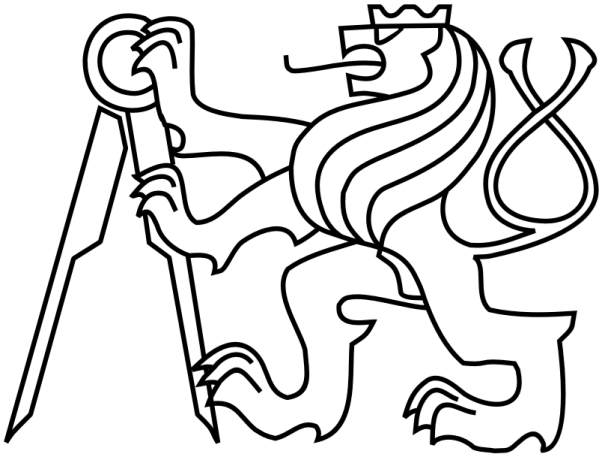
ČESKÉ VYSOKÉ UČENIE TECHNICKÉ V PRAHE

FAKULTA STROJNÍCKA

**Ústav výrobných strojov a zariadení**



# Diplomová práca

**Monitoring stavu stroja pomocou vlastnej SCADA aplikácie**

**2021** **Bc. Dávid Križan**

# ZADÁNÍ DIPLOMOVÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Križan** Jméno: **Dávid** Osobní číslo: **466409**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav výrobních strojů a zařízení**

Studijní program: **Průmysl 4.0**

Studijní obor: **bez oboru**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Monitoring stavu stroje pomocí vlastní SCADA aplikace**

Název diplomové práce anglicky:

**Custom SCADA application for monitoring machine states**

Pokyny pro vypracování:

Popis tématu: Cílem práce je za pomoci volně dostupných knihoven zprovoznit komunikaci s řídicím systémem stroje skrz OPC UA protokol a vytvořit uživatelskou obrazovku se stavy stroje s využitím nástrojů tvorby uživatelských rozhraní; Osnova práce: Rešerše SCADA systémů, možnosti jazyka python, rešerše nástrojů tvorby uživatelských obrazovek, tvorba simulačního datového modelu, zprovoznění komunikace OPC UA server - klient, ukládání sbíraných dat do databáze, tvorba uživatelského rozhraní; Rozsah grafické části: 0; Rozsah textové části: 60-80 stran;

Seznam doporučené literatury:

[1] ZEZULKA, František. Prostředky průmyslové automatizace. VUTIUM, 2004; [2] MAHNKE, Wolfgang; LEITNER, Stefan-Helmut; DAMM, Matthias. OPC unified architecture. Springer Science & Business Media, 2009; [3] LACKO, Ľuboslav. 1001 tipů a triků pro SQL. Computer Press, 2015.

Jméno a pracoviště vedoucí(ho) diplomové práce:

**Ing. Štěpán Fiala, Ph.D., ústav výrobních strojů a zařízení FS**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **29.04.2021** Termín odevzdání diplomové práce: **25.07.2021**

Platnost zadání diplomové práce: **30.09.2021**

| Ing. Štěpán Fiala, Ph.D.<br>podpis vedoucí(ho) práce | Ing. Matěj Sulitka, Ph.D.<br>podpis vedoucí(ho) ústavu/katedry | prof. Ing. Michael Valášek, DrSc.<br>podpis děkana(ky) |
|---|---|---|

## III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Podpis studenta

Datum převzetí zadání

**Prehlásenie**

Prehlasujem, že som svoju diplomovú prácu vypracoval samostatne a že som uviedol v priloženom zozname všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržiavaní etických princípov pri príprave vysokoškolských záverečným prací, vydaným ČVUT v Prahe 1.7.2009.

Nemám závažný dôvod proti užitiu tohto školského diela § 60 Zákona č.121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon).

V Prahe dňa 20.7.2021 …….

podpis

**Poďakovanie**

Moje poďakovanie patrí vedúcemu práce Ing. Štěpánovi Fialovi, Ph.D. za cenné rady pri vypracovaní tejto diplomovej práce. Taktiež by som sa chcel poďakovať rodičom a všetkým, ktorí ma podporovali pri vysokoškolskom štúdiu.

# Anotácia

| | |
|---|---|
| Autor: | Bc. Dávid Križan |
| Názov DP: | Monitoring stavu stroja pomocou vlastnej SCADA aplikácie |
| Rozsah práce: | 67 strán, 42 obrázkov, 2 tabuľky |
| Školský rok vyhotovenia: | 2021 |
| Škola: | ČVUT – Fakulta strojnícka |
| Ústav: | Ú12135 – Ústav výrobných strojov a zariadení |
| Vedúci DP: | Ing. Štěpán Fiala, Ph.D. |
| Zadávateľ: | ČVUT – FS |
| Využitie: | Monitorovanie strojov vo výrobnom závode |
| Kľúčové slová: | monitorovanie, scada, programovanie, opc-ua, priemysel 4.0, užívateľské rozhranie |
| Anotácie: | Sledovanie stavu stroja zohráva v dnešných podnikoch dôležitú úlohu nielen pri monitorovaní samotného výrobného procesu, ale môže byť mocným nástrojom aj pre management výroby, napríklad pri plánovaní kapacít alebo predikcii údržby. Cieľom tejto diplomovej práce je za pomoci voľne dostupných knihovní a open-source programov vytvoriť komunikáciu s riadiacim systémom pomocou OPC UA serveru. Pre zobrazenie získaných dát bude vytvorené grafické užívateľské rozhranie a historické dáta budú uložené do databázového systému. Nástroj bude otestovaný na reálnom zariadení. |

# Annotation

| | |
|---|---|
| Author: | Bc. Dávid Križan |
| Diploma thesis topic: | Custom SCADA application for monitoring machine states |
| Extent : | 67 pages, 42 figures, 2 tables |
| Academic year: | 2021 |
| University: | CTU – Faculty of Mechanical Engineering |
| Department: | Ú12135 – Department of Production Machines and Equipment |
| Supervisor: | Ing. Štěpán Fiala, Ph.D. |
| Submitter: | CTU – Faculty of Mechanical Engineering |
| Application: | Monitoring of machines in industrial company |
| Keywords: | monitoring, scada, programming, opc-ua, industry 4.0, user interface |
| Annotation: | Monitoring of machine states is important for today´s industrial companies not only for the control of the process itself but can be also powerful tool for management in problems of capacity planning or predictive maintenance. Therefor the aim of this thesis is to set communication channel using opc-ua protocol and open-source programs. Graphical user interface will be made to show machine´s states and historical data will be stored in database. This tool will be tested on real machine. |

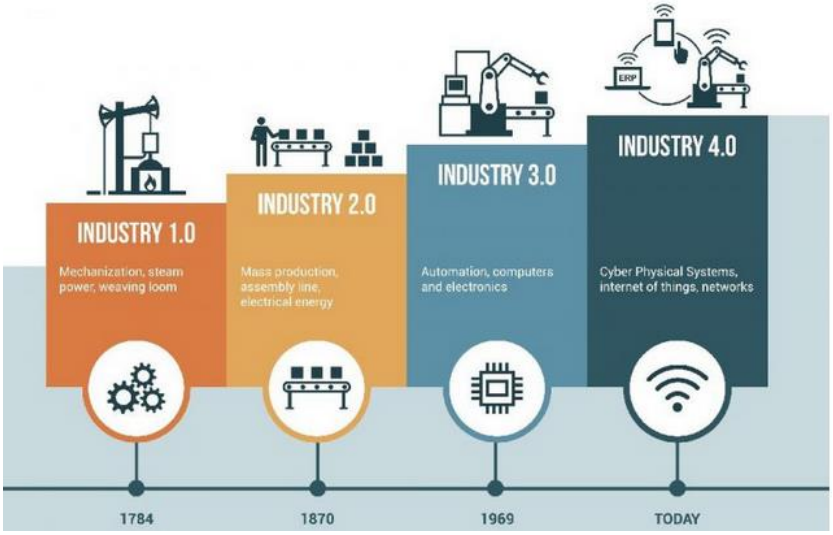
# Zoznam použitých skratiek a veličín

| | |
|---|---|
| SCADA | Supervisory Control And Data Aquisition |
| GUI | Graphical User Interface |
| HMI | Human-Machine Interface |
| IoT | Internet of Thing |
| MES | Manufacturing Execution System |
| MOM | Manufacturing Operations Management |
| ERP | Enterprise Resource Planning |
| MIS | Management Information System |
| PLC | Programmable logic controller |
| OPC UA | Open Platform Communications United Architecture |
| XML | Extensible Markup Language |
| HTML | HyperText Markup Language |
| Q&A | Questions And Answers |
| CSS | Cascading Style Sheet |
| JIT | Just In Time |
| PHP | Hypertext Preprocessor |
| SQL | Structured Query Language |
| 3D | Three Dimensional |
| STL | StereoLithoGraphy |
| VPN | Virtual Private Network |
| °C | Stupeň Celsia |

# Obsah

# 1 Úvod

Štvrtá priemyselná revolúcia je v dnešných dňoch často skloňovaným pojmom. Ide o trend, ktorý so sebou prináša množstvo zmien, či už vo fungovaní podniku, ale aj na trhu práce. Mnoho ľudí si to spája s robotizáciou a automatizáciou, čo je ale nesprávna interpretácia pojmu. Zmyslom tejto „revolúcie“ je digitalizácia – komunikácia, interakcia medzi dielčími časťami výrobného celku. Základom pre tento koncept je Internet of Thing (internet veci), to znamená konektivita všetkých zariadení tak, aby ich riadenie mohlo prebiehať na diaľku. V praxi je možné tento koncept vidieť napríklad vo forme monitorovacích systémov. Tieto systémy majú za úlohu zbierať dáta a kontrolovať ich. Ak takému systému pridáme možnosť riadenia procesu, ide o kyberneticko-fyzikálny systém splňujúci predstavy priemyslu 4.0. Existuje mnoho spoločností, ktoré ponúkajú takéto systémy na mieru, avšak ich ceny sú často vysoké a prevádzka, resp. údržba takéhoto systému vyžaduje ďalšie náklady. Alternatívnym riešením je vlastný SCADA systém, ktorý môže byť vytvorený s využitím open – source softvérov, jeho funkcionalita je takmer neobmedzená a odpadá hrozba, že firma, ktorá je zodpovedná za systém, napríklad zbankrotuje a tento systém sa pôsobením času stane nepoužiteľný. Cieľom tejto práce je vytvorenie takéhoto systému SCADA, ktorý umožní vzdialené monitorovanie a riadenie stroja – vytvorenie užívateľského rozhrania a uchovávanie stavu stroja v databáze. Na obr. 1 je možné vidieť stupne priemyselnej revolúcie a ich stručnú charakteristiku.




*obr. 1: Vývoj priemyslu [1]*

# 2 Interakcia obsluhy so strojom

Interakcia človeka s moderným strojom prebieha pomocou grafického prostredia, ktoré má za účel jednoduchou formou poskytovať údaje o stroji operátorovi. Pomocou grafického rozhrania je takisto možné stroj ovládať.

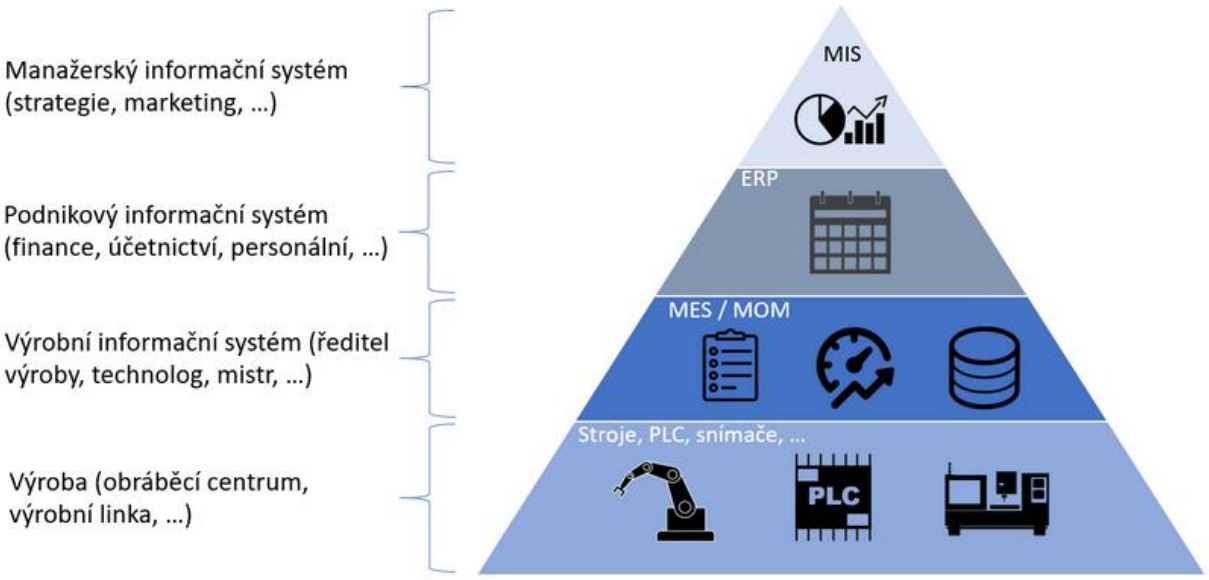
## 2.1 Rozhranie človek-stroj (HMI)

V spojitosti so SCADA systémom sa často spomína aj skratka **HMI** (**h**uman **m**achine **i**nterface), čo sa dá voľne preložiť ako rozhranie človek-stroj, ide teda o sprostredkovanie komunikácie medzi strojom a operátorom daného stroja. Aj keď nejde priamo o dispečerské riadenie, tak je nutné to zdôrazniť, nakoľko princíp je veľmi podobný. Každý moderný stroj už dnes obsahuje HMI panel, čo je zariadenie (obrazovka), ktoré sa väčšinou nachádza v bezprostrednej blízkosti stroja a dáva informáciu operátorovi o aktuálnom stave stroja, prípadne sa na ňom zobrazujú akčné tlačidlá pre spustenie určitej činnosti alebo aj pracovné inštrukcie, poprípade chybové hlášky. To znamená, že niekde na pozadí musí prebiehať zber dát zo snímačov a senzor. Rozdiel oproti SCADA systém je ten, že toto riešenie nepotrebuje vytvárať databázu a uchovávať dáta z minulosti (alebo len v obmedzenej miere) , zaujíma ho stav tu a teraz. Takéto riešenie môže byť takisto vzdialene prepojené tak, aby bolo možné vykonať niektoré činnosti (napr. reštart stroja) na diaľku – týmto by sme sa dostali na úroveň dispečerského riadenia procesu. Príkladom HMI panelu môže je teach-pendant (obr. 2), ktorý slúži na komunikáciu s robotom.


*obr. 2: HMI panel robotu [2]*

## 2.2 Dispečerský systém (SCADA)

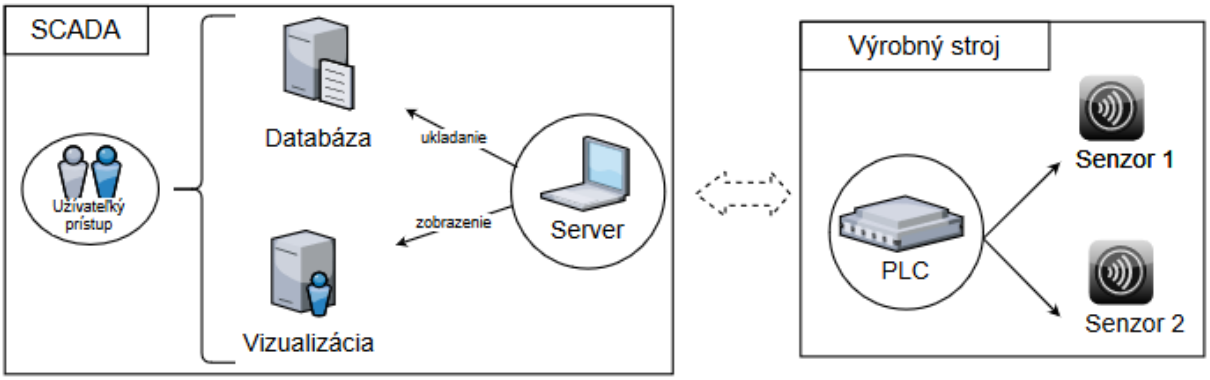
Označenie **SCADA** (**s**upervisory **c**ontrol **a**nd **d**ata **a**quisition) sa dá voľne preložiť ako dispečerské riadenie a zber dát. Je možné to chápať ako centralizáciu v podniku, to znamená, že informácie z výrobných strojov, liniek, staníc sú zachytené a sprístupnené napríklad pre oddelenie technológie, ktoré má vďaka tomuto systému prehľad o výrobných procesoch, bez nutnosti fyzickej prítomnosti na danom pracovisku. Takýto systém by mal dokonca umožniť kompetentným osobám zasahovať v prípade nutnosti do procesu výroby (napr. úprava rýchlosti pohybu výrobnej linky). Na obr. 3 je zobrazená hierarchia moderného podniku. SCADA systém patrí do úrovne výrobných informačných systémov (MES). Nie je teda primárne určený pre najvyšších manažérov firmy, ale pre ľudí priamo zodpovedných za chod výroby – riaditeľ výroby, technológ, majster. Avšak výstupné dáta byť využívané v rámci ďalších informačných systémov, napr. ekonomickým oddelením (kontrola čerpaného paliva, počtu použitých dielov, ...) alebo oddelením marketingu (spoľahlivosť procesu, chybovosť).




*obr. 3: Hierarchia podniku [3]*

Už aj z tohto jednoduchého obrázku je možné odvodiť funkciu tohto systému. Jeho úlohou je zbierať dáta z podriadených celkov (resp. ich riadiacich systémov) a ukladať ich do databázy

pre ďalšie využitie v nadriadených informačných systémoch. Takisto je vhodné, ak takýto systém má vytvorené užívateľské prostredie, ktoré môže zobrazovať aktuálny stav a zároveň je možné pomocou prednastavených tlačidiel ovládať určité funkcie podriadených systémov.




*obr. 4: Schéma SCADA systému*

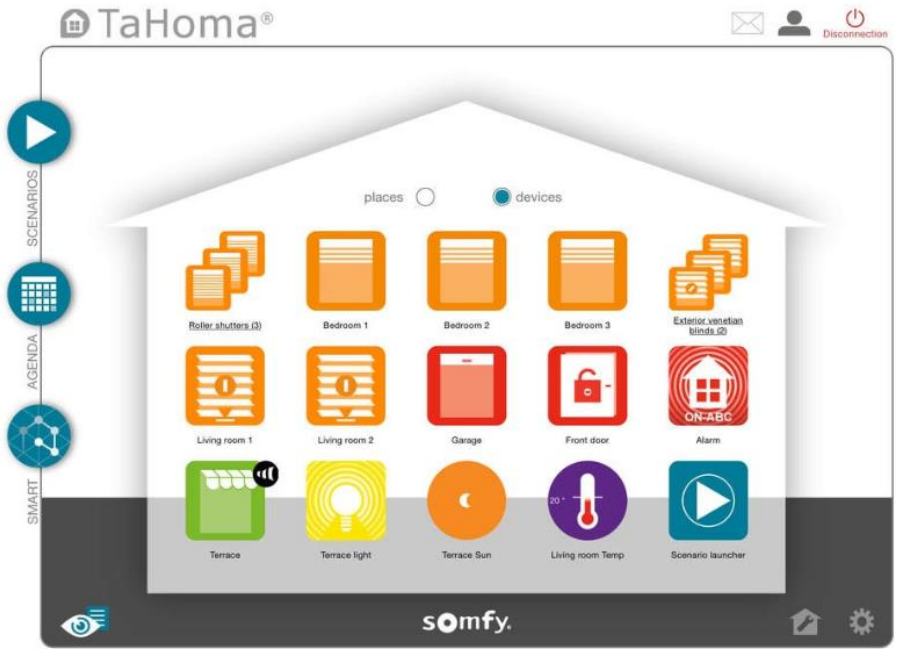
Schéma na obr. 4 zobrazuje zjednodušenú stavbu SCADA systému. Na pravej strane obrázku sa nachádza časť výrobný stroj, ktorá reprezentuje ľubovoľný výrobný celok (ohraňovací lis, linka, robot) v podniku. Väčšina dnešných strojov obsahuje senzorické systémy, ktoré sú previazané s riadiacim systémom pomocou PLC – programovateľného logického automatu. Toto zariadenie zbiera pomocou pripojených senzorov dáta, ktoré budú následne prenesené pomocou ľubovoľného komunikačného kanála (ethernet, wi-fi, ..) na vzdialený server. Následne je možné tieto informácie ukladať do predpripravenej databázy, poprípade vytvoriť ich vizualizáciu. K týmto dvom častiam musí byť vytvorený užívateľský prístup (autorizovaný) tak, aby bolo možné zobraziť tieto dáta z ľubovoľného zariadenia v sieti. To je hlavná podstata tohto systému – zobraziť/zhromažďovať dáta bez nutnosti fyzickej prítomnosti pri stroji.

