VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Jakub Hurník

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

HMI APLIKACE V PROSTŘEDÍ ANDROID KOMUNIKUJÍCÍ POMOCÍ OPC UA
HMI APPLICATION FOR ANDROID USING OPC UA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jakub Hurník
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jakub Arm
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Jakub Hurník ***ID:*** 186090
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

HMI aplikace v prostředí Android komunikující pomocí OPC UA

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit OPC UA klienta v prostředí Android, který slouží jako rozhraní člověk-stroj (HMI) laboratorního modelu „Vrtačka“.

1. Realizujte připojení laboratorního modelu „Vrtačka“ na PLC S7-1500 a v prostředí TIA Portal vytvořte vzorovou úlohu pro řízení modelu.
2. Seznamte se s komunikačním standardem OPC UA a stručně jej popište.
3. Naimplementujte OS Android do Raspberry PI nebo jiného kitu s displejem.
4. Navrhněte a realizujte HMI aplikaci pro OS Android.
5. Demonstrujte funkčnost a zhodnoťte řešení (jitter, systémové zdroje).

DOPORUČENÁ LITERATURA:

OPC Foundation. Unified Architecture. opcfoundation.org [online]. ©2017 [cit. 2017-09-11]. Dostupné z: https://opcfoundation.org/about/opc-technologies/opc-ua/

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 20.5.2019

Vedoucí práce: Ing. Jakub Arm
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Tato bakalářská práce se zabývá realizací připojení laboratorního modelu Vrtačka, vytvořením vzorové úlohy pro tento model a následně vytvoření HMI aplikace pro android. Teoretická část popisuje komunikační standart OPC UA a nastíní možnosti při tvorbě HMI aplikace. Praktická část obsahuje tvorbu programu pro PLC v prostředí Tia Portal, implementaci Androidu do Raspberry PI, vývoj HMI aplikace a její otestování.

KLÍČOVÁ SLOVA

PLC, Simatic S7-1500, TIA Portal, OPC UA, Android, HMI aplikace.

ABSTRACT

This bachelor thesis deals with the connection of the laboratory model of the drilling machine, creating a model task for this model and implementing the Android operating system into Raspberry PI. The theoretical part describes the communication standard OPC UA and outlines the possibilities of creating the HMI application. The practical part includes creation of program for PLC in Tia Portal environment, implementation of Android to Raspberry PI, development of HMI application and functionality testing.

KEYWORDS

PLC, Simatic S7-1500, TIA Portal, OPC UA, Android, HMI application.

HURNÍK, Jakub. *HMI aplikace v prostředí Android komunikující pomocí OPC UA*. Brno, Rok, 57 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Jakub Arm,

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „HMI aplikace v prostředí Android komunikující pomocí OPC UA“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing.Jakubu Armovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Úvod

Hlavním cílem této bakalářské práce je vytvořit OPC UA klienta v prostředí Android, který slouží jako rozhraní mezi uživatelem a laboratorním modelem vrtačky. Celá práce by šla rozdělit na dvě pomyslné části - v první se budu zabývat teorií a shrnu, co vše je potřeba znát, než se pustím do samotné praktické činnosti a v té druhé detailněji popíšu postup tvorby aplikace.

Nejprve ukážu strukturu celého projektu, který se ve skutečnosti skládá ze dvou aplikací. Je tu program ovladající plc a hmi aplikace spuštěná na Android zařízení. Tu si rozeberu hned v další kapitole.

Následovat bude asi největší část teorie shrnující veškeré dostupné materiály. Školní laboratorní model vrtačky využitý k praktickému otestování bylo třeba nejprve trochu upravit, aby byl vhodný pro další používání ve výuce. Programovatlný automat, na kterým úloha běží, je od firmy Siemens - konkrétně PLC Simatic S7-1500. Pro program na plc slouží prostředí Tia Portál a je na výběr z několika prohramovacích jazyků. Android aplikace je pak napsána v jazyce Java, kde jsou tří hlavní vývojové prostředí (Eclipse, NetBeans, Android studio). Nejdůležitější část teorie se zabývá komunikačním protokolem Opc Ua.

V praktické části mé práce při vytvoření projektu ukážu, jak vhodně nakonfigurovat plc a zhotovím program, který ho bude ovládat. Skládá se ze dvou módu - manuálního a automatického. Manuální mód testuje především funkčnost android aplikace a spojení s opc serverem. Automatický mód oproti tomu testuje převážně plc program a čtení dat pomocí subscription. Jedná se o stavový automat, který využívá i data dostupná ze senzorů. HMI apliakce pro operační systém Android je trochu složitější. Jako vývojové prostředí jsem si zvolil Android Studio pro jeho přívětivou uživatelskou stránku. Pro komunikaci s Opc serverem jsou potřeba knihovny, které definují základní funkce Opc Ua. Tyto knihovny jsou zmíněny i v teoretické části práce. Poslední nutná věc, aby celý projekt byl provozuschopný, je konfigurace OPC Serveru, sestavení aplikace a její instalace na zařízení.

1 Návrh projektu

1.1 Stanovené cíle

Na úplném začátku práce jsem si stanovil cíle, kterých bych chtěl postupně dosáhnout. Tyto jednotlivé kroky v podstatě vycházejí ze zadání bakalářské práce, jen jsou mnohem konkrétnější.

- Vytvoření projektu v Tia Portále a konfigurace PLC
- Vytvoření programu pro ovládání PLC
- Konfigurace OPC UA Serveru
- Implementace OS Android do Raspberry
- Vytvoření projektu v Android Studiu
- Implementace knihoven
- Grafické rozhraní aplikace
- Vytvroření programu android aplikace
  - Ovládání tlačítek
  - Animace (pomocí Image Switcher)
  - Připojení k serveru
  - Zápis dat
  - Čtení dat pomocí subscription
- Otestování funkčnosti jednotlivých části aplikace
  - Otestování PLC programu
  - Otestování funkčnosti OPC UA Serveru
  - Otestování aplikace (navázání spojení, zápis, čtení)

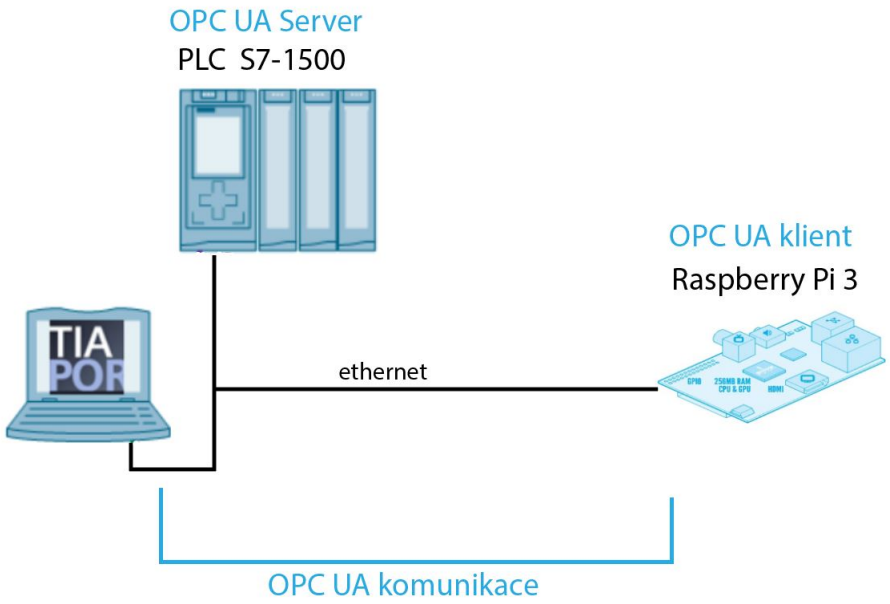
1.2 Požadavky na aplikaci

Tady bych přiblížil, co by měla aplikace splňovat. Jedná se především o tyhle body:

- univerzálnost - aby fungovala na jakémkoliv typu androidu
- spolehlivost - musí být ošetřeny všechny chyby
  (například když zápis dat vyhodí chybu, je nutné ji odchytit a zpracovat)
- zápis a čtení v reálném čase
- uživatelsky přívětivé prostředí

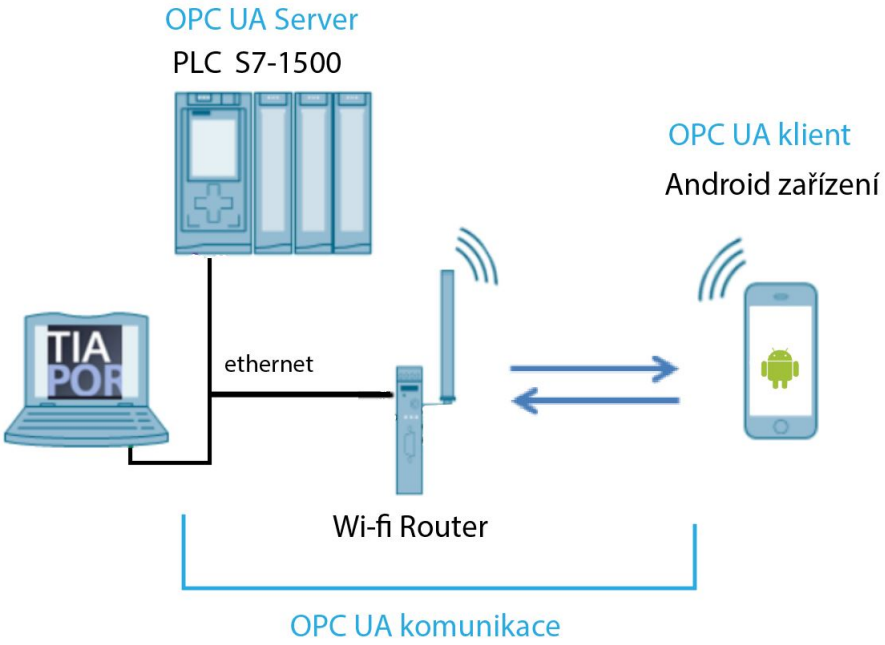
1.3 Možnosti řešení

V této kapitole nastiním možnosti řešení. Dle zadání mám implementovat operační systém Android do Raspberry, které použijeme jako klienta. Aby však šla hmi aplikace provozovat na Raspberry, je nutné k němu přidat i dotykový panel nebo obrazovku a vstupní zařízení (myš, klávesnici). Tato metoda je jednoduchá v tom, že Raspberry je přímo spojené s Plc ethernetovým kabelem a nevznikají problémy s konfigurací wi-fi routeru, tak jako v následujicím případě.



Obr. 1.1: Ukázka řešení s Raspberry

Jako mnohem praktičtější řešení se jeví využít jako klienta Android zařízení (mobil nebo tablet). Ke komunikaci je použit wi-fi router připojený do sítě plc pomocí ethernetového kabelu. Toto řešení je univerzálnější a v praxi mnohem lépe využitelné. Navíc se jedná o bezdrátový způsob komunikace a dá se tak ovládat daný stroj (v tomto konkrétním případě model vrtačky) dálkově.

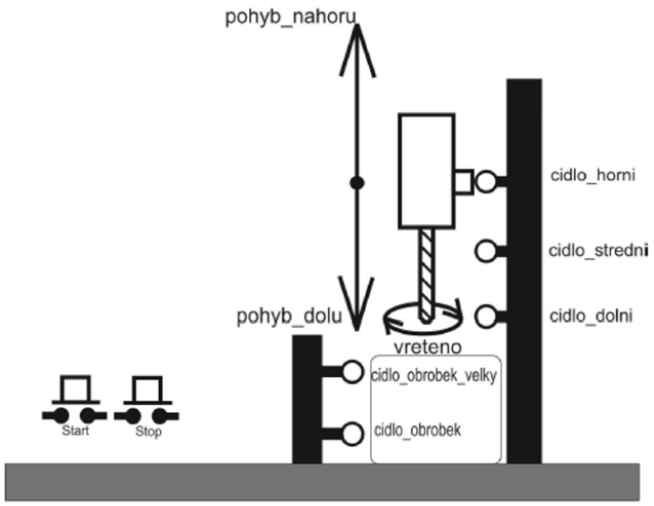


Obr. 1.2: Ukázka řešení s mobilním zařízením

2 Použité technologie

2.1 Základni popis modelu vrtačka

Model vrtačka je určen pro simulaci vrtání malých a velkých obrobků, kdy malé obrobky vrtá najednou, velké nadvakrát. Vrtání nadvakrát probíhá tak, že roztočený vrták nejprve dojede jen do poloviny obrobku, poté za stálého točení vrtáků vyjede nahoru a následně se vrací dolů. Tato metoda je použita kvůli toho, aby v praxi nedocházelo k zadření vrtáku. Po provrtání celého obrobku vrták vyjede do horní polohy, ve které zůstane do dalšího povelu.



Obr. 2.1: Schéma modelu vrtačka [3]

Práci vrtačky signalizují dvě LED diody – červená a zelená. Při aktivním vrtání svítí zelená LED dioda. Po provrtání obrobku, kdy se vrták přemisťuje do horní polohy, svítí červená LED dioda [3].

2.1.1 Čidla modelu

Magnetický jazýčkový spínač

Magnetický jazýčkový snímač se skládá ze dvou části – z magnetu, který se upevňuje na pohyblivé části modelu a senzoru, který je umístěn na pevné konstrukci vrtačky. K povrchu jsou jak magnet, tak senzor přichyceny pomocí šroubů. Na modelu jsou použity tři tyto senzory – na snímání horní, střední a dolní polohy [4].

