EDÍCIA VYSOKOŠKOLSKÝCH UČEBNÍC

DIGITÁLNE A INTELIGENTNÉ TECHNOLÓGIE PRE INDUSTRY 4.0

Erik Kučera

STU FEI
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

SPEKTRUM STU

DIGITÁLNE A INTELIGENTNÉ TECHNOLÓGIE PRE INDUSTRY 4.0

DIGITÁLNE A INTELIGENTNÉ TECHNOLÓGIE PRE INDUSTRY 4.0

Erik Kučera

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
2024

Vysokoškolská učebnica poskytuje základný prehľad vybraných digitálnych a inteligentných technológií konceptu Industry 4.0 a ich aplikácií. Medzi tieto technológie patrí napríklad virtuálna, rozšírená a zmiešaná realita, cloud computing, Internet of Things a podobne.
Publikácia je cieľavedome zameraná na rozšírenie vedomostí a zručností študentov a odborníkov z praxe v oblasti digitalizácie mechatronických a výrobných systémov s využitím moderných informačných, softvérových a komunikačných prostriedkov. Digitálne a inteligentné technológie sa v dnešnej dobe využívajú pre široké spektrum úloh v rámci projektovania a návrhu moderných výrob. Publikácia prináša viacero názorných edukačných prípadových štúdií na osvojenie si prezentovaných konceptov, pričom sú využívané bežne dostupné softvérové alebo hardvérové prostriedky. Je vhodným doplnkom tak pre študentov technického zamerania v oblasti aplikovanej a inžinierskej informatiky, kybernetiky, automatizácie a mechatroniky, ako aj pre študentov prírodovedného zamerania zaoberajúcich sa teoretickou informatikou.

doc. Ing. Erik Kučera, PhD.

Recenzenti: Ing. Ivana Budinská, PhD. – Ústav informatiky, Slovenská akadémia vied
doc. Ing. Martin Juhás, PhD. – Ústav aplikovanej informatiky, automatizácie a mechatroniky, Materiálovotechnologická fakulta STU so sídlom v Trnave

Vydanie publikácie podporila Kultúrna a edukačná grantová agentúra MŠVVaM SR v rámci grantov 010STU-4/2023 a 021STU-4/2024.

Schválilo Vedenie Fakulty elektrotechniky a informatiky STU v Bratislave.

ISBN 978-80-227-5430-9

Poďakovanie

Týmto by som chcel v prvom rade poďakovať prof. Ing. Štefanovi Kozákovi, PhD., prof. Ing. Alene Kozákovej, PhD., prof. Ing. Danici Rosinovej, PhD., a doc. Ing. Petrovi Drahošovi, PhD., za dlhodobý odborný mentoring, bez ktorého by táto učebnica nevznikla. Chcem sa poďakovať Ing. Simone Lopatnikovej, Ing. Danijele Pavlovič, Ing. Michaele Štetákovej, Ing. Mihaelovi Miksadovi a Ing. Veronike Patkovej za prípravu opisovaných edukačných prípadových štúdií a zvlášť Ing. Simone Lopatnikovej aj za gramatickú a grafickú korektúru učebnice. Ďakujem Ing. Erichovi Starkovi, PhD., a Ing. Szabine Bucsai za výbornú spoluprácu počas vývoja systému pre monitorovanie a ovládanie mechatronických systémov s využitím rozšírenej reality, vďaka ktorej vznikla kapitola o počítačom generovanej realite. Veľká vďaka patrí talentovanému grafikovi Ing. Romanovi Leskovskému za nezištnú pomoc pri príprave vybraných ilustrácií v učebnici. Nepochybne treba poďakovať aj doc. Ing. Otovi Haffnerovi, PhD., a Ing. Ľubošovi Galandákovi za ich cenné rady a dlhodobú spoluprácu.

Obsah

Úvod **1**

1 Štvrtá priemyselná revolúcia — Industry 4.0 **3**
1.1 Priemyselné revolúcie — vývoj a trendy . . . 3
1.2 Súčasná priemyselná revolúcia Industry 4.0 . . . 4
1.3 Kľúčové technológie pre Industry 4.0 . . . 7

2 Automatizačná pyramída a technické prostriedky automatického riadenia **12**
2.1 Automatické riadenie a diskrétne udalostné systémy . . . 12
2.2 Automatizačná pyramída . . . 16
2.3 Programovateľné logické kontroléry (PLC) . . . 18
2.3.1 Rozdelenie PLC . . . 18
2.3.2 Programovacie jazyky PLC . . . 20
2.4 Modbus . . . 24
2.5 Konvergencia informačných a operačných technológií . . . 25
2.6 OPC Unified Architecture . . . 28
2.6.1 Komunikácia klient-server . . . 29
2.6.2 Komunikácia publish-subscribe (PubSub) . . . 32
2.6.3 OPC UA Field eXchange . . . 33

3 Vybrané digitálne technológie pre Industry 4.0 **35**
3.1 Cloud computing . . . 35
3.2 Internet of Things . . . 38
3.3 Edge computing . . . 42
3.4 Kontajnerizácia . . . 43
3.4.1 Porovnanie virtuálneho stroja a kontajnera . . . 44
3.4.2 Docker . . . 46
3.5 Počítačom generovaná realita . . . 47
3.5.1 Definícia PGR podľa The Foundry . . . 47
3.5.2 Definícia PGR podľa Milgrama a Kishiho . . . 49
3.5.3 Vybrané aplikácie počítačom generovanej reality . . . 50
3.5.4 Spolupráca a výmena informácií s priemyselnými partnermi . . . 54
3.6 Ďalšie technológie a softvérové nástroje využívané v učebnici . . . 59
3.6.1 Komunikačný protokol MQTT . . . 59
3.6.2 Node-RED . . . 63
3.6.3 OpenPLC . . . 69

3.6.4 Codesys . . . 69
3.6.5 Factory I/O . . . 70
3.6.6 Multiplatformový vývoj mobilných aplikácií . . . 71
3.6.7 Angular a Ionic . . . 73
3.6.8 3D engine Unity . . . 74

4 Edukačné prípadové štúdie **77**
4.1 Prvá prípadová štúdia: Prepojenie PLC s cloudovou aplikáciou Microsoft Azure IoT Central . . . 80
4.1.1 Špecifikácia a správanie diskrétneho udalostného systému . . . 81
4.1.2 Komponenty použité na tvorbu modelu linky . . . 82
4.1.3 Riadenie diskrétneho udalostného systému . . . 87
4.1.4 Komunikácia jednotlivých subsystémov . . . 93
4.1.5 Prepojenie OpenPLC runtime s Node-RED . . . 93
4.1.6 OPC UA server a klient . . . 98
4.1.7 Aplikácia v cloude Microsoft Azure . . . 103
4.2 Druhá prípadová štúdia: Prepojenie PLC s cloudovou aplikáciou realizovanou pomocou Node-RED . . . 109
4.2.1 Riadenie diskrétneho udalostného systému . . . 110
4.2.2 Komunikácia jednotlivých subsystémov . . . 112
4.2.3 OPC UA server v Codesys runtime . . . 113
4.2.4 Prepojenie runtimu Codesys s lokálnym Node-REDom . . . 115
4.2.5 Node-RED dashboard v cloude Microsoft Azure . . . 117
4.3 Tretia prípadová štúdia: Prepojenie PLC s webovou aplikáciou a databázou 120
4.3.1 Špecifikácia a správanie diskrétneho udalostného systému . . . 121
4.3.2 Komponenty použité na tvorbu modelu linky . . . 122
4.3.3 Riadenie diskrétneho udalostného systému . . . 126
4.3.4 Prepojenie Factory I/O s Codesys pomocou OPC UA . . . 131
4.3.5 Node-RED a práca s dátami pomocou OPC UA a MQTT . . . 135
4.3.6 PHP webová aplikácia . . . 144
4.3.7 Spätná komunikácia v Node-RED s OPC UA serverom . . . 152
4.4 Štvrtá prípadová štúdia: Prepojenie PLC s mobilnou aplikáciou . . . 154
4.4.1 Špecifikácia a správanie diskrétneho udalostného systému . . . 155
4.4.2 Špecifikácia mobilnej aplikácie . . . 157
4.4.3 Komunikácia jednotlivých subsystémov . . . 158
4.4.4 Riadenie diskrétneho udalostného systému . . . 160
4.4.5 Konfigurácia komunikácie medzi softPLC a Factory I/O . . . 168

4.4.6 Implementácia komunikácie v Node-RED . . . 171
4.4.7 Implementácia mobilnej aplikácie . . . 175
4.4.8 Zhrnutie prípadovej štúdie . . . 180
4.5 Piata prípadová štúdia: Prepojenie PLC s aplikáciou rozšírenej reality . . 185
4.5.1 Špecifikácia a zostavenie laboratórneho modelu diskrétneho udalostného systému . . . 186
4.5.2 Komunikácia jednotlivých subsystémov a inštalácia . . . 188
4.5.3 Riadenie diskrétneho udalostného systému . . . 191
4.5.4 Konfigurácia OPC UA, Node-RED, MQTT a MySQL . . . 198
4.5.5 Implementácia prepojenia PLC a rozšírenej reality . . . 200
4.5.6 Testovanie a validácia systému . . . 208

5 Vybrané aplikácie konceptu Industry 4.0 211
5.1 Matematické modely pre digitalizáciu výrobných procesov s podporou Petriho sietí . . . 211
5.1.1 Softvérový nástroj PN2ARDUINO . . . 214
5.1.2 Softvérové nástroje PetriNet editor a PetriNet engine . . . 218
5.1.3 Modelovanie a riadenie diskrétnych udalostných a hybridných systémov s využitím Petriho sietí a OPC UA . . . 219
5.2 Ovládanie a monitorovanie mechatronických systémov s využitím IoT a rozšírenej reality . . . 222
5.3 Nízkonákladová edukačná súprava pre výučbu hlbokého učenia pre aplikácie kontroly kvality . . . 229
5.4 Edukačno-vývojové platformy pre digitálne technológie Industry 4.0 . . . 233
5.4.1 Edukačno-vývojová platforma pre interoperabilitu . . . 234
5.4.2 Edukačno-vývojová platforma pre Industry 4.0 komponenty . . . 236
5.4.3 Edukačno-vývojová platforma pre virtuálnu a zmiešanú realitu . . . 237
5.4.4 Edukačno-vývojová platforma pre orientáciu na služby v reálnom čase . . . 239

6 Výhľad do budúcnosti — príchod konceptu Industry 5.0 241
6.1 Pozadie vzniku konceptu Industry 5.0 . . . 242
6.2 Paradigmy konceptu Industry 5.0 . . . 243
6.3 Technológie a aplikácie v rámci konceptu Industry 5.0 . . . 245

Záver 254

Zoznam použitej literatúry 256

Zoznam obrázkov a tabuliek

Obrázok č. 1 Priemyselné revolúcie a ich ťažiskové aspekty [56] 5
Obrázok č. 2 Atribúty Industry 4.0 [56] 7
Obrázok č. 3 Kyberneticko-fyzikálny systém (angl. cyber-physical system) . 8
Obrázok č. 4 Digitálne technológie pre zefektívnenie procesov v Industry 4.0 [56] 9
Obrázok č. 5 Správanie diskrétneho udalostného systému [54] 15
Obrázok č. 6 Automatizačná pyramída [56] 16
Obrázok č. 7 Architektúra PLC [31] 18
Obrázok č. 8 Certifikačná sada priemyselného počítača Unipi Axon 19
Obrázok č. 9 Ladder Diagram [37] 21
Obrázok č. 10 Function Block Diagram [37] 22
Obrázok č. 11 Structured Text [37] 22
Obrázok č. 12 Instruction List [37] 23
Obrázok č. 13 Sequential Function Chart [37] 24
Obrázok č. 14 Konvergencia IT a OT [66] 26
Obrázok č. 15 OPC UA ako prienik viacerých konceptov/technológií [76] . . . 29
Obrázok č. 16 Komunikačný model klient-server [76] 30
Obrázok č. 17 Postavenie OPC UA v rámci automatizačnej pyramídy [36] . . . 30
Obrázok č. 18 Komunikačný model klient-server *subscription* [12] 32
Obrázok č. 19 OPC UA a komunikačné protokoly [29] 33
Obrázok č. 20 Dve metódy komunikácie PubSub modelu [76] 34
Obrázok č. 21 Modely služieb cloud computingu [56] 36
Obrázok č. 22 Fungovanie systému Internet of Things [56] 39
Obrázok č. 23 Porovnanie spotrebiteľského a priemyselného IoT 41
Obrázok č. 24 Spôsoby komunikácie v IoT systéme 41
Obrázok č. 25 Edge computing v kontexte sieťovej komunikácie [92] 42
Obrázok č. 26 Porovnanie virtuálneho stroja a kontajnera [4] 45
Obrázok č. 27 Virtuálna, rozšírená a zmiešaná realita podľa The Foundry — príklad prevzatý z literatúry [46] 48
Obrázok č. 28 Virtuálno-realitné kontinuum podľa Milgrama a Kishiho [56] . 49
Obrázok č. 29 Reprezentácia rozšíreného sveta Augmented World [18] 50
Obrázok č. 30 Vrstvy Web of Things rozšírené na Web of Augmented Things [18] 51

Obrázok č. 31 Doplňujúce informácie o produkte podľa konceptu *rozšírených vecí* [80] . . . 52
Obrázok č. 32 Ovládanie mikrokontroléra Arduino pomocou Microsoft HoloLens [85] . . . 52
Obrázok č. 33 Ilustrácia funkcionality PTC Vuforia Studio [26] . . . 53
Obrázok č. 34 Ilustrácia testbedu pre Industry 4.0 [95] . . . 54
Obrázok č. 35 Testovanie rozšírenej a zmiešanej reality pre údržbu rezacieho stroja [56] . . . 56
Obrázok č. 36 Fungovanie komunikačného protokolu MQTT — príklad prevzatý z literatúry [40] . . . 60
Obrázok č. 37 Okno editora Node-RED a jeho hlavné komponenty [42] . . . 64
Obrázok č. 38 Paleta uzlov [42] . . . 65
Obrázok č. 39 Uzol Inject [42] . . . 66
Obrázok č. 40 Uzol Debug [42] . . . 66
Obrázok č. 41 Uzol Function [42] . . . 66
Obrázok č. 42 Uzol Change [42] . . . 67
Obrázok č. 43 Uzol Switch [42] . . . 67
Obrázok č. 44 Uzol Template [42] . . . 67
Obrázok č. 45 Predvolený pracovný priestor editora Node-RED [42] . . . 68
Obrázok č. 46 Stav uzla [42] . . . 68
Obrázok č. 47 Codesys IDE a runtime [6] . . . 70
Obrázok č. 48 Ukážka systému vo Factory I/O [98] . . . 71
Obrázok č. 49 Architektúra Ionic aplikácie [76] . . . 74
Obrázok č. 50 Rozhranie editora Unity [83] . . . 75
Obrázok č. 51 Panel s nástrojmi v rámci editora Unity [83] . . . 76
Obrázok č. 52 OpenPLC v prepojení s Node-RED a Microsoft Azure . . . 80
Obrázok č. 53 Schéma modelového výrobného systému [66] . . . 81
Obrázok č. 54 Celkový pohľad na virtuálnu výrobnú linku [66] . . . 82
Obrázok č. 55 Pásový dopravník [66] . . . 83
Obrázok č. 56 Zakrivený pásový dopravník [66] . . . 83
Obrázok č. 57 Zarovnávač [66] . . . 84
Obrázok č. 58 Kĺzačka [66] . . . 84
Obrázok č. 59 Neopracovaný materiál [66] . . . 85
Obrázok č. 60 Retroreflexný senzor a odrazka [66] . . . 85
Obrázok č. 61 Vysielač [66] . . . 86

Obrázok č. 62 Odstraňovač [66] . . . 86
Obrázok č. 63 Obrábacie centrum [66] . . . 87
Obrázok č. 64 Premenné v riadiacom programe [66] . . . 88
Obrázok č. 65 Power rails [66] . . . 89
Obrázok č. 66 Contacts [66] . . . 89
Obrázok č. 67 Coils [66] . . . 89
Obrázok č. 68 Set — reset [66] . . . 90
Obrázok č. 69 TOF [66] . . . 90
Obrázok č. 70 ADD [66] . . . 90
Obrázok č. 71 SUB [66] . . . 90
Obrázok č. 72 Rebríkový diagram — časť 1. [66] . . . 91
Obrázok č. 73 Rebríkový diagram — časť 2. [66] . . . 92
Obrázok č. 74 Sekvenčný diagram komunikácie [66] . . . 93
Obrázok č. 75 Factory I/O — Modbus klient [66] . . . 95
Obrázok č. 76 Factory I/O — Modbus klient 2 [66] . . . 96
Obrázok č. 77 Node-RED ako Modbus klient [66] . . . 97
Obrázok č. 78 Vytváranie OPC UA servera a jeho adresného priestoru v Node-RED [66] . . . 100
Obrázok č. 79 Funkcia na napĺňanie OPC UA servera hodnotami získanými pomocou Modbus protokolu z OpenPLC — *Function: Modbus to OPC UA namespace* [66] . . . 101
Obrázok č. 80 Ukážka OPC UA klienta UaExpert [66] . . . 102
Obrázok č. 81 Device connection groups [66] . . . 106
Obrázok č. 82 Dashboard v Azure IoT Central [66] . . . 107
Obrázok č. 83 Tok posielanie dát v Node-RED (teplota ovzdušia) [66] . . . 108
Obrázok č. 84 Tok v Node-RED zabezpečujúci pozastavenie a spustenie výrobného systému [66] . . . 108
Obrázok č. 85 Codesys v prepojení s OPC UA a MS Azure . . . 109
Obrázok č. 86 Codesys — deklarácia premenných [66] . . . 110
Obrázok č. 87 Codesys — rebríkový diagram [66] . . . 111
Obrázok č. 88 Sekvenčný diagram pre edukačnú prípadovú štúdiu č. 2 [66] . . . 112
Obrázok č. 89 Skenovanie siete [66] . . . 113
Obrázok č. 90 Konfigurácia premenných určených pre OPC UA server [66] . . . 114
Obrázok č. 91 OPC UA Client Factory I/O [66] . . . 115
Obrázok č. 92 Lokálny Node-RED — OPC UA klient [66] . . . 116

Obrázok č. 93 Node-RED v cloude [66] 118
Obrázok č. 94 Dashboard v Node-RED [66] 119
Obrázok č. 95 Schéma komunikácie jednotlivých častí prípadovej štúdie . . . 120
Obrázok č. 96 Priebeh komunikácie jednotlivých častí prípadovej štúdie . . . 121
Obrázok č. 97 Ukážka materiálov [98] 121
Obrázok č. 98 Schéma udalostného systému 122
Obrázok č. 99 Pneumatický triedič [98] 123
Obrázok č. 100 Difúzny senzor [98] 123
Obrázok č. 101 Vizuálny senzor [98] 123
Obrázok č. 102 Nastavenia pre vysielač (emitter) [98] 124
Obrázok č. 103 Senzory videnia [98] 124
Obrázok č. 104 Pohľad na virtuálny výrobný systém (uhol č. 1) [98] 125
Obrázok č. 105 Pohľad na virtuálny výrobný systém (uhol č. 2) [98] 125
Obrázok č. 106 Pohľad na virtuálny výrobný systém (uhol č. 3) [98] 126
Obrázok č. 107 Premenné riadiaceho programu [98] 128
Obrázok č. 108 Ukážka zápisu hodnoty do premennej [98] 129
Obrázok č. 109 Ukážka zápisu hodnoty do premennej [98] 129
Obrázok č. 110 Ukážka kódu na riadenie dopravníka a triedenie [98] 130
Obrázok č. 111 Riadiaci kód pre modrú vetvu [98] 130
Obrázok č. 112 Ukážka kódu slúžiaceho na inkrementáciu premenných [98] . 131
Obrázok č. 113 Objekt Symbol Configuration [98] 132
Obrázok č. 114 Zapnutie PLC servera [98] 133
Obrázok č. 115 Skenovanie siete a zvolenie zariadenia [98] 133
Obrázok č. 116 Párovanie premenných [98] 134
Obrázok č. 117 Príkazový riadok po spustení lokálneho Node-RED [98] 135
Obrázok č. 118 Prázdny Node-RED editor [98] 136
Obrázok č. 119 Nastavenie uzla Inject [98] 137
Obrázok č. 120 NodeId v programe UaExpert [98] 138
Obrázok č. 121 Uzol OpcUa Item s nastavením [98] 138
Obrázok č. 122 Nastavenie uzla OpcUa-Client a endpointu [98] 139
Obrázok č. 123 Prepojenie uzlov pre čítanie z OPC UA servera [98] 139
Obrázok č. 124 Načítanie správ a ich posielanie na MQTT broker (výstupy) [98] 140
Obrázok č. 125 Načítanie premenných a ich posielanie na MQTT broker (vstupy) [98] 140

