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

TVORBA KOMUNIKACE PŘES ROZHRANÍ OPC UA MEZI PC A PLC SIEMENS SIMATIC S7-1200

CREATION OF COMMUNICATION VIA OPC UA INTERFACE BETWEEN PC AND PLC SIEMENS SIMATIC S7-1200

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Nina Štěrbová
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
Studentka:	**Nina Štěrbová**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
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:

Tvorba komunikace přes rozhraní OPC UA mezi PC a PLC Siemens SIMATIC S7–1200

Stručná charakteristika problematiky úkolu:

Práce je věnována problematice výměny (aplikačních) dat mezi průmyslovým řídicím systémem a osobním počítačem s využitím komunikačního protokolu OPC UA. Je uvažována konfigurace, kdy na straně řídicího systému (PLC/IPC) je aktivní OPC UA server, na straně PC je spuštěn OPC UA klient (komunikující se serverem). Student vytvoří na PLC Siemens SIMATIC S7–1200 jednoduchou testovací aplikaci a provede konfiguraci OPC UA serveru. Dále student vytvoří jednoduchou aplikaci OPC UA klienta na straně PC. Preferovaný programovací jazyk pro tvorbu klienta je C#. OPC UA stack (framework, na kterém bude klient založen) si student zvolí sám. Vytvořený OPC UA klient by měl být schopný z PLC vyčítat obsah (zveřejněných) proměnných, eventuálně do proměnných i zapisovat. Preferována jsou jednoduchá a přehledná řešení aplikací, podstatný je zejména edukační přínos práce.

Cíle bakalářské práce:

Proveďte stručnou rešerši problematiky datové komunikace v průmyslu, pozornost věnujte zejména protokolu OPC UA.
Nastudujte základy práce ve vývojovém prostředí TIA Portal pro programování PLC Siemens SIMATIC S7–1200.
Nastudujte konfiguraci OPC UA serveru na straně PLC SIMATIC S7–1200.
Vytvořte na PLC jednoduchý program pro testování komunikace přes OPC UA a aktivujte OPC UA server.
Zvolte programovací jazyk a vhodný framework pro účely vytvoření OPC UA klienta na straně PC.
Popište postup tvorby OPC UA klienta (s praktickou ukázkou jednoduché klientské aplikace).
Stručně zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

OPC Foundation. OPC Unified Architecture: Interoperability for Industrie 4.0 and the Internet of Things [online]. [cit. 2021-10-14]. Dostupné z: https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN.pdf

Siemens. OPC UA [online]. [cit. 2021-10-14]. Dostupné z: https://new.siemens.com/cz/cs/products/automation/industrial-communication/opc-ua.html

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

Bakalářská práce se zabývá datovou výměnou mezi PC a PLC prostřednictvím moderního komunikačního rozhraní OPC UA. Je uvažována konfigurace serveru na straně PLC a klientská aplikace pro PC včetně popisu tvorby. Důraz je kladen zejména na edukační přínos, proto jsou vytvořené aplikace spíše jednoduššího charakteru. Práce je dále doplněna o stručné shrnutí poznatků v oblasti průmyslové komunikace, OPC UA protokolu a TIA Portal pro snazší pochopení dané problematiky.

ABSTRACT

This bachelor's thesis introduces a data exchange between a PC and PLC via a modern communication interface OPC UA. The thesis deals with a configuration of the server on a PLC and client applications for a PC including a description of its development. In this work the emphasis is especially put on educational purposes thus the aforementioned applications are of a simple character. The thesis is completed with a brief summary of findings in the field of industrial communication, OPC UA protocol and TIA Portal in order to better understand the topic.

KLÍČOVÁ SLOVA

OPC UA klient, OPC UA server, SIMATIC S7-1200, TIA Portal

KEYWORDS

OPC UA Client, OPC UA Server, SIMATIC S7-1200, TIA Portal

ÚSTAV AUTOMATIZACE A INFORMATIKY

2022

BIBLIOGRAFICKÁ CITACE

ŠTĚRBOVÁ, Nina. *Tvorba komunikace přes rozhraní OPC UA mezi PC a PLC Siemens SIMATIC S7-1200*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, 2022, 63 s. Bakalářská práce. Vedoucí práce: Ing. et Ing. Stanislav Lang, Ph.D.

PODĚKOVÁNÍ

Touto cestou bych ráda podělovala společnosti Eprin za poskytnuté zázemí, hardware a odborné konzultace. Vedoucímu bakalářské práce Ing. et Ing. Stanislavu Langovi, Ph.D. bych chtěla poděkovat za cenné připomínky. Dále velké díky patří rodině, která mi v průběhu studia poskytovala psychickou i odbornou oporu. A na závěr bych chtěla poděkovat Martinovi Kmentovi za konzultace ohledně programování.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, vypracovala 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 autorka uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědoma následku porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 20. 5. 2022 .....

Nina Štěrbová

OBSAH

1 ÚVOD

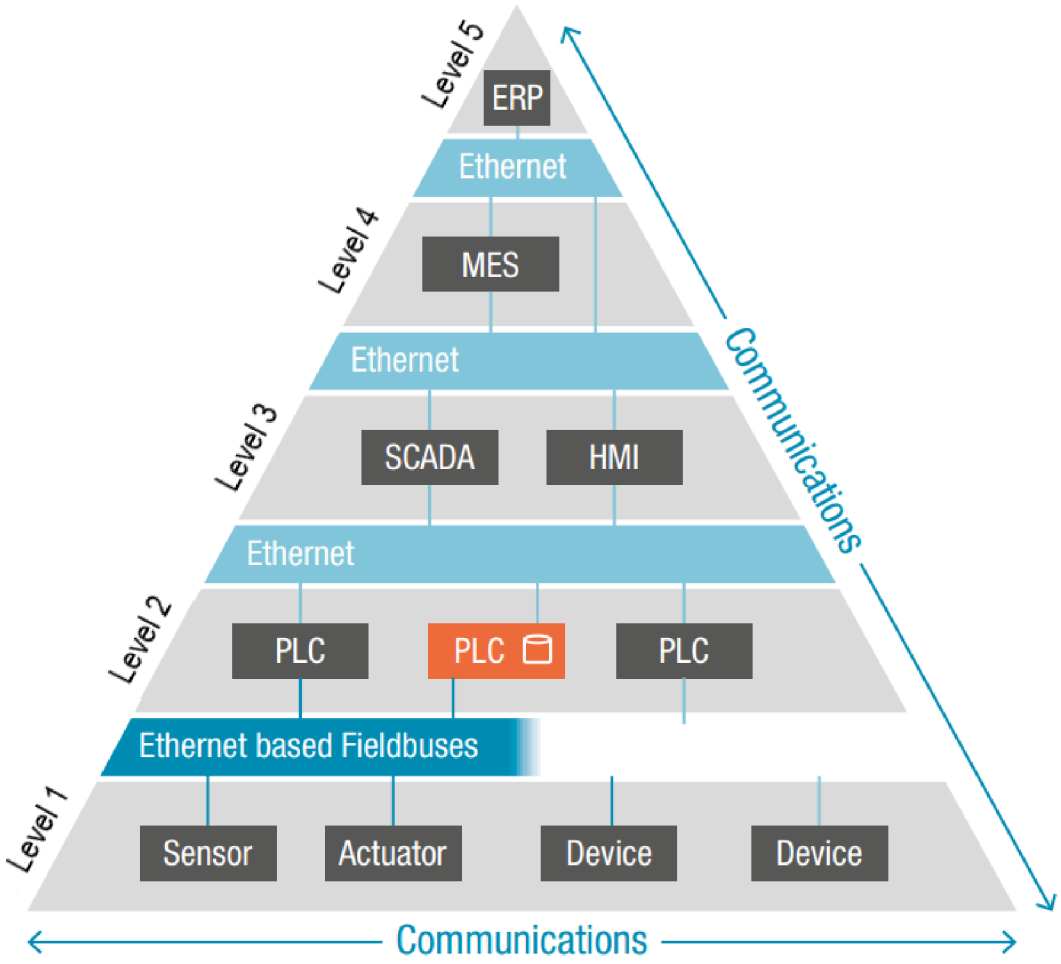
Průmyslové komunikační sítě se začaly zavádět během 80. let. Do té doby centrálně řízené systémy a tzv. point-to-point spojení přestaly být dostačují, a snahou tak bylo se přeorientovat na decentralizované řízení. To vedlo k zavedení prvních průmyslových sítí, jimiž byly sběrnice. Nedlouho nato se průmyslová komunikace zaměřila na protokoly založené na Ethernetu. Významným nedostatkem dosavadních protokolů je složitější komunikace mezi zařízeními a systémy od různých dodavatelů. Řešením je otevřený protokol OPC UA spravovaný OPC Foundation. Jedná se o komunikační platformu nezávislou na operačním systému a výrobci, která nachází nejen uplatnění v horizontální komunikaci, ale i ve vertikální, kdy dokáže například propojit procesní patra podniku přímo s managementem. Takovéto propojení podniku zefektivní tok dat a jejich následné zpracování.

Bakalářská práce se zabývá komunikací mezi PLC a PC právě přes komunikační rozhraní OPC UA, které uvažuje konfiguraci OPC UA server na straně PLC Siemens SIMATIC S7-1200 a klientskou aplikaci na straně PC s operačním systémem Windows 10. Z úvodu práce je pro kontext stručně popsáno několik klíčových pojmů z oblasti průmyslových komunikačních sítí. Následuje kapitola, v níž je popsán vývoj a základní charakteristiky OPC UA protokolu. Práce má primárně edukační účel, proto je zde popsáno vývojové prostředí TIA Portal pro tvorbu PLC programů včetně návodu na založení nového projektu a konfigurace OPC UA serveru. V neposlední řadě se práce věnuje vývoji jednoduchých klientských aplikací, které jsou se zmíněným PLC schopny komunikovat. Jako ukázka byla vytvořena konzolová aplikace v jazyce Python 3 a OPC UA klient s grafickým uživatelským rozhraním v jazyce C#.

2 ÚVOD DO PRŮMYSLOVÉ KOMUNIKACE

Organizace průmyslového podniku byla (a v mnoha případech stále ještě je) jasně strukturovaná a hierarchicky postavená. Jedná se o tzv. automatizační pyramidu zobrazenou na Obr. 1. Pyramidu obvykle tvoří pět pater. Na nejnižším patře figurují senzory, akční členy či podobná zařízení. Druhé patro patří PLC, která řídí procesy probíhající na první úrovni a zároveň od nich shromažďují informace, které předávají o patro výš operátorským systémům. Operátorské systémy (SCADA, HMI) jsou následně schopné data vyhodnotit a podle toho upravit parametry v řídicích systémech. Čtvrtá úroveň patří plánování výroby a logistice, k čemuž se využívá Manufacturing Execution System (MES). Nejvyšší patro reprezentuje management se systémem Enterprise Resource Planning (ERP).

V rámci každé úrovně pyramidy jsou zavedené komunikační sítě, které jsou napojené na síť podřazené, resp. nadřazené úrovně. Cílem procesu digitalizace je propojit jednotlivé část podniku a zajistit efektivnější tok dat. Tím se jasně vymezené hranice mezi úrovněmi pomyslné pyramidy smazávají, což má za následek nové požadavky na průmyslové komunikační sítě.



Obr. 1: Automatizační pyramida [1]

2.1 ISO/OSI model

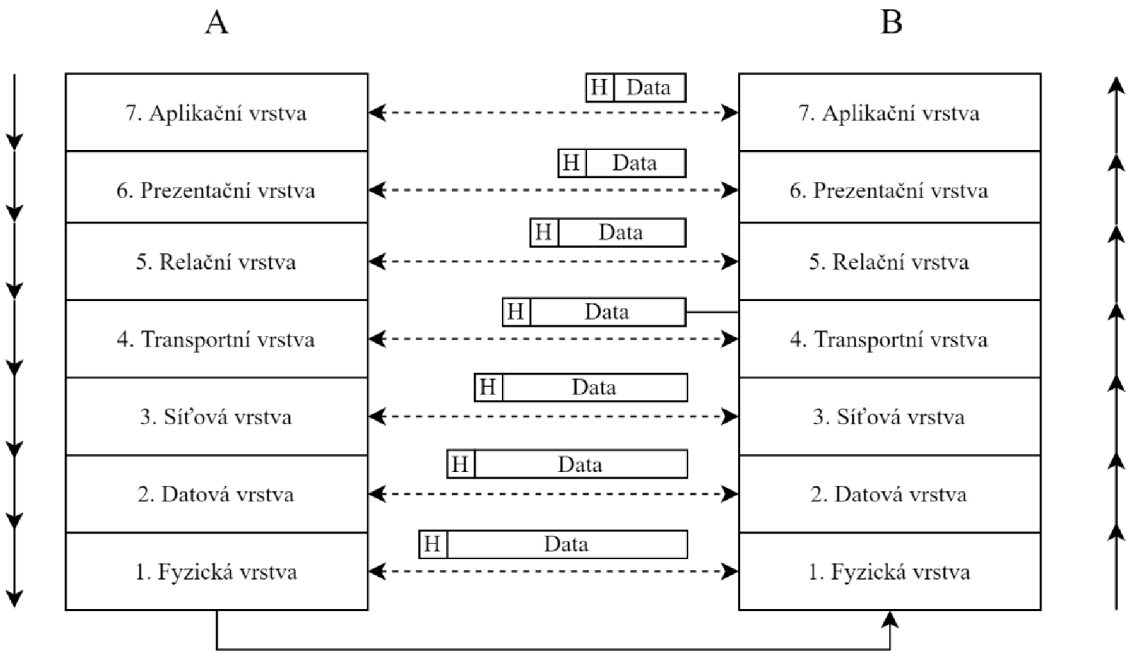
Pro rozvoj průmyslových sítí byl základním pilířem referenční model ISO/OSI, přestože se původně jednalo o koncept vytvořený pro počítačové sítě. V kontextu dílčích částí datové komunikace se došlo k závěru, že by měl mít model hierarchickou strukturu tvořenou sedmi vrstvami. OSI model charakterizují tři základní parametry:

- **Protokol** – je sada pravidel, které řídí komunikaci mezi vrstvami na stejných úrovní různých systémů. Vrstvy stejné úrovně jsou označovány jako tzv. peer layers.
- **Služba** – jedná se o jakoukoli dostupnou službu, kterou poskytuje jedna vrstva (tzv. service provider) vrstvě sousední (tzv. service user). OSI model definuje pouze funkcionalitu služeb, nikoli způsob, jakým mají být implementovány.
- **Rozhraní** – mezi dvěma sousedními vrstvami se nachází rozhraní, které specifikuje dostupné služby zprostředkovávané danou vrstvou [1].

Na Obr. 2 je zobrazena struktura OSI modelu znázorňující přenos dat mezi dvěma systémy. Datový packet vstupuje do celé výměny přes sedmou aplikační vrstvu a postupně postupuje až na nejspodnější fyzickou vrstvu, kde se následně uskuteční samotný přenos jednotlivých bitů dat přes komunikační médium. Každá vrstva přijatá data zpracuje a připraví ve vhodném formátu pro následující vrstvu.

Model se všemi sedmi vrstvami je značně náročný na realizaci v praxi. Vůbec první průmyslovou aplikací v oblasti automatizace, která byla založena na OSI modelu, byl Manufacturing Automation Protocol (MAP) využívající všech sedm vrstev, což byl důvod, proč se neuchytil. Kvůli své robustnosti a komplexnosti byl finančně velmi nákladný a jeho implementace nenašla obecného využití. Je tedy obvyklé, že se počet vrstev minimalizuje. V průmyslu se často používá první, druhá a sedmá vrstva. V některých případech jsou určité funkce vynechaných vrstev potřebné, proto se požadované funkce připojují k využitým vrstvám. Třetí a čtvrtá vrstva může být sloučena buď s druhou nebo sedmou vrstvou, pátá a šestá vrstva vždy se sedmou. Využití pouze první, druhé a sedmé vrstvy nemusí být pravidlem. Například komunikační standard ControlNet implementuje i třetí a čtvrtou vrstvu [1].

Nad sedmou vrstvou OSI/ISO modelu stojí uživatelská vrstva, která propojuje řízené objekty. K tomuto může být použit OPC UA protokol, který z pohledu zvenčí skryje podrobnosti o síti a připraví data pro snazší předání jinému systému [2].

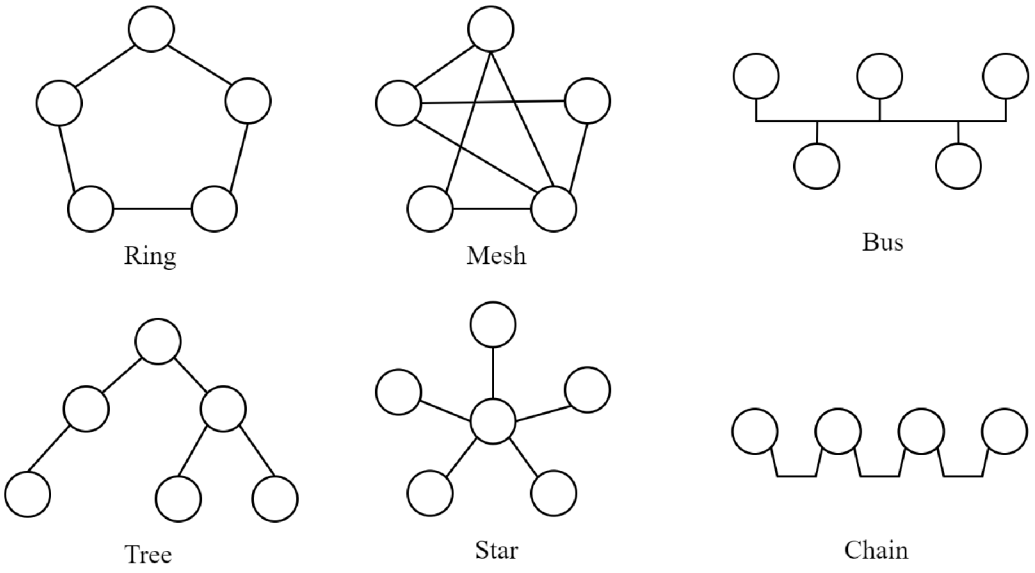


Obr. 2: Referenční model ISO/OSI

2.2 Síťové topologie

Důležitou charakteristikou sítí je jejich fyzické uspořádání, které se vybírá dle konkrétních požadavků. Vybrané topologie jsou znázorněny na Obr. 3 a dále stručně popsány.

- **Star** – topologie typická pro centralizované řízení procesů, kdy ve středu figuruje PLC, ke kterému jsou napojena periferní zařízení. Kvůli nadbytečné kabeláži se od ní na čas ustoupilo. Znovu se začala používat se zavedením Ethernetu.
- **Ring** – uzly jsou řazeny jeden za druhým, přičemž každý má dvě rozhraní (vstup a oddělený výstup). Protože není potřeba adresovat každý uzel, jedná se o metodu zajišťující poměrně rychlou výměnu dat. Významným protokolem využívajícím ring topologii je INTERBUS.
- **Chain** – topologie se vyznačuje tím, že každý uzel disponuje malých síťovým přepínačem (switch), kterým je spojen s dalším uzlem. Toto rozložení využívá například PROFINET.
- **Bus** – nejběžnější síťová topologie, která efektivně nahradila hvězdicovou topologii centralizovaného systému [1].
- **Mesh** – topologie, ve které může být jeden uzel propojen se dvěma a více uzly v síti.
- **Tree –** uzly jsou v hierarchickém uspořádání.