## 2.3 Grafické užívateľské rozhranie (GUI)

**GUI (g**raphic **u**ser **i**nterface) alebo inak grafické užívateľské rozhranie je termín, ktorý už priamo súvisí so zobrazovaním údajov. Príkladom môže byť mobilná aplikácia, pomocou ktorej je možné skontrolovať stav vlastnej chytrej domácnosti hoci aj z druhej strany zemegule. Ide teda o určitú formu vizualizácie, ktorá umožňuje lepšie sa orientovať v dostupných dátach. Základným formátom pre zobrazovanie informácií sú jednotky a nuly. Keď sa vrátim k chytrej domácnosti, tak by to znamenalo, že ak by mi napríklad svietilo svetlo v kuchyni, tak by hodnota jeho senzoru (označme ho k) bola 1. Senzor na ďalšie svetlo (označme ho z), ktoré je zhasnuté, by mal mať hodnotu 0. Takto pozbierané údaje môžeme zapísať napríklad takto:

- k = 1
- z = 0

V tomto prípade by to ešte bolo celkom pochopiteľné a aj prehľadné. Ale predstavme, že takýchto senzorov bude 20. A tu vznikol problém s prehľadnosťou a zrozumiteľnosťou nazbieraných dát. Z tohto dôvodu bol vyvinutý spôsob zobrazenia dát, ktorému hovoríme grafické užívateľské rozhranie, kde pomocou interaktívnych grafických prvkov zobrazujeme informácie tak, aby bolo zrozumiteľné a jednoduché aj pre laika ovládať takýto systém. Na obr. 5 je uvedený príklad, ako môže vyzerať takéto rozhranie.




*obr. 5: Chytrá domácnosť [4]*

## 2.4 Prostriedky tvorby užívateľských a dispečerských rozhraní

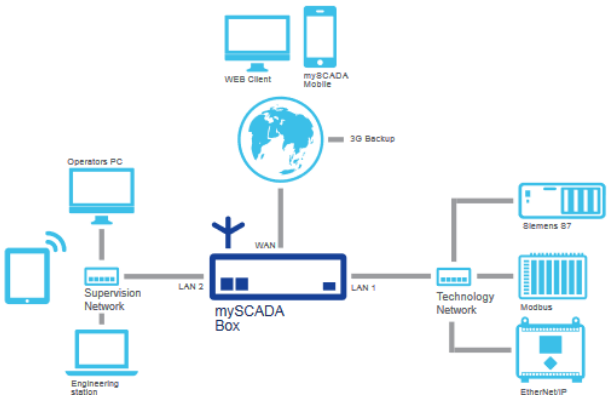
Existujú dve základné varianty, akými je možné riešiť systém SCADA v podniku. Jedným prístupom je využitie komerčných softvérov, ktoré sú priamo určené pre systém SCADA, druhou možnosťou je vytvorenie vlastného systému, na mieru.

### 2.4.1 Komerčne dostupné varianty

Na trhu sa objavuje niekoľko výrobcov takýchto systémov. Vo väčšine prípadov ide o profesionálne, platené softvéry so širokou funkcionalitou. V tejto časti práce sa pokúsim aspoň niektoré z nich popísať.

**MySCADA**

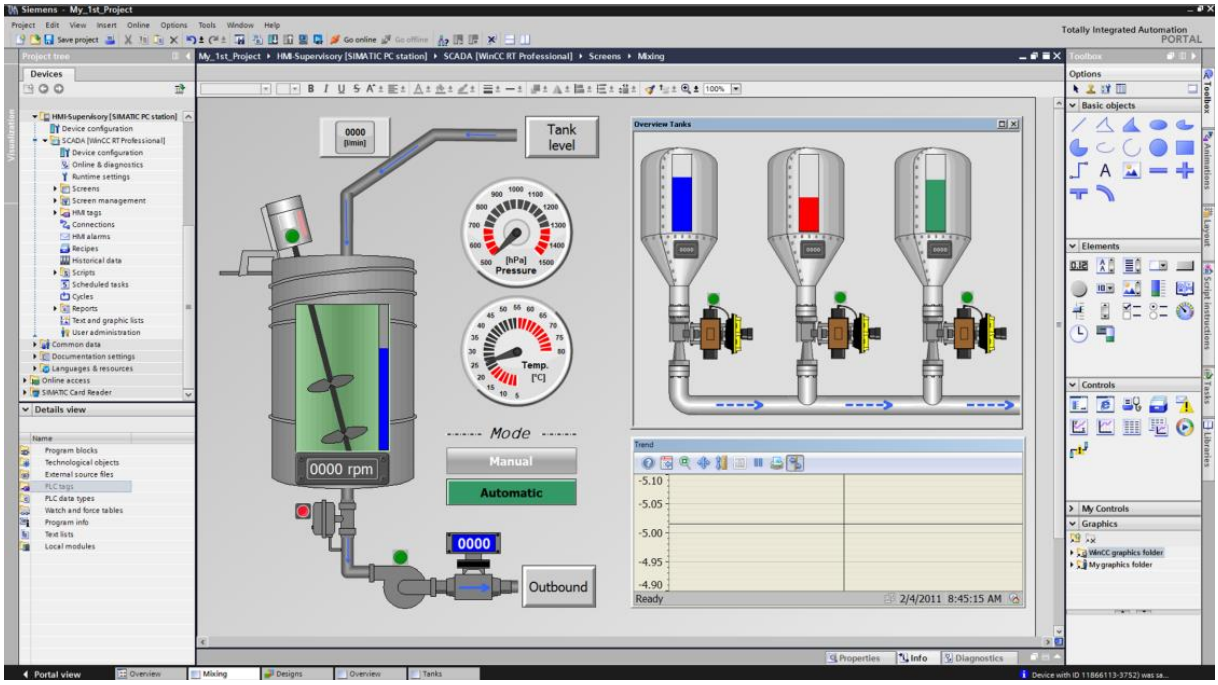
MySCADA je česká spoločnosť zaoberajúca sa zberom a vizualizáciou dát. Ich systém je vybudovaný pomocou jazyku C++, čo zaručuje funkčnosť na rôznych operačných systémoch, android nevynímajúc. Výhodná je aj široká konektivita – OPC UA, Modbus TCP (RTU), Siemens S7, EtherNet/IP, Toyopuc, Melsec-Q. Táto spoločnosť má širokú ponuku nástroj – od zberu dát, cez vytváranie vlastných HMI panelov, reportov až po kustomizované vizualizácie. Nasledujúci obr. 6 vysvetľuje prepojenie technologickej a supervizorovanej siete pomocou produktu mySCADA Box, ktorý je možno vložiť priamo do rozvádzača stroja, pripojiť PLC a pripraviť vizualizáciu procesu pomocou aplikácie myPRO inštalovanej priamo v zariadení myBox. [5]




*obr. 6: Štruktúra myBox [6]*

## SIMATIC WINCC

Je to softvér vyvinutý spoločnosťou SIEMENS, ako nástroj pre zber a vizualizáciu dát. Ako môže vyzerať vizualizácia je zrejme z obr. 7. WinCC umožňuje zber dát z PLC zariadení značky Siemens ako aj zariadení ostatných výrobcov pomocou OPC modulu. Princíp funkcie je zaznamenávanie dát z PLC a ich následné vyhodnocovanie, ako napríklad: priemerovanie, maximum, minimum a podobne. Toto vyhodnocovanie je možné nastaviť rôzne pre rôznych užívateľov, podľa potreby. Dáta, ktoré už nie sú aktuálne (závisí na nastavení systému), sú presunuté do SQL databázy ako dáta historické (zistené aj dopočítané hodnoty). Výhodou **TIA** (**t**otally **i**ntegrated **a**utomation) stratégie, v preklade úplná integrácia automatizácie, je prepojenie hodnôt z PLC priamo do programu WinCC a teda nie je nutné opakované dohľadávanie premenných a ich tvorba. Výsledkom je zníženie chybovosti v systéme. [7]




*obr. 7: WinCC vizualizácia [8]*

### 2.4.2 Vlastné riešenie grafického rozhrania

Každá aplikácia je špecifická, a preto je dnešným trendom vizualizácia na mieru. Navyše to prispieva k rýchlemu, správnemu a presnému zobrazeniu informácií. Na toto slúžia špecializované programy popísané vyššie, ktoré sú cenovo dosť nákladné a ich funkcionalita môže byť obmedzená. Navyše, ak by do budúcna existoval predpoklad pre tvorbu vlastného SCADA systému bolo by vhodnejšie vlastné riešenie.

Vytvorenie užívateľského rozhrania pozostáva z dvoch základných krokov. Prvým je vytvorenie prostredia ako takého – menu, textové polia... Ďalším krokom je napísanie kódu, ktorý bude vykonaný po stlačení určitého tlačidla alebo po vyplnení textového poľa.

Nasledujúca kapitola je zameraná na rozbor možností riešenia vlastného GUI a ich vzájomné porovnanie. Na konci kapitoly je tabuľka, kde sa na základe viackriteriálneho rozhodovania určí softvérový nástroj, ktorý bude použitý pre vlastnú tvorbu.

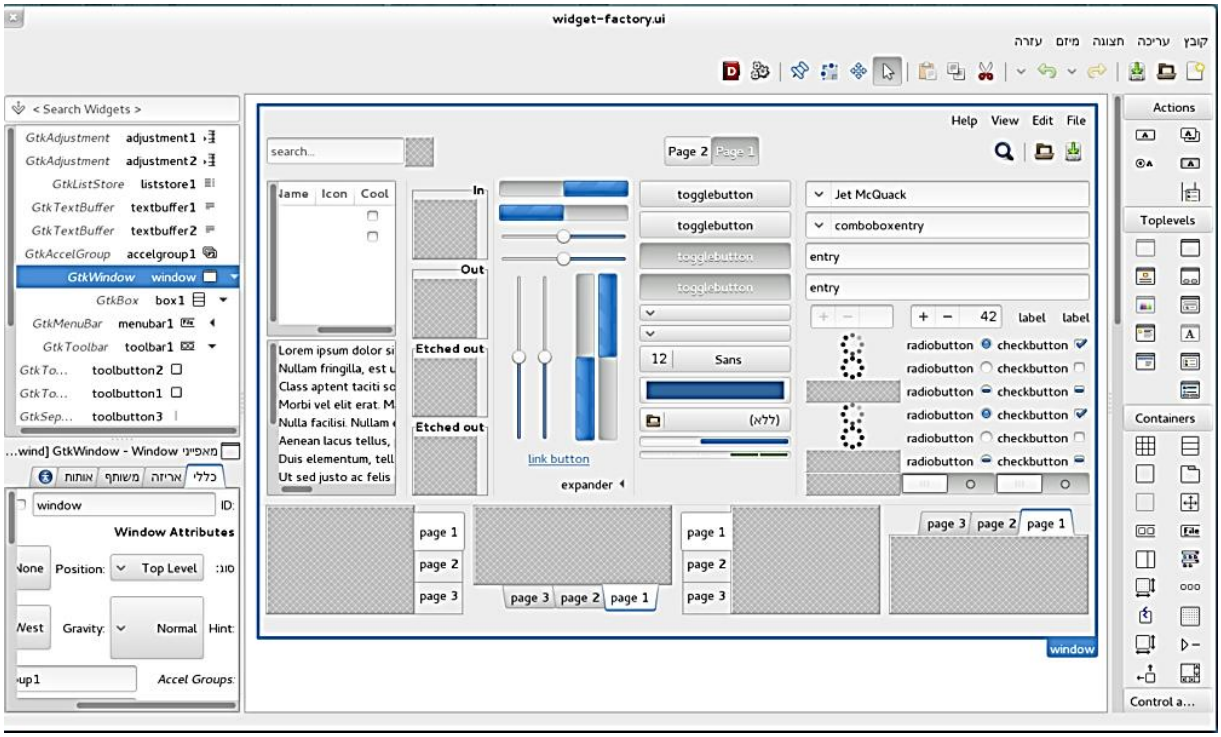
Rozhodnutie bude založené na nasledujúcich parametroch:

- Integrácia s programovacím jazykom
  - Vyjadruje komplikovanosť prenosu informácie medzi Front-end a Back-end
- Vlastné riešenie
  - Voľnosť pri vytváraní grafického prostredia
- Knižnice
  - Množstvo predpripravených materiálov, vzorov
- Zložitosť
  - Množstvo času potrebné pre tvorbu GUI (čas potrebný na učenie)
- Funkčnosť
  - Možnosť zobrazenia požadovaných informácií, aj s ohľadom na potenciálne rozšírenia
- Cena
  - Náklady na licenciu
- Vzhľad

## Glade

Je komplexný nástroj, ktorý slúži na „automatické“ generovanie grafického prostredia. Tento softvér je open-source, to znamená, že je možné ho využívať bez nutnosti akýchkoľvek licencií, poplatkov. Zjednodušene, ide o aplikáciu s intuitívnym prostredím pre tvorbu požadovaného GUI. Po jeho vytvorení je možné vygenerovať súbor vo formáte XML, ktorému rozumie množstvo bežne používaných programovacích jazykov (C/C++, Java, Python) pri použití špeciálnych knihovní. Tento XML kód sa importuje po programu pomocou určitých knihovní (napr. PyGTK pro jazyk Python). Nakoniec je nutné programovať akcie, ktoré budú vykonané po stlačení prednastavených tlačidiel, pretože Glade tvorí iba grafickú stránku veci.

Nevýhodou je, že vzhľad (obr. 8) vytvorený touto cestou je trochu zastaralý. Dôvodom sú prednastavené prvky (tlačidlá, ikony...), ktoré je možné využívať, avšak tieto prvky nie sú tak interaktívne, ako sme dnes zvyknutý napríklad z moderných stránok Facebooku alebo Instagramu. [9]




*obr. 8: Glade [10]*

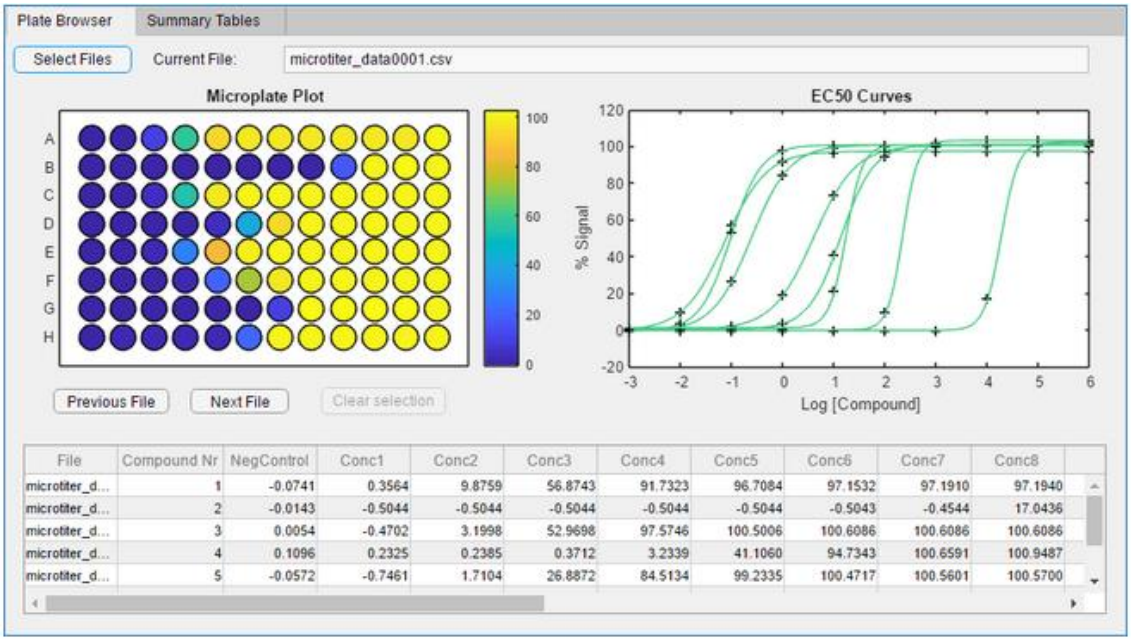
## Guide & App designer

**Matlab (mat**rix **lab**oratory), v preklade maticové laboratórium, je programovací jazyk, ktorý bol vyvinutý ako náhrada za programovací jazyk Fortran, hlavne za účelom matematických výpočtov. [11]

Dnes je vďaka dlhodobému vývoju tento program využívaný v mnohých oboroch. Je vhodný napríklad na simuláciu chovania elektrických obvodov, mechanických systémov alebo aj dátovú analýzu. Bohužiaľ už nepatrí medzi voľne šíriteľné programy a teda jeho dostupnosť je výrazne obmedzená. Základná verzia Matlabu stojí v prepočte cca 20 000 CZK/rok, bez ďalších rozšírení. [12]

Matlab ponúka intuitívne prostredie pre tvorbu rozhrania s názov **GUIDE**. Ide o kresliace plátno, do ktorého je možno pridávať prvky - sú pridávané vo forme „blokov“. Vzhľad takto vytvoreného rozhrania pôsobí zastaralým dojmom.

Ďalšou, o niečo modernejšou, možnosťou v prostredí matlabu je nástroj **APP DESIGNER** (obr. 9). Princíp tvorby je rovnaký ako v prípade GUIDE, ale prostredie je modernejšie a funkcionalita je rozsiahlejšia. Sú k dispozícii prvky ako napríklad „checkboxy“, záložky alebo kalendár.

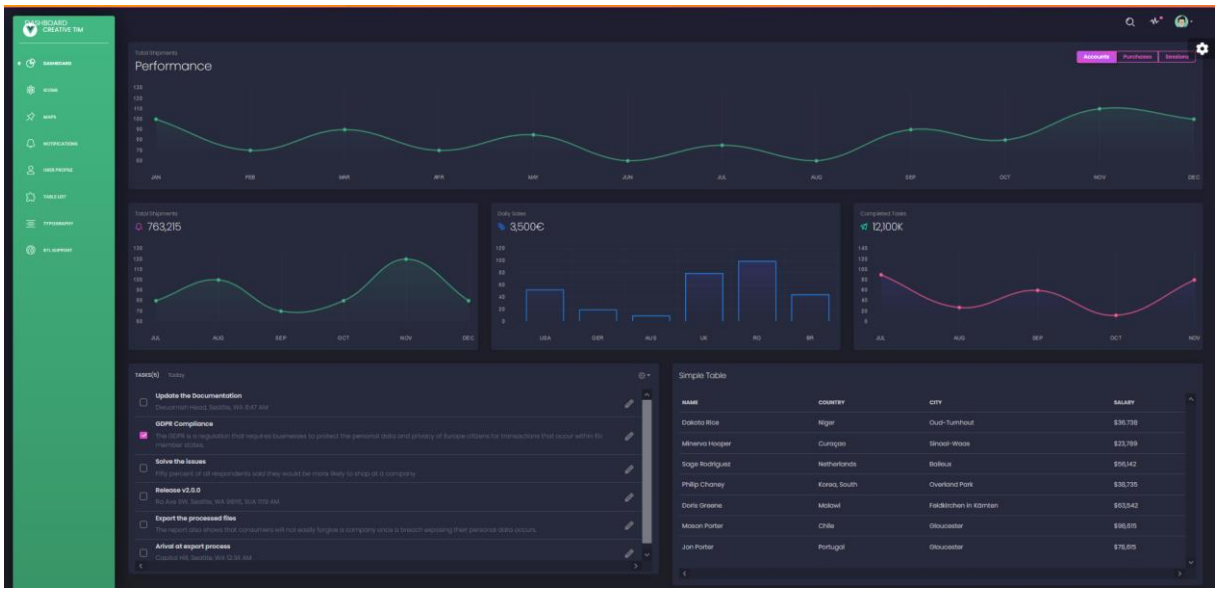



| File | Compound Nr | NegControl | Conc1 | Conc2 | Conc3 | Conc4 | Conc5 | Conc6 | Conc7 | Conc8 |
|---|---|---|---|---|---|---|---|---|---|---|
| microtiter_d... | 1 | -0.0741 | 0.3564 | 9.8759 | 56.8743 | 91.7323 | 96.7084 | 97.1532 | 97.1910 | 97.1940 |
| microtiter_d... | 2 | -0.0143 | -0.5044 | -0.5044 | -0.5044 | -0.5044 | -0.5044 | -0.5043 | -0.4544 | 17.0436 |
| microtiter_d... | 3 | 0.0054 | -0.4702 | 3.1998 | 52.9698 | 97.5746 | 100.5006 | 100.6086 | 100.6086 | 100.6086 |
| microtiter_d... | 4 | 0.1096 | 0.2325 | 0.2385 | 0.3712 | 3.2339 | 41.1060 | 94.7343 | 100.6591 | 100.9487 |
| microtiter_d... | 5 | -0.0572 | -0.7461 | 1.7104 | 26.8872 | 84.5134 | 99.2335 | 100.4717 | 100.5601 | 100.5700 |

*obr. 9: Matlab App designer [12]*

## Vue.js

Ďalšou možnosťou ako vytvoriť grafické rozhranie je vytvorenie webovej stránky (obr. 10). Táto stránka môže byť napísaná napr. v jazyku HTML. Z pohľadu toho, že dáta budú priebežne načítavané a chceme aby sa priebežne menili aj zobrazované dáta, pôjde o tzv. dynamický web – stránka nebude pevne daná pomocou HTML kódu . Na vytvorenie takejto webovej stránky je vhodné využiť Framework, napríklad Vue.js, ktorý používa hlavne jazyk html v kombinácií s Javascriptom. Pomocou pri tvorbe takejto stránky sú aj rôzne knižnice, ktoré obsahujú zbierku bežne používaných prvkov, ako napríklad tlačidlá, vyhľadávacie polia, kalendáre, kde stačí vybrať najvhodnejší prvok a prípadne ho poopraviť pre vlastnú potrebu. V prípade Vue ide napríklad o knižnicu Vuetify alebo Element.