Obrázok č. 126 Načítanie premenných a ich posielanie na MQTT broker (INT a riadiace premenné) [98] 141
Obrázok č. 127 Uzol Function s kódom na vytvorenie query [98] 142
Obrázok č. 128 Ukážka štruktúry databázy [98] 143
Obrázok č. 129 Tok pre načítanie dát a uloženie do databázy [98] 143
Obrázok č. 130 Ukážka webovej aplikácie — Automatic Control [98] 149
Obrázok č. 131 Ukážka webovej aplikácie — Manual Control [98] 150
Obrázok č. 132 Ukážka webovej aplikácie — System Model [98] 151
Obrázok č. 133 Ukážka prepojenia webovej aplikácie s virtuálnou továrňou [98] 151
Obrázok č. 134 Ukážka prepojenia webovej aplikácie s virtuálnou továrňou [98] 152
Obrázok č. 135 Spätná komunikácia s OPC UA Serverom v Node-RED [98] . . 153
Obrázok č. 136 Architektúra aplikačného systému štvrtej prípadovej štúdie . . 154
Obrázok č. 137 3D model skladového systému [76] 155
Obrázok č. 138 Vývojový diagram skladového systému [76] 157
Obrázok č. 139 Diagram prípadov použitia mobilnej aplikácie [76] 158
Obrázok č. 140 Sekvenčný diagram — prípad č.1 [76] 159
Obrázok č. 141 Sekvenčný diagram — prípad č.2 [76] 160
Obrázok č. 142 Zoznam globálnych premenných pre riadenie skladového systému vo Factory I/O [76] 162
Obrázok č. 143 Zoznam globálnych premenných pre prepojenie medzi manuálnym a automatickým riadením [76] 162
Obrázok č. 144 Pomocný program Control [76] 163
Obrázok č. 145 Hlavný program Main [76] 164
Obrázok č. 146 Makro pre Receiving Step [76] 165
Obrázok č. 147 Objekt Symbol Configuration [76] 169
Obrázok č. 148 Výber zariadenia [76] 170
Obrázok č. 149 Mapovanie senzorov a aktuátorov vo Factory IO na OPC UA premenné [76] 171
Obrázok č. 150 Implementácia komunikácie OPC UA — MQTT [76] 173
Obrázok č. 151 Testovanie komunikácie OPC UA — MQTT [76] 174
Obrázok č. 152 Komunikácia MQTT — OPC UA [76] 175
Obrázok č. 153 Obrazovky v mobilnej aplikácii 176
Obrázok č. 154 Testovanie komunikácie MQTT klient — MQTT broker [76] . . 180
Obrázok č. 155 Prepojenie skladového systému s mobilnou aplikáciou [76] . . 181
Obrázok č. 156 Obrazovky v mobilnej aplikácie v Android emulátore [76] . . . 182

Obrázok č. 157 Obrazovky na iOS zariadení [76] 183
Obrázok č. 158 Architektúra systému 185
Obrázok č. 159 RevPi Connect+ feat. CODESYS [69] 187
Obrázok č. 160 RevPi DIO [69] 187
Obrázok č. 161 Napájací zdroj MDR-60-24 na DIN lištu [69] 188
Obrázok č. 162 Fischertechnik dierovací stroj s dopravníkovým pásom [69] . . 189
Obrázok č. 163 Doska plošných spojov Fischertechnik [24] 189
Obrázok č. 164 Zapojenie laboratórneho systému [69] 190
Obrázok č. 165 Grafické rozhranie Raspberry Pi OS [69] 190
Obrázok č. 166 Rozhranie PiCtory [69] 192
Obrázok č. 167 CODESYS pripojenie na PLC [69] 193
Obrázok č. 168 CODESYS štruktúra projektu a zoznam zariadení [69] 194
Obrázok č. 169 CODESYS premenné [69] 194
Obrázok č. 170 CODESYS program ovládajúci laboratórny systém [69] 196
Obrázok č. 171 CODESYS OPC UA licencia [69] 198
Obrázok č. 172 Programový tok v Node-RED [69] 199
Obrázok č. 173 Node-RED uzol typu inject [69] 200
Obrázok č. 174 Uzol MQTT out [69] 201
Obrázok č. 175 Hodnoty v databáze [69] 203
Obrázok č. 176 Náhľad zobrazenia údajov v Unity [69] 204
Obrázok č. 177 Vizuálny skript na posúvanie objektu [69] 204
Obrázok č. 178 MQTT prijímanie správ v Unity [69] 205
Obrázok č. 179 Vloženie dát z MQTT brokera do popup okna [69] 206
Obrázok č. 180 Vývojový diagram aplikácie [69] 207
Obrázok č. 181 Obrazovky z Android aplikácie pre rozšírenú realitu [69] . . . 208
Obrázok č. 182 OPC UA klient — UaExpert [69] 209
Obrázok č. 183 Hodnoty v programe MQTTX [69] 209
Obrázok č. 184 Dekompozícia stavov na parciálne stavy [55] 212
Obrázok č. 185 Znázornenie situácie po spustení prechodu [55] 213
Obrázok č. 186 Schéma navrhovaného riešenia — implementácia Petriho siete do mikrokontroléra [56] 215
Obrázok č. 187 Schéma navrhovaného riešenia — implementácia Petriho siete do počítača [56] 215
Obrázok č. 188 Prostredie softvérovej aplikácie PN2ARDUINO [55] 217
Obrázok č. 189 Model Petriho siete parkoviska [9] 218

Obrázok č. 190 Schéma laboratórneho modelu parkoviska [9] 219
Obrázok č. 191 Schéma navrhovaného riešenia — Petriho siete v kombinácií s komunikáciou cez OPC UA [68] 220
Obrázok č. 192 Grafické rozhranie nadstavby PNEditora — nastavovanie hrany/miesta Petriho siete na spojitý typ [68] 221
Obrázok č. 193 Grafické rozhranie nadstavby PNEditora — pripojenie na OPC UA server [68] 221
Obrázok č. 194 Grafické rozhranie nadstavby PNEditora — prechod s podmienkou súvisiacou s premennou OPC UA adresného priestoru [68] . . 221
Obrázok č. 195 Názorná schéma systému pre monitorovanie a ovládanie mechatronických systémov s využitím rozšírenej reality [56] . . . 225
Obrázok č. 196 Rozpoznané mechatronické zariadenie s dynamicky generovaným GUI v rozšírenej realite [84] 228
Obrázok č. 197 Návrh výrobnej linky v softvérovej aplikácii Inventor: 1 — nakladací posuvník, 2 — prepravný pás, 3 — stojan na kamery v spojení s tunelom na kamery, 4 — smartfón s kamerou, 5 — triediaci mechanizmus [73] 231
Obrázok č. 198 Výrobná linka zostavená zo stavebnice LEGO — pohľad spredu: 1 — nakladací posuvník, 2 — prepravný pás, 3 — stojan na kameru v spojení s tunelom na kamery, 4 — triediaci mechanizmus, 5 — Arduino UNO, 6 — dvojitý H-most L298N, 7 — batériový blok, 8 — infračervený senzor prekážok TCRT5000, 9, 10 — servomotory DS04 360, 11 — servomotor FS90R [73] . . 231
Obrázok č. 199 Obslužná aplikácia s GUI vytvorená pomocou knižnice tried Windows Presentation Foundation [73] 232
Obrázok č. 200 Sieť OPC UA a metodický postup pri agregovaní nového zariadenia AAS do OPC UA servera 234
Obrázok č. 201 Komunikačná platforma agregačného servera OPC UA-ICP umožňuje širokú konektivitu 235
Obrázok č. 202 Interoperabilné digitálne dvojča a reálna linka tvoria „Industry 4.0 komponent" 236
Obrázok č. 203 Schéma funkcionality edukačno-vývojovej platformy pre virtuálnu a zmiešanú realitu [100] 238
Obrázok č. 204 Pohľad na výrobnú linku v prostredí zmiešanej reality [100] . . 238
Obrázok č. 205 Architektúra originálneho návrhu ovládania akčného člena hlasom [4] 239

Obrázok č. 206 Ovládací panel (HMI) s vizualizáciou hlasových povelov pre raziaci stroj s dopravným pásom [5] 240
Obrázok č. 207 17 cieľov udržateľného rozvoja podľa Organizácie spojených národov [72] 242
Obrázok č. 208 Paradigmy Industry 5.0 244
Obrázok č. 209 Ilustrácia využitia zmiešanej reality v priemysle (*generované umelou inteligenciou s pomocou modelu DALL-E*) 246
Obrázok č. 210 Open-source BCI zariadenie Ultracortex "Mark IV" EEG [71] . 247
Obrázok č. 211 Príklad funkcionalít digitálneho dvojčaťa zameraného na človeka 248
Obrázok č. 212 Ilustrácia spolupráce ľudského operátora s kolaboratívnym robotom (*generované umelou inteligenciou s pomocou modelu DALL-E*) . . . 249
Obrázok č. 213 Proces tvorby domu pomocou aditívnej výroby — špecializovanej 3D tlače [19] 251

Tabuľka č. 1 Premenné Modbus servera v OpenPLC Engine [43] 94
Tabuľka č. 2 Premenné v lokálnom Node-REDe [66] 118
Tabuľka č. 3 Vývojové prostredia [76] 175
Tabuľka č. 4 Frameworky pre vývoj mobilnej aplikácie [76] 175
Tabuľka č. 5 Mobilné zariadenie pre vývoj mobilnej aplikácie [76] 176
Tabuľka č. 6 Použité vstupné a výstupné piny 191
Tabuľka č. 7 Porovnanie dvoch konceptov riadenia s využitím Petriho sietí [56] 215

Zoznam programových modulov

1 Kód na pripojenie k databáze [98] 144
2 Ukážka dopytu na získanie údajov z databázy [98] 145
3 Funkcia v jazyku Javascript zabezpečujúca aktualizáciu údajov [98] 146
4 Pripojenie na MQTT broker a odoslanie správy [98] 147
5 Akcia AutomaticTransport_entry [76] 166
6 Akcia AutomaticTransport [76] 167
7 Akcia *ManualTransport_entry* [76] 167
8 Objekt connection [76] 178
9 Metóda *connect*() a *disconnect*() [76] 178
10 Metóda *observeMultiple*() [76] 179
11 Metóda *publishMessage*() [76] 179
12 Objekt *publishStartProcess* [76] 180
13 Lokálne premenné v Codesyse [69] 195
14 Ukážka funkcií v Node-RED [69] 199
15 Funkcia v Node-RED pre nastavenie globálnej premennej [69] 202
16 Funkcia v Node-RED pre kontrolu zmeny globálnej premennej [69] 202
17 Funkcia v Node-RED pre nastavenie INSERT SQL dopytu [69] 202

Zoznam skratiek a značiek

3D	trojdimenzionálny
AAS	administratívna schránka aktív (angl. Asset Administrative Shell)
AI	umelá inteligencia (angl. artificial intelligence)
AJAX	Asynchronous Javascript and XML
AMQP	Advanced Message Queuing Protocol
API	rozhranie pre programovanie aplikácií (angl. Application programming interface)
AR	rozšírená realita (angl. augmented reality)
ASCII	American Standard Code for Information Interchange
BCI	rozhranie mozog-počítač (angl. brain-computer interface)
CAD	angl. computer-aided design
CI/CD	Continues Integration/Continues Delivery
CIoT	Spotrebiteľský Internet vecí (angl. Consumer Internet of Things)
CoAP	Constrained Application Protocol
CPS	kyberneticko-fyzikálny systém (angl. cyber-physical system)
CRM	riadenie vzťahov so zákazníkmi (angl. Customer Relationship Management)
CSS	Cascading Style Sheets
ČVUT	České vysoké učení technické v Praze
DDS	Data Distribution Service
DES	diskrétny udalostný systém (angl. Discrete event system)
DIO	digitálny vstup a výstup (angl. digital input-output)
FBD	diagram funkčných blokov (angl. Function Block Diagram)
FEI	Fakulta elektrotechniky a informatiky
ERP	plánovanie podnikových zdrojov (angl. Enterprise Resource Planning)
EVP	edukačno-vývojová platforma
GB	gigabajt
HDMI	High-Definition Multimedia Interface
HMI	rozhranie človek-stroj (angl. human-machine interface)
HTML	HyperText Markup Language
IaaS	Infraštruktúra ako služba (angl. Infrastructure as a Service)
IBM	International Business Machines Corporation
ICP	Industrial Communication Platform

ID	identifikátor
IDE	vývojové prostredie (angl. Integrated Development Environment)
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IIoT	Priemyselný Internet vecí (angl. Industrial Internet of Things)
IoT	Internet vecí (angl. Internet of Things)
IoS	Internet služieb (angl. Internet of Services)
IPC	priemyselný počítač (angl. Industrial PC)
IKT	informačno-komunikačné technológie
IL	zoznam inštrukcií (angl. Instruction List)
IMC	angl. Internal Model Control
IT	informačné technológie
LD	rebríkový diagram (angl. Ladder Diagram)
LED	angl. light-emitting diode
LDS ME	Local Discovery Server with Multicast Extensions
LQ	L-lineárny systém, Q-kvadratický funkcionál; kvadraticky optimálny regulátor pre lineárny systém
LQG	kvadraticky optimálny regulátor pre lineárny systém, na ktorý pôsobia šumy s gaussovským rozdelením
JS	JavaScript
JSON	angl. JavaScript Object Notation
kB	kilobajt
M2M	komunikácia stroj-stroj alebo zariadenie-zariadenie (angl. machine-to-machine)
MB	megabajt
MQTT	Message Queuing Telemetry Transport
MR	zmiešaná realita (angl. mixed reality)
OOP	objektovo orientované programovanie
OPC DA	Object Linking and Embedding for Process Control Data Access
OPC UA	Open Platform Communications Unified Architecture
OPC UA FX	OPC UA Field eXchange
OS	operačný systém
OSI	Open Systems Interconnection
OT	operačné technológie
TCP	Transmission Control Protocol
TSN	angl. Time-Sensitive Networking
PaaS	Platforma ako služba (angl. Platform as a Service)

PGR	počítačom generovaná realita
PHP	PHP: Hypertext Preprocessor
PID	proporcionálny, integračný a derivačný (regulátor)
PLC	programovateľný logický kontrolér (angl. programmable logic controller)
PN	Petriho sieť (angl. Petri Net)
PubSub	angl. publish-subscribe
PWA	progresívna webová aplikácia
QoS	kvalita služieb (angl. Quality of Service)
QR	angl. quick response
REST	Representational State Transfer
RT	reálny čas (angl. real time)
RTU	Remote Terminal Unit
SaaS	Softvér ako služba (angl. Software as a Service)
SAS	Shared access signature
SCADA / MES	angl. Supervisory Control and Data Aquisition / Manufacturing Execution Systems
SDK	software development kit
SFC	sekvenčný funkčný diagram (angl. Sequential Function Chart)
SOA	servisne-orientovaná architektúra (angl. service-oriented architecture)
SQL	Structured Query Language
ST	štruktúrovaný text (angl. Structured Text)
STEM	veda, technológie, inžiniersto a matematika (angl. science, technology, engineering, and mathematics)
STU BA	Slovenská technická univerzita v Bratislave
UADP	Unified Access Data Plane
UDP	User Datagram Protocol
UI	používateľské rozhranie (angl. user interface)
USA	Spojené štáty americké (angl. United States of America)
USB	univerzálna sériová zbernica (angl. Universal Serial Bus)
ÚAMT	Ústav automobilovej mechatroniky
XR	angl. eXtended reality
VM	virtuálny stroj (angl. virtual machine)
VR	virtuálna realita

Úvod

Štvrtá priemyselná revolúcia, všeobecne známa aj ako Industry 4.0, umožnila zaviesť nielen v Európe, ale aj v celom svete metodiku na modernizáciu, oživenie a zmenu charakteru výrobných procesov a celého výrobného sektora. V súčasnosti táto zmena v priemyselnej paradigme dosiahla globálne rozmery a umožňuje transformovať tradičné výrobné procesy a továrne na moderné digitalizované továrne a integrované konkurencieschopnejšie a efektívnejšie podniky. Na dosiahnutie tohto cieľa je v súčasnosti vytváraná efektívna metodika prechodu od tradičných výrobných systémov na nové platformy, ktoré využívajú koncepty digitalizovanej počítačovej štruktúry s podporou riadiacich, informačných, komunikačných a rozhodovacích činností.

Digitalizácia a s ňou spojené nastupujúce inovatívne technológie sa stávajú neodmysliteľnými nástrojmi pre očakávanú a nutnú transformáciu priemyselných procesov. Tieto exponenciálne technológie umožňujú posilňovať výkon, zvyšovať kvalitu výroby i výrobkov a zabezpečiť flexibilitu celých výrob. Mnohé výrobné podniky sa preto stále viac a viac obracajú k inovatívnym riešeniam, ktoré sú schopné aplikovať a využiť potenciál digitálnych a inteligentných technológií.

Moderné výrobné systémy sú komplexné, inteligentné a integrované architektúry s podporou pokročilých inteligentných riadiacich, informačno-komunikačných a senzorických systémov. Základom modernizácie výrob s využitím konceptu Industry 4.0 sú kyberneticko-fyzikálne systémy (angl. *cyber-physical systems*). Využívajú mnohé moderné digitálne a inteligentné technológie, Internet of Things, Industrial Internet of Things, cloudové systémy, počítačom generovanú realitu (virtuálna, rozšírená a zmiešaná realita) atď.

Táto učebnica sa zaoberá vybranými digitálnymi a inteligentnými technológiami a edukačnými prípadovými štúdiami (aplikáciami) pre oblasť Industry 4.0 zameranými na monitorovanie a riadenie virtuálnych modelov výrobných systémov, ktoré svojim charakterom predstavujú diskrétne udalostné systémy. Monitorovanie a riadenie je podporované inovatívnymi prostriedkami, ako sú komunikačný štandard OPC Unified Architecture, komunikačný protokol MQTT, middleware Node-RED, webové, mobilné a cloudové aplikácie. Posledné spomínané prostriedky slúžia ako moderné, konfigurovateľné a škálovateľné rozhrania človek-stroj (HMI — human-machine interface). Moderné HMI riešenia prechádzajú z uzavretých technológií a jednoúčelových dotykových panelov na otvorené technológie, webové a mobilné aplikácie. Výrobné podniky experimentujú s novými cenovo dostupnejšími typmi programovateľných logických kontrolérov (PLC — programmable logic controller), ktoré sú oproti konvenčným modelom založené na open-source technológiách (napr. séria Revolution Pi, Controllino atď.).