Obr. 3: Síťové topologie

2.3 Průmyslové sítě

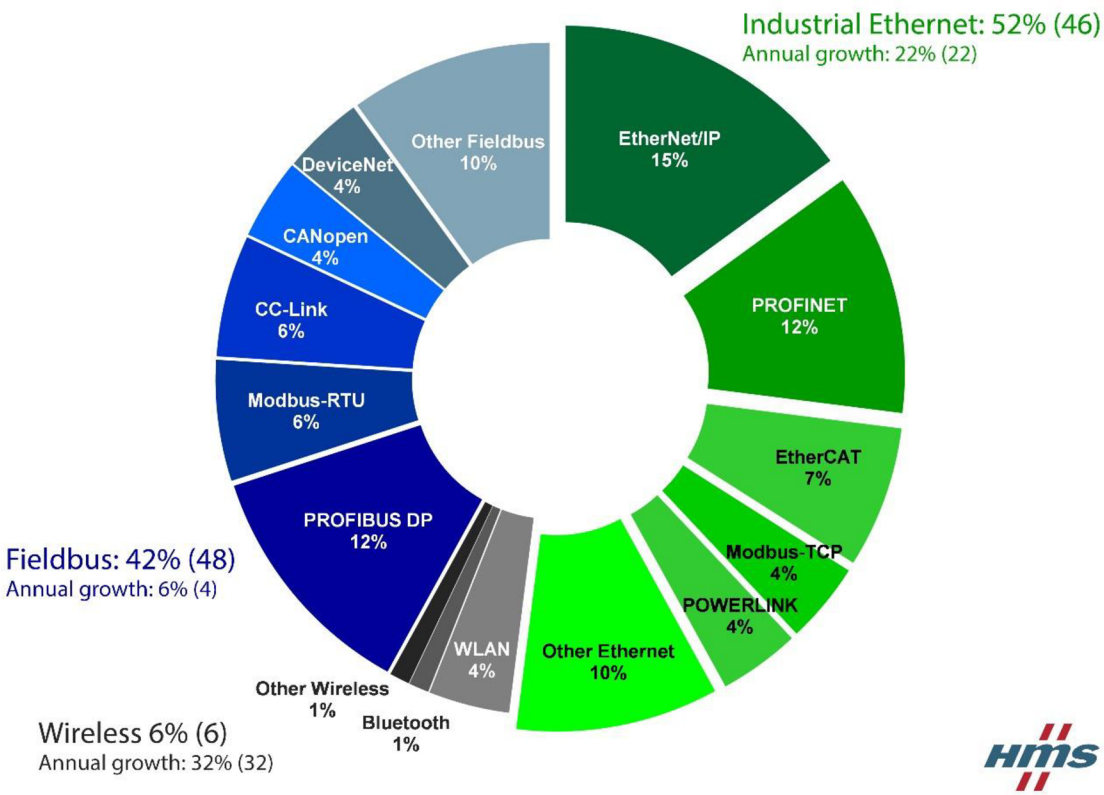
Prvním druhem průmyslových sítí byly sběrnice. Sběrnice je komunikační systém, který může propojit až několik desítek uzlů (zařízení). Propojení se realizuje s využitím vybrané síťové topologie, která respektuje případné hierarchické postavení jednotlivých uzlů. Výkon sběrnic je závislý na mnoha aspektech, jako je počet uzlů nebo přenosová rychlost. S rozvojem průmyslu ale přestal výkon stačit, a tak se protokoly začaly doplňovat o nové funkce, které ale vyžadovaly jiné fyzické provedení. To bylo finančně náročné. Sběrnice následně prošly standardizací a v současnosti jsou vedené jako mezinárodní standard IEC 61158. Příkladem takovýchto sběrnice je Profibus, ControlNet, InterBus (dnes již nepoužívaný) a další. Mezi sběrnice se řadí též standardy DeviceNet a CANopen, které nejsou ale vedeny jako IEC 61158 [3, 4].

Na přelomu tisíciletí se v průmyslových sítí začala čerpat inspirace z IT technologií a pozornost se přenesla na sítě založené na Ethernetu. Vznikly nové standardy jako EtherCAT, Profinet, EthernetIP, Ethernet Powerlink nebo Modbus TCP [4, 5].

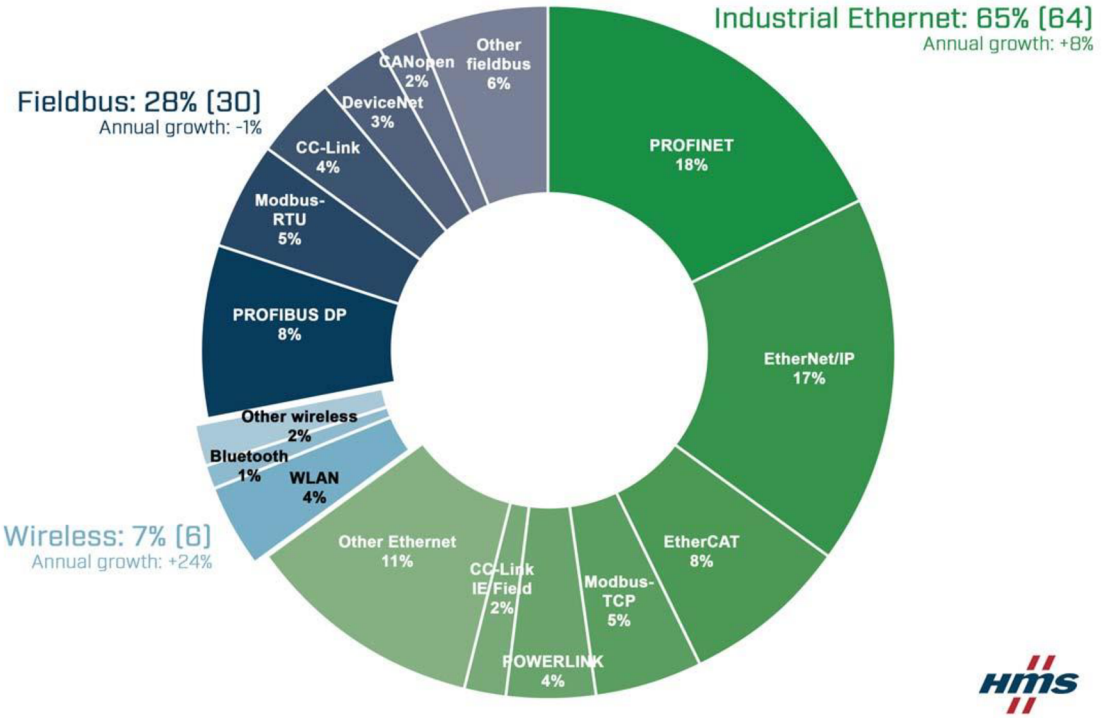
V průmyslu se pracuje také s bezdrátovými sítěmi. Jejich výhodou je snížení kabeláže a možnost propojení zařízení, která by pomocí kabelů nemohla být propojena. Nevýhodou je pro určité aplikace nedostatečná spolehlivost [3].

2.4 Další vývoj průmyslových sítí

Podle jedné z analýz trhu [4] dlouho dominovaly v průmyslových sítích sběrnice. Tento stav se k roku 2018 změnil a do popředí se dostal průmyslový Ethernet. Obr. 5 zobrazuje analýzu trhu z roku 2021. V porovnání se stavem z roku 2018 znázorněném na Obr. 4 je patrný ústup sběrnic, a naopak výrazný vzestup standardů založených na průmyslovém Ethernetu.



Obr. 4: Analýza průmyslových sítí na trhu z roku 2018 [6]



Obr. 5: Analýza průmyslových sítí na trhu z roku 2021 [7]

Protože v průmyslových aplikacích je často kritickým parametrem čas, zaměřuje se vývoj v oblasti průmyslové komunikace mimo jiné na přenos dat s co nejmenší časovou odezvou. V tomto ohledu se jako perspektivní technologie jeví Time-Sensitive Networking (TSN). TSN je série asi 60 IEEE standardů, které zajišťují přesnější determinismus a snižují latenci. Díky skutečnosti, že 13 standardů je zaměřených na bezpečnost a spolehlivost, je v rámci TSN zohledněna i tato stránka. Technologii využívá Ethernet, který je tímto rozšiřován o možnost funkce v reálném čase, nebo dále například protokol OPC UA, který začíná s TSN spolupracovat v kontextu modelu Pub/Sub [3].

V průmyslu se také přenáší čím dál větší pozornosti k celulárním sítím, zejména pak 5G. Jejich potenciál tkví v relativně širokém přenosovém pásmu a přesném determinismu umožňujícím synchronizované sledování procesu a lepší koordinaci akcí. Další výhodou je možné propojení výroby s řízením podniku. Nese to s sebou ale i otázku bezpečnosti, kdy provozovatel výroby je nucen vkládat kontrolu nad řízením sítě do rukou třetí strany [3].

3 OPC UA

V reakci na rozvíjející se průmysl se koncem 90. let minulého století dalo dohromady několik světových výrobců automatizačních technologií (Fisher-Rosemount, Intellution, Opto 22 a Rockwell Software), kteří si dali za cíl vyvinout standard, jenž by umožnil komunikaci mezi procesními zařízeními (např. inteligentními snímači, akčními členy) a řídicími systémy (např. PLC, HMI nebo SCADA systémy). Zásadním požadavkem byla interoperabilita různorodých zařízení a systémů, proto se vývojáři snažili vytvořit technologii, kterou by si lehce osvojil každý výrobce. Výsledkem byla série specifikací uvedená na trh v roce 1996, která nesla souhrnné označení OPC, dnes OPC Classic. Zkratka vychází ze slov *OLE for Process Control,* a jak může název napovídat, standard využíval pro přístup k síti technologii Windows OLE od Microsoftu, která mimo jiné pomohla k rychlému uvedení první verze na trh. Základními funkcemi OPC bylo čtení, zapisování a monitorování řídicích a procesních dat [2, 8].

V průběhu vývoje OPC standardu se k zakládajícím firmám přidaly další významné společnosti jako Siemens, Rockwell nebo Honeywell, což dalo vznik neziskové organizace OPC Foundation se sídlem ve Phoenixu, Arizona, jejímž úkolem je vývoj a správa OPC standardu. Organizace má v současnosti již přes 730 členů z celého světa, kteří svými členskými příspěvky umožňují její fungování. OPC Foundation nespolupracuje pouze s výrobci implementujícími OPC technologii do svých výrobků, ale spolupráci navazuje i s koncovými uživateli a jinými institucemi či organizacemi jako PLCopen, AIM, VDMA, MTConnect, AutomationML a další, což usnadňuje integrování standardu i do nových průmyslových odvětví [5].

OPC komunikace je postavena na modelu klient/server a uplatňuje se u nejrůznějších aplikací, proto se v rámci specifikací definovaly různé přístupové metody, a to podle pravidel (Distributed) Component Object Model (COM/DCOM), kdy COM zprostředkovával interní procesní komunikace a DCOM k tomu přidal výměnu informací napříč sítěmi. Nejvýznamnější a zároveň první specifikací je OPC DA, která bude dále spolu s OPC A&E a OPC HDA stručně představena. Dalšími specifikacemi jsou například OPC Batch, OPC Security, OPC Data Exchange (OPC DX), OPC Compliance Tests [1] [5].

- **OPC Data Access (OPC DA)** – umožňuje periodickou výměnu dat mezi distribuovanými zařízeními. Server zprostředkovává informace v podobě uspořádané trojice obsahující konkrétní hodnotu, vlastnosti a časovou známku. Typicky se OPC DA používal pro komunikaci mezi systémy různých výrobců nebo pro přenos dat do HMI či jiných monitorovacích systémů.
- **OPC Alarm & Events (OPC A&E)** – je specifikace založená na událostmi řízené komunikaci, kdy server zpřístupňuje informace v podobě notifikací. Klient má možnost se na serveru přihlásit k odběru určité události, a jakmile k ní dojde, server pošle upozornění všem příslušným klientům/odběratelům. OPC A&E definuje tři typy událostí – základním je *OPCSimpleEvent*, ze

kterého jsou odvozeny *OPCConditionEvent* a *OPCTrackingEvent*. Poslední zmíněný typ rozšiřuje *OPCSimpleEvent* o možnost sledování spouštěče události. *OPCConditionEvent* upozorňuje odběratele na výskyt události za specifikovaných podmínek.

- **OPC Historical Data Access (OPC HDA)** – využívají klienti pro přístup k starším datům uložených na serveru [1].

Používání původní verze OPC má v současných automatizačních systémech několik nevýhod. Jelikož OPC využívá technologii COM/DCOM, byly klientská aplikace a server omezený pouze na operační systém Windows nebo na fungování v rámci lokální sítě. Zároveň byla komunikace realizována primárně proprietárními protokoly. K tomu samotný COM/DCOM má limity v kompatibilitě s jinými verzemi Windows i v zabezpečení. To se řešilo dalšími tunelovacími nástroji, které umožnily průchod přes firewall, anebo komplikovanými DCOM konfiguracemi. Další nevýhodou byl jednoduchý datový model, který nepodporoval komplexnější datové struktury. Rozhraní bylo navíc dostupné jen pro C++ aplikace, takže aplikace v JAVA nebo .NET se musely nejdříve zabalit do další vrstvy, aby se k příslušným datům dostaly. Opět to vedlo k navyšování komplikovanosti programu a pracovní zátěže [2, 8].

Za účelem odstranit výše popsané nedostatky, definovala OPC Foundation v roce 2003 novou, ucelenou specifikaci s názvem OPC Unified Architecture (OPC UA), která se pro snazší světovou adopci později standardizovala jako IEC 62541 (2012). V OPC UA je původní standard kompletně přepsán tak, aby vyhovoval nárokům na současnou průmyslovou komunikaci. To znamená, že je vhodný pro M2M komunikaci i pro komunikaci napříč úrovněmi automatizačního systému. Standard je nezávislý na platformě a výrobci, čemuž pomáhá skutečnost, že je postaven na architektuře orientované na služby (SOA, Service-Oriented Architecture). Dále zajišťuje bezpečnou a spolehlivou výměnu dat díky integraci nejnovějších bezpečnostních mechanismů [5].

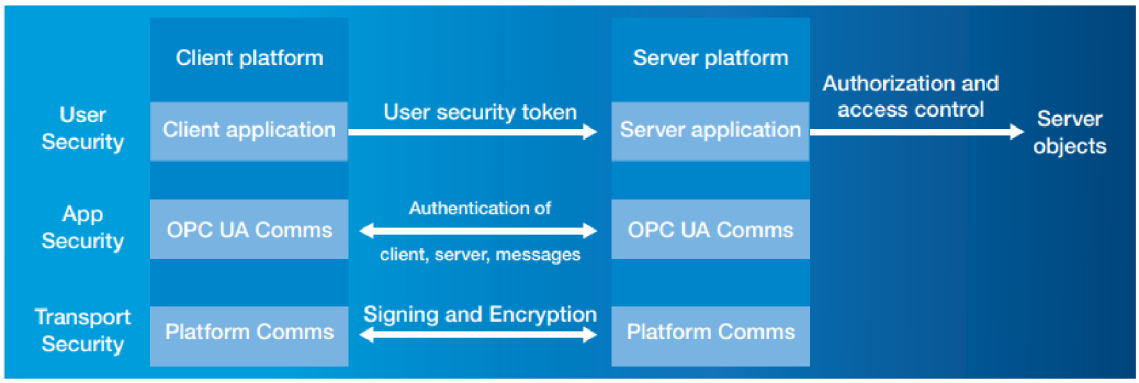
Jednou z významnějších vlastností OPC UA je, že definuje několik standardizovaných informačních modelů. Hlavní informační model zprostředkovává základní objekty, proměnné, reference a datové typy. Navrch k němu OPC UA ještě definuje čtyři další informační modely:

- **Data Access (DA)** – popisuje modelování real-time dat. Tj. dat reprezentujících aktuální stav nebo chování procesních parametrů. Zahrnuje to definice jak analogových, tak diskrétních proměnných. Zdrojem takovýchto dat jsou senzory, PLC, enkodéry a další.
- **Alarm and Conditions (AC)** – informační model definující jakým způsobem zpracovat určité stavy a události.
- **Historical Access (HA)** – umožňuje klientovi přístup k starším hodnotám proměnných a událostí. Ty se mohou číst, modifikovat i zapisovat. Mohou být uloženy v databázi, archivu nebo jiném úložišti.

- **Program** – OPC UA metody se používají pouze pro spuštění operací, které po provedení okamžitě skončí. Některé operace však vyžadující delší čas exekuce, a právě pro využívají OPC UA Program [1].

3.1 Bezpečnost

Významnou roli hrála při vývoji OPC UA bezpečnost. V samotném základu standardu jsou zabudované ověřené bezpečnostní mechanismy, které se využívají i pro internetovou komunikaci, např. SSL, TLS a AES. V rámci nich je zahrnuta autentizace, autorizace a šifrování, čímž se zaručí bezpečné připojení a komunikace a zabrání se přístupu neoprávněných entit. Dále OPC UA zajišťuje nastavitelné časové prodlevy (time-out) a automatickou detekci chyb, což umožňuje v případě potřeby obnovení komunikace mezi klientem a serverem bez ztráty dat. Jak naznačuje Obr. 6, OPC UA zabezpečení je rozdělené do tři vrstev:



Obr. 6: Úrovně zabezpečení [8]

- **OPC UA User level security**. Tyto mechanismy se aktivují při sestavení aktivní relace. Klient pošle serveru zakódovaný bezpečnostní token, podle kterého server ověřuje identitu klienta a na základně toho případně zpřístupní příslušné objekty ve svém OPC UA rozhraní. OPC UA nicméně nespecifikuje přesné autorizační mechanismy, jejich volba záleží na konkrétní aplikaci.
- **Application level security**. Také součástí sestavování relace. Zajišťuje výměnu digitálně podepsaných certifikátů.
- **OPC UA Transport level security**. V této úrovni zabezpečení jde primárně o integritu dat, která zabraňuje odposlouchávání a modifikaci přenášených informací [5].

Pro sestavení relace mezi klientem a serverem se vytváří tzv. Secure Channel, který je charakterizován dvěma parametry:

- **SecurityMode** – určující zabezpečení zprávy:
  None – zprávy nejsou zabezpečeny,
  Sign – zprávy mají digitální podpis,
  Sign&Encrypt – zprávy jsou podepsány a navíc zašifrovány.

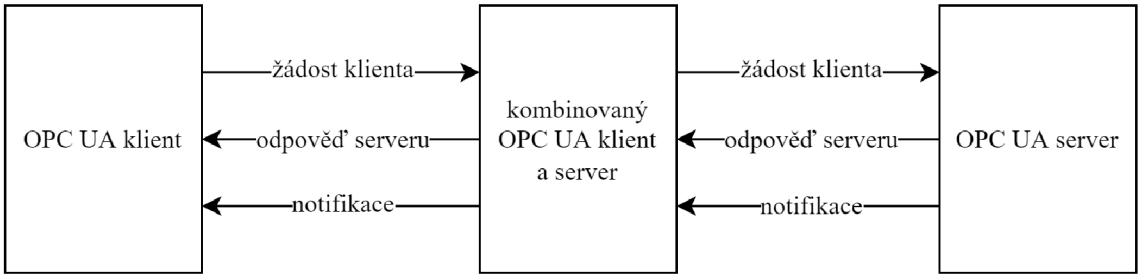
- **SecurityPolicy –** specifikující šifrovací algoritmy:
  None – bez zabezpečení,
  Basic128Rsa15 – použití zabezpečovací algoritmů,
  Basic256 – použití zabezpečovacích algoritmů s delšími klíči [6].