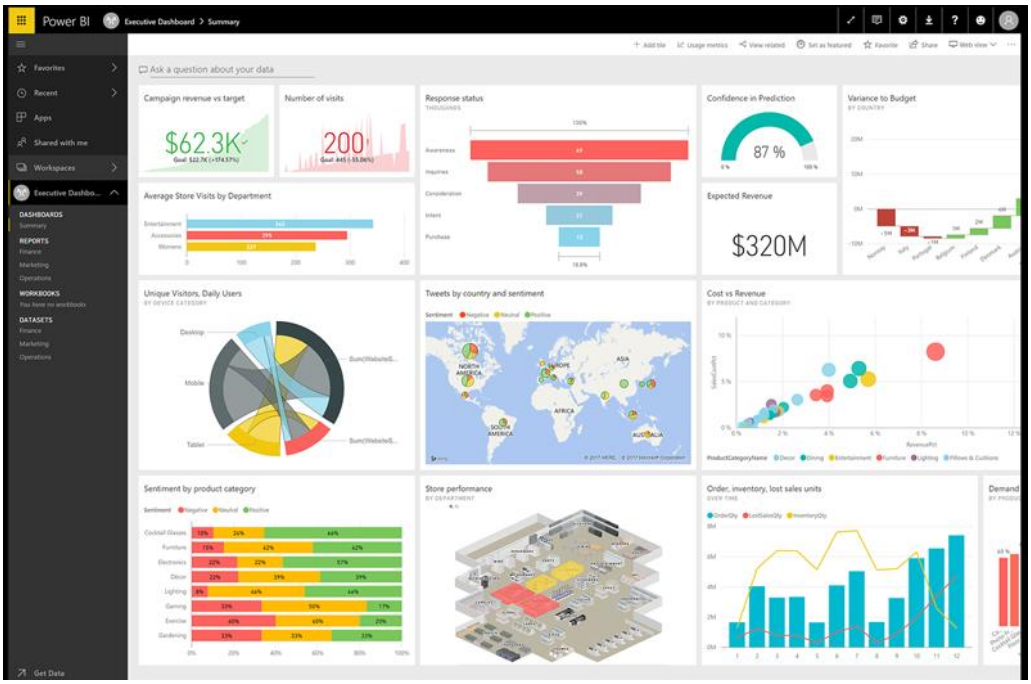

*obr. 10: Vue.js GUI [13]*

Takéto riešenie je vhodné, pretože takúto vizualizáciu je možné zdieľať na akékoľvek zariadenie. Užívateľovi stačí poznať adresu stránky a mať oprávnenie na prezeranie. Výhodu oproti predchádzajúcim riešeniam je to, že možnosti nie sú obmedzené tvorcom systému. Je možné si vyrábať vlastné prvky, čo by v prípade budovania vlastného SCADA systému bolo veľmi žiadané. V začiatku je tvorba rozhrania náročnejšia, avšak po vypracovaní vlastnej knižnice by bolo možné tvoriť grafické prostredie veľmi efektívne.

## Power BI

Softvér od spoločnosti Microsoft, ktorý je určený na obchodnú analýzu. Informácie sú zobrazené formou nástenky (obr. 11), ktorú je možné nastavovať/zobrazovať bez nutnosti programovania. Zobrazené údaje v grafoch alebo tabuľkách sú interaktívne – napríklad je možné jednoducho vyberať, ktoré údaje chceme zobraziť v grafe alebo je možné škálovať grafy – napríklad postupne vybrať mesiac, týždeň, deň, hodinu, v ktorej ma zaujímajú hodnoty, ktoré chcem zobraziť. Z nazbieraných dát je takisto možné zobraziť rôzne nástenky, napríklad jednu tabuľku pre kanceláriu technológie, ďalšiu pre finančné oddelenie a inú pre manažéra. Veľmi užitočnou funkciou sú reporty. Je možné nastaviť si presný čas, v ktorý je snímka nástenky poslaná na e-mail užívateľa, poprípade prednastaviť si prahové hodnoty meraných údajov tak, aby pri ich prekročení bolo zaslané upozornenie. Ďalšou možnosťou pre zobrazenie dát je využitie jazyka Q&A, pomocou ktorého je možné vyhľadávať podobne ako pomocou jazyka SQL v databázach. [14]

Power BI disponuje viacerými možnosťami importu dát – napríklad databázy, excel, web, xml, json, ale aj napríklad python skript a mnoho ďalších. Cena základného softvéru je 10 dolárov pre 1 užívateľa.

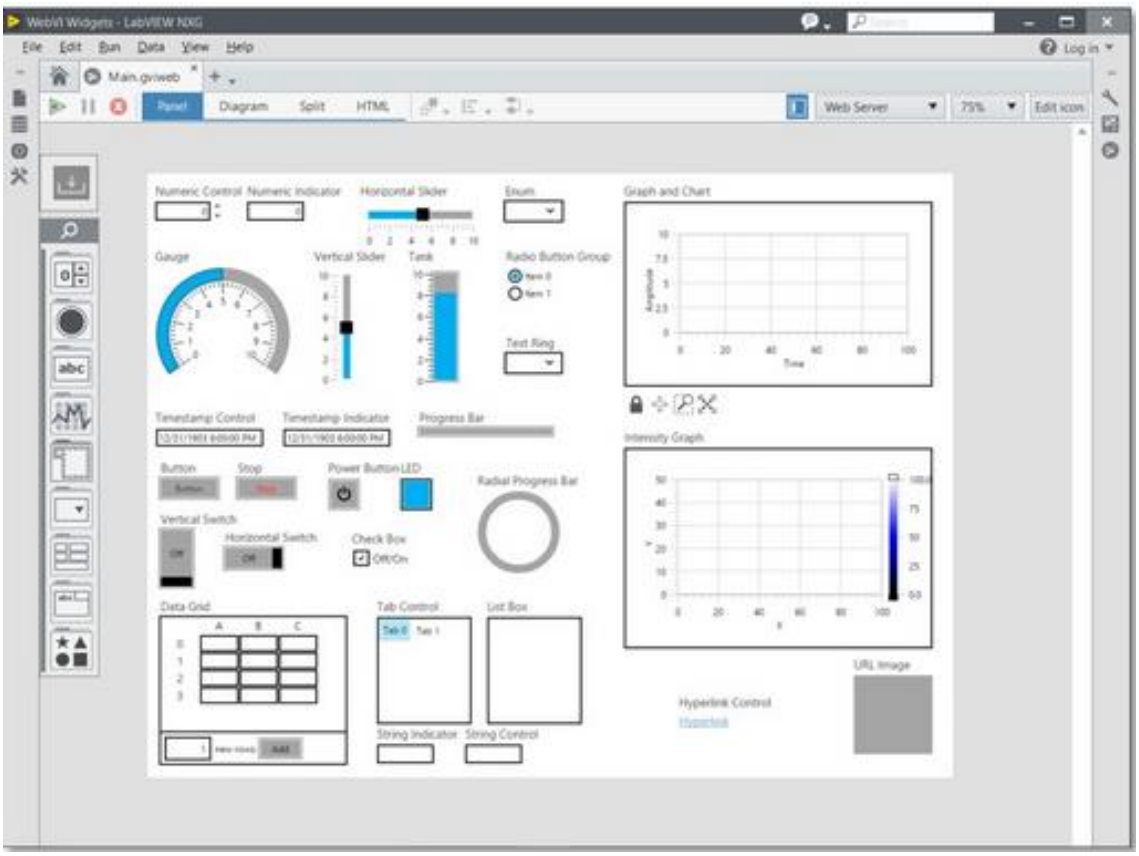



*obr. 11: Power BI [15]*

## LabVIEW Next Generation

LabView je programovacie prostredie, v ktorom je kód vytváraný formou blokového diagramu. Výstupy a vstupy do programu je možné zadávať v časti nazvanej „front panel“, do ktorého je možné pridávať indikátory rôznych premenných, listové zoznamy a mnoho ďalšieho.

Pre zjednodušenie tvorby vizualizácie a jej umiestenie na web vytvorili vývojári špeciálny webový modul. Ide o modul, ktorý dokáže pretransformovať vytvorený „front panel“ do webovej stránky, so zachovaním správnej funkcionality celého kódu. To zaručuje perfektnú interakciu medzi programovacím jazykom (vlastný jazyk v grafickej podobe) a žiadanou vizualizáciou. Výhodou je tvorba stránky bez potreby znalosti „front-end“ jazykov ako napr. Javascript a takisto množstvo „widgetov“, ktoré vizualizácií dodajú moderný vzhľad (obr. 12). Transformácia stránky prebieha takto – front panel je preložený do HTML+CSS kódu a blokový diagram je vyjadrený pomocou Javascriptu. To dodáva možnosť dodatočného upravovania stránky mimo LabVIEW, ako aj možnosť pridávania funkcionalít tretích strán. [16]




*obr. 12: LabVIEW NXG [16]*

## 2.5 Zhrnutie kapitoly

Pre zhodnotenie je použitá metóda viackriteriálneho rozhodovania, kde každé kritérium bude ohodnotené pre každý softvér v rozsahu 1-4 body (čím viac bodov, tým vhodnejší). Možnosť s najväčším súčtom bodov je najvhodnejšia. Obdobná forma hodnotenia bude použitá aj v ďalšej časti práce. Podrobné porovnanie obsahuje tab. 1.

*tab. 1: Rozhodovacia tabuľka pre výber variantu tvorby užívateľského rozhrania*

| **Kritéria** | **Glade** | **Designer** | **Vue.js** | **PowerBI** | **LabVIEW** |
|---|---|---|---|---|---|
| **Integrácia s jazykom** (zložitosť prenosu front-end a back-end) | **3** | 4 | **1** | 2 | 4 |
| **Vlastné riešenie** (voľnosť pri tvorení GUI) | **2** | 3 | **4** | 1 | 3 |
| **Knižnice** (komunita) | **1** | 2 | **4** | 3 | 3 |
| **Zložitosť** (čas strávený tvorbou) | **3** | 2 | **1** | 4 | 3 |
| **Funkčnosť** (splnenie princípu SCADA) | **1** | 3 | **4** | 2 | 4 |
| **Cena** (náklady na softvér) | **4** | 1 | **4** | 3 | 1 |
| **Vzhľad** (výsledný grafický efekt) | **1** | 2 | **4** | 3 | 3 |
| **Súčet** | **15** | **17** | **22** | **18** | **21** |

Na základe viackriteriálneho rozhodovania je zrejme, že najvhodnejším kandidátom na tvorbu GUI je variant Vue.js, a teda rozhranie bude vytvorené ako webová stránka pomocou kombinácie jazykov html a Javascript. Toto riešenie je dosiahlo najvyššiu bodovú hodnotu (4) v 5 kritériách, okrem zložitosti a integrácie s programovacím jazykom. Znamená to, že tvorba (hlavne prvého) GUI bude časovo náročnejšia oproti konkurentom a bude nutné vyriešiť

problémy pre správnu spoluprácu s back-end, ktorý musí byť rozdielny, pretože spomínané jazyky nie sú vhodné pre dátovú analýzu, a teda dôjde k prepojeniu s iným programovacím jazykom, ktorý bude zodpovedný za získanie a spracovanie informácií.

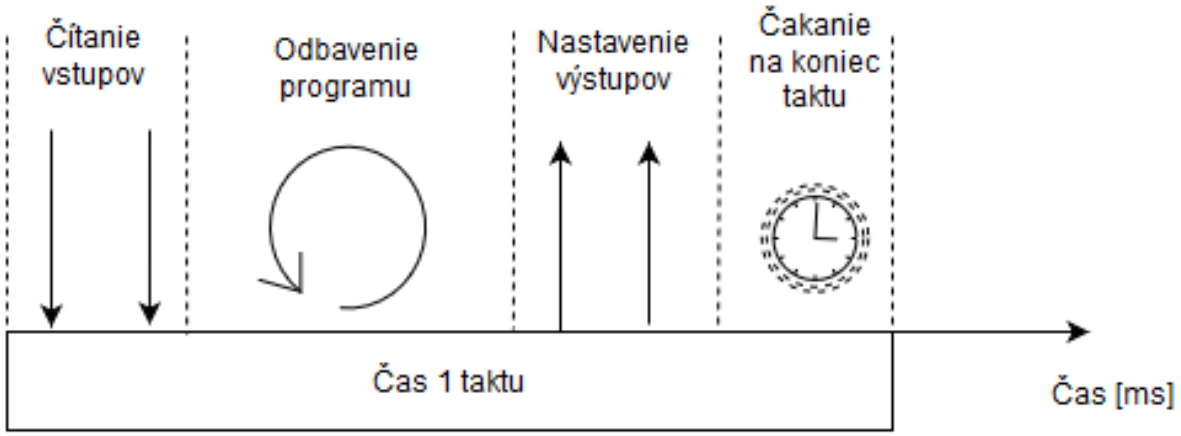
Takéto riešenie zabezpečí takmer neobmedzené možnosti pri predspracovaní dát a zároveň umožní aby grafické rozhranie malo moderný vzhľad a jeho funkčnosť bude zaistená na rôznych platformách/zariadeniach.

# 3 Programovateľný logický automat (PLC)

**P**rogrammable **l**ogic **c**ontroller**,** v preklade programovateľný logický automat, je zariadenie, ktoré v podniku nahradilo funkciu desiatok relé spínačov.

*„Programovatelné automaty (Programmable Logic Controlers PLC) se staly nejvýznamnějším řídicím prostředkem pro řízení technologických procesů, výrobních linek a strojů již během první poloviny 80. let.“* [17 s. 18]

Hlavný dôvodom je neustále sa zvyšujúca požiadavka na bezpečnosť a zber informácií, a teda rastúci počet snímačov, senzorov, kontroliek a tlačidiel, ktoré treba obsluhovať. Tieto zaradenia bolo čoraz zložitejšie udržať v bezporuchovom stave. Navyše pri takomto riešení bolo často nutné, aby obsluha kontrolovala stav a vykonávala určité akcie. U zložitejších funkčných celkov vznikali problémy, kedy nebolo možné napevno nastaviť danú reakciu, bolo nutné sa rozhodovať, podľa rôznych indikátorov. S postupným rozmachom automatizácie túto úlohu v zložitejších aplikáciách prebral počítač – PLC. Je to zariadenie, ktoré je schopné prijímať vstupy (hodnoty zo senzorov, čidiel,..) vo forme napätia/prúdu a na základe programu (vytvára programátor) sú schopné nastavovať požadované výstupy (reakcie – zapnúť motor, rozsvietiť kontrolku). Tento cyklus sa v určitom časovom intervale (takte) cyklicky opakuje. Priebeh jedného cyklu je vidieť na obr. 13. [18]




*obr. 13: Cyklus PLC*

Takt PLC začína načítaním vstupov – údaje senzorov. Ďalším krokom je exekúcia užívateľského programu. Následne prebieha nastavovanie hodnôt výstupov (zapnutie motora, ..). Nakoniec

programovateľný automat „čaká“ na ďalší takt. Celý tento postup sa potom cyklicky opakuje. V praxi nie je doba cyklu pevná, závisí na zložitosti programu. Trvanie taktu predlžuje napríklad aj dĺžka odozvy analógových vstupov. Možné využitie PLC:

- **Zber dát**
- Kontrola procesu
- Riadenie postupných krokov obsluhy – v prepojení s HMI
- Riadenie celého výrobného procesu – bez obsluhy

Z pohľadu SCADA systému je dôležité práve využitie PLC na zber dát, pretože v každom cykle je možné dostať práve 1 hodnotu všetkých vstupov a výstupov. Tieto hodnoty sú uložené vo forme **tagov** – pojem využívaný pre pomenovanie týchto premenných. Každý vstup alebo výstup je teda reprezentovaný jedným tagom. Tieto hodnoty môžeme ukladať a vytvoriť databázu, ktorá môžu slúžiť pre už spomínanú vizualizáciu, prípadne kontrolu procesu.


*obr. 14: Siemens PLC [19]*

Programovateľný automat (fyzický vzhľad na obr. 14) sa takisto stará o bezpečnosť riadeného zariadenia. Pre tento účel sa využívajú napríklad:

- Odblokovacie podmienky (Sprístupnenie výstupov až po splnení určitej podmienky)
- Určenie prípustného rozmedzia hodnôt
- Pripojený kontrolný hardvér (Bezpečnostný skener, dvojručné ovládanie)
- Dvojnásobné odbavenie programu a porovnanie výsledkov
- Kontrola toku materiálu (Snímač prítomnosti kusu)

## 3.1 Možnosti komunikácie s programovateľným automatom

Nasledujúca podkapitola bude pojednávať o možnostiach komunikácie medzi PC a PLC. Cieľom je vybrať čo najvhodnejšiu, s ohľadom na charakter. Výber zohľadňuje komunikáciu prostredníctvom fyzickej vrstvy ethernetu.

### 3.1.1 Komunikačný protokol MQTT

Protokol postavený nad TCP/IP. Je primárne určený pre aplikácie IoT (**i**nternet **o**f **t**hing), kde mnoho zariadení je združených pomocou centrálneho systému, ktorý zbierané dáta ďalej triedi a sprístupňuje vzdialeným užívateľom. Komunikácia neprebieha štýlom server – klient, ale dáta sú neustále odosielané a vráti sa iba potvrdenie o ich doručení, ide teda o princíp publish - subscribe. Tento typ komunikácie neumožňuje dispečerské riadenie procesu. [20]

### 3.1.2 Komunikačný protokol OPC UA

Komunikačný protokol, ktorý je vhodný nielen na získavanie dát, ale aj na komunikáciu medzi zariadeniami, ako napr. HMI a PLC a teda je vhodný pre použitie SCADA. Pre jeho použitie je nutné, aby zariadenia disponovali prostriedkom na tvorbu opc ua serveru. Veľká časť zariadení má tento prostriedok už v základnej výbave, prípadne ich je možné vybaviť pomocou takéhoto modulu. Tento štandard je na rozdiel od jeho predchodcu (opc) univerzálny, a teda umožňuje prenos procesných, historických a alarmových dát pomocou jedného protokolu. Adresný priestor je vytvorený pomocou uzlov, na ktoré sa je možné pripojiť pomocou programovacieho jazyka, napríklad s využitím opc-ua knižníc. Táto komunikácia bola navrhnutá s ohľadom na bezpečnosť – umožňuje šifrovaný prenos do MES/ERP systému. [21]

*„Cieľom UA je zabezpečiť rozhranie medzi serverom (dodávateľ informácií) a klientom (príjemca). Pre výmenu dát medzi serverom a klientom využívajú transportné mechanizmy. Tento základný koncept umožňuje klientovi pristupovať k najtriviálnejším dátam bez nutnosti chápania celého informačného modelu tvoreného komplexným systémom.“* [22 s. 10, preložené z anglického jazyka]

### 3.1.3 EtherCAT

Master-slave komunikačný protokol, ktorý funguje pomocou siete ethernet. Je schopný pracovať v real-time vďaka cyklom na úrovni milisekúnd. Táto forma komunikácie je vhodná skôr ako doplnok ku komunikácii OPC-UA, kde EtherCAT je využiteľný v horizontálnej integrácii medzi riadiacimi systémami strojov a komunikácia OPC-UA je vhodnejšia na vertikálnu integráciu – medzi real-time a vyššou organizačnou vrstvou (ERP). [23]

### 3.1.4 PROFINET

Komunikačný protokol fungujúci na princípe klient-server. Jeho realizácia prebieha pomocou priemyselného ethernetu. Je vhodný na real-time zber dát, avšak priame ovplyvňovanie procesu cez profinet nie je možné. Jeho výhodou je krátky čas cyklu – až 1 milisekunda. To ho predurčuje na použitie v časovo náročných aplikáciách, napríklad pri riadení pohonov. [24]

## 3.2 Zhrnutie kapitoly

Účelom tejto kapitoly bolo nájsť najvhodnejší komunikačný protokol pre aplikáciu s využitím v systéme SCADA. Najvhodnejším variantom sa zdá byť využitie **OPC-UA** servera, ktorým disponuje väčšina moderných zariadení v podniku a jeho funkčnosť je plne vyhovujúca potrebám zamýšľaného SCADA systému. Veľkou výhodou je jednoduché prepojenie pomocou programovacích jazykov a teda môže byť plne využitý ich potenciál na spracovanie dát.