V súčasnosti počítačové siete neslúžia len na prepojenie a pripojenie bežných počítačových zostáv tak, ako to bolo v minulosti. Aktuálne nadobudli nový zmysel, keď sa do nich začali pripájať vnorené (embedded) systémy a mobilné zariadenia. V dnešnej dobe sa tieto hranice posunuli až na úroveň pripájania jednotlivých senzorov a aktuátorov. Výsledkom je vznik novej paradigmy — Internet of Things (IoT), teda Internetu vecí. Mnohé popredné spoločnosti na Slovensku a v zahraničí hľadajú inovatívne metódy ovládania a diagnostiky zariadení pripojených do sietí Internet of Things. Nové možnosti otvoril rozvoj počítačom generovanej reality — najmä rozšírenej a zmiešanej reality. Preto sa jedna z uvedených edukačných prípadových štúdií venuje rozhraniu pre monitorovanie a ovládanie modelu výrobného systému v prostredí rozšírenej reality realizovanej na obrazovke tabletu.

Zámerom tejto učebnice je predstaviť moderné digitálne alebo inteligentné technológie na konkrétnych aplikáciách (edukačných prípadových štúdiách). Je určená širokému okruhu čitateľov, od bežných užívateľov až po odborníkov z priemyslu a služieb, ktorí majú záujem aplikovať uvedené progresívne technológie v praxi. Veríme, že každý čitateľ, ktorý sa zaoberá o predmetné oblasti, nájde v tejto učebnici cenné informácie, ktoré mu prinesú úžitok v jeho práci alebo v každodennom živote.

1 Štvrtá priemyselná revolúcia — Industry 4.0

Priemyselná činnosť je významnou hospodárskou a ekonomickou aktivitou človeka. Zohráva kľúčovú úlohu v rozvoji spoločnosti a pri vytváraní hodnôt. Priemysel predstavuje všetky činnosti, v priebehu ktorých dochádza k premene primárnych surovín, materiálov alebo iných produktov na výrobky, ktoré je možné charakterizovať ako nové. Výstupom výrobných procesov s využitím digitálnych a inteligentných technológií sú tak nové produkty. Môžu to byť výrobky, ktoré sú ďalej určení na spracovanie alebo výrobky určené na konečnú spotrebu. V dnešnej dobe je teda priemysel viac než len transformáciou surovín, ide o dynamický proces tvorby inovatívnych a konkurencieschopných riešení, ktoré prispievajú k celkovému rozvoju spoločnosti [78].

1.1 Priemyselné revolúcie — vývoj a trendy

Tak ako sa vyvíjala ľudská spoločnosť naprieč históriou, rozvíjala sa aj priemyselná výroba. Je možné tvrdiť, že dynamika priemyselného rozvoja, uplatňovanie výskumných a vedeckých poznatkov a technologického pokroku v priemyselných odvetviach, vždy určovala aj dynamiku a stupeň spoločenského rozvoja. Vo vývoji priemyslu možno identifikovať obdobia zásadných zlomov — revolúcií, charakterizovaných prívlastkami prvá, druhá, tretia a štvrtá. Revolúciou nazývame **hlbokú kvalitatívnu zmenu**, ktorá umožňuje prechod od starého usporiadania k novému usporiadaniu [78].

Priemyselná revolúcia, ktorá otvorila dvere pre dramatické zmeny v spoločnosti, v 18. storočí mala nesmierny dopad na spôsob výroby. Tento posun smeroval od ručnej (manuálnej) výroby v manufaktúrach k využívaniu moderných tovární so strojmi. Pre túto epochálnu transformáciu sa neskôr ustálil pojem **prvá priemyselná revolúcia**. Dôležitým míľnikom boli nové zariadenia poháňané vodnou a parnou energiou. James Watt a jeho vynález, parný stroj, sa stali symbolmi prvej priemyselnej revolúcie. S rozvojom nových technológií sa spoločnosť začala hromadne presúvať do miest, ktoré sa stávali centrami priemyselnej aktivity. Parný stroj so sebou nesie symboliku nielen technického pokroku, ale aj sociálnej transformácie. Sťahovanie obyvateľstva do urbanizovaných oblastí sa stalo nevyhnutnosťou, čo radikálne zmenilo spôsob života. Životné návyky a aktivity spojené s novým typom mestského života spojenom s prácou v továrni sa odlišovali od života, ktorý v tom čase ponúkal postupne sa vyľudňujúci vidiek

O 100 rokov neskôr, teda na prelome 19. a 20. storočia, prišlo obdobie **druhej priemyselnej revolúcie**, ktorá prebiehala necelých sto rokov. Elektrifikácia a motorizácia boli hlavnými hnacími silami tejto éry meniacej spôsob, akým ľudia žili a pracovali. Dôležitým míľnikom bol aj technologický rozvoj v chémii a vznik nových segmentov chemického priemyslu (najmä petrochémie). Tiež je nutné spomenúť vznik automo-

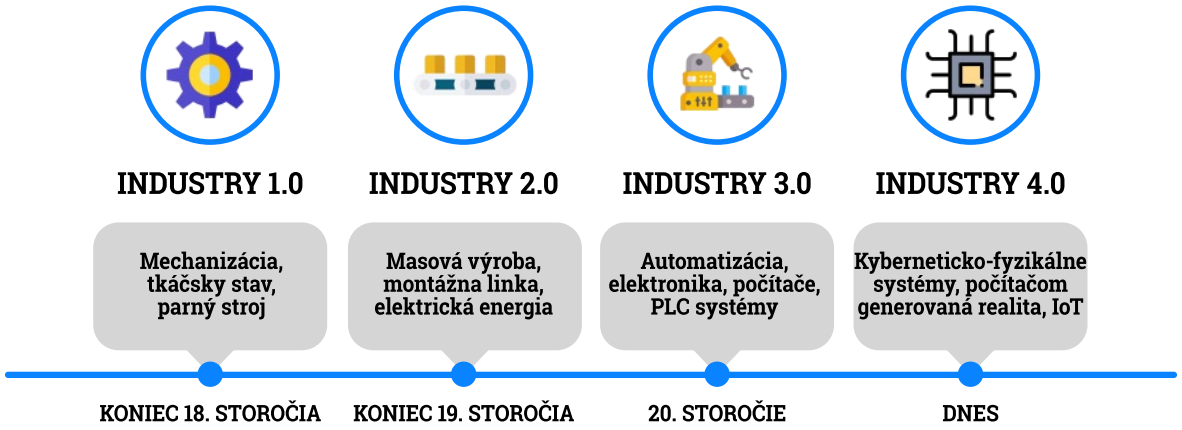
bilového priemyslu (vznik prvého benzínového motora). Okrem toho, druhá priemyselná revolúcia zanechala hlbokú stopu aj v oblasti letectva. Už v tejto dobe sa začali konštruovať motorové lietadlá, čo otvorilo dvere pre revolúciu v leteckom priemysle. Prvý automobil postavil Henry Ford v roku 1896, ktorý má zásluhu na konštrukcii prvej pohyblivej montážnej linke a pásovej výrobe. Na území dnešnej Českej republiky bol prvý český automobil — model President — zostavený v Kopřivnici až v roku 1898. S príchodom automobilov sa úzko spája aj návrh výrobných liniek. Prvé motocykle predstavili vynálezcovia Laurin a Klement v roku 1899. V oblasti kultúry je nutné spomenúť vznik kinematografie. Druhá priemyselná revolúcia predstavovala obdobie dynamických zmien a technologických inovácií, ktoré formovali vtedajší svet a jeho spoločnosť.

Za počiatok **tretej priemyselnej revolúcie** sa považuje koniec 60. a začiatok 70. rokov minulého storočia. Táto revolúcia je charakteristická rozvojom elektroniky a mikroelektroniky a využívaním výpočtovej techniky na automatické riadenie. Ide o obdobie, počas ktorého doznieval ekonomický boom po skočení druhej svetovej vojny a ktoré sa spájalo s masovou produkciou (ba až nadprodukciou) a masovou spotrebou. Tento proces súvisel s využívaním automatizácie a so zavádzaním riadiacich, informačných a elektronických systémov do priemyselných výrob. V súvislosti s týmito zmenami sa objavujú prvé štruktúry a algoritmy riadenia strojov a procesov, začiatok digitalizácie výrob (Duslo Šaľa, Třinecké železiarne, Slovnaft, Jaslovské Bohunice, Mochovce, Vojany, Nováky, Gabčíkovo atď.). Prebiehal tiež vývoj nových progresívnych materiálov s vysokou pevnosťou a húževnatosťou a vynikajúcimi chemickými, fyzikálnymi a mechanickými vlastnosťami. V záverečných etapách tejto revolúcie sa začínajú objavovať nové výrobné postupy (aditívne technológie — 3D tlač, rapid prototyping, reverzné inžinierstvo) a široká škála internetových služieb, IKT podpora výroby, zdieľanie informácií medzi ľuďmi. Prichádzajú alternatívne zdroje energie, pričom domy a autá sa stávajú malými elektrárňami využívajúcimi na svoj pohon alternatívne zdroje energie [33].

1.2 Súčasná priemyselná revolúcia Industry 4.0

Aktuálne prebiehajúci rozvoj priemyslu je identifikovaný ako **štvrtá priemyselná revolúcia — tzv. Industry 4.0**. Koncept Industry 4.0 (tiež Priemysel 4.0) je rozsiahlou systematickou zmenou, ktorá zásadným a rozhodujúcim spôsobom ovplyvňuje rozličné oblasti sveta práce [3]. V prípade Industry 4.0 nejde totiž o parciálne zavedenie nových technológií do výroby s následným postupným prispôsobovaním fungovania jednotlivých nadväzujúcich systémov, ale o množinu nových technológií a foriem aplikácie s rôznym stupňom technickej vyspelosti a systémových účinkov; pomenúva veľké zmeny prudko vstupujúce do súčasného priemyslu. Nositeľom týchto zmien je najmä digitalizácia, automatizácia, mechatronizácia a integrácia informačných a komunikačných technológií

na všetkých úrovniach riadenia podnikových procesov a služieb. Súčasná priemyselná revolúcia je teda širokospektrálna a vyznačuje sa integráciou vedomostí a skúseností z predchádzajúcich revolúcií (obrázok č. 1).



Obrázok č. 1: Priemyselné revolúcie a ich ťažiskové aspekty [56]

Štvrtá priemyselná revolúcia Industry 4.0 je dominantne založená na **rozvoji a integrácii nových progresívnych informačno-komunikačných technológií** (IoT, výpočtová a umelá inteligencia, virtuálna, rozšírená a zmiešaná realita, kontajnerizácia...), ktoré boli ešte nedávno výhradne súčasťou sektora služieb a zábavy, **do priemyselných aplikácií**.

Koncept Industry 4.0 prináša nový pohľad na organizáciu a riadenie komplexných výrob s využitím integrovaného multidisciplinárneho prístupu. Základom Industry 4.0 sú tzv. **kyberneticko-fyzikálne systémy**. Vyznačujú sa využívaním informačno-komunikačných technológií na monitorovanie a riadenie fyzických procesov a systémov [51].

Štvrtá priemyselná revolúcia (Industry 4.0) je postavená na týchto **atribútoch** alebo paradigmách (obrázok č. 2): [33]

- **Interoperabilita** — vo všeobecnosti predstavuje možnosť a schopnosť inteligentných strojov a ľudských bytostí spolu komunikovať a vymieňať si informácie tak, že sú navzájom kompatibilné. Koncept Industry 4.0 je v digitálnom svete primárne založený na výmene dát medzi rôznymi technologickými systémami a zariadeniami spôsobom označovaným ako M2M (angl. Machine to Machine) alebo niekedy tiež Industrial IoT. Kompatibilná komunikácia a výmena dát je základným predpokladom v Industry 4.0 na to, aby obrovské množstvo rozmanitých systémov od jednoduchých senzorov, cez komplexné stroje, riadiace systémy, HMI, servery až po cloudy mohli bez problémov digitálne spolupracovať s porozumením. Interoperabilita umožňuje týmto systémom efektívnu spoluprácu bez ohľadu na ich

pôvod, výrobcu, platformu alebo programovací jazyk. Interoperabilita sa dosahuje komplexnou štandardizáciou vo viacerých aspektoch. Pre Industry 4.0 bol globálne prijatý štandard OPC Unified Architecture (OPC UA) ako kľúčový prostriedok pre interoperabilnú výmenu dát.

- **Virtualizácia** — je schopnosť vytvoriť digitálny model, ktorý čo najvernejšie zobrazuje reálny objekt alebo významné vlastnosti zariadenia. Vzájomným prepojením reálnych zariadení a ich virtuálnych kópií môžeme poskladať celú inteligentnú továreň. Pri virtualizácii sa používajú aj reálne údaje získané zo snímačov strojov a zariadení pri ich prevádzke. Tento aspekt sa úzko spája s termínom digitálne dvojča. Digitálne dvojčatá môžu byť použité na simuláciu digitálnych modelov v reálnom čase a optimalizáciu procesov počas celého životného cyklu zariadení alebo produktov a tak zvyšovať efektívnosť výroby. Súčasťou virtualizácie sú aj moderné HMI, ktoré poznáme pod pojmami virtuálna realita (VR), rozšírená realita (AR) a zmiešaná realita (MR).

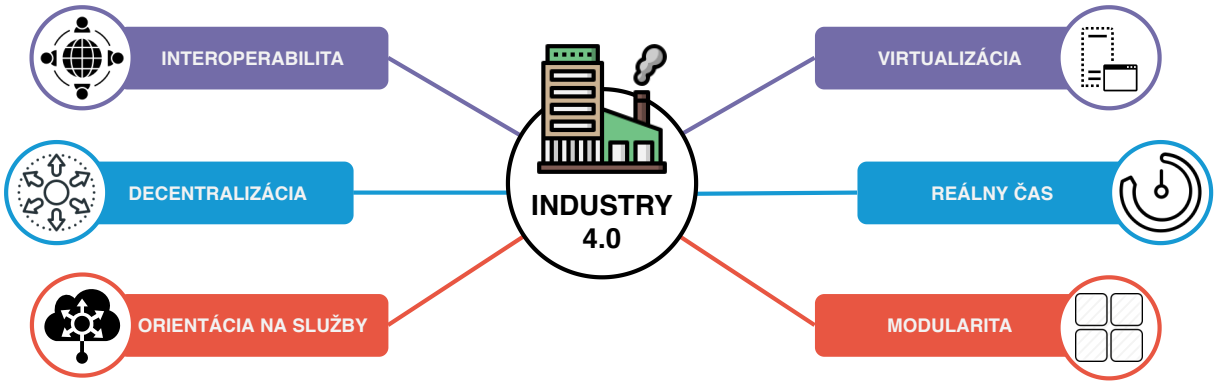
- **Decentralizácia** — je možnosť a schopnosť každého stroja realizovať operácie a decentralizované (autonómne) riadenie a realizovať maximálne kvalifikované rozhodnutia, ktoré smerujú k optimalizácii výroby. Decentralizácia teda znamená distribúciu rozhodovacích procesov a zodpovedností na rôzne úrovne výrobného reťazca. To umožňuje rýchlejšie a flexibilnejšie rozhodovanie dôležité v prostredí, kde sa rýchlo menia podmienky a požiadavky.

- **RT (real-time) činnosť** — pre inteligentné riadenie výroby je nevyhnutné zbierať dáta a analyzovať ich v reálnom čase. Na základe zozbieraných informácií je možné v reálnom čase rekonfigurovať výroby, zohľadniť poruchy a nájsť riešenia, napríklad na poruchu zariadenia, resp. presunutie výroby na iné miesto a podobne. Kombinácia OPC UA a TSN (angl. Time-Sensitive Networking) umožňuje presný a spoľahlivý prenos dát v reálnom čase a nízku latenciu v celom výrobnom prostredí.

- **Orientácia na služby** — zahŕňa komunikáciu a výmenu informácií cez Internet, poskytovanie informácií iným stranám a podobne. Koncept Industry 4.0 je orientovaný nielen na poskytovanie služieb a hodnôt nielen výrobkom, ale aj na služby, ktoré sú poskytované počas celého životného cyklu výrobku.

- **Modularita** — je schopnosť inteligentného podniku pružne sa adaptovať na situáciu vo výrobe zmenou softvérových a hardvérových modulov, ich zdieľaním a rekonfiguráciou. Modulárny prístup v Industry 4.0 umožňuje vytváranie flexibilných a

ľahko prispôsobiteľných systémov. Rôzne moduly a komponenty môžu byť kombinované a prispôsobované podľa aktuálnych potrieb výroby. Táto modularita zjednodušuje údržbu, aktualizácie a rozšírenie výrobných systémov. Dôležitou technológiou v tomto smere je kontajnerizácia (napr. s využitím Dockeru).



Obrázok č. 2: Atribúty Industry 4.0 [56]

1.3 Kľúčové technológie pre Industry 4.0

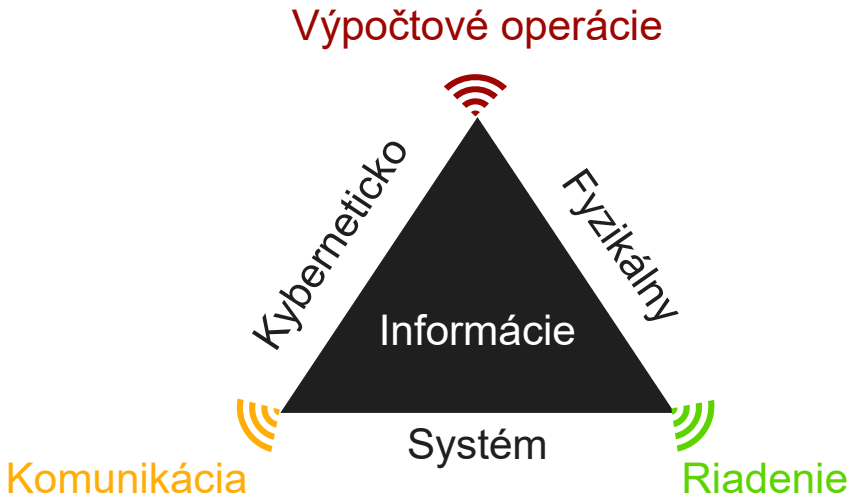
Základ konceptu Industry 4.0 tvoria kyberneticko-fyzikálne systémy — fabriky budúcnosti (tzv. **digitálne továrne**). Pojem kyberneticko-fyzikálne systémy (obrázok č. 3) vyjadruje, že všetko vo výrobnom procese by malo byť vytvorené tak, aby výrobky, zariadenia a ľudia navzájom komunikovali a vzájomne si vymieňali (pokiaľ možno bezdrôtovo) informácie a príkazy. Výrobok v sebe už od začiatku výroby ponesie „digitálne informácie“, čo mu umožní v každom štádiu výroby komunikovať s okolím, čím sa stáva kyberneticko-fyzikálnym prvkom systém, ktorý zabezpečí prepojenie virtuálneho sveta s realitou [56].

Z predchádzajúceho textu vyplýva, že v ére Industry 4.0 sú v digitálnej továrni kľúčové tri komponenty: [51]:

1. **vybavenie procesov (smart) snímačmi**, ktoré neustále automaticky zbierajú dáta zo strojov, zariadení, výrobných pásov, skladov, materiálov a iných komponentov vo výrobe;

2. **komunikačná infraštruktúra**, cez ktorú dokážu stroje komunikovať medzi sebou, s ľuďmi, aj s rôznymi informačnými systémami;

3. **využívanie softvéru**, ktorý dokáže všetky dáta nielen zhromažďovať, ale následne ich aj vyhodnocovať, analyzovať a vyvodzovať z nich závery — inými slovami, softvér vnáša do zhluku dát inteligenciu a poriadok.

