SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

Slovenská technická univerzita v Bratislave, Fakulta elektrotechniky a informatiky

Ing. Filip Žemla

Autoreferát dizertačnej práce

Digitalizácia udalostných systémov s využitím inteligentných technológií pre Industry 4.0

na získanie akademickej hodnosti doktor (philosophiae doctor, PhD.)

v doktorandskom študijnom programe: Mechatronické systémy
v študijnom odbore: Kybernetika
forma štúdia: denná

Miesto a dátum: Bratislava, 30.05.2023

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta elektrotechniky a informatiky

Ing. Filip Žemla

Autoreferát dizertačnej práce

Digitalizácia udalostných systémov s využitím inteligentných technológií pre Industry 4.0

na získanie akademickej hodnosti doktor (philosophiae doctor, PhD.)

v doktorandskom študijnom programe: Mechatronické systémy

Miesto a dátum: Bratislava, 30.05.2023

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia

Dizertačná práca bola vypracovaná na:

Ústave automobilovej mechatroniky
Fakulta elektrotechniky a informatiky STU v Bratislave
FEI STU BA, Ilkovičova 3, 812 19 Bratislava

Predkladateľ:

Ing. Filip Žemla
Ústav automobilovej mechatroniky
Fakulta elektrotechniky a informatiky
Slovenská technická univerzita v Bratislave
Ilkovičova 3, 812 19, Bratislava

Školiteľ:

prof. Ing. Danica Rosinová, PhD.
Ústav automobilovej mechatroniky
FEI STU BA, Ilkovičova 3, 812 19 Bratislava

Oponenti:

doc. Ing. Martin Juhás, PhD.
Ústav aplikovanej informatiky, automatizácie a mechatroniky
MTF STU TT, Jána Bottu 2781/25, 917 24 Trnava

Ing. Ivana Budinská, PhD.
Ústav informatiky SAV
Slovenská akadémia vied, Dúbravská cesta 9, 845 07 Bratislava

Autoreferát bol rozoslaný: .....

Obhajoba dizertačnej práce sa koná: .....

Na: Fakulte elektrotechniky a informatiky STU v Bratislave, Ilkovičova 3, 812 19 Bratislava, v miestnosti D-101

.....

prof. Ing. Vladimír Kutiš, PhD.

Abstrakt

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta elektrotechniky a informatiky
Ústav automobilovej mechatroniky

Študijný program: Mechatronické systémy
Študijný odbor: Kybernetika
Autor: Ing. Filip Žemla
Školiteľka: prof. Ing. Danica Rosinová, PhD

Názov práce: Digitalizácia udalostných systémov s využitím inteligentných technológií pre Industry 4.0
Kľúčové slová: rozšírená realita, PLC, Unity, výrobný proces, Industry 4.0, digitalizácia

Nachádzame sa v období 4. priemyselnej revolúcie, v období implementácie nových technológií do výroby. Jednou z takýchto technológií je aj implementácia rozšírenej reality vo výrobnom procese. To nám poskytuje nové spôsoby zobrazovania dát a ovládania zariadení. Cieľom dizertačnej práce je navrhnúť a implementovať riešenie, ktoré poskytne zobrazenie údajov a umožní nové spôsoby monitorovania, diagnostiky a riadenia diskrétneho udalostných systému s podporou rozšírenej reality. V práci sú využívané aj široko akceptované technológie pre Industry 4.0, ako je napríklad protokol OPC UA Unified Architecture. Ďalej budú využité technológie ako cloud computing, rozšírená realita, webové aplikácie a priemyselné komunikačné protokoly. Predložená práca stručne charakterizuje aktuálnu situáciu a trendy v zmysle budovania Industry 4.0 so zameraním na monitorovanie a riadenie udalostných systémov s využitím najnovších digitálnych technológií. Práca opisuje základný koncept a štruktúru navrhovaného riešenia a postup realizácie pre zvolenú prípadovú štúdiu.

Abstract

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
Faculty of electrical engineering and information technology
Institute of Automotive Mechatronics

Study Programme: Mechatronics systems
Study Branch: Cybernetics
Author: Ing. Filip Žemla
Supervisor: prof. Ing. Danica Rosinová, PhD

Thesis: Digitization of event systems using intelligent technologies for Industry 4.0
Keywords: Mixed reality, PLC, Unity, manufacturing process, Industry 4.0, digitization

We are in the period of the 4th industrial revolution, in the period of implementation of new technologies in production. One such technology is the implementation of augmented reality in the production process. This gives us new ways of displaying data and controlling devices. The aim of the dissertation is to design and implement a solution that will provide data display and enable new ways of monitoring, diagnosis and control of a discrete event system with the support of augmented reality. The work also uses widely accepted technologies for Industry 4.0, such as the OPC UA Unified Architecture protocol. Technologies such as cloud computing, augmented reality, web applications and industrial communication protocols will also be used. The presented work briefly characterizes the current situation and trends in terms of building Industry 4.0 with a focus on monitoring and controlling of event systems using the latest digital technologies. The work describes the basic concept and structure of the proposed solution and the implementation procedure for the chosen case study.

Obsah

Úvod

Nástupom 4. priemyselnej revolúcie Industry 4.0 sa spája aj vznik nových technológií a ich spojení. Využitie elektrickej energie, pary a automatizácie boli v predchádzajúcich revolúciách neodmysliteľnou súčasťou, rovnako ako v dnešnej dobe využitie internetu a prepojenie zariadení. Zariadenia slúžiace na rôzne účely môžeme tvoriť predovšetkým na základe vyššieho výkonu a menším rozmerom hardvéru, čo nám umožňuje zachytávanie dát z výroby, pričom vzniká veľký objem dát. K samotnému spracovaniu dát je nutné zabezpečenie dostatočného hardvérového systému k čomu slúži využitie cloudových technológií. Vďaka tomuto využitiu je možné vytvorenie digitálneho výrobného prostredia.

Práca sa zaoberá implementáciou prvkov Industry 4.0 do mechatronických systémov vo výrobnom procese. Úvodné časti sa venujú aktuálnemu stavu v skúmanej oblasti a prípadovým štúdiám zaoberajúcim sa touto problematikou. Na základe súčasného stavu definujeme ciele a postup realizácie navrhovanej digitálnej platformy.

V hlavnej časti dizertačnej práce je opísaný postup návrhu digitálnej platformy a získané výsledky. Návrh sa člení na niekoľko častí. Prvá časť sa venuje vývoju automatizácií laboratórnych výrobných modelov, tvorbe sieťovej štruktúre a vývoju 2D vizualizácie. V druhej časti opisujeme vývoj lokálneho servera, ktorý poskytuje dáta z laboratórnych výrobných modelov pre všetky ostatné technológie využité v práci. V tretej časti sa zaoberáme vývoju cloudových technológií a tvorbe webových technológií. A v poslednej časti opisujeme vývoj Unity aplikácie pre zmiešanú realitu, ktorá bude nahraná do MS HoloLens 2 zariadenia.

1 Analýza súčasného stavu

V analýze súčasného stavu sme sa predovšetkým venovali výsledkom a súčasnému stavu v oblasti rozšírenej a zmiešanej reality a komunikačným prepojením medzi smart zariadeniami. Zameriavame sa na výsledky venované monitorovaniu a riadeniu mechatronických systémov s využitím počítačom generovanej reality. Nakoľko sa naša práca týka viacerých technológií a odborov, analýzu môžeme rozdeliť na dva celky:

- Analýza komunikácie medzi smart zariadeniami
- Zmiešaná a rozšírená realita

Komunikácia medzi zariadeniami

Dôležitou časťou analýzy je preskúmanie možností použitia komunikačných technológií medzi smart zariadeniami, lokálnym serverom a cloudovým serverom. Na základe analýzy možno vybrať optimálne komunikačné protokoly, rozhrania a taktiež cloudové technológie. Jedným z analyzovaných výskumov bola simulačná továreň [1] od autorov z univerzity v Cincinnati v Spojených štátoch amerických. Výsledkom štúdie bolo pripojenie výrobného zariadenia ovládaného PLC zariadením na lokálny server KEPServerEX. Následne sa dáta odosielali na cloudový server ThingWorx, odkiaľ boli dáta odosielané do AR aplikácie.

Sústava používala dva komunikačné protokoly, Ethernet protokol a OPC UA protokol. Obsahovala aj dva servery, pričom jeden bol lokálny server KEPServerEX a druhý bol cloudový server Thingworx. Riadiaci počítač so SCADA systémom prijímal dáta z PLC zariadenia cez OPC UA protokol, ktoré následne odosielal do KEPServerEX servera. KEPServerEX prijaté dáta následne odosielal do cloudového servera a na lokálnej sieti do AR zariadenia. KEPServerEX server prijímal dáta z riadiaceho počítača cez protokol OPC UA a ďalej vysielal dáta v Ethernet protokole. Nakoľko bol server Thingworx cloudový server, logicky používal Ethernet protokol.

Ďalšou experimentálnou sústavou bola platforma obchodovania s energiou s Peer to Peer (P2P) komunikáciou [2], ktorou sa zaoberali autori z kanadskej univerzity v Newfoundland. Práca sa zaoberala vzdialeným prístupom k dátam zariadenia a ovládaním zariadenia na diaľku. Tento projekt bol realizovaný pomocou P2P

komunikácie s využitím MQTT protokolu. Platforma obsahovala dva lokálne servery, prvým z nich bol Node-Red, ktorý zastupoval pozíciu MQTT brokera. Druhý server poskytoval webové užívateľské rozhranie. Po prihlásení užívateľa bola zahájená P2P komunikácia so zariadeniami.

Zmiešaná a rozšírená realita

V tejto časti analýzy sme sa venovali možnostiam zobrazenia rozšírenej reality a ukotvenia virtuálnych objektov v realite. Tomuto problému sa venovali aj ďalšie štúdie. Prvou z nich je projekt, ktorý sa venuje zobrazeniu nových dielov v konštrukcii pomocou rozšírenej reality [3]. Na tomto projekte sa podieľali autori z gréckej univerzity v Patras. XR aplikácia obsahuje možnosť pridania základných geometrických 3D modelov ako sú napríklad kocky, valce, gule, pyramídy atď. v rôznych rozmeroch, ktoré užívateľ dokáže meniť. 3D modely je možné umiestniť kdekoľvek v priestore. Pri funkcii „uložiť model“ sa zoskenuje stroj, na ktorom je umiestnený 3D model. Skenovanie pozostáva z natočenia videa pozorovania stroja v rôznych uhloch. Po dokončení skenovania sa video rozdelí na sériu obrázkov príslušného stroja a odošle na cloudový server. Na cloudovom serveri sa následne vytvorí predpoveď (obraz) stroja, ktorá pri nasledujúcom zobrazení stroja v realite rozpozná stroj a umožní mu pridať virtuálny návrh. Pri uložení návrhu sa spolu s videom odošle aj virtuálny návrh, ktorý sa odošle ako pole voxelov (častíc objemu).

Ďalším príkladom je štúdium ovládania robotov pomocou rozšírenej reality [4], ktorou sa zaoberali vedci z univerzity v New Yorku. V tomto prípade projekt obsahoval niekoľko robotov, ktorí disponovali Raspbery Pi modulmi a bolo možné ich ovládať pomocou AR aplikácie kreslením jazdnej dráhy. Aplikácia pozostávala z vopred načítanej scény, ktorú bolo možné vložiť na akýkoľvek rovný povrch. Po načítaní scény bolo možné ovládať jednotlivé roboty pomocou kreslenia dráhy na tablete. Nakoľko šlo o jednoduchšie Raspbery Pi moduly, komunikácia prebiehala cez TCP/IP protokol.

Práca [5] pomocou mobilného zariadenia s AR aplikáciou umožňuje zobrazovať hodnoty zo snímačov v zariadení. Na základe zosnímaných hodnôt je možné zobraziť poruchu v reálnom zariadení. Aplikácia tiež umožňuje záznam streamovať aj do iných zariadení s AR aplikáciou či webového prehliadača.

Za zmienku stojí aj patent [6] zo spoločnosti Meta, kde je uvedené, že AR aplikácia pre AR headset pristupuje ku knižnici používateľských rozhraní pre skupiny zariadení. Po rozpoznaní objektu v priestore jednoducho načíta rozhranie pre daný objekt.

Ďalšou prácou, ktorou sme sa inšpirovali, je dizertačná práca doktoranda z Ústavu automobilovej mechatroniky STU FEI Ericha Starka, Moderné metódy ovládania a monitorovania mechatronických systémov s využitím počítačom generovanej reality, [7]. Táto práca sa vyznačuje spojením rozšírenej reality a cloudových technológií s IoT. AR aplikácia bola vyvinutá v 3D engine Unity pre iOS zariadenie (iPad). Rozpoznávanie objektov bolo zrealizované pomocou 3D mapy vytvorenej v prostredí Wikitude Studio. Po rozpoznaní objektu sa AR aplikácia pripojila na cloudový server InfluxDB ku digitálnej kópii objektu. Z lokálneho servera zároveň stiahla definíciu užívateľského rozhrania, vďaka ktorej zobrazí v 3D engine užívateľské rozhranie. Samotná komunikácia v rámci projektu je založená na MQTT protokole.

V roku 2022 bol vyvinutý projekt Situovaná vizualizácia údajov IIoT do zariadenia MS HoloLens 2, [8] na univerzite aplikovaných vied v St. Pöltene. Projekt sa zaoberá zberom dát z PLC zariadenia priemyselnej pece, ktoré následne odosiela do lokálneho serveru pomocou OPC UA protokolu. Lokálny server je založený na Node Red nástroji a poskytuje vizualizáciu tvorenú v Dashboard knižnici. Lokálny server ďalej vysiela dáta do MS HoloLens 2 zariadenia pomocou protokolu MQTT. Dáta sú následne zobrazované na výrobnom systéme s možnosťou zobrazenia dát v Dashboarde cez webový prehliadač.

V minulom roku bol taktiež publikovaný projekt [9], ktorý sa venuje moderným metódam pre interakciu človeka a stroja. V publikácii je uvedených viacero možností, využitie umelej inteligencie, mobilných zariadení, IIoT a možností Industry 4.0, ale najmä možnosť skenovania reálnych objektov a následne ich transformácie do 3D objektov.

V závere analýzy viacerých štúdii možno stručne zhrnúť pozitíva a negatíva jednotlivých projektov, Tabuľka 1. Na základe toho vieme definovať vedecký prínos našej práce.

