	1	TypeOfProcess	Byte	Skalár
Vstupní	2	Priority	Boolean	Skalár
Vstupní	3	Parameter1	Byte	Skalár
Vstupní	4	Parameter2	Uint16	Skalár
Výstupní	0	IDOfReservation	Uint16	Skalár
Výstupní	1	TimeOfReservation	DateTime	Skalár
Výstupní	2	Price	Float	Skalár

IsValid - Ověření změny parametrů procesu

Následující metoda slouží k ověření, jestli se nezměnili poptané parametry procesu. V tabulce 5.2 jsou popsány argumenty metody. Jako vstupní argument se využívá pouze identifikátor rezervace, pod kterým jej má OPC UA uložen v paměti. Nachází se zde i 4 výstupní argumenty, které informují o případné změně parametrů a přímo příznakem, jestli proběhla změna.

Metoda je využívána ve stavech *3. Rezervace* a *4. Transport*.

Tab. 5.2: OPC UA metoda pro ověření změny parametrů procesu - IsValid

NodeId(String)	IsValid	**Zobrazované jméno**	IsValid	
Druh argumentu	**ID**	**Název**	**Datový typ**	**Rozměr**
Vstupní	0	IDOfReservation	UInt16	Skalár
Výstupní	0	IDOfReservation	Uint16	Skalár
Výstupní	1	Price	Float	Skalár
Výstupní	2	TimeOfReservation	DateTime	Skalár
Výstupní	3	Change	Boolean	Skalár

Rezervation - Provedení rezervace určité poptávky

Rezervace poptávky, která byla dříve poptána, probíhá pomocí téhle metody. Jako vstupní argument se využívá identifikátor rezervace. Návratové argumenty jsou pa-

rametry poptávky a příznak jestli se poptávka zarezervovala.

Metoda je využívána ve stavu *3. Rezervace*, který je zobrazen na obrázku 5.6.

Tab. 5.3: OPC UA metoda pro zarezervování poptávky - Rezervation

NodeId(String)	Rezervation	**Zobrazované jméno**	Rezervation	
Druh argumentu	**ID**	**Název**	**Datový typ**	**Rozměr**
Vstupní	0	IDOfReservation	UInt16	Skalár
Výstupní	0	IDOfReservation	Uint16	Skalár
Výstupní	1	Price	Float	Skalár
Výstupní	2	TimeOfReservation	DateTime	Skalár
Výstupní	3	Reserved	Boolean	Skalár

CancelRezervation - Zrušení domluvené rezervace

Pro zrušení domluvené rezervace, je využita tahle metoda. Je zde využit pouze jeden vstupní argument identifikátor rezervace. Návratový argument je zde pouze příznak, jestli byla rezervace zrušena.

Využita je ve stavu, kdy je potřeba zrušit zarezervované procesy. Jedná se o stav *4. Transport* zobrazený na obrázku 5.7.

Tab. 5.4: OPC UA metoda pro zrušení rezervace - CancelRezervation

NodeId(String)	CancelRezervation	**Zobrazované jméno**	CancelRezervation	
Druh argumentu	**ID**	**Název**	**Datový typ**	**Rozměr**
Vstupní	0	IDOfReservation	UInt16	Skalár
Výstupní	0	Canceled	Boolean	Skalár

DoProcess - Zahájení zarezervovaného procesu

Zahájení předem zarezervovaného procesu je provedeno touhle metodou. Je potřeba jeden vstupní argument, kterým je identifikátor rezervace. Jako výstupní argument je zde příznak, jestli proces započal.

Metoda je využita ve stavu *5. Výroba*. Tenhle stav je zobrazen na obrázku 5.8.

Tab. 5.5: OPC UA metoda pro zahájení procesu - DoProcess

NodeId(String)	DoProcess	**Zobrazované jméno**	DoProcess	
Druh argumentu	**ID**	**Název**	**Datový typ**	**Rozměr**
Vstupní	0	IDOfReservation	UInt16	Skalár
Výstupní	0	Started	Boolean	Skalár

IsFinished - Dotaz na dokončení operace

Aby AAS interpreter věděl, jestli již skončil zahájený proces, je využíváno téhle metody. Obsahuje jeden vstupní argument, kterým je identifikátor rezervace. Výstupní argument je příznak, jestli již proces byl dokončen.

Je využita ve stavu *5. Výroba*, který je zobrazen na obrázku 5.8.

Tab. 5.6: OPC UA metoda dotazování na dokončení zahájené operace - IsFinished

NodeId(String)	IsFinished	**Zobrazované jméno**	IsFinished	
Druh argumentu	**ID**	**Název**	**Datový typ**	**Rozměr**
Vstupní	0	IDOfReservation	UInt16	Skalár
Výstupní	0	Finished	Boolean	Skalár

Occupancy - Oznámení o uvolnění/obsazení buňky

Pro oznámení buňce, že se objevila, nebo zmizela sklenice z NFC čtečky slouží tato metoda. Nachází se zde pouze jeden vstupní argument, kterým se oznamuje buňce, že byla obsazena, nebo uvolněna. Pokud se odešle logická hodnota 0, buňka dostane informaci o tom, že byla uvolněna.

Metoda je využívána ve stavu *4. Transport* (obrázek 5.7) a *5. Výroba* (obrázek 5.8).

Tab. 5.7: OPC UA metoda na informování o obsažení/uvolnění buňky - Occupancy

NodeId(String)	Occupancy	**Zobrazované jméno**	Occupancy	
Druh argumentu	**ID**	**Název**	**Datový typ**	**Rozměr**
Vstupní	0	Occupancy	Boolean	Skalár

6 Softwarová implementace

Tato kapitola se zabývá implementací software pro AAS interpreter. Nejprve jsme popsali vývojový software, který byl využit při vývoji, včetně vývojového prostředí (IDE) a frameworku. Dále jsme popsali hlavní program, který slouží k interpretaci. Následně jsme popsal využité knihovny, jak ty, které jsme my vyvíjeli, tak i ty, které jsme získali z externích zdrojů. Na konci téhle kapitoly jsme popsali vývoj zkušebního OPC UA serveru, který následně sloužil k testování celkové funkce AAS interpreteru.

6.1 Vývojový software

Vývojový software je klíčový pro jakýkoli softwarový projekt. Pro tento projekt byly vybrány následující nástroje:

6.1.1 Vývojové prostředí

Pro vývoj AAS interpreteru jsme využil open-source IDE *Visual Studio Code* od firmy Microsoft Corporation. Prostředí je poskytováno zdarma. Zároveň existuje ve verzích pro Windows, macOS i Linux. Jedná se o prostředí, které má širokou podporu programovacích jazyků. Nachází se zde možnost rozsáhle konfigurace a personalizace. Díky integrovanému prohlížeči doplňků je instalace rozšíření velmi jednoduchá.[41]

6.1.2 Využitý framework

Aby jsme měli programování pro *ESP32 S3*, využívám framework *ESP-IDF* ve verzi 5.1.1. *ESP-IDF* je oficiální vývojový framework, který poskytuje rozsáhle API pro programování rodiny mikroprocesorů *ESP32*. Lze využít integrovaných funkcí pro konfiguraci firmwaru, sestavení a nahrání softwaru. [39]

Framework zároveň poskytuje možnost ladění skrze JTAG pomocí *OpenOCD* SW. Pomocí zmíněných způsobů ladění lze mít přímý přístup k paměti a hardwarovým funkcím. Díky nim lze sledovat a ovládat běh programu, nastavovat breakpointy, sledovat hodnoty proměnných a další funkce.[38]

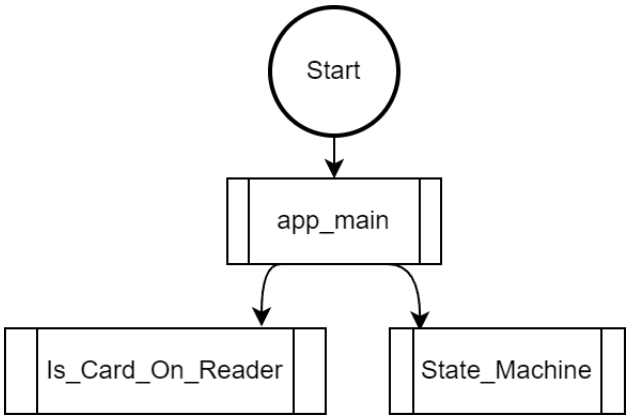
6.2 Hlavní program

Celý hlavní program se skládá z hlavní funkce *app_main*, která se spustí hned po startu mikroprocesoru, a dvou úloh, *Is_Card_On_Reader* a *State_Machine*, které

jsou v nekonečné smyčce. Hlavní program je znázorněn na vývojovém diagramu, který je na obrázku 6.1.