Obr. 2.2: Magnetický jazýčkový spínač

Mechanický koncový mikrospínač

Pro zaznamenání malého či velkého obrobku je na modelu vrtačka použit mechanický koncový mikrospínač. Jedná se o jednopólový ON – ON spínač, kdy je v každém stavu sepnut alespoň jeden výstup. Na modelu jsou použity tyto mikrospínače dva, které jsou umístěny nad sebou. Pokud dojde k sepnutí jen spodního mikrospínače, jedná se o malý obrobek. Při sepnutí obou mikrospínačů zároveň jde o velký obrobek [4].

Obr. 2.3: Mechanický koncový mikrospínač [22]

2.1.2 DC motory

K rozpohybování modelu jsou použity dva stejnosměrné motory od Mega motor, které jsou napájený napětím 24 V. První motor slouží k roztočení vrtáku, který se pohybuje jen jedním směrem – po směru hodinových ručiček. Druhý motor ovládá pohyb vrtáku ve vertikálním směru. Změna směru se nastavuje pomocí ovládací karty, která je umístěna pod samotným modelem [4].

2.2 Programovatelný automat

Programovatelný logický automat (PLC) je zařízení, které zpracovává přicházející vstupní signály a podle naprogramovaných logických a časových funkcí posílá na výstup odpovídající signály. Výstupní signály ovládají akční členy v technologickém procesu jako například motory, ventily, klapky a podobně. Základními prvky programovatelného automatu jsou: Centrální procesor CPU, Modul vstupních signálů a Modul výstupních signálů [2].

2.2.1 PLC Simatic S7-1500

Jak už bylo zmíněno, při této práci využívám PLC Simatic S7-1500 od firmy Siemens. Abych byl přesný, jedná se konkrétně o model CPU 1512C-1 PN. Všechny PLC z řady S7-1500 s verzi 2.0 a vyšší už mají zabudovaný OPC UA Server. Pro jeho nastavení se musí použít Tia Portál nejméně verze V14.

Obr. 2.4: PLC Simatic S7-1500 [21]

2.2.2 Prostředí TIA Portal

TIA Portal je sdílené pracovní prostředí pro integraci přístrojů se systémy SIMATIC. Tento software v sobě obsahuje vše pro automatizaci stroje nebo výrobního zařízení. To znamená, že se jedná o prostředí, kde probíhá konfigurace, programování i samotná diagnostika již běžícího systému (řídicí systém, elektrické pohony, ovládací prvky atd.) [1].

V tomto případě byla použita verze prostředí TIA Portal – STEP7 Professional V14, která podporuje práci s PLC S7-1500. STEP 7 je základní software pro konfiguraci a programování SIMATIC řídících systému, který usnadňuje uživateli práci ve všech částech projektu [8].

Při práci v tomto prostředí máme na výběr z několik programovacích jazyků. Mezi grafické jazyky spadá Ladder Diagram. Jedná se o velmi jednoduchý a při menších projektech přehledný program, který vychází z principu kreslení elektrických schémat. Pokud bychom chtěli spíše algebraický programovací jazyk, můžeme šáhnout po takzvaném Seznamu instrukcí (IL) nebo Strukturovaném textu (ST). IL se skládá se sekvence daných instukcí a každá začíná na novém řádku, zatímco ST jako vyšší jazyk je tvořen posloupností symbolických instukcí a je velmi podobný například jazyku Pascal.

2.3 OS Android

Platforma Android byla vytvořena společnosti Google v roce 2007. Jedná se open source operační systém založený na Linuxovém jádře, který je určen hlavně pro chytré telefony, PDA, navigace a jiné mobilní zařízení. Cílem platformy Android je být co nejvíce otevřen vývojářům i uživatelům [12].

Android aplikace se tvoří pomocí programovacího jazyku Java. Pro práci s tímto jazykem je k dispozici mnoho vývojových prostředí, mezi hlavní patří Eclipse, NetBeans a Android Studio.

2.3.1 Java

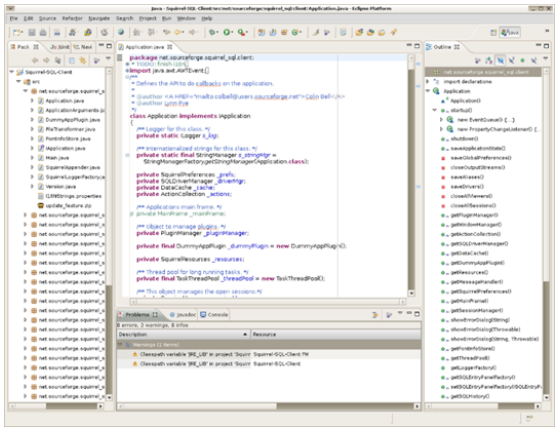
Java je objektově orientovaný programovací jazyk vyvinutý firmou Sun Microsystems, který vznikl v roce 1995. Java vychází z jazyka C++, avšak na rozdíl od něho neobsahuje žádné složitější konstrukce (např. ukazatele). Místo ukazatelů pracuje Java s referencemi a tím pádem odpadá hrozba zápisu do neplatné paměti. Obsahuje automatickou správu paměti (garbage collector), která automaticky čistí paměť od nepotřebných objektů.

Program napsaný v jazyce Java je přenositelný na jakoukoliv platformu, která obsahuje virtuální stroj jazyka Java (JVM). Kód se kompiluje jen jednou, vzniká bajtový kód, který je pak interpretován virtuálním strojem.

Jak už bylo řečeno, pro práci s jazykem Java existuje mnoho vývojových prostředí. Abychom pomocí nich mohli tvořit aplikace v Javě je ještě potřeba Java Development Kit (JDK) se kterým vývojová prostředí spolupracují. Jedná se o sadu základních nástrojů pro vývoj aplikací na platformu Java. Zahrnuje v sobě i Java kompilátor, který právě převádí zdrojový kód Javy do bajtového kódu se kterým pak pracuje JVM [17].

2.3.2 Eclipse

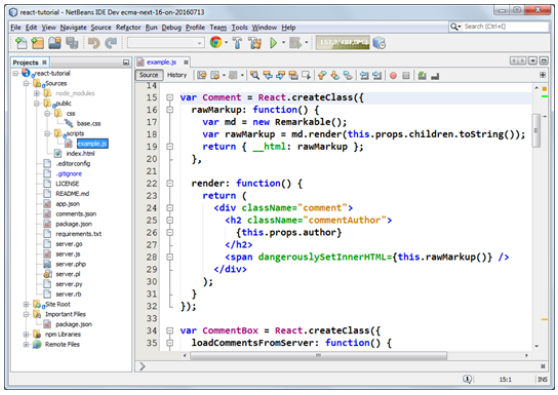
Eclipse je open source vývojové prostředí, které je napsané v jazyku Java. Má ovšem knihovny, které jsou napsány v kódu platformy, na kterém je tento program nainstalován, což zvyšuje rychlost odezvy na uživatelské rozhraní. Primárně je toto vývojové prostředí určeno pro tvorbu aplikací v jazyce Java, avšak díky přídavným pluginům zvládá i jazyky jako C++ a PHP [13].



Obr. 2.5: Ukázka vývojového prostředí Eclipse[13]

2.3.3 NetBeans IDE

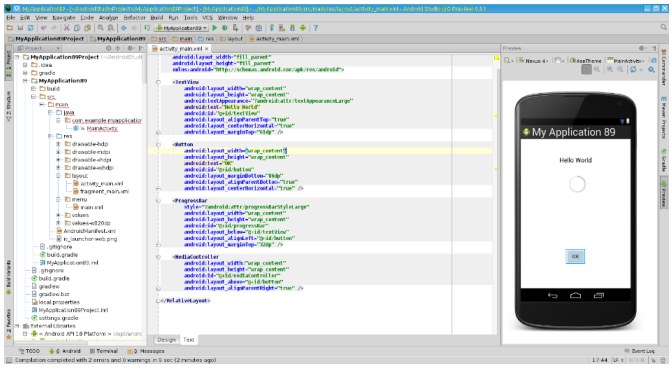
NetBeans je taktéž open source integrované vývojové prostředí, které je primárně slouží pro práci s jazykem Java, ale podporuje i další jazyky (PHP, HTML, C, C++ a další). Vývojové prostředí NetBeans je napsáno v jazyce Java a vzniklo v roce 2000 [14].



Obr. 2.6: Ukázka vývojového prostředí NetBeans[14]

2.3.4 Android Studio

Vývojové prostředí Android Studio vzniklo spolupráci společnosti Google a JetBrains. Je založeno na prostředí IntelliJ IDEA od kterého získalo možnosti práce s kódem jako navigaci v kódu, našeptávání, refaktoring nebo analýzu kódu. Při psaní v XML kódu Android Studio automaticky zobrazuje náhled. Po instalaci Studia už není nutné instalovat další přídavné moduly nebo pluginy, vše je už obsaženo v samotném instalačním balíčku. Na rozdíl od dvou předchozích vývojových prostředí Android Studio podporuju jako jazyky pouze Javu a C/C++ a umožňuje tvořit pouze Android aplikace. Je však rychlejší a má menší nároky na výkon počítače [15, 16].



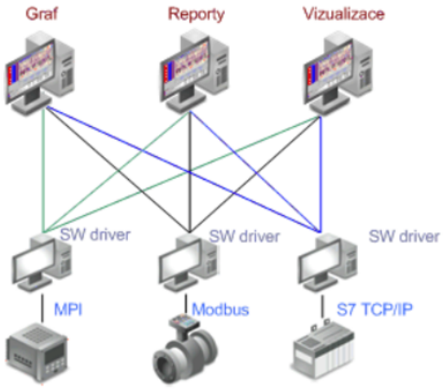
Obr. 2.7: Ukázka vývojového prostředí Android Studio[15]

2.4 OPC UA

2.4.1 OPC

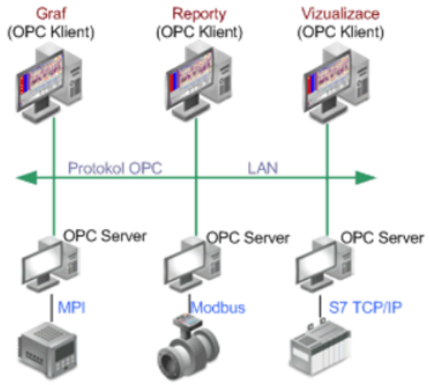
OPC je komunikační protokol, který má za cíl vytvořit jednotné komunikační rozhraní mezi Hardware a Software produkty průmyslové automatizace. Umožňuje propojení komponentů různých výrobců, které mezi sebou komunikují přes komunikační standart – protokol OPC. Tato specifikace OPC byla založena na technologiích společnosti Microsoft, proto také podporuje jen OS Windows.

Před vytvořením OPC protokolu muselo mít každé zařízení nainstalovaný speciální ovladač v počítači, který umožňoval čtení a zápis dat z tohoto zařízení. Při instalaci více těchto ovladačů docházelo k ovlivňování komunikace, nebo nekompatibilitě s operačním systémem.



Obr. 2.8: Schéma přenosu dat bez OPC[9]

S příchodem standartu OPC už nebylo dále nutno mít na každém zařízení nainstalovaný ovladač, jako jediné komunikační rozhraní slouží právě OPC. Jedná se o komunikační síť (např. pomocí ethernetu, LAN) do které se dají připojovat OPC Servery (zdroje signálu) a OPC klienti (stanice pro zpracování těchto dat) [9].



Obr. 2.9: Schéma přenosu dat pomocí OPC[9]

2.4.2 OPC UA

Na rozdíl od klasického rozhraní OPC, které bylo vytvořeno firmou Microsoft, standart OPC UA byl vyvinut organizací OPC Foundation. Jedná se o platformově nezávislý standart pro komunikaci různých zařízení (systémů) mezi servery a klienty. To znamená, že podporuje všechny druhy zařízení, od klasických stolních počítačů přes servery, PLC až po micro-kontroléry, a to i pro různé operační systémy (Windows, Apple OSX, Android, Linux). Komunikace probíhá prostřednictvím přenosu

zpráv po různých typech sítí. Díky ověřování identity serverů a klientů se jedná o zabezpečenou komunikaci [5].

OPC UA definuje standartní množinu služeb, které mohou servery poskytovat. Každý server pak oznamuje klientům, které služby podporuje. Servery definují datový model, který je pak za běhu k dispozici klientům. Mohou poskytovat okamžitá i historická data, oznamovat události i alarmy, což je oproti klasickému OPC, který tyto úkony definoval odděleně, značné zlepšení.

Komunikace OPC UA podporuje dva protokoly. První, binární protokol se vyznačuje URL specifikací (opc.tcp://Server) a druhý je protokol Web služby – SOAP (http://Server) [10].

Zabezpečení

Jelikož komunikace pomocí Opc Ua probíhá přes internet je nutné, aby měla víc vrstev ochrany. Jedná se o tyto bezpečnostní prvky:

- Security Policy - určuje míru zabezpečení. Můžeme si zvolit, jaké šifrování chceme použít.
  - None
  - Basic128Rsa15 (už není považováno za bezpečné)
  - Basic256
  - Basic256Sha256
- Message Security Mode - určuje zabezpečení zpáv
  - None
  - Signed (Odesílatel šifruje podpis soukromým klíčem, příjemce si kontroluje odesílatele pomocí veřejného klíče)
  - Signed & Encrypted (zpáva je zašifrována veřejným klíčem příjemce, který si ji svým soukromým klíčem rozšifruje)
- User Authentification - určuje možnosti uživatele, jak se přihlásit na server
  - Anonymous (Anonymní přihlášení)
  - Username (Přihlášení pomocí jména a hesla)
  - X.509 (Přihlášení pomocí certifikátu)
  - Issued (Přihlášení pomocí tokenu vygenerovaného serverem)

O míře zabezpečení rozhoduje uživatel sám a dá se nastavit na straně serveru - určíme mu, které druhy zabezpečení má podporovat [6].

