VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

OPC UA KLIENT V JAZYKU PYTHON

OPC UA CLIENT IN PYTHON

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jan Zmrzlý
AUTHOR

VEDOUCÍ PRÁCE — Ing. et Ing. Stanislav Lang, Ph.D.
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Jan Zmrzlý**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**Ing. et Ing. Stanislav Lang, Ph.D.**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

OPC UA klient v jazyku Python

Stručná charakteristika problematiky úkolu:

Práce je zaměřena na možnosti využití jazyku Python pro datovou komunikaci prostřednictvím protokolu OPC UA. Je uvažována komunikace mezi PC a PLC, kdy na straně PLC je aktivován (vestavěný) OPC UA server, na straně PC je vytvořen OPC UA klient. Student v rámci práce nastuduje základní informace o protokolu OPC UA, dále naprogramuje jednoduchou testovací aplikaci na straně PLC a provede konfiguraci OPC UA serveru. Jádrem práce bude vytvoření jednoduchého OPC UA klienta pro PC v jazyku Python. Klient by měl být schopen připojit se PLC a číst hodnoty zpřístupněných proměnných.

Cíle bakalářské práce:

Nastudujte programovací jazyk Python.
Nastudujte komunikační protokol OPC UA, shrňte základní poznatky.
Nastudujte základy práce s PLC B&R Automation (včetně konfigurace OPC UA serveru).
Proveďte průzkum v oblasti dostupných softwarových knihoven pro Python umožňujících komunikaci přes protokol OPC UA.
Vytvořte vlastní aplikaci OPC UA klienta (pokud možno včetně uživatelsky přívětivého GUI).
Ověřte funkčnost OPC UA klienta na komunikaci s PLC B&R Automation (stačí v módu ArSim).
Stručně zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

OPC Foundation. OPC Unified Architecture: Interoperability for Industrie 4.0 and the Internet of Things [online]. [cit. 2021-10-18]. Dostupné z: https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN.pdf

B&R Automation [online]. [cit. 2021-10-18]. Dostupné z: http://www.br-automation.com/.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

OPC UA (*Unified Architecture*) je moderní průmyslový komunikačním standard. Jedná se o očekávanou evoluci navazující na úspěšný protokol OPC Classic. V dnešní době převážná část zařízení disponujících touto technologií je schopna mezi sebou komunikovat modelem server – klient. Smyslem této bakalářské práce je shrnutí poznatků o průmyslovém standardu OPC UA a vytvoření OPC UA klienta v programovacím jazyku Python. Funkčnost klienta je ověřena na komunikaci s PLC firmy B&R.

ABSTRACT

OPC UA (Unified Architecture) is a modern industry communication standard. This is an expected evolution following the successful OPC Classic protocol. Nowadays, the majority of devices with these technologies are able to communicate with the server model server – client. The purpose of this bachelor's thesis is to summarize the knowledge about the OPC UA industry standard and the creation of an OPC UA client in the Python programming language. The client's functionality is verified on communication with the B&R PLC.

KLÍČOVÁ SLOVA

OPC, OPC Unified Architecture, OPC Foundation, OPC UA klient, PLC, FreeOpcUa, OPC Classic, konfigurace OPC UA serveru na PLC B&R

KEYWORDS

OPC, OPC Unified Architecture, OPC Foundation, OPC UA client, PLC, FreeOpcUa, OPC Classic, configuration of OPC UA server on PLC B&R

ÚSTAV AUTOMATIZACE A INFORMATIKY

2022

BIBLIOGRAFICKÁ CITACE

ZMRZLÝ, Jan. *OPC UA klient v jazyku Python*. Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/139739. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Stanislav Lang.

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat svému vedoucímu Ing. et Ing. Stanislavu Langovi, Ph.D. za podnětné připomínky při řešení práce. Díky jeho podnětům se povedlo práci ve velké míře zkvalitnit. Dále také děkuji panu Ing. Marku Vidlákovi za cenné informace k naplnění praktického potenciálu mé práce.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 20. 5. 2022 ……

Jan Zmrzlý

OBSAH

1 ÚVOD

OPC UA (*Unified Architecture*) je moderní průmyslový komunikačním standard. Za jeho vznikem stojí potřeba propojit komponenty různých společností, aby jejich práce byla synergická. V 90. letech 20. století se formoval požadavek, který vystihuje následující příklad: „Výrobní linka je složená z řídících, ovládacích a měřících komponentů od různých výrobců. Nastává problém, že do zařízení budou integrovány různé komunikační technologie. Nutností tedy je vytvořit průmyslový komunikační standard, aby si komponenty v rámci montážní linky mohly *rozumět.*“

OPC UA navazuje na svého předchůdce OPC Classic. S touto změnou ovšem nepřichází velká revoluce, ale spíše se jedná o očekávanou evoluci. Výhodou se pak stává, že funkčnost OPC UA není založena pouze na OS Windows.

První kapitola je věnována programovacímu jazyku Python a dále je rámci práce shrnutý historický vývoj komunikačního protokolu OPC UA (včetně jeho předchůdce OPC Classic). Novější OPC UA pak integruje veškerou funkcionalitu svého předchůdce. Nechybí ani shrnutí problematiky modelování přenášených dat, adresování a zabezpečení komunikace.

V druhé části práce je popsán vývoj jednoduchého OPC UA klienta s příjemným grafickým rozhraním. Klient je vytvořen s pomocí programovacího jazyka Python a knihovny *FreeOpcUa*. Práce obsahuje rovněž ukázku velice jednoduchého klienta a komentář hlavních funkcí knihovny *FreeOpcUa*. Finálně je ověřena funkčnost klienta na komunikaci s PLC firmy B&R a firmy Beckhoff.

2 PŘEHLED SOUČASNÉHO STAVU POZNÁNÍ

2.1 Programovací jazyk Python

Python je vysokoúrovňový programovací jazyk, jenž byl navržen už v roce 1991 Guidem van Rossumem. Programovací jazyk Python patří v dnešní době k jazykům, jejichž využití velice roste na popularitě. V posledních letech se mluví o klíčovém programovacím jazyku, který je skvělý pro tvorbu aplikací využívajících strojové vidění, analýzu dat a programování umělé inteligence. Tento krátký výčet ukazuje, jak je Python všestranný, ale konkrétních případů využití je nespočet. Rozsáhlé užití Pythonu je dáno tím, že programovací jazyk nabízí velice dobrou dynamickou kontrolu datových typů a také podporuje různá programovací paradigmata. Jedná se jazyk poměrně jednoduchý na naučení a v roce 2018 se zařadil mezi nejoblíbenější programovací jazyky. [1, 2]

Vlastnosti jazyka Python

Jazyk bývá zařazován mezi takzvané skriptovací jazyky. Touto skutečností ale nesmíme být zmateni, jelikož možnosti využití jsou daleko větší. Mohou být pomocí něj vytvářeny plnohodnotné aplikace s grafickým rozhraním. Při tvorbě složitější aplikace lze využít objektově orientované programování, ale zároveň i programování funkcionální. Funkcionální programování může být použito pouze v omezené míře, ale lze jej vhodně kombinovat s objektově orientovaným programováním. Jednou z předních výhod Pythonu je, že samotný kód je velmi krátký a tím pádem i velice dobře čitelný. Následující ukázka znázorňuje srovnání jednoduchého kódu napsaného v jazyku Python a jazyku C, jehož hlavní funkcí je vynásobit dvě čísla. Na první pohled je jasné, že skript napsaný v Pythonu je o mnoho řádků kratší a minimálně pro začátečníka velmi dobře přehledný. [1, 2, 3]

- Ukázka programu pro násobení dvou čísel v jazyku C

```c
#include <stdio.h>

void nasobDveCisla(int cislo_1, int cislo_2){
    int nasobek;
    nasobek  = cislo_1 * cislo_2;
    printf("%i\n", nasobek);
}

int  main(void){
    int x;
    nasobDveCisla(3, 4);
}
```

- Ukázka programu pro násobení dvou čísel v jazyku Python

```
def nasob_dve_cisla(cislo_1, cislo_2):
    print(cislo_1 * cislo_2)
    return cislo_1 * cislo_2

x = nasob_dve_cisla(3, 4)
```

Další velkou předností jazyku je, že je velice dobře naučitelný. Například programovací jazyk ABC byl vytvořen pro výuku začínajících programátorů. Bohužel pro praxi a tvorbu složitějších aplikací bylo jeho použití ne zcela vhodné. Ovšem jazyk Python je vhodný pro začátečníky a zároveň pro použití v praxi při práci na rozsáhlejších aplikacích. Při jeho tvorbě byl kladen důraz na efektivitu práce. Uživatelská přívětivost je dána i faktem, že pro tvůrce kódu je dostupné nepřeberné množství knihoven a různých *frameworků*. Tyto knihovny umožňují velice rychlé řešení specifických úloh. Právě jedné z knihoven bylo využito při tvorbě OPC UA klienta. Využití knihovního modulu samotné programování velice usnadní, urychlí a výsledný kód je dobře přehledný. [1, 2, 3]

Po tomto shrnutí výhod programovacího jazyku Python se může zdát, že jazyk nemá žádnou chybu a opravdu jej lze použít na veškeré aplikace. Samozřejmě, že i Python není stoprocentně dokonalý. Poměrně nebezpečnou skutečností je, že operační systém může *„zneužít“* lokální proměnné a nedokumentované funkce pro spuštění příkazu. [1]

Při tvorbě jakéhokoliv kódu je nutné si uvědomit, pro koho je psaný. Pokud jej vytváříme pouze sami pro sebe a zvládneme se v něm orientovat i za několik let, pak není problém psát kód vlastním stylem. Ovšem daleko častější je případ, že skript po nás bude někdo číst, upravovat nebo rozšiřovat, a proto je na místě držet se daných pravidel. Přehlednosti docílíme tím, že budeme respektovat PEP. PEP je zkratka pro Python Enhancement Proposal. Tato série dokumentů shrnuje funkce programovacího jazyku a určuje aspekty jako je styl a design. Pro začínající programátory je nejvíce doporučované si prostudovat PEP 8. Ve své podstatě je PEP 8 dokument, který shrnuje pravidla psaní Python skriptu tak, aby byl co nejvíce přehledný. Otázkou může být, jestli opravdu PEP 8 musíme znát. Ano, určitě jej potřebujeme, protože jeho znalostí přispíváme souladu s Filosofií Pythonu – *The Zen of Python*. Její úplné znění je obsaženo v PEP 20. [4, 5]

K naprogramování OPC UA klienta bylo nutné zvolit vhodnou knihovnu. Požadavky na knihovnu nebyly příliš specifické. Knihovna musela fungovat ve spojení s PLC B&R Automation a také měla vhodně sloužit k naprogramování uživatelského rozhraní (GUI). Dalo by se očekávat, že vhodných knihoven bude mnoho a výběr bude složitý. Série požadavků by tedy musela být rozšířena, aby zůstala jen jedna ideální. Ovšem jedinou vhodnou knihovnou se stala *FreeOpcUa*, která je distribuována pod licencí LGPL. Jejím obsahem je sada funkcí pro programování OPC UA serveru i klienta. Knihovna vznikala už na Pythonu 2, ale starý kód byl postupně nahrazen novým, efektivnějším. Do *FreeOpcUa* je postupně implementovaná sada *opcua-asyncio*. Ta zaručuje snadnější čtení kódu, menší riziko chyb, a také může aplikaci zrychlovat. Celkově knihovna umožňuje vytvořit jednoduchý program serveru i klienta v několika