Dalším bezpečnostním prvkem jsou tři možnosti pro ověření uživatele:

- **Anonymous** – ověření není potřeba,
- **User Password** – uživatel se musí ověřit uživatelským jménem a heslem,
- **Certifikát** – ověření uživatele je provedeno pomocí certifikátu [7].

3.2 Model Klient/Server

Základním komunikačním modelem, ze kterého OPC UA vychází, je klient/server. Průmyslové zařízení figurující jako server shromažďuje informace, ke kterým se potřebuje dostat jiné zařízení či osoba prostřednictvím klientské aplikace. Každý systém může figurovat jak jako klient, tak jako server, jak je znázorněno na Obr. 7, přičemž klient je schopen komunikovat s více servery, podobně jako server s více klienty [8].



Obr. 7: OPC UA klient/server systém [11, přeloženo, upraveno]

Pro tento způsob komunikace byly definovány služby, které klient volá a server zprostředkovává. Služby jsou dále rozděleny do několika logických skupin, tzv. Service sets [8].

- **Discovery Service set** – služba definující funkce pro vyhledávání serverů a připojení [11].
- **SecureChannel service set** – služby stanovující konfiguraci zabezpečení serveru a založení komunikačního kanálu, který zaručí diskrétnost a celistvost přenosu dat. Tyto služby nejsou v OPC UA aplikaci implementovány přímo, jsou ale poskytnuty v komunikačním zásobníku.
- **Session service set** – služby pro sestavení spojení (relace) na úrovni sedmé aplikační vrstvy.
- **NodeManagement service set** – služby poskytující rozhraní pro konfiguraci serverů. Umožňují klientům spravovat (přidávat, upravovat a mazat) uzly v address space.
- **View service set** – služby umožňující prohledávání uzlů v *address space.*

- **Attribute service set** – služby ke čtení a zapisování.
- **Method service set** – slouží k volání metod.
- **MonitoredItem service set** – služby sloužící k monitorování prvků v address space.
- **Subscription service set** – využívá se ke generování, úpravě a mazání zpráv pro MonitoredItems.
- **Query service set** – služby, které umožňují klientovi si vyselektovat uzly z address space podle konkrétního kritéria [5].

Server uchovává veškerá data v podobě uzlů (nodes), které jsou propojeny referencemi a vytváří strukturovanou síť označovanou jako address space [6]. Síť nemusí mít nutně hierarchickou strukturu, velice často jde o mesh topologii [9].

Uzly mohou reprezentovat informaci jakéhokoliv druhu a připadají právě do jedné kategorie popsané v Tab. 1.

Tab. 1: Stručný přehled kategorií uzlů [2]

Kategorie	**Popis**
Base NodeClass	Základní třída, ze které jsou všechny ostatní odvozeny.
Object NodeClass	Objekty reprezentující reálné entity jako systémové komponenty, hardwarové či softwarové komponenty.
ObjectType NodeClass	Definuje typy objektů.
Variable NodeClass	Proměnné, které se používají k vyjádření hodnoty systému.
VariableType NodeClass	Definuje typy proměnných.
DataType NodeClass	Definuje typy hodnot, kterých proměnné nabývají.
ReferenceType NodeClass	Definuje různé typy referencí mezi uzly.
Method NodeClass	Metody sloužící k definování funkcí, které iniciují nějakou akci. Jsou volány v rámci serveru.
View NodeClass	Definuje funkce, které zobrazí pouze určitou *část address space.*

V závislosti na kategorii jsou uzly popsány sérií atributů charakterizujícími jejich vlastnosti. Jejich seznam je fixní a nemůže být uživatelem modifikován. Existuje několik atributů, které jsou pro všechny kategorie uzlů stejné:

- **NodeId** – unikátní identifikátor v rámci address space,
- **NodeClass** – definuje třídu/kategorii uzlu,
- **DisplayName** – text, který může klient použít při zobrazení jako název,
- **BrowseName** – identifikuje uzel při prohledávání *address space* [2, 8].

K serveru se dá připojit přes jeho URL adresu, např. „opc.tcp://192.168.0.3:4840“, kde:

- „opc.tcp“ je identifikátor protokolu,
- „192.168.0.3“ je IP adresa rozhraní PROFINET a
- „4840“ je číslo TCP portu. Zmíněný port se používá standardně, nicméně je možné ho změnit.

Před navázáním spojení mezi klientem a serverem se klient dotazuje na požadovanou míru zabezpečení. Na tento dotaz server odpoví seznamem všech jím uznávaných nastavení zabezpečení, tzv. end points. Ukázkový end point může být např. „opc.tcp“://192.168.0.3:4840 – [SignAndEncrypt:Basic256Sha256:Binary], který je tvořen pěti částmi:

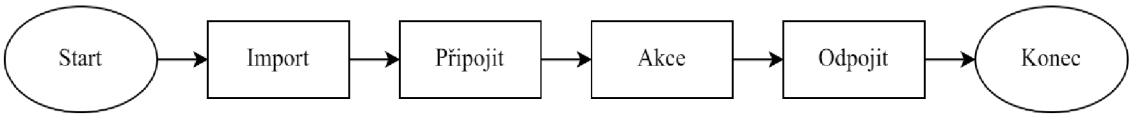
- Identifikátor pro OPC: „opc.tcp“,
- IP adresa: např. 192.168.0.3,
- Číslo portu: 4840 (standardně užívaný port, jinak číslo portu se dá změnit v konfiguraci),
- *Message Security Mode* (nastavení zabezpečení pro zprávy): SignAndEncrypt,
- *Security Policy* (kódování a hashování): Basic256Sha256:Binary.

Navázání spojení je možné pouze v případě, že klient dodržuje požadovaná bezpečnostní nastavení. Pokud se pro bezpečnou komunikaci používá některý z certifikátů, je důležité mít nastavené aktuální datum a čas, jinak se může, byť validní certifikát, vyhodnotit jako neplatný.

3.3 Klientská aplikace

Komunikační zásobníky pro vývoj OPC UA aplikací jsou v současnosti dostupné v ANSI C/C++, .NET a JAVA. Každé ze zmíněných SDK má své výhody ve vztahu k cílové platformě a požadavkům aplikace [10]. OPC UA standard lze implementovat ve všech programovacích jazycích [5].

Na Obr. 8 je znázorněn vývojový diagram popisující princip primitivní klientské aplikace. V textu následují Obr. 9 a Obr. 10 s korespondujícími ukázkami kódů v jazycích C# a Python 3. Prvním krokem je implementace vhodných knihoven. To zajišťuje příkaz `using Opc.UaFx.Client` (C#), resp. `from opcua import Client` (Python 3). Další příkazy slouží k připojení k serveru. V ukázkových kódech je použita adresa serveru „opc.tcp://127.0.0.3:4840“. Tuto adresu používá pomocný OPC UA server napsaný v jazyce Python 3, který je možné nalézt v Příloze D, adresu je ale samozřejmě možné změnit. Když je klient připojen k serveru, má možnost volat různé služby. V ukázkových kódech je použita funkce pro čtení. Klientská aplikace se přihlásí ke konkrétnímu uzlu pomocí jeho nodeID. V tomto případě se jedná o uzel s ID ns=2;i=2, který je definován v pomocném serveru v Příloze D. Klient přečtenou hodnotu uzlu vypíše na konzoli a na závěr se ze serveru odpojí.



Obr. 8: Vývojový diagram klientské aplikace

C#

```
using System;
using Opc.UaFx.Client;

OpcClient client = new OpcClient(“opc.tcp://127.0.0.3:4840”);
client.Connect()

var node = client.ReadNode(“ns=2;i=2”)
Console.WriteLine(“Value: ” + node);

client.Disconnect();
```

Obr. 9: Ukázka kódu primitivní klientské aplikace v jazyce C#

Python

```
from opcua import Client

client = Client(“opc.tcp://127.0.0.3:4840”)
client.connect()

node = client.get_node(“ns=2;i=2”)
readValue = node.get_value()
print(readValue)

client.disconnect()
```

Obr. 10: Ukázka kódu primitivní klientské aplikace v jazyce Python 3

3.4 Model Publisher/Subscriber

V případě komunikačního modelu klient/server je potřeba brát v úvahu jisté zpoždění. Jenže právě rychlost přenosu je v některých průmyslových aplikacích kritická. Proto OPC UA podporuje i komunikační model Publisher/Subscriber (PubSub), který v závislosti na použité technologii dokáže zajistit kratší doby cyklu. V poslední době se často spojuje s Time-Sensitive Networking (TSN), a tím tak zajistit i přenos dat v časově kritických aplikací. PubSub je komunikace typu one-to-many nebo many-to-one. Rozdíl mezi tímto modelem a modelem klient/server je způsob výměny dat. Publisher (vydavatel) a subscriber (odběratel) si nevyměňují data napřímo ale prostřednictvím Message Oriented Middleware (MOM). Této entitě posílá publisher svá data bez potřeby znalosti odběratelů, což platí i naopak – subscriber se připojí na dané rozhraní a nepotřebuje už vědět, od koho informace pochází. Lokální model klient/server je základem pro PubSub komunikaci [2, 14]. OPC UA aplikace podporují různé typy MOM infrastruktury, protokoly s IP multicast a další protokoly jako DDS nebo MQTT, které mohou být do modelu PubSub implementovány [8].

4 TIA PORTAL

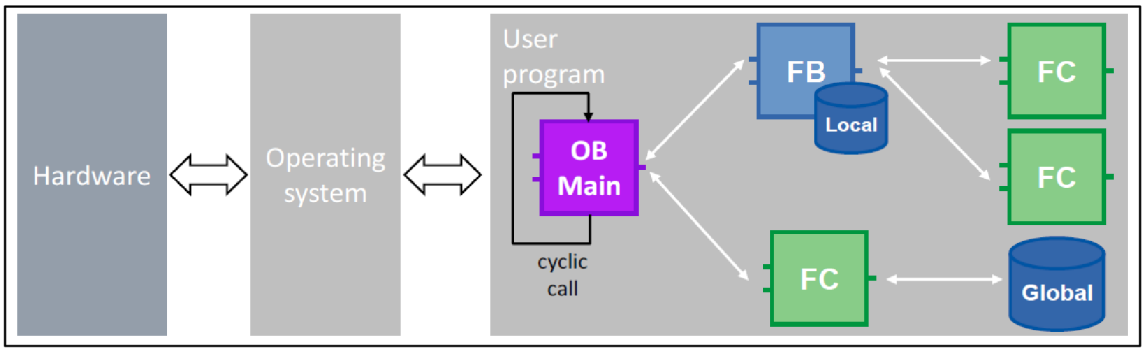
V devadesátých letech představila společnost Siemens plně integrovanou automatizaci (Totally Integrated Automation neboli TIA). Koncept se opírá o myšlenku propojení reálného světa se světem digitálním, které zajistí flexibilnější a efektivnější tok dat napříč celým podnikem. Díky tomu se snadněji sledují nejrůznější procesní parametry, jejichž včasná regulace pak může mít pozitivní vliv například na spotřebu zdrojů nebo životnost zařízení [11].

V roce 2010 byl na trh uveden TIA Portal – vývojové prostředí pro řídicí systémy generace SIMATIC, které usnadnilo realizaci konceptu TIA. Inženýrský softwarový systém umožňuje projektování a konfiguraci nejen řídicích systémů včetně decentralizované periferie (I/O), operátorských rozhraní pro strojní celky (HMI) a dispečerských systémů (SCADA), ale i komunikačních sítí a elektrických pohonů. Vkládání jednotlivých zařízení do projektu je zjednodušeno díky integrované databázi automatizačních komponent. TIA Portal dále disponuje datovou databází, díky níž je zajištěna soudržnost dat v celém automatizačním projektu. Zároveň daný projekt přirozeně strukturuje, což přispívá k určité flexibilitě, jelikož lze jeho jednotlivé části bez větších komplikací upravovat, doplňovat nebo odebírat. Tyto dvě zmíněné charakteristiky umožňují snadnou diagnostiku, údržbu a servis automatizační techniky v běžném provozu [16, 17].

Vývojové prostředí je sestaveno z několika dílčích softwarů. Klíčovou částí je STEP 7, který je určen pro konfiguraci PLC SIMATIC S7-1200/1500, S7-300/400. Dále je součástí TIA Portal WinCC, který podporuje konfiguraci panelů nebo průmyslových PC SIMATIC. TIA Portal je licencovaný software, přičemž uživatel si podle svých preferencí (aplikace) může vybrat mezi několika variantami [12].

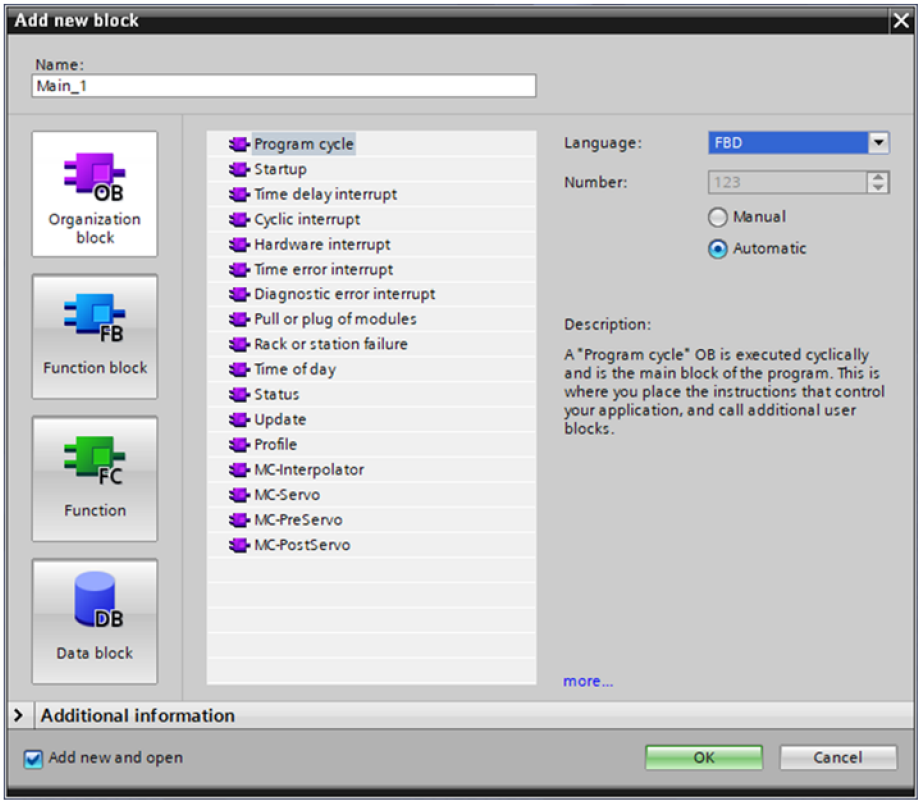
4.1 Struktura programu

Software v řídicích systémech SIMATIC má dvě části. Zaprvé je to operační systém, který řídí funkce a sekvence obecného rázu, například správu paměti nebo chybové stavy. Zadruhé je to samotný uživatelský program vytvořený speciálně pro účely dané automatizační úlohy. Jak je patrné ze schématu na Obr. 11, TIA Portal umožňuje při programování dodržovat jasnou strukturu kódu, a to za pomoci bloků, jež jsou několika typů [13].



Obr. 11: Operační systém a uživatelský program [19]

- **OB – Organizační blok** (Organization block). Organizační bloky fungují jako rozhraní mezi uživatelským programem a operačním systémem, jímž jsou volány. Existuje několik typů OB, přičemž každý má svá specifika (výčet všech OB je vidět na Obr. 12). Základním OB uživatelského programu je tzv. *Program cycle*, který je spouštěn cyklicky a do projektu v TIA Portal přidán automaticky s každým nově vloženým PLC. Každý OB má své jedinečné číslo, které určuje pořadí, v němž se spustí. Toto číslování se generuje automaticky, nicméně ho lze podle potřeby změnit manuálně. Počet OB v projektu je omezen. Protože se čísla liší model od modelu, Tab. 2 je znázorněn stručný přehled.



Obr. 12: Přidání nového Organizačního bloku

Tab. 2: Počet organizačních bloků pro modely S7-1200 a S7-1500 [19]

Typ organizačního bloku	S7-1200	S7-1500	Výhody
Cyclic OB, Startup OB	100	100	Modularizace uživatelského programu
Hardwarové přerušení	50	50	Možnost oddělených OB pro každou událost
Přerušení zpožděním	4	20	Modularizace uživatelského programu
Cyklické přerušení	4	20	
Časované přerušení	-	20	

- **DB – Datový blok** (Data block). Datové bloky slouží k ukládání dat. Ve STEP 7 (TIA Portal) se pracuje s dvěma typy DB/paměti – globální a lokální. Data v globálních DB jsou přístupná ze všech míst projekt, zatímco lokální paměť je navázaná na konkrétní FB, což je taky jediné místo, kde se dá měnit její obsah.
- **FB – Funkční blok** (Function block). Funkční bloky disponují úložištěm (tj. instance DB, která se automaticky vygeneruje při použití daného FB), kam se v pravidelných intervalech ukládají data deklarovaná v rozhraních *Input*, *Output*, *InOut*, *Static* a *Temp*. Poslední zmíněné je ale součástí volatilní paměti, takže příslušné proměnné je potřeba inicializovat při každé běhu cyklu.
- **FC – Funkce** (Function). V případě funkcí nedochází k průběžnému ukládání dat. To znamená, že hodnoty parametrů, s nimiž se v příslušné FC pracuje, je nutné při každém volání znovu nastavit [13].

Jednotlivé bloky se kromě svých funkcí liší také ve velikostech, které jsou opět pro každý model SIMATIC různé. Přehled je znázorněn v Tab. 3.

Tab. 3: Velikosti a maximální počet bloků pro SIMATIC [19]

Typ bloku	Velikost/počet	S7-300/400	S7-1200	S7-1500
DB	Max. velikost	64 kB	64 kB	64 kB (16 MB pro optimalizované CPU 1518)
	Max. počet	16 000	65 535	65 535
FC/FB	Max. velikost	64 kB	64 kB	512 kB
	Max. počet	7 999	65 535	65 535
FC/FB/DB	Celkový max. počet	4 096 (CPU319) 6 000 (CPU412)	1 024	10 000 (CPU 1518)

Rozhraní *In, InOut* a *Out* slouží ke vstupu a výstupu dat do FB a FC. K takovémuto převodu dat existují dvě metody. *Call-by-value*, kdy při volání bloku se hodnota daného parametru zkopíruje do proměnné deklarované v lokální paměti, kde je pro to vyhrazeno

místo. Druhý způsob je tzv. *Call-by-reference*. V tomto případě není nutné vyhrazovat část paměti, jelikož do bloku vstupuje pouze odkaz na konkrétní místo v globální paměti. Je dobrou praxí zajišťovat výměnu dat mezi bloky pouze pomocí jejich rozhraní a nezasahovat do lokálních pamětí. Udržuje to totiž jistou strukturu programu, která je pak výhodou při opětovném využívání části kódu nebo třeba při jeho upravování a rozšiřování [13].