Discovery Server

Discovery server slouží k vyhledávání OPC serverů. Můžeme jej rozdělit na 2 typy:

- Local Discovery Server – Tento typ se používá jen pokud se v lokální síti nachází více než jeden server.
- Global Discovery Server

Adresový prostor

Adresový prostor v opc ua funguje na bázi takzvaných uzlů (nodes), které spřístupňují informace připojenému klientovi. Tyto uzly jsou mezi sebou propojené referencemi a jsou charakterizovány svými atributy (id, typ uzlu, jméno, popis, oprávnění ke změně). Existuje osm druhů uzlů, konkrétně se jedná o tyto:

- Object
- Variable
- Method
- View
- ObjectType
- VariableType
- ReferenceType
- DataType

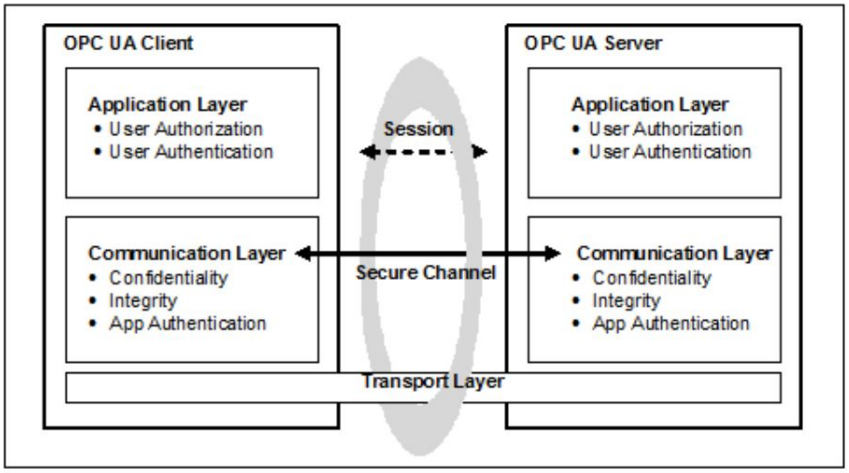
Každý uzel má své unikátní id, které se skládá ze tří částí:

1. Namespace Index - Slouží k rozlišení uzlů z ruzných subsystému
2. Identifier Type - Určuje typ identifikátoru
   (s = String, i = Numeric, g = GUID, b = OPAQUE (ByteString))
3. Identifier (Id) - Identifikátor daného uzlu

Tyto tři části se pak složí do jednoho id podle následujicího vzoru:
ns=<namespaceIndex>;<identifiertype>=<identifier> [6]

Secure channel a seasion

Propojení mezi klientem a serverem je tvořeno třemi vrstvami. Vrstvy Seasion a Secure channel mají za úkol zabezpečit spolehlivou a bezpečnou komunikaci. Nejprve se vytváří Secure channel, který se stará o nízkourovňovou komunikaci a až následně dojde k vytvoření Session. Život Seassion není závislý na Secure channel a proto narozdíl od něho při ztátě spojení nezaniká, ale jen vyčká než se vytvoří nový Secure Channel, který se k němu opět přiřadí. Nejnižší vrsta je takzvaná transportní vrstva, která slouží po navazání spojení k transportu dat [6].



Obr. 2.10: Vrstvy OPC UA komunikace [8]

Subscription

Pomocí Subscription lze získavat tři typy informací - změny hodnot, události a agregované hodnoty. Parametry subscription zadávané při jejich vytváření v klientovi určují, jak často bude server posílat změněné hodnoty (Publish interval) a jak často se bude danná proměnná vzorkovat (Sampling interval). Taky je zde možné definovat filtry, které události má klient dostávat a na jak velkou změnu hodnoty má reagovat [6].

2.5 Použité knihovny

2.5.1 SLF4J

Tato knihovna, celým názvem The Simple logging facade for Java, slouží jako jednoduchá logovací API pro Javu pro různé logovací prostředí. Její implementace jako například Logback a Log4j umožňují koncovému uživateli připojit se do požadované logovací struktury v době naszení [18].

2.5.2 Prosys-Opc-Ua-Sdk

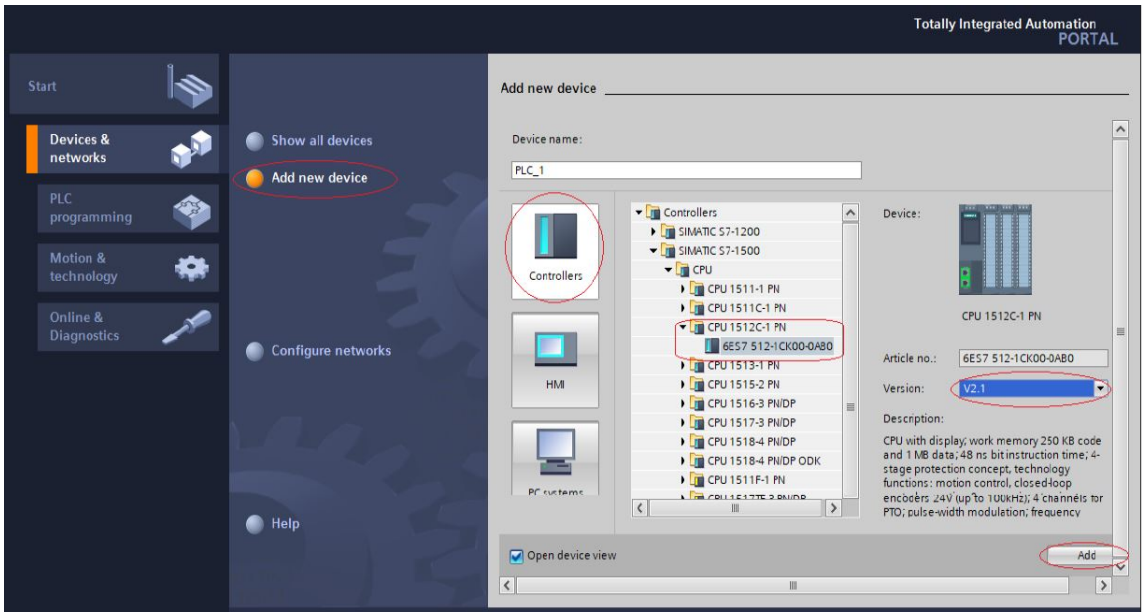
Prosys-opc-ua-sdk-client-server-evaluation je knihovna implementující nezbytnou infrastrukturu na straně klienta a serveru.

SDK pro klienta definuje Java prostředí pro komunikaci přes Opc Ua. Skládá se z objektů, které umožňují navrhovat vlastní aplikační logiku tak, aby skutečně zpracovávala data dostupná z serverů OPC UA s minimálním úsilím [19].

3 Tia Portal aplikace

3.1 HW konfigurace

Po vytvoření projektu je ze všeho nejduležitější vše správně nakonfigurovat. Rozkliknutím položky Configure a device -> Add new device se přiřadí nové zařízení. Ze skupiny Controllers se vybere SIMATIC S7-1500 -> CPU -> CPU 1512C PN -> 6ES7 512-1CK00-0AB0 a v pravém sloupci zvolí verze 2.1.



Obr. 3.1: Výběr controlleru v Project View

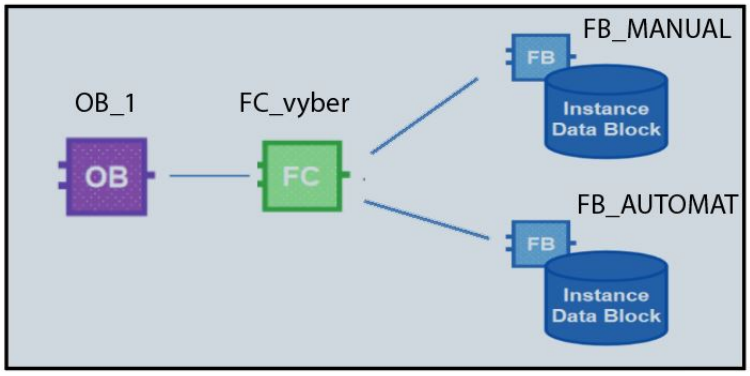
Po potvrzení se projekt přepne do Project view prostředí. Jedná se o prostředí pro práci se samotným projektem. Nyní nastává poslední krok konfigurace - nastavení IP adresy PLC, na které se bude program nahrávat. Je to z toho důvodu, že PLC umístěná ve školní laboratoři mají pevně nastavenou IP adresu. Pokud by IP adresa nebyla definována, Tia portál by ji při prvním nahrání projektu nahradil implicitní hodnotou. [7]

3.2 Struktura projektu

Program v Tia Portále se skládá z organizačních bloků, funkčních bloků a funkcí. Pro potřeby této aplikace jsme si vystačili s jednoduchým programem, který je složen z jednoho hlavního organizačního bloku (OB1), dvou funkčních bloků a jedné funkce. OB1 se spustí jen jednou, a to při zapnutí programu. Toho lze využít k inicializaci vstupních proměných. Následně OB1 cyklicky volá objekty (funkce a funkční bloky) v něm obsažené. Funkční bloky a funkce máme tyto:

- FC_vyber
- FB_MANUAL
- FB_AUTOMAT

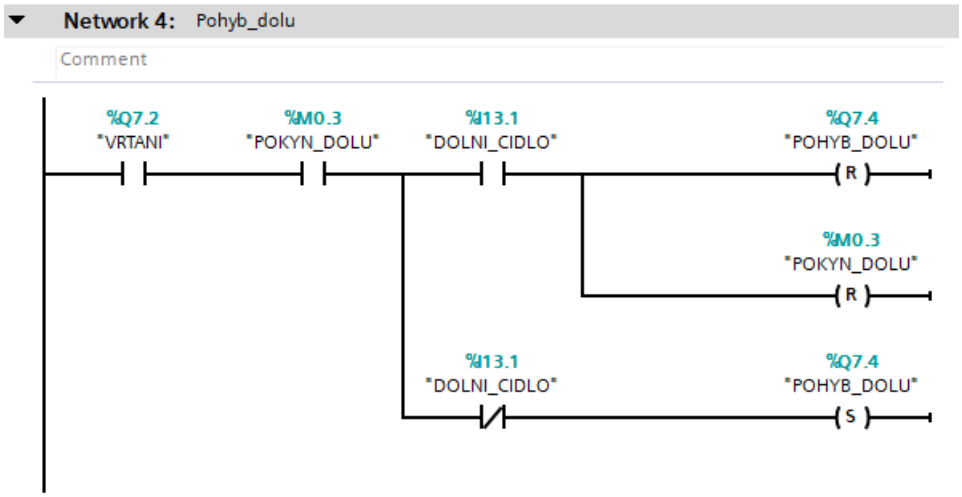
OB1 obsahuje jen funkci FC_vyber. Tam se na základě vstupních proměných z hmi aplikace rozhodne, zda přejde do funkčního bloku FB_MANUAL nebo FB_AUTOMAT.



Obr. 3.2: Struktura programu v Tia Portále

3.3 Manuální mód

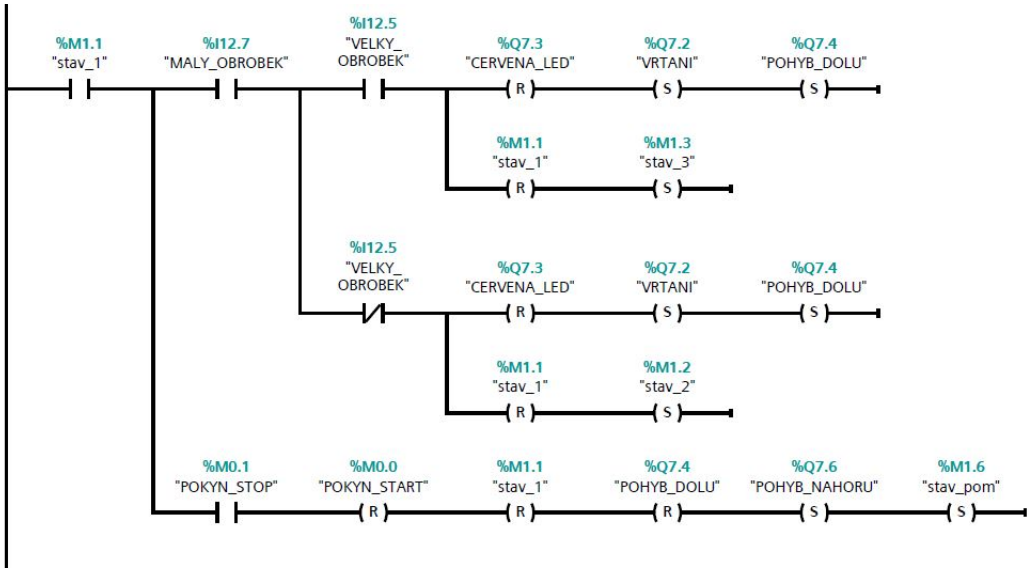
Manualní mód je jednoduchá verze ovládání vrtačky. K dispozici jsou tři tlačítka, která ovládají pohyb nahoru, dolů a točení vrtáku. Pohybovat s vrtací hlavicí je možné jen pokud se vrták točí, což je signalizováno rozsvícením zelené led diody. Program se skládá ze čtyř networků. První ošetřuje výchozí stavy a následující tři mají na starost každý jedno tlačítko.