Rozšírená realita založená na IIoT pre zhromažďovanie a vizualizáciu továrenských údajov [1]	
Pozitíva: • Spojenie cloudových technológií, rozšírenej reality a lokálneho servera • Využitie štandardov Industry 4.0 v komunikácii	**Negatíva:** • Absencia riadenia zariadenia v AR aplikácii • Absencia MQTT protokolu • Využitie jednoduchšej cloudovej platformy KEPServerEX
Návrh a implementácia platformy na obchodovanie s energiou typu peer-to-peer s otvoreným zdrojom IoT a blockchain pomocou ESP32-S2 [2]	
Pozitíva: • Realizácia komunikácie zariadení pomocou P2P komunikácie a MQTT protokolu • Poskytnutie vzdialenej vizualizácie dát v podobe webovej aplikácie	**Negatíva:** • Absencia priemyselného výrobného procesu • Absencia servera na cloudovej platforme
Kolaboratívny výrobný dizajn: zmiešaná realita a cloudový rámec pre návrh dielov [3]	
Pozitíva: • Možnosť ukladania a načítania 3D modelov v priestore • Rozpoznanie zariadenia na základe tvaru fyzického zariadenia	**Negatíva:** • Absencia riadenia zariadenia v AR aplikácii • Jednoduché užívateľské rozhranie

Príspevok k mobilnej interakcii zmiešanej reality so systémami s viacerými robotmi [4]	
Pozitíva: • Možnosť manipulácie robotov pomocou kreslenia dráh	**Negatíva:** • Potreba zdĺhavého skenovania plochy pre zobrazenie AR • Absencia cloudových technológií • Neposkytuje základné dáta o robotoch (ako napr. rýchlosť, názov zariadenia, poloha)
Používateľsky zameraný vývoj akceptácie strojov nezávislej od miesta pomocou rozšírenej reality [5]	
Pozitíva: • Zobrazenie skrytých častí zariadenia a ich hodnôt • Možnosť streamovania videa	**Negatíva:** • Absencia riadenia zariadenia v AR aplikácii • Jednoduché užívateľské rozhranie • Absencia cloudových technológií
Moderné metódy ovládania a monitorovania mechatronických systémov s využitím počítačom generovanej reality [7]	
Pozitíva: • Spojenie cloudových technológií, rozšírenej reality a lokálneho servera • Rozpoznávanie 3D objektov na základe reálnych objektov • Využitie MQTT protokolu	**Negatíva:** • Nevyužitie priemyselných komunikačných protokolov • Testovanie len na prototypových zariadeniach (nie na priemyselnej sústave) • Obmedzenie len na platformu iOS

Vizualizácia údajov IIoT situovaná do zariadenia MS HoloLens 2 [8]	
Pozitíva: • Rozpoznávanie 3D objektov na základe reálnych objektov • Využitie MQTT protokolu • Využitie MS HoloLens 2 na vizualizáciu dát • Využitie OPC UA protokolu	**Negatíva:** • Zostavenie UI v jednoduchej knižnici Node Redu s obmedzeným zobrazovaním dát • Nevyužitie cloudových technológií • Poskytnutie dát iba na lokálnej sieti

Tabuľka 1 Porovnanie štúdií

Každé z preskúmaných riešení má svoje silnejšie ale aj slabšie stránky oproti ostatným. Cieľom dizertačnej práce je navrhnúť smart pracovisko so zaznamenávaním dát, ich zálohovaním na cloudovom úložisku. S fyzickým systémom pracoviska bude možné interagovať cez aplikáciu v rozšírenej realite. Pre vzdialené sledovanie analytických hodnôt bude slúžiť webová aplikácia.

Na základe výskumu súčasného stavu využitia cloudových technológií a rozšírenej reality v IIoT sme narazili na niekoľko nedostatkov existujúcich riešení, ktoré je možné v rámci dizertačnej práce zlepšiť:

- Komunikácia zariadení bude realizovaná iba pomocou štandardov IIoT
- Dynamické zobrazenie užívateľského prostredia na základe rozpoznaného zariadenia
- Poskytnutie dát pre vzdialené zobrazenie pomocou internetového prehliadača

Zo získaných vedomostí sme ďalej mohli určiť požiadavky na náš systém, ktoré sme rozdelili na funkcionálne a nefunkcionálne požiadavky:

Funkcionálne požiadavky:

- Komunikácia zariadenia so zmiešanou realitou a lokálnym serverom založená na MQTT protokole,
- Možnosť komunikácie lokálneho servera s výrobným modelom založenej na OPC UA protokole,
- Vývoj aplikácie pre zmiešanú realitu v prostredí Unity,
- Využitie MS HoloLens 2 headset zariadenia pre zobrazenie zmiešanej reality,
- Dynamické zobrazenie vizualizácie jednotlivých výrobných modelov založené na definičných schémach,
- Rozoznanie výrobného modelu na základe jeho tvaru a rozmerov,
- Intuitívne vizualizovanie a riadenie všetkých dát výrobných modelov v zmiešanej realite,
- Poskytnutie dát pre vzdialenú vizualizáciu pomocou webovej aplikácie,
- Poskytnutie vizualizácie dát a riadenia výrobných modelov pomocou dotykových obrazoviek.

Nefunkcionálne požiadavky:

- Definičné schémy uchovávané vo formáte JSON,
- Využitie Azure cloudovej platformy pre správu serverov.

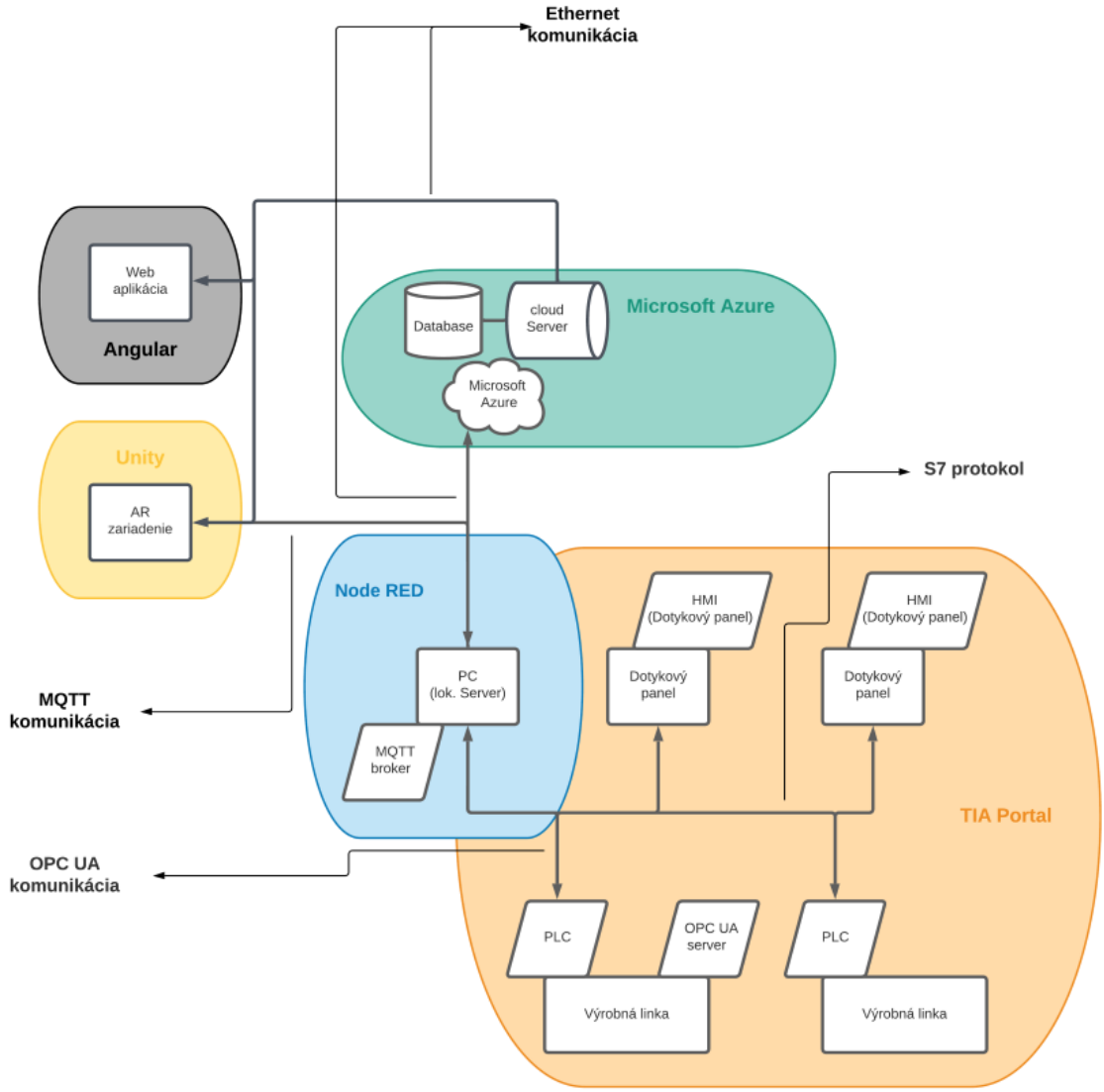
2 Ciele dizertačnej práce

Nachádzame sa v čase implementácie moderných informačno-komunikačných a digitálnych technológií spájaných so 4. priemyselnou revolúciou do priemyselných výrob a služieb. Koncept Industry 4.0 vyžaduje spracovanie veľkého objemu dát v reálnom čase pre možnosť analýzy, poskytnutia dát aplikáciám a vizualizácie výroby. Vďaka väčším hardvérovým možnostiam, rýchlejšiemu prenosu dát a zložitejším algoritmom sa nám sprístupnilo modernejšie a intuitívnejšie vizualizovanie výroby v podobe rozšírenej reality a digitálnych dvojčiat. Na základe analýzy aktuálneho stavu možno stanoviť ciele dizertačnej práce:

1. Analýza moderných postupov monitorovania a riadenia mechatronických systémov s využitím pokročilých priemyselných komunikačných technológií so zameraním na rozšírenú realitu a dynamického načítania používateľského rozhrania.
2. Návrh originálnej platformy, ktorá integruje pokročilé digitálne technológie, moderné metódy monitorovania a riadenia mechatronických systémov s využitím rozšírenej reality.
3. Implementácia navrhnutej platformy pre monitorovanie a riadenie udalostných systémov s využitím rozšírenej reality, priemyselných komunikačných technológií a overenie nových prístupov s využitím dynamického načítavania používateľského rozhrania.
4. Zhodnotenie prínosu navrhnutej platformy a nových metód monitorovania a riadenia mechatronických systémov a možností ich využitia v priemyselnej praxi v súlade s konceptom Industry 4.0.

3 Návrh integračnej platformy pre digitálne technológie

Návrh platformy vychádza zo štúdia dostupných technológií, analýzy súčasného stavu a existujúcich riešení. Platforma využíva široké spektrum technológií a preto je potrebné jej návrh rozdeliť do viacerých celkov. Nakoľko všetky časti platformy, vytvorenej v rámci dizertačnej práce, interagujú s ostatnými časťami, môžeme vytvoriť štruktúru projektu tejto platformy, ktorá je na Obr. 1.

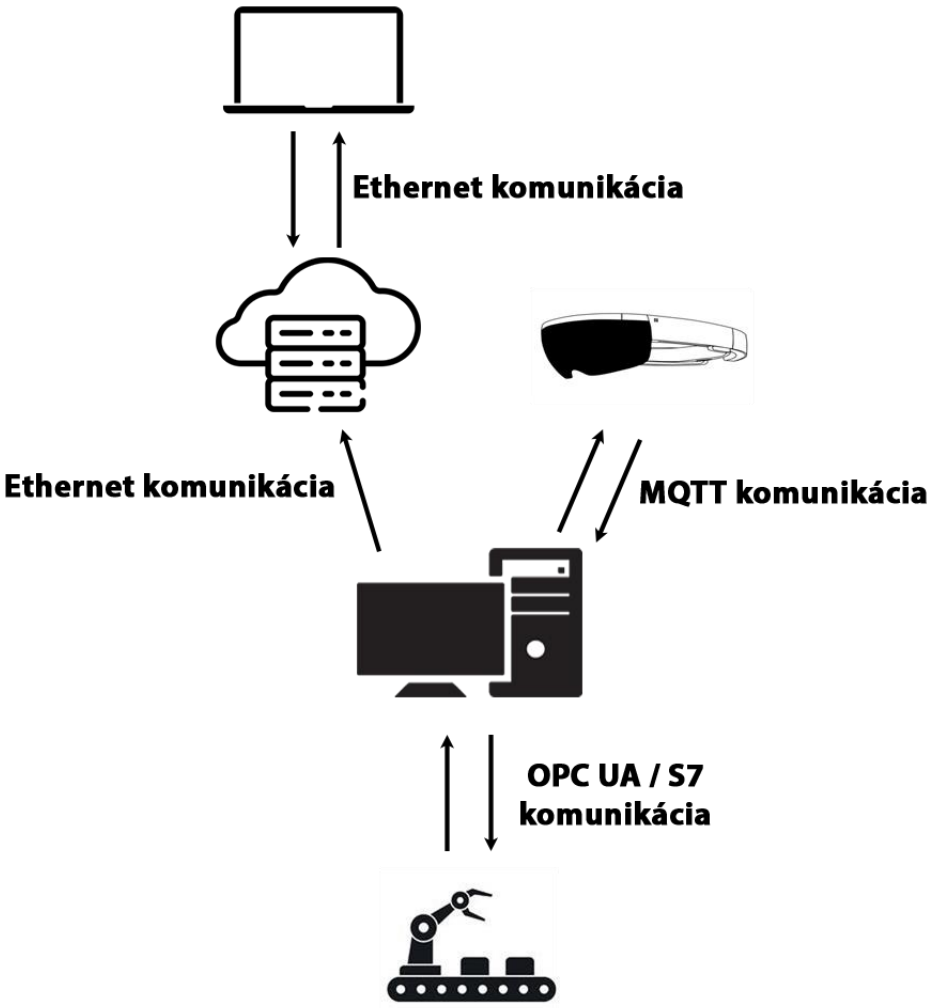


Obr. 1 Štruktúra projektu

V štruktúre projektu na Obr. 1 môžeme vidieť farebne rozdelené časti podľa použitých nástrojov. Prvou z nich je automatizácia výrobných liniek a tvorba HMI

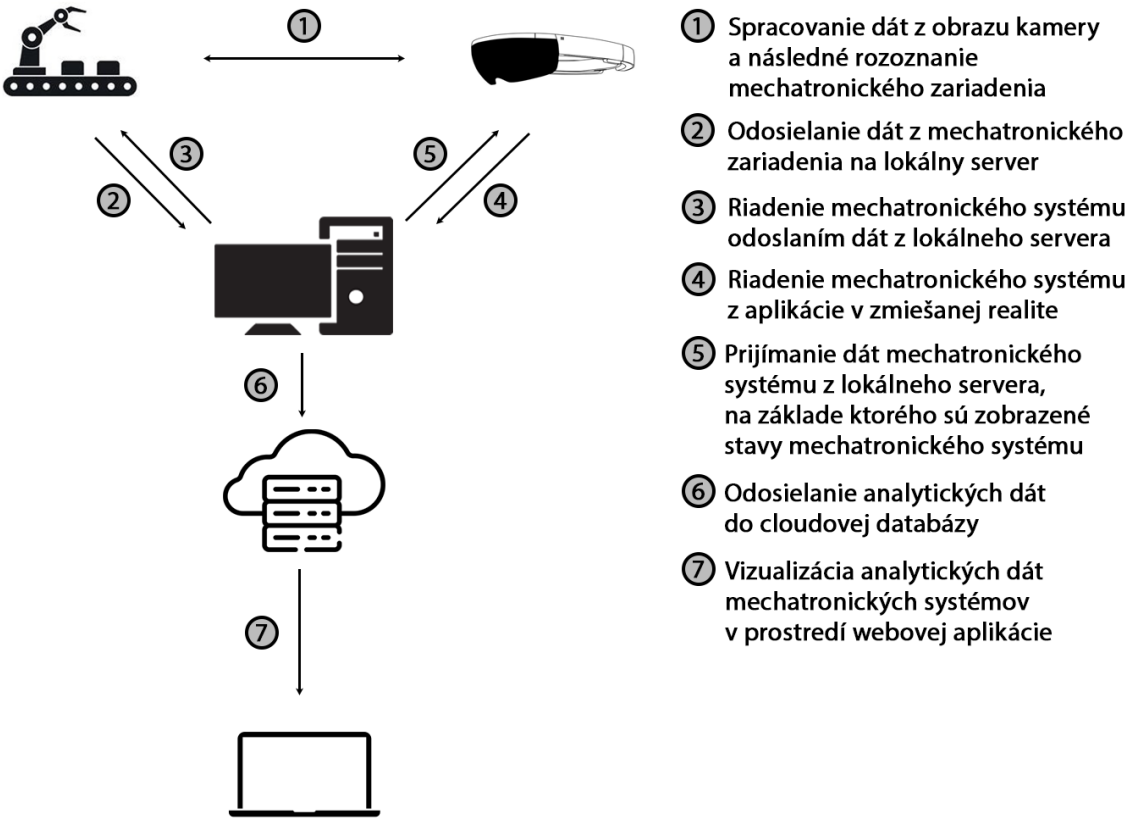
v prostredí TIA Portal (v štruktúre ako oranžová plocha). Druhou je tvorba lokálneho servera pre spracovanie dát z PLC zariadení a následné poskytnutie týchto dát ostatným častiam cez ďalšie komunikačné rozhranie (v štruktúre ako modrá plocha). Ďalšou časťou je tvorba cloudového servera, na ktorom sa ukladajú analytické dáta, ktoré sú ďalej poskytnuté klientskemu vzdialenému zariadeniu (v štruktúre ako tyrkysová plocha). Ďalšou časťou je frontend aplikácia tvorená v prostredí Angular pre vzdialenú vizualizáciu analytických dát (v štruktúre ako čierna plocha). Poslednou časťou je Unity aplikácia pre zariadenie MS HoloLens 2 (v štruktúre ako žltá plocha).