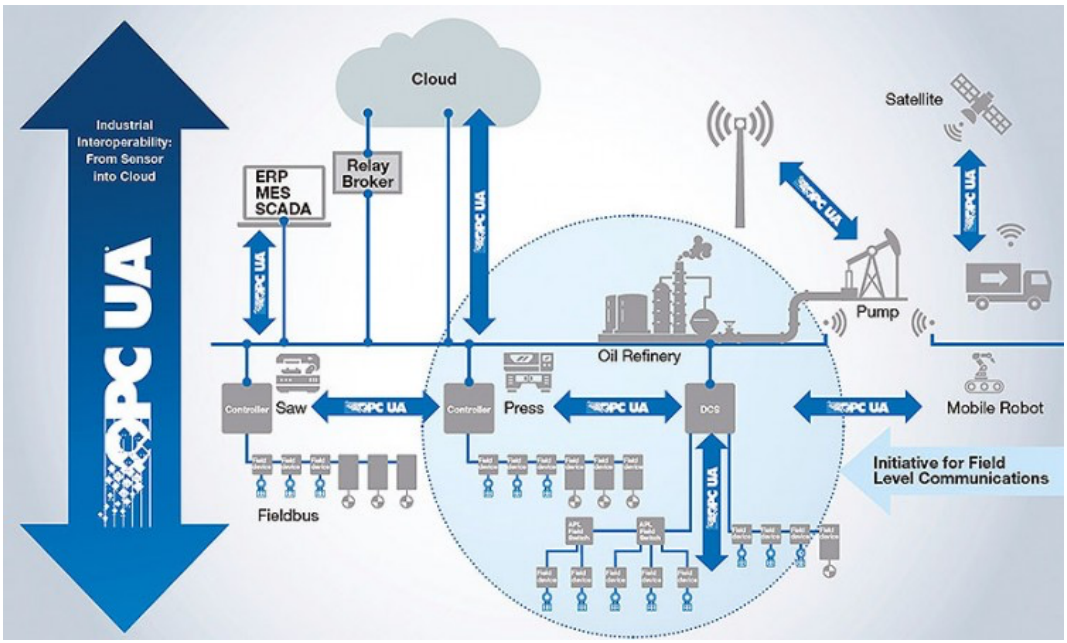
málo řádcích. Mimo bazální funkce obsahuje také jednoduchého klienta s grafickým rozhraním. Bohužel zmíněné GUI, které je součástí knihovny obsahuje chyby a v krajních případech nefunguje podle očekávání. Ovšem všechny skripty obsažené v knihovně se dají použít jako zdroj k naplnění cílů této bakalářské práce. Velkým benefitem je skutečnost, že součástí knihovny je funkční strom uzlů. [6]

2.2 Předchůdci OPC Unified Architecture

Historický vývoj

Vývoj OPC můžeme začít sledovat od roku 1990, kdy se poprvé na trhu objevil Windows 3.0. Nově v prostředí Windows (v porovnání s MS DOS) je možné mít spuštěných více aplikací současně a hlavním benefitem je, že operační systém umožňuje běžícím aplikacím výměnu dat za chodu. Tento mechanismus byl nazván *Dynamic Data Exchange* (DDE). Zanedlouho se však ukázalo, že DDE není příliš robustní a neexistuje pro něj síťová podpora. Hlavním nedostatkem se ale ukázala šířka pásma, která byla značně omezena. Následovalo značné množství pokusů tyto chyby opravit. Nejznámějším se stal bezpochyby počin Wonderware's InTouchTM SCADA software, který představil NetDDE^{TM}. Vývojářům se povedlo implementovat mechanismy síťového přenosu a zvětšit šířku pásma. Nevýhodou se ale ukázalo, že software je proprietární, a tedy za něj bylo nutné platit vývojářům. Tato skutečnost způsobila, že se nikdy nestal průmyslovým standardem. Když byl OLE 2.0 uveden na trh v roce 1992, bylo zřejmé, že nakonec nahradí téměř všechna použití DDE. Byl flexibilnější, robustnější a využíval efektivnější transportní mechanismy. Současně se začala formovat skupina vývojářů z oblastí průmyslového řízení a získávání dat v centrále Microsoftu v Redmondu. Tato skupina nesla označení WinSEM (*Windows in Science, Engineering nad Manufacturing*). Na začátku roku 1994 se objevuje poptávka po standardizaci rozhraní mezi jádrem SCADA a ovladači zařízení, které jsou odpovědné za získávání dat. Touto myšlenkou se zabývala společnost US Data, která v roce 1995 předložila nejzajímavější realizovatelný návrh. Za zmínku stojí, že už tehdy v něm byla specifikována velká řada klíčových OPC konceptů. Následně se formovala menší a specifičtěji zaměřená skupina (OPC TaskForce). Se vznikem této skupiny spojujeme počátky OPC. Původní členové OPC TaskForce byli Fisher-Rosemount (nyní Emerson Process Management), Intellution (nyní součást GE Fanuc), Intuitive Technology (nyní součást Wizcon Systems), OPTO 22 a Rockwell Software. Microsoft měl být zapojen do podpůrné a konzultační role. První pracovní verze byla spuštěna v prosinci roku 1995 a setkala se s pozitivním ohlasem. Konečně se dostáváme k vydání OPC verze 1.0 (29. srpna 1996) a opravené verzi 1.0 A (1997), kterou známe dnes spíše jako OPC Data Access. Po vydání prvních verzí bylo rozhodnuto, že specifikace OPC bude pod správou nezávislé neziskové organizace OPC Foundation. Organizace se poprvé přestavila na výstavě ISA Show v Chicagu v roce 1996, kde předvedla ukázky OPC serverů od různých společností. Další ukázky následovaly na technických veletrzích po celém světě. Netrvalo dlouho a na trh začaly

přicházet první komerční produkty. Stalo se tak už koncem roku 1996 a díky široké podpoře OPC bylo roku 1998 potvrzeno, že se skutečně stane průmyslovým standardem.



Obr. 1: Možnosti využití OPC UA k roku 2021 [7]

Dále bylo přidáno automatizační rozhraní a uživatelské rozhraní s označením *Alarms and Events* verze 1.0. Historickým rokem se pro OPC Foundation stal rok 2000, protože byly vydány nové specifikace *Historical Data Access, Batch and Security*. Posledním historickým bodem je rok 2001, kdy byl OPC Foundation vydán program *Compliance Testing* pro servery OPC Data Access. Co přinesly další požadavky na standardizaci, je patrné z Obr. 1: [8]

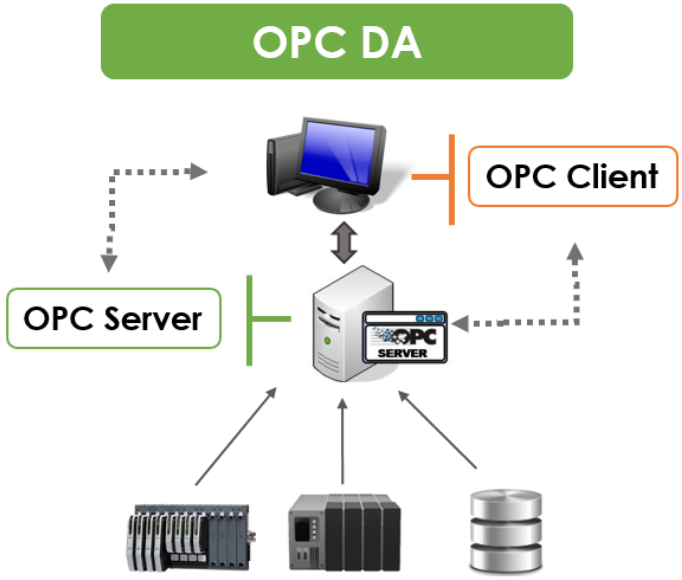
OPC Classic

OPC Classic je specifikace založená na technologii COM/DCOM od společnosti Microsoft pro výměnu dat mezi softwarovými komponenty. Znamená to, že funguje pouze na zařízeních s OS Windows. Zároveň tato technologie umožňuje komunikaci mezi aplikacemi nebo částmi aplikace. Postupně byla ale nahrazována souborem technologií .NET. OPC Foundation vydala řadu softwarových rozhraní pro standardizaci přenosu informací od úrovně procesní až po úroveň řídící. Nejčastější využití toto rozhraní našlo v průmyslové automatizaci u HMI (*human–machine interface*) a SCADA (*Supervisory Control and Data Acquisition*) systémů. Tyto systémy zpracovávají aktuální data ze zařízení, poskytují je spolu s uloženými daty správě aplikace (operátorovi). Zahrnutými rozhraními jsou OPC Data Access (OPC DA), OPC Alarms & Events (OPC AE) a OPC Historical Data Access (OPC HDA). Tato rozhraní používají pro komunikaci zavedený

model klient — server. Model je velice výhodný z toho hlediska, že k jednomu serveru můžeme připojit více klientů a naopak. [9, 10]

OPC Data Access

OPC DA je stále nejpoužívanější průmyslovou specifikací. Primární funkcí je čtení, přepis a sledování proměnných. Hlavní motivací pro vytvoření OPC DA bylo přesouvání dat z PLC, DCS do HMI a dalších zobrazovacích systémů. [9]



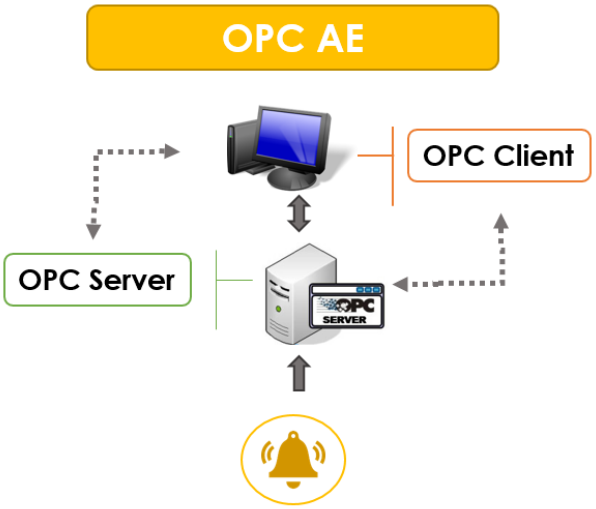
Obr. 2: Specifikace OPC Data Access je zabudovaná v 99 % zařízeních s OPC [11]

Nejpoužívanější průmyslovou specifikací se stalo z důvodu, že je implementován v 99% technologických produktů, které v současnosti využívají technologickou specifikaci OPC. Pomocí klienta instalovaného v zobrazovacím zařízení je možné číst konkrétní proměnné. Po navázání spojení server klientovi udá způsob, jak procházet hierarchií adresního prostoru (*AddressSpace hierarchy*), pomocí kterého pak klient nachází jednotlivé položky. Pro položky jsou pak dále definovány jejich vlastnosti, například datový typ a přístupová práva. Schopnosti specifikace jsou patrné z Obr. 2: [9]

Klient seskupuje jednotlivé položky do skupin (*OPCGroup*) se stejným časem aktualizace. Předcházející Obr. 2: názorně ukazuje rozdělení jednotlivých objektů vytvořených na serveru OPC klientem. Klient v časových cyklech kontroluje pouze zvolené proměnné. Po každém cyklu získá ze serveru pouze proměnné, které byly změněny. [9]