4.1.1 Knihovny

TIA Portal umožňuje tvorbu vlastních knihoven, kam lze ukládat konfigurace jednotlivých zařízení jako PLC a HMI, řídicí bloky, PLC tags, nastavení tag tables a watch tables atd. Rozlišují se dva typy knihoven:

- **Projektová knihovna** (Project library), která je dostupná pouze v rámci projektu.
- **Globální knihovna** (Global library), která je použitelná pro různé projekty.

Obsah knihoven se ukládá do úložišť dvojího typu – *Master copies* a *Types*. Master copies jsou kopie konfigurací různých prvků, například řídicích bloků, hardwaru nebo tag tables, a v projektu nejsou s ničím provázány. Naproti tomu úložiště typu Types jsou napojena na konkrétní místa v projektu, kde se používají. To znamená, že jestliže se v rámci knihovny provede jejich změna, promítne se to automaticky v celém projektu [13].

4.2 Programovací jazyky

Při tvorbě programu/jednotlivých bloků (OB, FB, FC) v TIA Portal má uživatel možnost si zvolit jeden z následujících jazyků, které vývojové prostředí podporuje:

- Ladder diagram (LAD),
- Function block (FBD),
- Structured Control Language (SCL),
- Graph (v S7-1200 nepodporován),
- Statement list (STL) (v S7-1200 nepodporován).

V případě řídicích systémů S7-1200 a S7-1500 se všechny jazyky kompilují přímo do strojového kódů. U modelů S7-300 a S7-400 to platí pouze pro jazyk SCL. Jazyky LAD a FBD se nejdříve převedou na STL a až následně do strojového kódu [13].

4.3 Datové typy

V následujících tabulkách je přehled standardních datových typů podporovaných v TIA Portal. Tab. 4 ukazuje výčet základních datových typů v souvislosti s normou EN 61131-3. V Tab. 5 je seznam strukturovaných datových typů.

Tab. 4: Základní datové typy [19]

Typ	Popis	S7-300/400	S7-1200	S7-1500
Bit	BOOL	✓	✓	✓
	BYTE			
	WORD			
	DWORD			
	LWORD	-	-	✓
Charakter	CHAR (8 bit)	✓	✓	✓
Čísla	INT (16 bit)	✓	✓	✓
	DINT (32 bit)			
	REAL (32 bit)			
	SINT (8 bit)	-	✓	✓
	USINT (8 bit)			
	UINT (16 bit)			
	UDINT (32 bit)			
	LREAL (64 bit)			
	LINT (64 bit)	-	-	✓
	ULINT (64 bit)			
Čas	TIME	✓	✓	✓
	DATE			
	TIME_OF_DAY			
	S5TIME	✓	-	✓
	LTIME	-	-	✓
	L_TIME_OF_DAY			

Tab. 5: Seznam strukturovaných datových typů [19]

Typ	Popis	S7-300/400	S7-1200	S-1500
Čas	DATE_AND_TIME	✓	-	
	DTL	-	✓	✓
	L_DATE_AND_TIME	-	-	✓
Charakter	STRING	✓	✓	✓
Pole	ARRAY	✓	✓	✓
Struktura	STRUCT	✓	✓	✓

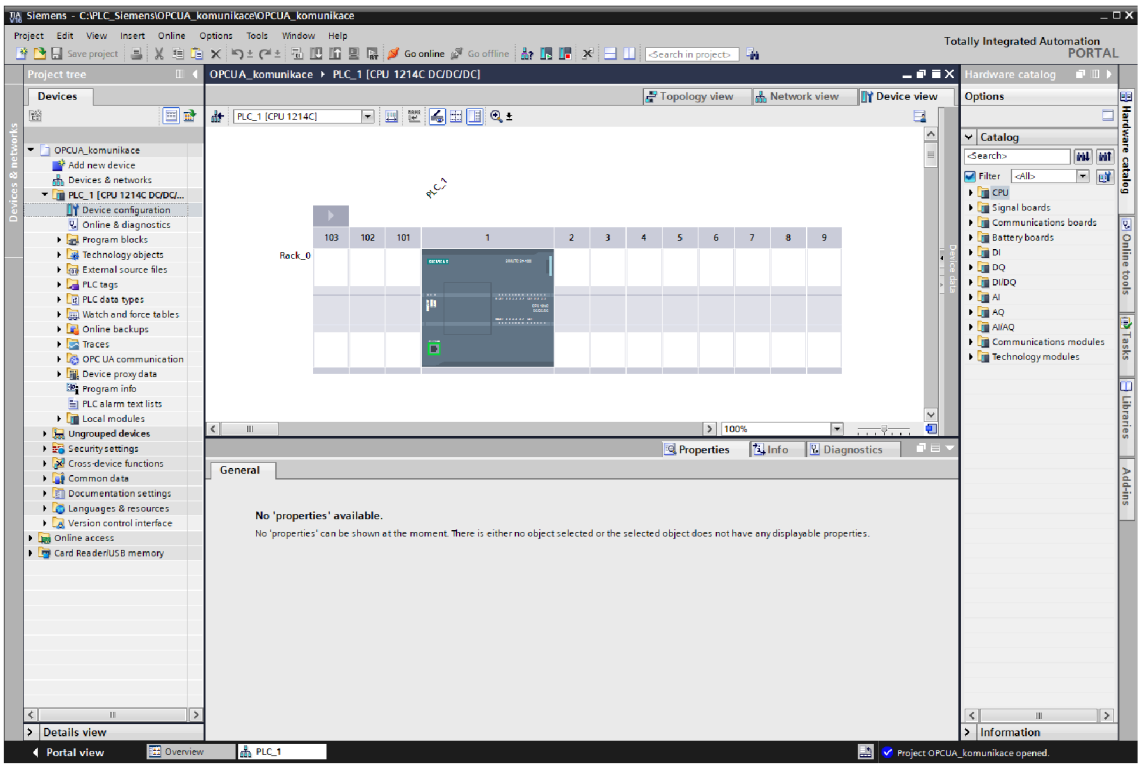
Array reprezentuje datovou strukturu tvořenou několika prvky stejného datového typu, které je možné adresovat pomocí indexu. Naproti tomu *Struct* je struktura, kterou můžou tvořit prvky různých datových typů. Takovéto prvky se deklarují centrálně a jsou následně dostupné z celého projektu. Výhodou tak je, že jejich případnou změnu stačí

provést pouze v místě deklarace, protože ostatní instance se zaktualizují automaticky [13].

4.4 Nový projekt

TIA Portal má dva základní pohledy – *Portal view* a *Project view*, který je znázorněn na Obr. 13. Portal view se zobrazí po spuštění aplikace a nabízí komplexnější pohled na celou úlohu. Umožňuje do projektu přidávat zařízení, stejně tak i řídicí bloky. Dále se zde nachází například správa vizualizace nebo online diagnostiky. K detailnější práci v projektu slouží Project view.

V Portal view přes *Start* → *Create new project* se založí nový projekt, který je do jisté míry možné spravovat právě v Portal view. Protože mnohé funkce jsou dostupné i v Project view, omezí se další popis pouze na tento pohled.



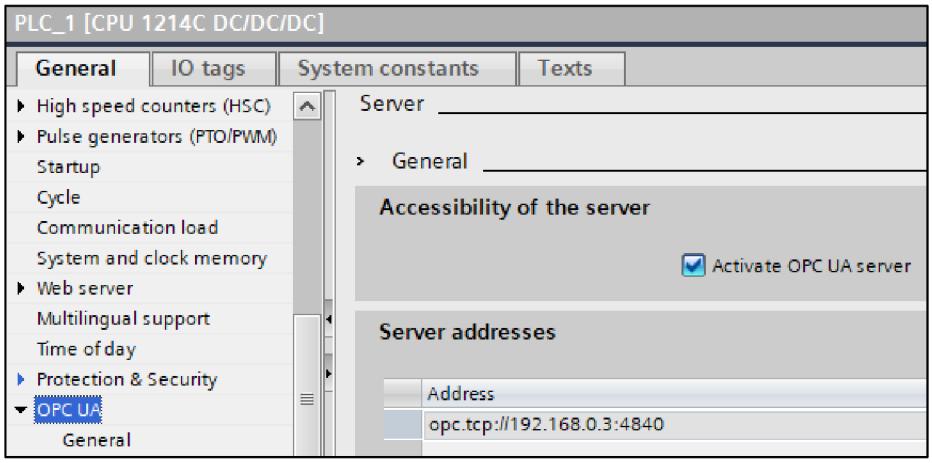
Obr. 13: Project view

Standardně je v nově založeném projektu prvním krokem přidání nového zařízení, a to v *Project tree* pomocí *Add new device*. Následně se zobrazí nabídka PLC a HMI, ze které lze vybírat. V rámci této bakalářské práce se pracovalo se SIMATIC S7-1200 CPU 1212C DC/DC/DC. Potvrzením výběru se zařízení zobrazí v Project tree, kde se objeví další funkce pro práci se zařízením, např. přidání funkčních bloků, správa PLC tags, watch tables a force tables aj. Podrobný postup je popsán v Příloze A.

4.5 Konfigurace OPC UA serveru na SIMATIC S7-1200

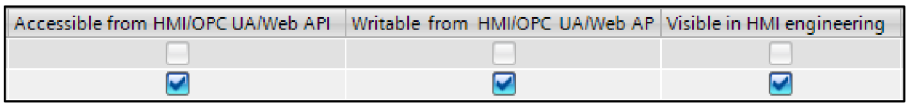
Řídicí systémy S7-1200 s firmwarem V4.4 a vyšším disponují OPC UA serverem, k němuž je možný přístup prostřednictvím PROFINET rozhraní integrovaného v CPU. Z bezpečnostních důvodů není ve výchozím nastavení na S7-1200 OPC UA server aktivován. Výhodou je, že je-li server jednou aktivován a následně deaktivován, konfigurace zůstane v projektu uložena, což ulehčuje práci při opětovném spouštění. Podrobný návod lze najít v Příloze B. Postup je následující:

U daného PLC v Device configuration zobrazit *Properties → General → OPC UA* a v sekci General zaškrtnou pole „Activate OPC UA server“, viz Obr. 14. Název aplikace se generuje automaticky, nicméně v případě potřeby je jej možné změnit přes *Properties → General → Application name*.



Obr. 14: Aktivace OPC UA serveru

- V sekci Security přidat serverový certifikát, který slouží pro ověřování identity serveru a určuje míru zabezpečení.
- Nastavit správnou Runtime License. V případě S7-1200 je na výběr pouze jedna v porovnání s S7-1500, kde jsou k výběru tři.
- Na rozdíl od S7-1500, S7-1200 nepodporuje standardní serverové rozhraní, takže není dostupná možnost „Activate standard SIMATIC server interface“, proto se musí rozhraní OPC UA serveru přidat ručně. A to přes *Project tree → OPC UA communication*.
- Serverové rozhraní umožňuje definovat uzly, které jsou OPC UA klientům přístupné, a to jak pro čtení, tak pro zapisování, jak je vidět na Obr. 15. U každého tagu se dají tato práva ještě individuálně omezit pouze na jedno z nich. V případě, že se u nějakého tagu odeberou jak práva na čtení, tak práva na zapisování, OPC UA server tento tag odstraní ze svého address space. Pro S7-1200 s firmwarem V4.4 nejsou v rozhraní OPC UA serveru podporovány strukturované datové typy (Struct, Array).



Obr. 15: Nastavení práv OPC UA klienta

- *Optimized access* pro DB v projektu musí být deaktivovaný [6].
- Na závěr se projekt musí zkompilovat. Dále je důležité znovu do PLC nahrát hardwarovou konfiguraci.

Aktivovaný OPC UA server se na S7-1200 spustí v okamžiku nahrání projektu do CPU a zůstane spuštěný i případě, kdy se CPU přepne do STOP modu, což znamená, že OPC UA klienti mají stále přístup k požadovaným informacím. Přístupné jsou ale pouze ty informace, které byly aktuální v momentu změny z RUN do STOP modu. Pokud klient posílá serveru nové hodnoty proměnných, server je přijme a při jejich následném čtení se klientovi zobrazí právě tyto nově zaslané hodnoty, nicméně kvůli STOP modu je CPU nezpracuje [6].

Ke každému OPC UA serveru se přiděluje název aplikace (*application name*), který má definovaný formát skládající se ze dvou částí: „SIMATIC.S7-1200.OPC-UA.Application:“ + název CPU, na němž je server aktivován. Příkladem je „SIMATIC.S7-1200.OPC-UA.Application:PLC_1“. Název se v projektu generuje automaticky při aktivaci serveru, je ho ale možné v nastavení změnit [6].

4.5.1 Parametry OPC UA serveru na SIMATIC S7-1200

Následující tabulka Tab. 6 dává přehled parametrů OPC UA serveru aktivovaném na SIMATIC S7-1200 pro firmware V4.5.

Tab. 6: Parametry OPC UA serveru na SIMATIC S7-1200 [20]

Parametr	**Min.**	**Max.**
Samplovací interval [ms]	100	10 000
Interval publikace [ms]	200	20 000 000
Počet relací		10
Počet odběrů za relaci		50
Počet uzlů pro uživatelem definované serverové rozhraní		2 000
Počet monitorovaných uzlů		1 000

5 VLASTNÍ ŘEŠENÍ

V rámci této práce byly vytvořené dvě klientské aplikace a jeden PLC program. Použitý hardware a software je uveden v Tab. 7.

Tab. 7: Použitý hardware a software

Komponenta	Verze	Množství
PLC Siemens S7-1200 CPU 1214C DC/DC/DC 6ES7 214-1AG40-0XB0	Firmware 4.4	1
TIA Portal	V16 Basic	1
Visual Studio	Community 2022	1
Opc.UaFx Client	2.26.0 (k 13. 4. 2022)	1
IDLE	Python 3.10.0	1
opcua	0.98.13 (k 3. 3. 2021)	1

5.1 Projekt v TIA Portal

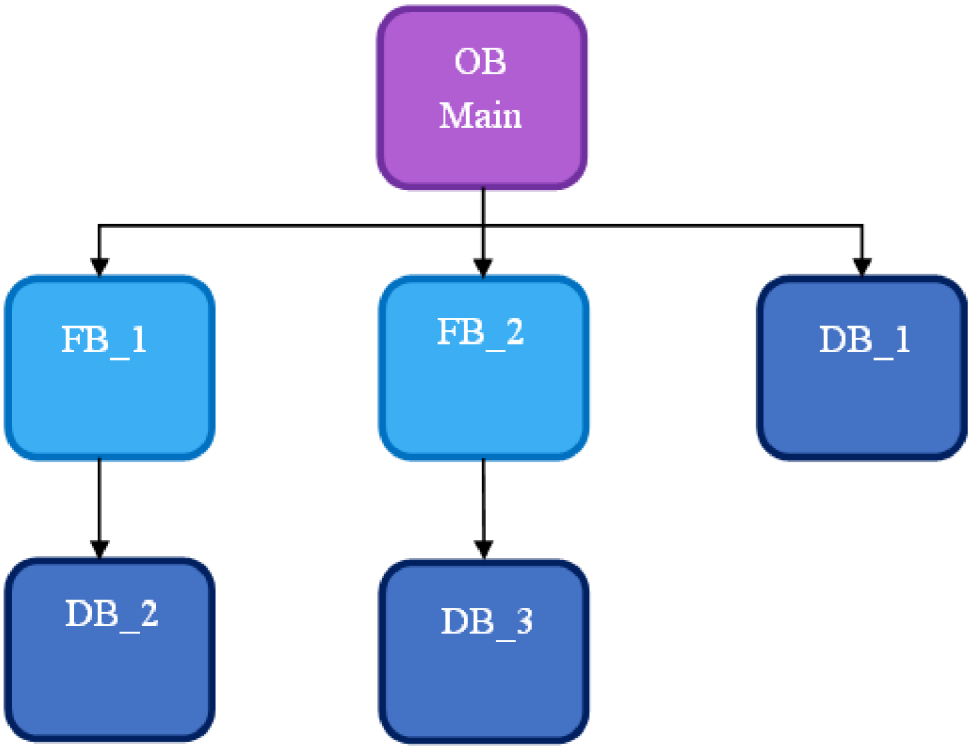
Pro potřebu otestování správné konfigurace serveru na straně PLC a funkčnosti klientských aplikací byl vytvořen jednoduchý PLC program, který je možné nalézt v Příloze 1. Na Obr. 16 je zobrazena struktura programu. Program tvoří jeden Organizační blok OB Main, pod který spadají dva Funkční bloky FB_1 a FB_2 (ke kterým patří Datové bloky DB_2 a DB_3) a jeden Datový blok DB_1.

V rámci hlavního cyklu (OB Main) se program rozhoduje zavolá-li FB_1 nebo FB_2. Rozhodování probíhá na základně hodnoty proměnné *programChoice*, která je uložena v DB_1. Proměnná je zpřístupněna přes OPC UA a to jak ke čtení, tak právě k zapisování. Je typu Int16 a předdefinovaná hodnota je nastavena na 1. Je-li proměnná rovna jedné, Main volá FB_1. Jakmile se změní *programChoice* na hodnotu dva, přepne se program z FB_1 na FB_2. Oba funkční bloky jsou naprogramovány tak, aby v různých sekvencích přiváděly jedničky a nuly na výstupy PLC.

Zmíněný DB_1 obsahuje proměnné napojené na měnící se výstupy, které jsou klientům zpřístupněny v OPC UA rozhraní. Zahrnuje to proměnné datového typu Bool, Int a String. Seznam příslušných proměnných je uveden v Tab. 8. Adresa serveru PLC je nastavena na „opc.tcp://192.168.0.3:4840.