Pro přístup k ethernetovému modulu *W5500* a NFC modulu jsou využity semafory[1], které nám zajistí, že k určité komponentě bude přistupovat pouze jedna úloha. Tenhle způsob je nutný pří více spuštěných úlohách v jeden čas.

Zároveň v hlavním programu je definováno makro[2] *NFC_STATE_DEBUG*, které slouží k vypsání textového řetězce na standardní textový výstup a navíc vypíše všem připojeným Telnet klientům stejný textový řetězec.



Obr. 6.1: Vývojový diagram hlavního programu

6.2.1 Funkce *app_main*

Tahle funkce slouží k základnímu nastavení všech základních využitých komponent a následné zavolání úloh *Is_Card_On_Reader* a *State_Machine*, které obstarávají další funkcionalitu programu. Vývojový diagram je zobrazen na obrázku 6.2.

Jako první je potřeba načíst ID AAS interpreteru z flash uložiště. Tenhle identifikátor je uložen v paměti pod názvem *DataNFC*. Pokud je v kódu definováno makro *IDOFINTERPRETTER*, tak je prvně uloženo do paměti. Pokud je dále měněn kód, již se bude nahrávat nové ID.

Pro možnost sdílet proměnné mezi úlohami je vytvořena proměnná *Parametry* datové struktury *TaskParams*, která obsahuje:

1. *SemaphoreHandle_t* - *xNFCReader* - Handler pro semafor zajišťují nekolidování při práci s NFC modulem.
2. *SemaphoreHandle_t* - *xEthernet* - Handler pro semafor zajišťující nekolidování při práci s ethernetovým modulem *W5500*.
3. *bool* - *CardOnReader* - Příznak, který označuje přítomnost NFC zařízení.

[1]Semafor v FreeRTOS slouží k oznamování skutečností mezi úlohami.[40]

[2]Makro je část kódu. Pokud je někde v kódu umístěn text makra, nahradí jej preprocesor obsahem makra.

4. *pn532_t - NFC_Reader* - Datová struktura knihovny *pn532* zajišťující práci s NFC modulem.
5. *tNeopixelContext - Svetelka* - Datová struktura knihovny *neopixel* zajišťující práci s adresovatelnými RGB LED.

Následující krok v našem procesu je zaměřen na konfiguraci indikačních RGB LED. Tyto LED jsou nezbytné pro vizuální zobrazení stavu našeho systému. K nastavení těchto LED používáme externí knihovnu s názvem *neopixel*.

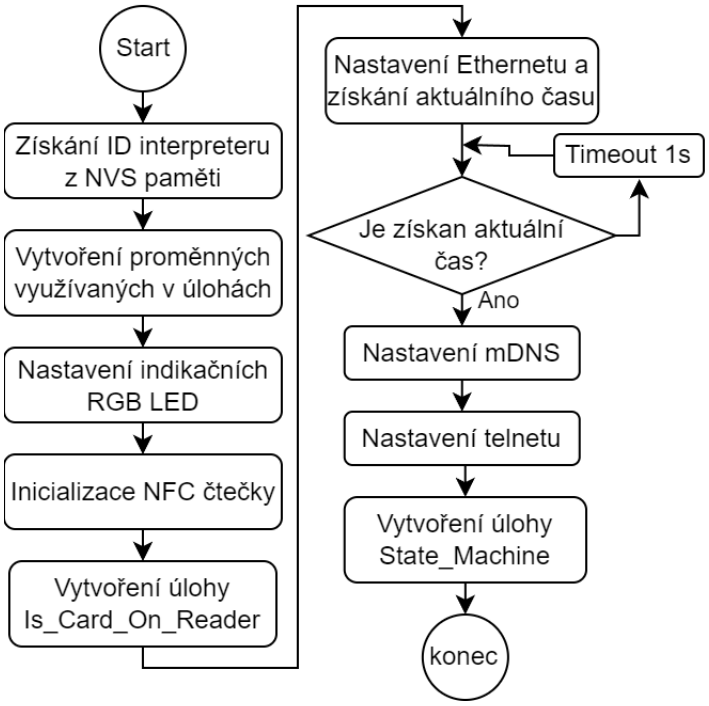
Následuje krok, kterým je inicializace NFC čtečky. Tento proces je realizován pomocí mé vlastní knihovny s názvem *NFC_Reader*. Tato knihovna je navržena tak, aby efektivně pracovala s externí knihovnou *pn532*. Komunikace s NFC čtečkou je nastavena na sběrnici SPI. Po dokončení této inicializace je vytvořena úloha s názvem *Is_Card_On_Reader*. Tato úloha je neustále v provozu a její hlavní funkcí je kontrola, zda je na NFC čtečce přítomno NFC zařízení. Tímto způsobem je zajištěno, že jakékoli NFC zařízení umístěné na čtečce bude okamžitě detekováno.

Další důležitý úkon je nastavení Ethernetu a pomocí něj získat aktuální čas. Vše je prováděno pomocí mé knihovny *OPC_klient*, kde je využita funkce z ukázkového příkladu kódu pro práci s Ethernetovým modulem. Zároveň je získán pomocí protokolu SNTP aktuální čas. Protože nastavení Ethernetového modulu a získání času trvá nějaký čas, je ověřováno, jestli je již získán aktuální čas pomocí statické proměnné *CasNastaven*, která se změní do logického stavu 1 při nastavení aktuálního času.

Dále je nastaven protokol mDNS, který slouží k překladu jmen v naší síti. Tedy není nutné při potřebě připojit se k AAS interpreteru zadávat přímo IP adresu, ale lze zadat textovou adresu. V našem případě nastavujeme textovou adresu interpreteru na textový řetězec *interpreter_XX*, kde XX je ID AAS interpreteru.

Po tomhle kroku je vytvořen Telnet server, pro možnost získat zprávy o stavu stavového automatu po síti. Zde se nám hodí velice výše zmíněný protokol mDNS, kdy nám stačí vědět pouze ID AAS interpreteru, ke kterému se chceme připojit, namísto celé IP adresy.

Finální krok je vytvoření úlohy *State_Machine*, která slouží k interpretaci AAS modelu.



Obr. 6.2: Vývojový diagram funkce *app_main*

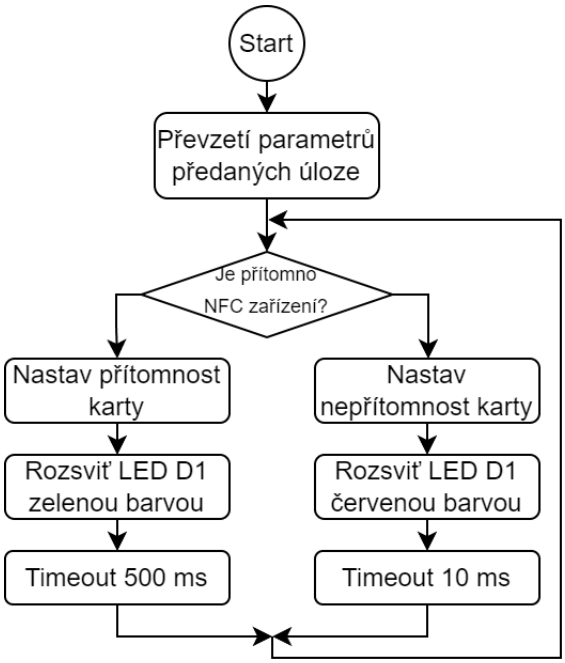
6.2.2 Úloha *Is_Card_On_Reader*

Jedná se o úlohu, která ma za úkol zjišťovat přítomnost NFC zařízení a jeho indikaci RGB LED v nekonečné smyčce. Vývojový diagram je zobrazen na obrázku 6.3.

Jako první krok při vytvoření úlohy musí být načteny vstupní parametry, které byly nastaveny v funkci *app_main*.

Následně je pomocí mé knihovny *NFC_reader* za pomoci externí knihovny *pn532* zjištěna přítomnost NFC zařízení na NFC čtečce. Poté dle přítomnosti je nastavena proměnná, která indikuje přítomnost karty. Adresovatelná RGB LED D4 indikuje červenou barvou nepřítomnost NFC zařízení a zelená naopak indikuje přítomnost NFC zařízení.