OPC Alarm and Events

Rozhraní OPC A&E umožňuje přijímat hlášení (*alarm*) a událostí (*events*). *Events* jsou notifikace, které pouze informují klienta, že v systému nastala specifická událost. *Alarm* jsou upozornění, která informují klienta o změně podmínek v sledovaném systému. Grafické zobrazení těchto funkcí je na Obr. 3: Jako specifickou podmínku můžeme například uvažovat změnu teploty v indukční peci. V tomto případě dojde k upozornění, pokud teplota příliš klesne nebo naopak příliš stoupne. Jednotlivé alarmy často požadují jejich potvrzení obsluhou systému. Potvrzení je možné vystavit prostřednictvím rozhraní OPC A&E. Stejně jako u OPC DA je nutné, aby se klient připojil k serveru, ze kterého pak následně získává všechna upozornění poskytnutá serverem. Samozřejmě všechna upozornění nejsou nezbytně nutná a mohou být klientem filtrována. [9]



Obr. 3: Funkcionalita OPC Alarm and Events [11]

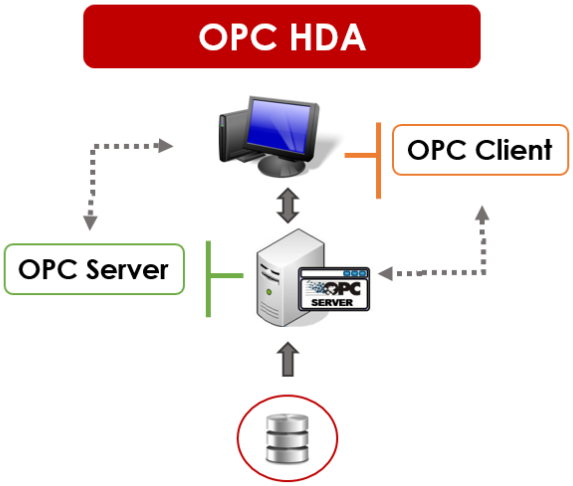
Klient OPC A&E se podobně jako v případě OPC DA připojí k serveru a může začít *alarm* nebo *event* odebírat. Tímto krokem se vytvoří objekt nazvaný *OPCEventSubscription*. V dalším kroku pomocí *OPCEventSubscription* začne klient získávat zprávy o událostech. Zmíněné filtrování může být nastaveno specificky pro každý odběr. Rozdílem oproti OPC DA je, že nejde o získávání konkrétních informací jako jsou proměnné určené ke čtení. Klientovi jsou poskytovány informace o všech konaných procesech a opět je vhodné je dále třídit například podle typu události, zdroje události nebo jejich priority. [9]

OPC Historical Data Access

V úvodu další specifikace využijeme opět pro srovnání OPC Data Acess, kde dochází k získávání dat ze serveru v reálném čase. Oproti tomu OPC Historical Data Acess umožňuje přístup k datům již uloženým. Využití specifikace je velmi rozsáhlé, od práce s jednoduchými sériovými záznamy po komplexní SCADA systémy. [9]

Klient je připojen k HDA serveru, kde je vytvořen objekt nazvaný *OPCHDAServer*. Tento objekt poskytuje uživateli všechny metody pro čtení a aktualizaci záznamů. Dalším vytvořeným objektem je *OPCHDABrowser*, který slouží k procházení adresního prostoru HDA serveru. Způsob fungování je patrný z Obr. 4: [9]

Specifikace převážně slouží ke čtení záznamů, a to třemi různými způsoby. Prvním způsobem je čtení surových dat z archivů. Podmínky získání dat jsou určeny klientem, který pak získává konkrétní počet proměnných z určitého časového rozmezí. Druhým způsobem je čtení jedné nebo více proměnných specifikovaných časovou známkou (*timestamps*). Poslední možnou cestou je získávání proměnných (*aggregate values*) z databází záznamů. Proměnné pak vždy obsahují související kvalitu a časovou známku. Mimo čtení uložených záznamů OPC HAD také umožňuje data vkládat, nahrazovat a odstraňovat. [9]



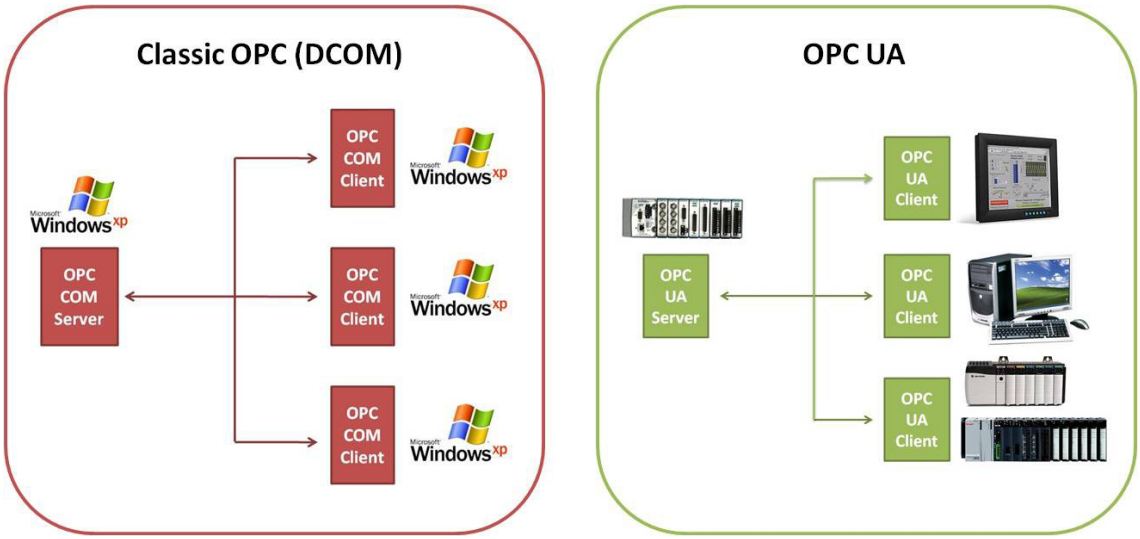
Obr. 4: Funkcionalita OPC Historical Data Access [11]

2.3 Motivace pro vytvoření OPC UA

Hlavní motivací k vytvoření OPC Unified Architecture bylo nahrazení všech stávajících specifikacích založených na COM bez ztráty jakýchkoliv funkcionalit nebo výkonu. Názorné schéma je uvedeno na Obr. 5: Jde o způsob přenosu dat založený na komunikačních standardech jako jsou TCP/IP, HTTP a SOAP. Velkou výhodou je, že OPC UA může fungovat na jiných platformách než OS Windows a je tedy možné ho použít v PLC automatech a jiných zařízeních. OPC UA integruje všechny své předchůdce, a umožňuje tedy přenos procesních dat, alarmů a uložených dat. [9, 12]

U OPC Classic bylo modelovaní dat velice omezené a bylo nutné jej nahradit objektově orientovaným modelem. Model musel obsahovat rozšířený typový systém, také musel být schopný nabízet metainformace a popisovat komplexnější systémy. Jak je

zřejmé, tak vzrostl požadavek na zlepšení modelování dat, ale zároveň zůstalo důležité zachování podpory jednodušších modelů. [9]

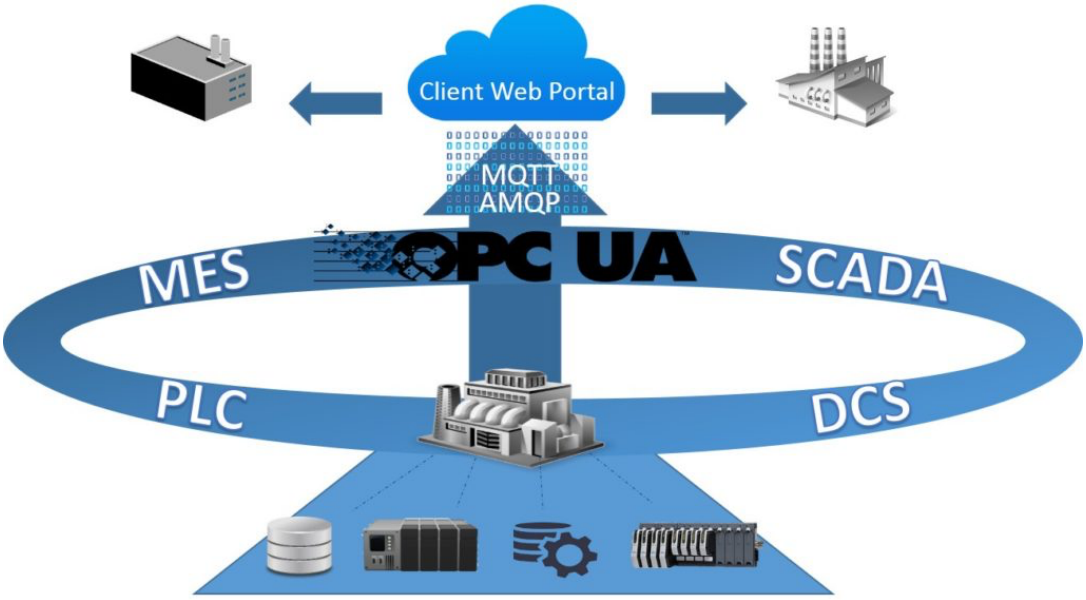


Obr. 5: OPC UA nemusí striktně používat Windows, komunikuje s vestavěnými servery [12]

Za tímto účelem se formovala skupina čtyřiceti zástupců průmyslových firem, která definovala základní požadavky a způsoby využití OPC Unified Architecture. Tato skupina nebyla složena pouze z OPC členů, ale participovaly v ní i organizace jako IEC a ISA, které taktéž OPC využívaly. OPC Foundation si v počátku definovala hlavní cíle a to: „Jak popsat a přenášet data spolupracujících organizací.“ a „Jaká data budou popisována a přenášena budoucím modelem.“. Posledním důležitým cílem se stalo určení způsobu, jak přejít z úspěšné standardizace Classic OPC, do které členové skupiny dříve investovali nemalé částky, na novou průmyslovou standardizaci OPC UA. [9]

2.4 Komunikační protokol OPC UA

Základem specifikace je komunikace a výměna dat mezi klientem a serverem založená na mapování a navazování vzájemných spojení, zabezpečení komunikace a struktuře poslaných dat. Zamýšlený cíl využití je velice dobře zřejmý z Obr. 6: Protokol je definován jako službově orientovaná architektura *Service Oriented Architecture* (SOA). Prakticky to znamená, že klient posílá serveru dotazy a ten na ně adekvátně reaguje — odpovídá. Jakmile je navázáno fyzické spojení vzniká zabezpečený kanál (*SecuredChanel*) a relace (*Session*). Aby se mohl klient dotazovat serveru, je nutné mít *SecuredChanel* aktivní. Tyto části jsou dohromady označovány jako komunikační zásobník (*Stack*). Adresní prostor je pak popsaný pomocí uzlů. [13]



Obr. 6: OPC UA integruje všechny funkcionality OPC Classic [11]

Komunikace a navázání spojení

Komunikace je zaručena prostřednictvím zásobníku nazývaným *Stack*, který se skládá ze tří vrstev. Těmito vrstvami jsou:

- Transportní vrstva (*TransportLayer)*
- Komunikační vrstva (*Secure Chanel Layer)*
- Aplikační vrstva (*Application Layer)*