Tab. 8: Proměnné v OPC UA rozhraní

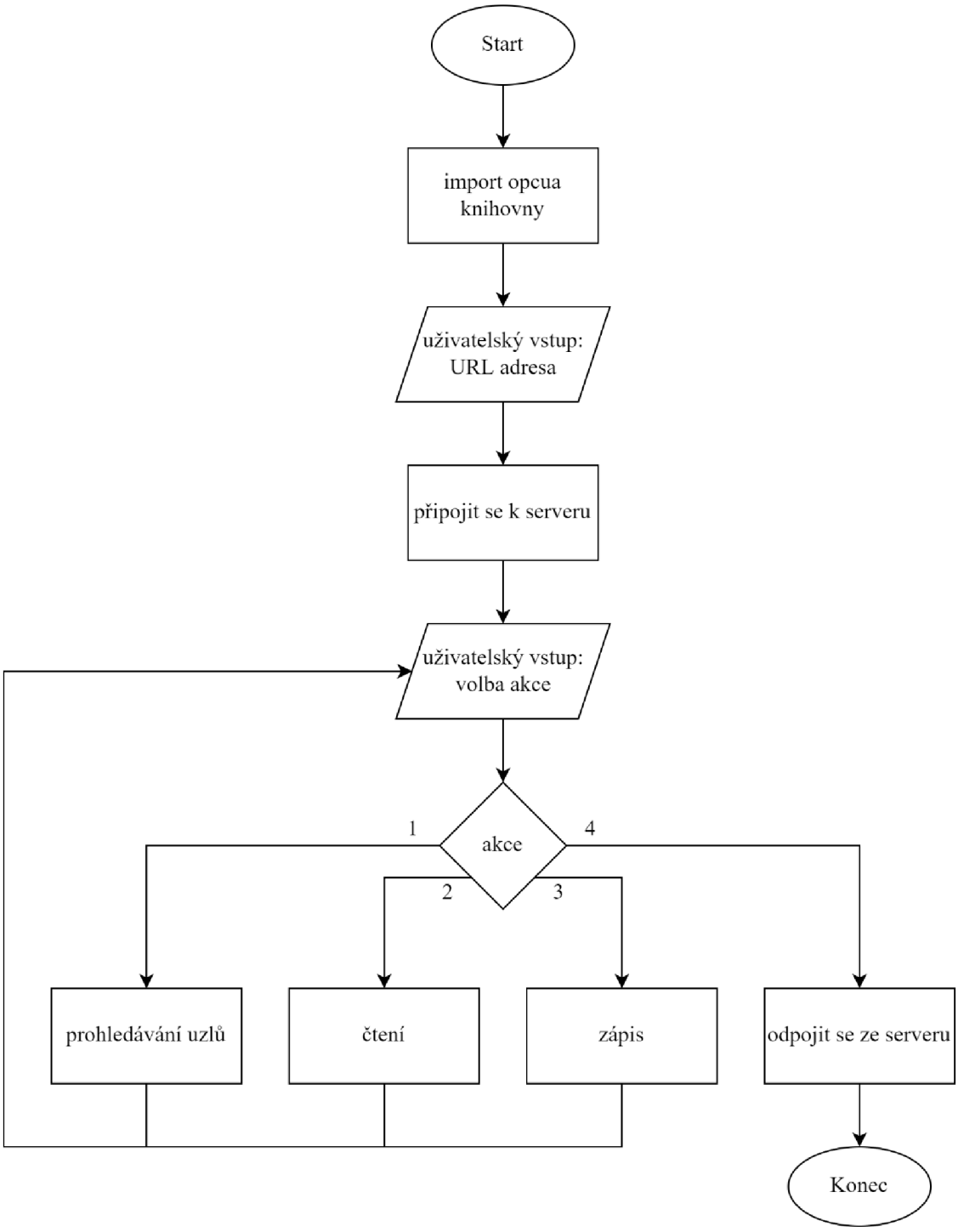
Proměnná	Datový typ	NodeID	Možnost čtení	Možnost zápisu
programChoice	Int16	ns=4;i=7	✓	✓
runningProgram	String	ns=4;i=8	✓	-
led0	Bool	ns=4;i=9	✓	-
led1	Bool	ns=4;i=10	✓	-
led2	Bool	ns=4;i=11	✓	-
led3	Bool	ns=4;i=12	✓	-
led4	Bool	ns=4;i=13	✓	-
led5	Bool	ns=4;i=14	✓	-



Obr. 16: Struktura PLC programu

5.2 Klientská aplikace v jazyce Python 3

První klient je vytvořen v jazyce Python 3 za použití knihovny opcua 0.98.13. Knihovna vytváří možnost nízko-úrovňového i vysoko-úrovňového programování. Protože jde o jednoduchou konzolovou aplikaci, která má za účel demonstrovat základní funkce pro komunikaci mezi OPC UA klientem a server, byly využity metody z vysoko-úrovňového programování. Vývojový diagram na Obr. 17 zobrazuje princip aplikace.



Obr. 17: Vývojový diagram klientské aplikace v Pythonu 3

Po spuštění klientské aplikace se v konzoli uživateli zobrazí příkaz k vložení URL adresy serveru, k němuž se následně pokusí připojit. Po úspěšném připojení si uživatel může vybrat z nabídky možných akcí:

- List of nodes (pozn. do páté úrovně),
- Read value of a node,
- Write new value,
- Disconnect.

Připojení k serveru i odpojení z něj (akce 4 – Disconnect) je řešeno metodami, které jsou použity v ukázkovém kódu na Obr. 10 v kapitole 3.3. Funkce čtení dává možnost si přečíst aktuální hodnotu uzlu, který si uživatel definuje pomocí jeho nodeID. Použitá metoda je taktéž v ukázkovém kódu na Obr. 10.

Další funkcí klientské aplikace je prohledávání address space, kterou zajistí akce označená číslem 1 – List of nodes. Program si najde uzly navazující na kořenový uzel (root node), tzv. objects nodes, které se dají vykládat jako vstupní body do sítě uzlů. K jejich vyhledání se použije metoda `get_objects_node()`. Následně se k nalezeným objects nodes vyhledají uzly s referencemi, a to pomocí metody `get_children()`. Tato metoda se dá použít ve vztahu s jakýmkoli uzlem, je-li požadováno najít propojené uzly, které jsou v sítí o úroveň níž. Program je napsán tak, aby vyhledal a vypsal všechny uzly do páté úrovně. Tato práce se zabývá primárně komunikací mezi PC a PLC S7-1200, na němž bylo nakonfigurováno serverové rozhraní, pro nějž pětiúrovňové prohledávání uzlů bylo dostačující.

Poslední, ještě nespecifikovanou akcí aplikace, je zapisování označené číslem 3 – Write new value. K tomu byly použité dvě metody. První z nich je `get_node()`, která najde uzel na základně uživatelem zadaného nodeID. Poté je použita druhá metoda, a sice `set_value()`, která podle požadavků uživatele přepíše hodnotu vybraného uzlu. Při přepisování hodnot uzlů je třeba brát v potaz datový typ uzlu. PLC program disponuje pouze jedinou proměnnou, která má povolené zapisování. Je datového typu Int16, proto je uživatelský vstup konvertován z typu String do Int16. Ke konverzi datových typů byla využita knihovna numpy.

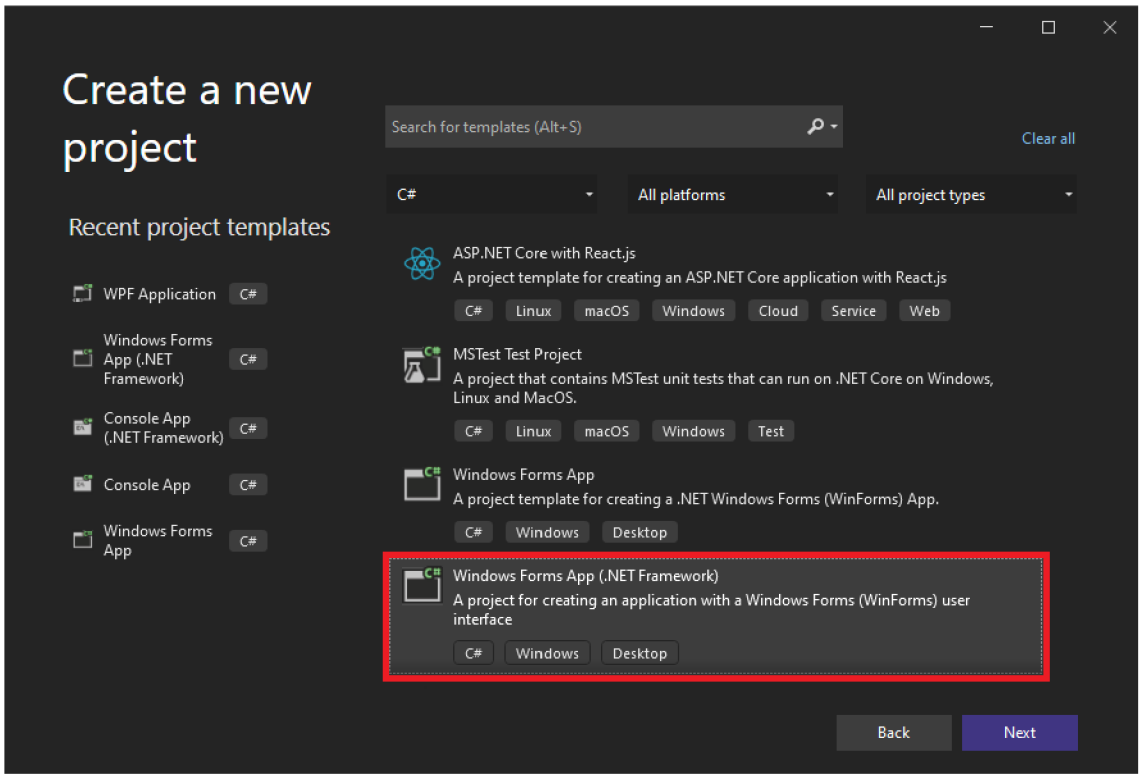
5.3 Klientská aplikace v jazyce C#

Na základě preferencí uvedených v charakteristice zadání byla druhá klientská aplikace vytvořena v jazyce C#. K samotné tvorbě bylo použito vývojové prostředí Visual Studio Community 2022. Z volně dostupných frameworků bylo vybráno OPC UA .NET SDK for Clients implementované jako NuGet Package Opc.UaFx.Client ve verzi 2.26.0 aktuální ke dni 13. dubna 2022. SDK spravované německou společností Traeger Industry Components GmbH[1] je založeno na .NET Framework a .NET Standard. Vytvořený kód lze nalézt v Příloze 3.

5.3.1 Windows Forms

Tento klient je vytvořen s GUI za použití Windows Forms. Obr. 18 zobrazuje způsob, jakým ve Visual Studiu vybrat právě tento typ projektu. Windows Forms je .NET framework k tvorbě aplikací prostřednictvím grafického designeru. Alternativou je WPF (Windows Presentation Foundation) s pokročilejší technologií.

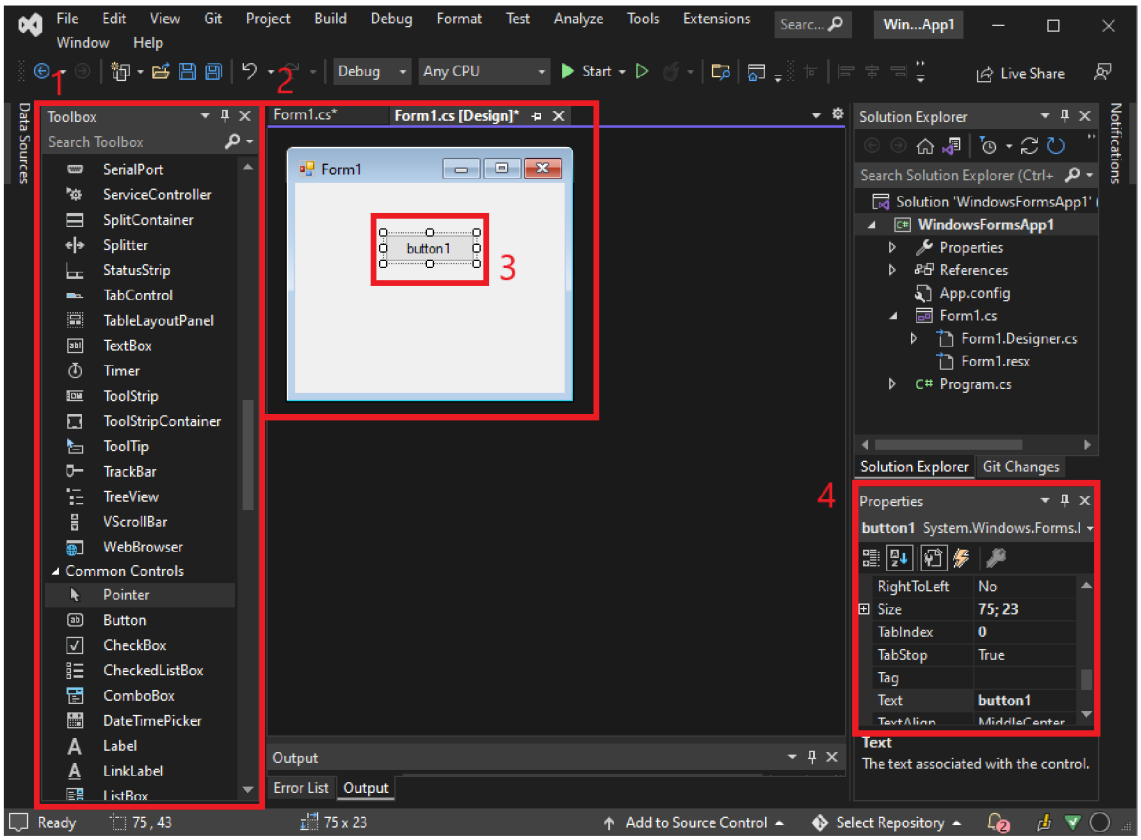
[1] *Společnost Traeger Industry Components GmbH se zabývá průmyslovou komunikací [21] a je členem OPC Foundation [22].*



Obr. 18: Založení Windows Forms projektu

Ve Visual Studiu se nabízí dvě zobrazení – grafický návrh (který je otevřen na Obr. 20) nebo zdrojový kód. Při návrhu aplikace lze využívat předpřipravených grafických nástrojů v panelu Tool Box (viz Obr. 20, blok 1). Pokud se Tool Box nezobrazuje automaticky, je možné ho zapnout přes záložku View → Tool Box, eventuálně klávesovou zkratkou Ctrl+Alt+x. Obr. 20, blok 2 zobrazuje formulář, kam stačí přetahovat vybrané prvky z Tool Boxu. Každý nástroj je možné si dle vlastních požadavků přizpůsobit a dále nastavit jeho parametry. K tomu slouží záložka Properties, která je zobrazena v bloku 4 na Obr. 20. Po vložení jakéhokoli nástroje se dvojklikem vytvoří událost. To znamená, že se automaticky vygeneruje část kódu – funkce, která daný prvek ovládá. Obsah kódu je už v rukou programátorů, kteří si ho přizpůsobí svým požadavkům. Pro demonstraci bylo do formuláře vloženo tlačítko (viz Obr. 20, blok 3).

Na následujícím Obr. 19 je zdrojový kód, který se vztahuje k formuláři na Obr. 20. Formulář je objekt třídy Form1 (kterou je možné přejmenovat) dědící z třídy Form. Třída Form1 se nachází ve více souborech, proto je označena klíčovým slovem `partial`. Během práce v návrháři se generuje zdrojový kód, který je oddělen v souboru Form1.Designer.cs. V ukázkovém kódu se na 10. řádku nachází metoda `InitializeComponent()`. Jedná se o konstruktor, který při spuštění aplikace vytvoří všechny prvky ve formuláři a přiřadí jim požadované vlastnosti.



Obr. 20: Nový Windows Form

```
1 using System;
2 using System.Windows.Forms;
3
4 namespace WindowsFormsApp1
5 {
6     public partial class Form1 : Form
7     {
8         public Form1()
9         {
10            InitializeComponent();
11        }
12
13        private void button1_Click(object sender, EventArgs e)
14        {
15            // vlastní řešení
16        }
17    }
18 }
```

Obr. 19: Ukázka kódu Windows Form

K vytvoření návrhu klientské aplikace (viz Příloha 3, resp. Příloha C) byly použity tyto prvku:

- **TextBox** – jedná se o textové pole, která dává uživateli možnost vkládání textu. Dá se nastavit jednořádkové i víceřádkové vkládání textu. Podporuje i možnost maskování hesla.
- **Button** – tlačítko, které spustí akci poté, co na něj uživatel klikne.
- **Label**. Uživatelem needitovatelné textové pole, které slouží k informativním či popisovacím účelům.
- **ComboBox** – editovatelné textové pole, které zobrazuje drop-down nabídku možných vstupů.
- **TreeView** – ¨pole, které zobrazuje hierarchicky uspořádané prvky.

5.3.2 Funkce klientské aplikace

Vytvořený klient se dá rozdělit do pěti sekcí, viz Přílohu C. První sekce je věnována připojování k serveru. Aplikace obsahuje prvky vypsané v Tab. 1 včetně jejich popisu.

Tab. 9: Prvky a funkce klientské aplikace – sekce Server

Prvek	Funkce	Popis
Label	lblServerUri	Needitovatelné textové pole sloužící jako popisek.
TextBox	txtServerUri	Editovatelné textové pole, kam uživatel zadá adresu serveru.
Button	btnConnect	Tlačítko, po jehož zmáčknutí se klient pokusí připojit k serveru definovaném adresou v txtServerUri.
Button	btnDisconnect	Tlačítko, po jehož zmáčknutí se klient pokusí se ze serveru odpojit.

Při spuštění aplikace se vytvoří objekty třídy OpcClient a Uri. Poté, co je zadaná správná adresa serveru a je zmáčknuto tlačítko Connect, pokusí se klient připojit k serveru a to na základě metody `Connect()`. Analogická metoda – `Disconnect()` – slouží k odpojení ze serveru.

Po úspěšném připojení se zpřístupní zbylé sekce v klientské aplikaci. Druhá sekce obsahuje jediný prvek, kterým je TreeView. Po úspěšném připojení k serveru se v tomto okně zobrazí kořenový uzel (root node). Vyhledání kořenového uzlu zajistí metoda `BrowseNode(OpcObjectTypes.ObjectsFolder)`. Při rozkliknutí jednoho uzlu se zavolá událost, která vyhledá všechny napojené uzly. K tomu slouží metoda `Children()`. Zároveň se automaticky doplní nodeID vybraného uzlu do textových polí v sekcí čtení a zápis

Další oblastí je sekce pro čtení, jejíž prvky jsou popásány v Tab. 10.

Tab. 10: Prvky a funkce klientské aplikace – sekce Read

Prvek	**Funkce**	**Popis**
Label	lblNodeId	Needitovatelné textové pole sloužící jako popisek.
TextBox	txtNodeToRead	Editovatelné textové pole, kam uživatel zadá nodeID uzlu, jehož aktuální hodnotu si chce přečíst
Button	btnReadNode	Tlačítko, po jehož zmáčknutí se klient pokusí jednorázově přečíst hodnot vybraného uzlu
Button	btnMonitor	Tlačítko, po jehož zmáčknutí se klient v případě dosud neaktivního monitorování pokusí spustit monitoring vybraného uzlu. Pokud se tlačítko zmáčkne ve chvíli, kdy probíhá monitorování, dojde k jeho ukončení.
Label	lblReadValue	Needitovatelné textové pole, do kterého se vypíše přečtená hodnota vybraného uzlu
Label	lblMonitoredValue	Needitovatelné textové pole, do kterého se pravidelně vypisuje hodnota monitorovaného uzlu

První z tlačítek zajišťuje přečtení hodnoty uzlu, který uživatel specifikuje buď zapsáním NodeID do textového pole nebo si uzel vybere z nalezených uzlů v TreeView. Ke čtení se používá metoda `ReadNode()`. K monitorování je nejprve potřeba vytvořit objekt třídy Subscription. Následně se ke spuštění monitorování použije metoda `SubscribeDataChange()`. K odhlášení odběru a ukončení monitoringu slouží metoda `Unsubscribe()`.

Předposlední sekcí aplikace je sekce pro zápis nové hodnoty do vybraného uzlu. Přehled funkcí je v Tab. 11.

Tab. 11: Prvky a funkce klientské aplikace – sekce Write

Prvek	**Funkce**	**Popis**
Label	lblNodeIdWrite	Needitovatelné textové pole sloužící jako popisek.
Label	lblWriteNewValue	Needitovatelné textové pole sloužící jako popisek.
Label	lblDataType	Needitovatelné textové pole sloužící jako popisek.
TextBox	txtNodeIdWrite	Textové pole, do něhož uživatel zadá nodeID uzlu, který chce přepsat. V případě označení některého z uzlů v TreeView se nodeId vypíše do textového pole automaticky, je možné to ale přepsat.
TextBox	txtNewValue	Textové pole, do něhož uživatel zadá novou hodnotu uzlu
ComboBox	cBoxDataTypes	Výběr datového typu, kterým je přepisovaný uzel charakterizován.
Button	btnWrite	Tlačítko, které slouží k přepsání vybraného uzlu uživatelem zadanou hodnotou.