V projekte sa nachádza niekoľko druhov fyzických zariadení, na ktorých sa nachádzajú nahraté aplikácie z predošlej časti textu. Rovnako, ako bola opísaná komunikácia aplikácií, tak aj jednotlivé zariadenia medzi sebou komunikujú. Na Obr. 2 je znázornená komunikácia, smer odosielania informácií a interakcia jednotlivých zariadení.



Obr. 2 Komunikácia zariadení

Spôsob prenosu a spracovania dát je na Obr. 3. Zariadenie na zobrazenie zmiešanej reality rozpoznáva reálne objekty pomocou kamery. Po ich rozpoznaní sa pripojí ku mechatronickému zariadeniu a vyžiada si inicializačné dáta. Lokálny server slúži ako prostredník medzi mechatronickým systémom a zariadením na zobrazenie zmiešanej reality, pričom s oboma zariadeniami komunikuje pomocou rozdielnych komunikačných protokolov. Ďalej odosiela analytické dáta do cloudovej databázy, ktorá poskytuje uložené dáta pre vzdialenú vizualizáciu dát vo webovom prehliadači.



Obr. 3 Riadenie a vizualizácia mechatronických systémov

Jednotlivé časti štruktúry projektu, ich aplikácie, fyzické zariadenia a tvorbu si opíšeme detailne v nasledujúcej časti.

3.1 Automatizácia výrobných procesov

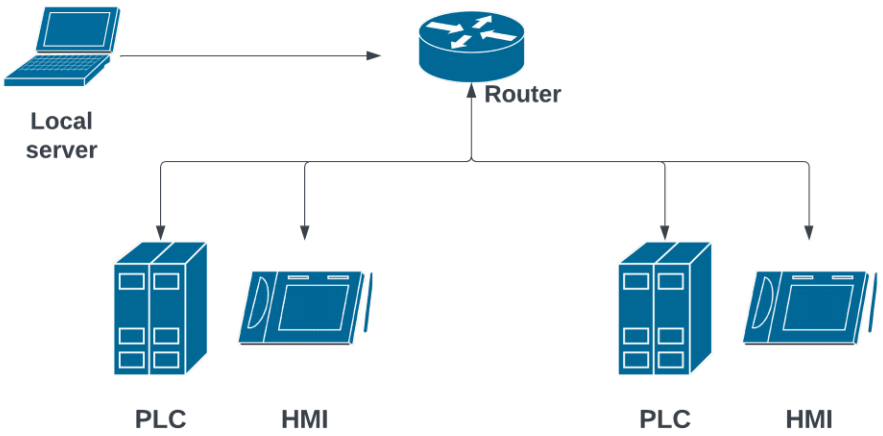
Dôležitou súčasťou projektu bol vhodný výber laboratórnych výrobných modelov a PLC zariadení. PLC zariadenie sme vybrali od renomovaného výrobcu s bohatou profesionálnou históriou Siemens. Ten poskytuje na svojich zariadeniach komunikáciu s použitím vlastného protokolu S7 a na vyšších modeloch (S7-1500 a niektoré S7-1200) komunikáciu s využitím OPC UA protokolu.

Pri návrhu platformy (systému) sme postupovali v súlade s princípmi Industry 4.0. V tomto prípade požadujeme, aby navrhovaný systém bol modulárny. Preto sme vybrali komunikáciu s využitím OPC UA protokolu, ktorý je kompatibilný aj so zariadeniami od iných výrobcov. V práci budú využité 2 laboratórne modely výrobných systémov a preto sme sa rozhodli v záujme overenia modularity použiť oba protokoly, pričom jeden model bude využívať S7 protokol a druhý OPC UA protokol.

Ako už bolo spomenuté, oba laboratórne modely sú automatizované pomocou Siemens PLC zariadení a každé z nich obsahuje dotykový displej Siemens HMI KTP 700 Basic pre ovládanie modelu a zobrazenie jeho dát. Toto zariadenie môžeme vidieť na Obr. 4.

Obr. 4 Siemens HMI KTP 700 Basic

Všetky zariadenia komunikujú na úrovni lokálnej siete bez prístupu na internet. Zariadenia dokážu medzi sebou komunikovať, ale nakoľko ide o 2 samostatné systémy, komunikácia bude prebiehať medzi PLC zariadením, HMI zariadením daného laboratórneho modelu a lokálnym serverom. Zapojenie jednotlivých zariadení v sieti je na Obr. 5.

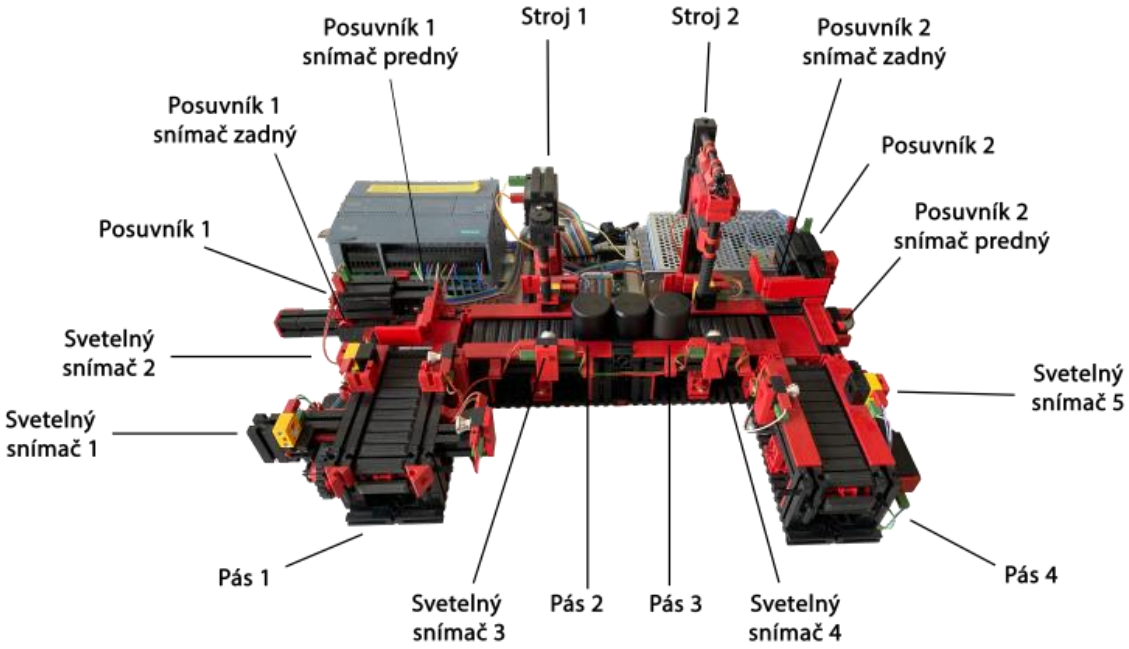


Obr. 5 Zapojenie zariadení v lokálnej sieti

V navrhovanom riešení využívame ako výrobné modely vzdelávacie modely od značky Fischertechnik. Jednotlivé laboratórne výrobné modely, ich vizualizáciu a funkcionalitu opíšeme detailne v nasledujúcej časti.

Model výrobnej linky

Prvým laboratórnym výrobným modelom je model výrobnej linky v tvare písmena U (ďalej ako model 1). Model pozostáva z nasledujúcich výkonových členov: 4 pásov, 2 obrábacích strojov, 2 posuvníkov, pričom oba posuvníky umožňujú pohyb vpred aj vzad. Na detekciu polohy výrobku vo výrobnom procese je k dispozícii 5 svetelných snímačov. Na detekciu polôh posuvníkov je model vybavený 4 koncovými snímačmi (2 zadné a 2 predné). Všetky z vymenovaných súčastí laboratórneho výrobného modelu sú vyznačené na Obr. 6.



Obr. 6 Model výrobnej linky (model 1)

Pre riadenie tohto modelu bolo využité menšie PLC zariadenie, model S7 1215C DC/DC/DC. PLC zariadenie tohto typu obsahuje 14 digitálnych vstupov, 10 digitálnych výstupov a 2 analógové výstupy. Pre potreby automatizácie tejto sústavy bolo potrebných 9 digitálnych vstupov a 10 digitálnych výstupov.

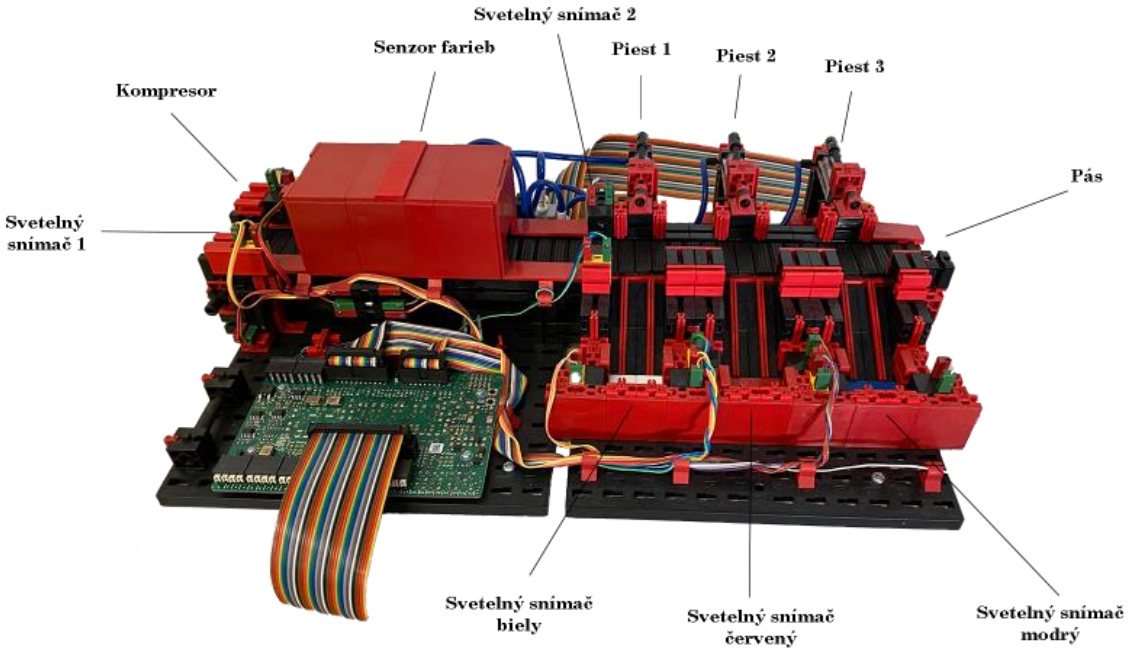
Obr. 7 PLC zariadenie, model S7 1215C DC/DC/DC

Pre laboratórny výrobný model sme zostavili algoritmus v prostredí TIA Portal, kde sme navrhli opracovanie výrobku a jeho presun cez celý výrobný proces. Ďalej algoritmus obsahoval nasledujúce ošetrenia a funkcionality:

- ošetrenie kolízie výrobkov na výrobnej linke pri viacerých výrobkoch súčasne sa nachádzajúcich na výrobnom modeli,
- detekcia straty výrobku z pásov, ak sa do 5 sekúnd nevykoná ďalší krok riadiaceho algoritmu, zapne sa alarm na danom páse,
- napočítavanie motohodín pre všetky pásy a stroje,
- napočítavanie počtu kusov výrobkov na výrobnej linke a počtu hotových výrobkov,
- manuálne resetovanie oboch počtov výrobkov,
- možnosť prevádzky výrobnej linky v manuálnom a automatickom režime,
- funkcia štart, ošetrenie pre rýchle vypnutie linky v prípade nečakanej kolízie.

Model triediacej linky s detekciou farby

Ďalším laboratórnym výrobným modelom je model triediacej linky s detekciou farby (ďalej ako model 2). Tento systém poskytuje uskladnenie výrobku na základe rozpoznania farby. Pozostáva z aktívnych členov ako pás, kompresor a 3 piesty. Rozpoznanie farby výrobku vykonáva senzor farieb a informácie o polohe výrobku poskytuje 5 svetelných senzorov. Všetky vymenované diely laboratórneho modelu sú znázornené na Obr. 8.



Obr. 8 Model triediacej linky s detekciou farby (model 2)

Pre model 2 sme využili výkonné PLC zariadenie, model S7-1516-3 PN/DP, ktoré poskytuje vyšší výkon a omnoho viac funkcií ako PLC v predošlom prípade pre model 1. Samotné PLC zariadenia S7-15xx nedisponujú vstupmi a výstupmi. Preto bolo potrebné priradiť PLC zariadeniu vstupno-výstupné moduly. Na Obr. 9 môžeme vidieť nami navrhnutý a zostavený riadiaci systém, kde sa nachádza PLC zariadenie zapojené v kombinácii so 4 modulmi a 1 priemyselným zdrojom. Vstupno-výstupné moduly nám poskytujú 32 digitálnych vstupov, 32 digitálnych výstupov, 8 analógových vstupov a 4 analógové výstupy.

Obr. 9 PLC zariadenie, model S7 1516-3 PN/DP s modulmi

Posledným krokom bolo zostavenie algoritmu a vizualizácie pre model 2. Model je navrhnutý na 3 typy výrobkov, výrobok červenej, bielej a modrej farby. Po vložení výrobku na linku sa výrobok presunie na pozíciu farebného senzora, kde sa zosníma farba výrobku v priebehu 0,5 sekundy.

Algoritmus obsahuje aj nasledujúce funkcionality a ošetrenia:

- detekcia kolízie na plný úložný priestor, v prípade plného stavu spustí alarm,
- funkcia štart, ošetrenie pre rýchle vypnutie linky v prípade nečakanej kolízie,
- viacero režimov ovládania – manuálny, automatický a semi-automatický režim,
- napočítavanie motohodín pre všetky akčné členy,
- napočítavanie vytriedených výrobkov podľa rozpoznanej farby a celkový počet výrobkov.

3.2 Vývoj lokálneho servera

Lokálny server v našom prípade potrebuje spracovať údaje zo všetkých PLC zariadení v systéme, uložiť ich do databázy a poskytnúť dáta prenosným zariadeniam. Preto sme kládli veľký dôraz na možnosť komunikácie s využitím viacerých protokolov a na potrebný výkon pre spracovanie dát.

Pri našich požiadavkách na komunikáciu (Obr. 2) bolo ideálne využitie nástroja Node Red na tvorbu lokálneho systému. Node Red je založený na kombinácii Javascriptu a modulárneho programovania (prepájania logických buniek). Node Red sám o sebe neobsahuje všetky potrebné nástroje, či už na komunikáciu, tvorbu grafiky alebo rôznych logických funkcií. Preto bola potrebná implementácia knižníc tretích strán, ďalej uvádzame, ktoré knižnice sme využili:

- **node-red-contrib-opcua** – knižnica na vytvorenie OPC UA servera a umožnenie jednoduchej komunikácie na tomto protokole,
- **plcindustry** – knižnica pre umožnenie komunikácie so Siemens zariadeniami založenej na S7 protokole,
- **node-red-contrib-mssql** – knižnica pre komunikáciu s MS SQL databázami,

- **node-red-contrib-aedes** – knižnica na vytvorenie MQTT brokera a komunikáciu založenú na MQTT protokole.

Z použitých knižníc vyplýva, že lokálny server zastával 4 funkcie, konkrétne 4 rôzne komunikácie založené na rôznych protokoloch. Dalo by sa povedať, že v našom prípade je lokálny server srdce projektu, ktoré prepája všetky celky medzi sebou (Obr. 1).