Transportní vrstva – jedná se o úplně nejzákladnější vrstvu, která obsahuje komunikační protokol. Vytváří se ihned po navázání spojení mezi serverem a klientem. Její součástí jsou šifrovací a ověřovací mechanismy, které zabrání čtení šifrovaných zpráv třetí stranou. Mezi podporované protokoly patří: HTTP/SOAP, HTTPS a TCP/IP. [13]

Komunikační vrstva – stejně jako transportní vrstva se vytváří ihned po spojení serveru a klienta a představuje jejich zabezpečený kanál *SecuredChanel*. Každé připojení musí mít svůj zabezpečený kanál, který se od ostatních odlišuje identifikátorem kanálu (*Secure-ChanelId*) a bezpečnostní známkou (*SecurityToken*). Bezpečnostní známka se vytváří jako symetrický klíč, a díky této vlastnosti má nižší nároky na výpočetní výkon. Pro zajištění spojení je nutné bezpečnostní známku obnovovat, protože její životnost je, na rozdíl od identifikátoru, dočasná. [13]

Aplikační vrstva – jedná se o vrstvu obsahující relační vrstvu (Session), ve které probíhají veškeré volání a zpracovávaní požadavků. V aplikační vrstvě předává klient serveru svoje

přihlašovací údaje. Server po získání těchto údajů označí klienta jako specifického uživatele. OPC UA nemá definovanou podobu uživatelů. Definovaný je pouze způsob, jak klient předává své přihlašovací údaje serveru. Nutností je relaci udržovat pořád aktivní v podobě požadavků klienta na server, jelikož relace zaniká po určité době nečinnosti. [13]

Pro navázaní spojení stačí implementovat pouze několik málo funkcí. Jedná se o velkou výhodu, kdy na řešení jednoduchých úkolů nemusíme aplikovat robustní software. Ovšem základní funkce můžeme velmi efektivně rozšiřovat. Může nastat situace, kdy pro řešení konkrétního problému potřebujeme různé sady funkcí. To řeší možnost vytváření profilů, ve kterých jsou obsaženy různé seznamy funkcionalit. Součástí profilů bývají nejčastěji sady služeb, využité kódování, informace o zabezpečení a další prvky OPC UA. [9, 13]

Modelování a adresování

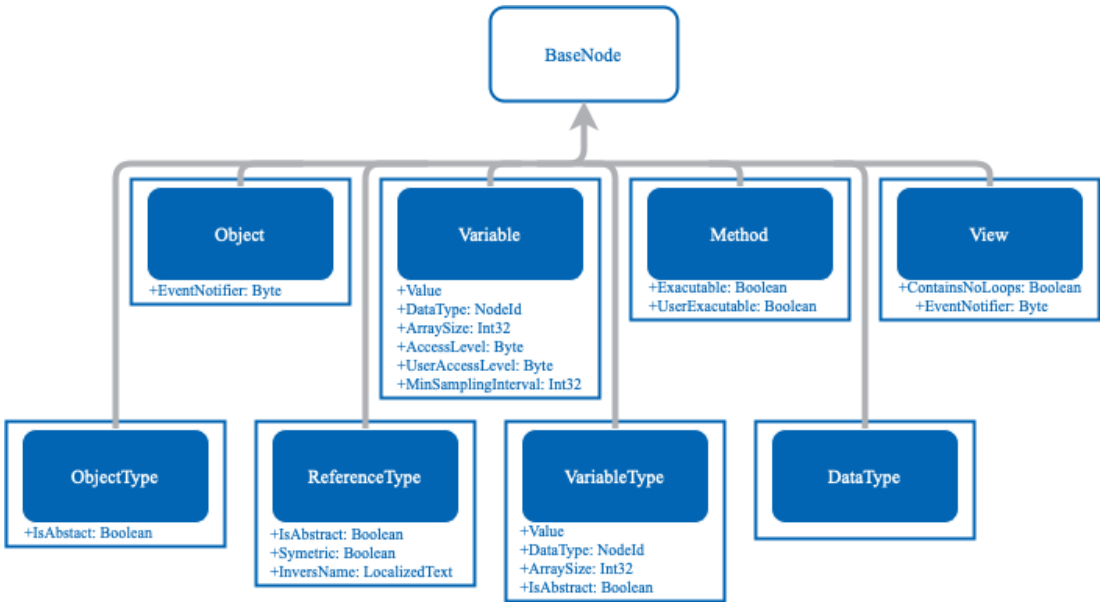
Ve srovnání s Classic OPC se možnosti modelování dat výrazně změnily. OPC UA používá nejmodernější, bezpečné a spolehlivé technologie, které nejsou závislé na platformě. Díky vylepšením je klient schopný lépe zpracovávat informace, které dostává. U OPC Classic byla proměnná doplněna pouze štítkem (*tag name*) a o některé základní informace. Zatímco navazující standardizace získává data doplněná o daleko více informací. Patří mezi ně i například typ senzoru, kterým byla veličina naměřena [9].

Základní myšlenkou modelování je propojení uzlů pomocí referencí. Jednotlivé uzly pak patří do různých tříd uzlů *NodeClasses*. Uzly si nesou určité znaky, které nelze rozšiřovat. Může nastat situace, kdy je nutnost popsat je dalším znakem. Pak mohou být informace o uzlu rozšířeny pomocí specifických vlastností (*properties*). Tyto vlastnosti mohou například obsahovat informace o jednotce naměřené veličiny. Co se týče referencí, tak požadavky na jejich integritu jsou velmi nízké. Mohou nastat případy, kdy reference ukazují na neexistující uzly, nebo dokonce na servery, které nejsou pro klienta dostupné. Z referencí se také mohou stát smyčky. Ovšem klienti se musí vypořádat se všemi nástrahami způsobenými referencemi. [9]

Pokud by modelování dat bylo v celé podobě ponecháno na jednotlivých prodejcích, nastal by problém, že stejná data by byla modelována odlišnými způsoby. Tato skutečnost by s sebou přinesla mnohá úskalí spojená s komunikací mezi serverem a klientem. OPC Foundation vyřešila tuto komplikaci tak, že do UA zařízení je na začátku implementován pouze základní model, který může být pak následně rozšířen prodejcem. Do tohoto modelu jsou zahrnuté například typy, mapování, dotazy a odpovědi služeb. Tato data jsou serveru poskytována stejně jako v případě modelu základního. [9]

Data se nachází v adresním prostoru (*AddressSpace*). Jak už bylo zmíněno, celý prostor je vytvořený z uzlů, které jsou mezi sebou propojené referencemi. V závislosti napojení referencí na určité skupiny uzlů vznikají složitější objekty. Každý z těchto uzlů je pak definovaný základními atributy, které jsou společné pro všechny uzly. Jsou

to: Identifikátor uzlu (*NodeId*), prohledávané jméno (*BrowserName*), zobrazovací jméno (*DisplayName*) a třída uzlu (*NodeClass*). V rámci adresního prostoru se uzly třídí do osmi základních *NodeClasses*. Způsob dělení a atributy jednotlivých tříd jsou patrné z Obr. 7: [9, 13]



Obr. 7: Dělení uzlů do osmi základních tříd (*NodeClasses*) [14]

Zabezpečení

Velkou předností OPC UA je možnost zabezpečení komunikace. Zabezpečení je velice intenzivně vyvíjeno tak, aby bylo, co nejodolnější nástrahám třetích stran. Zabezpečování komunikace probíhá ve dvou úrovních. V první úrovni probíhá šifrování a podepisování zpráv. Jedná se o vnější zabezpečení. V druhé úrovni je zaznamenávána aktivita serveru a následně dochází k uzavíraní přebytečných a neužitečných spojení. Všichni účastníci komunikace musí mít svůj vlastní certifikát. Tento aplikační certifikát přímo definuje aplikaci nebo stroj. Z popisu je zřejmé, že vnitřní zabezpečení není přímo vycházející z protokolu OPC UA, ovšem u vnějšího se jedná o opak, a tedy jde o přední výhodu této průmyslové standardizace. [9, 13]

Vnější zabezpečení dále ještě můžeme rozdělit na dvě dílčí vrstvy. Ve vrstvě označené jako *SecurityMode* probíhá ověřování aplikačních certifikátů. Jak už bylo zmíněno, jsou tyto certifikáty unikátní. Nejjednodušší spojení je realizováno jako *SecurityMode None*. Znamená to, že na obou stranách jsou akceptovatelné jakékoliv ověřovací certifikáty, ke kterým jsou vždy přiloženy informace o druhé straně. Dalším *SecurityMode* je *Sign*. V tomto případě musí být shodné certifikáty na obou stranách, aby komunikace mohla probíhat. Může nastat i případ, kdy je na jedné straně poskytnut certifikát důvěryhodnou certifikační autoritou. Posledním možným modem je *SignEndEncrypted*, kde je Sign rozšířen o dodatečné šifrování. Druhá vrstva se pak specializuje na způsoby šifrování. Opět je zde možnost, kdy k šifrování nedochází, a pak tedy je *SecurityPolicy None*. Mezi další šifrovací algoritmy pak patří: Basic128Rsa15,

Basic256 a Basic265Sha. Pro správné fungování komunikace klienta se serverem, je nutné mít na každé straně alespoň jednu stejnou *SecurityPolicy*. V krajních případech může být zabezpečení nakonfigurováno tak, aby žádné nebylo přijímané. Takovéto nastavení by bylo vhodné při testování prvotních spojení, ale v dalších aplikacích je vhodné se jej vyvarovat. Jedním z benefitů pramenících z průmyslové standardizace OPC UA je právě možnost zabezpečené komunikace. OPC Foundation doporučuje při komunikaci používat zabezpečení typu oboustranné autentizace. V tomto případě klient i server umožňují připojení pouze důvěryhodným partnerům s tím, že je vyžadováno speciální nastavení na straně klienta i serveru. [9, 13]

OPC UA, Modbus a EtherCAT

Komunikační standard OPC UA Unified Architecture v dnešní době pořád patří mezi čerstvé technologie na poli M2M (*machine-to-machine*) komunikace. Nejedná se o úplnou revoluci ze strany OPC Foundation, ale spíše očekanou evoluci. V porovnání s dalšími komunikačními protokoly přináší mnoho zdokonalení, ale ta se neobejdou bez určitých úskalí. Pro porovnání s otevřeným protokolem Modbus, který přenáší informace pomocí sběrnic a je často používaný ke stejnému účelu, nabízí zabezpečení komunikace. Ze zmíněného protokolu bychom tedy mohli odebírat data pouze se znalostí IP adresy a konkrétního portu. Velkou předností OPC UA je kompatibilnost zařízení od různých výrobců. Ale i díky této skutečnosti se nejedná o nejrozšířenější protokol. Nejpoužívanějším protokolem průmyslové komunikace je EtherCAT Automation Protocol (EAP). EtherCAT zaručuje *hard-real time* komunikaci mezi řídícím systémem a prvkem řízeným, velmi efektivně využívá šířku pásma a poskytuje technologie pro dosažení vysokého stupně synchronizace mezi zařízeními připojenými do sítě. Pro průmyslové využití není nezbytně nutné volit pouze jeden komunikační protokol. Logické je, že pro různá použití se hodí různé specifikace. EAP nemůžeme plnohodnotně nahradit OPC UA a naopak, velkým benefitem je pak, když na sebe dokážou vhodně navazovat. [15]

3 OCP UA KLIENT V JAZYKU PYTHON

Hlavním cílem bakalářské práce je vytvořit OPC UA klienta v jazyku Python. V dřívějších letech byla na podobné téma řešena práce pana Staňka, který se zabýval tvorbou klienta stejné průmyslové standardizace. Pan Staněk neměl přímo určený programovacího jazyk, ale byl zde kladen důraz na specifickou funkcionalitu. V jeho práci je dobře provedený průzkum programovacích jazyků, které poskytují frameworky pro tvorbu OPC UA serverů a klientů.