K přepsání hodnoty uzlu, specifikovaném pomocí jeho nodeID, se používá metoda `WriteNode()`. Je potřeba brát úvahu, že vstup uživatele, kterým zadává novou hodnotu k přepsání, je datového typu String, Datový typ cílového uzlu může být ovšem odlišný. Z toho důvodu je v ComboBoxu výběr datových typů, do nichž klient dokáže vstup konvertovat.

Posledním, dosud nezmíněným blokem je needitovatelné víceřádkové textové pole, kam se vypisují chybové hlášky a informace o prováděných akcí.

6 ZHODNOCENÍ

Prvním dílčím úkolem praktické části této práce byla konfigurace serveru na straně řídicího systému SIMATIC S7-1200. K prvnímu ověření správného nastavení byl použit referenční OPC UA klient UaExpert vyvinutý společností Unified Automation GmbH[2]. Klientem bylo možné se k serveru připojit, procházet address space, číst aktuální hodnoty vybraných uzlů a přepisovat je.

Konzolová aplikace napsaná programovacím jazykem Python 3 byla testována nejprve s vlastním serverem také napsaným v Pythonu 3 (viz Přílohu D), poté i s volně dostupným Prosys OPC UA serverem. Všechny její funkce se jevily jako funkční. Pro pohodlnější prohledávání address space a vypisování uzlů by bylo vhodnější zavedené grafického rozhraní.

Druhá klientská aplikace naspaná v C# se otestovala jak s vlastním serverem (viz Přílohu D), tak se serverem Prosys. Všechny její funkce se projevily jako funkční.

Na závěr se testovala komunikace mezi klienty a PLC. Klienty byly spuštěny na PC s operačním systémem Windows 10, který byl s PLC propojen ethernetovým kabelem. Oba klienty úspěšně s PLC komunikovaly. Při monitorování bylo znát jisté zpoždění, které je ovlivněno typem komunikace a intervaly vzorkování (100 ms) a publikování (200 ms) na straně PLC.

[2] *Společnost Unified Automation GmbH se zabývá vývojem softwaru a zařízení v průmyslové automatizaci [22].*

7 ZÁVĚR

OPC UA protokol je velice perspektivním komunikačním rozhraním, které usnadňuje a zefektivňuje datový tok napříč celým podnikem. Díky tomu je možné monitorovat parametry procesních zařízení a rychleji reagovat na chybové stavy. Konstantní přísun aktuálních dat umožňuje například analyzovat trendové vývoje strojů, zajišťovat vhodně načasovanou údržbu atp. OPC UA tak šetří čas i finanční prostředky a umožňuje spolehlivější a rychlejší výrobu, ale i vývoj nových technologií.

V úvodu práce bylo nastíněno několik základních pojmů z oblasti průmyslové komunikace. Zmíněn byl mimo jiné i referenční model ISO/OSI nebo automatizační pyramida. Speciální kapitola byla věnována představení OPC UA protokolu včetně obecného popisu tvorby klientské aplikace v jazycích Python 3 a C#. Dále bylo představeno vývojové prostředí TIA Portal a ukázána konfigurace serveru na straně PLC Siemens SIMATIC S7-1200.

V rámci práce byly úspěšně vytvořeny dvě jednoduché klientské aplikace – konzolová aplikace v jazyce Python 3 a klient s grafickým uživatelským rozhraním v C#. Vedle toho byl k příležitostnému testování napsán jednoduchý skript OPC UA serveru v jazyce Python 3.

Dále byl vytvořen program pro PLC Siemens S7-1200, na němž byl nakonfigurován OPC UA server. Testování komunikace mezi PLC a oběma klienty bylo úspěšné. Všechny programy je možné najít v přílohách.

Protože hlavním účelem bakalářské práce byl edukační přínos, řešení kladla důraz na jednoduchost a přehlednost. Zároveň byly vytvořeny postupy pro založení nového projektu v prostředí TIA Portal, pro konfiguraci serveru i pro tvorbu klientských aplikací. Vše je též k nalezení v přílohách.

8 SEZNAM POUŽITÉ LITERATURY

[1] CORBETT, Felix. *How Industry 4.0 is Changing the Architecture of Industrial Automation* [online]. In: . Maisach-Gernlinden, Germany, s. 7 [cit. 2022-05-16]. Dostupné z: https://www.ttieurope.com/content/dam/tti-europe/about/distribution-center/TTIW010-How%20IoT%20is%20changing%20IA%20architectures-EN-FINAL.pdf

[2] ZURAWSKI, Richard. *Industrial Communication Technology Handbook*. 2. Boca Raton: CRC Press, Taylor & Francis Group, 2015, 1756 s. ISBN 9781138071810.

[3] Úvod do sítí průmyslové automatizace - 1. díl. *ElektroPrůmysl.cz* [online]. 2019, **9**(01), 4-7 [cit. 2022-05-16]. Dostupné z: https://www.elektroprumysl.cz/automatizace/uvod-do-siti-prumyslove-automatizace-1-dil

[4] STEFANO, Vitturi, Zunino CLAUDIO a Sauter THILO. Industrial Communication Systems and Their Future Challenges: Next-Generation Ethernet, IIoT, and 5G. *Proceedings of the IEEE* [online]. 2019, **6**(107), 944-961 [cit. 2022-05-16]. ISSN 0018-9219. Dostupné z: https://ieeexplore.ieee.org/abstract/document/8715451?casa_token=mvtyT2Nv268AAAAA:ZZokY8Om9E_Pobtij5x0zUv2jFH4uIL3JKcGZQbt1ZEiYwRa3hhasHbNeKfGYQy-PdURGLvH3z0x#full-text-header

[5] WOLLSCHLAEGER, Martin a Thilo SAUTER. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine* [online]. **11**(1), 17-27 [cit. 2022-05-16]. ISSN 1941-0115. Dostupné z: https://ieeexplore.ieee.org/abstract/document/7883994?casa_token=XIWz0zuIQawAAAAA:Y7gPBYTNdZuR25Mz6-JgvgXTTT_TU4X6Sukam66dVCtyCGeIpp_ktGf0cnAg6J3ls7ZcBfnuptCe

[6] Industrial Ethernet is now bigger than fieldbuses: Industrial network market shares 2018 according to HMS. In: *HMS Networks* [online]. 2022 [cit. 2022-05-16]. Dostupné z: https://www.hms-networks.com/news-and-insights/news-from-hms/2018/02/27/industrial-ethernet-is-now-bigger-than-fieldbuses

[7] Continued growth for industrial networks despite pandemic: Industrial network market shares 2021 according to HMS Networks. In: *HMS Networks* [online]. 2022 [cit. 2022-05-16]. Dostupné z: https://www.hms-networks.com/news-and-insights/news-from-hms/2021/03/31/continued-growth-for-industrial-networks-despite-pandemic

[8] *OPC Unified Architecture: Interoperability for Industrie 4.0 and the Internet of Things* [online]. OPC Foundation, 1-56 [cit. 2022-05-16]. Dostupné z: https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN.pdf

[9] *OPC UA-Server of the S7-1200 CPUs* [online]. München: Siemens [cit. 2022-05-16]. Dostupné z: https://support.industry.siemens.com/cs/document/109775168/opc-ua-server-of-the-s7-1200-cpus?dti=0&lc=en-WW

[10] *Programming an OPC UA .NET Client with C# for the SIMATIC NET OPC UA Server* [online]. München: Siemens [cit. 2022-05-16]. Dostupné z: https://support.industry.siemens.com/cs/document/42014088/programming-an-opc-ua-net-client-with-c-for-the-simatic-net-opc-ua-server?dti=0&lc=en-WW

[11] OPC 10000-1: OPC Unified Architecture. In: *OPC UA Online Reference: Online versions of OPC UA specifications and information models* [online]. [cit. 2022-05-16]. Dostupné z: https://reference.opcfoundation.org/v104/Core/docs/Part1/

[12] Browsing for OPC UA Nodes. In: *QuickOPC User's Guide and Reference* [online]. [cit. 2022-05-16]. Dostupné z: http://opclabs.doc-that.com/files/onlinedocs/QuickOpc/Latest/User%27s%20Guide%20and%20Reference-QuickOPC/Browsing%20for%20OPC%20UA%20Nodes.html

[13] Finding the Right SDK. In: *Unified Automation* [online]. [cit. 2022-05-16]. Dostupné z: https://www.unified-automation.com/products/sdk-overview/choose-sdk.html

[14] *Průmyslová komunikace: OPC UA* [online]. In: . Siemens - Česká republika, 2022 [cit. 2022-05-16]. Dostupné z: https://new.siemens.com/cz/cs/products/automation/industrial-communication/opc-ua.html

[15] *Siemens: TIA – plně integrovaná automatizace napříč celým podnikem* [online]. In: . München: Siemens - Česká republika, 2022 [cit. 2022-05-16]. Dostupné z: https://new.siemens.com/cz/cs/products/automation/industry-software/automatizacni-software/tia-portal.html

[16] Siemens TIA Portal – jednotné vývojové prostředí pro automatizaci v průmyslu. *Automa: časopis pro automatizační techniku* [online]. 2011, (3) [cit. 2022-05-16]. ISSN 1210-9592. Dostupné z: https://automa.cz/cz/casopis-clanky/siemens-tia-portal-jednotne-vyvojove-prostredi-pro-automatizaci-v-prumyslu-2011_03_43212_6058/

[17] Totally Integrated Automation Portal. In: *Siemens* [online]. München, 2022 [cit. 2022-05-16]. Dostupné z: https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html

[18] *TIA Portal V16*. München, 2020.

[19] *Programin Guideline for S7-1200/S7-1500* [online]. München: Siemens [cit. 2022-05-16]. Dostupné z: https://support.industry.siemens.com/cs/document/90885040/programming-guideline-for-s7-1200-s7-1500?dti=0&lc=en-WW

[20] What are the system limits of the OPC UA Server with S7-1500 and S7-1200?. In: *Siemens* [online]. München, 2009-2022 [cit. 2022-05-16]. Dostupné z: https://support.industry.siemens.com/cs/document/109755846/what-are-the-system-limits-of-the-opc-ua-server-with-s7-1500-and-s7-1200-?dti=0&lc=en-US

[21] *Traeger: Industry Components* [online]. Weiden [cit. 2022-05-17]. Dostupné z: https://www.traeger.de/

[22] Members. In: *OPC Foundation* [online]. 2022 [cit. 2022-05-17]. Dostupné z: https://opcfoundation.org/members

SEZNAM ZKRATEK

AES	Advanced Encryption
COM	Component Object Model
DB	Datový blok
DCOM	Distributed Component Object Model
ERP	Enterprise Resource Planning (Plánování podnikových zdrojů)
FB	Funkční blok
FC	Funkce
GUI	Graphic User Interface (Grafické uživatelské rozhraní)
HMI	Humman-Machine Interface
IEC	Interational Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization (Mezinárodní organizace pro normalizaci)
MAP	Manufacturing Automation Protocol
MES	Manufacturing Execution Systém (Výrobní informační systém)
MOM	Message Oriented Middleware
OB	Organizační blok
OPC	OLE for Process Control
OPC A&E	OPC Alarm & Events
OPC DA	OPC Data Access
OPC DX	OPC Data Exchange
OPC HDA	OPC Historical Data Access
OPC UA	OPC Unified Architecture
OSI	Open Systém Interconnection
PLC	Programovatelný logický automat (Programmable Logic Controller)
PubSub	Publisher/Subscriber
RTE	Real-Time Ethernet
SCADA	Supervisory Control And Data Acquisition
SDK	Software Development Kit
SOA	Service-Oriented Architecture
SSL	Secure Socket Layer
TIA	Totally Integrated Automation
TLS	Transport Layer Security
TSN	Time-Sensitive Networking
WPF	Windows Presentation Foundation

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

Textové přílohy

A Založení nového projektu
B Konfigurace serveru
C Klientská aplikace C#
D OPC UA server v Pythonu

Elektronické přílohy

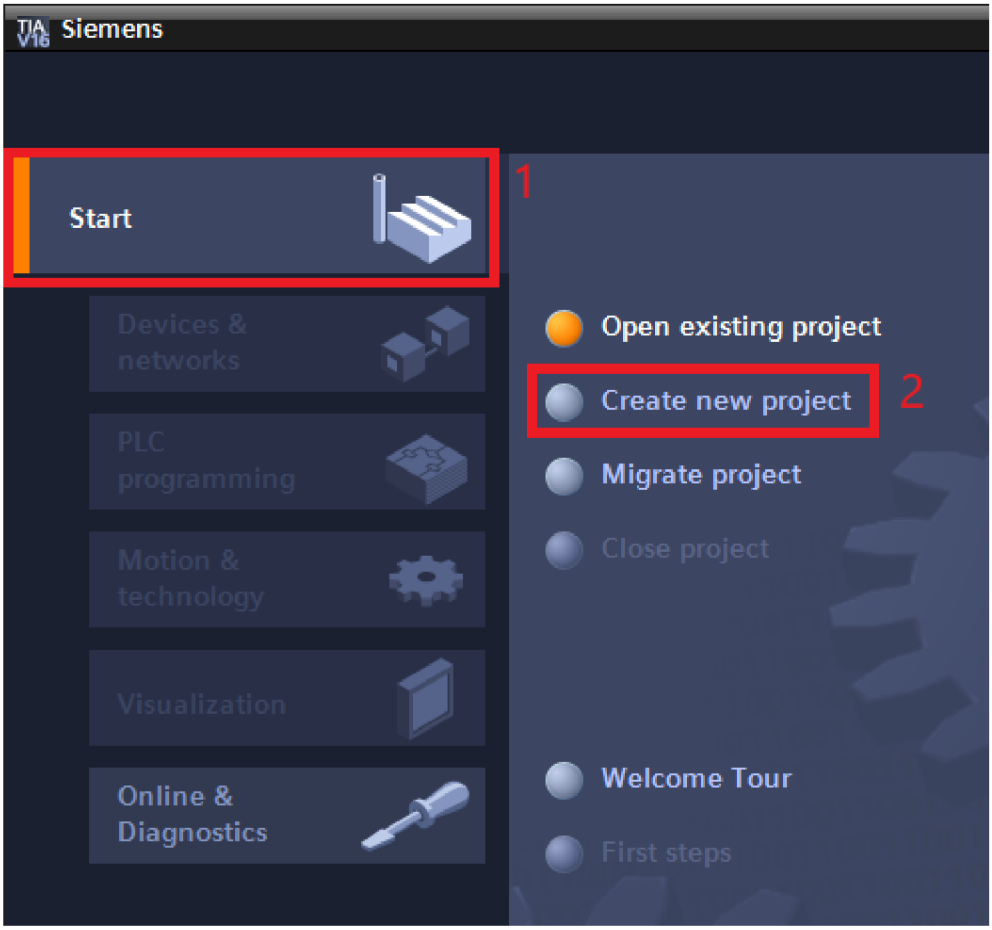
1 PLC program
2 Klient Python
3 Klient C#
4 Server Python

A ZALOŽENÍ NOVÉHO PROJEKTU

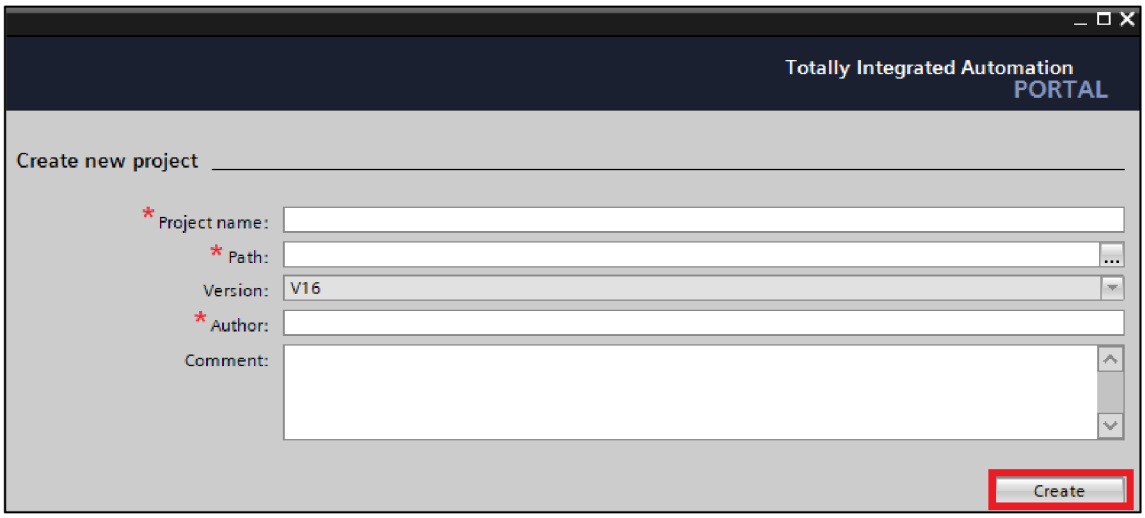
V následujících krocích je popsán postup založení nového projektu v TIA Portal V16.

Krok 1: Spustit TIA Portal a otevřít Portal view.

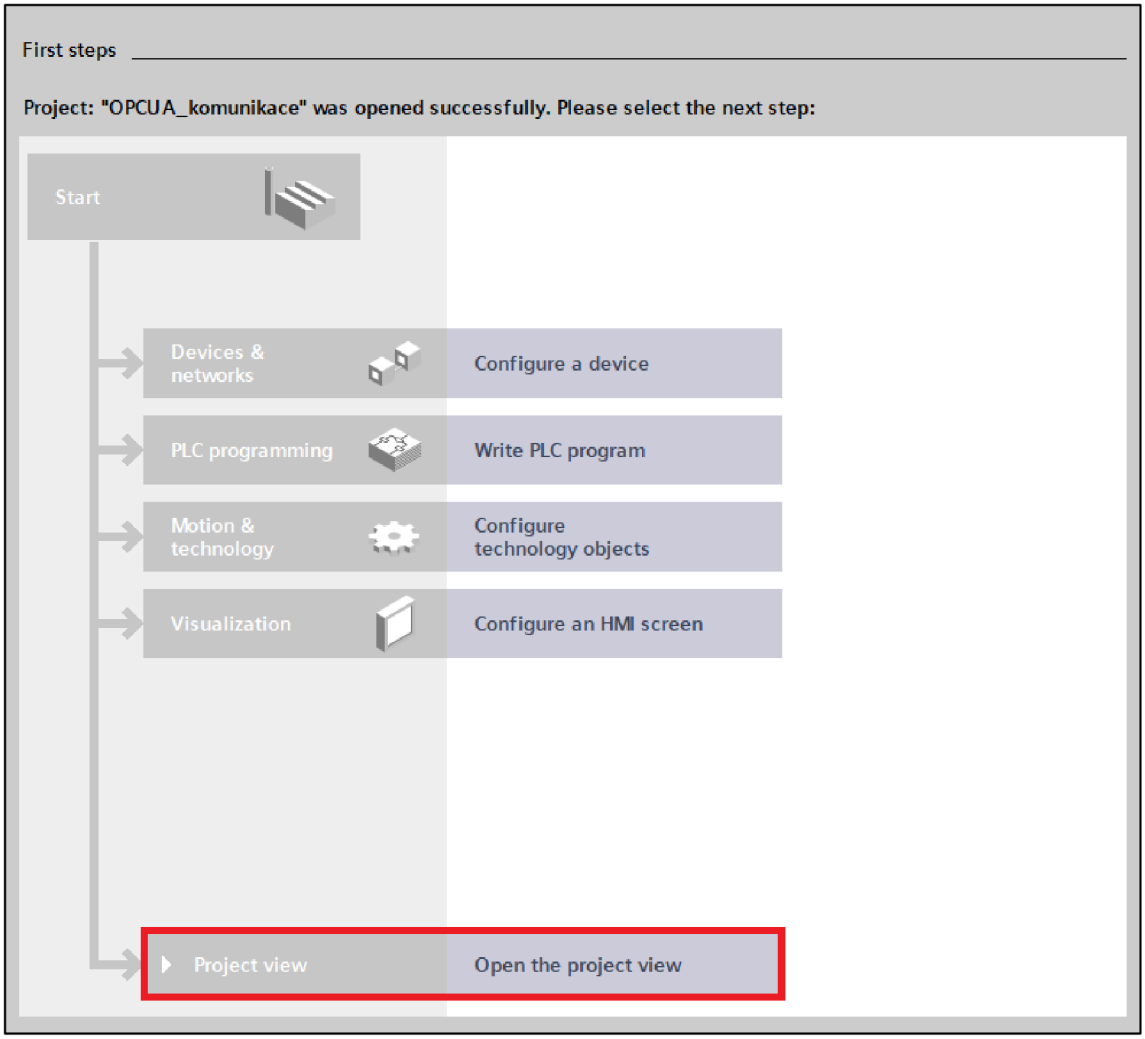
Krok 2: *Start* (1) → *Create new project* (2).