Lze si všimnout rozdílu v čekání na další iteraci, kdy při nepřítomnosti je doba čekání jen 10 ms, protože AAS interpreter nemá při nepřítomnosti NFC zařízení žádný další úkol.



Obr. 6.3: Vývojový diagram úlohy *Is_Card_On_Reader*

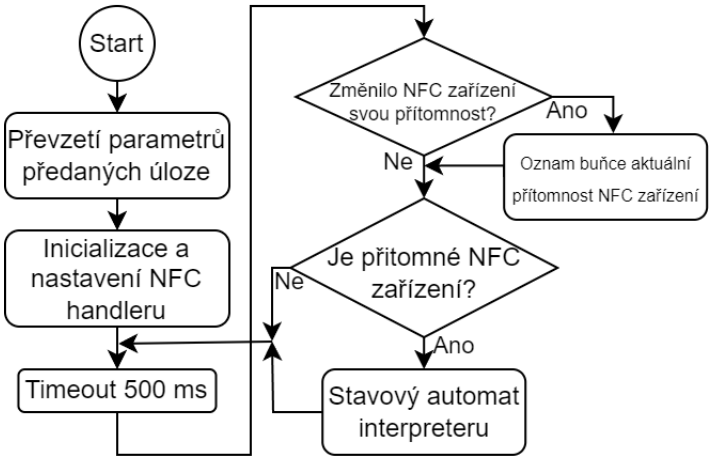
6.2.3 Úloha *State_Machine*

Zmíněná úloha má na starosti celý proces interpretace popsaný v kapitole 5.3. Vývojový diagram úlohy je znázorněn na obrázku 6.4. Jedná se o nekonečnou úlohu, kde její část probíhá v iteracích.

Jako první úloha načte předané parametry z funkce *app_main*. Díky načteným parametrům dokáže v následujícím kroku inicializovat a nastavit NFC handler, který slouží k zajištění integrity dat mezi NFC zařízením a mikroprocesorem.

Po začátku iterace je potřeba ověřit, jestli se nezměnila přítomnost NFC zařízení. V případě, že ano, je nutné tuhle skutečnost sdělit přítomné buňce, u které se AAS interpreter nachází. S tím souvisí i další rozhodování, které rozhodne, jestli se dle přítomnosti NFC zařízení vykoná krok stavového automatu, nebo započne další iterace.

Stavový automat provádí operace, které jsou dříve vyjmenovány v kapitole 5.3. Po dokončení stavu započne další iterace.



Obr. 6.4: Vývojový diagram úlohy *State_Machine*

6.3 Použité knihovny

Aby se zlepšila přehlednost kódu a zároveň zjednodušila práce s komponentami a protokoly, jsou využívány knihovny. Velká výhoda knihoven je možnost jejich testování po částech a případná údržba kódu.

6.3.1 Vlastní knihovny

Protože neexistují knihovny na námi vyžadované funkce, identifikátoru jsme si vyvinout vlastní, které nám požadovanou funkcionalitu nabídnou.

Do každé mé knihovny jsme definovali 2 makra pro ladění. První makro *NFC_RECIPES_ALL_DEBUG* slouží pro vypis všech informací. Druhé makro *NFC_RECIPES_DEBUG* slouží k výpisu jen hlavních informací. Obdobně jsou pojmenovány i v dalších knihovnách.

Práce s NFC čtečkou na vysoké úrovni - NFC_reader

Protože jsme požadovali jednodušší práci s knihovnou *pn532*, vytvořili jsme tuhle knihovnu. Jedná se o knihovnu, která definuje datové struktury pro ukládání informací o sklenici (*TRecipeInfo*) a kroky receptu (*TRecipeStep*) zmíněných v kapitole 5.2. Dále knihovna obsahuje funkce, které jsou využity k inicializaci NFC modulu, načítání informací o sklenici a kroků receptu do pole kroků receptu, správu pole kroků receptu a zápis informací o sklenici a kroků receptu do NFC zařízení.

Velkou zajímavost téhle funkce je jednoduchost zápisu dat do paměti NFC zařízení, kdy knihovna rozezná o jaký typ NFC zařízení se jedná dle délky ID zařízení. Dle toho zvolí algoritmus pro zápis dat do paměti. Mezi velmi využívanou funkci

v následné knihovně, která zajišťuje integritu (*NFC_handler*), je možnost data do sklenice nahrát a záhy poté ověřit, jestli jsou zapsány správně.

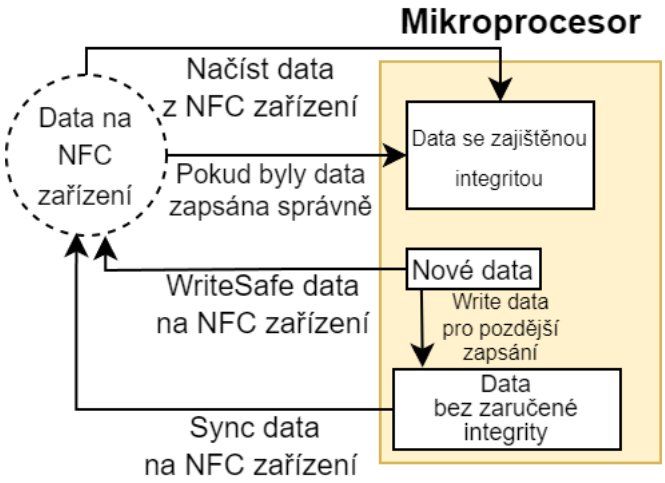
Knihovna dle aktivity, která probíhá mezi NFC zařízením, pracuje s RGB LED D1. Kdy při zápisu do paměti NFC zařízení rozsvítí diodu červenou barvou a při čtení rozsvítí diodu zelenou barvou.

Knihovna zároveň ověřuje korektnost receptu pomocí kontrolního součtu zmíněného v kapitole 5.2.

Zajištění integrity dat na NFC zařízení - NFC_handler

V námi definovaném požadavku na AAS interpreter v kapitole 5.1 je požadavek na udržení integrity mezi NFC zařízením a mikroprocesorem. K tomuto účelu nám slouží tahle knihovna. Pracuje jako nástavba nad knihovnou *NFC_reader*.

Knihovna definuje datovou strukturu *THandlerData*, která uvnitř sebe uchovává data karty se zaručenou integritou, tak i upravená data, která se teprve nahrají do paměti NFC zařízení. Dále poskytuje funkce pro práci s výše uvedenou strukturou. Funkce jsou znázorněny na obrázku 6.5.



Obr. 6.5: Proces zajištění integrity dat mezi NFC zařízením a mikroprocesorem

Pro správné využití knihovny je nejprve nutno strukturu *THandlerData* nastavit pomocí funkce *NFC_Handler_Init* a *NFC_Handler_SetUp*. Následně se pomocí funkce *NFC_Handler_LoadData* načtou data z NFC zařízení. Pro úpravu informací o sklenici nebo kroku receptu je potřeba provést kopii pomocí příkazu *NFC_Handler_GetRecipeInfo* nebo *NFC_Handler_GetRecipeStep*. S touhle kopií můžeme dále pracovat. Po úpravě dat je možnost data nahrát hned NFC tagu (pomocí funkce *NFC_Handler_WriteSafeInfo*), nebo do pracovní paměti (pomocí funkce *NFC_Handler_WriteInfo*). Obdobné funkce existují i pro kroky receptu.

Změny, které se provedli v pracovní paměti, se nahrají všechny najednou do NFC zařízení pomocí funkce *NFC_Handler_Sync*.

Komunikace s OPC UA serverem - OPC_klient

Správné nastavení Ethernetového čipu *W5500* obstarává má knihovna *OPC_klient*. Dále aplikuje OPC UA metody zmíněné v kapitole 5.4 a vytváří z nich klasické *C* funkce, které komunikují s OPC UA serverem. Tyhle elementární funkce jsou využity v knihovně *NFC_recipes*.

Práci s Ethernetovým modulem obstarává kód inspirovaný od ukázkového kódu pro komunikaci skrze Ethernet. Je zde zároveň získán aktuální čas skrze protokol Simple Network Time Protocol. Jako výchozí server je zvolen *tik.cesnet.cz* a jako záložní je využit *tak.cesnet.cz*. Pokud bychom nechtěli mít připojený AAS interpreter k veřejnému internetu, museli bychom mít v lokální síti SNTP server.