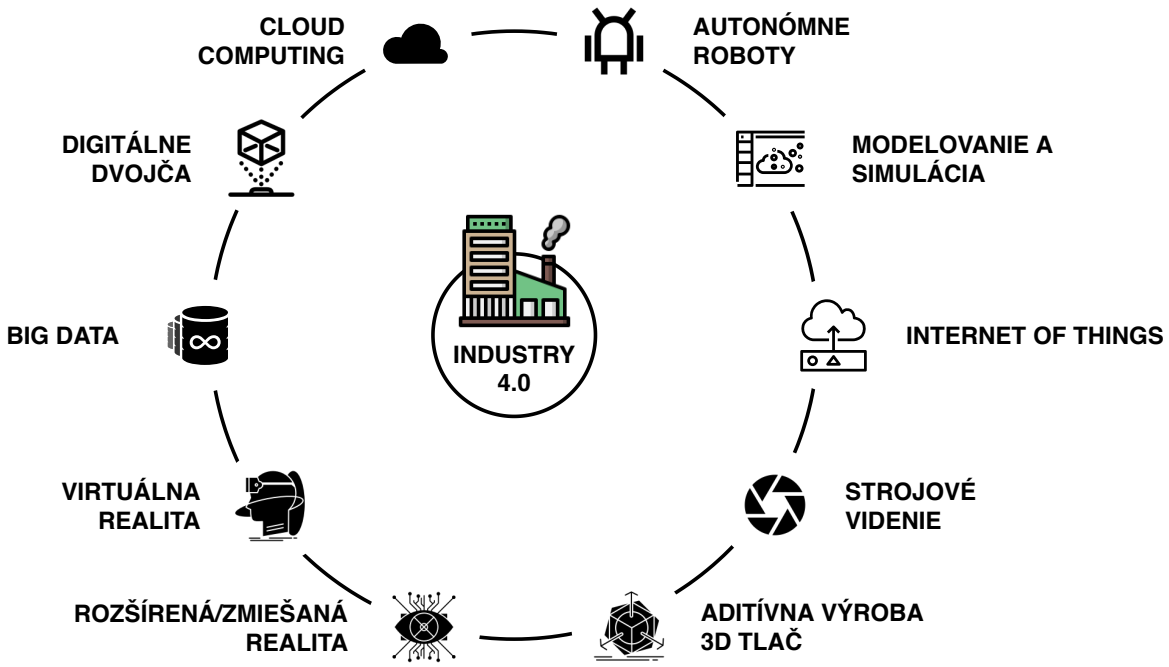


Obrázok č. 3: Kyberneticko-fyzikálny systém (angl. cyber-physical system)

Industry 4.0 tak závisí od rozvoja celého počtu nových a inovatívnych **digitálnych alebo inteligentných technológií** (obrázok č. 4) a konceptov, ako sú [33]:

- **aplikácie informačno-komunikačných technológií** v procese digitalizácie informácií a integrovania systémov na všetkých úrovniach vytvárania produktov a celého výrobného cyklu vrátane logistiky;
- **kyberneticko-fyzikálne systémy**, ktoré využívajú informačno-komunikačné technológie na monitorovanie a riadenie fyzických procesov a systémov;
- **sieťová komunikácia zahrňujúca bezdrôtové a internetové technológie**, ktoré slúžia na prepojenie strojov, systémov a ľudí tak v rámci továrne, ako aj s dodávateľmi a distribútormi;
- **modelovanie, simulácia a virtualizácia** návrhu dizajnu výrobkov a výrobných procesov;
- **zhromažďovanie veľkého množstva dát a ich analyzovanie a využívanie**, a to buď priamo v továrni alebo pomocou analýzy veľkých súborov dát a cloud computingu – Big Data;
- **väčšej podpory pracovných síl prostredníctvom informačno-komunikačných technológií**, vrátane rozšírenej a zmiešanej reality, inteligentných nástrojov či robotov.

Prienik digitálnych technológií do procesov v Industry 4.0 je v tejto učebnici zameraný najmä na nasledovné technológie a koncepty. Pomocou nich boli realizované originálne aplikácie vo forme edukačných prípadových štúdií, ktoré sú opísané v ďalších



Obrázok č. 4: Digitálne technológie pre zefektívnenie procesov v Industry 4.0 [56]

kapitolách.

Cloudové technológie umožňujú zber, ukladanie a analytické spracovanie rozsiahlych súborov dát (Big Data). Ďalej umožňujú pracovať so sémantikou a ontológiami. Analýzou je možné detegovať „neviditeľné" procesy a vlastnosti. Cloud computing je pojem pre využívanie softvéru alebo hardvéru formou služieb prostredníctvom Internetu. V súčasnosti ide o jeden z najdôležitejších trendov, na ktorý prechádzajú aj najväčší poskytovatelia podnikových softvérových služieb. Cloudové prostredie je známe svojou flexibilitou a schopnosťou rýchlo reagovať na meniace sa potreby. Priemyselné spoločnosti môžu využívať cloudové služby na požiadanie, čím minimalizujú investície do vlastného hardvéru a zároveň získavajú prístup k výkonným výpočtovým zdrojom a technológiám.

Internet of Things umožňuje prepojenie jednotlivých zariadení prostredníctvom Internetu bez aktívnej účasti človeka. Zariadenia môžu byť napríklad roboty, automobily, domáce spotrebiče, nositeľná elektronika, ba dokonca len samotné aktuátory a senzory, ktoré si spolu vymieňajú informácie a spolupracujú. Industrial Internet of Things (IIoT), teda priemyselný Internet vecí, sa zaoberá prepojením priemyselných zariadení pre zefektívnenie automatizácie a riadenie výrobných procesov. Existencia Internet of Things je umožnená okrem iného vďaka miniaturizácii, znižovaniu spotreby a zároveň aj ceny čipov a bezdrôtových technológií, ktoré sa tak môžu zaobísť bez veľkej batérie a komunikujú navzájom s veľmi malou spotrebou. IIoT umožňuje implementáciu prediktívnej

údržby, čo znamená, že na základe zhromaždených dát je možné predvídať, kedy a akým spôsobom by mohlo dôjsť k poruche zariadenia. Týmto spôsobom je možné minimalizovať neplánované výpadky a zvýšiť dostupnosť výrobných liniek.

Ďalšou z technológií je **počítačom generovaná realita (PGR)** (angl. extended reality), do ktorej zaraďujeme virtuálnu, rozšírenú a zmiešanú realitu. PGR, ako moderná technológia, je v Industry 4.0 využitá pri virtualizácii efektívneho projektovania optimálnych výrobných štruktúr a pracovných operácií s ich efektívnym ergonomickým vyhodnotením a návrhom. V digitálnych podnikoch sa v súčasnosti hľadajú nové formy monitorovania, riadenia, diagnostiky a vizualizácie procesov. Počítačom generovaná realita takéto formy prináša, pričom sledovať počítačom generovanú realitu je možné buď použitím kompatibilného headsetu alebo mobilného zariadenia (smartfón, tablet). Rozšírená a zmiešaná realita sa objavujú v rôznych oblastiach: zábava, reklama (vizualizácie nábytku a iných produktov, hry), výučba (rozšírenie papierovej učebnice o modely, vizualizácie fyzikálnych javov, anatómie organizmov), postup skladania výrobkov (detské stavebnice, nábytok). Pre výcvik hasičov a armády sa využíva umiestnenie virtuálnych cieľov do reálneho prostredia. Ďalšou doménou je automobilový priemysel (projekcia informácií na čelné sklo). V návrhu nových výrobkov sa dnes už veľmi často využíva najmä virtuálna realita. Rozšírená a zmiešaná realita nachádza v digitalizovaných výrobách svoje miesto najmä pri diagnostike, riadení a tiež napríklad aj pri vzdialenej údržbe.

Vyššie uvedené digitálne technológie je možné chápať a nazývať aj ako **inteligentné technológie**. V tomto prípade je dôležité uviesť vysvetlenie, čo toto označenie znamená. Inteligentné technológie možno chápať ako špeciálnu kategóriu digitálnych technológií, ktoré v sebe nesú určitú formu inteligencie. V širšom zmysle inteligentné technológie sú koncepty alebo riešenia, ktoré využívajú pokročilé algoritmy, strojové učenie, výpočtovú či umelú inteligenciu alebo iné metódy na analýzu dát a predvídanie trendov, aby sa mohli samostatne rozhodovať, adaptovať na zmeny v prostredí alebo optimalizovať procesy bez potreby neustálej ľudskej interakcie — t. j. vykonávať **autonómne akcie**. Takisto je možné uviesť, že inteligentné technológie by mali dokázať riešiť zložité a komplexné problémy, ktoré by človek riešiť nezvládol, a to s vysokou rýchlosťou a pohotovosťou reakcií, spoľahlivosťou a funkčnou bezpečnosťou, či dokonca zabezpečujú náhradu sofistikovaných senzoricko-motorických funkcií človeka. Inteligentné technológie sú kľúčové pre automatizáciu, zvýšenie efektivity a inovácie v mnohých odvetviach. Cloudové technológie poskytujú flexibilnú a škálovateľnú výpočtovú kapacitu a úložisko. Práve možnosť škálovania je jeden z pilierov cloud computingu. Inteligencia týchto technológií spočíva v ich schopnosti dynamicky prerozdeľovať zdroje na základe požiadaviek aplikácií, optimalizovať výkon a bezpečnosť a automaticky sa prispôsobiť

meniacim sa podmienkam siete a aplikácií. Zariadenia a sieťové prvky v rámci konceptu Internet of Things môžeme považovať za inteligentné v tom prípade, ak umožňujú monitorovať, zbierať, vymieňať si a analyzovať dáta autonómne, čím umožňujú inteligentné rozhodovanie a vykonávanie akcií bez ľudskej intervencie. Príkladom sú pokročilé aplikácie prediktívnej údržby. V prípade virtuálnej, rozšírenej a zmiešanej reality sa využívajú pokročilé algoritmy na spracovanie obrazu, lokalizáciu a rozpoznávanie objektov, priestorové mapovanie, aby boli používateľom sprostredkovanú presvedčivé a interaktívne zážitky. V kontexte Industry 4.0 vyššie uvedené technológie predstavujú stavebné bloky pre vytvorenie inteligentných, autonómnych systémov schopných optimalizácie výrobných procesov, zvyšovania efektivity a inovácie produktov. Ich schopnosť adaptovať sa na zmeny, spracúvať a analyzovať dáta v reálnom čase, a autonómne rozhodovať a konať, ich činí nielen digitálnymi, ale aj inteligentnými technológiami.

V koncepte Industry 4.0 začínajú hrať významnú úlohu aj **open-source technológie** v oblasti **riadenia procesov** a **rozhraní človek-stroj** HMI (human-machine interface). Ešte nedávno bolo nepredstaviteľné, že by riadenie rozsiahlejších výrobných systémov bolo realizované iným spôsobom ako prostredníctvom striktne uzavretých PLC od veľkých výrobcov a HMI malo inú podobu ako priemyselný panel (často výhradne od rovnakého výrobcu ako PLC). S nástupom moderných konceptov výroby sa začínajú presadzovať open-source PLC (napríklad Revolution Pi alebo UniPi), ktoré sa vyznačujú väčšou flexibilitou, nižšou cenou a zároveň si zachovávajú špecifické vlastnosti nutné pre priemyselné prostredie (odolnosť voči náročnejším podmienkam, real-time činnosť atď.). V oblasti HMI sa tiež objavuje trend smerujúci k otvorenejším riešeniam. HMI je možné v dnešnej dobe realizovať nielen na priemyselnom paneli, ale aj vo forme webovej či mobilnej aplikácie, ba dokonca v počítačom generovanej realite.

2 Automatizačná pyramída a technické prostriedky automatického riadenia

V tejto časti učebnice sú uvedené základné informácie o riadení systémov v digitalizovaných výrobách so zameraním na diskrétne udalostné systémy.

2.1 Automatické riadenie a diskrétne udalostné systémy

Problematika automatického riadenia už viac než 70 rokov predstavuje jednu z najdynamickejších vývojových a aplikačných oblastí výrazne ovplyvňujúcich kvalitu života vo všetkých oblastiach ľudskej činnosti. Automatizácia je chápaná ako súbor metód, stratégií, procesov a implementácií technických a programových prostriedkov a nástrojov schopných splniť požadované ciele nezávisle bez väčších interakcií človeka, t. j. automaticky. Základom automatizácie procesov sú metódy a algoritmy **riadenia**. *Automatické riadenie predstavuje cieľavedomú činnosť, ktorej hlavným cieľom je také ovplyvňovanie vstupov riadeného procesu, aby bolo zabezpečené požadované chovanie sa systému* (*sledovanie referenčnej premennej, žiadanej hodnoty a pod.*) [50]. **Systém** chápeme ako „súhrn“ vzájomne pôsobiacich komponentov tvoriacich časť prostredia, ktorú môžeme od okolia oddeliť pomocou fyzickej alebo myšlienkovej hranice [32].

Vývoj systémov riadenia predstavuje komplexnú disciplínu, ktorá zahŕňa rôzne oblasti (modelovanie, riadenie, optimalizácia, simulácia atď.). Pozostáva zo základných krokov, ktorými sú špecifikácia požadovaných vlastností, návrh algoritmu riadenia, jeho implementácia a testovanie. Keďže v každom z krokov ide vo všeobecnosti o náročné a dôležité operácie, je vhodné a potrebné vytvoriť **model** riadeného systému [91]. Táto učebnica sa zaoberá **automatickým** riadením.

V závislosti od cieľov riadenia a dynamiky riadenie procesu je možné pri riadení možné využiť rôzne typy štruktúr riadenia. Podmienkou správnej činnosti systémov riadenia je zabezpečenie **stability** a požadovanej **kvality** riadenia. Základnou štruktúrou riadenia je **jednoduchý spätnoväzbový regulačný obvod**. Na zabezpečenie cieľov riadenia pri zložitejších systémoch sa využívajú komplexnejšie štruktúry, ktorých základom je kombinácia **priamoväzbového** a **spätnoväzbového** riadenia, **kaskádne** štruktúry riadenia sú realizované ako sériové usporiadanie dvoch alebo viacerých spätnoväzbových štruktúr.

Metódy automatického riadenia rozdeliť na dva základné typy, a to na: **konvenčené metódy** (na báze PID algoritmov) a **moderné metódy automatického riadenia** (LQ, LQG, prediktívne a robustné riadenie a pod.). Konvenčné metódy na báze PID sa využívajú aj v súčasnosti a sú neustále modernizované a dopĺňané o princípy robustnosti, optimalizácie a inteligencie. Medzi takéto metódy patria napríklad metódy s vnútorným

modelom (IMC — angl. Internal Model Control), metódy využívajúce algebraický polynomiálny prístup, metódy priamej syntézy a podobne. Moderné prístupy a metódy riadenia umožňujú zohľadniť v riadiacich zákonoch neurčitosti modelu, vplyv porúch, ohraničenia riadiacich zásahov a jeho prírastkov, ako aj dopravné oneskorenia. Ďalšie prístupy využívajú metódy výpočtovej inteligencie na báze fuzzy logiky, umelých neurónových sietí a genetických algoritmov [50].

Automatické riadenie môžeme technicky uskutočniť viacerými spôsobmi:

- **Logické (on/off) riadenie** — využíva k riadeniu dvojhodnotové veličiny riadiacej premennej, ktorej pôsobenie využíva len dve možnosti akčného zásahu (napr. ventil je otvorený / zatvorený a pod.). Podobne aj informácia o stave objektu je dvojhodnotová veličina (napr. hladina je nad / pod minimálnou hodnotou, teplota je nad / pod 10°C a pod.). Dvojhodnotové veličiny sú formálne vyjadrované hodnotami 0 a 1 a sú analogické s premennými výrokovej logiky, a preto sú vzťahy medzi premennými nazývané logické funkcie a riadiace obvody pracujúce na tomto princípe sú logické riadiace obvody.

- **Spojité riadenie** — predstavuje výpočet riadiaceho zásahu na základe výstupov alebo stavov riadeného systému, ktoré sú veličinami spojitými v čase. Tieto časovo-spojité veličiny môžu byť získané meraním alebo simulačne z modelu systému.

- **Diskrétne riadenie** — riadiaci systém je realizovaný ako diskrétny algoritmus implementovaný na počítači (PLC, mikrokontrolér a pod.). V tomto prípade je nutné spojitý signál previesť na diskrétny, teda diskretizovať. Diskrétne systémy delime na časovo-diskrétne (angl. discrete-time) a udalostné (angl. discrete-event) systémy.

  - **Časovo-diskrétne (angl. discrete-time) riadenie** — diskretizovaná spojitá veličina je reprezentovaná postupnosťou diskrétnych hodnôt (impulzov) snímaných v časových intervaloch spravidla rovnakej dĺžky (perióda vzorkovania). Medzi okamihmi vzorkovania nie je regulovaná veličina meraná a upravovaný nie je ani riadiaci zásah. Veľkosť periódy vzorkovania závisí od dynamiky riadeného systému a jej určenie je kľúčové pre správnu činnosť riadiaceho systému (pozri Shannonovu-Kotelnikovovu teorému o vzorkovaní).
  - **Udalostné (angl. discrete-event) riadenie** — riadenie udalostných systémov zahŕňa monitorovanie, detekciu, spracovanie a reakciu na udalosti v systéme. Udalosťou môže byť čokoľvek, čo má význam pre systém, napríklad chyba v systéme, transakcia v databáze alebo používateľská interakcia.

- **Hybridné riadenie** — ide o všetky kombinácie spojitého, časovo-diskrétneho, udalostného a riadenia.

- **Fuzzy (expertné) riadenie** — je taký druh inteligentného riadenia, ktorý na základe znalosti experta umožňuje riadiť aj zložité procesy, ktoré sa nedajú jednoducho opísať matematickým modelom. Expert riadi proces na základe skúseností, ktoré sú vyjadrené pravidlami typu *ak-potom* (ak hladina pomaly klesá, tak otvor trochu prívod vody a pod.). Fuzzy regulátor priradí hodnotám vstupnej veličiny hodnotu jazykovej premennej, a to pomocou tzv. funkcie príslušnosti (fuzzifikácia). V ďalšom kroku určí fuzzy regulátor na základe bázy pravidiel a zvolenej metódy vyvodzovania (inferencia) jazykové hodnoty výstupnej veličiny (riadiaceho zásahu). Podľa zvolenej metódy inferencie dostávame priamo ostrú (angl. crisp) číselnú hodnotu výstupu alebo je potrebné previesť jej jazykové vyjadrenie na konkrétnu číselnú hodnotu výstupu (defuzzifikácia).

Logické a diskrétne riadenie je možné realizovať na jednom **programovateľnom logickom kontroléri** (**resp. počítači**) — **programmable logic computer (PLC)** alebo pomocou **mikrokontroléra**. Dnes sa vo väčšine prípadov spojité riadenie realizuje s využitím výpočtovej techniky s použitím malej periódy vzorkovania (vzhľadom na dynamiku riadeného procesu).

Je potrebné tu zdôrazniť **systémový prístup k automatizácii**. Riešenie problémov automatizácie zasahuje do rôznych odborov a je nutné riešiť ho komplexne. Vymenujme niektoré problémy, ktoré sa riešia pri zavádzaní automatizácie do praxe:

- rozhodovanie o účelnosti automatizácie v danej oblasti;

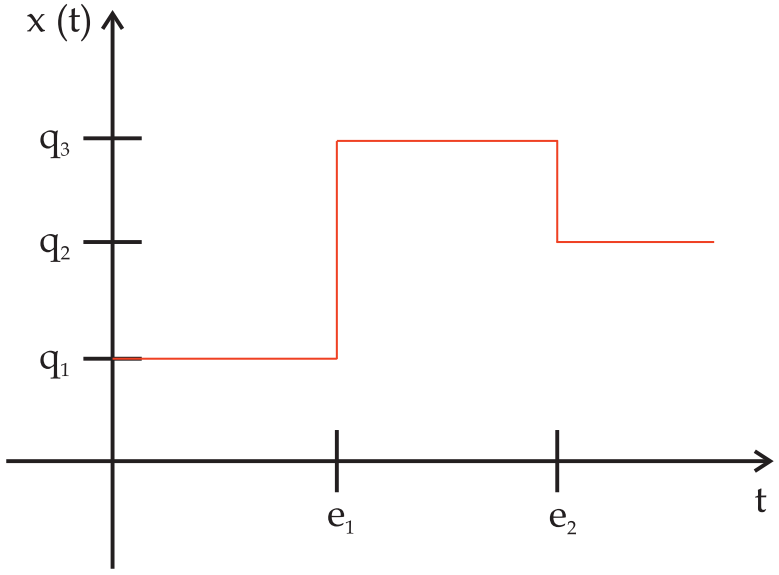
- riešenie technickej realizácie automatizácie a použitie vhodných technických prostriedkov;

- nasadenie počítačov a otázka ich programového vybavenia;

- sociálne a ekonomické aspekty automatizácie.