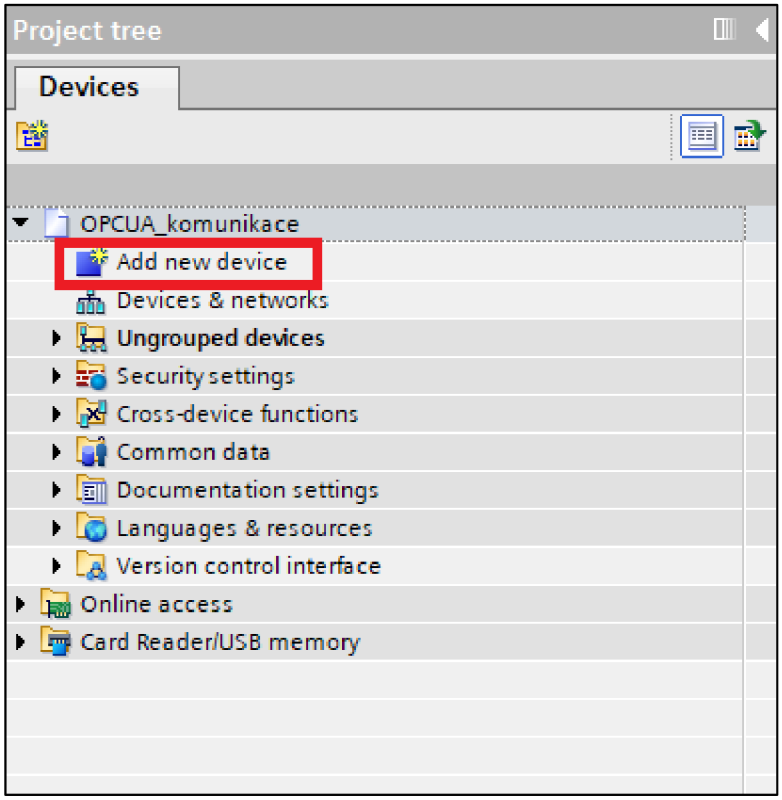
Krok 3: Vyplnit povinné položky *Project name, Path, Author* → *Create*.



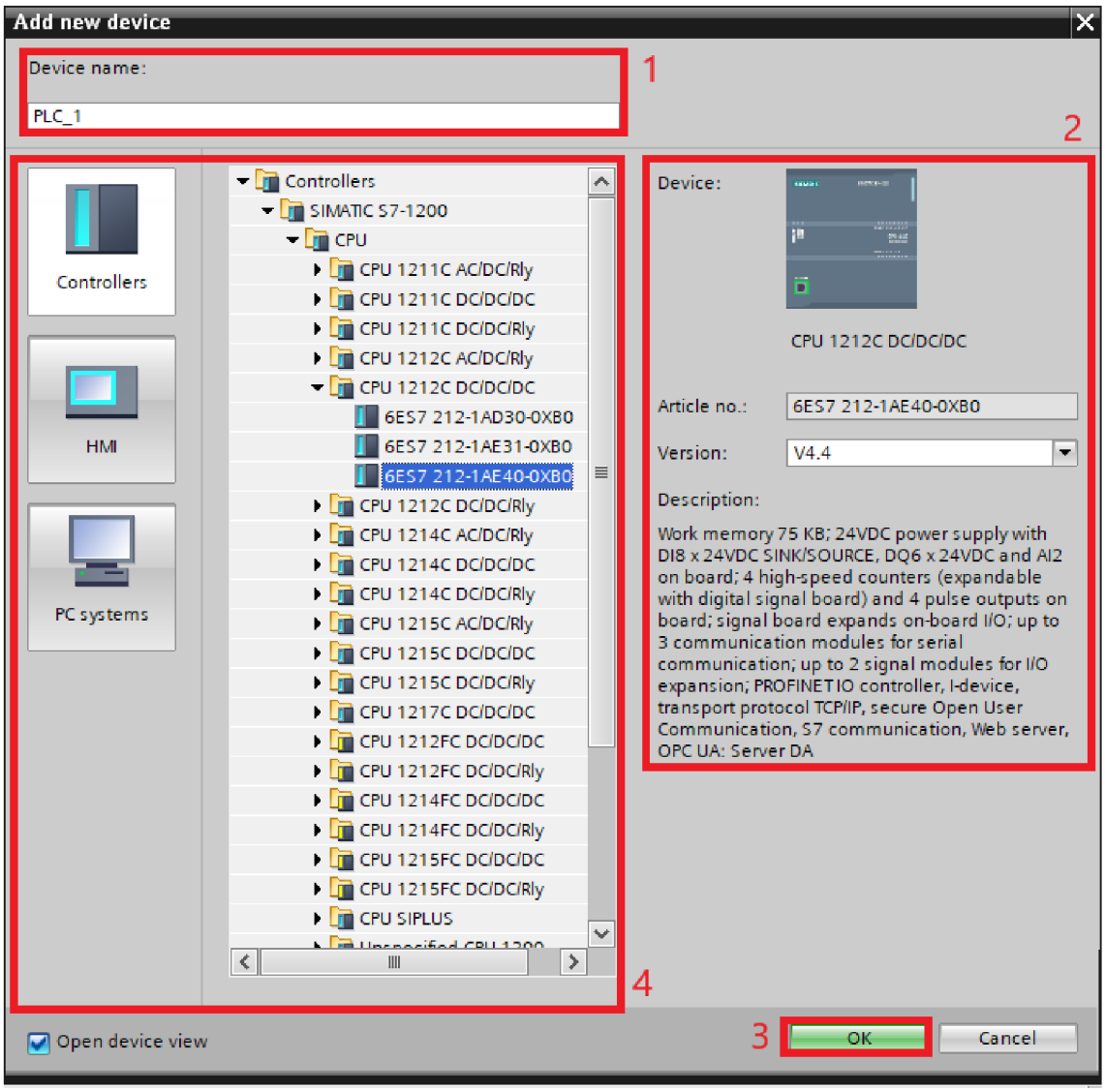
Krok 4: Po vytvoření nového projektu se zobrazí následující okno. Projekt je dále možné spravovat v Portal view nebo v Project view. Bude popsán postup v Project view. Proto je potřeba přepnout zobrazení TIA Portal z Portal view na Project view.



Krok 5: V této fázi je projekt již založen, je ovšem prázdný. Prvním krokem je přidání nového zařízení, a to přes *Project tree* → název projektu (v tomto případě *OPCUA_komunikace*) → *Add new device*.



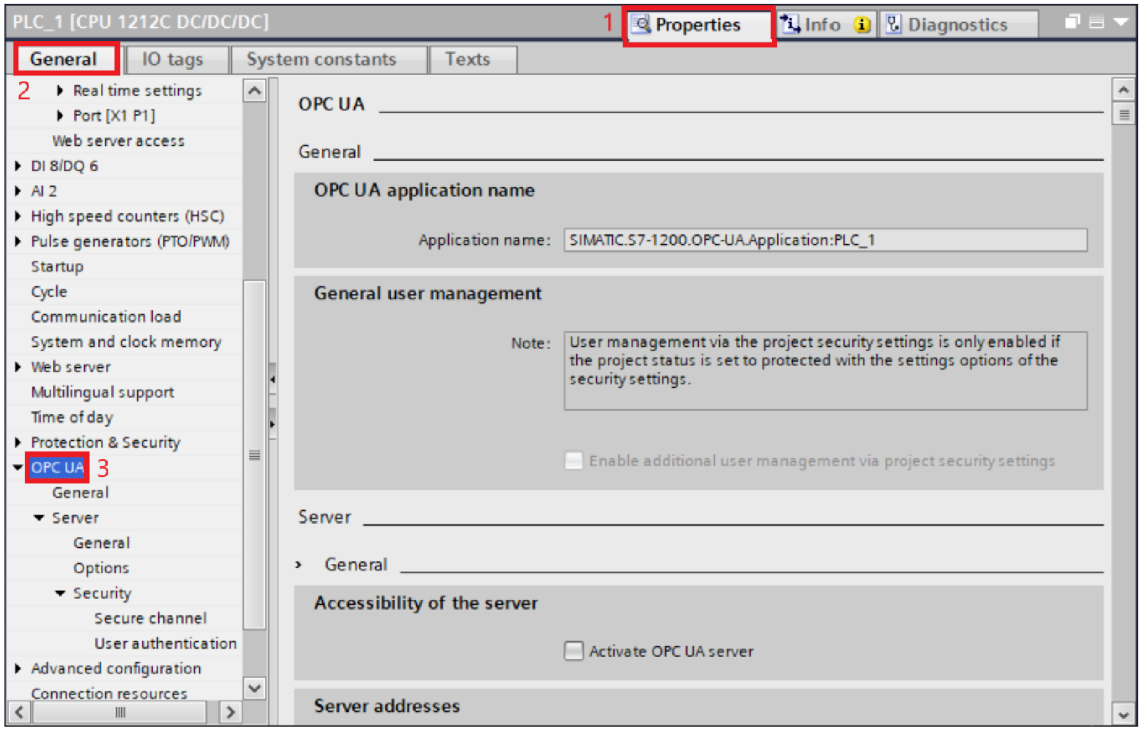
Krok 6: Objeví se okno Add new device, kde si uživatel vybere z databáze zařízení (4), dle svých preferencí může zvolit název nového zařízení či nechat automaticky vygenerovaný název (1). V pravé části okna (2) se zobrazí informace o zvoleném zařízení. Po potvrzení výběru (3) se nové zařízení objeví v Project tree, kde se následně dají přidávat programové bloky, PLC tagy, Watch tables, Force tables a dále spravovat konfiguraci.



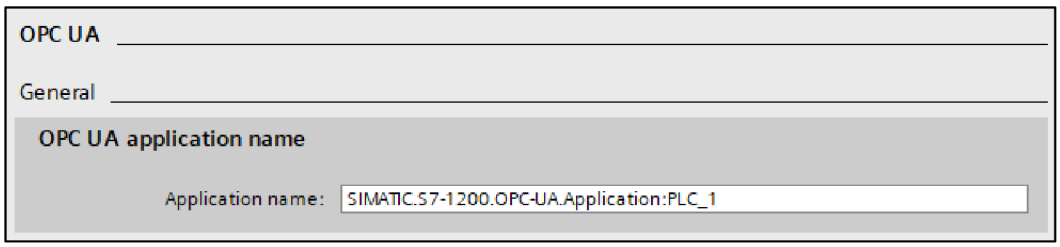
B KONFIGURACE SERVERU

V následujících krocích bude popsán postup konfigurace OPC UA serveru pro PLC Siemens SIMATIC S7-1200 v TIA Portal. Předpokládá se již vytvořený projekt s vloženým PLC.

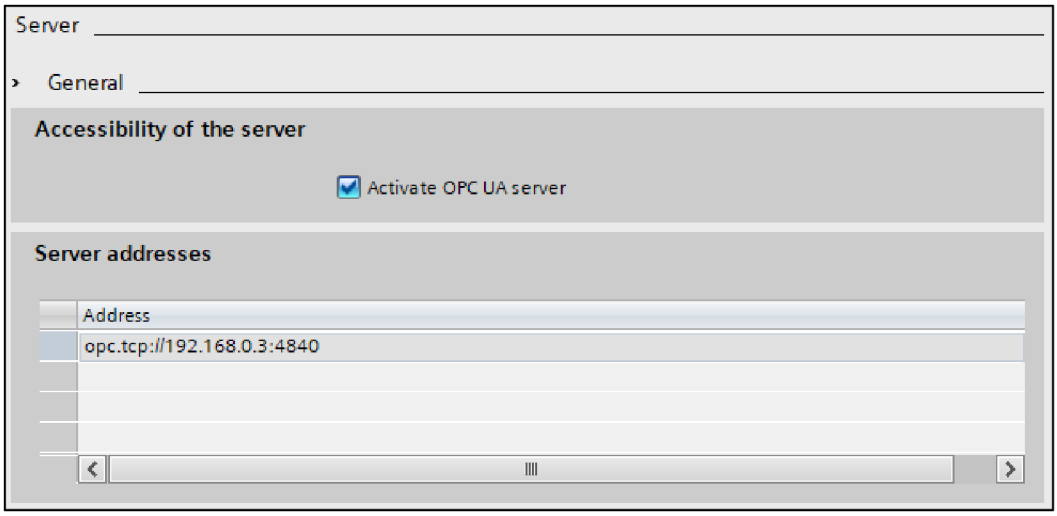
Krok 1: V Project tree se u daného PLC rozklikne Device configuration, což otevře následující okno. Pokud se okno nezobrazí, bude pravděpodobně srolované ve spodní části TIA Portal. Přes *Properties* (1) → *General* (2) → *OPC UA* (3) se lze dostat ke správě OPC UA serveru.



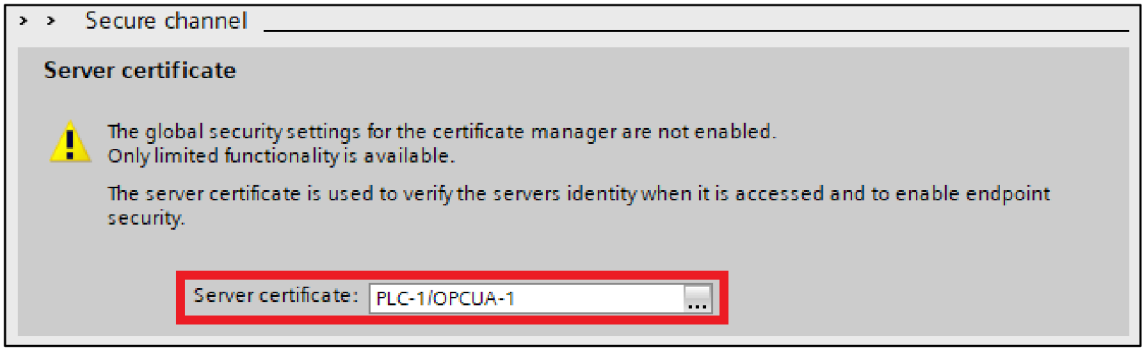
Krok 2: Název aplikace se generuje automaticky, nicméně je možnost ho v případě potřeby změnit. Přes *OPC UA* → *General* → *OPC UA application name* lze změnit název aplikace.



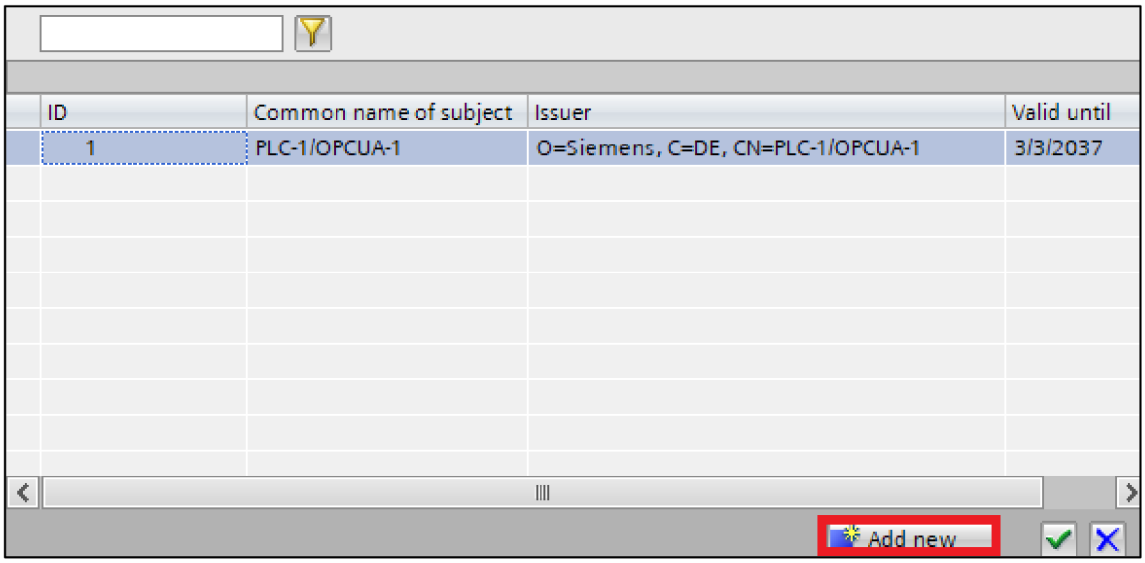
Krok 3: Pro aktivaci samotného serveru je třeba zaškrtnout políčko Activate server přes *OPC UA → Server → General*