Práce s recepty na NFC zařízení - NFC_recipes

Knihovna slouží ke správě receptů uložených v paměti mikroprocesoru. Zároveň jsou zde předdefinovány celé recepty, které je možno nahrát do NFC zařízení. Další funkcionalitu, kterou knihovna poskytuje jsou funkce k práci s OPC UA metodami implementovanými v knihovně *OPC_klient*.

Je zde definována struktura *CellInfo*, která slouží k udržení informací o buňce. Je zde uloženo ID buňky, IP adresa a její délka. Dále je v poli uložena informace, které procesní úkony dokáže buňka vykonat. Druhá datová struktura *Reservation* slouží k předávání informací ohledně poptávek a rezervací.

Dále jsou zde výčtovým datovým typem (*enum*) definovány stavy stavového automatu, který se nachází v úloze *State_Machine* (popsáno v kapitole 6.2.3). Následně jsou zde popsány výčtovým datovým typem také druhy procesů (*StorageAlcohol*, *Shaker*, atd.), typy alkoholických součástí nápojů (*Vodka*, *Rum*, atd.) a typy nealkoholických částí nápojů (*Water*, *Cola*, atd.).

6.3.2 Externí knihovny

Pro různé funkce FW jsme našli na internetu již hotové funkce. Tedy by bylo zbytečné programovat je od začátku, když svými možnostmi nám poskytují všechny potřebné funkce.

Práce s NFC čtečkou na nízké úrovni - pn532

Práci na nízké úrovni s NFC čipem *PN532* obstarává stejnojmenná knihovna. Byla stažena z Github.com od uživatele *@thanhbinh89* .[43] Jedná se o adaptaci knihovny,

která je původně pro Arduino IDE, pro mikroprocesory *ESP32*. Implementována je pouze komunikace skrze SPI. Knihovna nabízí práci s NFC tagy *MIFARE® Classic*, *MIFARE® Ultralight* a *NTag2xx*. Knihovna je využívána mou nadstavbovou knihovnou *NFC_reader*.

Pro správnou funkci je nejprve nutné využít funkci *pn532_spi_init*, která nastaví do struktury *pn532_t* parametry SPI sběrnice. Následně je nutné začít přenos dat funkcí *pn532_begin*. Následně již mohou být volány funkce pro práci s *PN532* čipem.

Implementace protokolu OPC UA - open62541

Knihovna *open62541* je open-source implementace OPC UA. Je navržena tak, aby byla co nejvíce přizpůsobitelná a snadno integrovatelná do existujících systémů. Jedná se o certifikovanou knihovnu přímo od *OPC Foundation*.[42] Já jsem konkrétně využil adaptaci knihovny pro mikroprocesor *ESP32*, staženou z Github.com od uživatele *@cmbahadir*. Jedná se o vzorový kód pro OPC UA server. Ovšem obsahuje i možnosti pro vytvoření OPC UA klienta.[44]

Knihovnu využívá knihovna *OPC_Klient*. Zde je využití ve formě OPC UA klienta velmi jednoduché. První je potřeba využít funkci pro vytvoření nastavení klienta *UA_Client_new UA_Client_connect*, která slouží k připojení k OPC UA serveru. Následně je pro zavolání OPC UA metody nutné využít funkci *UA_Client_call*. Po dokončení práce s OPC UA serverem je nutné OPC UA klienta odpojit pomocí příkazu *UA_Client_disconnect* a uvolnit paměť po datech klienta pomocí příkazu *UA_Client_delete*.

Ovládání adresovatelných LED WS2812B - neopixel

Knihovna neopixel od firmy ZORXX Software je využita k ovládání adresovatelných RGB LED. Knihovna má velmi jednoduchou instalaci, kdy stačí vybrat v *ESP-IDF Component Registry* knihovnu a ona se sama přidá do projektu.[45]

Proces práce s LED je následující. Jako první je potřeba inicializovat RGB LED funkcí *neopixel_Init*. Dalším krokem je již nastavení LED pomocí funkce *neopixel_SetPixel*. Po skončení práce s LED se provede odinicializování funkcí *neopixel_Deinit*.

Přiřazení jmen Zařízením v síti - mDNS - mdns

Knihovna, která dovoluje využít funkcionalitu multicast Domain Name System (multicast Domain Name System), je využita k připojení k AAS interpreteru pomocí doménového názvu.

Knihovna se používá následujícím způsobem. První je potřeba inicializovat funkci *mdns_init*. Poté lze nastavit *hostname*, tedy doménovou adresu zařízení, funkcí

mdns_hostname_set. Následně se nastaví jméno instance, název zařízení v síti, pomoci funkce *mdns_instance_name_set*. Poslední, co je potřeba udělat, je nastavit protokoly, které zařízení podporuje. K tomu slouží funkce *mdns_service_add*.

Implementace protokolu Telnet - libtelnet

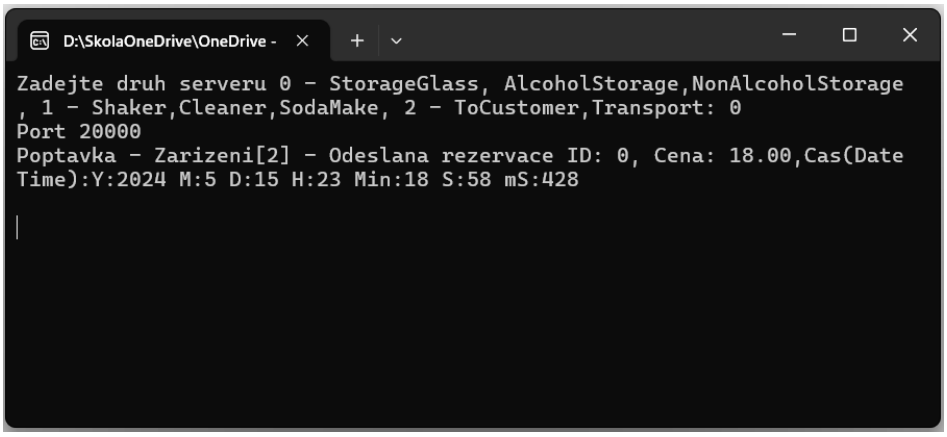
Pro možnost monitorovat chování AAS interpreterů přes síť jsem zvolil protokol Telnet (Teletype network). K jeho implementaci jsem využil knihovnu *libtelnet* od autora *@seanmiddleditch*. Jedná se o bezpečnou a správnou implementaci Telnet protokolu. Díky knihovně lze vytvářet Telnet server i Telnet klienta.[46]

V našem případě jsem vytvářel jen Telnet server. Pro vytvoření je využívána funkce *telnet_server_create* a následně je textový řetězec posílán skrze funkci *telnet_printf*.

6.4 Testovací OPC UA server

Protože jsem potřeboval vyzkoušet funkčnost OPC UA klienta, vytvořil jsem jednoduchý OPC UA server, který simuluje funkci OPC UA serveru nasazeného na procesní buňce. Server jsem naprogramoval v jazyce *C++*, opět na open-source implementaci protokolu OPC UA Open62541. Zde jsem již sám sestavil knihovnu ze zdrojových dat pomocí programu cmake.

OPC UA server je vyvinut jako konzolová aplikace pro operační systém Microsoft Windows. Server po spuštění nabízí výběr mezi ID virtuálními druhy buněk. Výběr lze taky provést argumentem při spouštění. To mi dovoluje spustit všechny OPC UA servery najednou pomocí *.bat* souboru. Výběr nyní slouží pouze ke změně portu, který OPC UA server využívá, ten je po zvolení ID vypsán do konzoly. Výpis je zobrazen na obrázku 6.6.



Obr. 6.6: Testovací OPC UA server

OPC UA server je nastaven pomocí funkce *UA_ServerConfig_setMinimal*. Následně jsou volány funkce pro zaregistrování OPC UA metod. Uvnitř funkcí jsou nastaveny vstupní a výstupní argumenty metody definované v kapitole 5.4. Poté je přidána OPC UA metoda na OPC UA server. Při přidávání je zvolena i *Callback*[3] funkce. V *Callback* funkci je zvoleno, co se stane při zavolání OPC UA metody. My pracujeme s vektorem, který obsahuje rezervace. Do vektoru jsou přidávány a měněny rezervace právě v *Callback* funkci. Zároveň je v ní zajištěn výpis informací (lze vidět na obrázku 6.6). Poslední krok, který je potřeba udělat, je server spustit. To se provede pomocí funkce *UA_Server_run*.