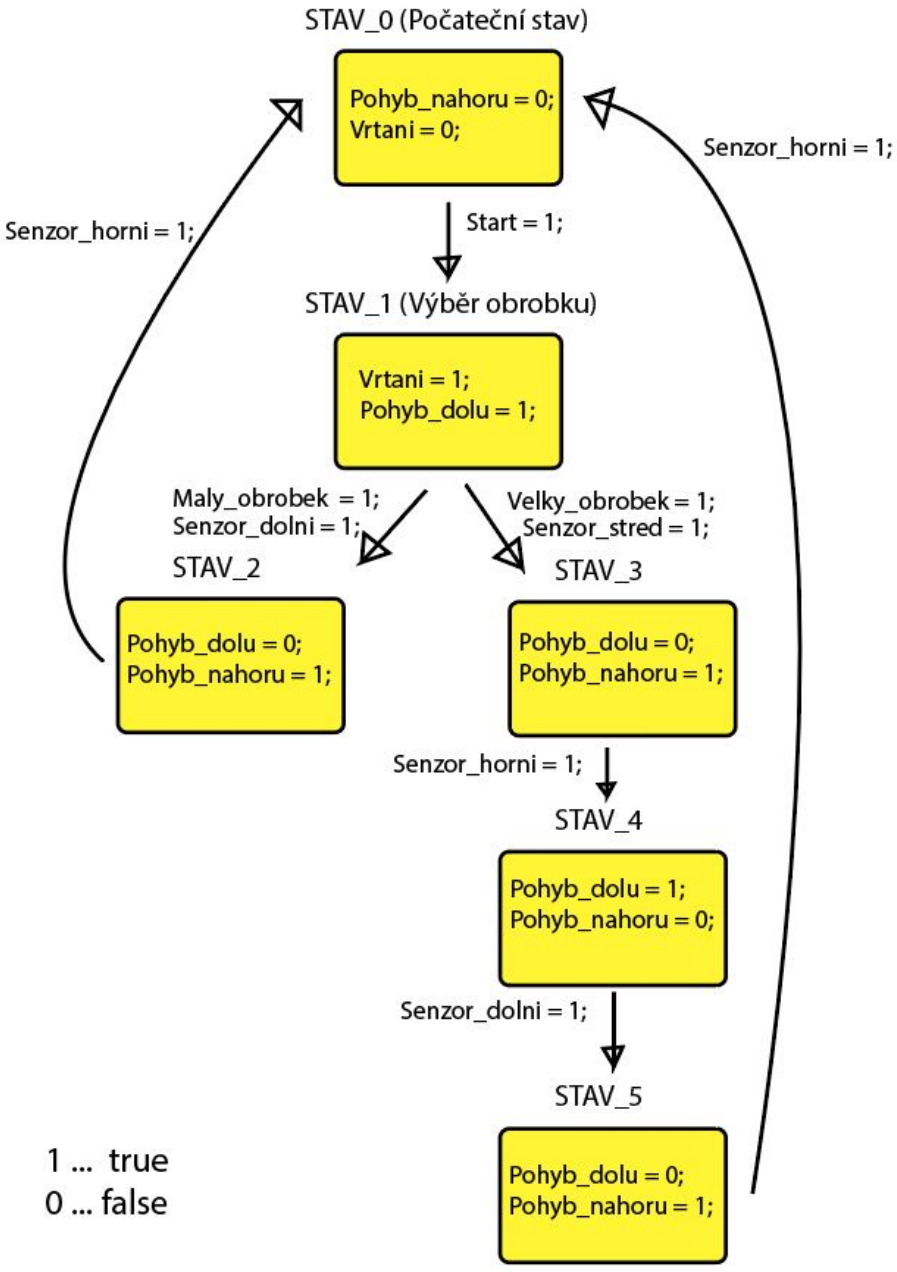
Obr. 3.3: Ukázka manuálního módu - Pohyb dolů

3.4 Automatický mód

Narozdíl od toho manuálního reaguje automatický mód pouze na tlačítko start / stop a data ze senzorů. Funguje tedy po zapnutí plně bez zásahu uživatele. Jedná se o stavový automat o šesti stavech, který si určí podle toho zda jde o malý či velký obrobek cestu, kterou bude dál postupovat (malý obrobek vrtá najednou, ten velký nadvakrtát).



Obr. 3.4: Ukázka automatického módu- stav 1



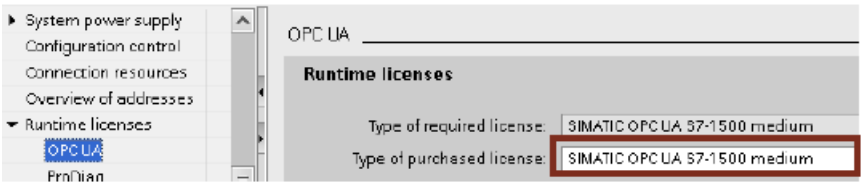
Obr. 3.5: Stavový automat automatického módu

4 KONFIGURACE OPC UA SERVERU

PLC automaty Simatic S7-1500 mají OPC UA server vestavěný přímo v sobě a dá se jednoduše nastavit v prostředí TIA Portal. Samotné nastavení se skládá ze třech základních kroků – povolení OPC UA serveru, nastavení zabezpečení a vytvoření uživatelů.

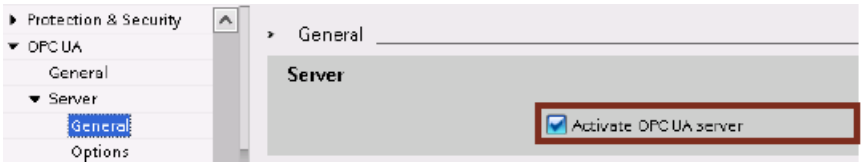
4.1 Povolení OPC UA Serveru

Prvním krokem konfigurace OPC UA Serceru je jeho povolení. To se provede tak, že přes nastavení v TIA Portalu nejprve zvolíme správnou licenci odpovídající použitému PLC, tedy licenci SIMATIC OPC UA S7-1500 small.

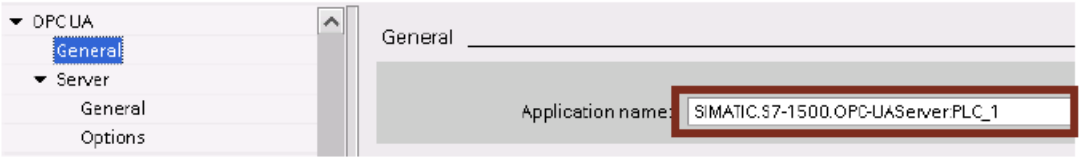


Obr. 4.1: Nastavení licence

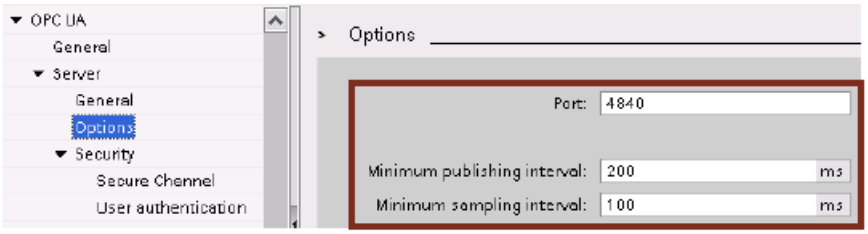
Následně se musí povolit aktivování OPC UA Serveru, zvolit jeho jméno a nastavit pro daný port minimální interval pro publikování a minimální vzorkovací interval. Tyto hodnoty říkají, jak často je server oprávněn posílat data klientovi a jak často může požadovat změnu dat po CPU. Poté už stačí celý projekt nahrát do PLC.



Obr. 4.2: Aktivování OPC UA Serveru



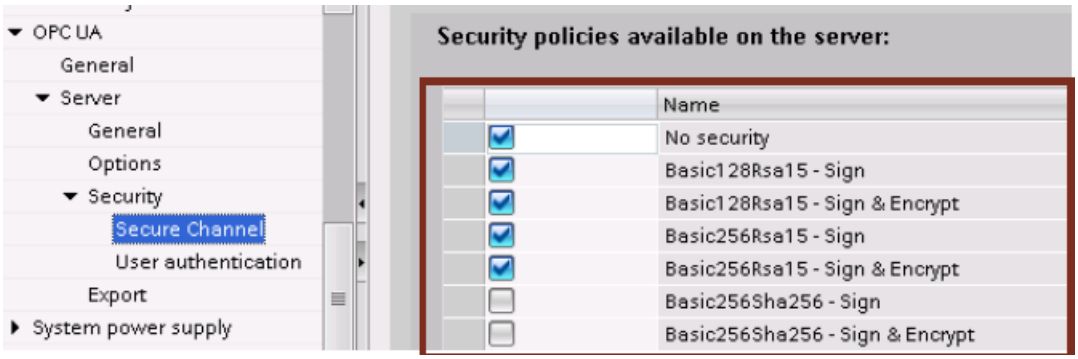
Obr. 4.3: Pojmenování OPC UA Serveru



Obr. 4.4: Nastavení licence

4.2 Nastavení zabezpečení

Aby šly používat softwarové certifikáty pro OPC UA Server je nutné nastavit jeho zabezpečení. Tedy v TIA Portalu povolit globální nastavení zabezpečení projektu a vytvořit si profil uživatele, kde se jen vyplní jméno a heslo. Následně lze specifikovat způsob šifrování a autentizace mezi klientem a OPC UA Serverem. Pro každý vybraný bezpečnostní prvek dojde k vytvoření samostatného koncového bodu, ke kterému se klient může připojit. Po dokončení tohoto nastavení se opět musí celý projekt nahrát do PLC.



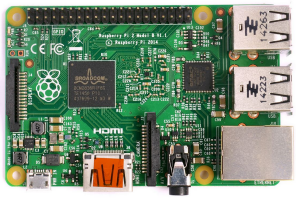
Obr. 4.5: Specifikace šifrování a autentizace

5 IMPLEMENTACE OS ANDROID

5.1 Android na Raspberry PI 3

Operační systém android není oficiálně kompatibilní s Raspberry Pi, proto jsem použil pouze experimentální verzi Androidu - konkrétně verzi 7.1.1 (Nougat) pro tablety. Tyto verze, které jsou stále ve vývoji, jsou dostupné na stránkach zabývajícími se Raspeberry a nejsou úplně bezchybné. Dají se sehnat i propracovanější, plně testované verze Androidu, ale ty už nejsou k dostání zdarma. Pro účely této práce je však použitá verze dostačující.

Pro svou práci jsem si vybral konkrétně model Raspberry PI 3 B+. Jedná se o jednočipový malý počítač. Oproti modelu 2 má 64 bitový procesor a integrovaný wi-fi model.



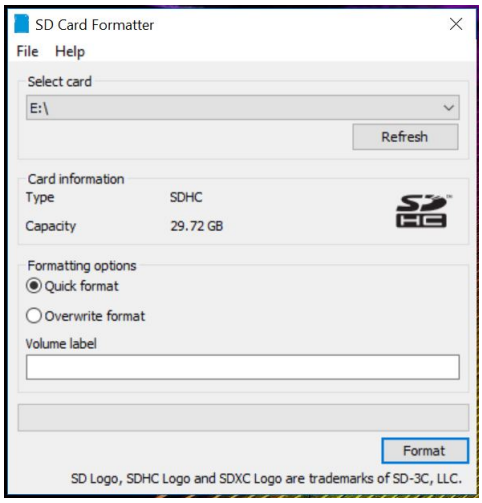
Obr. 5.1: Ukázka Raspberry PI 3

5.2 Formatování SD karty

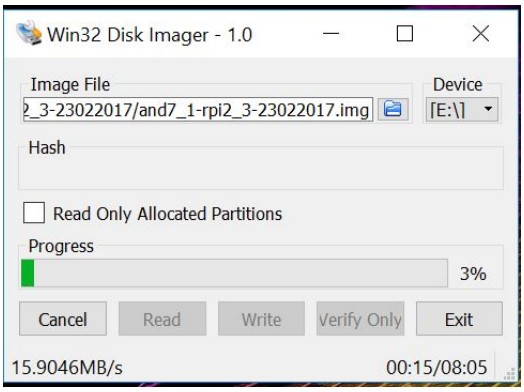
První krok při implementaci OS Android do Raspberry je předpřipravení SD karty, která musí mít nejméně velikost 8 GB (samotný OS nám zabere téměř 7,5 GB). K vyčištění karty byl použit volně dostupný program SD Card Formatter, kde se jen vybere adresář k formátovatováni.

5.3 Implementace OS Android

Dalším krokem je stáhnutí systému a jeho nahrání na kartu. Jelikož je Android open source, jak již bylo dříve řečeno, je volně dostupný pro všechny. Android se nám stáhne jako .img soubor. Pro přepsání systému na kartu je použit další volně dostupný program – Win 32 Disk Imager. Tady je nutné vybrat stáhnutý .img soubor a adresář (SD kartu), na který se má nahrát. Vše se potvrdí tlačítkem write a pak už stačí vyčkat na dokončení.



Obr. 5.2: Formatování SD karty



Obr. 5.3: Přepis systéme na SD kartu

Po úspěšném nahrání na kartu lze bezpečně odpojit SD kartu od počítače a vložit ji do Raspberry PI 3. Dále se připojí ethernetový kabel a zdroj napájení. Obrazovka je k Raspberry připojena pomocí HDMI kabelu. První bootování systému trvá přibližně 5 minut, při dalším spuštění už je však o dost rychlejší.

6 HMI APLIKACE PRO OS ANDROID

HMI Aplikace (Human Machine Interface) je, jak už název napovídá, rozhraní mezi strojem a člověkem.