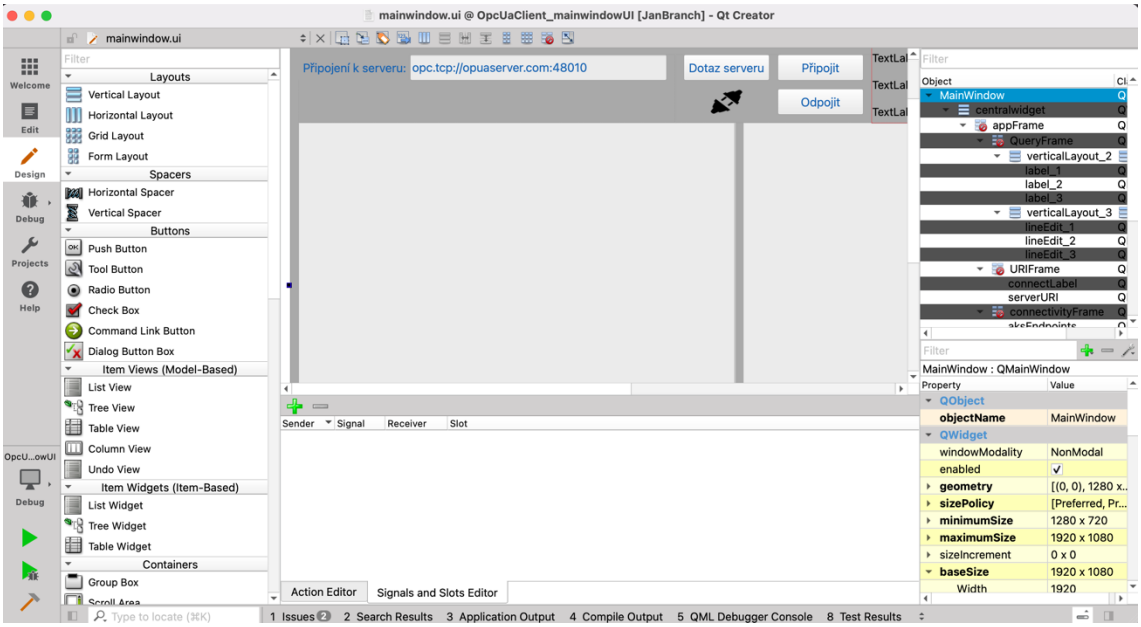
Příprava pro tvorbu UA klienta v jazyku Python je znázorněna na Obr. 8: Volba jazyka pro tuto práci byla dána. Dalším logickým úkolem byla vhodná volba knihovny, která práci výrazně usnadní. Na internetu je dostupných mnoho těchto knihoven, ale ne všechny působí dojmem, že práce s nimi bude příjemná a funkcionalita je otestována. V současnosti je nejvíce používanou knihovnou FreeOpcUa, která je volně dostupná pod licencí LGPL. Bližší informace o této knihovně jsou popsány v kapitole 2.1. Tato knihovna splňovala všechny požadavky pramenící ze zadání práce. Vhodné je zmínit, že tato knihovna obsahuje velké množství ukázek klientů i serverů různých složitostí. Dobře může tedy pomoci i úplným začátečníkům, kteří začínají teprve pronikat do produktů OPC Foundation.

Předposledním přípravným úkolem byla volba vývojového prostředí. Existuje nepřeberné množství vývojových prostředí, a proto je vhodné volit nejlépe to, které je nám doporučeno, nebo s ním máme předchozí zkušenosti. Jelikož se studenti programu Základy strojního inženýrství setkají pouze s prostředím Matlab, bylo na místě nechat si poradit od spolužáků z jiných oborů nebo fakult. Po srovnání několika málo vývojových prostředích bylo určeno, že práce bude vznikat ve Visual Studio Code. Práce s ním je rychlá a efektivní. Sám kontroluje syntaktické chyby už při psaní kódu a velice jednoduše může být propojen s *github.com*, kde může probíhat finalizace napsaných skriptů.

Obr. 8: Přípravná fáze tvorby UA klienta

Posledním úkolem, než mohla samotná práce začít vznikat, byla volba vhodné knihovny pro vytvoření přívětivého grafického rozhraní (*grafic user interface* – GUI). Jak už to často v případě Pythonu bývá, pro řešení se nabízí mnoho použitelných knihoven. Pro tvorbu GUI jsou pak nejčastěji používané Tkinter, Kivy, wxPython, Libavg a PyQt5. Důležitým faktorem se tedy stala skutečnost, aby pro danou knihovnu bylo dostupné velké množství video návodů a dokumentace. Po krátké rešerši bylo zjištěno, že nejlépe zpracované knihovny jsou Tkinter a PyQt5. Volba tedy zůstala mezi těmito dvěma. Finálně ale zvítězila PyQt5 kvůli dvěma hlavním důvodům. Prvním důvodem bylo, že knihovna dává možnost vznikat hezčím grafickým rozhraním, než je tomu v případě Tkinteru. Je předpokládáno, že práce nebude omezena pouze bakalářským

studiem, a proto byla zvolena knihovna, se kterou se bude snadno pracovat v budoucnu a GUI bude vypadat moderně. Druhým důvodem bylo, že k práci s PyQt5 lze využít aplikace Qt Creator, Obr. 9: V této aplikaci mohou být přímo vytvářeny hlavní okna, různé widgety, tabulky a kontextová menu.



Obr. 9: Ukázka vývoje UA klienta v prostředí Qt Creator

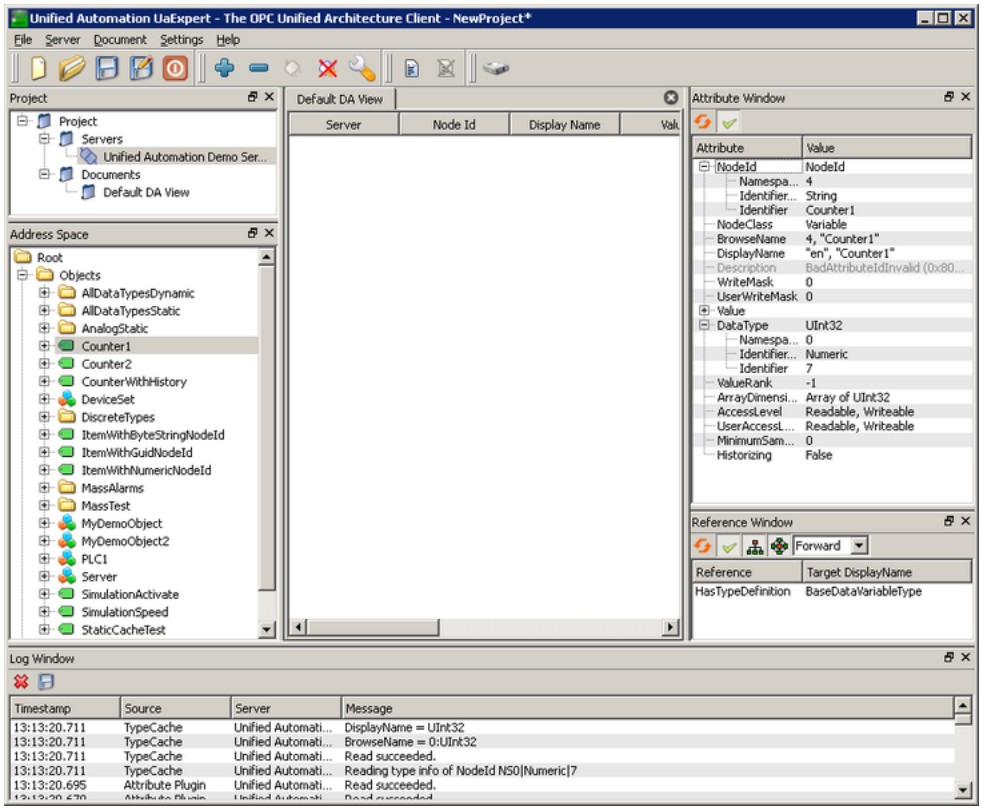
3.1 Požadavky na OPC UA klienta

Ve druhé kapitole věnované současnému přehledu poznání byly získány pevné základy pro stavbu UA klienta. Otázkou je proč samotného klienta vytvořit a jaký problém se vytvořením v průmyslové praxi vyřeší. Je tedy vhodné určit si fiktivní situaci, kdy bychom jako zákazníci takového klienta byli ochotni koupit.

Modelová situace, může vypadat následujícím způsobem:

„Firma specializující se na výrobu pneumatických pohonů musí každé zařízení testovat. Toto testování probíhá na testovacích strojích, které jsou na konci montážních linek. Pro jednoduchost je ve strojírenské firmě pouze jedna linka s jedním testovacím zařízením na konci a zkouší se pouze jeden typ pneumatických pohonů. Test je obsluhován operátorem, který sekvenci zkoušek řídí pomocí tabletu. Jednotlivé úkony stroje jsou pak řízeny programovatelným automatem (Programmable Logic Controller PLC). Operátor díky propracovanému HMI má velmi dobrý přehled o stavech, které v pneumatickém zařízení mohou nastat. Ovšem data stroje nejsou nikde zobrazována a ukládána. Tyto informace by pak bylo vhodné poskytovat údržbě stroje nebo inženýrům optimalizujícím jeho chod. V PLC je připojeno do podnikové sítě a je v něm zabudovaný OPC server, který si žádá aktivaci a nakonfigurování.“

Požadavky vyplývající z modelové situace by měl být schopný vyřešit dobře nakonfigurovaný UA klient. Jeho funkcionalita vůbec nemusí být robustní a lze ji přirovnat jednoúčelovému stroji. Jedná se o zařízení, které celý svůj služební život vykonává stejnou sérii jednoduchých úkonů. Tímto způsobem bude koncipován i vytvářený UA klient. Prvním požadavkem tedy je, aby se klient připojil do podnikové sítě, kde zkusí prověřit různé možnosti spojení. Prozkoumá různé *endpoints* serveru. Musí zaručeně být navázáno spojení. Jakmile bude klient schopný orientace na firemní síti je nutné navázat spojení s konkrétním serverem. Připojení k serveru musí provést uživatel vybráním správné IP adresy a finálním potvrzením tlačítka. Sotva bude klient připojený k serveru, sestaví se strom uzlů, který bude reprezentovat hierarchii *AddressSpace*. Procházení stromem uzlů je pak opět na uživateli, kterému by mělo být známo, jakou proměnnou chce odebírat. Po nalezení proměnné ji uživatel přidá do tabulky odebíraných proměnných. V této tabulce už bude reprezentována svými atributy. Poslední funkcionalitou pak bude ukládání zvolené proměnné do databáze.