Mezi další funkci, která mi ulehčila práci ladění FW mikroprocesoru, je automatické dokončení procesu. Po zavolání OPC UA metody *DoProcess* je spuštěn časovač na 10 s. Po uplynutí času je proces označen jako dokončený.

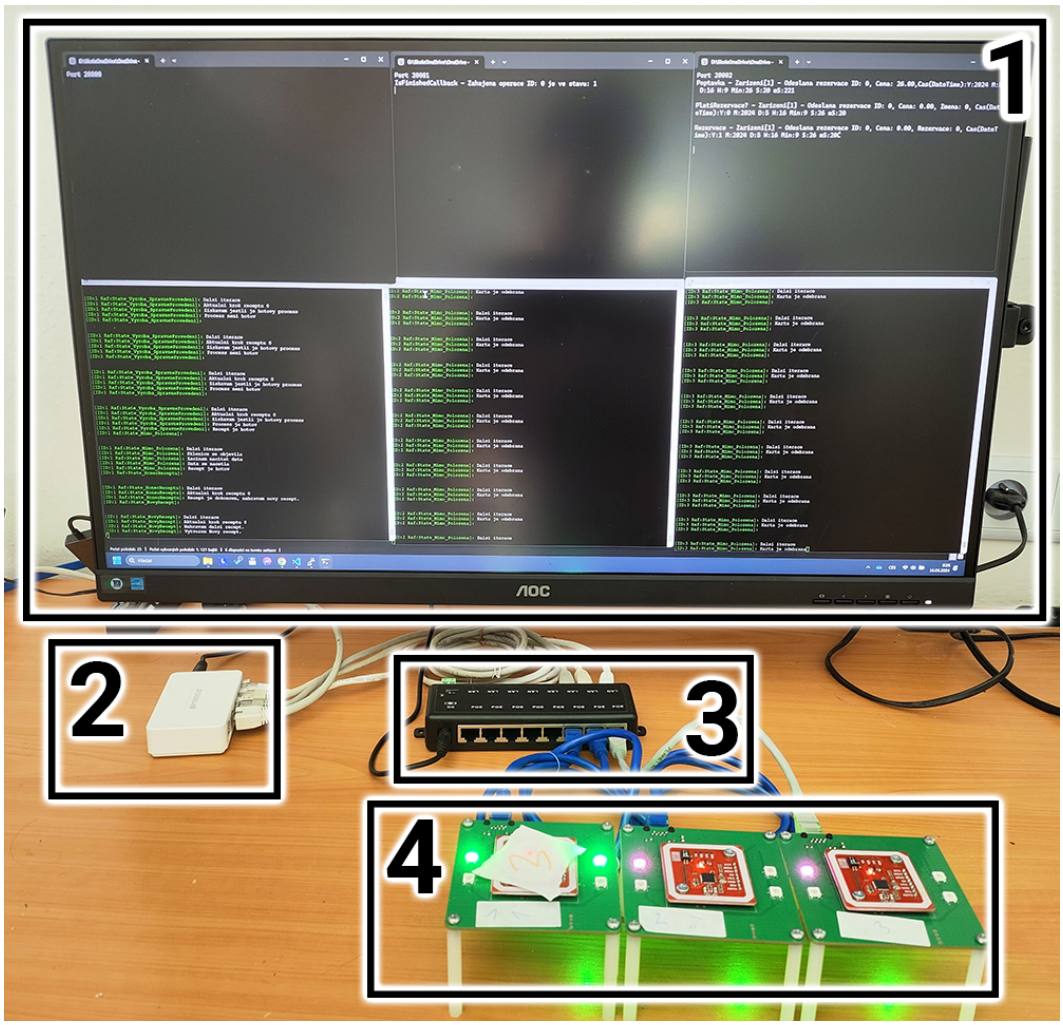
[3]Callback je funkce, která se zavolá poté, co bude provedena předem stanovená akce.

7 Testování a zhodnocení funkčnosti

Abychom ověřili funkčnost našeho FW AAS interpreteru, bylo nutné sestavit testovací stanoviště, kde se dají snadno sledovat všechny průběhy procesů, jak od AAS interpreterů, tak i od OPC UA serveru. Poté jsme definovali soubor scénářů pro testování. Nakonec jsme provedli komplexní zhodnocení a interpretaci výsledků získaných z těchto testů.

7.1 Testovací stanoviště

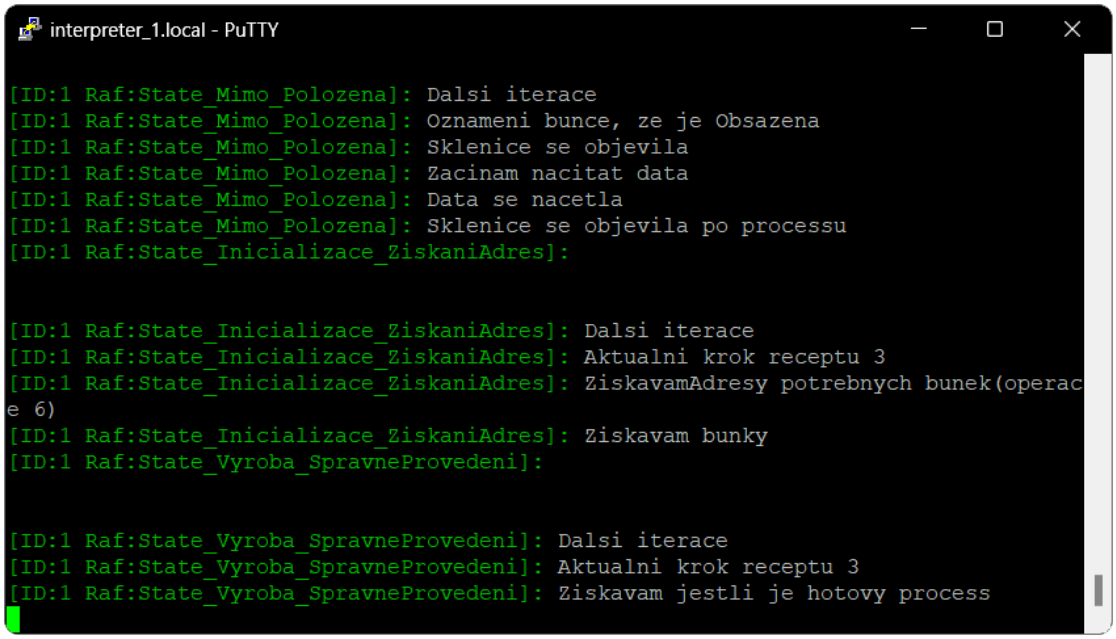
Navrhli jsme testovací stanoviště tak, aby bylo možné simulovat transportní operace (přemístěním NFC tagů rukou mezi AAS interpretery), které by se odehrávaly v testbedu I4.0 - Barman. Důležitým aspektem designu bylo také to, že stanoviště muselo umožnit sledování všech procesů probíhajících v systému.



Obr. 7.1: Přehled testovacího stanoviště AAS interpreterů

Zde je popis prvků, které se na testovacím stanovišti (obrázek 7.1) nacházejí:

1. Zobrazovací monitor - Monitor slouží k zobrazení informací z OPC UA serveru (první řádek konzol) a informací z AAS interpreteru skrze protokol Telnet (druhý řádek konzol). Jako Telnet klient byl zvolen program *PuTTY*[1]. Detailní pohled na konzoli je znázorněn na obrázku 7.2. Sloupce konzol odpovídají jednotlivým simulovaným procesním buňkám společně s příslušným AAS interpreterem.



Obr. 7.2: Konzole v programu *PuTTY* s připojeným AAS interpreterem skrze protokol Telnet

2. Ethernetový switch - Jedná se o síťový prvek, který dokáže propojit více AAS interpreterů do jedné sítě. Jedná se o switch *OPTFOCUS 5 port Ethernet switch*.
3. Pasivní PoE panel - Aby mohly být AAS interpretery napájeny skrze PoE, využil jsem dostupný PoE panel, který poskytuje pasivní PoE napájení pro 8 konektorů. Panel funguje tak, že se z jedné strany připojí ethernetový konektor s přístupem k internetu a z druhé strany se připojí ethernetový konektor, kterým bude AAS interpreter připojen. Napájení je zajištěno síťovým adaptérem s 12 V napěťovým výstupem a maximálním proudem 2 A.
4. AAS interpretery - Při testování jsem využil 3 kusy AAS interpreterů. Zařízení byla upraveny přemostěním tranzistoru Q1 popsaném z kapitole 4.2.3. Každý AAS interpreter jsem popsal ID zařízení.