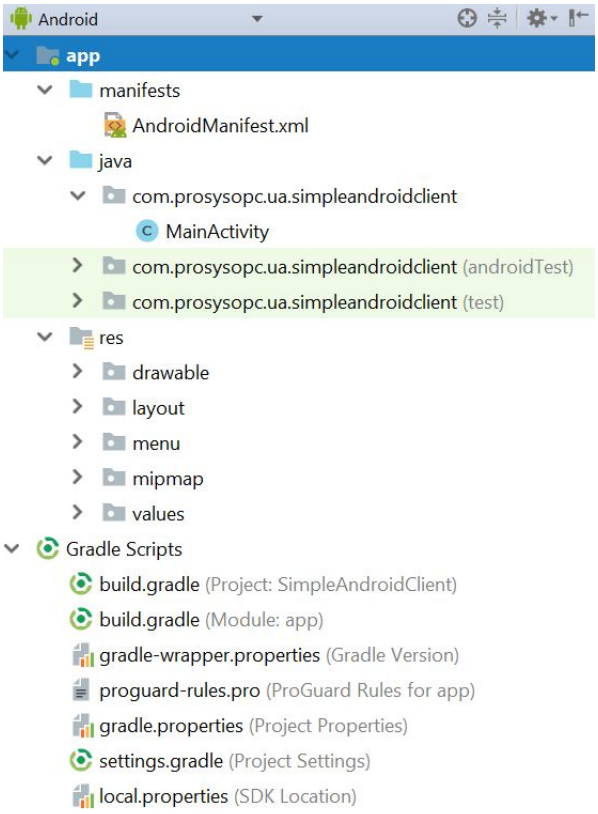
Cílem této práce je vytvořit HMI aplikaci která by běžela na OS Android. Jako vývojové prostředí jsem si vybral Android Studio. Nejprve je nutné do projektu implementovat všechny potřebné knihovny. Na to slouží v Android Studiu speciální soubor gradle. V tomto souboru se nachází nastavení nutné k sestavení projektu jako například minimální a maximální podporovaná verze sdk. Právě tam, do kolonky závislosti (dependencies) se vkládají knihovny.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 27
    buildToolsVersion "27.0.3"
    defaultConfig {
        applicationId "com.prosysopc.ua.simpleandroidclient"
        minSdkVersion 17
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

}
```

Obr. 6.1: Ukázka ze souboru gradle

6.1 Struktura projektu

Strukturu projektu v Android Studiu lze vidět v levém sloupci a dělí se na dvě hlavní části - složka Java, kde jsou uloženy jednotlivé kódy a složka res (resources) neboli zdroje. V téhle složce se nachází veškeré grafické podklady pro aplikaci, jako jsou styly, rozvržení oken a obrázky.



Obr. 6.2: Struktura projektu

6.2 Manifest

Jeden z nejdůležitějších souborů při tvorbě aplikace je manifest. Musí se nacházet v kořenovém adresáři a obsahuje informace o aplikaci, které předá androidu předtím, než se spustí samotný kód aplikace. V manifestu se kromě nastavení jednotlivých activit definují i práva samotné aplikace a mnohé další věci jako například ikona.

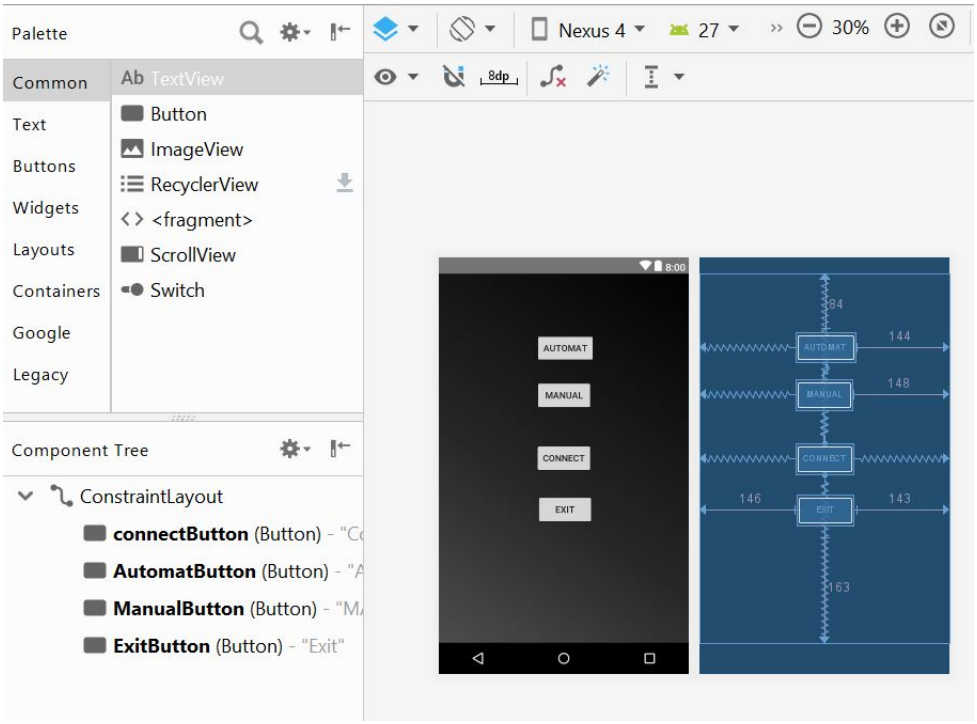
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.prosysopc.ua.simpleandroidclient">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Obr. 6.3: Ukázka kódu manifestu

6.3 Grafické rozhraní

Ve složce layout jsou uložené rozložení aplikace, v našem případě se jedná o hlavní stránku a stránky zvlášť pro manuální vrtání a automatické vrtání. Rozdíl mezi těmito dvěma módy je v tom, že manuální vrtání používá jen zápis dat (je tedy mnohem jednoduchší) a v automatickém vrtání jde hlavně o čtení dat a zobrazení informací uživateli.

Grafiku lze tvořit v Android studiu dvěma způsoby, k dispozici je textový editor a design editor. Oba jsou navzájem propojené, tedy změny provedené v jednom z nich se automaticky promítají i do toho druhého. Design editor se hodí především pro hrubý nástin aplikace, kdy pak jednotlivé tlačítka a další detaily ladíme v textovém rozhraní.



Obr. 6.4: Ukázka grafické části editoru

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/pozadi"
    android:theme="@style/ThemeOverlay.AppCompat.ActionBar"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Dark"
    tools:context="com.prosysopc.ua.simpleandroidclient.MainActivity">

    <Button
        android:id="@+id/connectButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Connect"
        android:onClick="ConnectButtonRun"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Obr. 6.5: Ukázka textové části editoru

6.4 Kód aplikace

6.4.1 Připojení k serveru

Připojení k serveru jsem kvůli problémům s vyhledávaním endpointů nakonec řešil tak, že jsem vložil do kódu aplikace přímo url serveru, na který se má klient připojit. Konkrétně tam mám implementovány dvě adresy url mezi kterými jde volit- první od serveru běžícího na Plc a druhý od serveru běžícího na mém počítači, kde jsem pomocí aplikace Prosys Simulation Server testoval svou aplikaci. Připojení k serveru neproběhne automaticky po spuštění aplikace, ale až po stisknutí některého z tlačítek se vytvoří nové Session.

Celou operaci můžeme rozdělit do tří základních kroků. Nejprve vytvoříme ua klienta, kde mu předaváme url adresu serveru. Následně dochází k nastavení na straně klienta - jméno a url adresa. Vytvoří se certifikát a nastaví se úroveň zabezpečení. Poté se pokusíme navázat spojení se serverem. Při selhání připojení vyskočí informační dialogové okno. Celá tato operace se odehrává v asynchroním vlákně na pozadí aplikace. Je to především z toho důvodu, aby nedocházelo ke zpomalování aplikace. Navíc Android neumožňuje přístup k síti v hlavním vlákně.

```
client = new UaClient(url1);

initialize(client);

client.setTimeout(60000);
client.setSecurityMode(SecurityMode.NONE);
client.setUserIdentity(new UserIdentity());
client.connect();
```

Obr. 6.6: Ukázka připojení k serveru

```
protected void initialize(UaClient client) throws SecureIdentityException, IOException, UnknownHostException {
    ApplicationDescription appDescription = new ApplicationDescription();
    appDescription.setApplicationName(new LocalizedText( s: "SimpleAndroidClient", Locale.ENGLISH));

    String android_id = Settings.Secure.getString(
            getApplicationContext().getContentResolver(),
            Settings.Secure.ANDROID_ID);

    appDescription.setApplicationUri("urn:" + android_id +":UA:SimpleAndroidClient");
    appDescription.setProductUri("urn:prosysopc.com:UA:SimpleAndroidClient");
    appDescription.setApplicationType(ApplicationType.Client);

    PkiFileBasedCertificateValidator validator = new PkiFileBasedCertificateValidator( s: getFilesDir().getPath() + "/PKI/CA");
    validator.setValidationListener(new CertificateValidationListener() {
        @Override
        public PkiFileBasedCertificateValidator.ValidationResult onValidate(
                Cert cert, ApplicationDescription applicationDescription,
                EnumSet<PkiFileBasedCertificateValidator.CertificateCheck> enumSet) {
            return PkiFileBasedCertificateValidator.ValidationResult.AcceptPermanently;
        }
    });
    client.setCertificateValidator(validator);


    final ApplicationIdentity identity = new ApplicationIdentity();
    identity.setApplicationDescription(appDescription);
    client.setApplicationIdentity(identity);
}
```

Obr. 6.7: Inicializace klienta

6.4.2 Zápis dat

Zápis dat se skládá se dvou kroků. Nejprve vybereme uzel (node), do kterého budeme chtít zapisovat data. Ten můžeme vybrat buď pomocí jeho názvu - definujeme ho textovým řetězcem String, nebo přes jeho id, pokud ho známe. Přistupovat k uzlům přes id je rychlejší, pokud tedy používáme daný uzel často, je vhodné si ho na začátku programu deklerovat pomocí stringu a následně ho volat už jen přes jeho id.

```
protected String doInBackground(String... strings) {
    String url = strings [0];
    String vyber_node = strings[1];
    UaClient client = null;
    String result = null;
    try {

        client = new UaClient(url1);

        initialize(client);

        client.setTimeout(60000);
        client.setSecurityMode(SecurityMode.NONE);
        client.setUserIdentity(new UserIdentity());
        client.connect();

        NodeId nodeCounter = new NodeId( i: 3, vyber_node);

        boolean status = client.writeValue(nodeCounter,  o: 0);
        System.out.println(status);

        client.disconnect();
    } catch (Exception e) {
        result = e.toString();
        //System.out.println(e);
    }
    return result;
}
```

Obr. 6.8: Ukázka zápisu dat

6.4.3 Čtení dat

Při čtení dat máme více možností. Jednoduchá verze čtení je velmi podobná zápisu dat, kdy v danou chvíli žádame server o data z jednoho uzlu. Tato metoda je však velice neefektivní a proto tu je funkce client.read(). Ta umožnujě čtení dat z několika uzlů najednou. Avšak ani tohle řešení se pro naší aplikaci moc nehodí, protože čte data jen na zavolání. To co hledáme je průbežné čtení dat na pozadí.

Nakonec jsem čtení vyřešil pomocí subscription, které moniturují změny na serveru a reagují na ně. Po vytvoření subscription do něj přidáme položky, které chceme sledovat. V mém případě se jedná o vstupy ze senzoru snímající polohu vrtací hlavice a kontolky led diod.

```
MonitoredDataItem dataItem = new MonitoredDataItem(id);
Subscription subscription = new Subscription();
subscription.addItem(dataItem);
client.addSubscription(subscription);
```

Obr. 6.9: Přidání sledované proměnné

Jen samotné subscription vytvořit nestačí, dalo by se říct, že se jedná jen o seznam proměnných, které se mají monitorovat. Nyní je potřeba něčeho, co bude změny odposlouchávat. K tomu účelu slouží takzvaný MonitoredDataItemListener. V něm definujeme, jak má aplikace na změnu hodnoty reagovat.

```
MonitoredDataItemListener dataChangeListener = new MonitoredDataItemListener()
{
@Override
public void onDataChange(MonitoredDataItem sender, DataValue prevValue, DataValue value)
{
    System.out.println("Monitor Data Change successfully");
    if(value!=null)
    {
    System.out.println("————————>"+value.getValue().getValue().toString());
    }
}
};
dataItem.setDataChangeListener(dataChangeListener);
```

Obr. 6.10: MonitoredDataItemListener

6.4.4 Vyskakovací okna

Vyskakovací okna jsou v této aplikaci použita pro potvrzení některých důležitých kroků uživatelem jako je vypnutí aplikace. Dále pak jako oznamovací okna v případě nenavazání připojení a podobně.

Každé nové vyskakovací okno se vytváří samostatně pomocí builderu. Atributy setTitle a setMessage poslouží k zobrazení textu. V tomto druhu vyskakovacích oken lze nastavit až dvě tlačítka odpovědí - pozitivní a negativní. Pod první parametr text se vkládá název tlačítka a do druhého jeho akci, kterou má provést. Je tu i možnost zápis skrátit do (dialog, which) -> ExitRun().