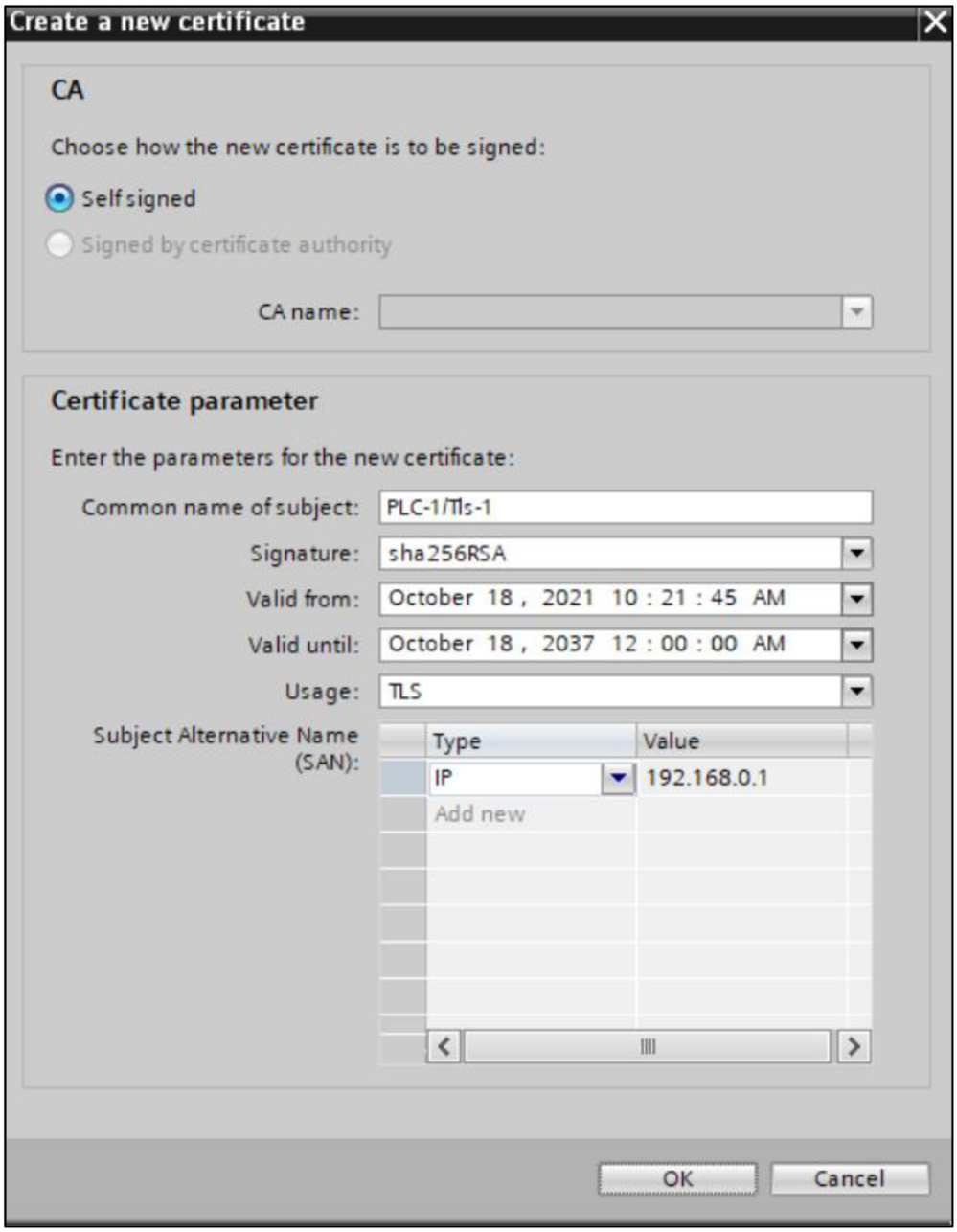


Krok 4: Přidat server certifikát přes *OPC UA → Server → Security → Server certificate*

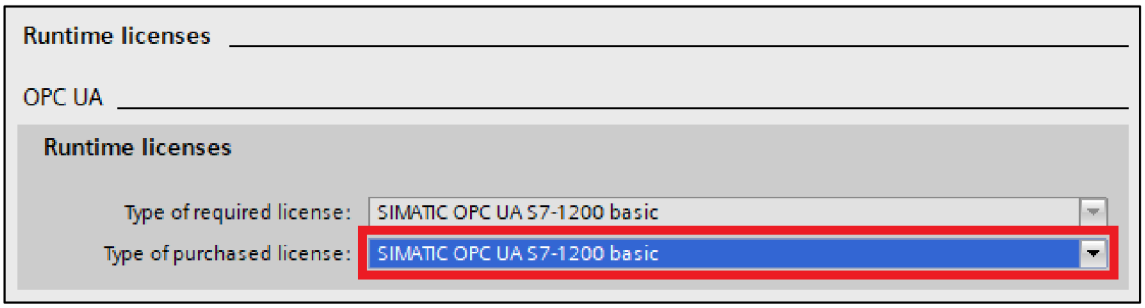


V případě, že se žádný nenabízí: rozkliknout tři tečky a pomocí Add new přidat nový certifikát, nastavit parametry a zelenou fajfkou potvrdit.

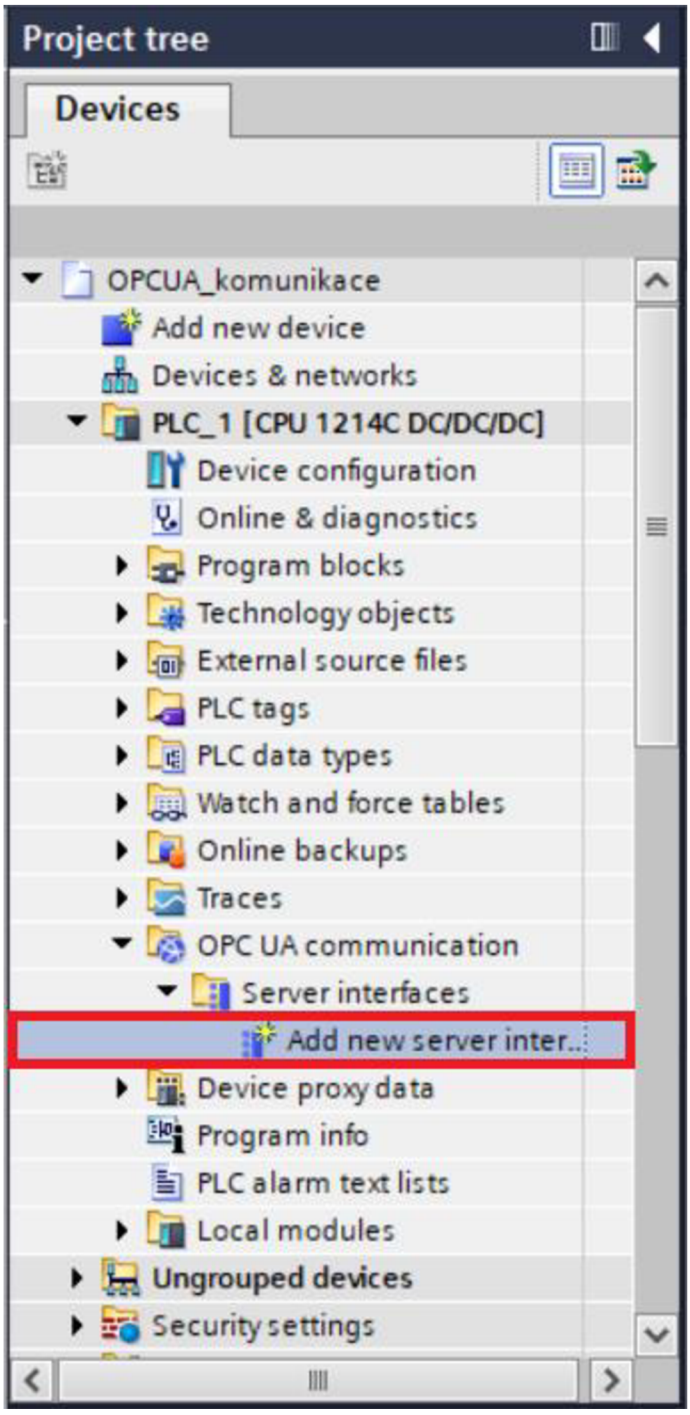


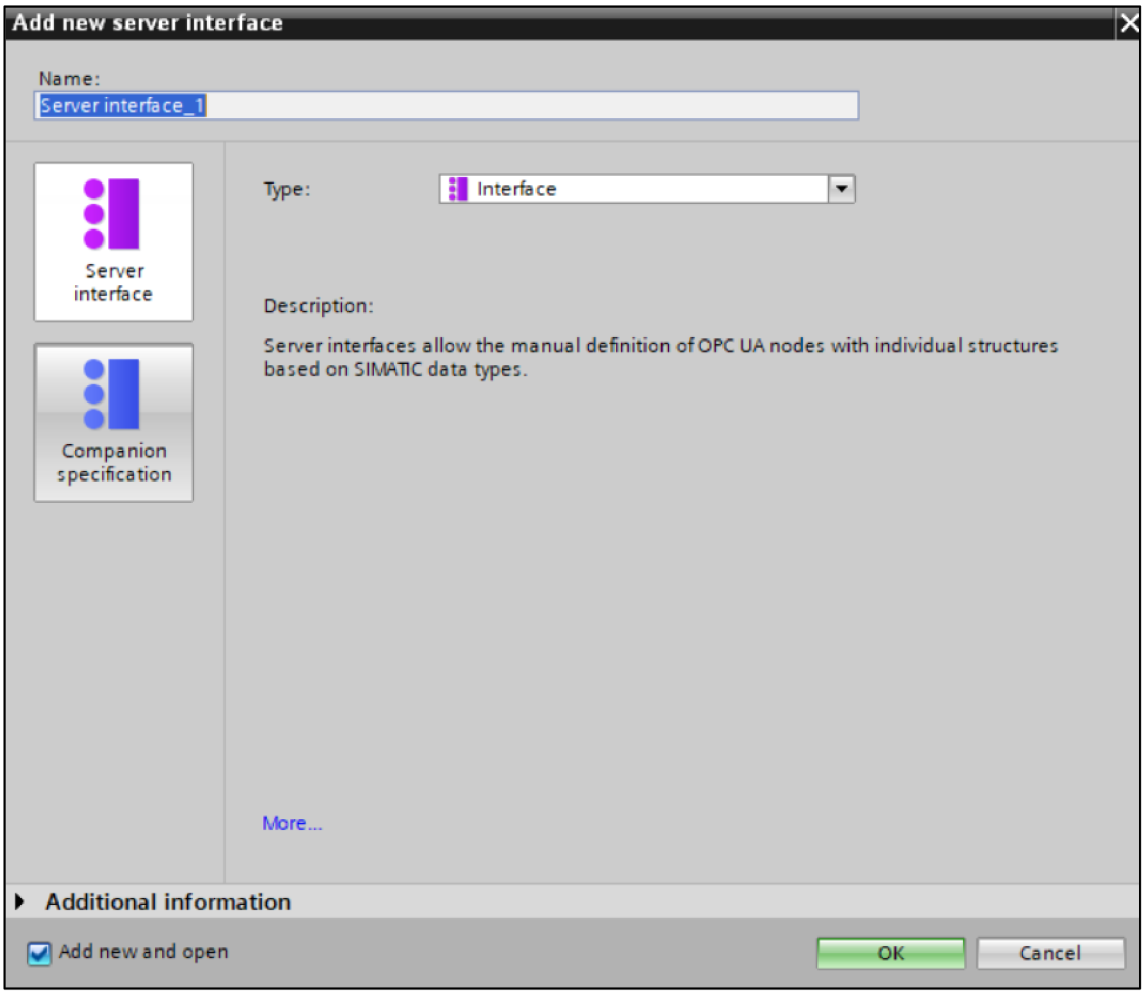


Krok 5: Přes *OPC UA* → *Runtime* licence vybrat licenci.

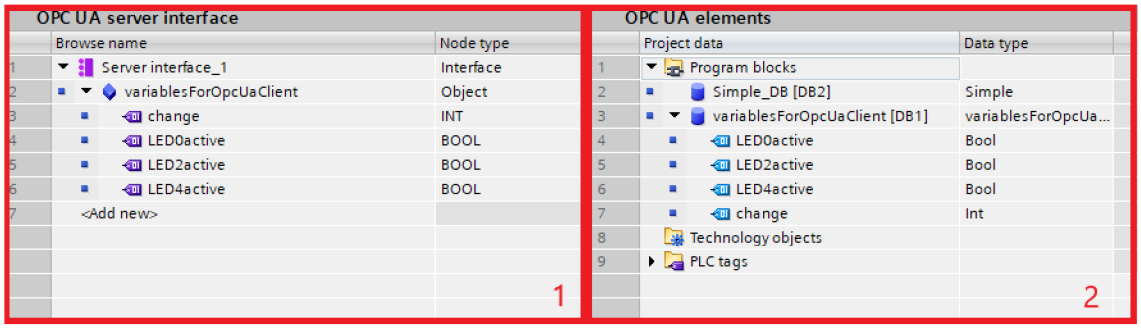


Krok 6: Server OPC UA rozhraní se přidá přes *Project tree → OPC UA communication →Server interfaces → Add new server interface.*





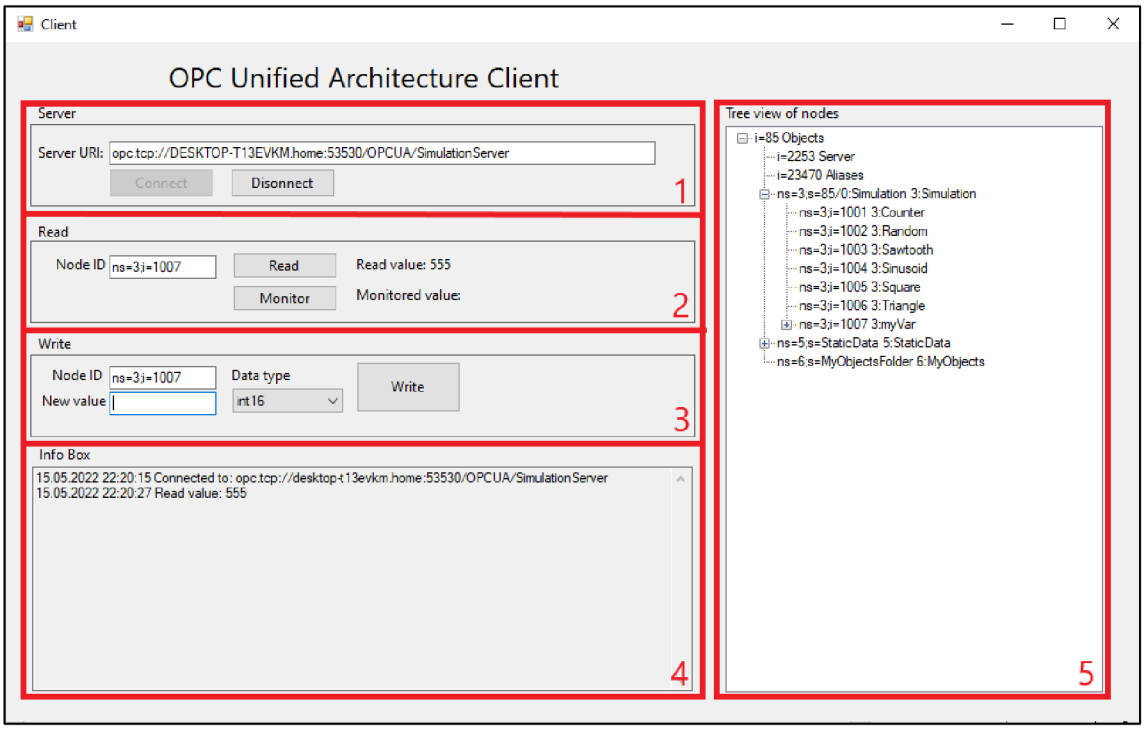
Krok 7: Vložení vybraných prvků přetažením z (2) do serverového OPC UA rozhraní (1).



Krok 8: Zkompilovat projekt a nahrát hardwarovou konfiguraci do PLC.

C KLIENTSKÁ APLIKACE C#

Tato příloha slouží k bližšímu popisu klientské aplikace (C#). Rozhraní klienta znázorněné na Obr. 21 se dá rozdělit do pěti bloků, jenž jsou více popsány v Tab. 12.



Obr. 21: Klientské GUI rozdělené do čtyř bloků

Tab. 12: Blokový popis klientské aplikace v C#

Blok	**Popis**
Blok 1	Obsahuje textové pole (*txtServerUri*) pro zadání URI serveru. Dále tlačítko **Connect** (*btnConnect*) pro připojení k server. Toto tlačítko je aktivní automaticky při spuštění aplikace a deaktivuje se po připojení k serveru. Druhé tlačítko je **Disconnect** (*btnDisconnect*) a slouží k odpojení ze serveru. V primárním nastavení je deaktivované, aktivuje se až po připojení k serveru.

Blok 2	Blok obsahující funkce ke čtení hodnot. Vlevo se nachází textové pole (*txtNodeToRead*), do nějž uživatel zadá NodeID požadovaného uzlu. Vedle textového pole je tlačítko **Read** (*btnReadNode*), které slouží k jednorázovému přečtení aktuální hodnoty uzlu. Hodnota se zobrazí vpravo od tlačítka. Druhým tlačítkem je **Monitor** (*btnMonitor*), které slouží k monitorování vybraného uzlu. Hodnota se zobrazuje napravo od tlačítka. Textové pole i obě tlačítka se zaktivují až po připojení k serveru a opět se deaktivují po odpojení ze serveru.
Blok 3	Tento blok obsahuje textové pole (*txtNodeIdWrite, txtNewValue*) pro NodeID a novou hodnotu, která se propíše do aktuální hodnoty uzlu po zmáčknutí tlačítka **Write** (*btnWrite*). Dále se v této části nachází box (*cBoxDataTypes*)s výběrem datového typu cílového uzlu. Textová pole i tlačítko se zaktivují až po připojení k serveru a opět se deaktivují po odpojení ze serveru.
Blok 4	V tomto bloku se uživateli zobrazují provedené akce včetně časové známky. Vypíšou se sem i případné chybové hlášky.
Blok 5	Blok obsahuje pole, kde se zobrazují hierarchicky uspořádaní uzly nalezené na serveru, k němuž je klient připojen.

D OPC UA SERVER V PYTHONU 3

Pro účely testování klientských aplikací v případech, kdy nebylo dostupné fyzické PLC, byl vytvořen vlastní server. Následující kód je napsán z jazyce Python a definuje nový server včetně tří proměnných, k nimž mají klienti čtecí i zapisovací práva. Pro kontrolu uživatele se v pravidelném cyklu vypisují hodnoty proměnných na konzoli. Tab. 13 zobrazuje parametry proměnných.

Tab. 13: Parametry proměnných na serveru

Název	**NodeId**	**Primární hodnota**
Random number	ns=2;i=2	
Number	ns=2;i=3	6.7
Word	ns=2;i=5	Court

Python

```
from opcua import Server
from random import randint
import time

server = Server()
try:
    # defining server
    url = "opc.tcp://127.0.0.3:4840"
    server.set_endpoint(url)
    nameOfServer = "OPCUA_Server"

    # creating new namespace
    addSpace = server.register_namespace(nameOfServer)

    # creating new object within namespace
    node = server.get_objects_node()
    var = node.add_object(addSpace, "Numbers")

    # adding variables to "Numbers"
    randNum = var.add_variable(addSpace, "Random number", 0)
    randNum.set_writable()
    rand = randint(-50, 999)
    randNum.set_value(rand)

    num1 = var.add_variable(addSpace, "Number", 0)
    num1.set_writable()
```

```
    num1Value = 6.7
    num1.set_value(num1Value)

    # creating new object within namespace
    node2 = server.get_objects_node()
    var2 = node2.add_object(addSpace, "Strings")

    # adding variables to "String"
    string1 = var2.add_variable(addSpace, "Word", 0)
    string1.set_writable()
    string1.set_value("Court")

    server.start()
    print("Server started at: " + url)

    while True:
        print("Random number: ", randNum.get_value())
        print("Number 1: ", num1.get_value())
        print("Word 1: ", string1.get_value())
        print()
        time.sleep(2)
except:
    print("Server problem")
```