Špecialista, ktorý sa zaoberá automatizáciou, musí mať aspoň základné znalosti z automatického riadenia, prostriedkov automatického riadenia, simulácie systémov, meracích systémov, základy elektroniky a elektrotechniky a ďalších oblastí. V dnešnej dobe počas štvrtej priemyselnej revolúcie sa k tomuto zoznamu pridávajú aj znalosti programovania a digitálnych technológií. Iba s týmito znalosťami je možné pristupovať k zavádzaniu automatizácie na rôznych pracoviskách a dosiahnuť to, aby prostriedky vynaložené na zavádzanie automatizácie boli využité efektívne.

Všeobecná a matematická teória systémov sa zaoberá takými systémami, ktoré sú dobre opísané matematickými rovnicami (analytické modely), vzťahmi z fyzikálnej mechaniky, teóriou obvodov, fyzikálnou chémiou a v rámci spomenutých vedných disciplín veličinami, ktoré sú spravidla definované na **spojitom stavovom priestore**. Prostredníctvom spojitých veličín sú definované spojité systémy, ktoré sú opísateľné diferenciálnymi rovnicami. Pre systémy, kde je vývoj ich stavu vzorkovaný v diskrétnych časových úsekoch a veličiny systému nadobúdajú hodnoty z konečných množín, používame označenie **diskrétne systémy**. Takéto systémy dokážeme opísať diferenčnými rovnicami. Slovo diskrétny nedáva v tomto prípade jednoznačnú informáciu, nakoľko ide o spojité systémy, avšak ich stavy v istých časoch vzorkujeme a hodnoty veličín aproximujeme hodnotami z konečných množín. Tento postup znamená diskretizáciu spojitého systému. Možno si však všimnúť, že vo svete okolo nás má mnoho reálnych systémov **stavy**, ktoré majú vo svojej podstate **diskrétny charakter**. Ide napríklad o počet výrobkov v sklade, indikáciu vypnutia / zapnutia svetla a pod. Ďalej si treba všimnúť, že veľa týchto systémov je riadených udalosťami v diskrétnom čase (napr. stlačenie tlačidla), tzn. sú **udalostne-aktivované (angl. event-driven)**. Preto takéto systémy nazývame **diskrétne udalostné systémy (DES — discrete event systems)**. Primárnym aspektom aktivácie systému sú udalosti a nie čas, diskrétny čas má len evidenčný charakter [54].



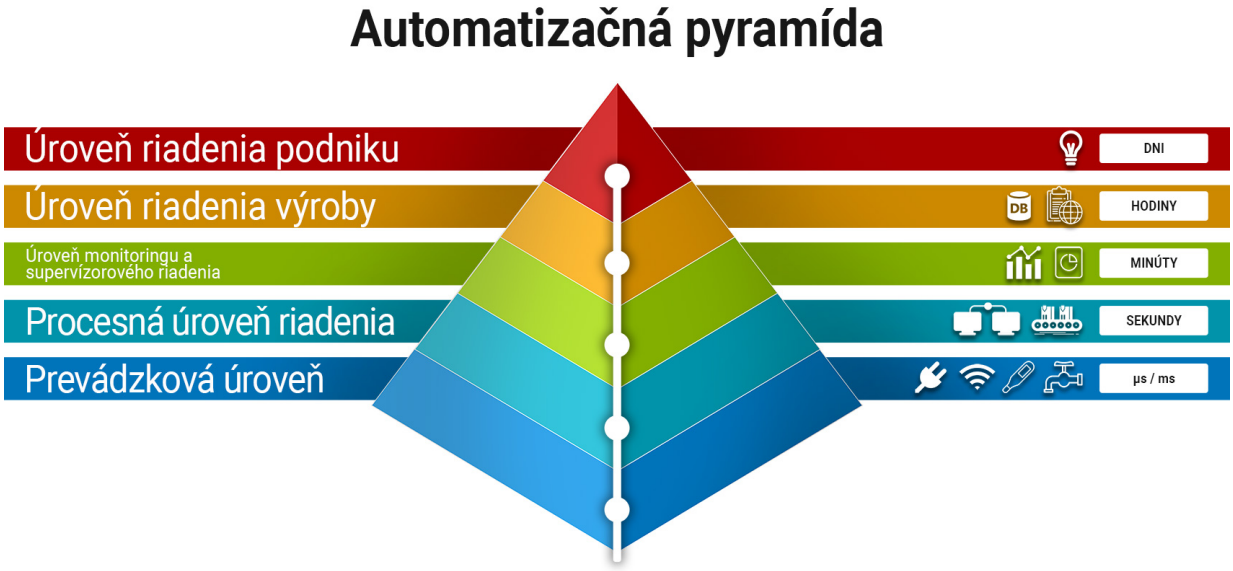
Obrázok č. 5: Správanie diskrétneho udalostného systému [54]

Ako ilustruje obrázok č. 5, správanie takéhoto systému možno charakterizovať diskrétnou stavovou premennou $x(t)$ (hodnoty $q_1, q_2 \ldots q_n$) a zmenami jej hodnoty v čase. U diskrétnych udalostných systémoch pozorujeme len skokové zmeny stavov, ktoré sú spôsobené výskytom udalostí. Pri skúmaní diskrétnych systémov možno postupnosť

okamihov $t_1, t_2 \ldots t_n$ udalostí nahradiť aritmetickou postupnosťou $1, 2 \ldots, n$. Systém obvykle špecifikujeme konečným predpisom, na základe ktorého je možné príslušnú postupnosť stavov generovať, aj keď počet možných postupností je nekonečný. Zmeny stavov nie je možné opísať diferenciálnymi ani diferenčnými rovnicami. Na modelovanie diskrétnych udalostných systémov využívame napríklad konečené automaty alebo Petriho siete.

2.2 Automatizačná pyramída

Priemyselné automatizované systémy sú spravidla štruktúrované do niekoľkých hierarchických úrovní, každá z týchto hierarchických úrovní obsahuje patričnú komunikačnú úroveň s rôznymi požiadavkami na komunikačný systém. Na obrázku č. 6 môžeme vidieť automatizačnú pyramídu. Každá úroveň pyramídy zachytáva skupinu technológií zabezpečujúcu určitú činnosť v rámci priemyselného podniku [21]. Automatizačná pyramída je kľúčovým konceptom v priemyselnej automatizácii a prislúchajúcich informačno-komunikačných technológiách. Táto pyramída ilustruje rôzne úrovne automatizácie od najnižších vrstiev, ktoré zahŕňajú hardvérové komponenty a priame riadiace systémy, až po vyššie úrovne, zahrňujúce podnikové rozhodovacie systémy a cloudové technológie.



Obrázok č. 6: Automatizačná pyramída [56]

1. **Prevádzková úroveň (angl. Field level)** — Táto úroveň sa niekedy nazýva aj úroveň zariadení alebo úroveň snímačov a akčných členov. Ide o najnižšiu úroveň v automatizovanom priemyselnom systéme. Úlohou zariadení v nej je prenos informácií medzi vyrábaným produktom a technologickým procesom. Informácie

môžu byť binárne alebo analógové (spojité). Z podstaty danej úrovne sú v nej najprísnejšie požiadavky na reálny čas. Na tejto úrovni je teda kriticky dôležitá nízka latencia komunikácie, pretože rýchla odozva je nevyhnutná pre efektívne riadenie procesov v reálnom čase.

2. **Procesná úroveň riadenia (angl. Control level)** — Ide o druhú úroveň, ktorá sa často nazýva aj ako riadiaca úroveň. Riadiace úlohy sú obvykle spracovávané prostredníctvom priemyselných počítačov — programovateľných logických kontrolérov (PLC). Pre tieto operácie je nutné zabezpečiť rýchlu časovú odozvu pre synchronizáciu jednotlivých strojov a zariadení a tiež na ošetrenie vzniknutých udalostí.

3. **Úroveň monitoringu a supervízorového riadenia (angl. Supervisory level)** — Ide o úroveň, v ktorej sa funkcie monitorovania a riadenia realizujú súčasne pre niekoľko procesov. Úlohy riadenia sú vykonávané kvalifikovanejším a komplexnejším spôsobom, ako je to na úrovni predošlej. Na tejto úrovni je realizovaná aj vizualizácia prijatých dát. Bežné je využitie klasických počítačov s nainštalovaným systémom HMI či SCADA, pričom v dnešnej dobe môžu na tento účel slúžiť aj mobilné zariadenia či webové aplikácie.

4. **Úroveň riadenia výroby (angl. Planning level)** — Zaisťuje plánovanie výroby na manažérskej úrovni. Význam naberajú rôzne vizualizačné nástroje a v dnešnej dobe aj sofisitikované cloudové aplikácie. Využívané sú aplikačné systémy typu MES. Na tejto úrovni sa spracúvajú a analyzujú dáta z nižších úrovní na optimalizáciu výrobných procesov. Latencia je menej kritická, keďže dáta sú často spracúvané v kratších intervaloch, nie v reálnom čase.

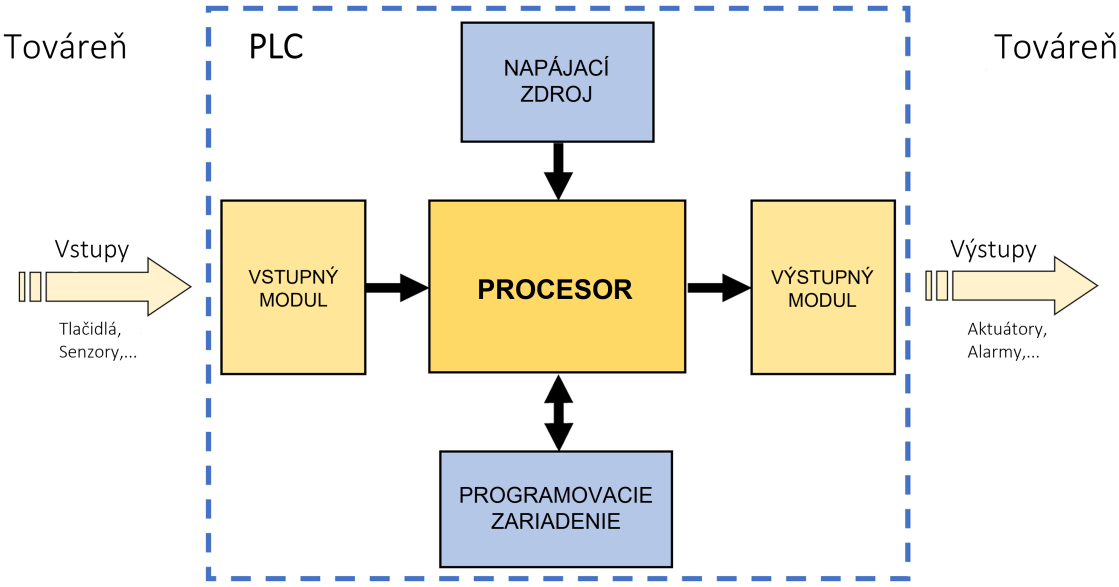
5. **Úroveň riadenia podniku (angl. Management level)** — Ide o najvyššiu (manažérsku) úroveň riadenia podniku. V praxi predstavuje vedenie spoločnosti, špecialistu na vyhodnocovanie úspor a podobne. Títo na základe vyhodnotených dát, ktoré sú predložené operátorom alebo aplikáciami z nižšej úrovni, rozhodujú o efektívnom chode priemyselného podniku. Využívané sú manažérske informačné systémy typu ERP. V tejto vrstve latencia prakticky nehrá rolu, pretože zameranie je na agregáciu a analýzu dát na dlhodobé plánovanie a rozhodovanie.

Ak sa pozrieme na automatizačnú pyramídu z hľadiska **veľkosti dátových tokov**, tak ako postupujeme vyššie po automatizačnej pyramíde, veľkosť dátových tokov sa zvyšuje. Na najnižších úrovniach sú dáta zvyčajne jednoduché signály alebo merania. Avšak na vyšších úrovniach, ako sú MES a ERP, sa dáta stávajú zložitejšími a obsiahlejšími, nakoľko predstavujú zhrnutia a analýzy z nižších úrovní. Tieto väčšie dátové toky

vyžadujú robustnejšie informačné systémy na ich spracovanie a analýzu, ale nevyžadujú takú nízku latenciu ako senzorické a riadiace systémy.

2.3 Programovateľné logické kontroléry (PLC)

V dnešnej dobe automatizácia a riadenie procesov v priemysle prebieha spravidla pomocou *programovateľných logických kontrolérov* (**angl. programmable logic controllers — PLC**). Programovateľný logický kontrolér je priemyselný digitálny počítač. PLC sa skladá z centrálnej procesorovej jednotky (CPU), vstupov, výstupov a napájacieho zdroja. PLC takisto obsahuje riadiaci program. PLC v stručnosti funguje tak, že prijíma vstupné signály od senzorov a následne ich spracováva programovateľnými operáciami a potom vysiela výstupné signály na riadiace prvky (aktuátory), ako sú rôzne ovládače, ventily, motory, relé a podobne [76]. Základná architektúra PLC je znázornená na obrázku č. 7.



Obrázok č. 7: Architektúra PLC [31]

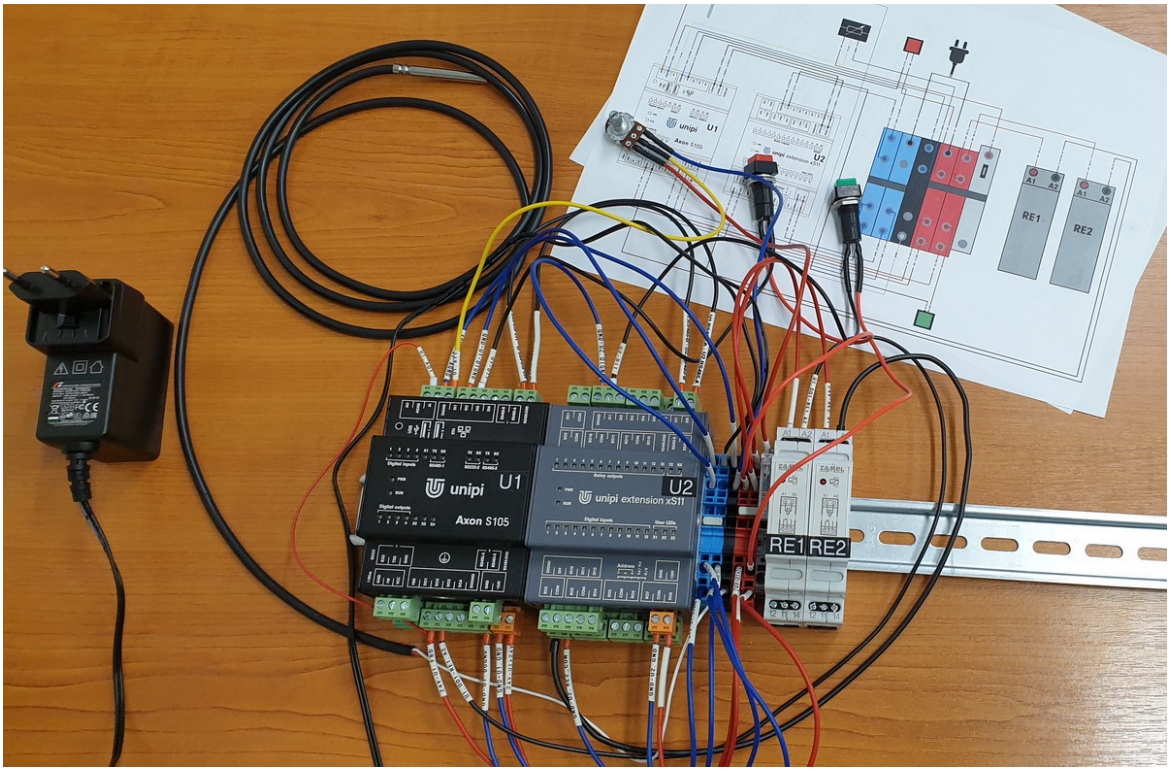
2.3.1 Rozdelenie PLC

Na trhu je v dnešnej dobe možné nájsť rozličné veľkosti (z hľadiska počtu vstupov a výstupov) a typy PLC od rôznych výrobcov. Pre jednoduchosť je možné PLC rozdeliť na **uzavreté** a **otvorené** (často tiež vo forme open-source) [76].

Uzavreté PLC majú typicky výrobcom určenú architektúru a softvérové nástroje, ktoré sú navrhnuté špecificky pre konkrétny hardvér. Integrácia s inými systémami alebo rozširovanie funkcionalít môže byť obmedzené alebo vyžaduje špecifické rozhrania alebo moduly od daného výrobcu. Uzavreté PLC môžu byť výhodné v situáciách, kde je potrebná veľmi špecifická aplikácia alebo vysoká miera spoľahlivosti a stabilnosti. Nevýhodou však je, že môžu byť menej flexibilné pri zmene požiadaviek alebo integrácii

s inými systémami [31]. Patrí sem väčšina PLC od veľkých výrobcov, ako je napríklad Siemens alebo Rockwell Automation.

Otvorené PLC sú tie, ktoré umožňujú spúšťať programy vytvorené v rozličných softvérových nástrojoch (napríklad Codesys alebo OpenPLC) kompatibilných pre daný typ PLC. Výhodou takýchto PLC je flexibilita a dostupnosť iných funkcionalít, ktoré štandardné zatvorené PLC nepodporujú [76]. Otvorené systémy PLC sú navrhnuté takým spôsobom, aby boli kompatibilné s rôznymi typmi hardvéru a softvéru, vrátane tých od iných výrobcov. Tieto systémy zvyčajne podporujú štandardné programovacie jazyky a komunikačné protokoly, čo uľahčuje integráciu s rôznymi zariadeniami a softvérovými riešeniami. Otvorené PLC sú flexibilnejšie a umožňujú lepšiu prispôsobivosť v dynamických priemyselných prostrediach, kde sa požiadavky často menia a je potrebná integrácia s rôznorodými technológiami. Medzi takéto PLC patrí napríklad WAGO PFC200 Controller. Špeciálnou podskupinou sú **PLC založené na open-source technológiách** a operačných systémoch. Ide napríklad o PLC zo série Revolution Pi alebo Unipi založených na výpočtovej jednotke mikropočítača Raspberry Pi [88].



Obrázok č. 8: Certifikačná sada priemyselného počítača Unipi Axon

Najmä v rámci edukácie a testovania je možné využívať aj tzv. **softPLC**. SoftPLC je softvérové (alebo niekedy nazývané tiež virtuálne) PLC, ktoré využíva štandardný osobný počítač ako svoju hardvérovú platformu a využíva špeciálny softvér pre dosiahnutie rovnakých funkcií ako bežné PLC [65]. Takýto softvér pozostáva z **vývojovej časti**

(editora PLC programov) a **runtimu** pre beh PLC programov.

Vývojová časť je chápaná ako prostredie pre implementáciu PLC programov. Tieto programy musia byť v súlade s štandardom IEC 61131-3 [86].