3.3 Vývoj cloudového servera

Ďalšou časťou bolo vytvorenie a nakonfigurovanie cloudového servera. V našom prípade boli na cloudový server kladené tri funkcionálne požiadavky:

- umožnenie ukladania dát,
- poskytnutie dát aplikáciám a
- poskytnutie webovej aplikácie pre verejnosť.

Na výber sme mali z viacerých cloudových platforiem, ale podľa analýzy sme sa rozhodli využiť platformu MS Azure. MS Azure poskytuje veľké množstvo služieb, z ktorých sme vybrali nasledujúce:

- *SQL server + SQL database,*
- *App Service,*
- *Static Web App.*

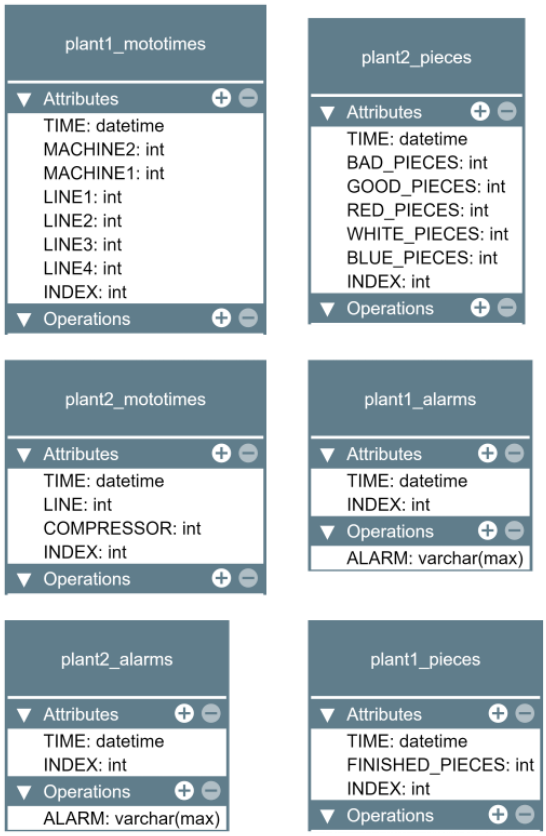
Ďalej detailne opíšeme všetky služby, ich tvorbu, konfiguráciu, použité nástroje a aplikácie, ktoré sme nahrali na cloudový server.

SQL server + SQL database

Najprv bolo potrebné vytvoriť databázu, do ktorej sa budú ukladať dáta, ktoré budú poskytnuté pre vzdialené zobrazovanie, ďalej budeme tieto dáta nazývať analytickými dátami. Vytvorili sme teda SQL server, ktorý sme nazvali *plscwithlove,* tento server sme uložili na dátové centrum (v Nemecku) a spolu so serverom sme zároveň vytvorili aj MySQL databázu o veľkosti 2GB. Ďalej sme nastavili SQL server ako privátny server, ktorému bolo potrebné pridať IP adresy počítača, na ktorom bol spustený lokálny server a IP adresu serveru s backendovou aplikáciou. Ako posledné sme nastavili bezpečnostné pravidlá na autentifikáciu pomocou prihlasovacích údajov.

Prihlasovacie údaje sme zvolili podľa všeobecných pokynov, pričom heslo bolo zložené z viac ako 15 znakov a obsahovalo číslice, malé znaky, veľké znaky a symboly.

Keďže sme mali nakonfigurovaný server, mohli sme vytvoriť štruktúru databázy. Každá z výrobných liniek obsahovala 3 skupiny analytických dát, boli nimi motohodiny, počet vyrobených kusov a alarmy. Podľa toho sme prispôsobili aj štruktúru databázy (Obr. 10). Vytvorili sme 6 tabuliek, v ktorých každý riadok zodpovedal jednému zápisu. Každá z tabuliek obsahovala primárny kľúč *index*, ktorý bol doplnený automaticky. Ostatné hodnoty zodpovedali typom premenných, ktoré sme zapisovali do tabuľky.



Obr. 10 UML diagram cloudovej databázy

App Service

Ďalším krokom bolo zabezpečiť prístup k dátam pomocou backend aplikácie. Nahratie aplikácie nám umožnil MS Azure prostredníctvom služby *App Service*. Tu bolo potrebné nakonfigurovať prostredie servera, zvolili sme pod akým operačným systémom má server pracovať, verziu Java platformy, jeho lokáciu a najmä, aký projekt má byť naň nahraný. Umiestnenie sme zvolili v dátovej centrále v Kanade, nakoľko v Európe bola táto služba spoplatnená. Operačný systém sme zvolili Linux, s ktorým sú

Java aplikácie kompatibilné a verziu Java platformy *Java 17 SE*. Ako zdroj projektu sme vybrali repozitár na *Github* platforme, kde sme neskôr nahrali projekt. V tomto prípade nahratie projektu prebieha automaticky po nahratí novej verzie projektu do repozitára.

Backendovú aplikáciu bolo potrebné vytvoriť a otestovať v prvotnej fáze na lokálnom prostredí. Na vývoj sme využili nástroj *InteliJ*, použili sme rovnakú verziu Java platformy ako na serveri a vytvorili sme nový projekt v *Springboot* frameworku verzie 3.0.4. Ako prvé sme prepojili aplikáciu s cloudovou databázou MySQL implementovaním *mssql-jdbc* knižnice a zadaním prihlasovacích údajov. Pre prácu s dátami v databáze sme využili knižnicu *springboot-hibernate*. Týmto sme vytvorili základ pre náš projekt a ďalej sme mohli pracovať na vytvorení koncových bodov pre prácu s dátami.

Projekt bol založený na architektúre REST API v *Springboot* frameworku. Pre každú z tabuliek bolo potrebné spraviť model so všetkými atribútmi. Pomocou knižnice *lombok* sme mohli jednoducho pridať metódy *getter* a *setter* pre všetky modely pomocou anotácií.

Ďalej sme vytvorili repozitár, ktorý sa priamo dotazoval na databázu. Rovnako ako modely, aj repozitáre bolo potrebné vytvoriť zvlášť pre každý model. Nakoľko repozitár vo frameworku Springboot slúži ako rozširujúca trieda triedy *JpaRepository*, nebolo potrebné pridať nové funkcie. Všetky potrebné funkcie už obsahovala trieda *JpaRepository*. Ďalej sme vytvorili servisy pre jednotlivé výrobné linky, ktoré nám poskytovali dáta získané z repozitárov pre triedy kontrolerov. Všeobecne je architektúra navrhnutá tak, že kontroler konfiguruje cestu a typ, na ktorú sa aplikácia dotazuje. Spracovanie dát prebehne v servise a repozitár poskytne dáta daného modelu. V kontroleri, ako už bolo spomenuté, sme nastavili mapovanie koncových bodov a http metódu, na ktorú reaguje. Keďže sa poskytovali dáta z backend aplikácie, tak išlo o http metódu typu GET.

Backend aplikáciu sme potom otestovali pomocou nástroja *Postman*, kde po zavolaní koncového bodu nám backend aplikácia vrátila dáta z databázy.

Static web app

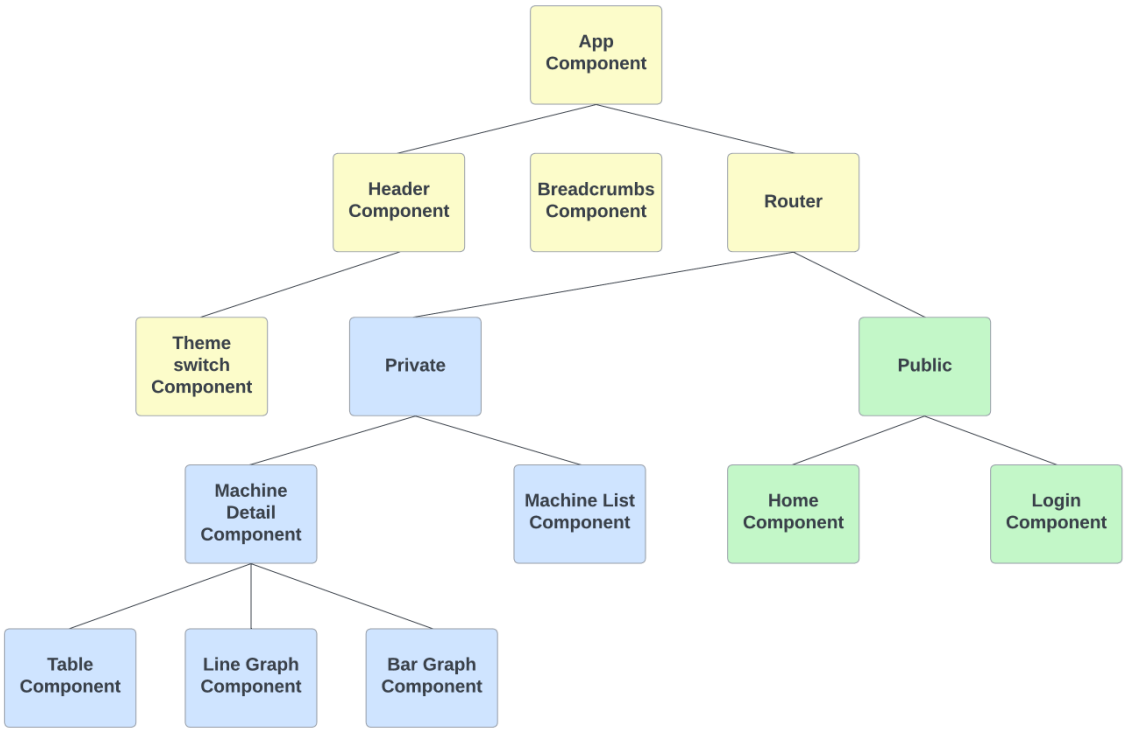
Rovnako ako v predošlej službe, aj pri tejto službe bolo potrebné založiť server a nastaviť zdroj, z ktorého sa nahrá frontend aplikácia. Postup bol rovnaký ako v predošlom prípade, založili sme nový repozitár na platforme *GitHub* a ten sme napojili na server. Po vytvorení servera sme sa mohli pustiť do vývoja frontend aplikácie.

Frontend aplikáciu sme sa rozhodli robiť v JS frameworku Angular, verzia 15.2.3. V súčasnosti je veľmi silným nástrojom pre vývoj vzhľadu knižnica *NgPrime*, ktorú sme v našom projekte použili pre definovanie vzhľadu elementov na webovej stránke.

Pred vývojom aplikácie bolo potrebné zostaviť diagram prípadov použitia, podľa ktorého bola aplikácia zostavená. Na Obr. 11 môžeme vidieť navrhnutý diagram, kde sa nachádzajú 2 druhy používateľov, sú nimi prihlásený a neprihlásený používateľ. Nakoľko webová aplikácia je verejne dostupná na webovej adrese: *https://wonderful-moss-094cedb03.2.azurestaticapps.net/*, bolo potrebné uvažovať aj nad neautorizovaným prístupom. Preto neprihlásený používateľ môže pristupovať iba ku hlavnej stránke a prihláseniu. Aplikácia okrem prihlásenia bude obsahovať aj funkcie ako prepnutie svetlého/tmavého režimu, zobrazenie všetkých liniek, zobrazenie analytických dát liniek v tabuľke a v grafickom rozhraní.

Obr. 11 Diagram prípadov použitia

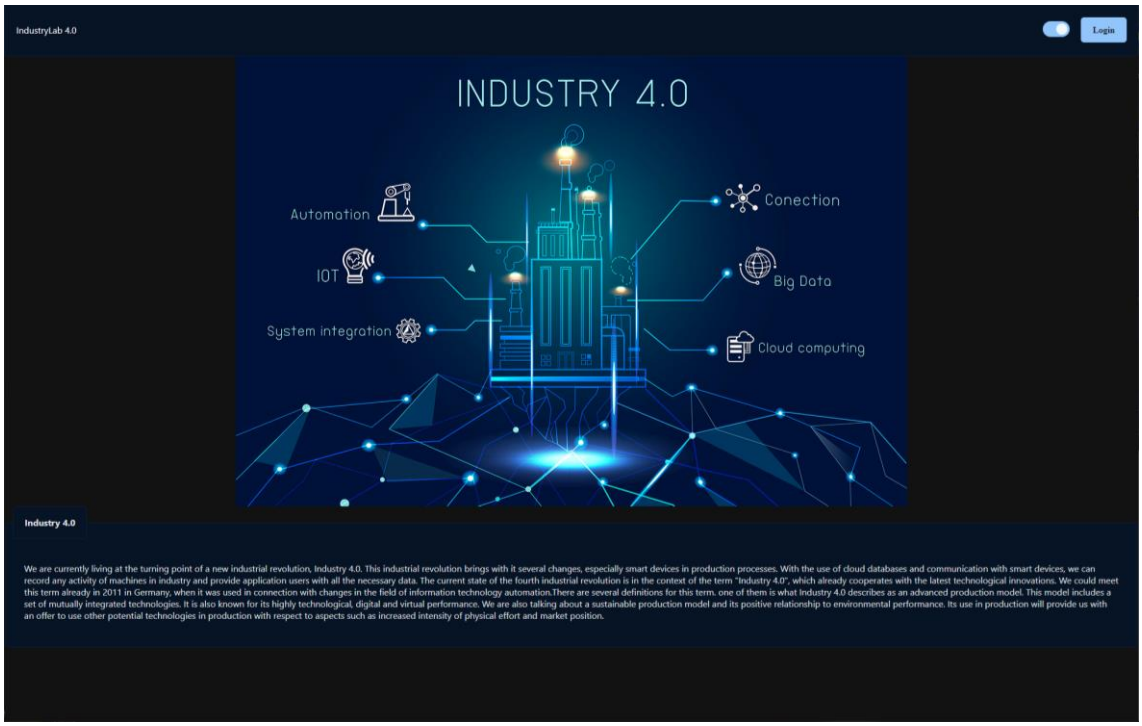
Aplikácia bola navrhnutá tak, aby spĺňala aktuálne štandardy webových aplikácií. Aby bola dostatočne prehľadná a intuitívna, minimalistická a bez zbytočného načítavania. Aplikácia je navrhnutá tak, aby bola prispôsobená na mobilné zariadenia rôznej veľkosti ako aj na obrazovky osobných počítačov. Aplikácia je zostavená pomocou JS frameworku Angular, preto pri zmene obsahu sa namiesto načítania celej aplikácie zo servera dynamicky aktualizuje iba obsah, ktorý sa mení. V aplikácii sme využili knižnicu *angular-router*, ktorá využíva koncepciu trás na riadenie navigácie v rámci aplikácie. Trasy sú definované ako kombinácia cesty (zvyčajne segmentu URL) a komponentu, ktorý by sa mal zobraziť pri navigovaní na cestu. Taktiež sme využili techniku *lazy loading*, ktorá umožňuje načítavať moduly Angularu podľa potreby, namiesto načítania celého obsahu pri spustení aplikácie. Toto môže pomôcť zlepšiť počiatočný čas načítania a celkový výkon aplikácie, najmä v prípadoch, keď má veľké množstvo komponentov, služieb alebo iných zdrojov. V našej aplikácii sa načíta počiatočný modul *app-component*, ktorý obsahuje komponent hlavnej stránky a komponent prihlásenia. Po prihlásení používateľa sa načíta privátny modul s komponentami privátnej časti, Obr. 12.



Obr. 12 Diagram komponentov vo frontend aplikácii

Ako už bolo spomenuté, aplikáciu možno rozdeliť do troch skupín. Sú nimi základná časť aplikácie, ktorá sa nachádza v celej aplikácii, verejná časť a privátna časť. Podľa toho je možné rozdeliť komponenty do diagramu. Na Obr. 12 môžeme vidieť diagram komponentov, kde jednotlivé časti sú farebne odlíšené. Základná časť (žltá farba) sa skladá z hlavičky, komponentu prepínača farebného režimu, navigácie (*breadcrumbs*) a smerovania. Verejná časť (zelená farba) je prístupná neprihlásenému používateľovi, do ktorej spadajú komponenty ako úvodná stránka a prihlásenie. Jednotlivé podstránky opíšeme detailne ďalej. Privátna časť (modrá farba) je prístupná až po prihlásení užívateľa a zahŕňa vizualizáciu dát laboratórnych výrobných modelov.