Obr. 10: Grafické rozhraní klienta UaExpert [16]

Takto by mohl fungovat jednoduše vytvářený UA klient. Požadavky na něj nejsou rozsáhlé, a proto aplikace klientů, které jsou dostupné volně na internetu, jsou příliš robustní k tomuto jednoduchému použití. Například volně dostupná verze UaExpert nabízí tyto možnosti:

- Sledování proměnných
- Sledování *Alarms&Conditions*

- Vykreslení trendu ze sledovaných dat
- Zobrazení diagnostiky serveru
- Jednoduchý Datalogger CSV Plugin
- Výkonnostní Plugin
- GDS Push-Model Plugin
- XMLNodeSet-Export View

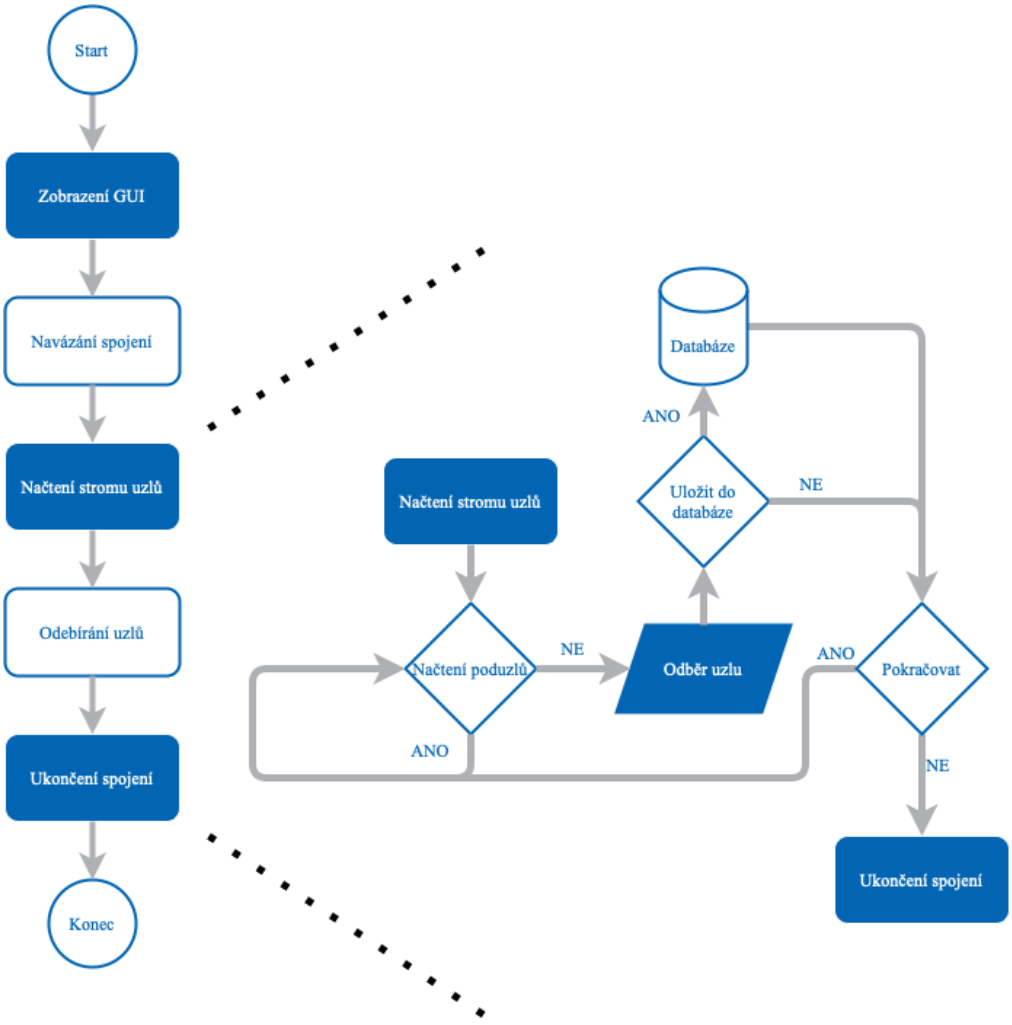
Grafické rozhraní UaExpert je patrné z Obr. 10: Je zcela zřejmé, že pro uvedenou modelovou situaci by bylo toto řešení poněkud těžkopádné. Nevýhodou pak je, že UaExpert nenabízí možnost ukládání dat do databáze. V jeho schopnostech je pak pouze vykreslení grafu, který bude zachycovat určitý historický trend. [16]

Upřesněné požadavky UA klienta

Nyní je vhodné více specifikovat požadavky na vytvářeného OPC UA klienta a nastínit způsob jeho fungování. Před určením funkcí byli otestováni dva různí klienti volně dostupní na internetu. Prvním testovaným klientem byl zmíněný UA Expert, jehož přednosti a nevýhody byly zmíněny již dříve. Druhou testovanou aplikací byl klient Integration Object. Tento klient má daleko moderněji zpracované GUI a jeho použití se jeví jako uživatelky přívětivější. Ovšem zprvu měl problém navázat spojení s PLC, a bylo tedy nutné nastavit certifikáty zabezpečení na straně klienta. Pro osobu neznalou průmyslové standardizace OPC UA se pak tento klient může jevit jako nepoužitelný. Vytvářený klient bude přímočarý a jednoduchý na použití.

Po spuštění aplikace je nechána prvotní interakce na uživateli. Ten vybírá ze dvou možností. První možností je, že uživatel zná TCP/IP adresu a port OPC UA serveru připojeného do podnikové sítě. Pak tedy může vepsat zmiňovanou adresu a kliknout na tlačítko *Připojit*. Druhou možností je, že se uživatel chce přímo dotázat, jaké servery na síti jsou mu dostupné. Klikne tedy na tlačítko *Dotaz sítě*. Po této volbě jsou mu nabídnuty dostupné severy. Při nejběžnějším použití je předpokládáno, že uživatel bude znát adresu a přímo se připojí k serveru. Po navázaní spojení je udržována aktivní relace. Řešení udržení aktivní relace je součástí knihovny *FreeOpcUa*. Ihned po sestavení relace jsou uživateli zobrazeny hierarchicky seřazené uzly *Nodes*. Další interakce je opět na uživateli. Běžný způsob užití předpokládá, že je uživatel od správce serveru seznámený s názvy proměnných *BrowserName* nebo jejich identifikačním číslem *NodeId*. Postupným procházením adresního prostoru *AddressSpace* dojde až k požadované proměnné. Jakmile je proměnná dostupná ve stromě uzlů, může ji uživatel začít odebírat. Pravým tlačítkem myši si rozbalí kontextové menu, ve kterém klikne na možnost odebírat uzel. Tento uzel se pak zobrazí v tabulce odebíraných proměnných. Tabulka má v GUI nejvíce prostoru. Je to proto, že v tabulce musí být vypsané základní atributy uzlu jako jsou: jméno proměnné, hodnota proměnné a časová známka. Nutné je taky počítat se zobrazením tlačítka na konci řádku. Toto tlačítko pak po stisknutí uživatelem aktivuje ukládání proměnné s jejími atributy do databáze.

Na základě stanovených požadavků jsou určeny hlavní funkce klienta a jejich použití je zde popsáno. Ovšem to nejsou veškeré funkcionality, které klient nabízí. Mnoho dalších příkazů je implementovaných v rámci kontextových menu jednotlivých widgetů z knihovny PyQt5 a aplikace Qt Creator. Například při přidání políčka pro vepsání URI adresy serveru, je automaticky přidáno funkční kontextové menu. Nevýhodou těchto přidaných menu je, že jsou předdefinovaná v anglickém jazyce, a proto se může stát, že celá musí být potlačena a je nutnost vytvořit nové. Samozřejmě velkou výhodou je použití Qt Creator, kde lze pracovat stylem *drag and drop*. Tedy během pouhých pár okamžiků je vytvořeno grafické rozhraní. Přímo v aplikaci lze pak jednotlivým widgetům přiřazovat funkce. Toto prostředí ale nepodporuje Python, a proto je potřeba skript přeložit. Knihovna PyQt5 byla zvolena mimo jiné na základě možnosti vytvářet moderně vypadající GUI. Tento požadavek se týká spíše designu a nesouvisí s funkcionalitou programovanou v jazyku Python. Pro splnění všech podmínek je vhodné prostudovat příklady stylů v Qt [17]. Prostudování tohoto zdroje velice usnadní práci při tvorbě přívětivě vypadajícího grafického rozhraní. Nakonec je důležité si uvědomit, že design nevychází z jazyka Python, ale z jazyka CSS.



Obr. 11: Vývojový diagram UA klienta

Dovětkem ke tvorbě konkretizovaných požadavků a vývojového diagramu je, že se studenti Základů strojního inženýrství během bakalářského studia s vývojovými diagramy aplikací nesetkají. Proto digram, zobrazený na Obr. 11: nebyl vytvořený v souladu s normami. Jeho vytvoření mělo za účel snadnější orientaci v programu a také mělo sloužit k odhalení nedostatků před samotným začátkem vývoje klienta OPC UA. Za povšimnutí stojí i políčka hlášení. Jedná se o stav, kdy je potřeba na uživatelskou interakci reagovat patřičným hlášením. Tato hlášení se vypisují do widgetu s informacemi o statusu.

Funkce a metody pro klienta z *FreeOpcUa*

Jak už bylo zmíněno, vytvořit OPC UA klienta v Pythonu je více než snadné. Klienta je možné napsat v pár řádcích kódu jen s malým množstvím funkcí z knihovny *FreeOpcUa*. Následuje výčet nejdůležitějších funkcí, které je vhodné při programování klienta použít. Návaznost je zachována tak, jak by měly být funkce implementovány:

- *def connect (self)* – jde o metodu vyššího levelu, klient se připojí k serveru a zahájí aktivní relaci. Konkrétněji se zavoláním této funkce provede další série příkazů knihovny. Ty jsou patrné z následující ukázky. [11]

```
def connect(self):
        """
        High level method
        Connect, create and activate session
        """
        self.connect_socket()
        try:
            self.send_hello()
            self.open_secure_channel()
            try:
                self.create_session()
                try:

                self.activate_session(username=self._username,
                                      password=self._password,
                                      certificate=self.user_certificate)
                except Exception:
                    # clean up the session
                    self.close_session()
                    raise
            except Exception:
                # clean up the secure channel
                self.close_secure_channel()
                raise
        except Exception:
            self.disconnect_socket()  # clean up open socket
            raise
```

- *def get_root_node (self)* – tato funkce svou podstatou není příliš zajímavá, protože slouží jako *wrapper* funkci *def get_node (self, nodeid)*. Ovšem nejde pouze o vracení jakéhokoliv uzlu z *AdressSpace,* ale funkce zmapuje kořenovou složku, kterou je nutné mít uloženou v proměnné. Klient s touto proměnnou musí dáte pracovat a procházet hierarchii *AddressSpace*.
- *def get_children (self)* – získá uzly ve směru procházení *AddressSpace*. Tyto uzly jsou nalezeny pomocí referencí, kterými jsou jednotlivé uzly propojeny.
- *def get_node (self*, *nodeid*) – při volání této funkce musí být známá *nodeid*. Ta může být poskytnuta provozovatelem serveru, nebo získána použitím UaExperta.
- *def get_value (self)* – zavoláním je získána hodnota uzlu. Pouze proměnné (a vlastnosti) mají svou hodnotu. Pro ostatní typy uzlů bude vygenerovaná výjimka.
- *def disconnect (self)* – jako v případě *def connect (self)* se jedná o metodu vyššího levelu, která ukončí nejprve relaci, poté spojení a finálně zavře zásuvku *socket*.