```
AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
builder.setTitle("Exit");
builder.setMessage("Opravdu chcete ukoncit aplikaci?");
builder.setPositiveButton( text: "Ano", ((dialog, which) -> ExitRun() ));
builder.setNegativeButton( text: "Ne",((dialog, which) -> dialog.cancel()) );
builder.show();
```

Obr. 6.11: Ukázka kódu vyskakovací okna

Zvláštní případ vyskakovacího okna je takzvaný výběrový list. Dá nám na výběr z několika prvků, ze kterých zvolíme jeden. Ten pak bude jako výstupní hodnota tohoto dialogového okna. Oproti předchozím zmíněným vyskakovacím oknům je poměrně složitější. Abychom mohli odchytit vybraný prvek, použijeme funkci OnClickListener(). Ta odchytí id prvku, na který jsme klikli. Poté už můžeme vypsat v oznamovacím okně info uživateli o tom, co si vybral a celý dialog zavřít.

6.4.5 Animace

Animace v aplikaci jsou velmi jednoduché, řešené pomocí takzvaných přepinačů obrázku (Image switcher). Nejprve je potřeba každý přepínač založit na začátku kódu a to příkazem:

private ImageSwitcher led_switch;

Teď když už je přepínač vytvořený je nutné ho nastavit. Aby šel nastavit, musí být aplikace v prostředí (layout), ve kterém se konkrétní přepínač nachází. Přepnutí layoutu se provede takto:

setContentView(R.layout.activity_automat);

V nastavení pak k přepínači přiřadíme jeho id, které má v layoutu. Následně ho pomocí makeView() zobrazíme. Teď už můžeme obrázek uvnitř kdykoliv měnit za jiný nacházejíci se ve složce resource / drawable. Slouží k tomu tento příkaz:

led_auto_switch = (ImageSwitcher) findViewById(R.id.imgsw_auto1);