[1]PuTTY je volně dostupný open-source Telnet a SSH klient.[47]

7.2 Testovací scénáře

K testování jsme využili fiktivní recept, který v sobě ukrýval kroky receptu. Pokud byl recept dokončen, AAS interpreter fiktivní recept do NFC tagu nahrál. Testování probíhalo při výskytu více NFC tagů. Při testování jsem volil následující scénáře:

1. Standardní - Jeden ze scénářů testování byl takový, že jsme nechali AAS intepreter vykonávat příkazy a pouze jsme přemísťovali NFC tagy dle příkazů z transportního OPC UA serveru.
2. Odolnost vůči odstranění - V tomhle scénáři byl NFC tag odstraňován z AAS interpreteru mimo časy, kdy měl proběhnout transport. Simulovalo se tak oddělání sklenice z neznámých důvodů
3. Transport na špatné místo - Zde je postup stejný jako ve standardním případě. Ovšem při provedení transportu je sklenice umístěna na špatné cílové zařízení transportu.

7.3 Dosažené výsledky

Hned na začátku testování jsme ověřili, jestli funguje ověřování korektního receptu, vložením prázdného NFC tagu na AAS interpreter. Zde proběhla správné rozpoznání nekorektního receptu. Následně AAS interpreter nahrál do paměti NFC zařízení recept *Návrat do skladu*.

První testovací scénář jsme otestovali na fiktivním receptu, tak i na receptu *Návrat do skladu*. Fáze dotazování u fiktivního receptu proběhla správně stejně jako rezervace. Následně mi bylo oznámeno 3. OPC UA serverem, že mám přemístit NFC tag na 2. AAS interpreter. Zde AAS interpreter oznámil 2. OPC UA serveru, že se objevila sklenice. Následně byl započat výrobní proces. Pomocí OPC UA metod bylo zjišťováno dokončení výrobního procesu. Po dokončení se nastavil krok receptu jako hotový a aktivní krok receptu na následující. Další fáze testování tohoto scénáře probíhala obdobně. Zde nebyl žádný problém a celý systém se choval přesně podle teoretických předpokladů.

Následujícím scénářem byla ověřena odolnost vůči odstranění. Vše probíhalo na stejném fiktivním receptu. Po odstranění a následném vrácení NFC tagu se stavový automat zachoval správně dle navržené funkce a dostal se do stavu, ve kterém měl pokračovat. Chybný stav nastal pouze v případě, že bylo zapisováno do NFC tagu a ten byl v tu chvíli odstraněn. Zde byla zapsána jen část změny paměti. To vzneslo nejistotu v následující postup AAS interpretru. Ve většině případů probíhal stav, který právě proběhl znovu, protože se nestihla zapsat změna do paměti. Tohle chování nám ovšem nevadí, protože zopakování OPC UA metody nezpůsobí nefunkčnost systému, pouze zpomalení celkového systému.

Na třetí testovací scénář jsme přidali do FW řešení v podobě funkce na přidání dalšího kroku receptu. Jedná se o transportní krok receptu. Zde při umístnění NFC tagu na špatný AAS intepreter, byl vytvořen a následně rezervován transport do správné buňky. Dokonce si FW poradil i se situací, kdy byl i po přidání zmíněného kroku opět NFC tag umístěn na špatnou buňku. Po finálním přesunutí NFC tagu na správnou buňku se postupně označovaly kroky receptu, které obsahovaly transportní operaci, jako dokončené. Recept následně pokračoval dál.

Testováním jsem došel k závěru, že celkové řešení AAS interpreteru neobsahuje fatální chyby, které by zapříčinily nefunkčnost celého systému.

Závěr

V rámci této bakalářské práce jsme se zaměřili na komplexní problematiku integrace Asset Administration Shell modelu do vestavěných systémů, konkrétně na AAS interpreter, který využívá technologii NFC.

V úvodní kapitole jsme se věnovali fenoménu Průmysl 4.0 a s tím spojenému konceptu AAS. Na konci této kapitoly jsme představili praktický příklad Průmyslu 4.0 testbed I4.0 - Barman, kde jsme již naznačili funkčnost zařízení, které jsme vyvíjeli.

V druhé kapitole jsme se věnovali detailnímu popisu protokolu OPC UA. Nejprve jsme popsali založení iniciativy OPC a historie OPC UA. Dále jsme uvedli cíle, které vedly k vytvoření tohoto protokolu. V závěrečná části kapitoly jsme se zaměřili na popis adresního prostoru, který se skládá z uzlů, třídy uzlů, jejich normativně stanovenými standardy a sadami služeb, které jsou podporovány v rámci protokolu OPC UA.

Třetí kapitola byla zaměřena na detailní analýzu NFC technologie. Byla zde představena technologie RFID, která je základem pro NFC technologii, následována klasifikací a popisem režimů přenosu. Závěrečná část kapitoly se věnovala charakteristice nejčastěji používaných NFC tagů.

Ve čtvrté kapitole, která je součástí praktické části práce, jsme se zabývá analýzou stávajícího a nově navrženého hardwaru univerzální NFC čtečky. Stávající návrh jsme podrobně rozebrali s využitím blokového diagramu a identifikovali problémy tohoto návrhu. Nový návrh jsme prezentovali pomocí blokového diagramu a podrobně popsali. Na fyzickém modelu jsme znázornili rozmístění jednotlivých prvků obvodu. Dále jsou specifikovány komponenty, s nimiž jsme pracovali v rámci softwarové implementace. V důsledku zjištěných chyb při testování zařízení jsme navrhli možná vylepšení. První vylepšení se týká funkce napájení skrze PoE a jeho bezpečnosti společně se splněním normativních požadavků, druhé vylepšení pak řeší eliminaci chyby v návrhu obvodu pro obsluhu adresovatelných RGB LED.

V páté kapitole jsme se zaměřili na návrh modelu AAS. Nejprve jsme definovali specifické požadavky na tento model, a to splnění principů Průmyslu 4.0, výběru vhodného NFC tagu a funkce pro správu uložených dat. Po stanovení těchto požadavků jsme specifikovali datovou strukturu, která je ukládána do paměti NFC tagu. Jednotlivé části této datové struktury a způsob ověřování správnosti receptu jsme podrobně popsali v následující části textu. Značnou část kapitoly jsme věnovali našemu návrhu interpretace, kde je nejprve popsán obecný stavový automat rozdělený do základních stavů. Tyto stavy jsou dále rozvedeny a ilustrovány pomocí vývojových diagramů. Z tohoto navrženého způsobu interpretace vyplynuly požadavky na OPC UA metody, které musí být v systému testbedu přítomny.

V šesté kapitole, která je zaměřena na softwarovou implementaci, jsme věnovali největší úsilí. V rámci zadání bylo nezbytné začlenit do firmwaru způsob interpretace a komunikace s OPC UA serverem. Pro naše specifické potřeby jsme vyvinuli řadu pokročilých funkcionalit. První z nich je zjednodušení práce s knihovnou pro komunikaci s NFC čipem, což nám umožnilo efektivnější psaní následujícího kódu. Další důležitou funkcionalitou, kterou jsme museli implementovat, je mechanismus udržující integritu mezi pamětí mikroprocesoru a pamětí NFC tagu. S touto pamětí souvisí také možnost správy receptů, kterou nám umožňuje další naše knihovna. Přítomnost a interakci s NFC tagem nám signalizují dvě adresovatelné RGB LED. Další dvě zůstávají volné pro další funkce. Po práci s pamětí jsme museli do firmwaru začlenit možnost komunikace s OPC UA serverem. To se nám podařilo díky adaptaci otevřené knihovny *open62541* pro náš mikroprocesor. Aby bylo ladění programu co nejjednodušší, začlenili jsme do programu podporu protokolu Telnet, což přesahuje rámec zadání naší práce. Díky tomuto protokolu můžeme sledovat stav stavového automatu, který provádí interpretaci, prostřednictvím sítě. S tímto protokolem souvisí protokol mDNS, který umožňuje jednodušší připojení prostřednictvím sítě jen pomocí doménového názvu. Součástí softwarové implementace je také vytvoření testovacího konzolového OPC UA serveru, aby bylo možné testování co nejvíce přiblížit prostředí testbedu. Zde jsme již sami sestavili knihovnu *open62541* pro implementaci OPC UA.