Na první pohled se může zdát neskutečné, jak málo stačí k vytvoření jednoduchého OPC UA klienta. Ale především díky jazyku Python tomu opravdu tak je. Další kapitola je věnována vytvoření nejjednoduššího OPC UA klienta. Tento klient obsahuje pouze funkce/metody uvedené v předchozím výčtu.

3.2 Jednoduchý klient v Pythonu bez grafického rozhraní

```
"""
Prepracoval a rozsiril: Jan Zmrzly v ramci bakalarske prace
Vyuziti: k vyuce a tvorbe jednoduchych GUI
Importovani knihoven FreeOPC Ua, dostupne z:
https://github.com/FreeOpcUa
Samotny kod je inspirovan Rocket Systems, dostupne z:
https://www.youtube.com/watch?v=12dnhcEmrU0
"""
from opcua import Client
import time
import datetime

#adresa serveru, musi byt poskytnuta provozovatelem
url = "opc.tcp://127.0.0.1:4840"
client = Client(url)
client.connect()
print("Klient pripojeny k serveru: ", format(url))

root = client.get_root_node()
print("Navazujicimi uzly jsou: ", root.get_children())

try:
    while True:
        Temp = client.get_node("ns=2;i=2")
        Temperature = Temp.get_value()
        print("Teplota je: " + str(Temperature))
        Press = client.get_node("ns=2;i=3")
        Pressure = Press.get_value()
        print("Tlak je: " + str(Pressure))
        print(datetime.datetime.now())

        time.sleep(2)
except KeyboardInterrupt:
    client.disconnect()
    print("\n Relace ukončena. Odpojeno od serveru: ",
format(url))
```

3.3 Návrh grafického rozhraní UA klient

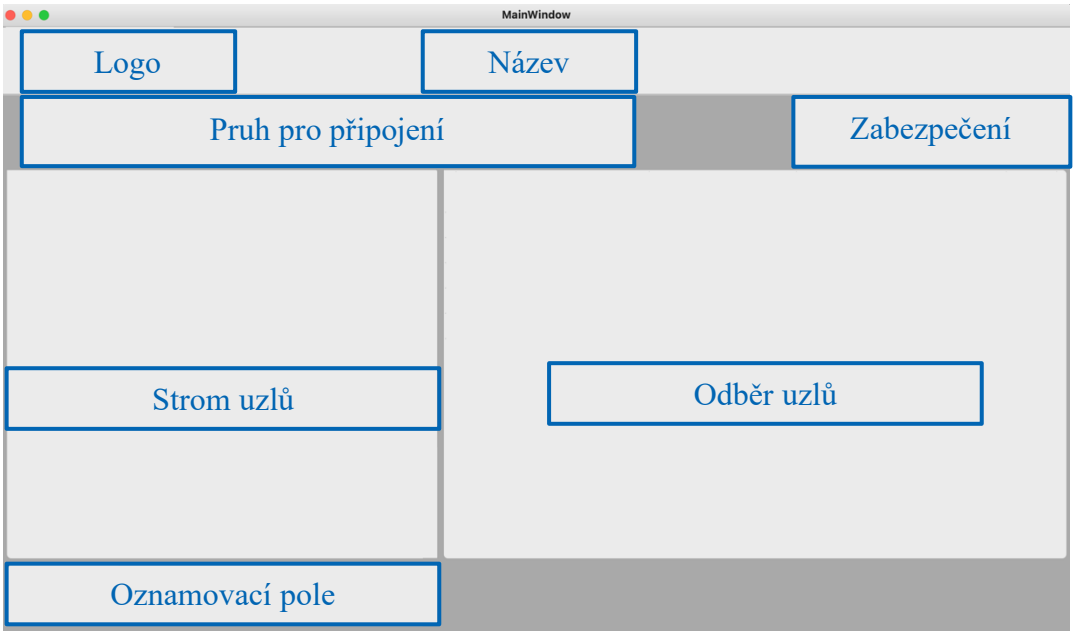
V předchozích částech byly postaveny silné základy pro vytvoření jednoduchého OPC UA klienta v jazyku Python. Nyní na těchto základech vzroste uživatelky příjemné rozhraní s požadovanou funkcionalitou. Je patrné, že návrh se rozložením polí inspiruje Ua Expertem. Ten je vyzkoušený uživately a do značné míry přívětivý. Jak bylo zmíněno dříve, jeho přívětivost roste s vetší znalostí problematiky OPC UA.

Hlavní okno je vytvářeno na nejmenší rozlišení 1280x720 px (HD). V horní části klienta se nachází samotný název. Název není žádným způsobem interaktivní. Zleva názvu je záměrně vynechané místo. To může sloužit pro umístění loga firmy. Pod touto

„hlavičkou“ se nachází pruh s ovládacími prvky – *pruh pro připojení*. Zcela vlevo je umístěno pole pro vložení připojovací adresy serveru. Uprostřed *pruhu pro připojení* se nachází série tlačítek. Na stejném řádku jsou umístěny tlačítka *Připojit*, *Odpojit*. Funkcionalita těchto tlačítek je patrná z názvu. Slouží k připojení nebo odpojení OPC UA serveru. Pod nimi se nachází tlačítko *Dotaz sítě*. Možnosti pro nastavení zabezpečení komunikace *SecurityMode* a *SecurityPolicy* se nachází v pravé části pruhu s ovládacími prvky.

Níže se nacházejí dvě nejrozsáhlejší oblasti celého uživatelského rozhraní. Nalevo se rozkládá oblast sloužící k zobrazení hierarchie *AddressSpace*, která je reprezentována stromem uzlů. Nutností je, aby oblast *stromu uzlů* byla dostatečně velká a pro uživatele byla orientace v adresním prostoru příjemná. Proto je nastavena nejmenší možná šířka *stromů uzlů* na 460 px. Tato oblast je dále rozdělená na dva sloupce. V širším sloupci může uživatel procházet adresní prostor. V užším sloupci se objevují *NodeId* jednotlivých uzlů. Defaultně jsou šířky sloupců nastaveny v poměru 6:1, uživatel si může rozměry změnit podle své potřeby. Napravo se pak nachází oblast *odběru uzlů*. Přes celou oblast se rozkládá tabulka z knihovny PyQt5. Její velikost je opět nastavena na největší možnou hodnotu a to na 740 px. Tato tabulka obsahuje čtyři sloupce: *Zobrazené jméno*, *Hodnota*, *TimeStamp* a sloupec pro tlačítko *Ukládat*. Šířky sloupců byly nastaveny spíše intuitivně, protože počet znaků v *Zobrazeném jméně* a *Hodnotě* se často velmi liší a není snadné nastavit na konkrétní hodnotu.

Nakonec se v dolní části nachází *oznamovací pole*. V grafickém rozhraní je kladen důraz na práci s malým spektrem barev, kdy je vybrána kombinace barev tmavě modré a šedé. Použitým fontem písma je *Roboto.* Schéma rozložení je přiloženo na Obr. 12:



Obr. 12: Návrh uživatelského rozhraní UA klienta

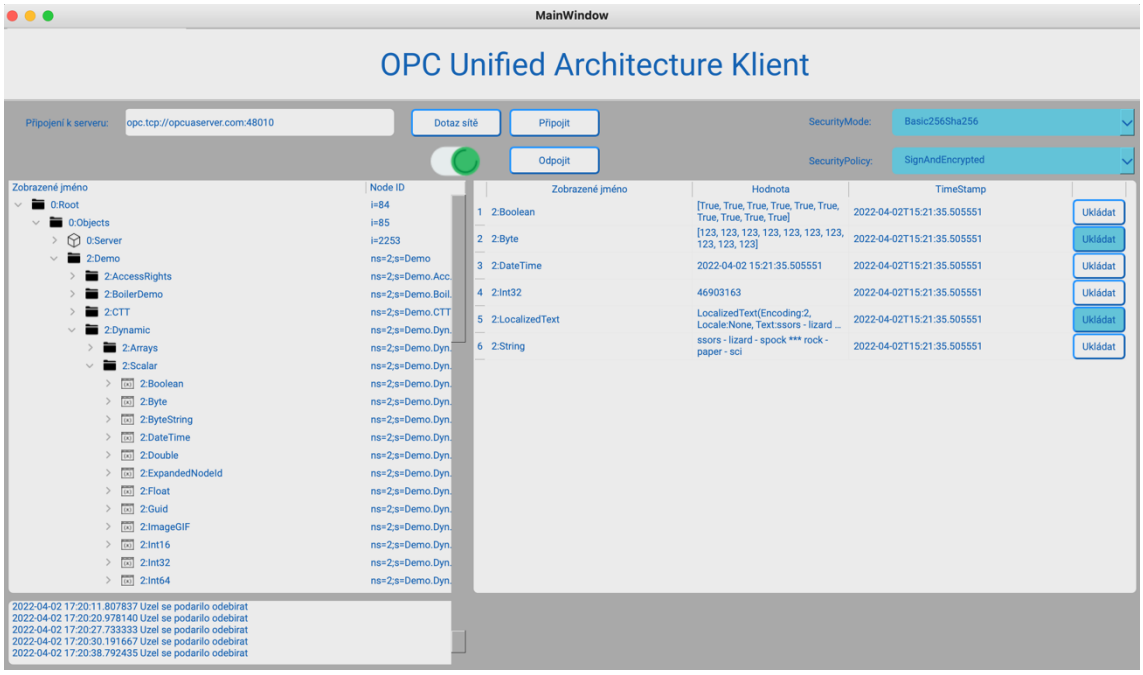
3.4 UA klient v Pythonu

Finální verze aplikace vychází z jednoduchého klienta v Pythonu. Aplikace je pochopitelně více složitá a obsahuje více metod a funkcí knihovny *FreeOpcUa*, Obr. 13: V textu není prostor na komentování všech částí UA klienta, a proto je doporučeno podívat se přímo do zdrojových kódů, kde jsou do značné míry vhodně okomentovány všechny operace. K prostudování a pochopení aplikace je nutné projít tyto skripty:

- *main_application.py*
- *client_application.py*
- *mainwindow.py*

Do aplikace byly implementovány všechny funkce, které vyplynuly ze zadání. Uživatel může zadat konkrétní IP adresu serveru, ke kterému se po kliknutí na tlačítko *Připojit* připojí. Následně se zobrazí *strom uzlů*, který může uživatel procházet do doby, než narazí na hledaný uzel. Po najití uzlu je nutné na něj kliknout pravým tlačítkem myši. Tímto kliknutím se vyvolá kontextové menu, ve kterém je jedna z nabízených možností *Odebírat uzel*. Pokud je uzel odebírán, objeví se jako řádek v tabulce *odběru uzlů*. Odebírané uzly mohou být po kliknutí na tlačítko *Ukládat* uloženy do databáze. K uložení dojde pouze pokud se změní hodnota proměnné. Do databáze se uloží objekt s hodnotami, které jsou zobrazeny v řádku. Tlačítko *Ukládat* se nachází na konci každého řádku.

V průběhu vývoje UA klienta bylo spojení testováno na online serveru, který je sponzorován One–Way Automation Inc. Více informací o projektu je přímo uvedeno na stránce *opcuaserver.com* [18]. Funkční server, ke kterému byl UA klient schopný se připojit, běží na adrese *opc.tcp://opcuaserver.com:48010*.