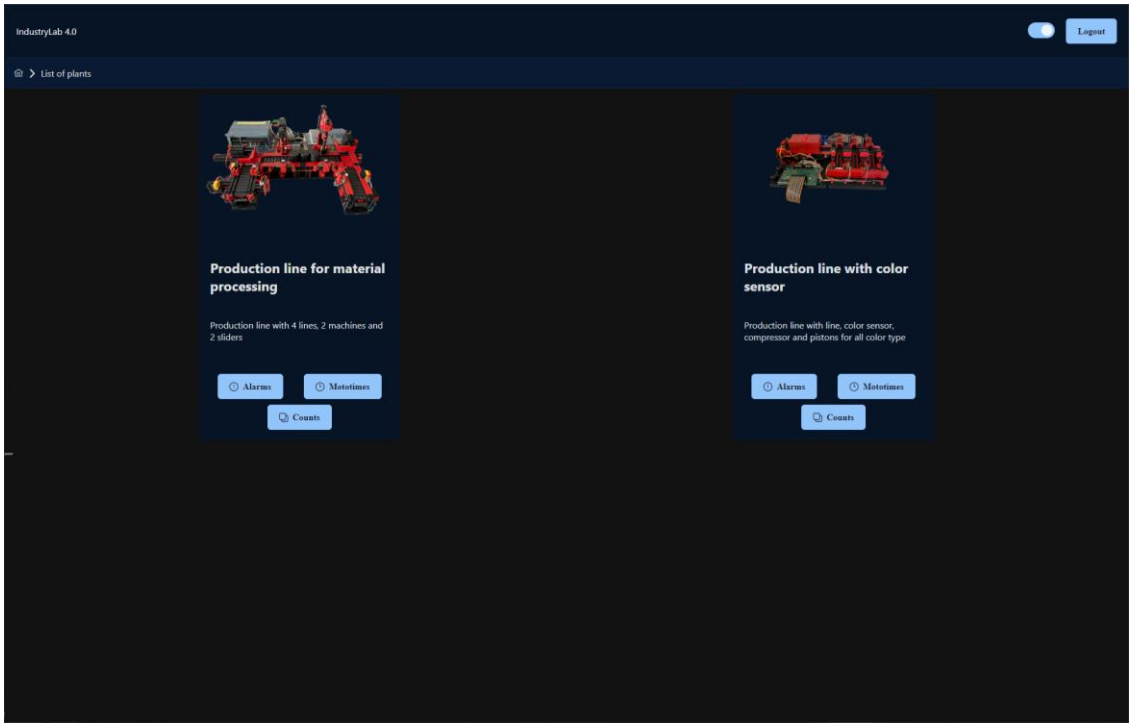
Po prvotnom načítaní webovej aplikácie sa dostaneme na úvodnú stránku (Obr. 13), tu sa nachádza hlavička, kde môžeme prepnúť stav farebného režimu alebo sa prihlásiť. Po kliknutí na tlačidlo *Login* nás aplikácia presmeruje na prihlasovací formulár. Nakoľko sa naša práca zaoberá implementáciou Industry 4.0 do priemyslu, na úvodnej stránke sa nachádza aj obrázok so sprievodným textom o Industry 4.0.



Obr. 13 Úvodná stránka frontend aplikácie

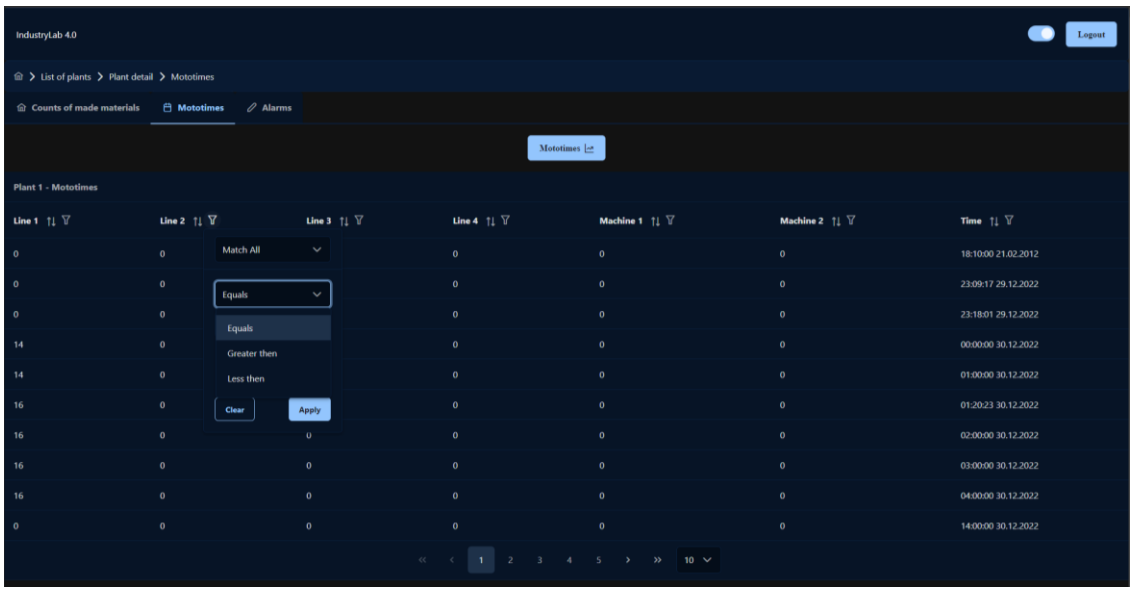
Po prihlásení sa užívateľ dostane do zoznamu výrobných liniek, ktorý je zobrazený na Obr. 14. Táto podstránka ponúka užívateľovi rýchly prehľad všetkých výrobných liniek v systéme. Každá z liniek poskytuje obrázok výrobnej linky, jeho

názov a stručný popis. Taktiež obsahuje tlačidlá, ktoré užívateľa presmerujú na zobrazenie jednotlivých dát, ktoré daná výrobná linka obsahuje.



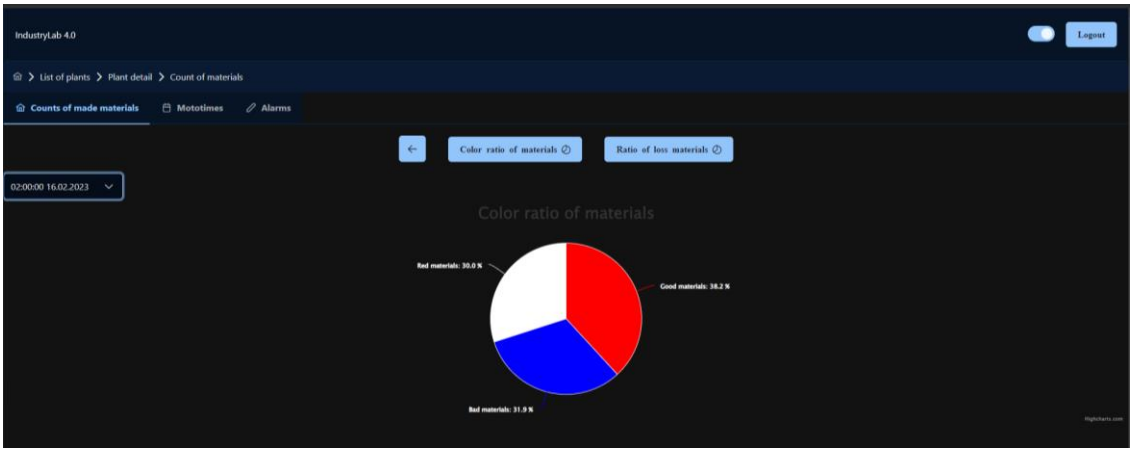
Obr. 14 Zoznam výrobných liniek

Zobrazenie dát výrobných liniek je reprezentované v tabuľkách (Obr. 15). Jednotlivé stĺpce je možné zoraďovať od najmenšieho po najväčšie (numericky, časovo alebo abecedne). Taktiež je umožnené triedenie dát podľa zadanej hodnoty. Tabuľky umožňujú aj stránkovanie a možnosť voľby počtu zobrazovaných riadkov. Pri každej premennej z výrobnej linky bolo možné zmeniť typ dát v podmenu. Každý typ dát umožňuje aj grafické zobrazenie dát po konfigurácii, ktorú si opíšeme neskôr. Ak dáta poskytujú grafické zobrazenie, nad tabuľkou je zobrazené tlačidlo s príslušnou ikonou. Grafická prezentácia dát je možná dvoma spôsobmi, čiarovým grafom (Obr. 16) a koláčovým grafom (Obr. 17). Čiarový graf umožňoval zobrazenie dát v časovej rovine s možnosťou priblíženia. Koláčový graf zobrazoval konkrétne dáta v jednom zápise a umožňoval výber daného zápisu pomocou html elementu *select*. Každá zmena dát v grafoch je vykonaná pomocou animácie.



Obr. 15 Tabuľka s dátami

Obr. 16 Čiarový graf frontend aplikácie



Obr. 17 Koláčový graf frontend aplikácie

V bežnej praxi je zaužívané, že sa vo webových aplikáciách stiahnu dáta a z nich sa vygeneruje obsah, ako napríklad dáta v tabuľke. Ale v kóde HTML tabuľka očakáva

presný typ dát a tak sa musia pre každé dáta robiť zvlášť komponenty. V našom prípade existuje jeden komponent pre zoznam všetkých výrobných liniek, jeden komponent pre tabuľku dát, jeden komponent pre čiarový graf a jeden pre koláčový graf. Ako sa jednotlivé komponenty vygenerujú, záleží od dát, ktoré zadáme v konfiguračnom súbore. Konfiguračné dáta pre model výrobnej linky sú zostavené vo forme objektu. V celom konfiguračnom súbore sa nachádza pole viacerých dátových objektov modelov, z ktorého sa vygenerujú karty v zozname. Kľúč *machine* obsahuje názov karty, kľúč *image* obsahuje cestu ku obrázku a kľúč *text* obsahuje sprievodný text zobrazený pod názvom.

Kľúč *detail* obsahuje informácie o tom, či daný model obsahuje dáta a aké. V našom prípade modely mohli obsahovať 3 druhy dát, boli nimi alarmy, motohodiny a počet vyrobených kusov. Preto *detail* obsahoval kľúče *alarms, mototimes* a *counts*. Každý z detailov obsahoval rovnaké dáta taktiež vo forme objektu. V tomto objekte kľúč *show* určoval, či model obsahuje tieto dáta. Ak ich neobsahoval, v karte modelu sa dané tlačidlo nezobrazovalo a ani v podmenu detailu modelu nebolo možné aplikáciu prekliknúť na dané dáta. Kľúč *title* obsahoval názov tabuľky a kľúč *path* cestu ku koncovému bodu, z ktorého sa dáta žiadali. V kľúči *header* sa nachádzali v poli všetky stĺpce, ktoré sa zobrazovali v tabuľke, každý zo stĺpcov obsahoval nastavenia ako jeho názov, či je možné ho triediť a o aký typ dát išlo. Typ dát určoval, aké filtrovanie bude použité na danom stĺpci. Ďalej kľúč *data* obsahoval všetky dáta, ktoré sa mali zobraziť v tabuľke. Jednotlivé dáta obsahovali ich názov v dátach, ktoré boli získané z backendu a akým spôsobom sa majú transformovať. V našom prípade sme transformovali iba dátumy na formát *dd.mm.yyyy*.

Posledným kľúčom bol kľúč *graphs*, ktorý určoval koľko a aké grafy pre daný typ dát bude poskytovaný. Konfiguračné dáta pre grafy boli vo formáte poľa, podľa ktorého sa vygenerovali tlačidla grafov nad tabuľkou. Jednotlivé grafy reprezentovali objekty, v ktorých kľúč *type* určoval, či sa jedná o čiarový graf alebo koláčový. Kľúč *buttonText* obsahoval text zobrazený v tlačidle nad tabuľkou. Kľúč *show* určoval, či graf bude predvolene zobrazovaný (prioritne pred tabuľkou) pri zobrazení dát. Kľúč *unit* určoval názov x osi a y osi vo formáte reťazca „osX~osY“. Klúč *moreTimes* určoval, či bude zobrazený html element *select* so všetkými časovými údajmi. Kľúč *data* určoval, ktoré dáta budú zobrazené na grafe a v kľúči *color* si môžeme navoliť vlastné farby pre jednotlivé dáta.

Ak necháme iba prázdne pole v kľúči *color*, tak sa farby určia defaultne. Ak necháme kľúč *graphs* taktiež iba prázdne pole, tak dáta nebudú obsahovať žiadne grafické zobrazenie dát.

3.4 Vývoj aplikácie pre zmiešanú realitu

Ďalšou časťou práce bola tvorba aplikácie pre zmiešanú realitu, ktorú sme realizovali v prostredí herného enginu Unity. Pracovali sme konkrétne vo verzii 2020.3.33f1, ktorá je overená na kompatibilitu práce so súborom nástrojov pre vývoj softvéru *MRTK*. Táto časť je jeden z pilierov z hľadiska prínosu dizertačnej práce, nakoľko umožňuje detekciu reálneho výrobného modelu v zmiešanej realite a obsahuje algoritmus pre generické renderovanie vizualizácie jednotlivých výrobných modelov. V rámci vývoja bolo potrebné implementovať do Unity aplikácie nasledovné knižnice a nastavenia:

- nastaviť aplikáciu pre vývoj *MRTK,*
- implementovať *MQTT* knižnicu na komunikáciu s lokálnym serverom,
- implementovať *Vuforia* knižnicu pre rozpoznávanie objektov,
- vytvoriť algoritmus pre vykreslenie používateľského rozhrania.

Ďalej opíšeme postup pri jednotlivých implementáciách funkcionalít do projektu.

MRTK

Po vytvorení projektu bolo ako prvé potrebné nastaviť projekt pre kompatibilnú prácu s *MRTK*. Nakoľko sa balík *MRTK* nenachádza v package manažérovi, implementáciu bolo potrebné vykonať pomocou externej aplikácie *MS Mixed Reality Feature Tool*. Pomocou tejto aplikácie sme mohli importovať balík MRTK do Unity projektu.

MQTT komunikácia

Na implementáciu MQTT komunikácie do Unity projektu sme využili knižnicu *M2MqttUnity*. Tá nám poskytla základné funkcie ako pripojenie a odpojenie sa ku MQTT brokeru, prijímanie dát z danej relácie a vysielanie hodnôt do nami zadaných MQTT relácií. Na týchto funkciách sme položili základy nášho algoritmu pre komunikáciu.

Funkcia *ConnectToMQTTServer* zabezpečovala pripojenie na MQTT brokera. Vstupnými dátami bol reťazec vo formáte *ipAdresa~port*. Na základe prijatých dát funkcia pripojila aplikáciu ku MQTT brokeru. Preddefinovaná hodnota vstupného reťazca bola *192.168.50.60~1883*, čo zabezpečovalo zavolať funkciu bez zadania údajov. Taktiež algoritmus obsahoval funkciu na odpojenie komunikácie, čo zabezpečovala funkcia *DisconnectFromMQTTServer*. Daná funkcia poskytovala taktiež aj odpojenie od všetkých relácií, ktoré boli aktívne v danom čase.

Odosielanie dát nám zabezpečovala funkcia *SendDataToMQTTServerByValue*. V tomto prípade bola vstupná premenná typu *GameObject*. Každé tlačidlo, ktoré interagovalo s niektorou z premenných z PLC zariadenia, totiž v sebe obsahovalo všetky potrebné dáta na komunikáciu. Toto umožnilo aj jednoduchú implementáciu nášho algoritmu, pri ktorom nebolo vopred zadané, aké údaje obsahuje. Z názvu objektu sa zistilo, o akú premennú ide a tá sa zobrala z globálnej premennej, ktorá obsahovala všetky dáta z PLC zariadenia. Následne sa dáta odoslali na MQTT broker.