# 4 Programovacie jazyky

Existuje nepreberné množstvo programovacích jazykov, avšak každý má svoje špecifikácie a každý sa hodí na niečo iné. V zásade rozdeľujeme jazyky na 2 skupiny:

- **Back-end** – vhodné na dátovú analýzu, matematické operácie, čítanie dát z opc-ua servera, ukladanie do databázy (vidí programátor)
- **Front-end** – vhodné na zobrazovanie (vidí koncový užívateľ)

## 4.1 Front-end jazyky

Budú použité na tvorbu vizuálnej stránky. Úlohou jazykov zmienených v tejto kapitole je zobraziť požadované informácie koncovému užívateľovi, v peknej, grafickej podobe.

### 4.1.1 Hypertextový značkový jazyk (Html)

HTML je programovací jazyk, určený primárne pre tvorbu webových stránok, ktoré sú prepojené pomocou hypertextových odkazov. Teda je vhodný pre tvorbu vizualizácie, ktorá by nebola závislá na zariadení, bola by umiestnená na webovej stránke, ktorú by som po príslušnej autorizácií dokázal prezrieť ktokoľvek a odkiaľkoľvek. Pre splnenie tejto požiadavky by bolo vhodné, aby bol kód stránky písaný v špecifikácii jazyka nazvanej HTML 5, ktorá umožňuje multi-platformovú podporu. [25]

### 4.1.2 Kaskádové štýly (CSS)

Jazyk, ktorý je vhodný pre úpravu vzhľadu stránky. Pomocou tohto jazyka nie je možné vytvoriť webovú stránku, používa sa v kombinácií napríklad s html. Súborom s príponou .css je možné napríklad meniť veľkosť, farbu písma alebo nastavovať okraje a mnoho ďalšieho. Využíva sa hlavne pri tvorbe webu s viacerými stránkami, pričom chceme zachovať jednotný vzhľad, na čo môžeme využiť jeden spoločný css súbor.

### 4.1.3 Javascript

Moderný programovací jazyk, ktorý je primárne určený pre tvorbu stránok - vyzerá dobre. Vďaka systému Node.js je možné použiť aj na back-end. Vďaka masívnemu využívaniu ponúka mnoho knihovní, ktoré umožňujú používať interaktívne animácie, zaujímavé grafy, diagramy to zaručuje moderný vzhľad stránok.

## 4.2 Back-end jazyky

Počítač nedokáže logicky uvažovať. Logiku, a teda základ každého programu, musí vytvoriť človek – programátor. Avšak programátor nerozumie jazyku počítača a naopak počítač nerozumie ľudskému jazyku. Preto vznikajú nové jazyky, ktoré sú na pomedzí a teda rozumie im človek a po preložení aj počítač. Úlohou programu napísaného v tomto jazyku bude čítať dáta z opc-ua servera zariadenia PLC, ich spracovanie a uloženie do databázy. Následne musia byť dáta poskytnuté pre vizualizáciu. Ale ktorý jazyk vybrať?

Dôležitou vlastnosťou jazyka je jeho forma:

- Interpretovaný
- Kompilovaný
- JIT

**Interpretovaný** jazyk je písaný vo forme, ktorej počítač nerozumie. Na preklad kódu do jazyka, ktorému rozumie počítač sa používa program – **interprét**. Výhodou tohoto riešenia je ľahšie upravovanie programu, hľadanie chýb a dobrá prenositeľnosť na inú platformu (avšak na tejto platformu musí existovať interprét pre príslušný jazyk). Nevýhodou je pomalosť prameniaca z potrebného prekladu, kedy je pre spustenie programu nutné, aby počítač mal k dispozícií zdrojový kód a aj potrebný interprét. [26]

**Kompilovaný** jazyk znamená, že kód napísaný v tomto jazyku sa musí najprv preložiť pomocou **kompilátoru** (kód preloží a optimalizuje) a až potom je zdrojový kód poskytnutý počítaču, ktorý ho môže okamžite spustiť, bude mu rozumieť. Nepotrebuje teda žiadny dodatočný program. Nevýhodou je horšia prenositeľnosť programu medzi platformami, poprípade dlhší čas kompilácie. [26]

**Jit** jazyk je „mixom“ kompilovaného a interpretovaného jazyka, kedy sú napísane programy skompilované tesne pred ich spustením (optimalizácia je menšia oproti kompilovanému jazyku). [26]

V súčasnosti je ale možné mnoho jazykov kompilovať aj interpretovať, to znamená že ide skôr o konkrétnu formu použitia než o vlastnosť jazyka.

Ďalším atribútom programovacieho jazyka je jeho úroveň – podobnosť s jazykom počítača. Čím nižšia úroveň, tým je podobnejší tomu strojovému. Takýto kód je horšie pochopiteľný ľudskou mysľou, jeho účelom je čo najvyššia rýchlosť a stabilita. Takýto jazyk sa používa zásadne skompilovaný, aby nedochádzalo k spomaľovaniu. Využíva sa napríklad na ovládače (drivery) alebo programy, ktoré úzko spolupracujú s hardvérom. Na druhú stranu jazyky s vyššou úrovňou sú ľahšie na učenie a takisto tvorba samotného programu je rýchlejšia [26]

Ešte dôležitejší atribút než level jazyka je možnosť práce s premennými: [26]

- **Silný/slabý** – možnosť jednoduchej konverzie medzi rôznymi typmi premenných, napríklad „integer“ na „string“ a podobne

- **Zjavný/odvodený** – poukazuje na to, či je striktne potrebná deklarácia typu premennej pred použitím alebo či je typ premennej odvodený z kontextu

- **Statický/dynamický** – u statického prebieha kontrola typu premenných iba raz - pred spustením programu, u dynamického ide o run-time kontrolu

- **Bezpečný/nebezpečný** – bezpečnosť jazyka spočíva v nemožnosti použitia operácií/konverzií na premenných, u ktorých by mohlo dôjsť k chybe/nedefinovanému správaniu, u nebezpečného jazyka sa o to musí postarať programátor

### 4.2.1 Jazyk C, C++ & C#

„Céčko“ je jeden zo základných programovacích jazykov, v minulosti spätý predovšetkým s operačným systémom UNIX. Dnes beží na rôznych platformách a mnoho interpreterov pre iné jazyky, je napísaných práve pomocou tohto programovacieho jazyka. Radí sa medzi nízkoúrovňové jazyky. Rozšírenie o objektovo-orientované programovanie prišlo v podobe C++. Ďalším variantom, ktorá vznikla je C#. Ide o vysokoúrovňový jazyk, ktorý je jednoduchší na učenie a práca s ním môže byť príjemnejšia.

Všetky tieto jazyky majú širokú komunitu a nespočetné množstvo knižníc. Pre všetky zmieňované jazyky je dostupná opcua knižnica, ktorá je vhodná pre pripojenie na opc-ua server skúmaného PLC, čo môže ušetriť množstvo práce. [27]

### 4.2.2 Python

Python je moderný vysokoúrovňový jazyk, ktorý sa stal obľúbeným hlavne pre jeho jednoduchosť. Hlavnou výhodou tohto jazyka je jeho čitateľnosť a zrozumiteľnosť. Naučenie sa základov tohto jazyka trvá naozaj krátku dobu. Na druhej strane je nutný interpret nakoľko ide o interpretovaný jazyk – najpoužívanejšie sú Microsoft Visual Studio alebo PyCharm. V porovnaní s ostatnými jazykmi je syntax značne jednoduchšia – rovnaký kód je omnoho kratší. Napriek tomu je Python veľmi mocný nástroj, hlavne v oblasti dátovej analýzy. Ďalej je hojne využívaný na webový (na strane servera) a softvérový vývoj. Má širokú komunitu užívateľov a obrovské množstvo knihovní (medzi nimi aj knihovne na prácu s opc-ua serverom). Jedná sa o momentálne najpoužívanejší programovací jazyk vôbec. [28]

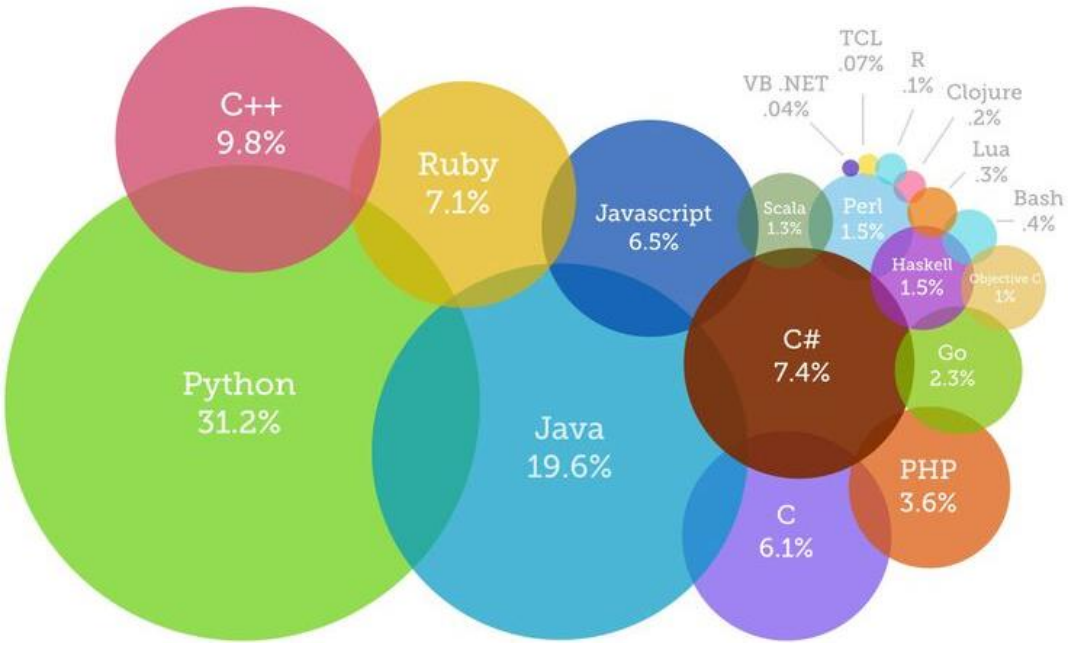
### 4.2.3 Java

Java je vysoko úrovňový jazyk, avšak zložitejší než Python. Hlavné využitie sú mobilné aplikácie a serverový back-end. Tento jazyk je interpretovaný, takisto ako Python. Jeho vykonávanie je teda pomalšie a nie je vhodné na výpočetne náročné aplikácie a hry. Výhodou tohoto jazyka je mohutná komunita a mnoho knižníc (aj pre opc-ua), nakoľko sa jedná o jeden z najpoužívanejších jazykov. [29]

### 4.2.4 Hypertextový preprocesor (PHP)

Programovací jazyk PHP je vhodný na tvorbu webových aplikácií. Má schopnosť generovať html stránku na strane serveru (na rozdiel od JavaScriptu, kde toto prebieha u klienta). Pomocou tohto jazyka je počítať matematiku, ale aj vytvárať grafy a mnoho ďalšieho. Nevýhodou je, že pripojenie sa na opc-ua server je oproti ostatným zmieneným jazykom zložitejšie.

Na obr. 15 sú zobrazené najrozšírenejšie jazyky používané v súčasnosti. Z grafu je vidieť, že asi tretina používateľov sa sústredí na Python, ďalej asi pätina preferuje Javu a desatina jazyk C++. Ostatné jazyky sú v nižšom zastúpení.




*obr. 15: Programovacie jazyky [30]*

### 4.2.5 Zhrnutie kapitoly

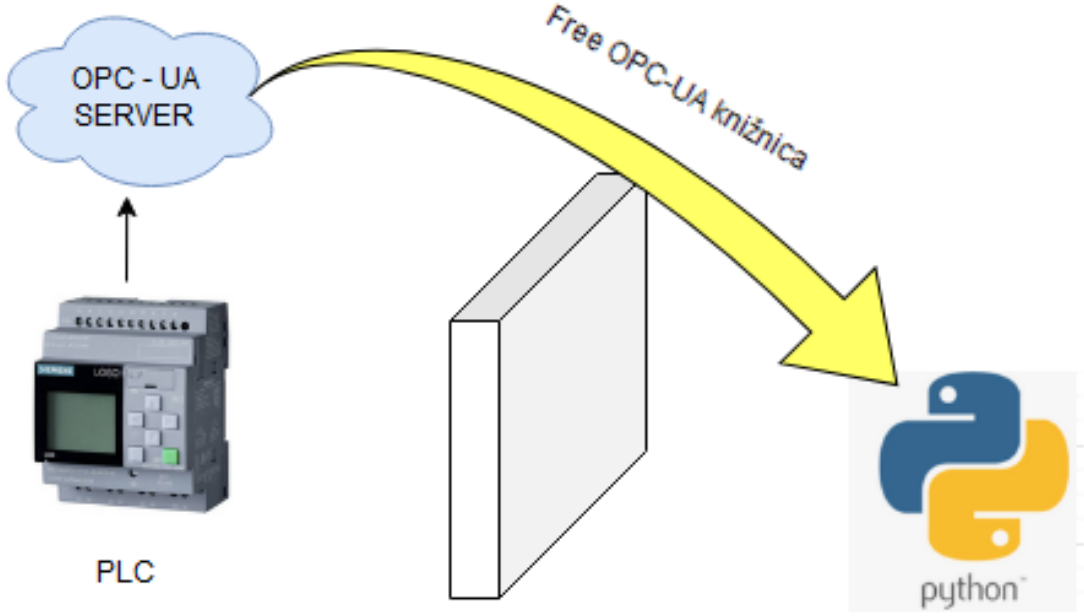
*tab. 2: Rozhodovacia tabuľka pre výber variantu programovacieho jazyka*

| *Kritéria* | *C++* | *Python* | *Java* | *PHP* |
|---|---|---|---|---|
| *Jednoduchosť* | 1 | **4** | 2 | **3** |
| *Vhodnosť použitia* | 3 | **4** | 3 | **2** |
| *Opc-ua knižnica* | 4 | **4** | 4 | **1** |
| *Komunita* | 2 | **4** | 3 | **1** |
| *Rýchlosť* | 4 | **1** | 3 | **2** |
| *Súčet* | *14* | *17* | *15* | *9* |

Z porovnávacej tabuľky (tab. 2) je najlepšou voľbou programovací jazyk Python, hlavne kvôli jeho jednoduchosti a čitateľnosti, rozsiahlej komunite a vhodnosti pre dané použitie – dátovú analýzu . Bohužiaľ nároky tohto jazyku na procesor sú horšie oproti konkurentom, avšak vzhľadom k charakteru použitia to nebude prekážka.

## 4.3 OPC server & Python

Základom pre zber dát tvorí prepojenie PLC a Pythonu. Toto prepojenie bude vytvorené pomocou opc-ua serveru, ktorý je v štandardnej výbave PLC a je nutné ho iba softvérovo aktivovať alebo je možné dokúpiť modul, ktorý túto tvorbu opc servera umožňuje. Vytvorenie vlastného kódu pre takúto komunikáciu, by mohlo byť časovo náročné. Vhodným riešením je preto využitie open-source Free OPC-UA Library, ktorá bola vytvorená užívateľmi github-u (https://github.com/FreeOpcUa). Najpoužívanejším repozitárom z tejto knižnice je python-opcua, ktorého obsahom je modul s názvom opcua, ktorú je možné importovať do python. Táto knižnica umožňuje pomocou súčasti Client vytvorenie komunikácie so serverom a pomocou jediného príkazu, je možné napríklad čítať hodnoty uzlov (nodes), ktoré sú vytvorené na serveri, vytvárať nové uzly, mazať ich a mnoho ďalšieho.



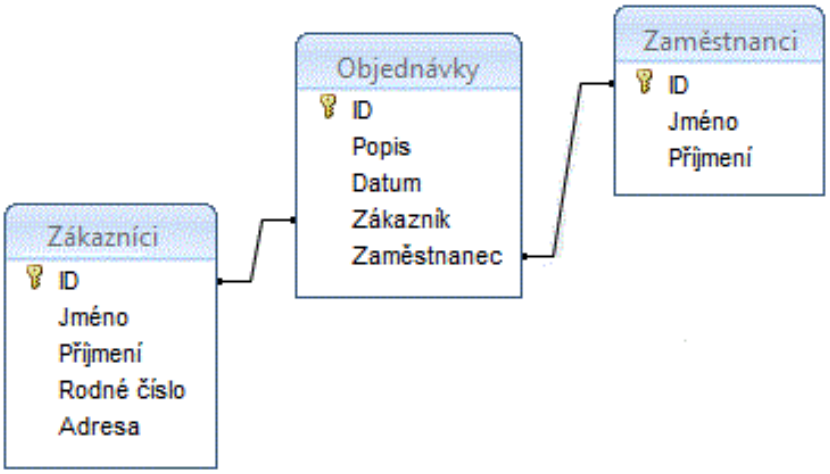

*obr. 16: Schéma komunikácie*

# 5 Databáza

Databáza je označenie systému na uchovávanie dát, v štruktúrovanom, organizovanom formáte. Takéto štruktúry sú vytvárané za rôznymi účelmi, či už ide o ukladanie údajov o zamestnancoch, skladových zásobách alebo vyrobených kusoch. V súvislosti so SCADA systémom je zber dát vykonávaný za účelom dlhodobých analýz, sledovania vývoja alebo napríklad hľadania príčiny zmätočnej výroby. [31]

Najjednoduchšími typmi databázy je **hierarchický** a **sieťový** model. Zjednodušene ide o rozvetvenú stromovú štruktúru, kde každý záznam je uzlom v tejto štruktúre. V prípade sieťového model sú záznamy doplnené o set atribútov (doplnkových informácií) – pre názornosť napríklad záznam ŠTUDENT, by mohol mať atribúty TITUL a ŠTÚDIJNÝ ODBOR.

O niečo modernejším a zložitejším typom sú databázy **relačné**. Je možné ich chápať ako súbor logicky prepojených tabuliek, na základe určitých vzťahov, ktoré je možné s výhodou využívať. Každá tabuľka sa nazýva entita a obsahuje určité atribúty (stĺpce), ktoré sú presne definované dátovým typom informácie, napríklad číslo alebo textový reťazec. Riadok v tabuľke tvorí samostatný záznam. [32]




*obr. 17: Relačná databáza [32]*

Najmodernejším prístupom sú objektovo orientované databázy. Informácie v takejto databáze sú namiesto tabuliek reprezentované pomocou objektov, čo je využiteľné hlavne vo svete programovania.

## 5.1 Jazyk SQL

Najčastejším spôsobom na prácu s databázou je SQL. Tento jazyk obsahuje niekoľko základných príkazov, pomocou ktorých je možné vyhľadať požadované dáta, vytvárať a mazať záznamy alebo aj vytvárať nové tabuľky.