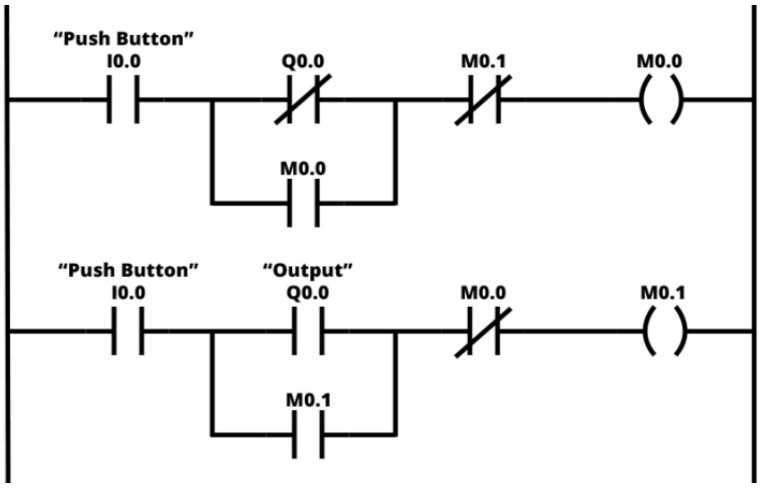
Runtime je jadrom softPLC systému a má na starosti jeho administratívu, spúšťanie programu a riadenie vstupných a výstupných procesov z programovej (vývojovej) časti. Takto získame virtuálne PLC v osobnom počítači, ktoré simuluje hardvérové PLC. S aktuátormi a senzormi môžeme komunikovať napríklad prostredníctvom protokolov Modbus TCP alebo OPC Unified Architecture.

2.3.2 Programovacie jazyky PLC

Norma IEC 61131-3 je medzinárodnou normou, ktorá špecifikuje požiadavky na návrh, implementáciu a testovanie programov v PLC [86]. Dôvodom existencie tejto normy je zabezpečenie interoperability medzi rôznymi PLC zariadeniami a zjednodušený vývoj a údržba PLC softvéru. Norma definuje päť programovacích jazykov [76]:

- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Structured Text (ST)
- Instruction List (IL)
- Sequential Function Chart (SFC)

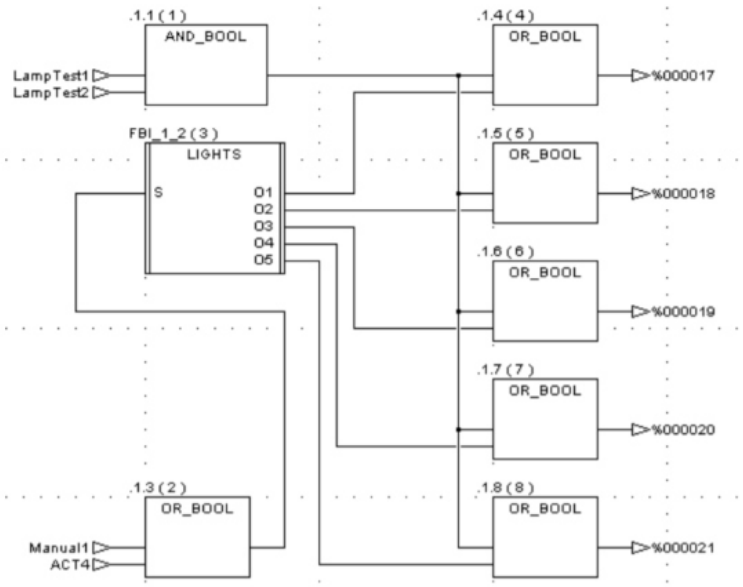
Ladder Diagram (LD) — je grafický jazyk (obrázok č. 9), ktorý zobrazuje riadiace obvody ako schémy zapojenia v oblasti elektrotechniky a aj z tohto dôvodu patrí medzi obľúbené programovacie jazyky pre PLC. Známy je tiež ako *rebríkový diagram* alebo *reléová logika*. Riadiace obvody obsahujú logické kontakty a cievky. Kontakt predstavuje stav vstupu, ktorý môže byť buď zapnutý (1), alebo vypnutý (0). Cievka predstavuje výstupný prvok alebo akciu. Kontakt a cievka môžu byť prepojené pomocou rôznych logických operácií, ako sú AND, OR a NOT, aby sa vytvoril riadiaci obvod. Tento programovací jazyk sa väčšinou využíva pre jednoduché riadiace procesy v priemysle, ako sú spínanie a regulácia vstupov a výstupov [76]. *Výhody* takéhoto programovania sú prehľadnosť zápisu, jasná definícia postupnosti zápisu, jednoduché ladenie a taktiež programovanie logických funkcií je veľmi rýchle. Hoci majú rebríkové diagramy veľa výhod, majú aj niektoré *nevýhody*. Napríklad štruktúra údajov je ťažko chrániteľná a údaje môžu byť vystavené riziku náhodnej modifikácie chybným kódom na inom mieste. Nie sú tiež vhodné na prácu so zložitými algoritmami, nakoľko môžu byť náročné na programovanie, dokumentovanie, ladenie a úpravu [69].



Obrázok č. 9: Ladder Diagram [37]

Function Block Diagram (FBD) — je to taktiež grafický jazyk (obrázok č. 10), kde sú riadiace obvody zostavené z *funkčných blokov*. Každý funkčný blok predstavuje nejakú časť funkcionality PLC a môže byť využitý na riešenie širokej škály riadiacich úloh. Funkčné bloky sú zvyčajne reprezentované grafickým symbolom, ktorý zobrazuje ich vstupy, výstupy a iné parametre. Funkčné bloky môžu byť navzájom prepojené na vytvorenie zložitejších riadiacich obvodov. FBD sa používa pre zložité riadiace procesy, ktoré zahŕňajú mnoho vstupov a výstupov a vyžadujú zložité logické operácie [76]. FBD je veľmi podobný programovaciemu jazyku LD, v ktorom sa u logických operácií namiesto sériovo-paralelného zapájania symbolov kontaktov relé využívajú štandardné značky hradiel AND, OR a podobne [66]. *Výhodou* využitia FBD je, že vo funkčnom bloku možno využiť ľubovoľný počet vstupov a výstupov. Pri použití viacerých vstupov a výstupov je možne pripojiť výstup jedného funkčného bloku na vstup iného, čím sa zostavuje schéma funkčného bloku. Tieto vlastnosti FBD vedia zvýšiť prehľadnosť programu. Ako *nevýhodu* FBD môžeme uviesť, že funkčné bloky sa môžu umiestniť kdekoľvek vo vývojovom prostredí, a tak sa kód môže zdať ako nezorganizovaný [69].

Structured Text (ST) — je textový jazyk (obrázok č. 11) inšpirovaný programovacími jazykmi C a Pascal. Riadiace obvody sú zostavené pomocou matematických a logických operácií a tiež aj cyklov a funkcií. Takisto poskytuje rôzne dátové typy, ako sú celé čísla, desatinné čísla, reťazce a polia. ST sa využíva na vytváranie komplexných riadiacich obvodov s vysokou úrovňou abstrakcie a znovupoužiteľnosti kódu [76]. *Výhody* ST oproti grafickým programovacím jazykom spočívajú v tom, že programy v ST sú ľahko čitateľné a zrozumiteľné pre ľudí s programovacími skúsenosťami. Sú umožnené zložitejšie operácie, ako sú matematické alebo porovnávacie operácie a podmienky. ST umožňuje používať funkcie, čo umožňuje znovupoužiteľnosť kódu a zjednodušuje programovanie. Ako *nevýhody* je možné uviesť vyžadovanie vyššej úrovne znalostí pro-



Obrázok č. 10: Function Block Diagram [37]

gramovania ako napríklad rebríkový diagram a tiež to, že programy môžu byť dlhšie a zložitejšie na čítanie ako programy v rebríkovom diagrame [69]. Takisto spracovanie veľkého množstva logických vstupov a výstupov môže byť neprehľadné [66].

```
>>>>>>>>>>>>>>>>> Structured Text Start <<<<<<<<<<<<<<<<<
VAR
  TIMER : TON;
END_VAR

  TIMER(IN := NOT pulse,
        PT := t#1s);  (* Blink timer *)
  pulse := TIMER.Q;

  (* Count every pulse *)
  IF pulse = 1 THEN
    count := count + 1;
  END_IF;
  (* Animate lights according to counter *)
  CASE count OF
    1: out1 := TRUE;
    2: out2 := TRUE;
    3: out3 := TRUE;
    4: out4 := TRUE;
    5: out5 := TRUE;
    6: out6 := TRUE;
    7: out7 := TRUE;
    8: out8 := TRUE;
  ELSE (* All lights are on, switch them off again and start from the beginning *)
    out1 := FALSE;
    out2 := FALSE;
    out3 := FALSE;
    out4 := FALSE;
    out5 := FALSE;
    out6 := FALSE;
    out7 := FALSE;
    out8 := FALSE;
```

Obrázok č. 11: Structured Text [37]

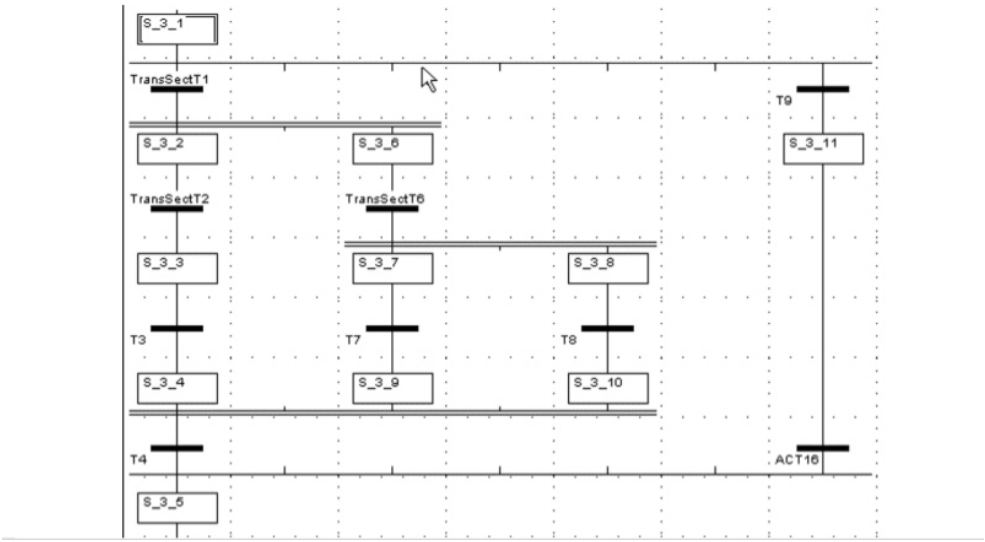
Instruction List (IL) — je nízkoúrovňový jazyk (obrázok č. 12), ktorý sa skladá z krátkych inštrukcií. Inštrukcie sa vykonávajú v poradí, v akom sú zapísané (sekvenčne). IL je inšpirovaný jazykom Assembler a poskytuje plnú kontrolu nad riadiacimi obvodmi.

Tento jazyk je už zastaralý a zväčša sa nepoužíva. Používal sa v aplikáciách s veľkým množstvom vstupov, výstupov a tam, kde bola potrebná maximálna rýchlosť a efektivita PLC, kde sa plne odrážajú jeho *výhody* [76]. Medzi *nevýhody* patrí neprehľadnosť a zložitá orientácia v kóde [66].

```
>>>>>>>>>>>>>>>>> Instruction List Start <<<<<<<<<<<<<<<<<
VAR
  RUN_TIMER : TON;  (* Blink timer *)
END_VAR
        (* Default for the marker *)
    LD      run_light1
    ST      run_light

        (* Create a 1.0 Hz. pulse *)
    LD      run_pulse
    STN RUN_TIMER.IN
    CAL RUN_TIMER(PT := t#1s)
    LD      RUN_TIMER.ET
    ST      animatetime
    LD      RUN_TIMER.Q
    ST      run_pulse
    JMPCN   end         (* No pulse yet, nothing to do *)

    LD      run_light8      (* Rotate all bits one position "up" *)
    ST                              run_light
    LD      run_light7
    ST                              run_light8
    LD      run_light6
    ST                              run_light7
    LD      run_light5
    ST                              run_light6
    LD      run_light4
    ST                              run_light5
    LD      run_light3
    ST                              run_light4
    LD      run_light2
    ST                              run_light3
    LD      run_light1
    ST                              run_light2
end:LD      run_light
    ST                              run_light1
```

Obrázok č. 12: Instruction List [37]

Sequential Function Chart (SFC) — je ďalší grafický jazyk (obrázok č. 13), kde sú riadiace obvody zostavené z blokov, ktoré predstavujú kroky daného procesu. Takýto program sa vykonáva sekvenčne a umožňuje modelovať a riadiť komplexné procesy zostavené z algoritmov s viacerými krokmi. Každý krok v bloku môže byť buď akcia, ktorá sa má vykonať, alebo podmienka, ktorá sa musí splniť na prechod do ďalšieho kroku. SFC sa často využíva pri riadení procesov, ktoré majú presne definované postupnosti krokov. Napríklad ide o výrobu produktov, riadenie dopravy alebo monitorovanie daného procesu. SFC umožňuje vytvárať a riadiť zložité procesy a zvyšuje účinnosť a presnosť riadiaceho systému [76]. Medzi *výhody* teda patrí to, že správanie programu je prehľadne zapísané. Ďalej je možné rozdelenie procesov do hlavných krokov, definovanie a ošetrenie rôznych stavov programu. Taktiež je vhodný pre jednoduchú prácu s ASCII reťazcami. Ako *nevýhodu* je možné spomenúť, že nie je najvhodnejší pre spracovanie veľkého počtu logických signálov a pre programovanie zložitých algoritmov [66]. Ďalšou nevýhodou je, že tento programovací jazyk je väčšinou vhodný len pre procesy sekvenčného typu a tiež to, že vlastné dátové typy zadefinované pomocou rôznych prvkov (napríklad pole alebo premenné rôznych typov) využívajú zbytočne veľa pamäte a to najmä vtedy, ak je väčšina členských značiek (atribútov) nevyužitá [69].



Obrázok č. 13: Sequential Function Chart [37]

2.4 Modbus

Modbus je komunikačný protokol určený pre automatizačné systémy. Architektúru má založenú na master-slave (Modbus RTU) alebo klient-server (Modbus TCP/IP). Jeho hlavnou úlohou je uľahčenie spoľahlivej a rýchlej komunikácie medzi prevádzkovými zariadeniami. Protokol Modbus bol vyvinutý v roku 1979 pre programovateľné logické kontroléry [66]. Odvtedy sa stal jedným z najbežnejších komunikačných protokolov v automatizácii a riadení. Protokol je verejne dostupný a nezávislý na konkrétnom výrobcovi či konkrétnej technológii, čo umožňuje interoperabilitu medzi rôznymi zariadeniami. Nevýhodou je nízka kybernetická bezpečnosť a z hľadiska dnešnej doby ťažkopádny prístup k údajom, preto je v niektorých aplikáciách nahrádzaný moderným štandardom OPC Unified Architecture.

Rozlišujú sa tieto **prevádzkové režimy prenosu údajov** protokolu Modbus:

- **MODBUS TCP/IP** — ide verziu pre Ethernet TCP/IP komunikáciu, ktorá je založená na modeli klient-server. Modbus TCP/IP je varianta protokolu, ktorá využíva Ethernetové rozhranie pre komunikáciu. Táto verzia umožňuje prenos dát cez IP siete, vrátane lokálnych a širokopásmových sietí. Modbus TCP/IP umožňuje vzdialenú komunikáciu a je široko používaný v priemyselných systémoch.

- **MODBUS RTU** — patrí sem asynchrónny prenos cez RS-232 alebo RS-485 sériové rozhrania, pričom ide o komunikačnú architektúru master-slave (v dnešnej dobe vzhľadom na politickú korektnosť sa aj tu niekedy využíva notácia klient-server [97]).

- **MODBUS ASCII** — je podobný protokolu RTU s výnimkou iného formátu údajov,

ale je pomerne zriedka používané. Dáta sú prenášané vo forme ASCII znakov, čo zjednodušuje ladenie a diagnostiku. V porovnaní s Modbus RTU má však nižšiu rýchlosť prenosu dát. Modbus ASCII je teda ďalší sériový variant, ktorý sa líši vo forme prenosu dát.

Ak by sme zašli do histórie Modbusu, tak bol to jeden z prvých komunikačných protokolov určených na výmenu dát. Výmena dát fungovala medzi **nadriadenou jednotkou** (**master**) a **podriadenými jednotkami** (**slave**) v distribuovaných systémoch riadenia a zberu dát. Považovať ho môžeme za protokol tretej až piatej vrstvy modelu OSI. Pretože bol veľmi jednoduchý, prehľadný a jednoducho implementovateľný, tak sa stal veľmi rozšíreným protokolom. Príchodom Ethernetu bol Modbus pretransformovaný do relačnej vrstvy [66].

Modbus ponúka 4 typy prenášaných dát:

- **Discrete Input** — jednobitové registre, teda hodnoty BOOL používané ako binárne vstupy (napríklad zo senzorov) a možno ich iba čítať. Dokáže ho spravidla zapisovať len server / slave.
- **Coils** — jednobitové registre, teda hodnoty BOOL, ktoré sa používajú na riadenie diskrétnych binárnych výstupov a možno ich čítať a zapisovať. Dokáže ho zapisovať nielen server, ale aj klient / master.
- **Input Register** — 16-bitové registre používané na vstup a možno ich iba čítať. Je podobný *Discrete Input*, akurát nejde o hodnotu BOOL, ale ide o celočíselnú hodnotu INT s veľkosťou 16 bitov, ktorá môže byť *unsigned* alebo *signed*.
- **Holding Register** — najuniverzálnejší 16-bitový register, možno ho čítať alebo zapisovať a možno ho použiť na rôzne účely vrátane vstupov, výstupov, konfiguračných údajov alebo akýchkoľvek požiadaviek na „udržiavanie“ údajov. Je podobný ako *Coil*, akurát nejde o hodnotu typu BOOL, ale ide o celočítelnú hodnotu INT s veľkosťou 16 bitov, ktorá môže byť *unsigned* alebo *signed*.

2.5 Konvergencia informačných a operačných technológií

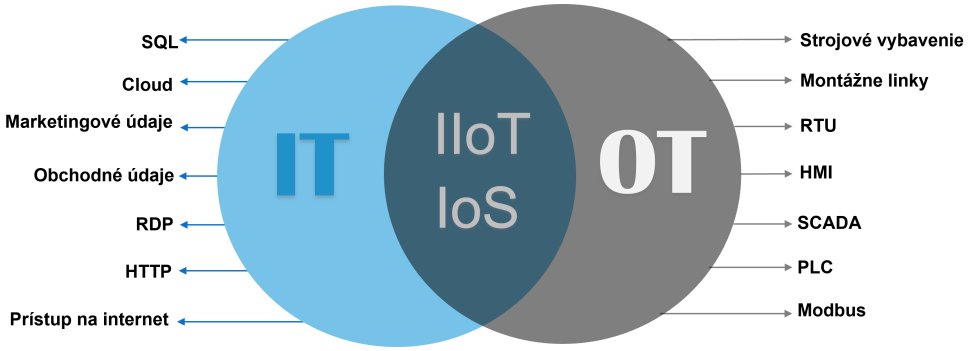
V tradičnom ponímaní sa **informačné systémy a technológie** (information technologies — IT) vyvíjali a využívali oddelene od priemyselných **operačných technologických systémov** resp. **operačných technológií** (operation technologies — OT).

Pod informačnými technológiami rozumieme napríklad zariadenia pre komunikáciu a celé spektrum technológií zameraných na spracovanie dát, vrátene hardvéru, softvéru, komunikačné technológie a ďalšie súvisiace služby. Príkladom môže byť CRM, ERP alebo e-mail.

Pod operačnými technológiami rozumieme hardvér a softvér, ktorý monitoruje alebo priamo spôsobuje zmeny, teda riadi fyzické zariadenia, procesy a udalosti v podniku. OT sú tak nevyhnutnou súčasťou priemyselného riadenia a automatizácie vo výrobnom podniku. Príkladom sú systémy SCADA, HMI alebo priemyselné logické kontroléry / počítače (PLC). Môžeme sa tiež stretnúť s pomenovaním prevádzkové technológie.

Názorné porovnanie IT a OT môžeme nájsť vo vhodnej literatúre [45].