V sedmé kapitole jsme podrobně popisovali proces testování celého navrženého systému. Nejprve jsme představili rozmístění testovacího stanoviště. Následně jsme definovali testovací scénáře, jejichž úkolem bylo ověřit jak základní funkčnost, tak i schopnost systému zvládnout výjimečné situace, které předpokládáme, že mohou nastat. Závěrem této kapitoly je konstatování, že navržený systém úspěšně prošel všemi našimi testovacími scénáři a může být považován za plně funkční a robustní vůči výjimečným situacím.

Další práce, které by mohly navázat na tuto práci, by se měly zaměřit na implementaci AAS interpreteru do systému testbedu. Současně je nezbytné začlenit OPC UA server do již existujících procesních buněk. Kromě toho je nutné rozšířit algoritmy pro výběr výherní procesní buňky AAS interpreterem.

Literatura

[1] ARORA, Nitin a SHARMA, Ashish K., AHATSHAM, Hayat a SHAHARE, Vivek (ed.), 2023. Introduction to Industry 4.0. In: NAMASUDRA, Suyel a AKKAYA, Kemal. *Blockchain and its Applications in Industry 4.0*. 2023. Springer Singapore, s. 29-59. ISBN 978-981-19-8729-8.

[2] PÁSEK, Jan. Digitální transformace průmyslu. Online. S. 21-24. Dostupné z: `https://www.vutbr.cz/www_base/priloha_fs.php?dpid=183201&skupina=dokument_priloha`. [cit. 2023-12-24].

[3] TECHNOLOGICKÉ CENTRUM AV ČR, 2017. *Technology brief: Technologie pro Průmysl 4.0*. Online. Dostupné z: `https://vedavyzkum.cz/z-domova/technologicke-centrum-av-cr/technology-brief-technologie-pro-prumysl-4-0`. [cit. 2023-12-05].

[4] 2015. Referenční model struktury Industrie 4.0 RAMI 4.0. *Automa*. Roč. 2015, č. 11, s. 43-44. ISSN 1210-9592.

[5] ZEZULKA, František; MARCOŇ, Petr; VESELÝ, Ivo a SAJDL, Ondřej, 2016. Industry 4.0 — An Introduction in the phenomenon. Online. *IFAC-PapersOnLine*. Vol. 2016, no. 49, article 25, s. 8-12. ISSN 2405-8963. Dostupné z: `https://doi.org/10.1016/j.ifacol.2016.12.002`. [cit. 2023-12-22].

[6] DE LEEUW, Valentijn a BARTOŠÍK, Petr, 2019. Industrie 4.0 Asset Adminstration Shell: koncepce a využití v praxi. Online. *Automa*. Roč. 2019, č. 8-9, s. 73-75. ISSN 1210-9592. Dostupné z: `https://www.automa.cz/Aton/FileRepository/pdf_articles/12444.pdf`. [cit. 2023-12-21].

[7] BHOSALE, Pushparaj; KASTNER, Wolfgang a SAUTER, Thilo, 2021. A Centralised or Distributed Risk Assessment using Asset Administration Shell. Online. In: *International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vasteras, Sweden: IEEE, s. 1-4. ISBN 978-1-7281-2989-1. Dostupné z: `https://doi.org/10.1109/ETFA45728.2021.9613152`. [cit. 2023-12-26].

[8] KACZMARCZYK, Václav; BAŠTÁN, Ondřej; BRADÁČ, Zdeněk a ARM, Jakub, 2018. An Industry 4.0 Testbed (Self-Acting Barman): Principles and Design. Online. *IFAC-PapersOnLine*. Roč. 2018, č. 51, article 6, s. 263-270. ISSN 2405-8963. Dostupné z: `https://doi.org/10.1016/j.ifacol.2018.07.164`. [cit. 2023-12-24].

[9] KACZMARCZYK, Václav, 2019. *INDUSTRY 4.0 TESTBED Barman*. Online. Dostupné z: http://www.drink.skupra.cz/. [cit. 2023-12-24].

[10] KLEMENT, Petr, 2023. *Algoritmy řízení výroby pro demonstrátor Průmyslu 4.0*. Online, Bakalářské práce, vedoucí Václav Kaczmarczyk, Ph.D. Brno: VUT v Brně. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=253820. [cit. 2023-12-24].

[11] BOCH, Jan, 2022. *Autonomní sklad kapalin s chladicím systémem*. Online, Bakalářské práce, vedoucí Michal Husák. Brno: VUT v Brně. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=241380. [cit. 2023-12-24].

[12] SCHWARZ, Michael a BÖRCSÖK, Josef, 2013. A survey on OPC and OPC-UA: About the standard, developments and investigations. Online. In: *XXII International Symposium on Information, Communication and Automation Technologies, ICAT*. Sarajevo, Bosnia and Herzegovina: IEEE, s. 1-6. ISBN 978-1-4799-0431-0. Dostupné z: https://doi.org/10.1109/ICAT.2013.6684065. [cit. 2023-12-28].

[13] VOJÁČEK, Antonín. *Průmyslová komunikace OPC UA - 1.díl - popis protokolu*. Online. HW SERVER S.R.O. Automatizace.hw.cz. 2020. Dostupné z: https://automatizace.hw.cz/prumyslova-komunikace-opc-ua-1dil-popis-protokolu.html. [cit. 2024-05-13].

[14] OPC FOUNDATION. *OPC foundation reference*. Online. 2024. Dostupné z: https://reference.opcfoundation.org/. [cit. 2024-05-13].d

[15] BRUCKNER, Dietmar; STĂNICĂ, Marius-Petru; BLAIR, Richard; SCHRIEGEL, Sebastian; KEHRER, Stephan et al., 2019. An Introduction to OPC UA TSN for Industrial Communication Systems. Online. *Proceedings of the IEEE*. Roč. 2019, č. 107, s. 1121-1131. ISSN 1558-2256. Dostupné z: https://doi.org/10.1109/JPROC.2018.2888703. [cit. 2023-12-28].

[16] COSKUN, Vedat; OK, Kerem a OZDENIZCI, Busra, 2012. *Near field communication: from theory to practice*. Chichester: Wiley. ISBN 978-111-9971-092.

[17] LAINE, Teemu, ARENDARENKO, Ernest (ed.), 2011. *Smart tagging technologies in pervasive learning environments*. Online. ResearchGate. Dostupné z: https://www.researchgate.net/publication/290628144_Smart_tagging_technologies_in_pervasive_learning_environments. [cit. 2023-12-28].

[18] MADLMAYE, Gerald; LANGER, Josef; KANTNER, Christian; SCHARINGER, Josef a SCHAUMULLER-BICHL, Ingrid, 2009. Risk Analysis of Over-the-Air Transactions in an NFC Ecosystem. Online. In: *International Workshop on Near Field Communication, NFC*. Hagenberg, Austria: IEEE, s. 1-6. ISBN 978-0-7695-3577-7. Dostupné z: https://doi.org/10.1109/NFC.2009.17. [cit. 2023-12-29].

[19] SUNG-WOOK, Park a IM-YEONG, Lee, 2016. Mutual Authentication Scheme Based on Lattice for NFC-PCM Payment Service Environment. *International Journal of Distributed Sensor Networks*. Roč. 2016, č. 12, s. 1-6. ISSN 1550-1329.

[20] ÚŘAD PRO TECHNICKOU NORMALIZACI METROLOGII A STÁTNÍ ZKUŠEBNICTVÍ, 2021. ČSN ISO/IEC 14443-1, *Karty a bezpečnostní zařízení pro osobní identifikaci - Bezkontaktní objekty s vazbou na blízko - Část 1: Fyzikální charakteristiky*. Čtvrté. Praha: Česká agentura pro standardizaci.

[21] ÚŘAD PRO TECHNICKOU NORMALIZACI METROLOGII A STÁTNÍ ZKUŠEBNICTVÍ, 2021. ČSN ISO/IEC 14443-2, *Karty a bezpečnostní zařízení pro osobní identifikaci - Bezkontaktní objekty s vazbou na blízko - Část 2: Radiofrekvenční výkonové a signálové rozhraní*. Čtvrté. Praha: Česká agentura pro standardizaci.