Obr. 13: UA klient testovaný na *opc.tcp://opcuaserver.com:48010*

4 OVĚŘENÍ FUNKČNOSTI KLIENTA

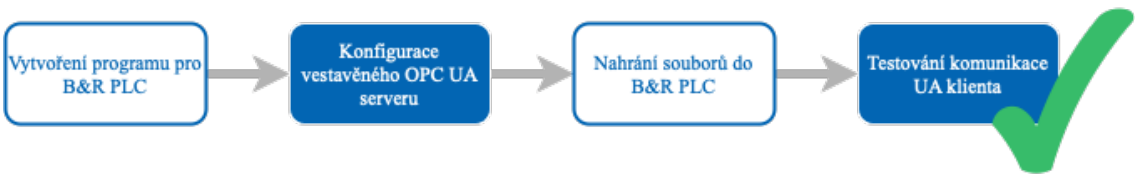
Ověření funkčnosti klienta bylo závěrečným krokem práce. Již při vývoji byl klient testován na online serveru patřícím pod projekt společnosti One–Way Automation Inc., Obr. 13: UA klienta bylo taky nutné ověřit na komunikaci s PLC B&R Automation. Výsledky tohoto testu nejsou příliš zajímavé, protože spojení není navázáno s reálným automatem. Daleko zajímavější je testování na zařízení společnosti Beckhoff. To proběhlo na reálném PLC ve skutečné podnikové síti.

4.1 Komunikace s PLC od společnosti B&R Automation

Naneštěstí nebylo k dispozici reálné PLC od společnosti B&R. Ovšem Automation Studio, kde probíhá vývoj programů do automatů společnosti B&R, nabízí možnost simulace reálného PLC v módu ArSim. Práce s ArSim je prakticky k nerozeznání od reálného programovatelného automatu. Nezbytné je připravit jednoduchý program v jazyku ST nebo C. Vytvořený program se pak nahraje do automatu a ten jej začne cyklicky provádět. Dále je nezbytně nutné nakonfigurovat vestavěný OPC UA server. Nastavení není příliš složité a zvládne ho i začátečník. Stačí postupovat podle pokynů v kartě *Help*. Firma B&R také nabízí různé druhy školení, které jsou pro studenty zdarma a jejich absolvování velmi rozšíří znalosti v oblasti programování PLC. Odkaz na registrace školení se pak nachází stránkách společnosti B&R [19].

Jakmile jsou veškerá nastavení hotová zbývá pouze nahrát soubory do programovatelného automatu. Celý proces je názorně shrnutý na Obr. 14: Pro připojení pomocí UA klienta pak stačí pouze zadat správnou adresu a port. Adresa a port se musí shodovat s nastavením ve vestavěném serveru.

Byl vytvořen jednoduchý program, který zobrazuje náhodné proměnné v intervalu 100 ms. Tyto proměnné jsou pojmenovány: *Temperature* a *Pressure*. Proměnné se společně s jejich atributy, zobrazenými na řádku UA klienta, povedlo ukládat do databáze SQL Lite. Vyplývá z toho, že UA klient je schopný komunikace s vestavěným OPC UA serverem na zařízení společnosti B&R Automation.



Obr. 14: Konfigurace vestavěného OPC UA serveru

4.2 Komunikace s PLC od společnosti Beckhoff

Daleko zajímavější bylo testování UA klienta na zařízení společnosti Beckhoff. Zajímavější bylo z toho důvodu, že automat byl reálný a připojený do skutečné podnikové sítě. Jedná se o PLC, které řídí jednoúčelový testovací stroj na pneumatické pohony a které se nachází v oddělení montáže. Nastavení vestavěného OPC UA serveru proběhlo podobně, jako v případě zařízení od společnosti B&R.

Programy v zařízeních reálného provozu často obsahují více než sto proměnných. Jinak tomu nebylo ani v případě automatu Beckhoff. Zpřístupněných bylo tedy pouze několik málo proměnných od každého typu.

UA klient byl spuštěn na počítači v oddělení engineeringu. Poté byla klientovi poskytnuta adresa a port pro připojení k serveru. Bez větších zdržení se klient zvládl připojit k serveru. Stejně jako v předchozím případě se podařilo proměnné společně s jejich atributy, zobrazenými na řádku UA klienta, ukládat do databáze SQL Lite. Vyplývá z toho, že UA klient je schopný komunikace s vestavěným OPC UA serverem na zařízení společnosti Beckhoff.

5 ZÁVĚR

V rámci bakalářské práce byla nastudována technologie OPC UA, která je v dnešní době stále velice moderní a progresivně se rozvíjí. V práci byl shrnut historický vývoj včetně specifikace OPC Classic, ze které průmyslový standard OPC UA vychází.

Hlavní částí práce bylo vytvoření OPC UA klienta s příjemným grafickým rozhraním. Klient byl vytvářen s pomocí programovacího jazyka Python, a tedy i jemu byla věnována vlastní kapitola. Aplikace byla vytvořena převážně za použití knihoven FreeOpcUa a PyQt5. Při vývoji programu se velice osvědčila práce s platformou *github.com,* která nabízí programátorům možnosti verzování projektů a případné spolupráce na vytvářených aplikacích.

Nejprve byl sestaven jednoduchý klient, jehož kód se nachází v kapitole *3.2 Jednoduchý klient v Pythonu bez grafického rozhraní.* Na něm byly vyzkoušeny hlavní funkce knihovny FreeOpcUa. Následovalo vytvoření aplikace s grafickým rozhraním. Ta byla otestována na komunikaci s online OPC UA serverem, PLC firmy Beckhoff a PLC firmy B&R.

Nad rámec byly klientovy přidány funkce SQL Lite databáze. Uživatel má možnost v případě potřeby ukládat změny hodnot proměnných do databáze s jejich příslušným jménem, hodnotou a časovou známkou. Práce může sloužit studentům při prvotním seznamováním s průmyslovou standardizací OPC UA. Na základně ní si mohou sestavit své vlastní jednoduché klienty a vyzkoušet je na komunikaci se svými PLC, případně dalším hardwarem podporujícím OPC UA.

Aktuální verze projektu dostupná z *https://github.com/JanZmrzly/klient-opc* a aplikace je šiřitelná pod licencí GPL-3.0. Do budoucna se předpokládá, že projekt bude udržován aktivní. Pro využití v praxi by bylo vhodné doplnit možnost nastavení způsobu zabezpečení komunikace, aby nemohl nastat případ, že data budou nevhodně emitována třetím stranám.

6 SEZNAM POUŽITÉ LITERATURY

[1] Python. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2022 [cit. 2022-03-16]. Dostupné z: https://cs.wikipedia.org/wiki/Python

[2] PECINOVSKÝ, Rudolf. *Začínáme programovat v jazyku Python*. Praha: Grada Publishing, 2020. Začínáme s.. ISBN: 978-80-271-1237-1.

[3] SUMMERFIELD, Mark. *Python 3: výukový kurz*. Brno: Computer Press, 2010. ISBN: 978-80251-2737-7.

[4] FINER, Jasmine. PEP8. *Realpython.com* [online]. 2018, 19.12.2018 [cit. 2022-03-16]. Dostupné z: https://realpython.com/python-pep8/

[5] PETERS, Tim. *PEP20* [online]. 2004, 19.08.2004 [cit. 2022-03-16]. Dostupné z: https://peps.python.org/pep-0020/

[6] HEINE, Andreas, Denis ŠTOGL, Joey FAULKNER a Alexander RYKOVANOV. FreeOpcUa. *Github.com* [online]. [cit. 2022-03-16]. Dostupné z: https://github.com/FreeOpcUa

[7] LUTZ, Peter. OPC UA: Rozšíření komunikace řídicích prvků pomocí TSN a APL. *Vseoprumyslu.cz* [online]. Česko: Control Engineering Česko, 2021 [cit. 2022-04-04]. Dostupné z: https://www.vseoprumyslu.cz/automatizace/site-a-komunikace/opc-ua-rozsireni-komunikace-ridicich-prvku-pomoci-tsn-a-apl.html

[8] History of OPC. *Opcconnect.com* [online]. OPCconnect.com, 2022 [cit. 2022-04-04]. Dostupné z: https://www.opcconnect.com/history.php

[9] DAMM, Matthias, Stefan-Helmut LEITNER a Wolfgang MAHNKE. *OPC Unified Architecture*. Berlin, Heidelberg: Springer-Verlag, 2009. ISBN: 9783540688983. Dostupné z: doi:10.1007/978-3-540-68899-0

[10] OPC Classic. *Opcfoundation.org* [online]. [cit. 2022-04-04]. Dostupné z: https://opcfoundation.org/about/opc-technologies/opc-classic/

[11] OPCBLOGADMIN. What is OPC?. *Integrationobjects.com* [online]. 2018 [cit. 2022-04-04]. Dostupné z: http://opcconnect.integrationobjects.com/what-is-opc-most-used-technology-automation-world/

[12] Why OPC UA Matters. *Ni.com* [online]. 2021 [cit. 2022-04-04]. Dostupné z: https://www.ni.com/cs-cz/innovations/white-papers/12/why-opc-ua-matters.html

[13] VOJÁČEK, Antonín. Průmyslová komunikace OPC UA - 1.díl - popis protokolu. *Automatizace.hw.cz* [online]. 2020, 22.07.2020, 1 [cit. 2022-03-16]. Dostupné z: https://automatizace.hw.cz/prumyslova-komunikace-opc-ua-1dil-popis-protokolu.html

[14] KARL. UA Overview. *Wiki.opcfoundation.org* [online]. 2015 [cit. 2022-04-04]. Dostupné z: http://wiki.opcfoundation.org/index.php?title=File:Uamodel.png

[15] FIGINI, Alessandro. EtherCAT. *Automa.cz* [online]. 2017 [cit. 2022-03-16]. Dostupné z: https://automa.cz/cz/casopis-clanky/ethercat-automation-protocol-2017_02_0_9811/

[16] *Unified Automation* [online]. Scottsdale: opcfoundation, 2021 [cit. 2022-04-04]. Dostupné z: https://www.unified-automation.com/products/development-tools/uaexpert.html

[17] QT Dokumentace. *Doc.qt.io* [online]. Finland: Free Software Foundation, 2022 [cit. 2022-04-04]. Dostupné z: https://doc.qt.io/qtforpython/overviews/stylesheet-examples.html

[18] OPC UA online server. *Opcuaserver.com* [online]. One-Way Automation [cit. 2022-04-04]. Dostupné z: http://opcuaserver.com

[19] Školení společnosti B&R. Br-automation.com [online]. Brno: B&R, 2022 [cit. 2022-04-04]. Dostupné z: https://www.br-automation.com/cs/akademie/skoleni/vsechny-zeme/skoleni-v-ceske-republice/

[20] KABELOVÁ, Alena a Libor DOSTÁLEK. *Velký průvodce protokoly TCP/IP a systémem DNS*. 5., aktualiz. vyd. Brno: Computer Press, 2008, 488 s. : il. ; 23 cm. ISBN 978-80-251-2236-5.

[21] GRUBER, Martin. *Mistrovství v SQL. Svazek 1*. Praha: Softpress, 2004, 480 stran : ilustrace. ISBN 80-86497-62-3.

[22] *Ikonky* [online]. 2022 [cit. 2022-05-08]. Dostupné z: https://icons8.com

7 SEZNAM OBRÁZKŮ

8 SEZNAM PŘÍLOH

- klient-opc-main (složka)
- opc_ua_server.py
- opc_ua_klient.py
- readme.pdf