*„SQL je dotazovací jazyk, takže přes propojenou aplikaci se serveru odevzdá dotaz a databázový server na něj odpoví, obvykle tím, že vygeneruje nejakou množinu výstupních údajů. Tento princip komunikace s databázovým serverem je velmi jednoduchý a efektivní. Samozřejmě ale jen z hlediska uživatele. Jazyk SQL totiž připomíná klasický přirozený jazyk (samozřejmě anglický), má však přesně definovaná syntaktická a lexikální pravidla.“* [33]

Najbežnejšie príkazy: [34]

- SELECT
  - príkaz pre filtráciu riadkov na základe určitého parametru
- WHERE
  - slúži pre zadanie parametru pre filtráciu
- ORDER BY
  - pomocou tohto príkazu je možné zoradiť záznamy podľa zvoleného atribútu
- JOIN
  - slúži na spájanie tabuliek
- INSERT
  - vloženie nového záznamu
- DELETE
  - zmazanie záznamu z tabuľky

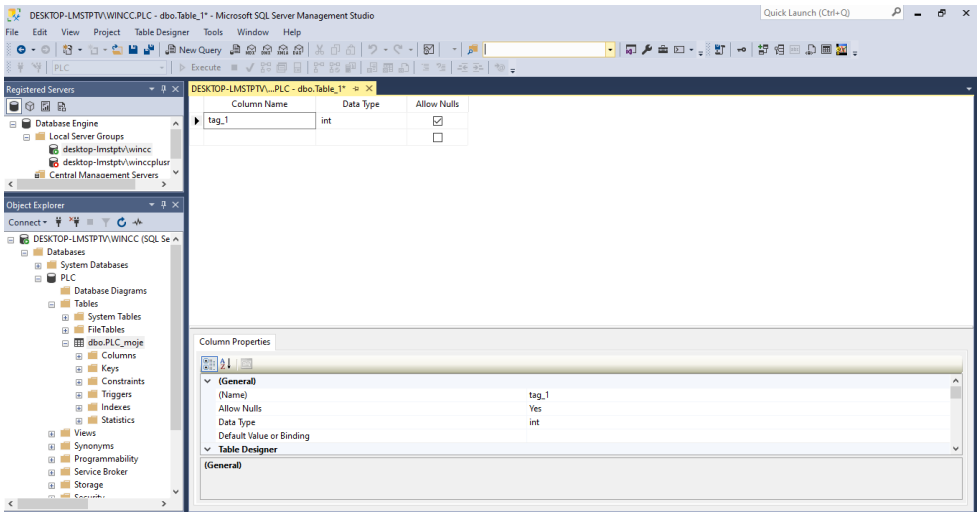
```
CREATE TABLE tabulka (
  sloupec NUMBER (2)
);
INSERT INTO tabulka (sloupec) VALUES (1);
INSERT INTO tabulka (sloupec) VALUES (2);
INSERT INTO tabulka (sloupec) VALUES (3);
```

*obr. 18: Príkaz SQL [35]*

## 5.2 Databáza & Python

Existuje mnoho možností ako pracovať s databázou v Python. Najjednoduchším riešením je využitie vstavanej knižnice s názvom **sqlite3**, pomocou ktorej je možné vytvoriť a spravovať databázu, ktorá bude uložená vo forme jednoduchého binárneho súboru. Potom je možné do tejto tabuľky vkladať dáta, vyhľadávať pomocou syntaxe známej z SQL. Nevýhodu je, že pre prácu s takto vytvorenou databázou musí mať každý užívateľ prístup k tomuto súboru, nainštalovaný python a schopnosť práce s ním. [36]

V praxi je bežné, že na spravovanie databáz sa využívajú špeciálne programy, napr. MySQL alebo SQL Server Management Studio. Aj pre prácu s touto formou databáze existuje knižnica, jej názov je **pyodbc**, ktorá však nie je súčasťou „povinnej výbavy" a teda je nutné ju dodatočne inštalovať. Výhodou toho riešenia je, že databáza je vytvorená v databázovom programe, čo znamená, že jej tvorba je príjemnejšia, zrozumiteľnejšia ako tvorba pomocou príkazov v prípade už spomínaného sqlite3. Takto pred vytvorenú databázu (tabuľku) je potom možné plniť pomocou hodnôt, ktoré bude python získavať z OPC-UA servera PLC. Jednoducho Python všetky získané hodnoty uloží do databázy, kde bude možné si tieto dáta prezerať, filtrovať a ďalej využívať. Takisto je samozrejme možný opačný smer spojenia – čítať dáta z databázy pomocou python-u. Takto získané dáta je možné jednoducho vypísať pomocou príkazu print, avšak v nie veľmi úhľadnej podobe. Je však možné podobu zobrazenia vylepšiť – na to slúži Data Frame Pandas. Je ale nutné ho inštalovať a importovať podobne ako pyodbc. [37]




*obr. 19: SQL Server Management Studio*

## 5.3 Zhrnutie kapitoly

Najvhodnejším spôsobom pre tvorbu databázy je forma SQL. S touto databázou je možné pracovať pomocou programovacieho jazyka python (knižnica pyodbc), čo je nevyhnutné pre správnu funkčnosť systému SCADA. Navyše v prostredí SQL sú schopné pracovať aj osoby, ktorým by takáto databáza mala slúžiť (technológ, financie) pomocou jazyka SQL.

# 6 Generátor dát

Pre vývoj SCADA systému je dôležité aj jeho testovanie, ktoré je z viacerých dôvodov nevhodné aplikovať priamo na reálnom stroji. Omnoho výhodnejším variantom je vytvorenie simulácie stroja, ktorý môže byť podobný, alebo v najlepšom prípade rovnaký ako stroj, ktorého stav budeme pomocou takéhoto systému monitorovať. Z toho dôvodu bolo vytvorené „digitálne dvojča“ lakovacej linky, ktorá sa nachádza v reálnom podniku a pre ktorú by tento systém mohol byť v budúcnosti nasadený.

Požadovaná simulácia sa skladá zo 4 základných prvkov:

- Zjednodušený 3D model výrobného systému
- Vizuálne rozpohybovanie linky
- Simulácia požadovaných hodnôt (na základe skutočného systému)
- Virtuálneho OPC serveru

## 6.1 Model výrobného systému

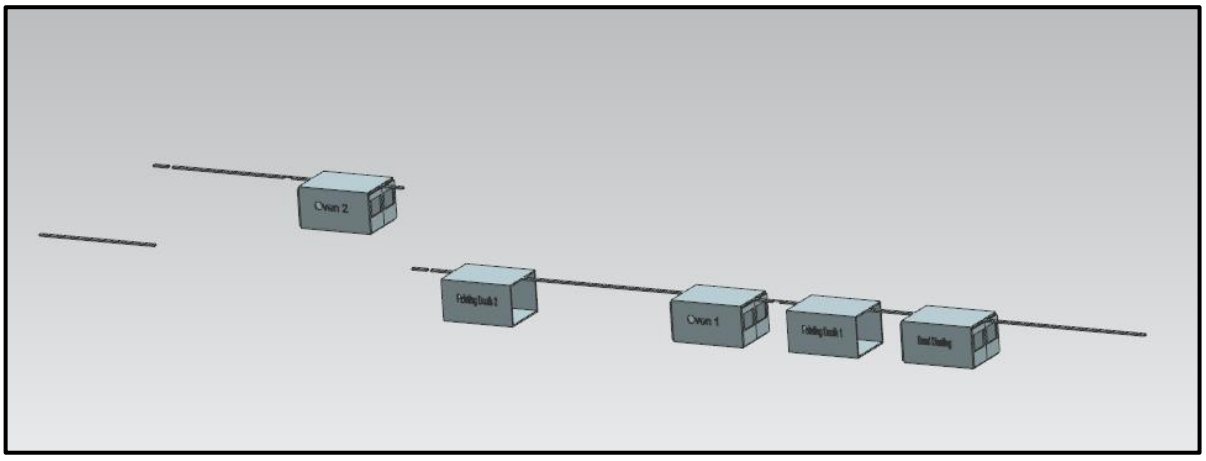
Pre vytvorenie zjednodušeného modelu systému bol použitý program Siemens NX 12.0. Lakovacia linka je vytvorená z niekoľkých základných komponentov:

- Boxy – procesné prvky (tryskacie zariadenie, vypaľovacia pec, lakovacia bunka)
- Dopravníky – technologické prvky (dráha lakovacieho vozíku)
- Výťahy – technologické prvky (doprava závesu medzi 1. a 2. poschodím)
- Lakovací záves (reprezentuje diel, ktorý sa pohybuje po lakovacej linke)

3D model kopíruje usporiadanie prvkov tak, ako je to aj na reálnom výrobnom systéme. Pre zjednodušenie boli zakrivené dráhy nahradené dráhami rovnými, čo ale nemá žiadny vplyv na vygenerované dáta. Ďalším zjednodušením je, že na lakovacej linke sa nachádza iba jeden vozík namiesto desiatok – dôsledkom je prehľadnejšia simulácia a jednoduchšie overenie funkčnosti, prípade kontrola, že vygenerované dáta sú správne.

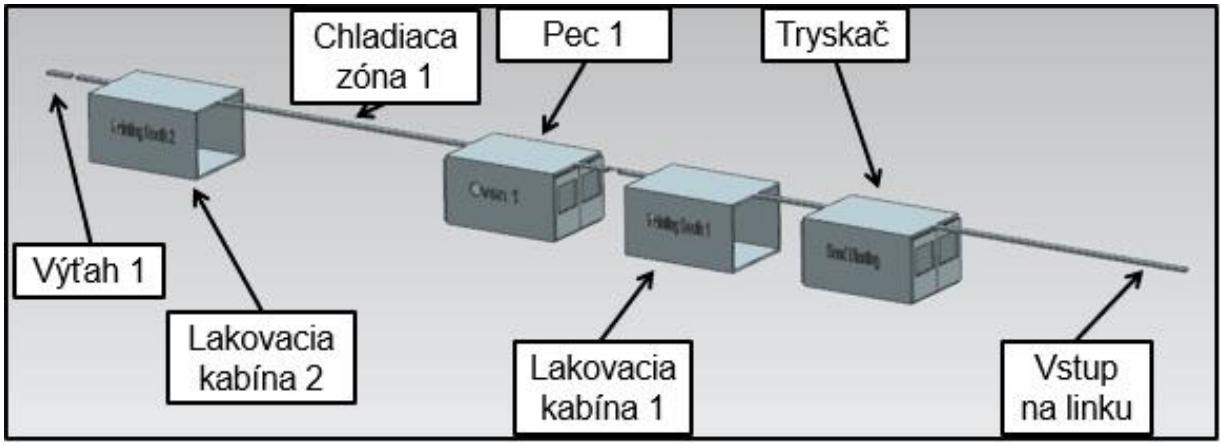
Pre vytvorenie modelu bola nutná obhliadka a taktiež boli použité firemné materiály podniku, kde sú pomocou tejto linky lakované zvarence, ktoré sú následne kompletované na ďalších montážnych linkách.

Na obr. 20 je zobrazená celá lakovacia linka, ktorá sa skladá z 2 poschodí. Dopravu medzi týmito poschodiami zabezpečujú 2 výťahy, ktoré umožňujú pohyb vo vertikálnom smere a majú schopnosť „postrčiť“ záves na nasledujúci dopravník.



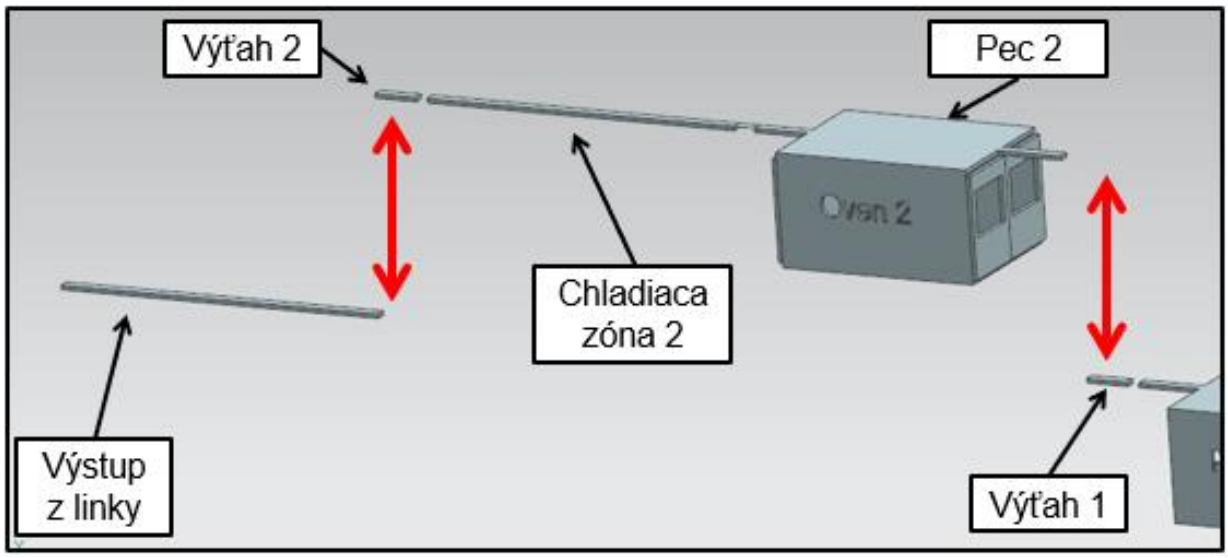

*obr. 20: Lakovacia linka – celá*

Na obr. 21 je zobrazené spodné poschodie lakovacej linky. Na pravej strane je vstup materiálu (operátori navesujú diely na prechádzajúci vozík). Na ľavej strane je výťah, ktorým diel putuje na vrchné poschodie. Chladiaca zóna je v realite opatrená ventilátormi, pre potreby generátoru zanedbané. Medzi jednotlivými procesnými prvkami sú dopravníky, ktoré zaisťujú pohyb závesu s dielom v horizontálnom smere. Rýchlosť týchto dopravníkov je možné meniť.




*obr. 21: Lakovacia linka - spodné poschodie*

Ďalší obr. 22 zobrazuje vrchné poschodie linky a taktiež koncovú časť spodného poschodia. Obdobne ako v prípade vstupu na linku je aj výstup materiálu z linky realizovaný operátormi, ktorí nalakované diely zvesujú zo závesu, ktorý putuje po dopravníku.




*obr. 22: Lakovacia linka - vrchné poschodie*

## 6.2 Simulácia – Virtuálne PLC

Ďalším krokom bolo spojazdnenie simulácie, ktorá vo svojej podstate reprezentuje beh programu na PLC. Pri voľbe simulačného programu boli zohľadňované viaceré parametre, ako napríklad možnosť vytvárania monitorovaných premenných – tagov (vizuálna simulácia nie je dostačujúca) alebo možnosť práce s protokolom opc-ua. Ako najvhodnejší nástroj pre tento účel bol vybraný program NI LabVIEW 2019 SP1. Táto verzia programu má implementované knižnice pre prácu s opc-ua protokolom, čo znamená, že splňuje najzásadnejšiu požiadavku na simulačný softvér použitý pre testovanie systému. Znamená to, že dáta, ktoré budú nasimulované v prostredí LabVIEW je možné propagovať na virtuálny opc server, odkiaľ je možné tieto dáta získať, napríklad pomocou programu UaExpert, ale aj pomocou programovacieho jazyka Python s využitím vhodnej knižnice.

### 6.2.1 Identifikácia hodnôt

Prvým krokom bola identifikácia hodnôt, ktoré sú prístupné na PLC reálneho výrobného systému. Táto informácia bola získaná z firemných materiálov spoločnosti. Zo všetkých údajov bolo potrebné definovať tie, ktoré majú vplyv na kvalitu, rýchlosť, funkčnosť výroby a teda ich stav je potrebné monitorovať. Formát údajov je zobrazený na obr. 23, avšak názvy boli z bezpečnostných dôvodov zmenené. S rešpektovaním dátových formátov, názvov a rozsahov hodnôt bola vytvorená simulácia, ktorej účelom je generovať dáta, ktoré majú nejaký fyzikálny zmysel. Účelom je vytvoriť logicky sa meniaci súbor dát, na ktorom je možné testovať komunikáciu a zobrazovať stav takejto simulácie.

**PaintShop**

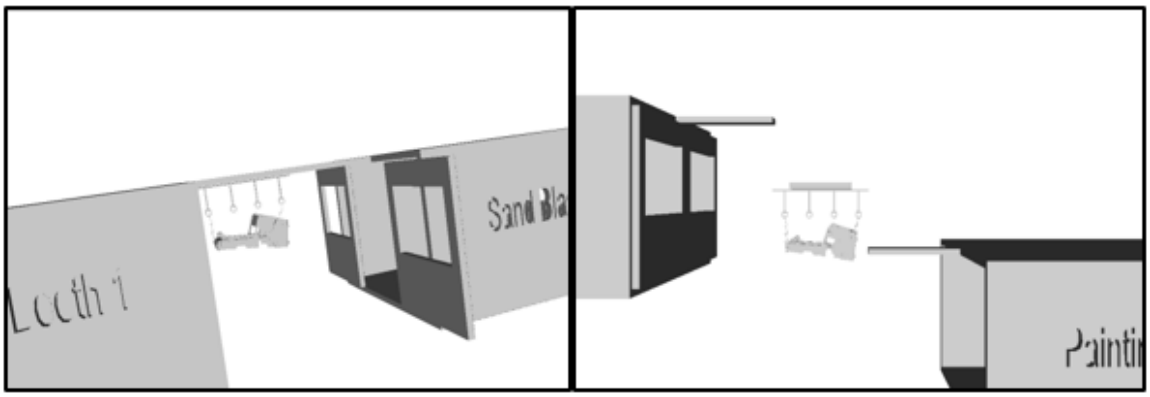
OPC UA Interface specification

| ID | OPC tag name | Tag name | Data type | Description |
|---|---|---|---|---|
| 1 | ns=2;s=opc.K1 | K1 | Bool | Drive status bit - moving |
| 2 | ns=2;s=opc.K2 | K2 | Int16 | Drive actual load - percentage - value * 0,1 |
| 3 | ns=2;s=opc.K3 | K3 | Int16 | Drive actual speed in m/min - value * 0,001 |
| 4 | ns=2;s=opc.K4 | K4 | Bool | Drive status bit - moving |
| 5 | ns=2;s=opc.K5 | K5 | Int16 | Drive actual load - percentage - value * 0,1 |

*obr. 23: Dostupné hodnoty PLC [zdroj: firemné materiály]*

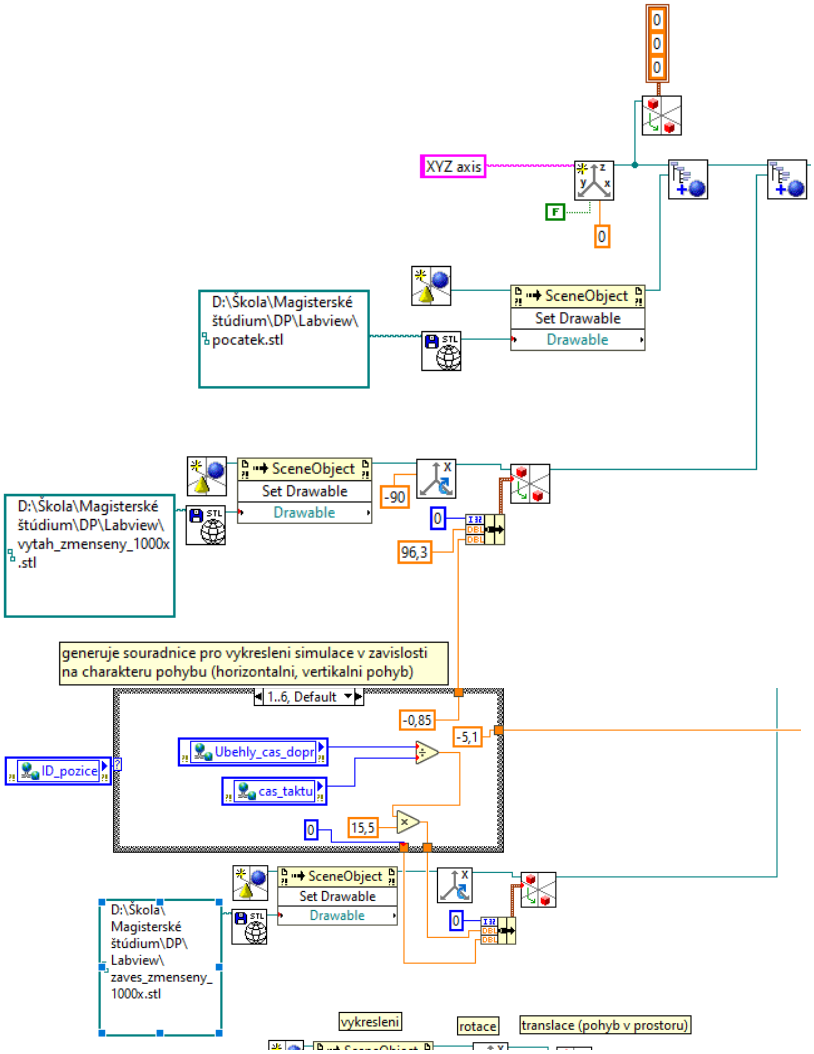
### 6.2.2 Rozpohybovanie linky

Pre prácu s 3D modelmi v prostredí LabVIEW je vhodné použiť formát STL. Takto formátované geometrie je potom možno pomocou vstavaných funkcií jednoducho načítať a zobraziť. Vzhľad takéhoto modelu je na obr. 24.



*obr. 24: 3D model v prostredí LabVIEW*

Dojem pohybujúcej sa linky je vytvorený pomocou zmeny súradníc polohy jednotlivých komponentov v čase. Všetky prvky majú na začiatku definované polohu, ktorú menia vzhľadom na ubehnutý čas, pričom sú zohľadnené aj prvky na ovládacom panely simulácie (zapnutie/vypnutie dopravníku, rýchlosť dopravníku...). V prostredí LabVIEW je kód písaný primárne pomocou blokových schém, podobne ako je tomu v programe Matlab Simulink. Príklad takejto schémy, ktorá slúži pre umiestenie prvku do priestoru v definovanom súradnicovom systéme je vidieť na obr. 25. Každý takýto program obsahuje aj druhú časť, tzv. front panel, ktorý obsahuje ovládacie a zobrazovacie prvky tak, aby bolo možné sledovať chod programu, hodnoty premenných, grafické výstupy, ale aj priamo zasahovať do chodu programu pomocou rôznych prepínačov a tlačidiel.