Funkcia *SubscribeTopics* nám poskytla pripojenie sa na MQTT reláciu. Na tejto funkcii boli postavené funkcie ako *StartSubscribePlant* a *GetJSONPlant*. Funkcia *StartSubscribePlant* umožňovala pripojenie sa k jednému z modelov. To zahŕňa odpojenie od všetkých relácií, vynulovanie aktuálnej globálnej premennej, pripojenie sa na reláciu *sustavaX_initializer*, *sustavaX_reader* a požiadať o inicializačné dáta. Funkcia *GetJSONPlant* sa zase pripojí na *sustavaX_json* reláciu a požiada o dáta používateľského rozhrania pre danú sústavu.

Ďalšou dôležitou funkciou bola *OnMessageArrivedHandler*. Táto zabezpečovala ošetrenie dát, ktoré Unity aplikácia prijímala od MQTT brokera. V tomto prípade sa rozpoznávalo, či sa jedná o dáta používateľského rozhrania alebo aktuálne dáta z výrobných modelov. Oba prípady spracovania dát opíšeme detailne neskôr.

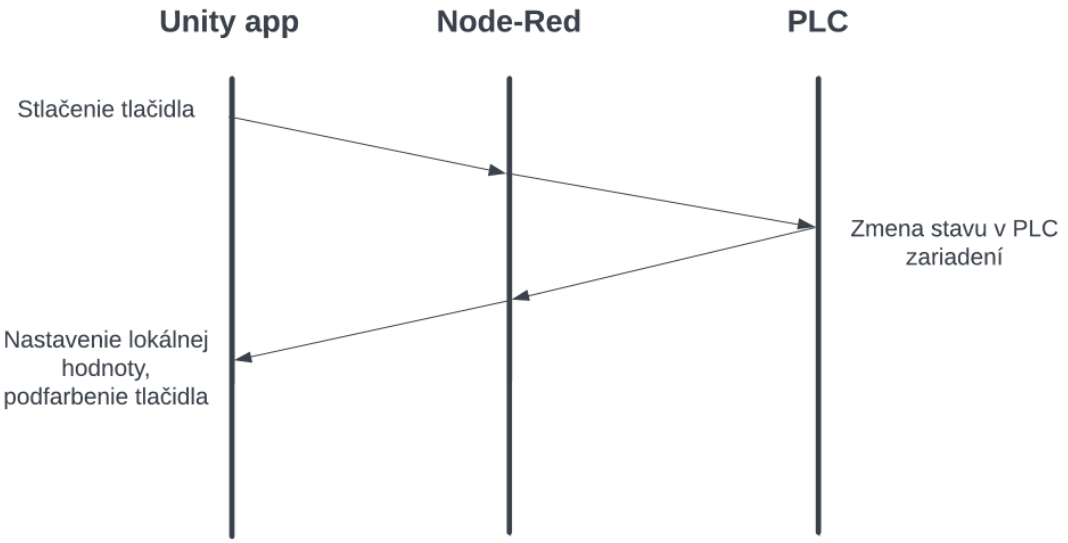
V komunikačnej štruktúre lokálny server bol sprostredkovateľ dát medzi PLC zariadením a Unity aplikáciou. Lokálny server teda nedržal informáciu o stave všetkých dát, tieto stavy bolo ale potrebné uchovávať v Unity aplikácii. Nakoľko ale Unity aplikácia nie je špecifikovaná na jeden typ výrobného modelu, bolo potrebné zabezpečiť dynamickú premennú s rôznymi typmi dát a rôznych kľúčov. Preto dôležitou súčasťou MQTT implementácie je aj spôsob uchovávania dát v Unity aplikácii.

Uchovávanie dát spočíva v nami vytvorenej statickej triede *GlobalVariables*. Táto trieda obsahovala premennú *MQTT_data* vo forme slovníka, ktorý bol založený na definovaní kľúča vo formáte reťazca a taktiež dát vo formáte reťazca. Dáta bolo potrebné ukladať v jednej forme, nakoľko Unity veľmi ťažko pracuje s dynamickými premennými. V triede sa nachádzali aj funkcie na obsluhu globálnych dát. Boli nimi funkcie ako getter, setter, vyprázdnenie slovníka, funkcie na kontrolu formátu dát alebo funkcie na konvertovanie formátu dát.

Getter a setter boli funkcie založené na vrátení alebo uložení dát na základe kľúča. V prípade settera, ak sa kľúč v slovníku nenachádzal, tak sa vytvoril nový kľúč s hodnotou, ktorú sme chceli zapísať. V prípade getteru, ak sa v slovníku daný kľúč nenachádzal, vrátila sa hodnota *null*. Globálnu premennú, jej getter a setter.

Dáta sa pri inicializácii nového modelu výrobnej linky naplnili všetkými premennými, ktoré dané PLC zariadenie obsahovalo a pri zmene dát už jednoducho iba aktualizovalo dáta na aktuálnu hodnotu. Pri MQTT komunikácii sa všetky dáta prijímali vo formáte reťazca a preto bolo potrebné ich najprv prekonvertovať na JSON formát, z ktorého sa potom získal názov premennej a jej hodnota. Globálne dáta nebolo možné aktualizovať z Unity aplikácie, každá zmena dát bola odoslaná do PLC zariadenia a na základe zmeny danej premennej v PLC zariadení sa prijala jej aktuálna hodnota. Týmto sme zabezpečili spoľahlivé zobrazovanie aktuálnych hodnôt všetkých premenných z PLC zariadenia a nebolo možné, aby sa v aplikácii zobrazovala nesprávna hodnota.

Príklad komunikácie môžeme vidieť na Obr. 18. Pri stlačení tlačidla vyšleme hodnotu danej premennej pomocou MQTT komunikácie, ktorá ju nastaví v PLC zariadení. Lokálny server zaznamená zmenu danej premennej a následne ju vyšle pomocou MQTT komunikácie do Unity aplikácie, kde sa zmena hodnoty premennej uloží do (z pohľadu aplikácie) globálnej premennej.



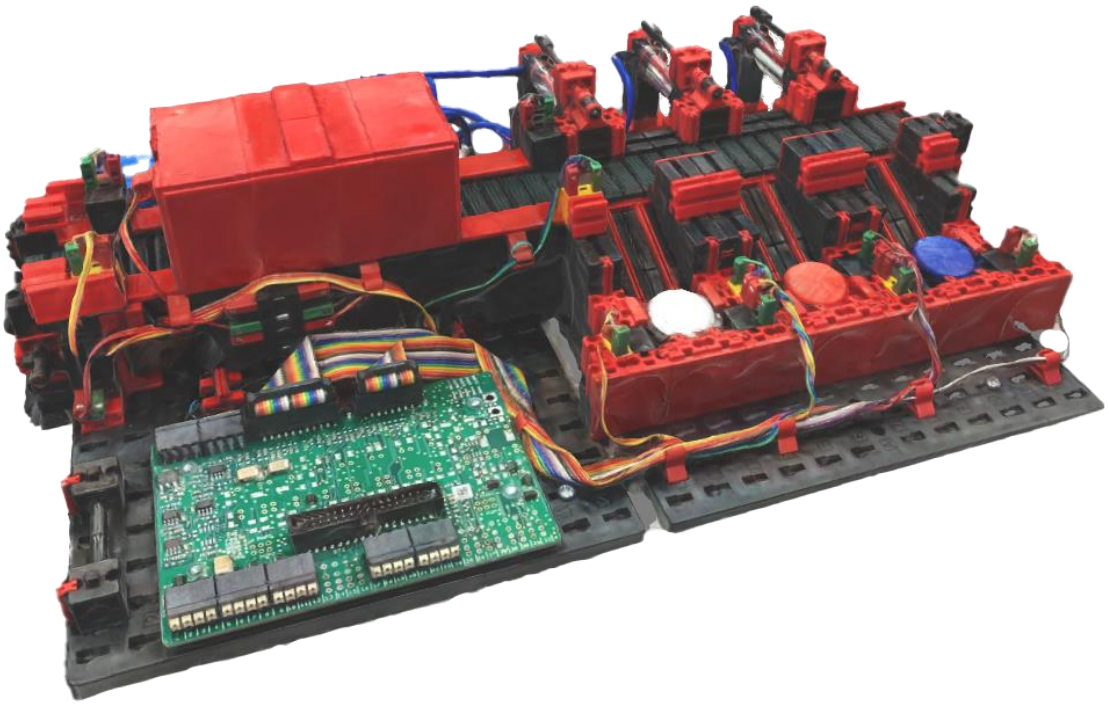
Obr. 18 Komunikácia v rámci celého projektu

Vuforia

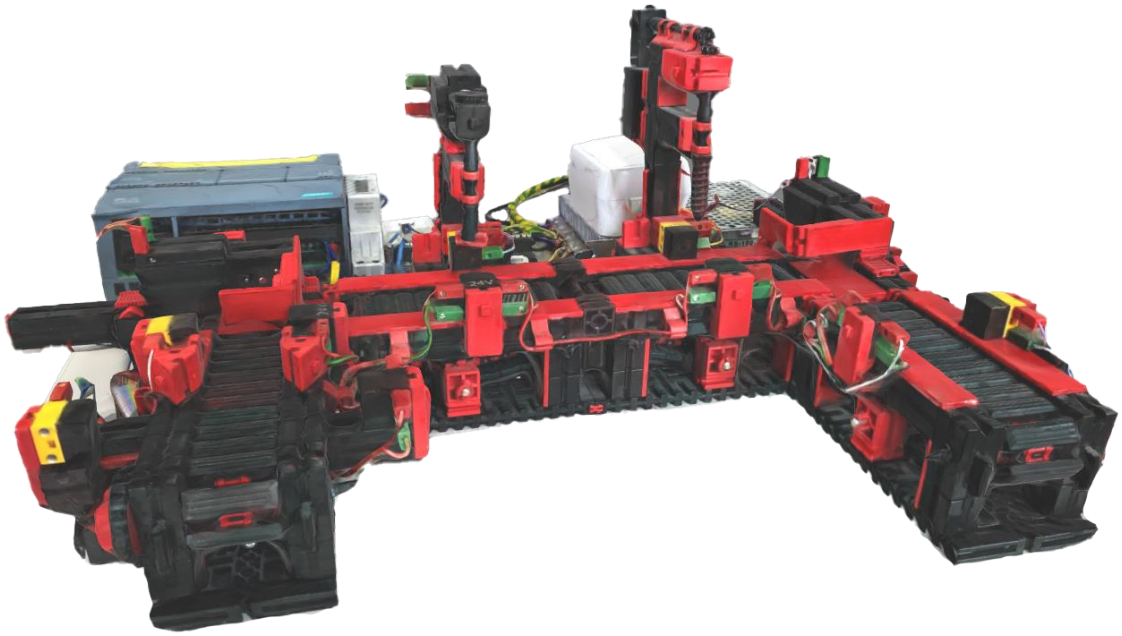
Vuforia je softvérová platforma pre rozšírenú realitu (AR), ktorá umožňuje vývojárom integrovať digitálne prvky do fyzického sveta pomocou mobilných zariadení. Implementácia *Vuforie* do Unity projektu nám umožnila rozpoznávať 3D objekty a sledovať ich v reálnom čase. Tomu ale predchádzalo vytvorenie 3D modelov výrobných liniek, na základe ktorých Vuforia dokázala rozpoznať reálny objekt.

Na tvorbu 3D objektov výrobných liniek sme otestovali viacero aplikácií, ako *Polycam*, *Scandy Pro*, *Metascan*, *Qlone* a *MafiScan*. Aplikácia *Polycam* bola najefektívnejšia, pričom bez poplatkov nám poskytla najviac autentické 3D modely s údajmi o veľkosti objektu, ktoré bolo možné exportovať v bežných typoch súborov 3D modelov, napr. vo formáte .obj alebo .fbx.

Pomocou aplikácie *Polycam* sme naskenovali 3D objekty pomocou mobilného zariadenia. V našom prípade išlo o iOS mobilné zariadenie iPhone 11 Pro Max. Zariadenie disponovalo dostatočne kvalitným fotoaparátom, ktorým sme zachytili viac ako 100 fotografií laboratórneho modelu výrobnej linky. Na základe týchto fotiek nám aplikácia umožnila vytvoriť 3D model výrobnej linky a exportovať ho do súboru .fbx. Týmto spôsobom sme vytvorili oba 3D modely výrobných liniek. Na Obr. 19 a Obr. 20 môžeme vidieť naskenované 3D modely pomocou aplikácie *Polycam.*



Obr. 19 Naskenovaný 3D model výrobnej linky (model 2)



Obr. 20 Naskenovaný 3D model triediacej linky s detekciou farby (model 1)

Pre detekciu objektov v reálnom svete bolo potrebné natrénovať rozpoznávanie reálnych modelov na základe získaných 3D modelov. To nám umožnila aplikácia *Vuforia Model Target Generator*. Na základe získaných 3D modelov sme vytvorili databázu *SustavyPro*, ktorá obsahovala údaje 3D modelov.

Do Unity projektu bolo potrebné importovať knižnicu *Vuforia Engine AR*. Natrénovanú databázu sme implementovali do Unity projektu a vytvorili všetky potrebné objekty pre rozpoznávanie reálnych objektov. Tu bolo potrebné všetky Vuforia objekty vložiť do prázdneho objektu s názvom *VuforiaContent*. Ak by sme tento krok vynechali, tak by Vuforia knižnica vôbec nezaznamenávala objekty. Objekty *Sustava1* a *Sustava2* predstavujú 3D objekty, na základe ktorých sa rozpoznávajú reálne objekty. V oboch objektoch bolo potrebné napojenie k databáze a cieľovému modelu v databáze. Po pripojení cieľového modelu sa nám automaticky doplnili rozmery objektu.

Ďalšou potrebnou konfiguráciou bolo nastaviť, čo sa má udiať pri rozpoznaní a stratení objektov. Na to nám slúži objekt *RecognitionObjects*, ktorý poskytuje funkcie, aké sa pri týchto udalostiach vykonajú. Pri rozpoznaní objektu sa Unity projekt pripojí ku všetkým MQTT reláciám, požiada lokálny server o JSON UI a po vyrenderovaní užívateľského prostredia si vyžiada všetky inicializačné dáta PLC zariadenia danej sústavy. Pri strate objektu sa odpojí od všetkých MQTT relácií a aktuálne vygenerované užívateľské prostredie vymaže zo scény.

Algoritmus vykreslenia používateľského rozhrania

Ako už bolo spomenuté, aplikácia renderuje používateľské rozhranie jednotlivých výrobných liniek na základe nami zadefinovanými dátami - JSON UI. Dáta sa nachádzajú v súboroch na lokálnom serveri, kde ich je možné zmeniť kedykoľvek bez potreby zostavenia novej verzie Unity aplikácie.

Pri prijímaní dát z lokálneho servera sa sleduje, z akej MQTT relácie sa prijali dáta. Ak boli dáta prijaté z relácie, ktorá nesie v sebe reťazec *json*, dáta sa ošetria podľa algoritmu na renderovanie užívateľského rozhrania. Tento algoritmus pozostáva z nasledovných krokov:

- preformátovanie dát z reťazca do JSON-u,
- odstránenie predošlého užívateľského prostredia, ak existuje v scéne,
- renderovanie nového užívateľského prostredia,
- pripojenie ku všetkým MQTT reláciám rozpoznanej výrobnej linky,
- požiadanie inicializačných dát aktuálnej výrobnej linky.