```
led_switch = (ImageSwitcher) findViewById(R.id.imgsw);
led_switch.setFactory(new ViewFactory() {
    @Override
    public View makeView() {
        ImageView myView = new ImageView(getApplicationContext());
        return myView;
    }
});
```

Obr. 6.12: Nastavení Image Switcheru

6.4.6 Tlačítka

Důležitou součástí aplikace jsou tlačítka. Ty zajišťují interakci mezi uživatelem a samotnou aplikací. Nejprve je vytvoříme jejich grafickou část - v patřičném Layoutu nadefinujeme tlačítku všechny potřebné parametry. Pro usnadnění práce to můžeme udělat v grafickém návrhovém rozhraní, kde jen vyplníme patřičné kolonky a každá změna se automaticky převádí do xml kódu.

Některé parametry se nastavují samy, jako například id prvku. Ostatní však jsou na nás. Vyzdvihnul bych asi ty nejdůležitější jako název prvku a parametr OnClick, který říká, co se stane po stisku tlačítka - určuje funkci, která se má spustit.

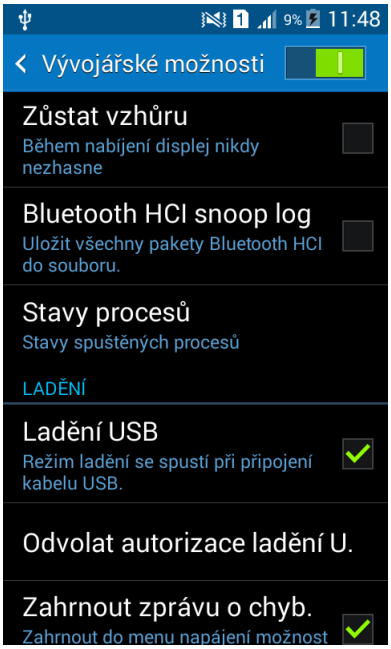
```
<Button
    android:id="@+id/StopButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp"
    android:layout_marginEnd="30dp"
    android:onClick="StopRun"
    android:text="@string/stop"
    app:layout_constraintEnd_toStartOf="@+id/guideline3"
    app:layout_constraintTop_toBottomOf="@+id/StartButton" />
```

Obr. 6.13: Ukázka xml kódu tlačítka

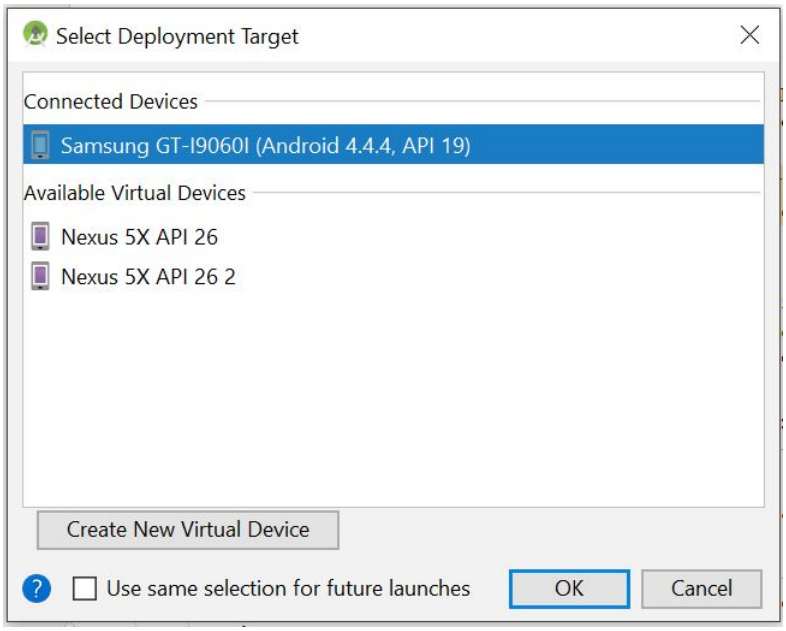
6.5 Sestavení projektu a simulace

Před samotným spuštěním apliakce je potřeba sestavit projekt. To provedeme tak, že v horním menu Android studia vybereme záložku Build -> Make Project. Chvíli to potrvá a jakmile se zobrazí dole v informačním okně, že sestavení proběhlo úspěšně můžeme přejít k spuštění aplikace. Nejprve v zařízení odemkneme vývojařský profil, přes vývojařské možnosti se dostaneme k ladění usb, které potvrdíme.

Nyní máme vše připraveno. Vedle záložky build, kterou jsme používali předtím, najdeme položku Run. Po rozkliknutí se objeví tabulka pro výběr zařízení, na kterém chceme aplikaci spusit. V zobrazeným okně máme možnost zvolit jak zařízení připojené přes usb, tak virtuální zařízení, které nám simuluje chování toho klasického, ale přímo na našem počítači.



Obr. 6.14: Nastavení zařízení

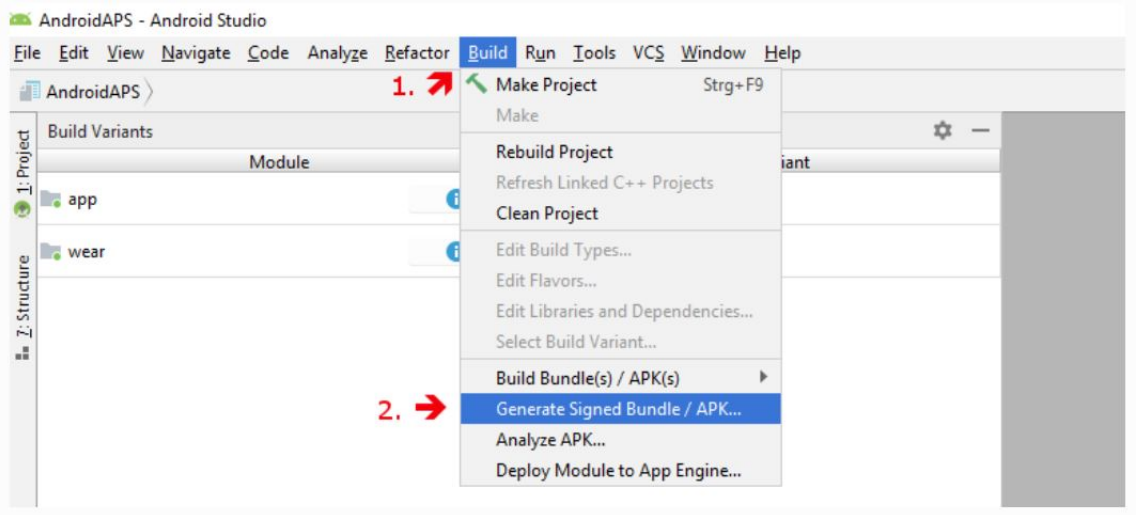


Obr. 6.15: Okno pro výběr zařízení

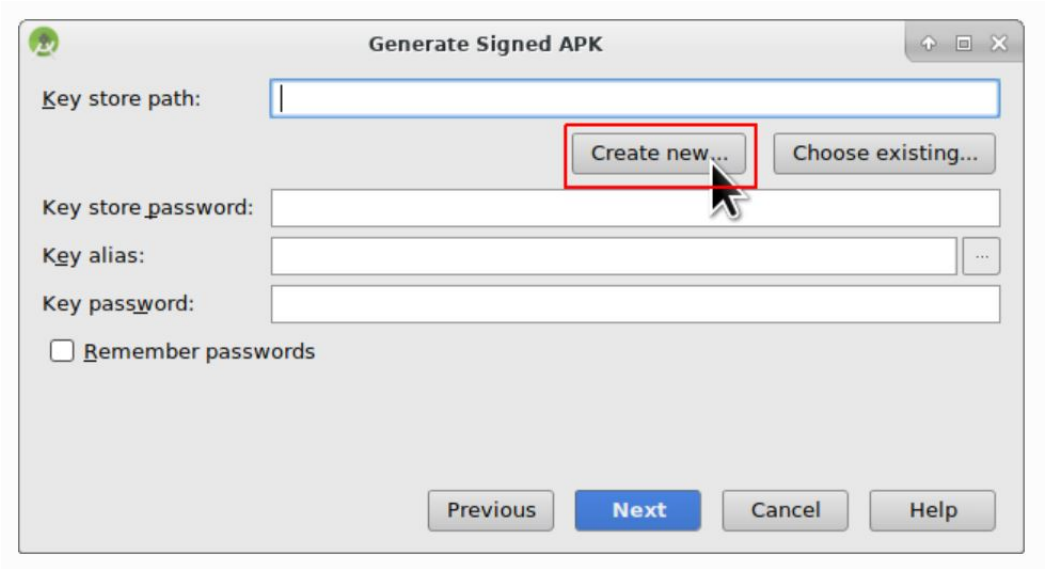
7 APK soubor a instalace do zařízení

Pro distribuci aplikace je více možností, ke každé je však potřeba nejdříve vytvořit APK soubor. V této kapitole ukážu jednoduchý postup, jak se to dá udělat v Android Studiu. APK soubor pak lze nahrát do obchodu google Play (Play store), nebo ho jednoduše přenést do zařízení, kde bude aplikace spouštěna.

Android z bezpečnostních důvodu přijímá pouze podepsaný kód, proto je nutné aplikaci digitálně podepsat. Ve vrchním panelu Android Studia jsem se přez záložku Build -> Generate Signed Bundle / APK k nastavení podpisu aplikace. V prvním dialogovém okně se zaklikne možnost APK a následně se zobrazí nastavení klíče.



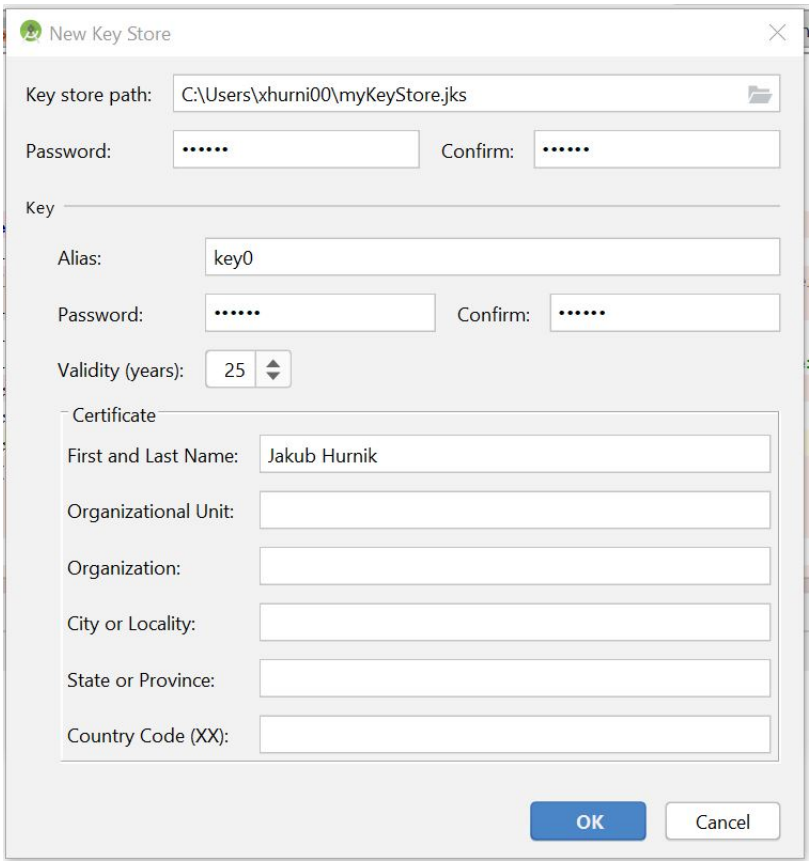
Obr. 7.1: Generování podpisu aplikace



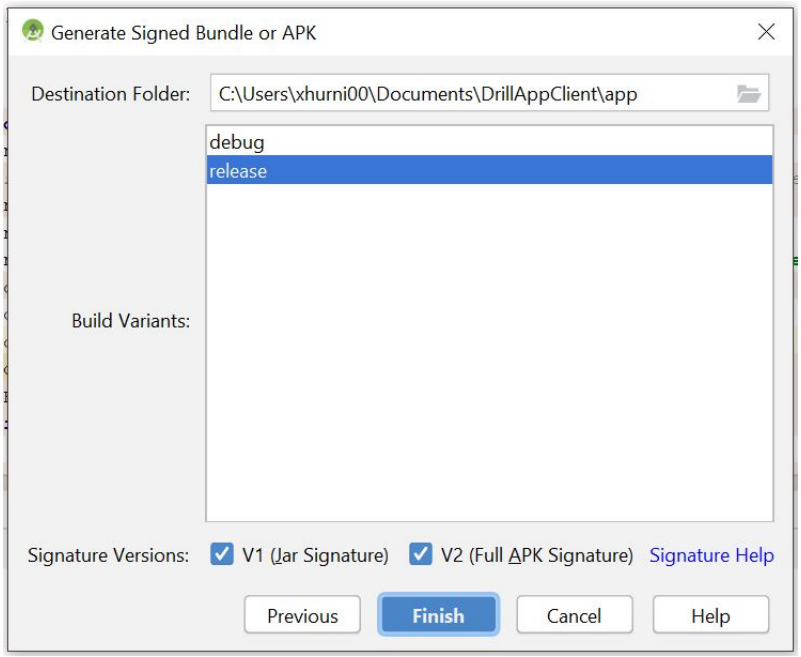
Obr. 7.2: Výběr klíče

Tam se vyplní cesta k existujícímu klíči, nebo v mém případě vytvoří nový. Po vyplnění hesel a informacích o autorovi (Povinné jsou jen položky jméno a příjmení) a potvrzení je ještě nutné vybrat typ sestavení a verze podpisu.

Že byl APK soubor úspěšně vygenerován lze vidět v kolonce Event Log. Poté už ho stačí jen vyhledat - uloží se do složky projektu app / release a přenést do zařízení. Já jsem soubor APK přenášel do Raspberry pomocí flash disku, nebo v případě tabletu přes usb kabel.



Obr. 7.3: Vytvoření nového klíče

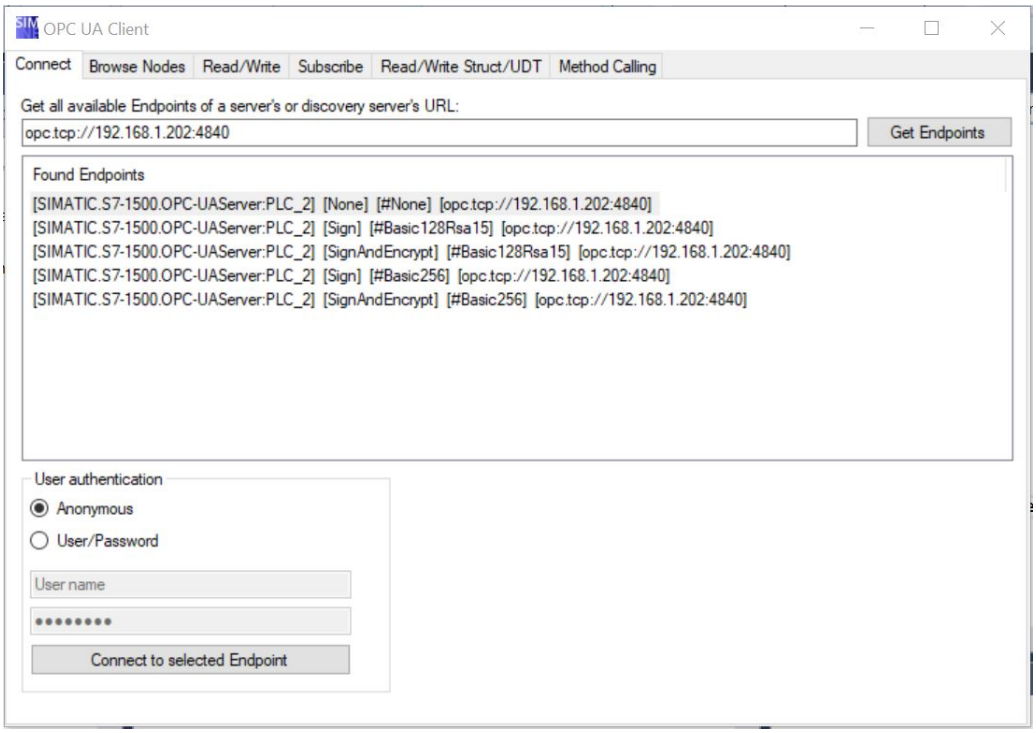


Obr. 7.4: Nastavení typu sestavení

8 Testování aplikace

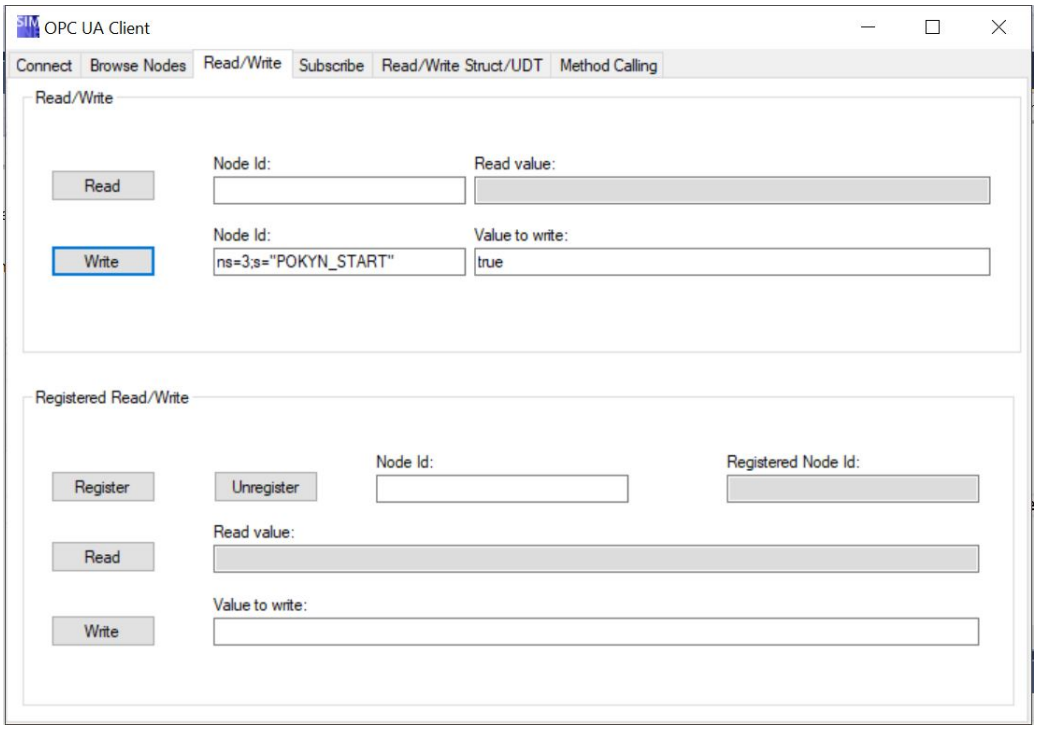
8.1 Testování Tia Portal aplikace

Abych si otestoval funkčnost programu pro plc a zároveň zjistil, jestli mám správně nastavený Opc ua server využil jsem klienta vytvořeného společnosti Siemens. Pomocí něho jsem simuloval přenos dat z hmi aplikace na server a vykonávání kódu. Klient je napsán v C++ a spouští se přes Visual studio. Po najetí uvodní stránky jsem zadal url serveru - 192.168.1.202:4840. Klient sám vyhledá všechny dostupné endpointy, kde jsem se připojil k tomu bez zabezpeční.



Obr. 8.1: Vyhledání dostupných endpointů

Přes záložku read / write jsem posílal vstupní hodnoty do plc a ověřil si tak, že plc daný program vykonává. Záznam z tohoto testování se nachází v přílohách na CD.

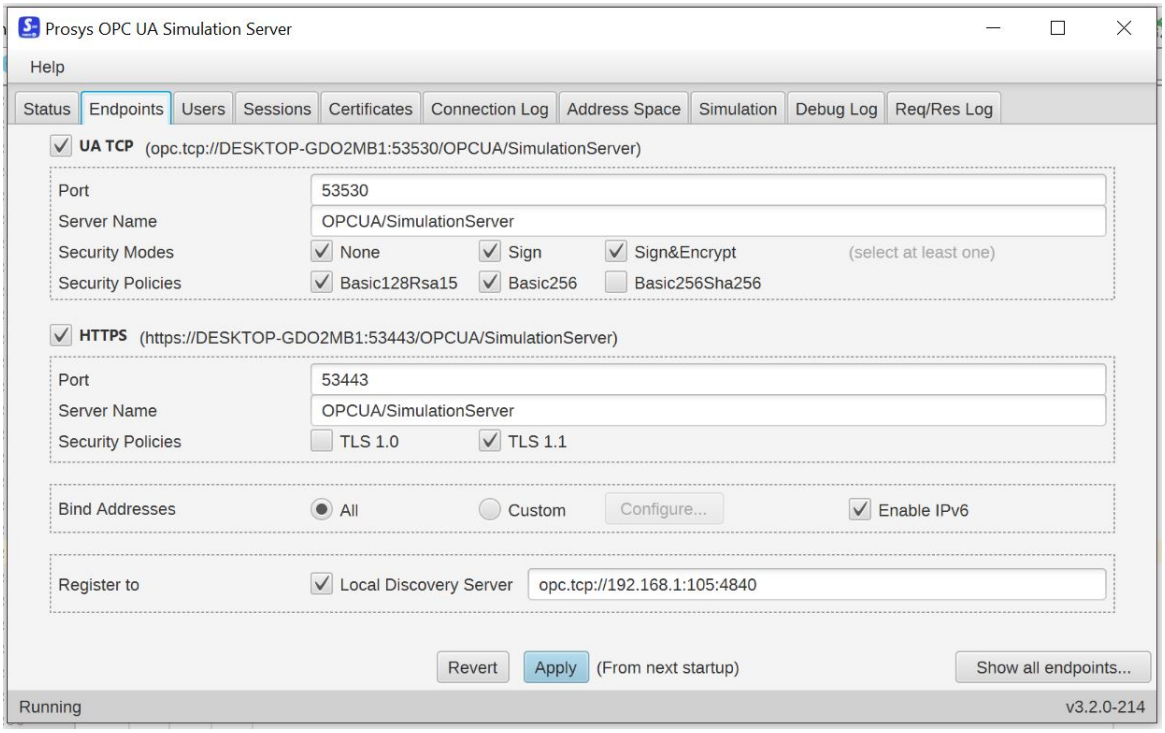


Obr. 8.2: Zápis dat při testování

8.2 Testování HMI Android aplikace

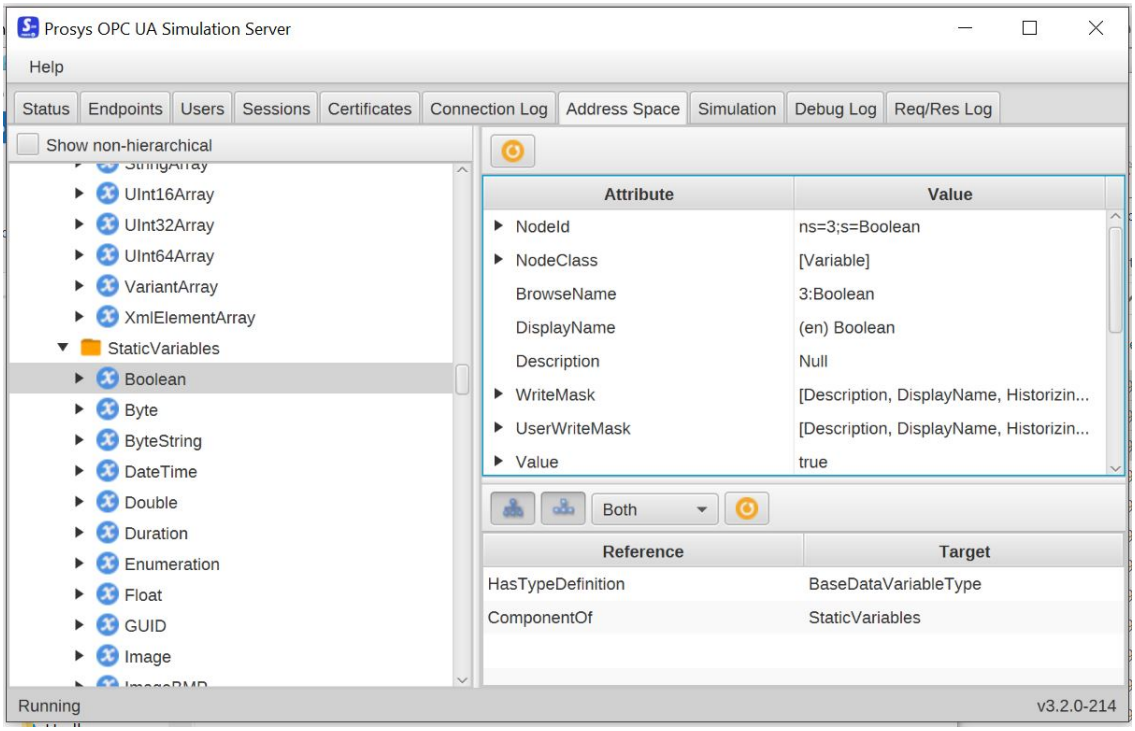
Testování Hmi aplikace bych rozdělil do dvou části - testování v reálném nasazení za použití Plc a simulace na serveru běžícím na počítači. Dále jsem i testoval obě možnosti zapojení, jak komunikaci přes wi-fi, tak propojení přímo ethernetovým kabelem, kdy jako klient posloužilo Raspberry Pi.

Jako testovací server jsem použil Prosys OPC UA Simulation Server vyvinutý společností Prosys. Tento program je po registraci na jejich stránkách k dostání zdarma a slouží právě pro účely testování opc ua klientů. Po jeho zapnutí server automaticky vytvoří endpointy s různými stupni zabezpečení vychazející z přednastavené url adresy. Ta lze sice změnit, ale je k tomu potřeba restartovat server.



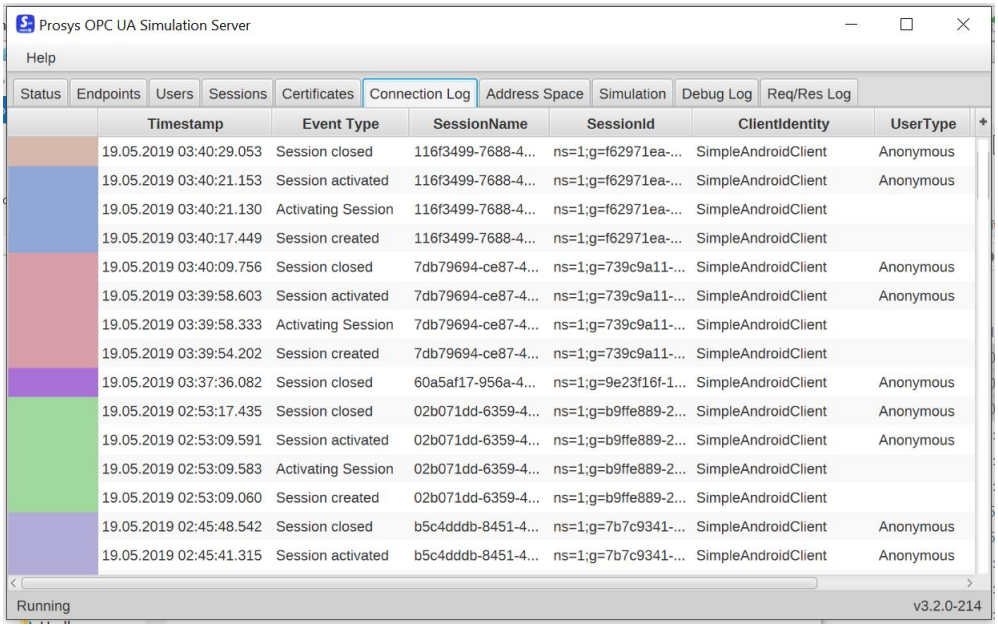
Obr. 8.3: Ukázka záložky Endpoints]

Nevýhoda tohoto serveru je, že nejde vytvářet a definovat své proměné. Je tu sice možnost nastavit si nové signály, ty ale mají náhodně generovanou hodnoutu nebo kolísají po křivce. V záložce Adress Space jsou všechny varianty proměných, které by mohlo reálné Plc vracet a právě s nimi můžeme prostřednictvím klienta interagovat.

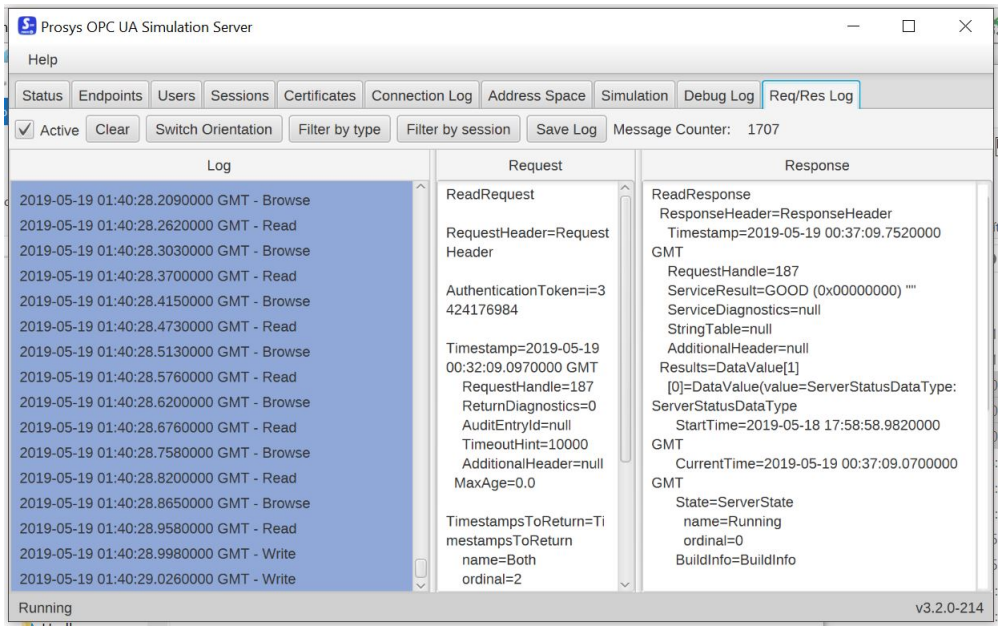


Obr. 8.4: Ukázka záložky Adress Space]

Další důležité záložky jsou Connection Log a Req / Res Log. Tyto dvě záložky jsou velmí pomocné jak při debugování, tak i při samotném chodu aplikace a graficky ukazují co se odehrává při komunikaci Opc Ua na pozadí. Connection Log zobrazuje všechny aktivní Session, které běží na serveru. Záložka Req / Res Log zachycuje požadavky klienta a přiřazuje k ním odpovědi přicházející ze serveru.



Obr. 8.5: Ukázka záložky Connection Log]



Obr. 8.6: Ukázka záložky Req / Res Log]

8.3 Testování rychlosti čtení / zápisu dat

Jako poslední test jsem provedl zkoušku rychlosti zápisu a čtení. Princip spočíval v tom, že jsem v cyklu provedl sto krát za sebou zápis / čtení dat a stopoval čas, za který se tato úloha zvládla vykonat. Tento výsledný čas jsem vydělil stem a dostal tak průměrný čas jednoho zápisu / čtení.