[22] *NTAG213/215/216 Datasheet: NFC Forum Type 2 Tag compliant IC with 144/504/888 bytes user memory*, 2015. Online. In: NXP SEMICONDUCTORS N.V. NXP. Dostupné z: https://www.nxp.com/docs/en/data-sheet/NTAG213_215_216.pdf. [cit. 2023-12-29].

[23] *MF1S50YYX_V1: MIFARE Classic EV1 1K - Mainstream contactless smart card IC for fast and easy solution development*, 2018. Online. In: NXP B.V. NXP. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MF1S50YYX_V1.pdf. [cit. 2023-12-29].

[24] *MF0ULX1: MIFARE Ultralight EV1 - Contactless ticket IC*, 2019. Online. In: NXP B.V. NXP. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MF0ULX1.pdf. [cit. 2023-12-29].

[25] *ST25TN512 ST25TN01K: NFC Forum Type 2 tag IC with up to 1.6 Kbits of EEPROM*, 2023. Online. In: STMICROELECTRONICS. STMicroelectronics. Dostupné z: https://www.st.com/resource/en/datasheet/st25tn512.pdf. [cit. 2023-12-29].

[26] *MF3D(H)x2: MIFARE DESFire EV2 contactless multi-application IC*, 2019. Online. In: NXP B.V. NXP. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MF3DX2_MF3DHX2_SDS.pdf. [cit. 2023-12-30].

[27] WILHITE, Tamara, 2023. *Differences in A & B for ISO 14443*. Online. In: LEAF GROUP LTD. Sapling. Dostupné z: https://www.sapling.com/10004163/differences-b-iso-14443. [cit. 2023-12-30].

[28] *ESP32*. Online. Espressif.com. 2024. Dostupné z: https://www.espressif.com/en/products/socs/esp32. [cit. 2024-04-30].

[29] NEVEN 7 S.R.O. *PN532 NFC RFID V3 modul*. Online. NEVEN 7 S.R.O. NEVEN. 2024. Dostupné z: https://www.neven.cz/kategorie/elektronicke-soucastky/nfc-a-rfid/ctecky/pn532-nfc-rfid-v3-modul/. [cit. 2024-05-10].

[30] *PN532/C1: Near Field Communication (NFC) controller*, 2017. Online. In: NXP. NXP B.V. Dostupné z: https://www.nxp.com/docs/en/nxp/data-sheets/PN532_C1.pdf. [cit. 2023-12-30].

[31] MIGELLE. *How Do PoE Switches Supply Power for PoE Powered Devices?* Online. FS.COM INC. FS.com Europe. 2023. Dostupné z: https://community.fs.com/article/poe-power-supply-how-does-poe-switch-deliver-power-for-poe-devices.html. [cit. 2024-04-30].

[32] CALETKA, Ondřej. *PoEtický dům: napájíme po ethernetovém kabelu*. Online. ROOT.CZ. 2016. Dostupné z: https://www.root.cz/clanky/poeticky-dum-napajime-po-ethernetovem-kabelu/. [cit. 2024-05-01].

[33] *Power over Ethernet (PoE) — Demystifying Mode A and Mode B*. Online. PLANET TECHNOLOGY USA. Planet Technology USA. 2015. Dostupné z: https://planetechusa.com/power-over-ethernet-poe-demystifying-mode-a-and-mode-b/. [cit. 2024-05-01].

[34] JIRÁSEK, František. *Návrh a realizace univerzální NFC čtečky*. Online, Bakalářské práce, vedoucí Ondřej Baštán. Brno: VUT v Brně, 2022. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/142683. [cit. 2024-05-08].

[35] FCC PRŮMYSLOVÉ SYSTÉMY S.R.O. *MODBUS TCP*. Online. FCC PRŮMYSLOVÉ SYSTÉMY S.R.O. FCC průmyslové systémy. 2024. Dostupné z: https://www.fccps.cz/modbus-tcp. [cit. 2024-05-08].

[36] *W5500 Datasheet: Version 1.1.0*. Online. In: WIZNET CO., LTD. Docs.WIZnet. 2013. Dostupné z: https://docs.wiznet.io/Product/iEthernet/W5500/datasheet. [cit. 2024-05-10].

[37] *Lekce 3 - Spojový seznam*. Online. MICHÁLEK, Ondřej. ITNETWORK S.R.O. Itnetwork.cz. 2019, 2021-08-01. Dostupné z: https://www.itnetwork.cz/algoritmy/datove-struktury/spojovy-seznam. [cit. 2024-05-02].

[38] ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. *JTAG Debugging*. Online. ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. Espressif ESP-IDF Programming Guide. 2024. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-guides/jtag-debugging/index.html. [cit. 2024-05-10].

[39] ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. *Get Started*. Online. ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. ESP-IDF Programming Guide. 2023. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/v5.1.1/esp32/get-started/index.html. [cit. 2024-05-15].

[40] ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. *FreeRTOS (ESP-IDF)*. Online. ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. ESP-IDF Programming Guide. 2024. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/system/freertos.html. [cit. 2024-05-15].

[41] MICROSOFT CORPORATION. *Visual Studio Code*. Online. 2024. Dostupné z: https://code.visualstudio.com/. [cit. 2024-05-15].

[42] OPEN62541, 2023. *Open62541*. Online. Dostupné z: https://www.open62541.org/. [cit. 2023-12-30].

[43] *Pn532-esp-idf* [@thanhbinh89]. Online. 2019. Dostupné z: GitHub, https://github.com/thanhbinh89/pn532-esp-idf/tree/master. [cit. 2024-05-15].

[44] *OPC UA Server on ESP32* [@cmbahadir]. Online. 2021, 22.1.2023. Dostupné z: GitHub, https://github.com/cmbahadir/opcua-esp32/tree/master. [cit. 2024-05-15].

[45] *ESP32 Neopixel Driver* [@zorxx]. Online. 2024. Dostupné z: GitHub, https://github.com/zorxx/neopixel. [cit. 2024-05-15].

[46] *Libtelnet - TELNET protocol handling library* [@seanmiddleditch]. Online. 2020. Dostupné z: GitHub, https://github.com/seanmiddleditch/libtelnet. [cit. 2024-05-15].

[47] TATHAM, Simon; LANES, Alexandra; HARRIS, Ben a NEVINS, Jacob. *PuTTY: a free SSH and Telnet client*. Online. 2024, 2024-04-15. Dostupné z: `https://www.chiark.greenend.org.uk/~sgtatham/putty/`. [cit. 2024-05-16].

Seznam symbolů a zkratek

AAS	Asset Administration Shell
NFC	Near field communication
I4.0	Průmysl 4.0
ERP	Enterprise resource planning
MES	Manufacturing execution system
SCADA	Supervisory control and data acquisition
PLC	Programmable logic controller
HMI	Human machine interface
SCARA	Selective compliant articulated robot arm
QoS	Quality of Service
HW	Hardware
SW	Software
FW	Firmware
API	Application programming interface
OPC UA	OPC Unified Architecture
OPC	OLE for process
SOA	Service Oriented Architecture
URI	Uniform Resource Identifier
GUID	Globally Unique Identifier
NFC	Near field communication
RFID	Radio frequency identification
CRC	Cyclic redundancy check
ASK	Amplitude-shift keying
GPIO	General-purpose input/output

PWM	Pulse width modulation
PoE	Power over Ethernet
FTDI	Future technologies devices international
JTAG	Joint test action group
3DES	Triple data encryption standard
HSU	Hight speed UART
UART	Universal asynchronous receiver-transmitter
I²C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
ESP-IDF	ESP IoT Development Framework
DHCP	Dynamic Host Configuration Protocol
mDNS	multicast Domain Name System
Telnet	Teletype network
DPS	Deska plošného spoje
IoT	Internet of Things
LED	Light-Emitting Diode
RGB	Red-Green-Blue
LAN	Local Area Network
PD	Powered Device
PSE	Power Sourcing Equipment
ID	Identification
TCP	Transmission Control Protocol
IP	Internet Protocol
DC	Direct Current
USB	Universal Serial Bus

IDE	integrated development environment
SNTP	Simple Network Time Protocol
SSH	Secure Shell

Seznam příloh

A Obsah elektronické přílohy

/ ..... Kořenový adresář přiloženého archivu
- `Text_prace` ..... Adresář s textem bakalářské práce
- `ESP32_Firmware_ASS_Interpreter` ..... Adresář s FW AAS interpreteru
- `Windows_Server_OPCUA` ..... Adresář s projektem OPC UA serveru