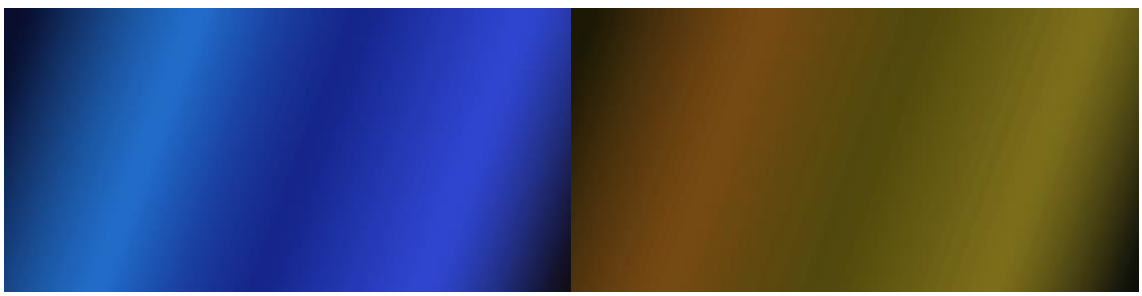
V programovacom jazyku C# sa pri formátovaní z reťazca očakáva, že budeme vopred vedieť, aký formát údajov nám reťazec poskytne. V našom prípade očakávame

formát. Dáta teda očakávame vo forme objektu, ktorý nesie informácie o názve výrobného modelu a jeho čísla. Na základe názvu modelu algoritmus určí, do ktorého objektu v scéne vyrenderuje všetky objekty. Teda, ak prijmeme názov *PLC1*, dáta sa vyrenderujú do 3D objektu s názvom *MenuPLC1*. Každý z 3D modelov výrobných liniek obsahuje v sebe takýto menu objekt, vďaka čomu zabezpečíme, že užívateľské prostredie výrobnej linky sa bude vždy nachádzať na súradniciach z rozpoznanej výrobnej linky. Číslo výrobnej linky nám zase určuje, ku ktorým MQTT reláciám sa pripojí Unity aplikácia.

Ďalej objekt obsahuje pole *content*, ktoré reprezentuje obsah všetkých objektov, ktoré sa nachádzajú v užívateľskom prostredí. Napríklad objekty typu *background*, ktoré v sebe obsahujú vnorené objekty ako texty, tlačidlá, štvorce alebo kruhy.

Objekt typu *background* v sebe nesie informácie: názov 3D objektu *name*, farbu pozadia *color*, viditeľnosť 3D objektu *visible*, polohu a rozmer 3D objektu *position* a vnorené 3D objekty *content*. Farbu pozadia môžeme nastaviť na červenú, zelenú, žltú, modrú, zlatú, čiernu alebo bielu. Farba pozadia je reprezentovaná ako komponent *Renderer* a farebné spektrum určuje mapa spektra. Na Obr. 21 môžeme vidieť takéto farebné spektrum v modrej a zlatej farbe. Každá farba, ktorú je možné nastaviť 3D objektu, bola vytvorená v aplikácii *Photoshop*. Nakoľko užívateľské prostredie môže obsahovať viacero panelov, viditeľnosť 3D objektu určuje, ktorý bude pri renderovaní aktívny. Pri prepínaní medzi panelmi sa potom aktuálne zobrazovaný panel deaktivuje a panel, na ktorý sa chceme prepnúť, sa aktivuje. Vnorené 3D objekty reprezentujú všetky objekty, ktoré sa na danom paneli zobrazujú, uvádzame ich ďalej.

Kľúč objektu *position* nesie dáta, ktoré obsahuje každý z objektov v JSON UI reprezentujúci 3D objekt. Obsahuje dáta *posY*, *posX*, *posZ*, ktoré sú reprezentované vo formáte float a určujú polohu 3D objektu v závislosti od jeho nadradeného 3D objektu. Obsahuje aj dáta *scaleX*, *scaleY* a *scaleZ*, ktoré určujú veľkosť v jednotlivých osiach.



Obr. 21 Mapa spektra v modrej farbe (vľavo) a zlatej farbe (vpravo)

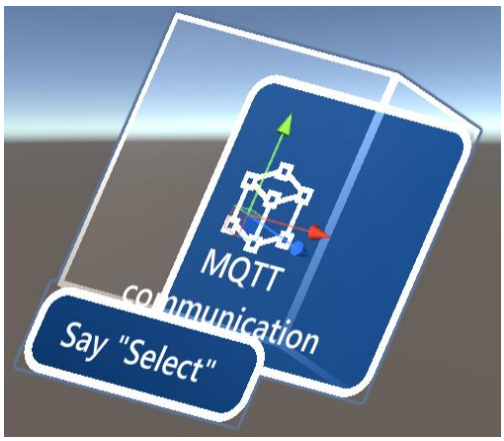
Kľúč objektu *content* obsahuje pole vnorených objektov, ktoré sa nachádzajú na danom paneli. V závislosti od typu 3D objektu môže dátový objekt obsahovať rôzne dáta. Každý dátový objekt reprezentujúci vnorený objekt obsahuje dáta ako *name, tag, gameObjectType* a *position*. V závislosti od typu, ktorý určuje kľúč *gameObjectType,* môže dátový objekt obsahovať navyše dáta *buttonProperties* (v prípade tlačidla), *textProperties* (v prípade textu) alebo *squadProperties* (v prípade kruhu a štvorca).

Spoločné dáta určujú základné vlastnosti 3D objektov. Kľúč *name* reprezentuje názov 3D objektu, kľúč *tag* určuje tag, ktorý sa priradí 3D objektu. Podľa tagu 3D objektu sa určuje, na akej MQTT relácii budú dáta odosielané. Kľúč *gameObjectType* určuje, o aký typ 3D objektu ide. V prípade tlačidla obsahuje hodnotu *button,* v prípade textu hodnotu *text,* v prípade štvorca hodnotu *squad* a v prípade kruhu *circle*. Kľuč *position,* ako už bolo spomenuté, určuje polohu a rozmer 3D objektu.

Ak ide o dáta 3D objektu textu, sú doplnené o hodnotu textu *text*, veľkosť fontu *fontSize,* farbu textu *textColor*, štýl textu *fontStyle*, vertikálne zarovnanie *verticalAlign* a horizontálne zarovnanie *horizontalAlign*. V prípade typu textu môže ísť o hodnoty *normal, bold* alebo *normal*. Farbu textu je možné zadefinovať pomocou hodnôt *black, white, red, blue, yellow* alebo *green*. Text je možné zarovnať vo vertikálnej rovine pomocou hodnôt *center, top, bottom* a v horizontálnej rovine *left, right, center* alebo *flush*. Hodnota *flush* zarovná text od jedného kraju po druhý.

3D objekty *squad* a *circle* sú najjednoduchšie objekty v algoritme a obsahujú navyše iba hodnotu *color*, pomocou ktorej môžeme meniť farbu pozadia objektu. *Circle* na rozdiel od *squad* 3D objektu sa pri renderovaní vyrenderuje s 50% polomerom.

3D objekty typu *button* sú v našom algoritme najzložitejšími objektami. Dáta navyše obsahujú text, ktorý sa zobrazuje na tlačidle a v štítku *text* a na ktorý aplikácia reaguje pri ovládaní rečou *label*. Nastavenie ikony tlačidla umožňuje kľúč *icon*, farbu tlačidla kľúč *color* (s rovnakými hodnotami ako pri pozadí). Kľúč *clickable* nesie booleovskú hodnotu, ktorá určuje, či je tlačidlo možné stlačiť a zoznam funkcií, ktoré sa po stlačení tlačidla vykonajú. Ak nie je povolené ovládanie tlačidla, zoznam funkcií je ignorovaný. Povolenie ovládania tlačidla určuje, či bude tlačidlo obsahovať aj vizualizáciu obrysu tlačidla v 3D prostredí (Obr. 22).



Obr. 22 3D vizualizácia tlačidla

Jednotlivé funkcie obsahujú dáta ako názov funkcie *functionName*, hodnotu *variable* a typ premennej *variableType*.

Na výber je z niekoľkých nasledujúcich funkcií:

- *SendDataToMQTTServerByValue* – odoslanie dát do PLC zariadenia s negovaním hodnoty,
- *SendDataToMQTTServer* – fixné odoslanie dát do PLC zariadenia so statickou hodnotu,
- *HandleVisiblity* – prepnutie aktívneho panelu,
- *HandleVisiblityByControl* – prepnutie aktívneho panelu na základe premennej,
- *HandleVisiblityByControlWithOtherValue* – rovnaká funkcia ako predošlá obohatená o vyslanie dát do PLC zariadenia.

Nakoľko niektoré 3D objekty, ako napríklad tlačidlo, obsahujú viacero vnorených 3D objektov, bolo by príliš komplikované vytvárať nové 3D objekty v skripte. Preto sme vytvorili preddefinované 3D objekty, v Unity nazývané ako *prefab*. Na základe prijatých dát sme použili konkrétne objekty typu *prefab* pre tlačidlá, texty, pozadia alebo štvorce/kruhy a upravili jeho konfiguráciu podľa typu objektu.

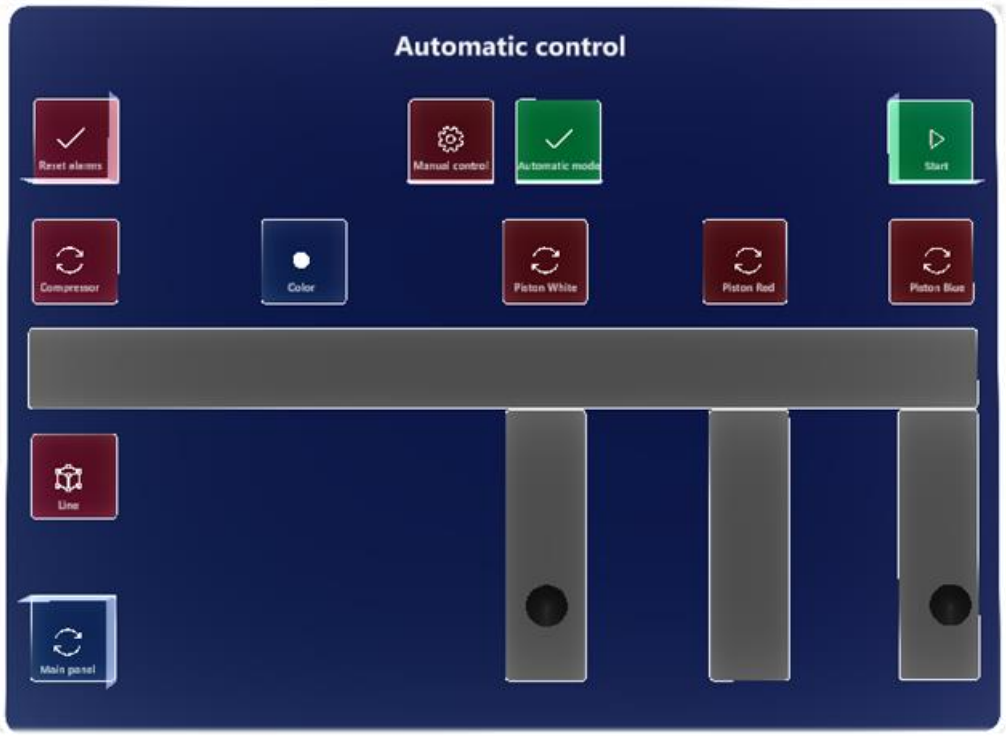
Vzhľad aplikácie

Nakoľko aplikácia bola vyvíjaná na zariadenie MS HoloLens 2, vzhľad aplikácie bol koncipovaný podľa MRTK dizajnu. Komponenty boli použité z knižnice *MRTK templates* a upravené podľa našich potrieb.

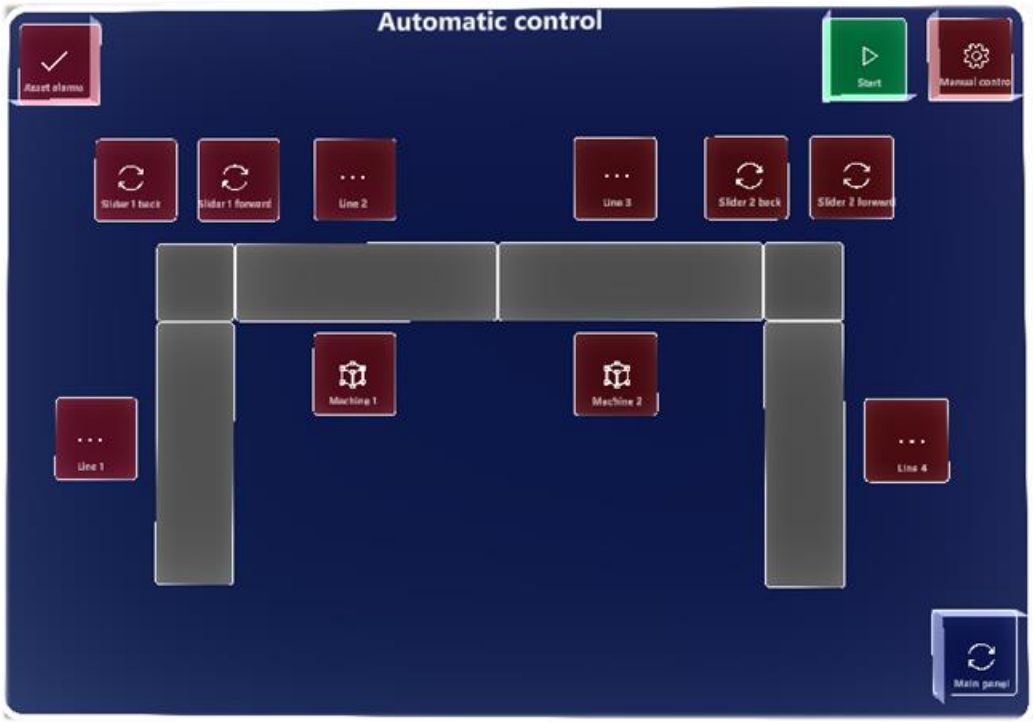
Aplikácia obsahovala hlavné menu s možnosťou aktivovania a deaktivovania komunikácie s MQTT brokerom, zobrazenia aktívnej výrobnej linky a debugovacími tlačidlami pre aktivovanie jednotlivých výrobných liniek. Nakoľko šlo o aplikáciu v zmiešanej realite, nebolo možné využiť bežné praktiky ako pri webových alebo mobilných aplikáciách. Menu teda bolo realizované nasledovaním užívateľa s možnosťou otvorenia a zatvorenia, nakoľko pri otvorenom stave zaberalo veľkú plochu. Pri interakcii sa menu animovalo.

Každá z liniek poskytovala viacero panelov, pričom pri rozpoznaní výrobnej linky sa zobrazil úvodný panel s možnosťou presmerovania na všetky panely, ktoré výrobné linky obsahovali. Presmerovanie bolo uskutočnené stlačením tlačidla. Každý panel v tejto sekcii je vygenerovaný z JSON UI.

Pre každý z režimov ovládania laboratórnych výrobných modelov bol navrhnutý panel podľa kapitoly 3.1. Pri prepínaní režimov ovládania sa automaticky prepínali aj panely príslušné aktivovanému režimu ovládania. Na Obr. 23 a Obr. 24 môžeme vidieť panely automatických režimov pri oboch výrobných linkách. Pri manuálnom režime môžeme vidieť rovnaké rozloženie objektov s tým rozdielom, že tlačidlá reagujú na stlačenie, pri automatickom režime sú tlačidlá blokované.