V rámci jedného podniku sa vedľa seba nachádza OT a aj IT. Oba tieto technologické smery neboli v minulosti účelovo vyvíjané na vzájomnú spoluprácu a ani spoluprácu s inými systémami. Príchodom nových digitálnych a inteligentných technológií, Internet of Things, analytických systémov Big Data a podobne, však nastáva požiadavka na konvergenciu IT a OT. Vďaka IIoT dokážu informácie prúdiť od zákazníkov k manažérom, plánovačom, na výrobnú plochu a zase naspäť [38]. Komplexné stroje sa začali integrovať so sieťovými prvkami a sú riadené pokročilým analytickým softvérom. Trend prepájania IT a OT je tak nesporný, hoci IT a OT doteraz tvorili dva separátne svety. Z tohto dôvodu je ťažké na Slovensku zohnať špecialistu na prepojenie PLC s cloudovým systémom alebo realizáciu edge computingu. Aj vzdelávací systém začal len postupne zohľadňovať požiadavku konvergencie IT a OT — alebo v širšom zmysle **konvergenciu informačno-komunikačných technológií a automatizácie**, ktorej následkom sú **kybernetico-fyzikálne systémy** — piliere **Industry 4.0** [56].



Obrázok č. 14: Konvergencia IT a OT [66]

Vzhľadom na požiadavky na implementáciu koncepcie Industry 4.0 budú zamestnávatelia potrebovať ľudí s novými zručnosťami a schopnosťami, najmä v digitálnej oblasti. Všetky typy škôl vrátane netechnických musia zmeniť svoje vyučovacie a vzdelávacie metódy a namiesto úzkej špecializácie na jednu oblasť by sa vzdelávanie malo zamerať na oveľa širší prehľad. Ľudí je tiež potrebné vzdelávať v systémovom a interdisciplinárnom myslení na všetkých typoch škôl. Hlavným cieľom štúdie v zdroji [27] bolo určenie oblastí, na ktoré by sa mal v budúcnosti sústrediť obsah vzdelávania v súvislosti s Industry 4.0. Ako jedna z výskumných metód bol realizovaný dotazníkový prieskum

v slovenských podnikoch. Podľa výsledkov výskumu by zmeny v kvalifikačnej štruktúre pracovnej sily súvisiace s implementáciou konceptu Industry 4.0 mali mať pozitívny vplyv na zvýšenie konkurencieschopnosti podnikov a zvýšenie efektívnosti výroby. Na základe zistení sa navrhuje, aby sa predpokladané pozitívne zmeny realizovali ako štrukturálne zmeny v rámci slovenského vzdelávacieho systému a presadzovanie odborného vzdelávania v podnikoch.

V oblasti priemyselnej komunikácie je pre zosúladenie IT a OT navrhnutý komunikačný štandard OPC Unified Architecture charakterizovaný v kapitole 2.6.

Donedávna sme poznali priemyselné počítače alebo PLC len ako zariadenia s uzavretým programovým systémom. Novým typom produktu, ktorý prináša spomínaná konvergencia, sú **open-source priemyselné počítače** (angl. Industrial PC — IPC). Rovnako ako konvenčné PLC poskytujú základný softvér pre tvorbu a beh programov v súlade s normou IEC 61131-3, avšak pridávajú možnosť inštalácie otvoreného operačného systému na báze Linuxu.

Jedným z takýchto produktov je rad priemyselných počítačov od českej spoločnosti **Unipi**. Ponúkajú varianty založené na výpočtovom module z Raspberry Pi 3 (séria **Unipi Neuron**) a novšie typy založené na vlastnom výpočtovom module vhodnom aj pre náročnejšiu automatizáciu (**Unipi Axon**). Súčasťou oboch typov je vlastný programový systém **Mervis** slúžiaci ako editor programov normy IEC 61131-3 a tiež ako SCADA / HMI. Je však tiež dodávané aj rozhranie API, ktoré sprostredkúva priamy prístup k vstupom, výstupom a komunikačným rozhraniam jednotiek. Vďaka API je možné riadenie hardvéru Unipi rýchlo a jednoducho integrovať do vlastného softvéru alebo do riešenia tretích strán a predstavuje tak ideálnu možnosť pre softvérových vývojárov. Podporované je aj prepojenie s middlewarom Node-RED. Výhodou produktov od Unipi (obrázok č. 8) je vynikajúca podpora zákazníkov a vývojárov od samotného výrobcu [88].

Ďalšou sériou produktov sú priemyselné počítače od nemeckej spoločnosti Kunbus s názvom **Revolution Pi**. Ako už u názvu vyplýva, tak využívajú výpočtový modul z Raspberry Pi 3 a 4. Ponúkajú dve série počítačov. Prvá z nich je **RevPi Core** a druhá je **RevPi Connect**. RevPi Connect obsahuje prídavný sieťový port, čím vieme tento počítač jednoduchým spôsobom využiť aj ako IIoT gateway. Ako editor programov normy IEC 61131-3 bol v minulosti odporúčaný softvér **logi.CAD 3**, od približne roku 2021 je preferované robustnejšie riešenie s názvom **CODESYS**, ktoré je predstavené a používané aj v tejto učebnici. Pre softvérových vývojárov je dostupný je open-source real-time operačný systém **Raspbian / Raspberry Pi OS with RT patch**. Podobne ako v predošlom prípade je podporované aj prepojenie s Node-RED. Zaujímavosťou je, že mottom spoločnosti je "Industry 4.0 — Don´t just claim it — Make it!" [53].

2.6 OPC Unified Architecture

Víziou OPC Foundation je ponúknuť **multiplatformový a multidoménový štandard** (**platformu**) pre **interoperabilitu** a výmenu dát od senzorov až po enterprise systémy. **OPC UA** (**Open Platform Communications Unified Architecture**) sa stáva jedným z najdôležitejších komunikačných štandardov pre riešenia Industry 4.0 a IIoT [67]. Pomocou OPC UA sa štandardizuje prístup k strojom, zariadeniam, snímačom, aktuátorom a iným systémom v priemyselnom prostredí. Umožňuje pohodlnú a od výrobcu nezávislú výmenu údajov. Zjednodušuje pripojenie v priemyselnom prostredí, takže je možné integrovať všetky zariadenia, automatizačné systémy a softvérové aplikácie pomocou bezpečného a platformovo nezávislého štandardu.

OPC UA sa okrem interoperability vyznačuje aj **nasledovnými vlastnosťami**, ktoré sú v súlade s požiadavkami konceptu Industry 4.0 [17]:

- **Nezávislosť na platforme**: Protokol je navrhnutý tak, aby bol nezávislý od konkrétnej platformy alebo operačného systému (na rozdiel od staršieho protokolu OPC Classic). To umožňuje komunikáciu medzi rôznymi zariadeniami a systémami bez ohľadu na ich hardvérové alebo softvérové prostredie.

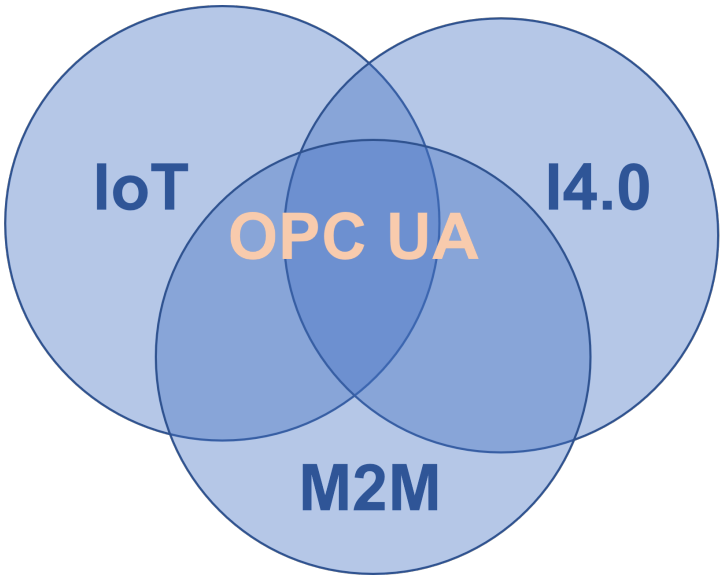
- **Bezpečnosť**: OPC UA poskytuje rôzne mechanizmy na zabezpečenie komunikácie a autentifikáciu. Patrí sem šifrovanie, podpisovanie dát, zabezpečené spojenia pomocou SSL/TLS a možnosť rôznych spôsobov autentifikácie.

- **Rozšíriteľnosť a modularita**: Architektúra OPC UA umožňuje rozšírenie a prispôsobenie protokolu na mieru konkrétnym potrebám aplikácií. Je možné definovať vlastné dátové typy, objekty a služby.

- **Široká funkcionalita**: OPC UA poskytuje rôzne mechanizmy na získavanie, prenos a spracovanie dát. Patrí sem zber a prenos reálnych hodnôt, historických dát, alarmov a udalostí, atď.

OPC UA sa začal široko používať v modernom priemyselnom automatizovanom prostredí, kde je potrebná spoľahlivá a bezpečná komunikácia medzi rôznymi zariadeniami (angl. *machine-to-machine — M2M komunikácia*) a systémami [76]. Je možné povedať, že je OPC UA prienikom M2M, IoT a Industry 4.0 (obrázok č. 15).

Kľúčovou vlastnosťou OPC UA je schopnosť **sémantickej interoperability**. Toto znamená, že nielen dokáže prenášať dáta medzi zariadeniami a systémami, ale tiež poskytuje kontext a význam týchto dát. Táto sémantická vrstva umožňuje, aby rôzne systémy nielenže *rozumeli* dátam, ale tiež vedeli, ako s nimi efektívne pracovať a integrovať ich do širších procesov. Tento prístup zvyšuje efektivitu a flexibilitu automatizačných



Obrázok č. 15: OPC UA ako prienik viacerých konceptov/technológií [76]

systémov, pretože umožňuje rôznym zariadeniam a aplikáciám rozumne spolupracovať bez potreby manuálneho nastavovania alebo špecifického programovania pre každú jednotlivú aplikáciu alebo zariadenie.

Komunikácia OPC UA spočíva na servisne-orientovanej architektúre (angl. *service-oriented architecture — SOA*) a ponúka dva komunikačné modely (spôsoby komunikácie):

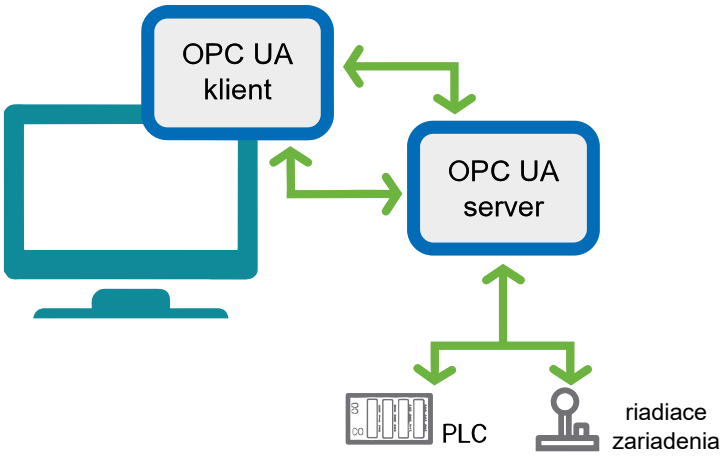
- klient-server;
- publish-subscribe.

Oba tieto spôsoby obsahujú **informačný model**. Informačný model definuje štruktúru a typy dát, ktoré sú dostupné v systéme a spôsob, akým sú tieto dáta organizované a zdieľané. Založený je na **objektovo-orientovanom princípe** a hierarchickom strome objektov [34]. Objekty v informačnom modeli predstavujú fyzické zariadenia, procesy, služby a dátové body, ktoré sú dostupnú pre komunikáciu cez OPC UA. Tieto objekty sú zadefinované v adresnom priestore, ktorý je zdieľaný medzi klientom a serverom. Objekty v adresnom priestore sú tvorené uzlami a ich referenciami [76]. Všetky uzly sú jednoznačne adresovateľné na základe indexu ich menného priestoru (angl. *namespace index*) a identifikátorom uzla (angl. *node ID*). Referencie definujú vzťahy s inými uzlami (napr. *HasSubtype*, *HasProperty* atď.) [14].

2.6.1 Komunikácia klient-server

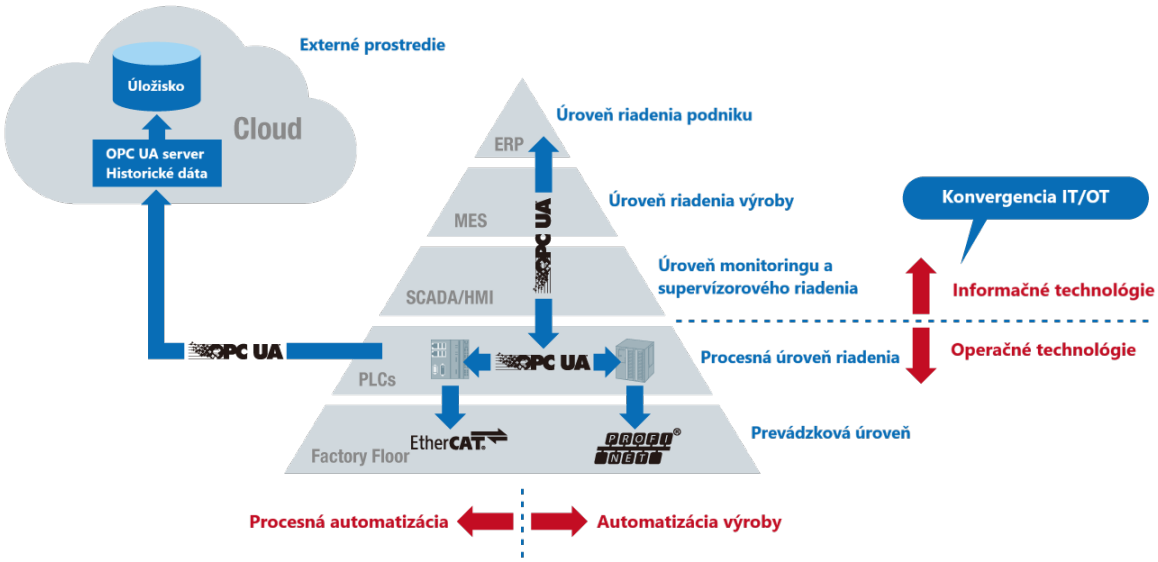
V komunikačnom modeli **klient-server** existuje hierarchický vzťah medzi klientmi a servermi. Server je zodpovedný za správu dát, poskytovanie prístupu k daným uzlom

a vykonávanie operácií nad nimi. Klienti sú aplikácie alebo systémy, ktoré žiadajú o prístup k dátam a vykonávajú operácie na serveri (obrázok č. 16).



Obrázok č. 16: Komunikačný model klient-server [76]

OPC UA server je zvyčajne pripojený k fyzickému zariadeniu a zabezpečuje komunikáciu so zariadením, odosiela alebo načítava hodnoty. Je to softvér, ktorý implementuje štandardy OPC UA a poskytuje tak štandardizované rozhrania OPC UA vonkajšiemu svetu, teda jednému alebo viacerým OPC UA klientom (ilustrácia na obrázku č. 17).



Obrázok č. 17: Postavenie OPC UA v rámci automatizačnej pyramídy [36]

Servery OPC sú škaľovateľné a môže ísť o zariadenia v podobe malých senzorických

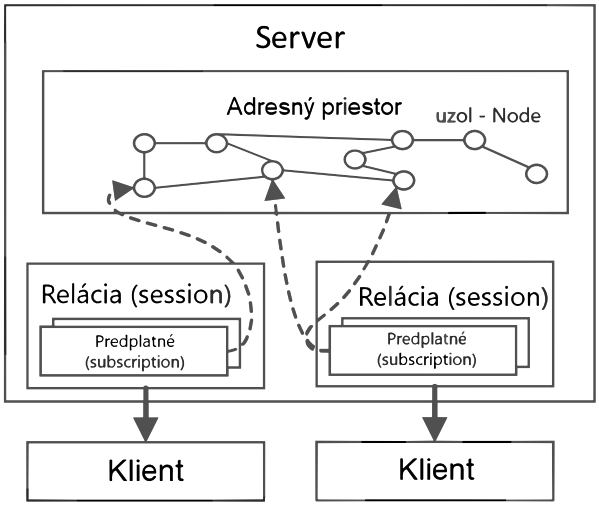
systémov s niekoľkými dátovými bodmi až po veľké stroje s tisíckami dátových bodov. Okrem toho môže stroj fungovať aj ako agregátor, teda byť hostiteľom servera OPC UA a aj vlastného klienta OPC UA [13]. Jeho klient sa môže pripojiť ku všetkým serverom OPC UA, ktoré hostia jednotlivé zariadenia v systéme, a zdieľať rovnaké údaje na svojom vlastnom serveri, čo znamená, že všetci používatelia sa musia pripojiť len k tomuto jedinému serveru, aby mali prístup k údajom z celého systému [69].

Servery OPC UA umožňujú pripojenie OPC UA klienta a následné čítanie údajov. S použitím preddefinovaných rozhraní štandardu OPC UA môže každý klient pristupovať k ľubovoľnému serveru OPC UA a vymieňať si údaje so serverom rovnakým spôsobom. Typickými klientmi OPC UA sú aplikácie, ktoré závisia od výmeny údajov s priemyselnými systémami. To, čo sa deje s údajmi v klientovi, je veľmi špecifické pre danú aplikáciu. Bežnými aplikáciami sú vizualizačné systémy a systémy SCADA alebo systémy MES [69].

SOA pre priemyselné aplikácie v modeli klient-server definuje niekoľko abstraktných metód — služieb pre výmenu dát medzi klientom a serverom. K základným službám patrí [76]:

- **Read**: Klient môže použiť túto službu na získanie hodnôt dátových bodov zo servera. Môže čítať jednu alebo viacero hodnôt naraz.
- **Write**: Klient môže pomocou tejto služby nastaviť hodnoty uzlov na serveri. Môže zapisovať jednu alebo viacero hodnôt naraz a získať potvrdenie o úspešnom zapísaní.
- **Browse**: Táto služba umožňuje klientovi získať informácie o dostupných uzloch, dátových bodoch a ich vlastnostiach na serveri. Klient môže prehliadať hierarchiu uzlov a získať informácie o ich štruktúre.
- **Subscribe**: Klient môže pomocou tejto služby vytvoriť *predplatné* na dané uzly. Server pravidelne posiela aktualizácie hodnôt uzlov klientovi, čím umožňuje sledovanie zmeny hodnôt v reálnom čase.

Existujú **dva spôsoby komunikácie** v rámci modelu klient-server. Najjednoduchším spôsobom je využitie služieb *Read* a *Write*, ktoré umožňujú OPC UA klientovi čítať a zapisovať jeden alebo viacero uzlov spravovaných v adresnom priestore servera OPC UA. Iným a sofistikovanejším spôsobom prístupu k dátam je model **klient-server *subscription*** (obrázok č. 18). Ide o preferovanú metódu pre klientov, ktorí potrebujú cyklické aktualizácie zmien hodnôt premenných [14].