- Výpočet průměrné rychlosti zápisu dat
  - počátek úlohy: 6:23:08:036
  - konec úlohy: 6:23:10:975
  - rozdíl: 2.939 s

$$\mathbf{2.939/100 = 0.02939s}$$

  - průměrna rychlost: 0.02939 s

- Výpočet průměrné rychlosti čtení dat
  - počátek úlohy: 6:25:09:792
  - konec úlohy: 6:25:12:853

$$\mathbf{3.093/100 = 0.03093s}$$

  - průměrna rychlost: 0.03093 s

Z výpočtu vyplýva, že zápis dat je o trochu rychlejší, což by souhlasilo s teorií.

9 Závěr

Cílem této práce bylo vytvořit OPC UA klienta v prostředí Android sloužícího jako rozhraní mezi uživatelem a laboratorním modelem vrtačky. Model vrtačky byl připojen k PLC Simatic S7-1500. Abych toho dosáhl musel jsem si nejprve nastudovat komunikaci OPC UA a programování v Jave. Dále jsem navrhl možnosti připojení, jak přes ethernetový kabel, tak bezdrátově přes wi-fi.

Nejprve jsem v kapitole Použité technologie shrnul, co všechno jsem měl k dispozici. Začal jsem popisem modelu Vrtačka a navazoval seznámením s PLC Simatic. Pak následovala část věnující se vývojovým prostředím, z kterých jsem si vybíral. V poslední části této kapitoly jsem popsal komunikaci OPC UA.

Když už jsem měl všechnu teorii připravenou, pustil jsem se do programu ovládající PLC. Ten jsem vytvářel v prostředí Tia Portál V14. Vznikl z toho program o dvou módech (automatický / manuální) zaměřený na to, aby dobře otestoval budoucí hmi aplikaci. Ještě než jsem opustil Tia Porál aktivoval jsem a nakonfiguroval OPC UA Server, který je v použití verzi plc už implementován.

Pak přišel čas se věnovat samotnému Androidu. Nejprve jsem se zaobíral implementací OS Android do Raspberry a poté následoval tvorbou hmi aplikace. Tam bylo nejprve nutné vložit potřebné knihovny a vytvořit grafické rozhráni. Jakmile jsem měl tuto část připravenou, začal jsem vytvářet vnitřní logiku aplikace, kde všechny příkazy jako čtení nebo zápis dat probihají na pozadí a jen přenášejí informace do hlavního vlákna. Grafická část je propojená s tou logickou pomocí interaktivních prvků (tlačítek).

Nakonec už zbývalo jen aplikaci sestavit a otestovat. Testoval jsem jak funkčnost plc programu, tak aplikace a to pomocí volně dostupných simulačních serverů a klientů od společností Siemens a Prosys.

Na závěr musím říct, že aplikace má určité nedostatky a je tu tedy místo pro budoucí zlepšování. Především se jedná o to, že v současném stavu nelze vybírat endpoint, ke kterému se chci připojit, ale je pevně daný. Další nevýhoda je, že aplikace jede v režimu nulového zabezpečení, což by v praxi představovalo velký problém.

Literatura

[1] SIEMENS AG *Profit from engineering efficiency – Totally integrated Automation Portal in practice* [online]. 2015 [cit. 2017-11-16]. Dostupné z: <https://c4b.gss.siemens.com/resources/images/articles/dffa-b10091-00-7600.pdf>.

[2] PÁSEK Jan. *Programovatelné automaty v řízení technologických procesů*. Brno, 2007.

[3] PÁSEK, Jan, ŠTOHL, Radek. *Programovatelné automaty – Laboratorní cvičení*.

[4] CZIPSZER, Jiří. *Řízení vrtačky – Inovace laboratorních úloh*. Brno, 2017.

[5] BALDA, Pavel. *Úvod do OPC Unified Architecture*. ZČU v Plzni, 2006.

[6] MAHNKE,Wolfgang, Stefan-Helmut LEITNER a Matthias DAMM. *OPC Unified Architecture*. Springer-Verlag Berlin Heidelberg, 2009. ISBN 978-3-540-68898-3.

[7] Jirgl,Miroslav. *Laboratorní cvičení BPGA - Cvičení se systémy SIEMENS* Brno, 2018

[8] Jouni Aro, Heikki Tahvanainen. *OPC UA Enables Secure Data Transfer and System Integrationsin Private and Public Networks*. 2015

[9] FOXON. *Co je OPC?* [online]. [cit. 2018-01-01] Dostupné z: <https://www.foxon.cz/cs/blogs/80-co-je-opc-opc-server-opc-klient-.html>.

[10] PROMOTIC. *Komunikace přes rozhraní OPC UA*.[online]. [cit. 2018-01-01] Dostupné z: <https://www.promotic.eu/cz/pmdoc/Subsystems/Comm/OPC/OPCUA.html>.

[11] SIEMENS. *STEP 7*.[online]. [cit. 2018-01-01] Dostupné z: <http://www1.siemens.cz/ad/current/index.php?vw=0&ctxnh=3a01fb9720&ctxp=home>.

[12] ANDROIDAPLIKACE. *Co je to ten Android?*.[online]. [cit. 2018-01-04] Dostupné z: <http://androidaplikace.cz/index.php/co-je-operacni-system-android/>.

[13] ECLIPSE. *Desktop IDEs*.[online]. [cit. 2018-01-04] Dostupné z: <http://www.eclipse.org/ide/>.

[14] NETBEANS. *NetBeans IDE - The Smarter and Faster Way to Code*. [online]. [cit. 2018-01-04] Dostupné z: <https://netbeans.org/features/index.html>.

[15] ZDROJAK. *Android Studio – nové vývojové prostředí*.[online]. [cit. 2018-01-04] Dostupné z: <https://www.zdrojak.cz/clanky/android-studio-nove-vyvojove-prostredi/>.

[16] ANDROID. *Everything you need to build on Android*. [online]. [cit. 2018-01-04] Dostupné z: <https://developer.android.com/studio/features.html>.

[17] BOSÁK, Tomáš. *Úvod do programovacieho jazyka Java*. [online]. 2006 [cit. 2018-01-04] Dostupné z: <http://programujte.com/clanek/2006041804-uvod-do-programovacieho-jazyka-java/>.

[18] QOS.ch. *SLF4J user manual*. [online]. 2019. [cit. 2019-05-05] Dostupné z: <https://www.slf4j.org/manual.html/>.

[19] Prosys OPC Ltd. *OPC UA SDK for Java*. [Online]. 2019. [cit. 2019-05-05] Dostupné z: <https://www.prosysopc.com/products/opc-ua-java-sdk/technical-details/>.

[20] The Apache Software Foundation. *HttpClient Overview*. [Online]. 2019. [cit. 2019-05-05] Dostupné z: <https://hc.apache.org/httpcomponents-client-4.5.x/index.html/>.

[21] PCC. *Simatic S7-1500*. [Online]. 2019. [cit. 2019-05-05] Dostupné z: <https://www.pccweb.com/2015/11/10/simatic-s7-1500-with-tia-portal/>.

[22] GES Electronics. *D2F-L2 / OMRON*. [Online]. 2019. [cit. 2019-05-05] Dostupné z: <https://www.ges.cz/cz/d2f-l2-GES06503961.html>.

Seznam symbolů, veličin a zkratek

OPC	OLE for control process
OLE	Object Linking and Embedding
UA	Unified Architecture
HMI	Human Machine Interface
PLC	programmable logic controller
OS	operating system
LED	Light-Emitting Diode
CPU	Central Processing Unit
LAN	Local Area Network
URL	Uniform Resource Locator
API	Application Programming Interface
APK	Android Package Kit

Seznam příloh

A Obsah přiloženého CD

- /
  - `DrillAppClient` ..... HMI Android aplikace
    - `app`
      - `libs` ..... Knihovny
      - `release` ..... Apk soubor určený k instalaci
      - `src` ..... Kód aplikace
    - `build`
    - `gradle`
  - `Model_Vrtacka_TiaPortalV14` ..... Program pro PLC
  - `Bakalarska_Prace-HMI_Aplikace_v_prostredi_Android_komunikujici_pomoci_OPC_UA-Jakub_Hurnik.pdf` ..... Elektronická verze dokumentace bakalářské práce