*obr. 25: LabVIEW – časť programu pre pohyb vybraného prvku*

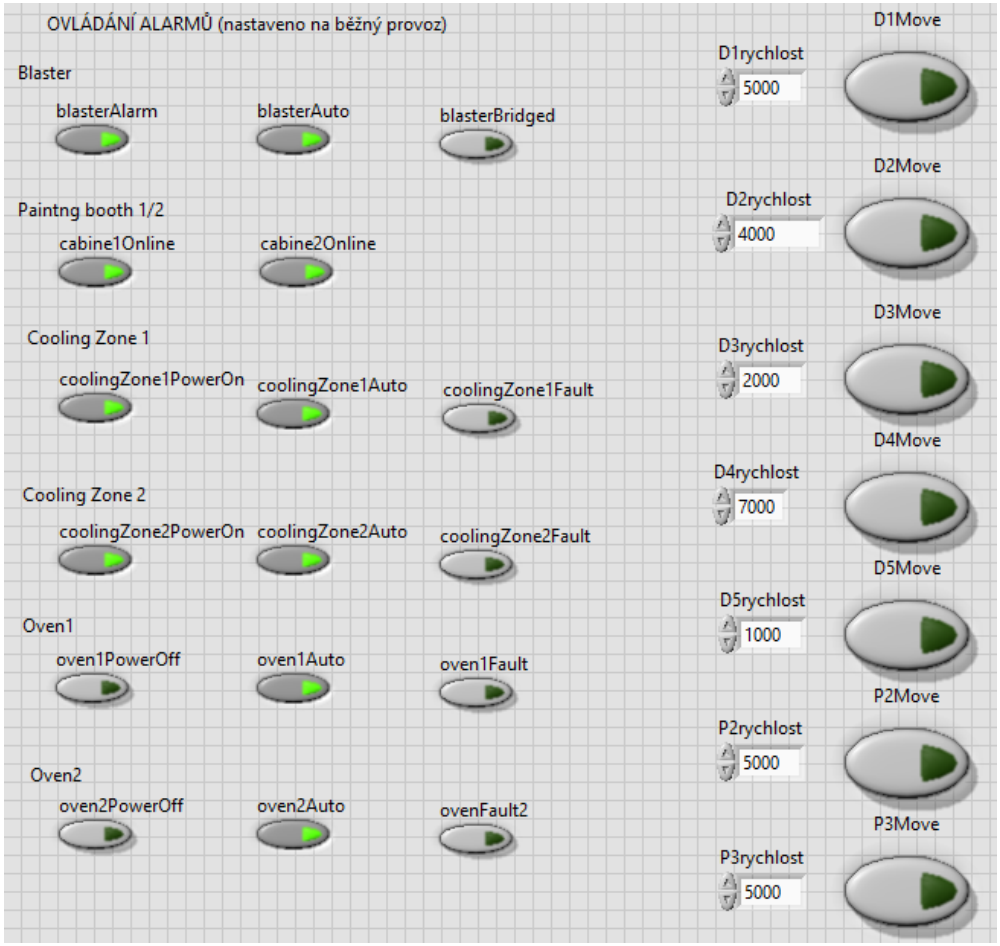
Na predchádzajúcom diagrame je vidieť tvorbu základného súradného systému v priestore [0,0,0], do ktorého sú následne pridávané ďalšie prvky vo forme STL súborov, ktoré sú najprv správne natočené a následne posunuté v súradnicovom systéme. Nepohyblivé prvky majú hodnoty posunutí vo všetkých osiach dané konštantou. U pohyblivých častí (lakovací záves, výťahy) sa hodnoty v niektorých osiach môžu meniť, v závislosti na smere požadovaného pohybu. Pohyb lakovacieho závesu je závislý na stave dopravníku, na ktorom sa tento lakovací záves nachádza – ak je daný dopravník zapnutý, diel sa pohybuje rýchlosťou, ktorá je nastavená na ovládacom panely v hlavnom simulačnom okne. 3D model každého komponentu použitého v simulácii musí byť takýmto spôsobom zavedený do programu, pričom na front panely simulácie je potom možné sledovať grafický výstup – vizuálna simulácia procesu.

## 6.3 Simulácia požadovaných hodnôt

Zo všetkých hodnôt, ktoré je možné z PLC vyčítať, bolo nutné vybrať tie, ktoré sú pre nás z pohľadu monitorovania zaujímavé. Boli vybrané:

- Rýchlosti dopravníkov
- Zapnutie/vypnutie dopravníkov
- Zaťaženie dopravníkov
- Teploty v peciach
- Zapnutie/vypnutie procesných prvkov (tryskač, pece, lakovacie bunky)
- Alarmy

Všetky tieto hodnoty bolo nutné simulovať v LabVIEW tak, aby sme boli schopní ovplyvniť ich hodnoty, prípadne, aby sa logicky menili, v závislosti na stave linky. K tomu slúži ovládací panel (obr. 26), ktorý vo svoje podstate simuluje HMI panel, ktorý sa na lakovacej linke nachádza. Dôležitým parametrom, ktorý musí spĺňovať stroj, ktorý chceme riadiť „dispečersky“ je, že ovládací panel takéhoto stroja musí byť elektronický (nesmú tam byť fyzické tlačidlá, prepínače, potenciometre). Po zmene hodnoty niektorej premennej, pomocou scady je nutné vytvoriť spätnú väzbu tak, aby sa zmena prejavila aj na užívateľskej obrazovke stroja. Simulovaný systém samozrejme túto podmienku spĺňuje, avšak u strojov, ktoré majú fyzické tlačidlá je možno tieto tagy iba monitorovať, avšak ich riadenie nie je možné aplikovať.

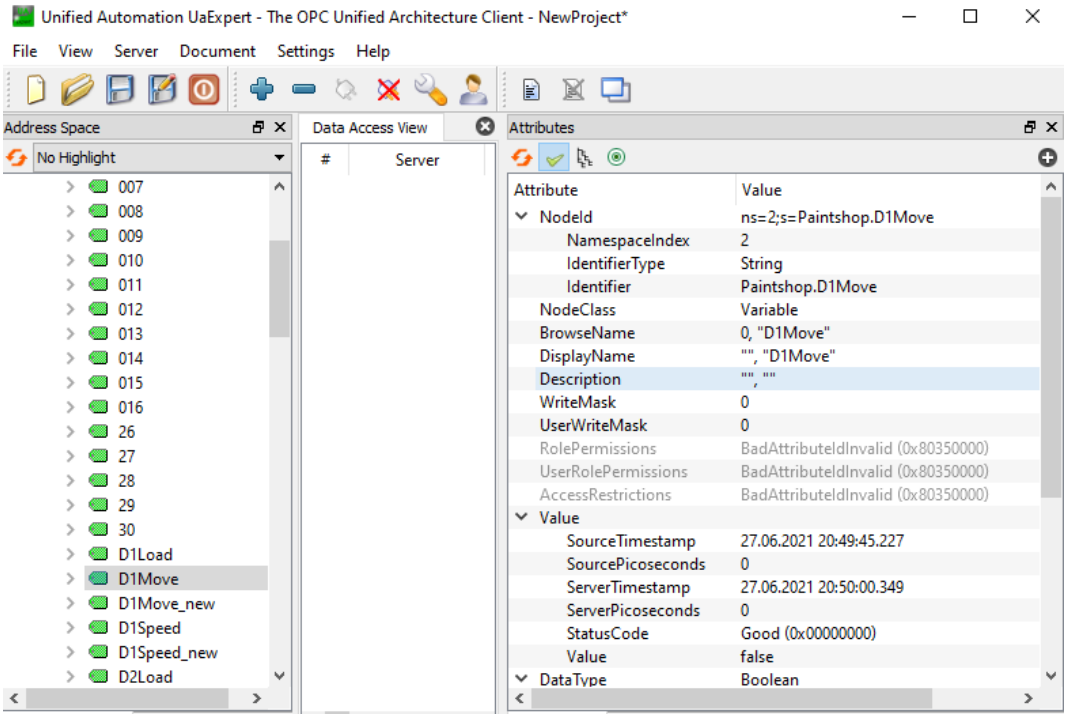



*obr. 26: LabVIEW - časť ovládacieho panelu*

## 6.4 Virtuálny OPC Server

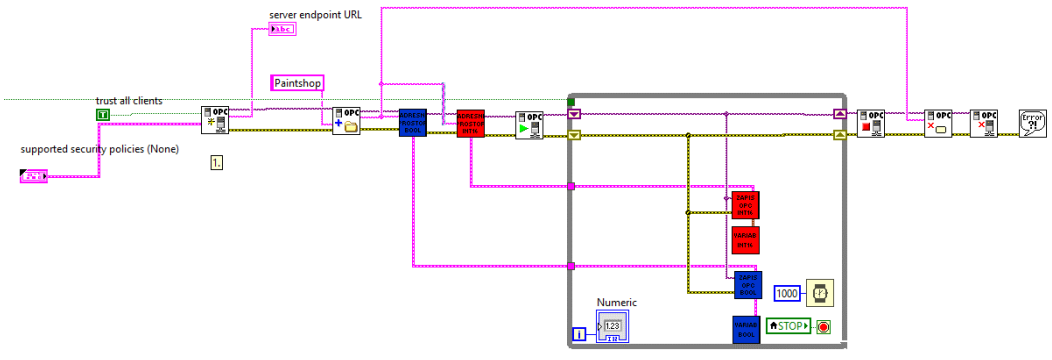
Jedným z cieľov práce je sfunkčnenie komunikácie s využitím opc-ua protokolu. Aby toto bolo možné otestovať, bola vytvorená simulácia opc servera. Novšie verzie LabVIEW obsahujú funkčné bloky, ktoré umožňujú komunikovať s využitím tohto protokolu. V prvom kroku bolo potrebné tento server zinicializovať a nastaviť jeho parametre, ako napríklad zabezpečenie alebo adresu. Ďalším krokom je tvorba priečinku, do ktorého budeme následne ukladať načítané premenné. Ďalej nasleduje tvorba adresného priestoru pre jednotlivé premenné. Každá premenná, ktorú chceme do tohto adresného priestoru pridať musí mať definovaný názov, dátový typ a následne aj prístupové práve (zápis/čítanie). Následne je takto predpripravený server spustený, pričom jeho úlohou je v slučke získavať hodnoty definovaných premenných zo simulácie a ukladať ich na vytvorený server.

Pre prvotné overenie správnej tvorby serveru bol použitý program UaExpert, verzia 1.5.1 (obr. 27). Tento softvér funguje ako opc klient, to znamená, že jeho úlohou je vyslať požiadavku serveru (napríklad na zobrazenie žiadanej hodnoty premennej), pričom server túto požiadavku splní.




*obr. 27: Overenie funkčnosti OPC serveru*

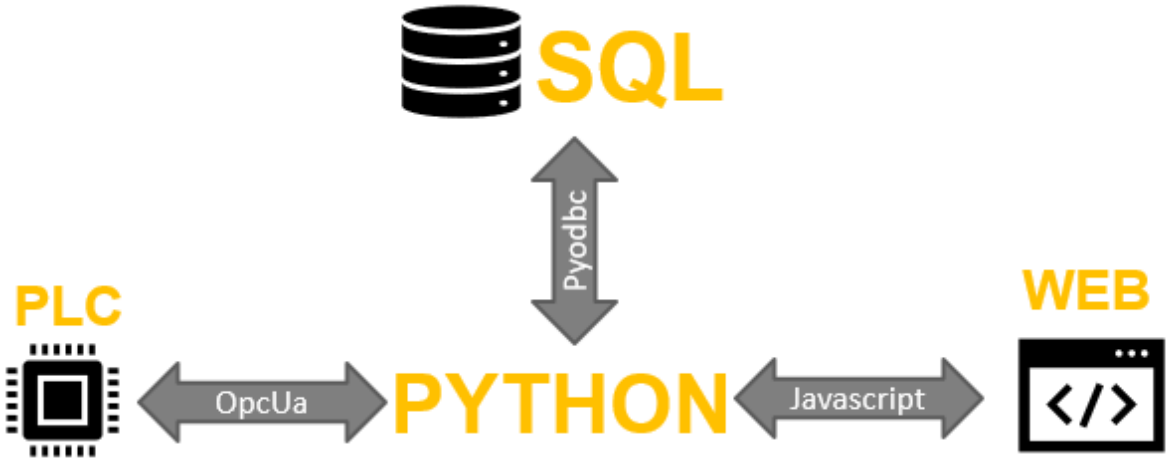
Z obrázka je zrejme, aké všetky atribúty je možné vyčítať pre zvolenú premennú - napríklad hodnotu alebo dátový typ. Pomocou tohto programu je možné jednoducho prehliadať všetky premenné, ktoré sú dostupné na serveri. Pre pripojenie na server je potrebné len jeho URL, prípadne autorizačné údaje. Kód pre tvorbu simulovaného serveru je na obr. 28.




*obr. 28: Kód pre vytvorenie a zápis na opc server*

# 7 SCADA systém

Pri návrhu štruktúry systému boli zohľadnené viaceré možnosti, ktorých vzájomné porovnanie a následný výber, sú uvedené v kapitolách 2.5 a 4.2.5. Výsledná štruktúra, ktorá bola v rámci práce spracovaná je zobrazená na obr. 29.




*obr. 29: Štruktúra výsledného systému*

Zo schémy je zrejmé, že kľúčovú úlohu v scada systéme zohráva programovací jazyk Python. Ako open-source software s nespočetným množstvom použiteľných knižníc, predovšetkým knižnice **OpcUa** pre komunikáciu s PLC prostredníctvom opc-ua protokolu a **Pyodbc** pre prácu s databázami, bol tento programovací jazyk vyhodnotený ako najvhodnejší. Ďalšou silnou stránkou je dátová analýza, ktorá je potrebná pri ďalšom spracovaní dát (kapacitné výpočty, efektivita).

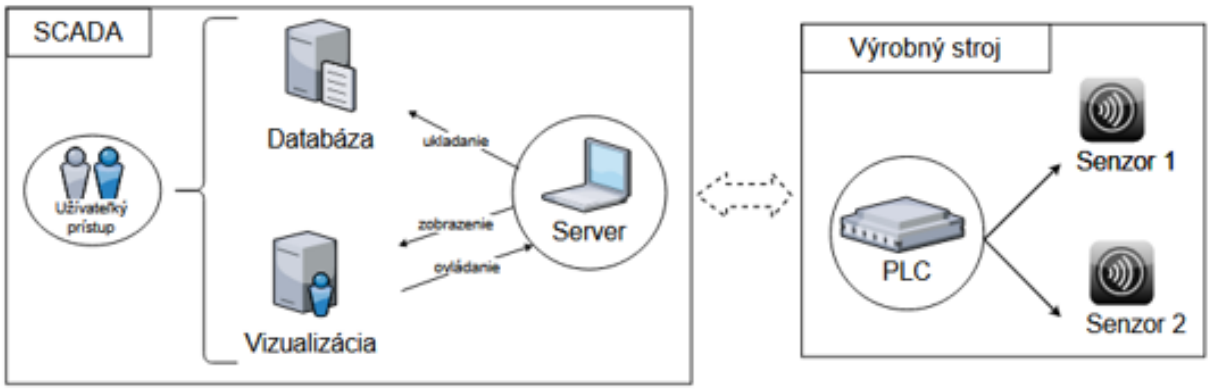
Symbol PLC nereprezentuje priamo programovateľný automat, ale predovšetkým jeho modul, ktorý umožňuje vytvoriť opc server na takomto zariadení. Takýto modul je štandardne integrovaný napríklad v PLC Siemens S7-1500, kde je potrebné si za použitie tejto funkcie priplatiť od 100 do 300€, v závislosti na type licencie. [38]

SQL databáza má v navrhnutom riešení 2 základné funkcie – jednou je uchovávanie všetkých monitorovaných dát a tou druhou je poskytovanie dát webovej aplikácii pre zobrazenie stavu zariadenia. Z tohto dôvodu je dostupnosť SQL servera kľúčová pre funkčnosť a použiteľnosť celej webovej aplikácie. Rýchlosť zápisu do databázy teda ovplyvňuje výsledný výkon systému.

Pravá strana diagramu, na rozdiel od častí spomenutých v predošlých odstavcoch, reprezentuje „front-endovú“ časť systému, čo znamená, že slúži priamo pre koncového užívateľa. Najvhodnejšia forma ako dáta používateľovi vizualizovať je webová stránka. Dôvodom je ľahká dostupnosť z rôznych zariadení, bez nutnosti inštalácie do pamäti zariadenie. Pomocou webu je tak možné kontrolovať stav zariadenia nielen z kancelárie, ale po zriadení VPN pripojenia aj z domova alebo zahraničia. Sprostredkovateľom komunikácie medzi Python-om a webom je programovací jazyk Javascript.

Organizácia systému a jeho fyzická stavba je zobrazená na obr. 30. Pravá strana reprezentuje riadiaci PLC program, ktorý získava informácie z rôznych senzorov a snímačov, umiestnených v konštrukcii stroja (pri vývoji aplikácie nahradený generátorom dát z kapitoly 6). Ľavá strana reprezentuje samotný systém scada.

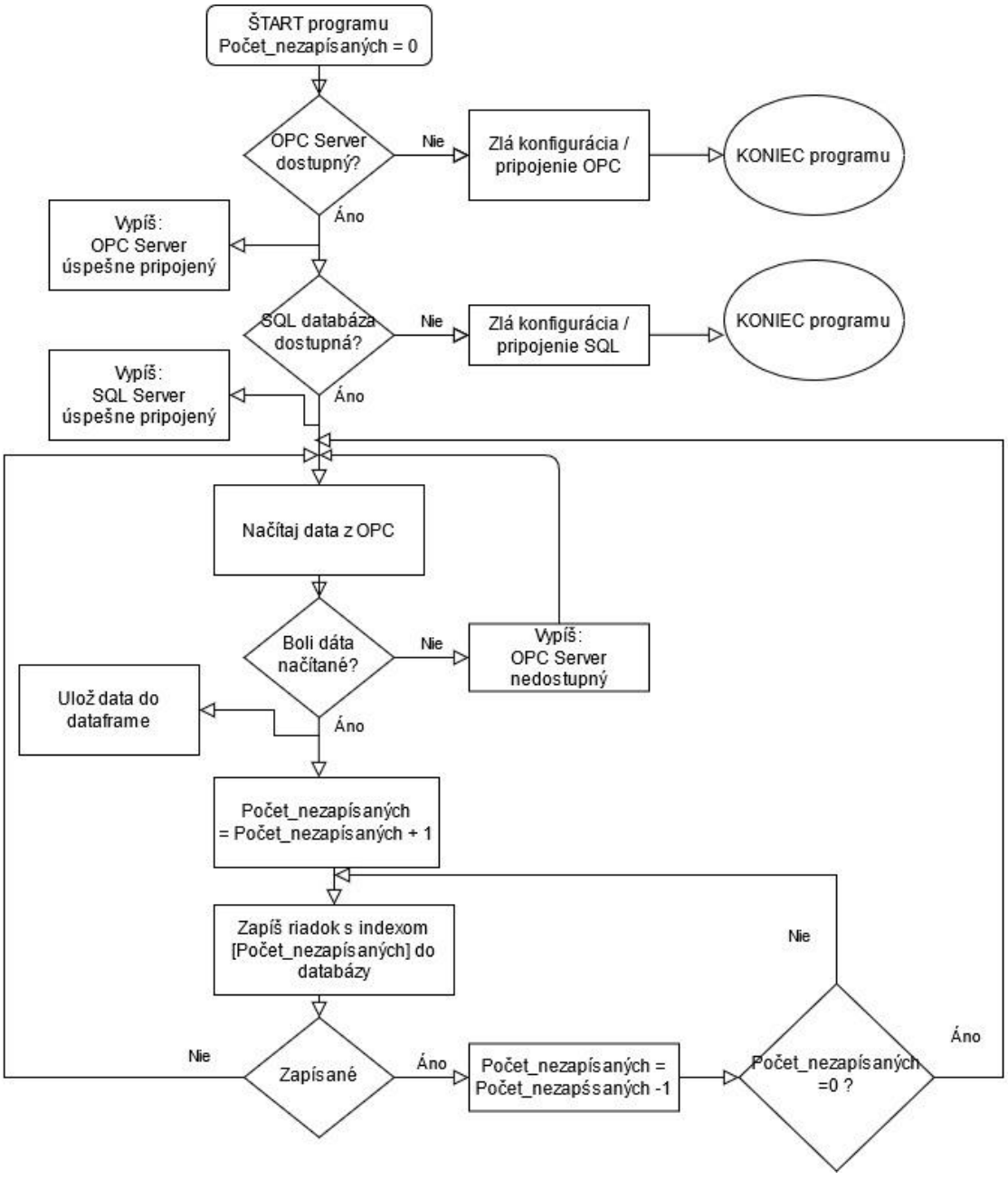
Z obrázku je zrejmé, že účelom navrhnutej vizualizácie nie je len monitorovanie stroja, ale umožňuje aj zasiahnutie do ovládania stroja, teda priamo možno ovplyvňovať stav monitorovaného zariadenia. Pre správnu funkciu je potrebné prepracovať riadiaci systém stroja tak, aby po úkone typu **write** (zapísaní hodnoty na opc server) z vonkajšieho zdroja – webu, bol tento systém schopný túto zmenu načítať a pracovať s ňou rovnako tak, ako keby táto zmena nebola prevedená prostredníctvom serveru, ale priamo prostredníctvom HMI panelu. Táto problematika bola v práci riešená len vo virtuálnom prostredí simulovaného PLC (LabVIEW), ale jej spojazdnenie v reálnom PLC nebolo preskúmané a nie je súčasťou tejto práce.




*obr. 30: Organizácia systému*

## 7.1 Python

Ako bolo uvedené vyššie, program napísaný v jazyku Python je dôležitý pre zber dát zo serveru stroja a ich uloženie do databázy. Celý program je (pre jeho rozsah) priložený v sekcii – Textové prílohy, na konci práce. Na obr. 31 je uvedený jeho vývojový diagram, ktorý popisuje jeho hlavnú činnosť.




*obr. 31: Vývojový diagram zberu dát*

Tento cyklus prebieha v nekonečnej slučke s taktom 5 sekúnd (takt je možné skrátiť).