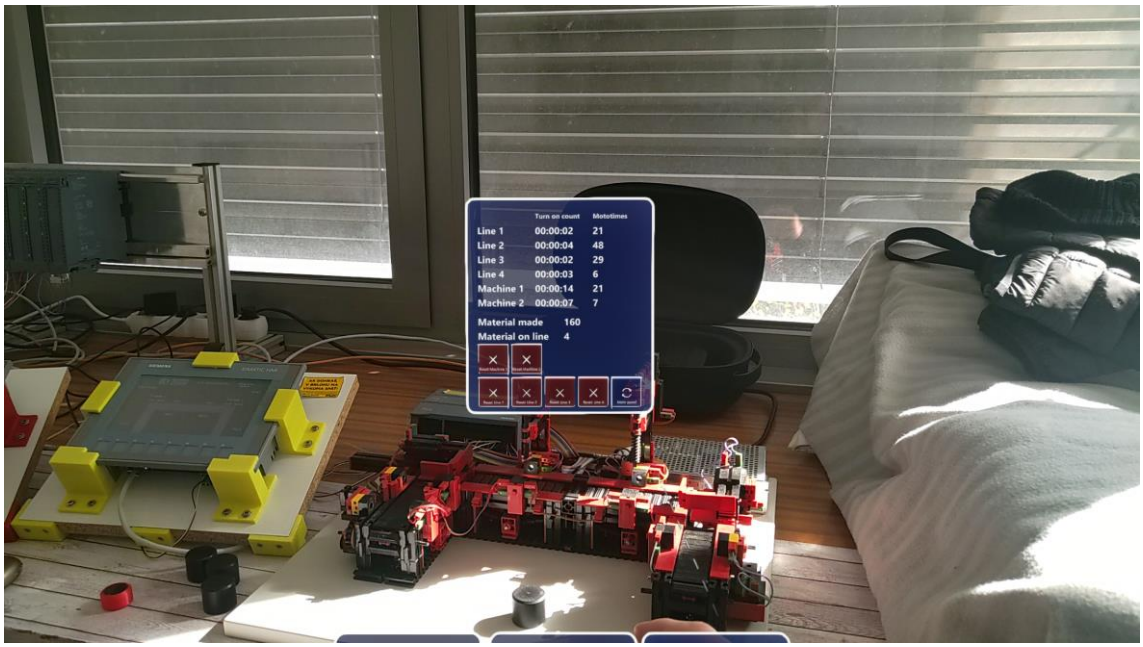


Obr. 23 Panel s možnosťou sledovania a ovládania triediacej linky



Obr. 24 Panel s možnosťou sledovania a ovládania výrobnej linky

Po rozpoznaní linky v reálnom svete sa panely zobrazili nad výrobnou linkou a po celú dobu si držali túto pozíciu. Pri strate detegovanej výrobnej linky sa panely vymazali zo scény virtuálnej reality. Pri pohybe používateľa sa panely naklápali za používateľom a tak bol zabezpečený dobrý zorný uhol z akejkoľvek pozície používateľa. Virtuálne panely umiestnené nad reálnymi výrobnými linkami môžeme vidieť na Obr. 25 a Obr. 26.



Obr. 25 Detekcia výrobnej linky v realite

Obr. 26 Detekcia triediacej linky v realite

4 Záver

Využitie rozšírenej reality vo výrobnom procese môže zefektívniť rýchlosť výroby, údržbu strojov, znížiť prestoje strojov a efektívne poskytnúť informácie o danom stroji. Implementácia dynamickej vizualizácie, založenej na zobrazení podľa dát v databáze príslušným strojom nám poskytne efektívne pridávať vizualizácie pre nové zariadenia vo výrobe. Tým sa eliminuje zdĺhavý proces programovania nových verzií aplikácie a umožní jednoducho pridať nové zariadenie aj s jeho vizualizáciou do databázy.

V práci sme sa oboznámili so súčasným stavom danej problematiky vo svete, kde sme analyzovali slabé aj silné stránky jednotlivých štúdií. Na základe výsledku našej analýzy sme vypracovali návrh riešenia digitálnej platformy a implementovali sme naše riešenie na laboratórne modely.

V poslednej časti sme sa venovali vývoju potrebných aplikácií a celého komplexného systému. Zaoberá sa opisom fyzického výrobného procesu a opisom použitých technológií. Ďalej sa kapitola zaoberá spracovaním dát a použitím komunikačných protokolov a tiež poskytnutím dát externým zariadeniam a vzdialeným zariadeniam. V tejto kapitole boli opísané všetky využité cloudové technológie a použité webové technológie. Návrh platformy uzatvárame vývojom Unity aplikácie v zmiešanej realite pre zariadenie HoloLens 2 a opisom navrhnutého algoritmu.

Prínosy práce

Za prínosy dizertačnej práce možno považovať:

- **Návrh a implementácia originálneho aplikačného systému pre monitorovanie a riadenie diskrétnych udalostných systémov s využitím inteligentných technológií**

  V rámci riešenia dizertačnej práce bol navrhnutý a implementovaný softvérový aplikačný systém pre podporu monitorovania, diagnostiky a riadenia diskrétnych udalostných systémov. Tento aplikačný systém je založený na moderných inteligentných technológiách a konceptoch, ako je Internet of Things, rozšírená realita, počítačové videnie, cloud computing a

podobne. Tieto technológie patria medzi piliere riešení, ktoré v praxi pomáhajú digitalizovať výrobné procesy, a spadajú pod koncept Industry 4.0.

- **Implementácia systému pre ovládanie a riadenie diskrétnych udalostných systémov na headset pre rozšírenú realitu**

  Aplikačný systém bol implementovaný na momentálne najvyspelejší headset pre rozšírenú / zmiešanú realitu Microsoft HoloLens 2 s využitím knižnice Mixed Reality Toolkit. Konvenčné riešenia využívajú mobilné zariadenia, ako sú smartfóny alebo tablet. Využitie headsetu tak prináša mnohé výhody a to najmä fakt, že ruky operátora ostávajú pri využívaní aplikačného systému voľné. Vyvinuté riešenie, tak môžeme považovať ako riešenie zamerané na človeka (angl. human-centric), čo je jeden z pilierov konceptu Industry 5.0.

- **Využitie priemyselných štandardov a priemyselných hardvérových komponentov v navrhnutom aplikačnom systéme**

  V aplikačnom systéme bol využitý aj priemyselný komunikačný štandard OPC UA a PLC, čím sa líši od iných riešení, ktoré sú často založené na bežných komunikačných protokoloch využiteľných len v spotrebiteľskom IoT a prototypovacej elektronike typu Arduino a podobne.

- **Modifikácia a rozšírenie konceptu definičných schém pre dynamické generovanie GUI v rozšírenej realite, jeho implementácia a overenie**

  Významným výsledkom práce je modifikácia a rozšírenie konceptu definičných schém pre dynamické generovanie grafického používateľského rozhrania pre systémy rozšírenej reality predstaveného v práci [101]. Modifikácia a rozšírenie spočíva vo využití zariadenia na vizualizáciu v rozšírenej realite MS HoloLens 2 a väčšími možnosťami nastavení a vizualizácie objektov.

- **Overenie navrhnutého a implementovaného aplikačného systému na dvoch laboratórnych mechatronických systémoch**

  Originálny aplikačný systém bol overený na dvoch laboratórnych udalostných mechatronických systémoch.

  a) laboratórny model výrobnej linky s optickými snímačmi a so snímačmi polohy, s dopravníkmi a 2 obrábacími nástrojmi

  b) model triediacej linky pre rozpoznávanie a triedenie rôznych obrobkov na základe detekcie farby

  Navrhnutý aplikačný systém pozostával s lokálneho serveru na báze Node-Redu, cloudového servera s databázou v prostredí MS Azure, aplikácie pre zobrazenie zmiešanej reality v prostredí Unity a webových aplikácií v prostredí Angular a Spring Boot 3. Dáta bolo možné zobraziť lokálne pomocou zmiešanej reality ale aj vzdialene cez webový prehliadač.

Prínosy dizertačnej práce deklarované v hore uvedených bodoch predstavujú nové trendy vo vývoji, výskume a aplikácií nových postupov a riešení v oblasti ovládania a monitorovania mechatronických systémov. Výsledky práce je možné použiť pre základ ďalšieho výskumu.

Zoznam použitej literatúry

[1] Jonathan Rosales, Sourabh Deshpande, Sam Anand, *IIoT based Augmented Reality for Factory Data Collection and Visualization*, Procedia Manufacturing 53, 2021, p. 618-627, ISSN 2351-9789,

[2] Mirza Jabbar Aziz Baig, M. Tariq Iqbal, Mohsin Jamil, Jahangir Khan, *Design and implementation of an open-Source IoT and blockchain-based peer-to-peer energy trading platform using ESP32-S2, Node-Red and, MQTT protocol*, Energy Reports 7,2021, p. 5733-5746, ISSN 2352-4847

[3] Dimitris Mourtzis, John Angelopoulos, Nikos Panopoulos, *Collaborative manufacturing design: a mixed reality and cloud-based framework for part design*, Procedia CIRP 100, 2021, p. 97-102, ISSN 2212-8271,

[4] Jared Alan Frank, Sai Prasanth Krishnamoorthy, Vikram Kapila, *Toward Mobile Mixed-Reality Interaction With Multi-Robot Systems*, IEEE Robotics and Automation Letters 2, 2017, p. 1901 – 1908, ISSN: 2377-3766

[5] Nedim Kovacevic, jantje Meinzer, Rainer Stark, *Nutzerzentrierte Entwicklung einer ortsunabhängigen Maschinenabnahme mittels Augmented Reality*, Entwerfen Entwickeln Erleben in Produktentwicklung und Design, 2021, p. 417-429, ISBN: 978-3-95908-450-5

[6] Facebook, *Device interaction in augmented reality* , USA patent klasifikovaný G06F3/04815 - Interakcia s trojrozmernými prostrediami ,číslo patentu: 20160274762. Vynálezcovia : Lopez Javier San Agupen Henrikstin Vysoký (Frederiksberg), Rasmus Dahl (Kodaň), Jonas Priesum (Kodaň)

[7] Ing. Erich Stark, *Moderné metódy ovládania a monitorovania mechatronických systémov s využitím počítacom generovanej reality*. Dizertačná práca (Vedúci práce: doc. Ing. Peter Drahoš, PhD.). Bratislava: FEI STU, 2019.

[8] M. Husinsky, A. Schlager, A. Jalaeefar, S. Klimpfinger and M. Schumach, "Situated Visualization of IIoT Data on the Hololens 2," 2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), Christchurch, New Zealand, 2022, pp. 472-476, doi: 10.1109/VRW55335.2022.00104.

[9] Carsten Wittenberg, *Challenges for the human-machine interaction in times of digitization, CPS & IIoT, and artificial intelligence in production systems*, IFAC-PapersOnLine, Volume 55, Issue 29, 2022, Pages 114-119, ISSN 2405-8963.

[10] Stephen Watts, Muhammad Raza, *SaaS vs PaaS vs IaaS: What's The Difference & How To Choose* [online]. Dostupné z: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/#ref1

[11] European Commision. Industry 5.0. [online]. Dostupné z: https://research-and-innovation.ec.europa.eu/research-area/industrial-research-and-innovation/industry-50_en

[12] Publication Office of the European Union. *Enabling Technologies for Industry 5.0*. [online]. Dostupné z: https://op.europa.eu/en/publication-detail/-/publication/8e5de100-2a1c-11eb-9d7e-01aa75ed71a1/language-en

[13] Publication Office of the European Union. *Industry 5.0*. [online]. Dostupné z: https://op.europa.eu/en/publication-detail/-/publication/468a892a-5097-11eb-b59f-01aa75ed71a1/language-en

[14] Library.automationDirect, *History of the PLC* [online]. Dostupné z: https://library.automationdirect.com/history-of-the-plc/

[15] Interaction-design. *Augment Reality* [online]. Dostupné z: https://www.interaction-design.org/literature/topics/augmented-reality

[16] Neurosys, *Most popular, proven, and appraised wearable AR devices* [online]. Dostupné z: https://neurosys.com/most-popular-wearable-ar-devices/

Publikačná činnosť autora

Vysokoškolské učebnice vydané v domácich vydavateľstvách

CIGÁNEK, Ján - ŽEMLA, Filip. *PLC systémy v mechatronike*. 1. vydanie. Bratislava : Slovenská chemická knižnica FCHPT STU, 2023. 88 s. ISBN 978-80-8208-097-4.

Vedecké práce v zahraničných karentovaných časopisoch

ŽEMLA, Filip - CIGÁNEK, Ján - ROSINOVÁ, Danica – KUČERA, Erik - HAFFNER, Oto. *Complex Positioning System for the Control and Visualization of Photovoltaic Systems*. Energies 2023, 16, 4001 ,ISSN: 1996-1073, SJR:Q2

Vedecké práce v ostatných domácich časopisoch

CIGÁNEK, Ján - ŽEMLA, Filip. Modeling of nonlinear dynamic systems using soft computing methods. In *Information Technology Applications*. Vol. 9, No. 1 (2020), s. 77-88. ISSN 1338-6468.

ŽEMLA, Filip - CIGÁNEK, Ján. An application for optimal material flow in industrial warehouses. In *Information Technology Applications*. Vol. 10, No. 1 (2021), s. 13-23. ISSN 1338-6468.

ŽEMLA, Filip - CIGÁNEK, Ján. Digitization of event systems in augmented reality using intelligent technologies. In *Information Technology Applications*. Vol. 10, No. 2 (2021), s. 25-35. ISSN 1338-6468.

ŽEMLA, Filip - CIGÁNEK, Ján. Remote data acquisition using cloud database. In *Information Technology Applications : 31st International Conference on Cybernetics & Informatics (K&I). Visegrád, Maďarsko. 11-14 September 2022*. Vol. 11, No. 1 (2022), s. 43-48. ISSN 1338-6468.

Publikované príspevky na zahraničných vedeckých konferenciách

CIGÁNEK, Ján - ŽEMLA, Filip. Design of digital twin for PLC system. In *2022 Cybernetics & Informatics (K&I) [elektronický zdroj] : Proceedings ; 31st International Conference; 11-14 September 2022 Visegrád, Maďarsko*. 1. vydanie. Danvers, Massachusetts, USA : IEEE, 2022, [6] s. ISBN 978-1-6654-8775-7. V databáze: DOI: 10.1109/KI55792.2022.9925961 ; SCOPUS: 2-s2.0-85142057777 ; IEEE: 9925961.

CIGÁNEK, Ján - ŽEMLA, Filip. Complex mechatronic system for the motion control of a garage door. In *2022 Cybernetics & Informatics (K&I) [elektronický zdroj] : Proceedings ; 31st International Conference; 11-14 September 2022 Visegrád, Maďarsko*. 1. vydanie. Danvers, Massachusetts, USA : IEEE, 2022, [6] s. ISBN 978-1-6654-8775-7. V databáze: DOI: 10.1109/KI55792.2022.9925970 ; SCOPUS: 2-s2.0-85142065578 ; IEEE: 9925970.

ŽEMLA, Filip - CIGÁNEK, Ján. Remote data acquisition using cloud database. In *2022 Cybernetics & Informatics (K&I) [elektronický zdroj] : Proceedings ; 31st International Conference; 11-14 September 2022 Visegrád, Maďarsko*. 1. vydanie. Danvers, Massachusetts, USA : IEEE, 2022, [6] s. ISBN 978-1-6654-8775-7. V databáze: DOI: 10.1109/KI55792.2022.9925932 ; SCOPUS: 2-s2.0-85142014342 ; IEEE: 9925932.

ŽEMLA, Filip - CIGÁNEK, Ján. Design and implementation of the application for the irrigation system. In *2022 Cybernetics & Informatics (K&I) [elektronický zdroj] : Proceedings ; 31st International Conference; 11-14 September 2022 Visegrád, Maďarsko*. 1. vydanie. Danvers, Massachusetts, USA : IEEE, 2022, [6] s. ISBN 978-1-6654-8775-7. V databáze: DOI: 10.1109/KI55792.2022.9925975 ; SCOPUS: 2-s2.0-85142018977 ; IEEE: 9925975.

Publikované príspevky na domácich vedeckých konferenciách

ŽEMLA, Filip - CIGÁNEK, Ján - ROSINOVÁ, Danica. Optimization of material flow in industrial warehouses. In *ELITECH´21 [elektronický zdroj] : 23th Conference of Doctoral Students, May 26, 2021*. 1. ed. Bratislava : Vydavateľstvo Spektrum STU, 2021, [6] s. ISBN 978-80-227-5098-1.

ŽEMLA, Filip - CIGÁNEK, Ján. Digitization of event systems using intelligent technologies for Industry 4.0. In *ELITECH´22 [elektronický zdroj] : 24th Conference of Doctoral Students. Bratislava, Slovakia. June 1, 2022*. 1. ed. Bratislava : Vydavateľstvo Spektrum STU, 2022, [6] s. ISBN 978-80-227-5192-6.