## 7.2 Vytvorenie grafického užívateľského rozhrania (GUI)

Problematika možností tvorby užívateľského rozhrania boli zhrnuté v kapitole 2.4. Ako najvhodnejší variant sa javí využitie frameworku Vue.js.

### 7.2.1 Návrh rozhrania

Tvorba webovej stránky je pomerne komplikovaný proces, preto je pri prvotnom návrhu (obr. 32) vhodné použiť softvér, ktorý je užívateľsky prívetivejší, napríklad Microsoft Office PowerPoint.

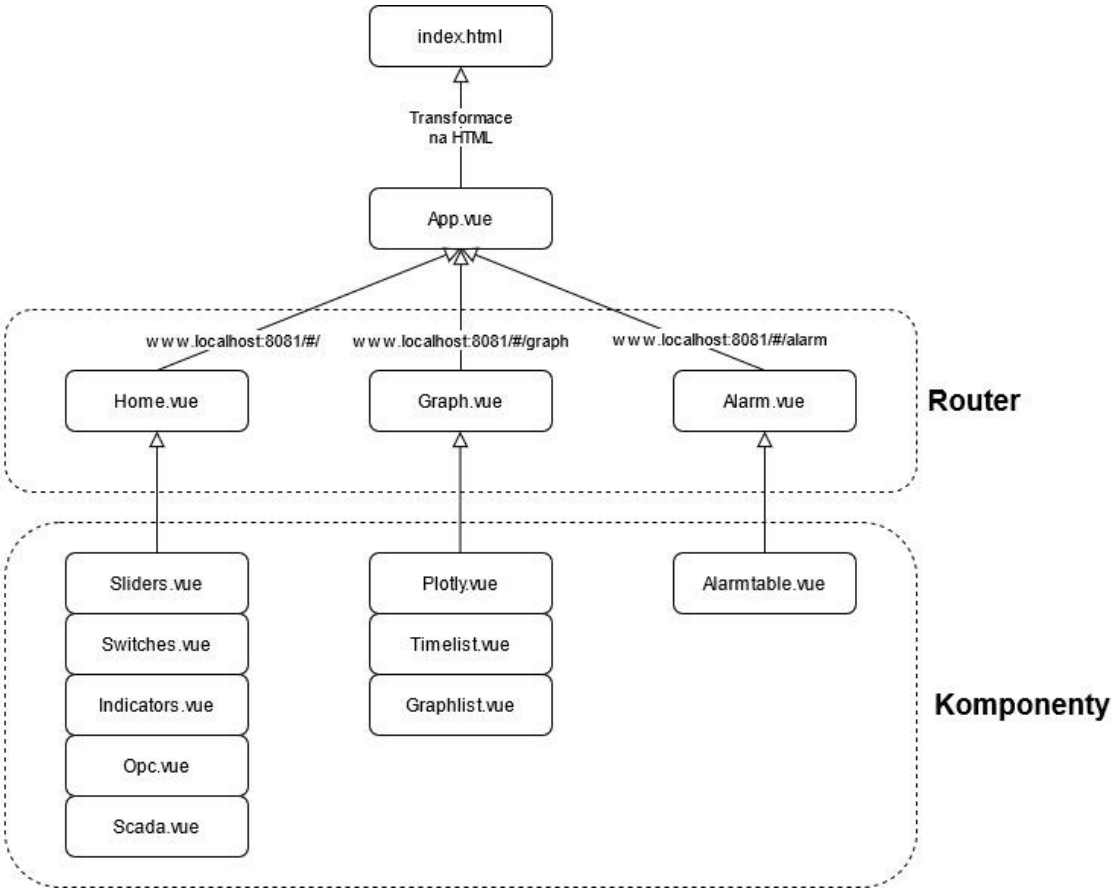
*obr. 32: Prvotný návrh GUI*

### 7.2.2 Tvorba rozhrania

Využitie Vue.js v prepojení s Javascript je spôsob, ako je možné tvoriť dynamickú webovú stránku, ktorá je zobraziteľná na množstve rôznych zariadení, pretože konečná stavba webovej stránky je v jazyku HTML.

Stránka je tvorená z komponentov, ktoré sa skladajú nad seba, vedľa seba alebo sa do seba vnárajú tak, aby bol výsledkom požadovaný grafický výstup. Veľmi nápomocná bola pri tvorbe rozhrania knižnica **Vuetify** [39], ktorá obsahuje množstvo predpripravených komponentov. Komponentom môže byť napríklad prepínacie tlačidlo, list alebo aj zložitejšia tabuľka. Pre lepšiu prehľadnosť výsledného rozhrania je vhodné použiť router, ktorý slúži na tvorbu viac-stránkových webov. Jednotlivé stránky, ktoré je následne možné prepínať medzi sebou, sú vytvorené z rôznych, prípadne z rovnakých komponentov s určitými zmenami (rôzne premenné, limitné údaje, farby, ...). Podľa návrhu boli vytvorené 3 obrazovky – Home, Graph a Alarm. Medzi týmito stránkami prepíname priamo pomocou zadania adresy stránky (napríklad - www.localhost:8081/#/home), prípadne pomocou tlačidiel v hornej časti stránky.

Celková architektúra front-endu je zobrazená na obr. 33, kde sú zobrazené všetky základné komponenty použité pre tvorbu jednotlivých stránok.



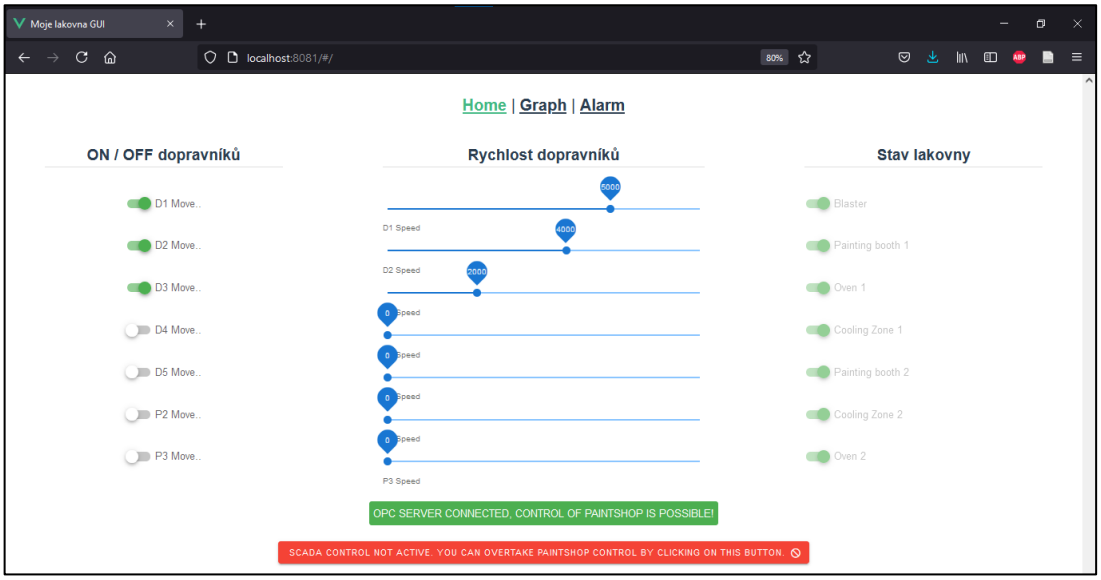

*obr. 33: Architektúra front-endu*

Vzhľad komponent a kód je možné prevziať a upraviť z knižnice Vuetify (obr. 34).


*obr. 34: Komponent Switch v knižnici Vuetify [39]*

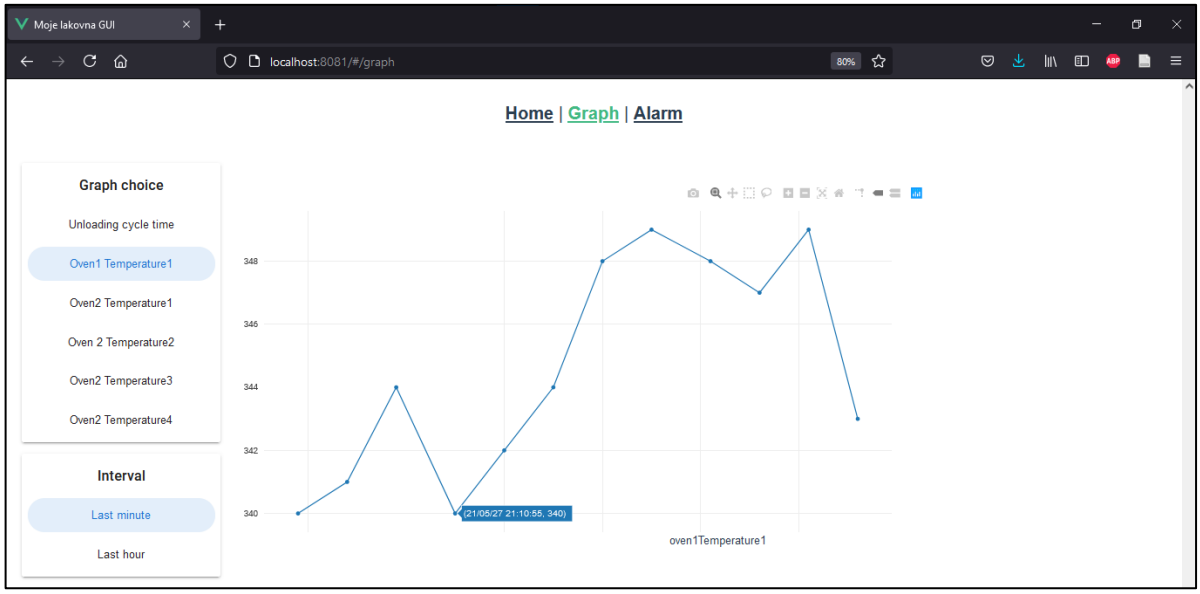
## 7.3 Prepojenie simulačného modelu a grafického rozhrania

Najdôležitejším prvkom pre prepojenie simulačného modelu a grafického rozhrania je knižnica **Eel**. Základným princípom je volanie funkcií napísaných v Python prostredníctvom komponentov na webovej stránke a následne vrátenie výsledku funkcie pre ďalšie využitie na webe. Hlavnými funkciami, ktoré boli takto využité je dotazovanie sa na hodnoty v SQL databáze alebo zápis požadovanej premennej na opc server. Všetky funkcie použité na webe sú uvedené v prílohe SKRIPT – app.py. Na nasledujúcich obrázkoch je zobrazená vybudovaná webová aplikácia, ktorá obsahuje tri základné obrazovky, ktorá obsahujú rôzne zobrazovacie a ovládacie prvky.



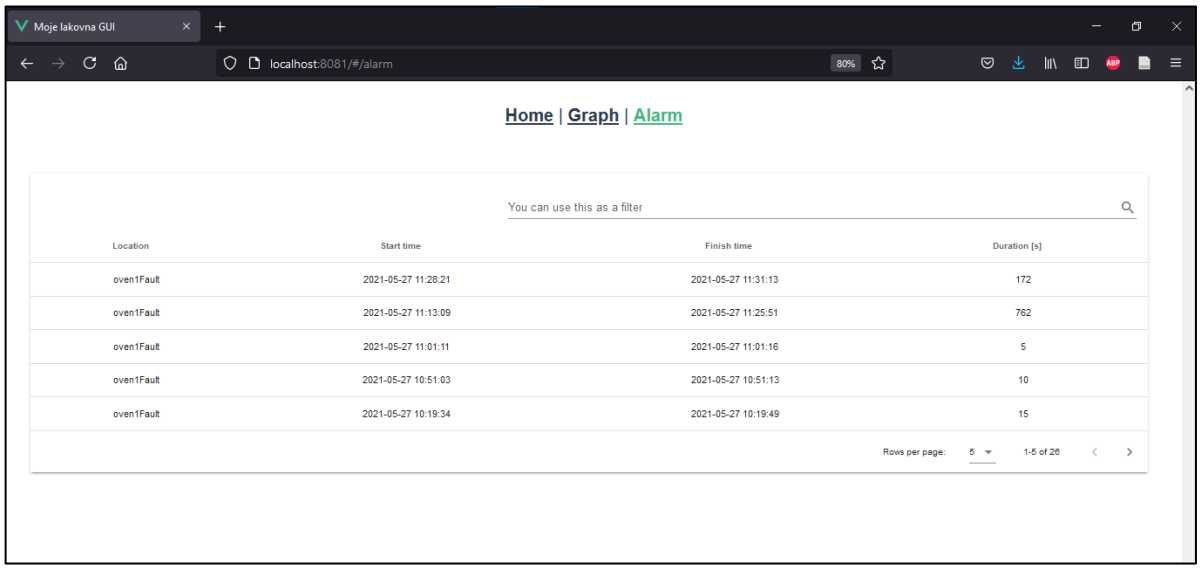

*obr. 35: Úvodná obrazovka*

Na obr. 35 je zobrazená základná obrazovka systému. Cieľom tejto obrazovky je informovať o stave zariadení v linke (dopravníky, procesné bunky). Po aktivovaní SCADA riadenia (červené tlačidlo) je možné ovládať stav dopravníkov v linke (zapnutie/vypnutie, prípadne ich rýchlosť). Zobrazované hodnoty sú automaticky aktualizované, preto nie je nutné webovú stránku obnovovať, aby došlo k zobrazenie aktuálneho stavu.




*obr. 36: Zobrazenie grafov*

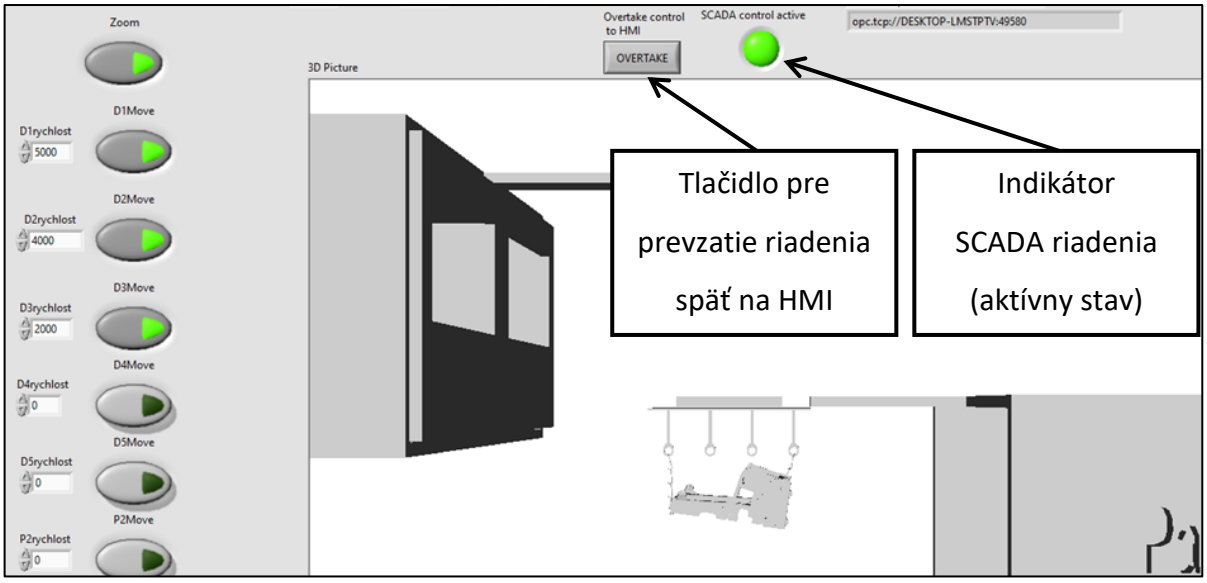
Obrazovka na obr. 36 má za úlohu zobraziť štatistiky zvolenej premennej za zvolené obdobie – napríklad vývoj teploty v peci 1 počas poslednej  minúty/hodiny.




*obr. 37: Zobrazenie alarmov*

Vyššie uvedený obr. 37 informuje a výskyte alarmových stavov a o dĺžke ich trvania. Je možné rôzne radenie a filtrovanie v zobrazenej tabuľke.

Porovnaním stavu zobrazeného na obr. 35 a obr. 38 je zrejme, že monitorovanie funguje a dáta zobrazené simuláciou sa zhodujú s dátami zobrazenými na úvodnej obrazovke webovej aplikácie.



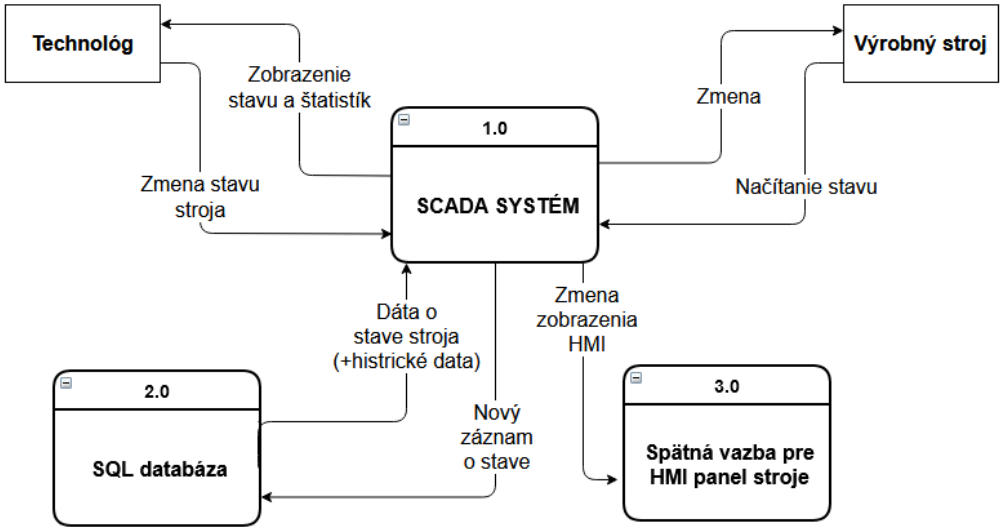

*obr. 38: Záznam zo simulácie*

## 7.4 Dispečerské riadenie

Aby sme systém mohli nazývať scada, je nutné splniť požiadavku na riadenie. Táto funkcia bola implementovaná v prípade ovládania simulácie. Problém pri riešení tejto úlohy bola spolupráca HMI panel, ktorý je v blízkosti stroja a ovládacích prvkov na webe. Navrhnuté riešenie oddelilo riadenie cez HMI panel a web tak, že v jednej chvíli môže byť aktívny iba jeden ovládací panel. Obidva ovládacie panely, na webe aj HMI, obsahujú tlačidlo, pomocou ktorého sú schopné prevziať kontrolu na daný panel. Pomocou panelu, ktorý nie je aktívny je možné monitorovať stroj, ale nie je možné zasahovať do jeho funkcie. Obidva ovládacie panely taktiež obsahujú indikátor, ktorý dáva informáciu o tom, ktorý panel je v danej chvíli aktivovaný. Toto riešenie bolo overené pomocou simulácie.

Pokiaľ je scada riadenie aktivované, potom je možné riadiť výrobný stroj pomocou webu. Po stlačení tlačidla na panely je tento údaj zapísaný do opc servera, odkiaľ si túto informáciu načíta stroj. Virtuálny PLC program je napísaný tak, aby zabezpečil aktualizáciu HMI panelu na

základe hodnoty z webu, a teda je schopný „stlačiť“ tlačidlo na paneli stroja, ak technológ „stlačí“ dané tlačidlo v aplikácii. Na obr. 38 je zobrazený ovládací panel simulácie, na ktorom je vyznačený indikátor scada riadenia a takisto aj tlačidlo, pomocou ktorého je možné kontrolu prevziať späť na HMI.




*obr. 39: Diagram dátových tokov*

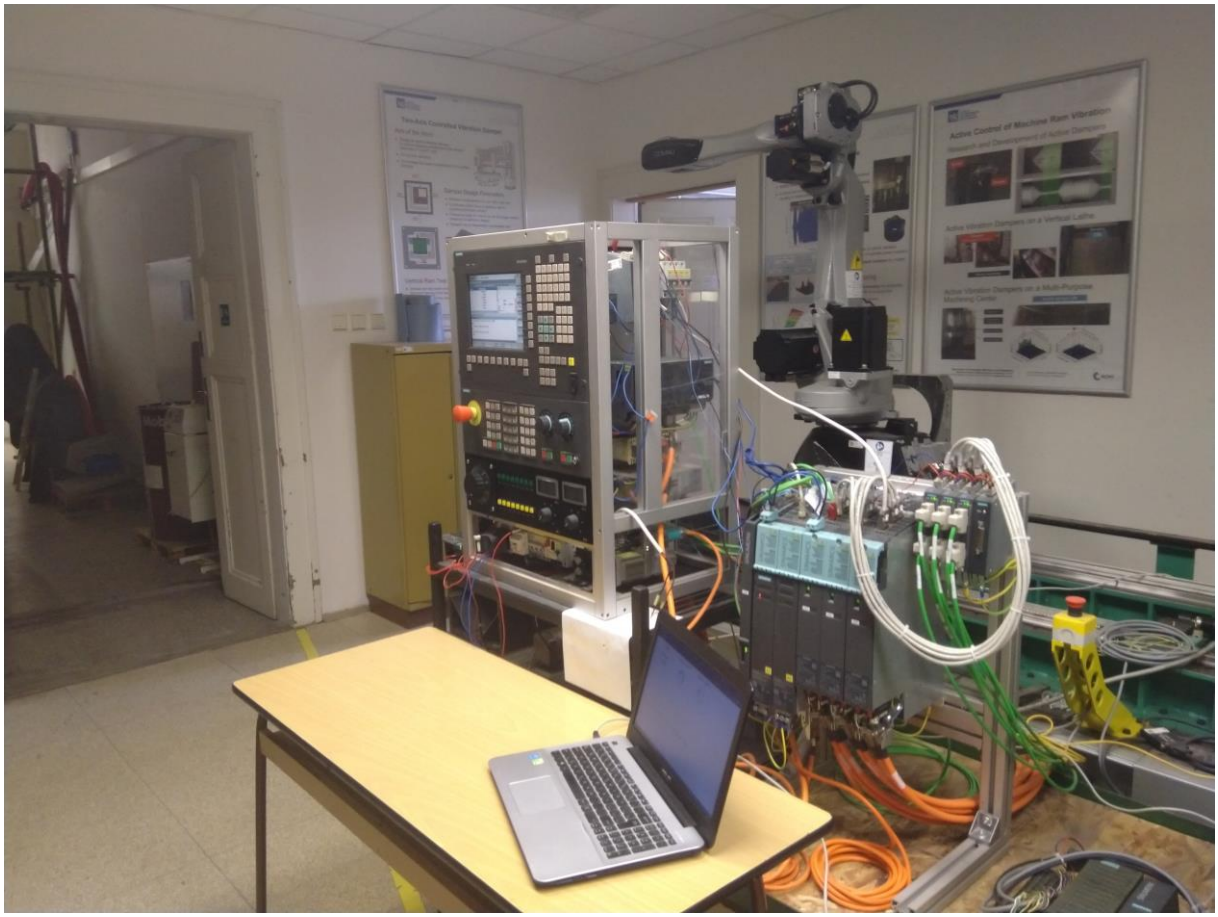
Dátový tok celého systému je zobrazený na obr. 39.

# 8 Overenie funkčnosti aplikácie

Z pohľadu funkčnosti navrhnutého systému sú dôležité dva body, ktoré je nutné overiť:

- Prenositeľnosť systému na iný výrobný stroj
- Overenie systému na reálnom riadiacom systéme

Pre tento účel bol navrhnutý experiment – cieľom je pripojenie sa na opc server riadiaceho systému robota (obr. 40) Sinumerik a monitorovať jeho stav (niekoľko vybraných premenných). Pre tento účel bolo vytvorené aj nové užívateľské rozhranie.



*obr. 40: Experiment*

Robot sa nachádza v priestoroch Ú12135 na Horskej. V priebehu testovania boli vyskúšané dva druhy komunikácie – eternetové pripojenie priamo na riadiaci systém a taktiež eternetové pripojenie prostredníctvom spoločnej siete (pripojenie z kancelárie, bez dosahu stroja). Obidva typy pripojenia boli spojazdnené bez komplikácií.

## 8.1 Úprava programov

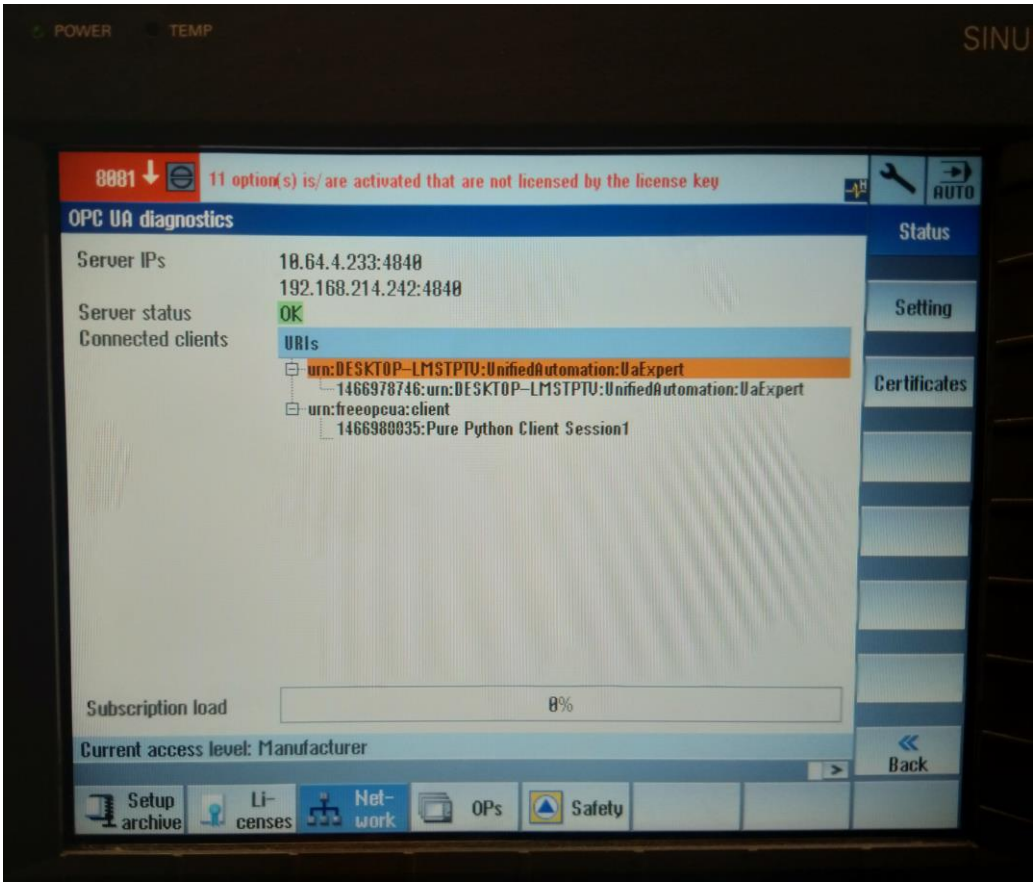
Účelom experimentu bolo aj overiť zložitosť zmeny programov pre monitoring stroja iného typu. Dĺžka úpravy programov je závislá na úrovni prepracovania rozhrania, ale v prípade jednoduchej grafiky stačí aj jeden pracovný deň.

Základné zmeny, ktoré je potrebné urobiť sú tieto:

- Vytvorenie miesta na serveri SQL pre nové premenné
- Úprava komponentov užívateľského rozhrania
- Úprava parametrov opc servera a načítaných tagov

## 8.2 Realizácia testu

Test potvrdil funkčnosť pripojenia pomocou programu UaExpert a takisto pomocou Python-u. Obidve inštancie vytvoreného komunikačného kanálu sú na obr. 41.

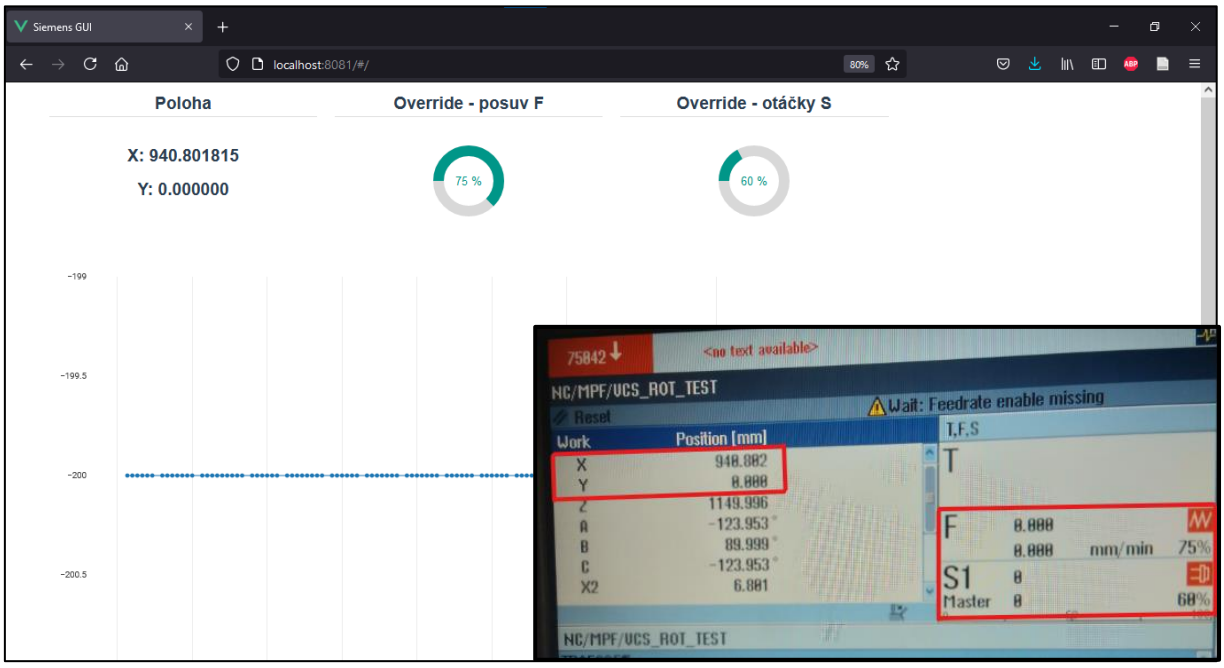



*obr. 41: Overenie pripojenia opc ua*

Vybranými monitorovanými prvkami boli:

- Poloha X a Y
- Override posuvu a otáčok
- Teplota z náhodného snímača

Z obr. 42 je zrejmé, že monitorovanie stroja prebehlo v poriadku. Teplota zo snímača indikuje -200 °C, čo znamená, že je nefunkčný, prípadne nepripojený.




*obr. 42: Porovnanie*

Zvolený experiment potvrdil funkčnosť systému v prípade monitorovanie. Scada riadenie systému nebolo v tejto práci overované, pretože jeho implementácia do PLC vyžaduje zásah do riadiaceho systému stroja.

# 9 Záver

Táto práca sa zaoberá systémom SCADA a jeho použitím v podniku. Úlohou prvej časti práce, ktorá sa venuje rešerši súčasného stavu danej problematiky, bolo preskúmať varianty riešení a zhodnotiť ich použiteľnosť / výhodnosť. Táto časť práce bola venovaná aj priemyselnej komunikácii, kde boli porovnané najbežnejšie komunikačné protokoly a došlo k výberu tej správnej varianty, ktorá bola využitá pri stavbe a testovaní vytvoreného systému. Teoretická časť popisuje aj využitie programovateľných automatov z pohľadu výrobného stroja, ale aj v rámci celého SCADA systému. Ďalej bola popísaná problematika databázových systémov a ich využiteľnosť z pohľadu managementu podniku.

Hlavným cieľom práce je návrh systému, ktorého úlohou je zobrazovať stav stroja, prípadne ho riadiť. Pre tento účel bol vytvorený generátor dát (simulácia výrobného procesu), ktorý je odvodený od už existujúceho výrobného zariadenie a snaží sa jeho chovanie simulovať. Tento generátor vznikol s využitím softwarov Siemens NX a LabVIEW, pričom funguje ako virtuálne PLC, ktorá bolo následne použité pre vývoj a testovanie vytvoreného systému.

Ďalším významným bodom bol návrh a vytvorenie grafického užívateľského rozhrania pomocou zvolených webových technológií, pričom bola zahrnutá požiadavka na jednoduchú prenositeľnosť a funkčnosť aj na stroji iného typu.

Stredobodom navrhnutého systému je programovací jazyk Python, ktorý nielenže zabezpečuje zber dát zo stroja prostredníctvom zvoleného komunikačného prostriedku (protokol opc ua), ale v konečnom dôsledku aj zabezpečuje ukladanie dát do databázy a ich prenos na obrazovku konečného užívateľa softwaru.

Vytvorený systém SCADA bol následne otestovaný na robotu s riadiacim systémom Sinumerik, čím došlo k overeniu možnosti/jednoduchosti prispôsobenia užívateľskej obrazovky na rozdielny stroj a takisto funkčnosť systému v prípade reálneho (nesimulovaného) stroja. V rámci tohto testovania bola implementovaná iba funkcia monitorovania zariadenia, nakoľko problematika radenia takéhoto stroja by vyžadovala zásahy do RS, čo bolo nad rámec tejto diplomovej práce.

# Zoznamy

## Zoznam použitej literatúry

[1] Priemysel 4.0. Buď FIT [online]. [cit. 2020-11-08]. Dostupné z: https://casopis.fit.cvut.cz/tema/priemysel-4-0/.

[2] Keller und Knappich Augsburg [online]. KUKA: © KUKA AG 2021 [cit. 12.7.2021]. Dostupné z: https://www.kuka.com/cs-cz.

[3] MES systém (Manufacuring Execution system). MEScenter [online]. [cit. 2020-10-03]. Dostupné z: http://www.mescenter.org/cz/clanky/5-co-je-to-mes-system.

[4] Chytrá domácnosť. Somfy [online]. [cit. 2020-10-13]. Dostupné z: https://www.samezaluzie.cz/akce/somfy-chytra-domacnost/.

[5] MySCADA [online]. [cit. 2020-10-06]. Dostupné z: https://www.myscada.org/cs/.

[6] Katalog produktů. MySCADA [online]. [cit. 2020-10-06]. Dostupné z: https://www.myscada.org/downloads/other/mySCADA_product%20guide%20CZ.pdf.

[7] Automa [online]. 2006 [cit. 2020-10-13]. Dostupné z: https://automa.cz/cz/casopis-clanky/prenos-dat-ze-systemu-scada-simatic-wincc-2006_06_31203_2708/.

[8] WinCC. Siemens SCADA [online]. [cit. 2020-10-13]. Dostupné z: https://assets.new.siemens.com/siemens/assets/api/uuid:69ce4c32-4537-4432-8bee-4a92cf8c2488/width:1125/quality:high/wincc-engineering-ui.png.

[9] Glade. LinuxExpres [online]. [cit. 2020-10-25]. Dostupné z: https://www.linuxexpres.cz/software/glade-graficke-prostredi-snadno-a-rychle.

[10] File:Glade 3.1.15. WikimediaCommons [online]. [cit. 2020-10-25]. Dostupné z: https://commons.wikimedia.org/wiki/File:Glade_3.15.1_Hebrew.png.

[11] Matlab. Wikipedie [online]. [cit. 2020-10-25]. Dostupné z: https://cs.wikipedia.org/wiki/MATLAB.

[12] MATLAB App Designer. MathWorks [online]. [cit. 2020-11-04]. Dostupné z: https://www.mathworks.com/products/matlab/app-designer.html.

[13] Themes. Vue.js [online]. [cit. 2020-10-28]. Dostupné z: https://vuejs.org/resources/themes.html.

[14] Rychlý start: Informace o možnostech. Microsoft [online]. [cit. 2020-10-28]. Dostupné z: https://docs.microsoft.com/cs-cz/power-bi/consumer/end-user-reading-view#next-steps.

[15] Funkce|Microsoft. Microsoft|Power BI [online]. [cit. 2020-10-28]. Dostupné z: https://powerbi.microsoft.com/cs-cz/features/.

[16] User Interfaces with LabVIEW NXG Web Module. GPower [online]. [cit. 2020-11-08]. Dostupné z: https://info.gpower.io/en/blog/user-interfaces-with-labview-nxg-web-module.

[17] Prostředky průmyslové automatizace [online]., [cit. 2021-7-5]. Dostupné z: https://www.sssebrno.cz/cardfiles/card-16511/card-17273/files/prostredky-prumyslove-automatizace.pdf_488fd7d151845338110571d83a0522981556547799.pdf.

[18] Co to je PLC a k čemu? Elektrika.cz [online]. [cit. 2020-10-20]. Dostupné z: https://elektrika.cz/data/clanky/co-to-je-plc-a-k-cemu.

[19] Siemens 6ED1052-1CC08-0BA0 6ED1052-1CC08-0BA0 PLC řídicí modul 24 V/DC. www.conrad.cz [online]. [cit. 2020-10-20]. Dostupné z: https://www.conrad.cz/p/siemens-6ed1052-1cc08-0ba0-6ed1052-1cc08-0ba0-plc-ridici-modul-24-vdc-1628682?&vat=true&gclid=CjwKCAjwlbr8BRA0EiwAnt4MTuTe59ANF2MBa-VWK1u1etb5evO54H1-zUDOMVi1d0aINrKBxIrVjBoCNCUQAvD_BwE.

[20] Představení protokolu MQTT a jeho aplikace v jednotkách Poseidon2 a Damocles2. HWgroup [online]. [cit. 2020-11-01]. Dostupné z: https://www.hw-group.com/cs/podpora/predstaveni-protokolu-mqtt-a-jeho-aplikace-v-jednotkach-poseidon2-a-damocles2.

[21] Průmyslová komunikace OPC UA - 1.díl - popis protokolu [online]. [cit. 2020-11-01]. Dostupné z: https://automatizace.hw.cz/prumyslova-komunikace-opc-ua-1dil-popis-protokolu.html.

[22] OPC Unified Architecture. Berlin: Springer, 2009. ISBN 978-3-540-68898-3.

[23] EtherCAT Automation Protocol [online]. [cit. 2020-11-01]. Dostupné z: https://automatizace.hw.cz/ethercat-automation-protocol.html.

[24] EtherNet / IP versus PROFINET [online]. [cit. 2020-11-01]. Dostupné z: https://automatizace.hw.cz/ethernet-ip-versus-profinet.html.

[25] Hypertext Markup Language. Wikipedie [online]. [cit. 2020-10-28]. Dostupné z: https://cs.wikipedia.org/wiki/Hypertext_Markup_Language.

[26] A Brief Decription [online]. [cit. 2020-11-01]. Dostupné z: https://www.cplusplus.com/info/description/.

[27] Understanding the Differences Between C#, C++, and C. C# STATION [online]. [cit. 2020-11-01]. Dostupné z: https://csharp-station.com/understanding-the-differences-between-c-c-and-c/.

[28] Python Introduction [online]. [cit. 2020-11-01]. Dostupné z: https://www.w3schools.com/python/python_intro.asp.

[29] Proč je dobré umět programovací jazyk Java. Engeto [online]. [cit. 2020-11-01]. Dostupné z: https://engeto.cz/blog/programovani/proc-je-dobre-umet-programovaci-jazyk-java/.

[30] 10 Best Programming Languages of 2020 You should know (Updated). Devsaran [online]. [cit. 2020-11-01]. Dostupné z: https://www.devsaran.com/blog/10-best-programming-languages-2020-you-should-know.

[31] Co je to databáze? Oracle [online]. [cit. 2020-11-03]. Dostupné z: https://www.oracle.com/cz/database/what-is-database.html.

[32] Databázové modely. Databáze [online]. [cit. 2020-11-03]. Dostupné z: http://www.databaze.chytrak.cz/modely.htm.

[33] LACKO, Ľuboslav. 1001 tipů a triků pro SQL. Brno: Computer Press, 2011. ISBN 978-80-251-3010-0.

[34] Základy relačních databází, jejich využití v programování webu [online]. , 8-10 [cit. 2020-11-03]. Dostupné z: https://kme.vse.cz/wp-content/uploads/page/534/4.-Z%C3%A1klady-rela%C4%8Dn%C3%ADch-datab%C3%A1z%C3%AD-jejich-vyu%C5%BEit%C3%AD-v-programov%C3%A1n%C3%AD-webu.pdf.

[35] SQL skripty, uživatelské procedury a funkce. Matematická biologie [online]. [cit. 2020-11-03]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=zaklady-informatiky-pro-biology--databazove-systemy-v-biomedicine--sql-skripty-uzivatelske-procedury-a-funkce.

[36] Relační Databáze. Nauč se Python! [online]. [cit. 2020-11-08]. Dostupné z: https://naucse.python.cz/2019/brno-podzim-2019-ut/beginners/database/.

[37] How to Connect Python to SQL Server using pyodbc. Data To Fish [online]. [cit. 2020-11-08]. Dostupné z: https://datatofish.com/how-to-connect-python-to-sql-server-using-pyodbc/.

[38] Digitalizace s TIA Portal V15 [online]. [cit. 2021-7-3]. Dostupné z: https://www.tianadosah.cz/upload/digitalizace-tia-portal-v15-a-opc-ua-info.pdf.

[39] Vuetify [online]. [cit. 2021-7-3]. Dostupné z: https://vuetifyjs.com/en/.

# Zoznam obrázkov

## Zoznam tabuliek



## Zoznam použitého softwaru

NI LabVIEW 2019 SP1

Siemens NX 12.0

UaExpert 1.5.1

Python 3.9

Visual Studio Code 1.57

Microsoft SQL Server Studio Management 18

Node.js 14

Microsoft Office 365

# Zoznam textových príloh

SKRIPT – scada.py

SKRIPT - app.py

SKRIPT – index.html

SKRIPT – sliders.vue

# Zoznam elektronických príloh

Projekt LabVIEW – generátor dát

Projekt Visual Studio Code – projekt pre monitorovanie simulácie a RS Sinumerik