Obrázok č. 18: Komunikačný model klient-server *subscription* [12]

Komunikačný model klient-server v OPC UA je nezávislý od konkrétnych technológií a sieťových protokolov. Štandard OPC UA umožňuje komunikáciu medzi klientov a serverom cez komunikačné protokoly [13]:

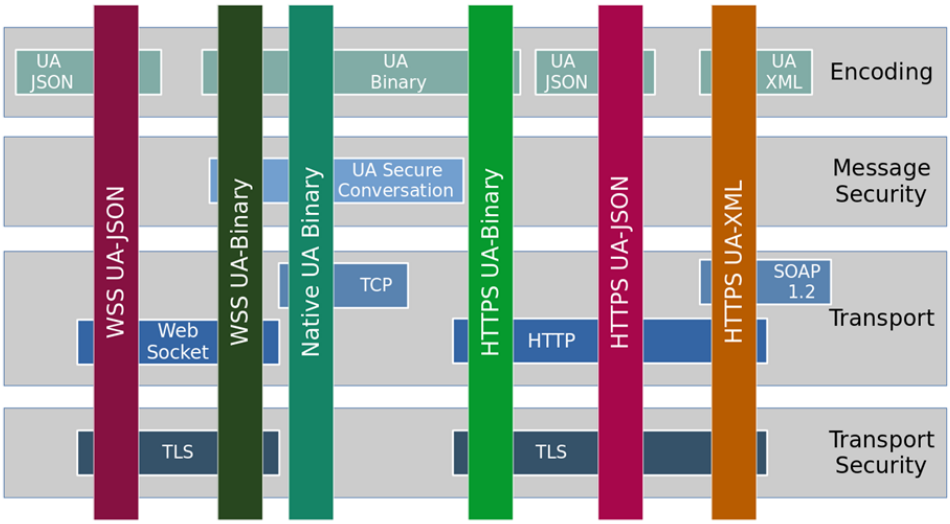
- **UA TCP**: informácie sú binárne zakódované a prenášané cez UA TCP. Ide o protokol založený na *connection/session*, ktorý vytvára nízkoúrovňový komunikačný kanál medzi klientom a serverom.
- **Hybridná komunikácia**: informácie sú binárne zakódované a prednášané cez TLS (angl. *Transport Layer Security*). Komunikácia môže prebiehať cez HTTPS alebo iné protokoly, ktoré využívajú TLS.
- **WebSocket**: informácie sú zakódované do JSON formátu a prenášané cez TLS. JSON je formát na štruktúrovanie/prenos údajov, ktorý uprednostňuje čitateľnosť pre človeka.

2.6.2 Komunikácia publish-subscribe (PubSub)

V tomto komunikačnom modeli máme dva (resp. tri) komponenty: odosielateľa (*publishera*) a odberateľa či prijímateľa (*subscribera*) a sieť medzi nimi. Skrátene sa nazýva tento model aj **PubSub**.

Publisher vytvorí množinu údajov (*DataSets*), ktorú zverejní do siete. Z rovnakej siete si subscriber môže vyfiltrovať a vybrať tie údaje, ktoré potrebuje. Na rozdiel od modelu *klient-server subscription* tu je odosielateľ tým, kto definuje množinu údajov, resp. to, čo sa bude odosielať k prijímateľovi [13].

V IIoT riešeniach, kde existuje veľké množstvo zariadení alebo senzorov, je PubSub



Obrázok č. 19: OPC UA a komunikačné protokoly [29]

ideálny pre distribúciu dát medzi mnohými uzlami bez potreby priamej dvojstrannej komunikácie medzi každým zariadením.

Pre aplikácie, ktoré vyžadujú okamžitú reakciu na určité udalosti alebo streamovanie dát v reálnom čase, je PubSub efektívny, pretože minimalizuje oneskorenia v komunikácii a umožňuje rýchlu distribúciu informácií.

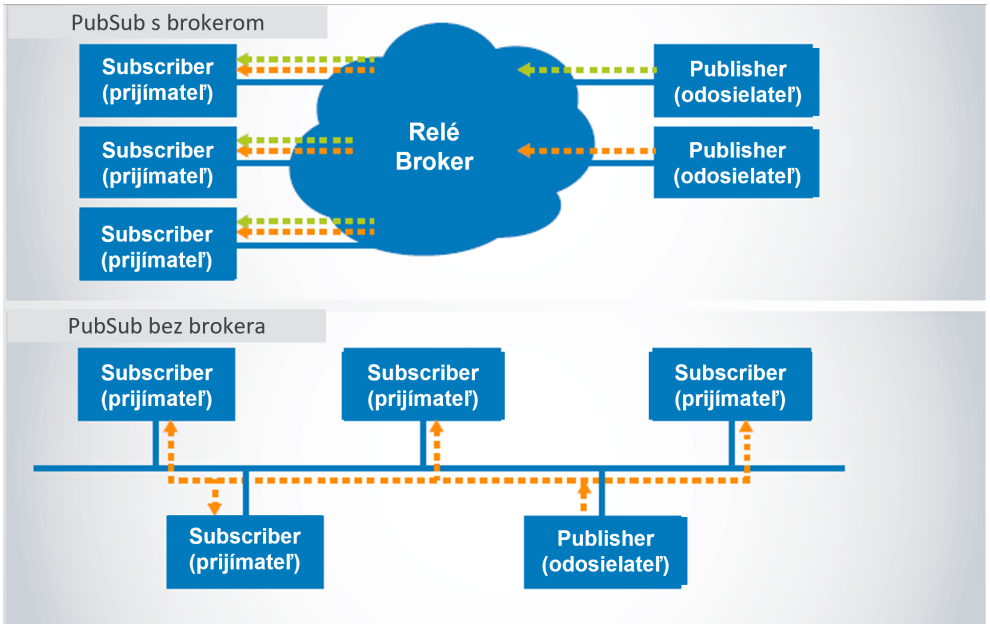
V situáciách, kde nie je vhodné alebo možné mať centralizovaný riadiaci systém, umožňuje PubSub distribuovanú architektúru, kde môže každý uzol nezávisle publikovať alebo odoberať dáta.

V závislosti od toho, aký typ sieti/protokolu je zvolený, model PubSub na výmenu dát poskytuje (obrázok č. 20) [76]:

- **OPC UA PubSub bez brokera**: môže využívať UDP Broadcast (alebo Unicast v niektorých prípadoch). Správy sú optimalizované binárne UADP. Tento typ je vhodný na komunikáciu v reálnom čase (deterministickú).
- **OPC UA PubSub s brokerom**: pre zdieľanie údajov v globálnych sieťach sa používajú protokoly ako MQTT a AMQP. Táto metóda poskytuje bezpečný a vysokoškálovateľný spôsob zdieľania dát medzi viacerými OPC UA odosielateľmi a viacerými OPC UA odberateľmi/prijímateľmi.

2.6.3 OPC UA Field eXchange

OPC UA Field eXchange (**OPC UA FX**) je technológia, ktorá rozširuje aktuálny štandard OPC UA pre priemyselné a automatizačné aplikácie. Tento štandard je navrhnutý tak, aby umožňoval efektívny a bezpečný prenos dát medzi zariadeniami a systémami na



Obrázok č. 20: Dve metódy komunikácie PubSub modelu [76]

najnižšom poschodí automatizačnej pyramídy, čím sa líši od OPC UA (obrázok č.17) [20]. Na tomto poschodí aktuálne fungujú protokoly ako EtherCAT alebo Profinet.

OPC UA FX úzko súvisí s **OPC UA over TSN** (**Time-Sensitive Networking**). TSN je súbor štandardov IEEE pre Ethernet, ktoré umožňujú **deterministickú komunikáciu v reálnom čase**. Vďaka tomu OPC UA over TSN poskytuje spoľahlivé, presné a synchronizované prenosy dát medzi zariadeniami v priemyselnej automatizácii. Vďaka integrácii s TSN umožňuje OPC UA FX deterministickú komunikáciu, čo je kľúčové pre aplikácie, kde je potrebná vysoká presnosť a nízka latencia, ako napríklad v robotike alebo pri presnom riadení pohybu.

Výsledkom použitia OPC UA FX na najnižšej úrovni podnikovej pyramídy je zvýšená efektivita, znížená latencia v komunikácii, a vylepšená interoperabilita a bezpečnosť v rámci priemyselných automatizačných procesov.

3 Vybrané digitálne technológie pre Industry 4.0

V tejto kapitole budú opísané vybrané technológie a softvérové nástroje využívané v edukačných prípadových štúdiách predstavených v predloženej učebnici. Kapitola využíva najmä jazykový aparát a pojmy uvedené v zdrojoch [56], [58] a [83].

3.1 Cloud computing

Cloud computing je na Internete založený model vývoja a využívania počítačových technológií. Je možné ho charakterizovať aj ako poskytovanie služieb, programov alebo komplexných softvérových aplikácií uložených na serveroch na Internete s tým, že používatelia k nim môžu pristupovať napríklad pomocou webového prehliadača alebo programového klienta danej aplikácie a používať prakticky odkiaľkoľvek. Ponuka aplikácií sa pohybuje od kancelárskych aplikácií, cez systémy pre distribuované výpočty (napr. pre úlohy výpočtovej inteligencie), až po celé operačné systémy prevádzkované v prehliadačoch [48].

Je možné rozlíšiť tri základné **modely služieb** cloud computingu (obrázok č. 21):

- **Infraštruktúra ako služba — Infrastructure as a Service (IaaS)** — Poskytovateľ cloudových služieb sa zaväzuje poskytnúť infraštruktúru. Typicky sa jedná a virtualizáciu serverov. Hlavnou výhodou tohto prístupu je fakt, že sa o problémy s hardvérom stará samotný poskytovateľ. Na druhú stranu, vzhľadom k tomu, že hardvér sa často berie ako niečo, čo vlastníme, na čo si môžeme siahnuť a sme za to zodpovední, je niekedy nemožné takýto prístup akceptovať. IaaS je však vhodné pre tých, ktorí vlastnia softvér (alebo licencie) a nechcú sa starať o hardvér. IaaS je flexibilné a škálovateľné, čo znamená, že zákazníci môžu pridávať alebo uberať zdroje podľa potreby.

- **Platforma ako služba — Platform as a Service (PaaS)** — Poskytovateľ garantuje komplexné prostriedky pre podporu celého životného cyklu tvorby a následnej podpory webových aplikácií a služieb. Toto všetko je dostupné na Internete spravidla bez možnosti stiahnutia softvéru. Koncepcia zahŕňa rôzne prostriedky pre vývoj aplikácií, ako je integrované vývojové prostredie (IDE) alebo aplikačné programové rozhranie (API), a tiež napríklad pre údržbu. Z tohto vyplýva, že PaaS poskytuje kompletné softvérové prostriedky pre vývoj softvérových aplikácii — napríklad pripravený operačný systém, databázy, webový server a podobne. Užívateľ sa teda stará len o nasadenie a monitorovanie aplikácie vo vybranom prostredí (Node.JS, Python, Java, C# atď.).

- **Softvér ako služba — Software as a Service (SaaS)** — Aplikácia je licencovaná ako služba prenajímaná konkrétnemu užívateľovi. Užívatelia si teda priamo kupujú prístup k aplikácii a nie samotnú aplikáciu. SaaS je ideálne pre tých, ktorí potrebujú iba bežný aplikačný softvér a vyžadujú prístup k nemu prístup odkiaľkoľvek a kedykoľvek. Príkladom môže byť sada aplikácií Microsoft Office 365 spúšťaná vo webovom prehliadači.

Je možné sa stretnúť aj s rôznymi inými modelmi služieb ako napr. *ID as a Service*, *Business as a Service* alebo *Network as a Service*. Doposiaľ sú však ako základné modely všeobecne prijímané len vyššie uvedené tri.

Obrázok č. 21: Modely služieb cloud computingu [56]

Cloudové služby nesú so sebou viaceré výhody a nevýhody. Medzi hlavné **výhody** možno zaradiť:

- absenciu nutnosti poznať princípy funkčnosti softvéru a hardvéru;
- efektívne riadenie a prácu vďaka dostupnosti dát odkiaľkoľvek – rast produktivity práce vo firmách;
- možnosti obnovenia dát v prípade problémov;

- principiálne vyššie zabezpečenie dát;
- jednoduchosť používateľského rozhrania;
- rýchle prispôsobenie IT zázemia rastu a potrebám užívateľa;
- možnosti okamžitého zvýšenia výkonu dátového centra (škálovateľnosť).

Cloudové služby sprevádza aj viacero nevýhod, pričom užívateľ si musí sám zvážiť, či výhody prevyšujú nad nevýhodami. Toto závisí najmä od konkrétnych požiadaviek na aplikáciu. Medzi **nevýhody** cloudových služieb patrí najmä:

- závislosť na internetovom pripojení;
- závislosť na poskytovateľovi;
- zlá reputácia cloud computingu – *tzv. Big Brother*, otázky ohľadne súkromia, bezpečnosti;
- menej funkcií, obmedzenie prispôsobenia a horšia stabilita – závisí však od prípadu k prípadu;
- migračné náklady;
- odlišné právne pravidlá poskytovateľa a klienta – poskytovateľ môže byť v USA a klient v inej krajine, ktorá je podriadená inej jurisdikcii.

Poznáme viacero možností **nasadenia cloudových systémov** [62]:

- **Verejný** — Verejný cloud umožňuje, aby systémy a služby boli jednoducho dostupné pre širokú verejnosť. IT giganti ako Amazon, Google a Microsoft ponúkajú cloudové služby prostredníctvom Internetu. Výhodami takéhoto cloudu sú spoľahlivosť, flexibilita, efektivita výdavkov, nezávislosť od lokality a vysoká škálovateľnost. Takýto model však prináša tiež nevýhody ako napríklad menšiu prispôsobiteľnosť riešenia.
- **Skupinový (zdieľaný)** – Tento typ nasadenia umožňuje, aby systém a služby boli prístupné istej skupine organizácií. Zdieľa infraštruktúru medzi viacerými organizáciami z konkrétnej komunity. Toto môže byť spravované interne danou organizáciou alebo treťou stranou. Výhodou uvedeného modelu je efektivita výdavkov, zdieľanie medzi organizáciami a tiež lepšie zabezpečenie, ako je to pri verejnom servisnom modeli. Nevýhodou môže byť uloženie dát na jednom mieste.

- **Súkromný** — Na rozdiel od predošlého, súkromný typ nasadenia umožňuje prístup k službám a systémom len v rámci jednej organizácie. Môže byť spravovaný interne danou organizáciou alebo treťou stranou. Medzi výhody patrí vysoká miera súkromia, väčšia kontrola nad službami a tiež väčšia efektivita. Medzi nevýhody možno zaradiť limitovanú škálovateľnosť v rámci fyzických zdrojov, spravidla vysokú cenu, obmedzenie dosahu a údržba. Niekedy sa tento typ ani nepovažuje za cloud.

- **Hybridný** — Je zmesou súkromného a verejného cloudu. Nekritické úlohy sa vykonávajú pomocou verejného cloudu, avšak kritické úlohy sa vykonávajú pomocou súkromného cloudu. Výhodami tohto riešenia sú flexibilita, škálovateľnosť, zabezpečenie a cenová efektivita. K nevýhodám je možné zaradiť problémy so sieťou vzhľadom ku komplexnosti riešenia, súlad bezpečnostných opatrení cloudu a samotnej organizácie a závislosť na internet IT infraštruktúre. Z tohto dôvodu je nutné zabezpečiť redundanciu v dátových centrách.

3.2 Internet of Things

Internet of Things (alebo v slovenčine niekedy Internet vecí), skrátene IoT, je v súčasnosti široko využívaný pojem v oblasti digitálnych a informačno-komunikačných technológií. Termín Internet of Things bol prvýkrát spomenutý profesorom Kevinom Ashtonom v rámci svojej prezentácie v roku 1999, pričom jeho prvá publikácia o IoT bola uverejnená v roku 2009 [2].

Existuje viacero definícií IoT. Vo všeobecnosti *IoT označuje sieť prepojených objektov (vecí), ktoré sú jednoznačne adresovateľné. Táto sieť je založená na štandardizovaných komunikačných protokoloch umožňujúcich výmenu a zdieľanie dát a informácií. Ich analýzou bude možné docieliť vyššiu pridanú hodnotu — zisk znalostí* [77]. Prepojenie zariadení by malo byť najmä bezdrôtové a malo by priniesť nové možnosti vzájomnej interakcie nielen medzi jednotlivými systémami, ale tiež priniesť nové možnosti ich ovládania, sledovania a zaistenie pokročilých služieb [57].

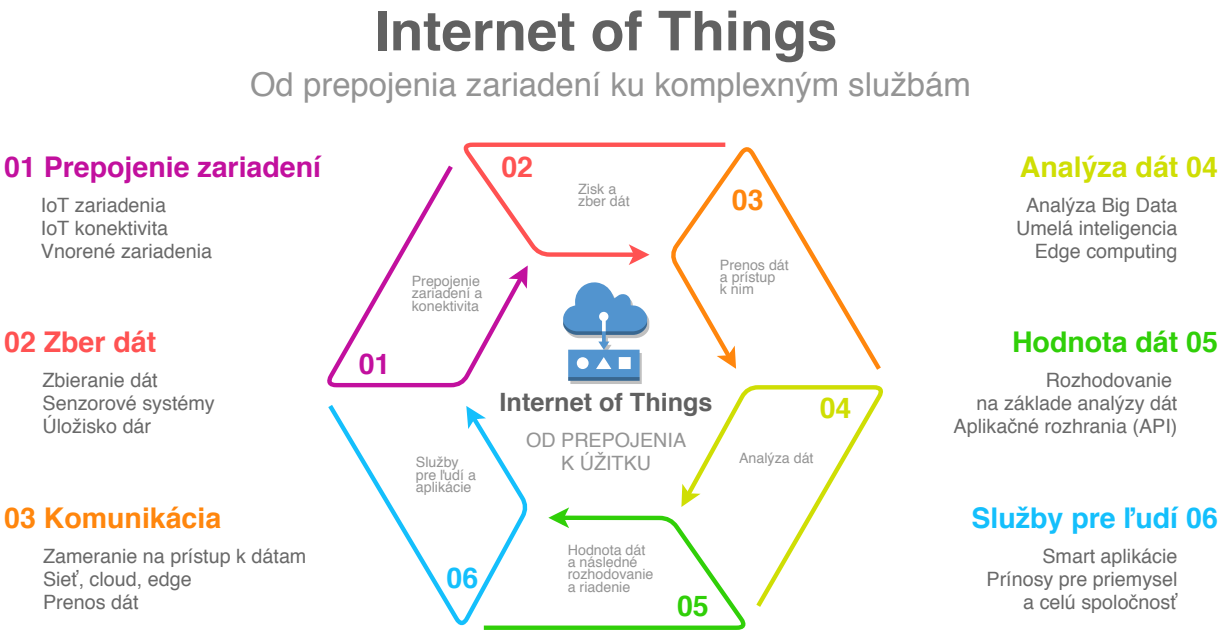
Slovo sieť nemusí znamenať len Internet, ako to evokuje pojem Internet of Things. Môže ísť aj o lokálnu sieť (LAN), v rámci ktorej môžu jednotlivé veci komunikovať, avšak s prístupom do Internetu pre možnosť zdieľania výsledkov.

Vec z pohľadu Internet of Things predstavuje neživý objekt, ktorý môže byť fyzický alebo virtuálny, obsahujúci elektronické prvky (senzory, aktuátory) a softvér. Pomocou senzorov sníma určitú veličinu alebo viacero veličín. Ide teda o zariadenie alebo systém autonómne poskytujúci dáta, ktoré sú zdieľané s ďalšími vecami alebo systémami. V IoT nie sú základom veci ale dáta nimi poskytované.

Cieľom IoT je prepojenie zariadení, systémov a služieb na poskytnutie **dát**, ktoré je následne možné previesť na **informácie** a tieto informácie na **znalosti** (angl. know-

ledge). Následne ich možno v danom systéme ďalej využiť napríklad aj na úlohy **inteligentného rozhodovania a riadenia**.

Striktne vzaté Internet of Things nie je príkladom konkrétnej digitálnej technológie, ide o zovšeobecňujúci pojem.



Obrázok č. 22: Fungovanie systému Internet of Things [56]

Hlavné požiadavky na systém IoT vychádzajú z vyššie uvedených cieľov. Architektúra IoT musí teda umožniť:

- zber dát / informácií / znalostí;
- uloženie dát / informácií / znalostí;
- analýzu dát / informácií / znalostí;
- zdieľanie / publikovanie výsledkov.

Dôležitým faktom je, že architektúra musí spĺňať prísne požiadavky na bezpečnosť. Medzi ďalšie čiastkové požiadavky je treba zaradiť aj **interoperabilný a efektívny** prenos dát. S týmto súvisí voľba vhodného prenosového štandardu a dátového modelu. V systémoch IoT je nutné spracovávanie veľkého objemu dát (**tzv. Big Data**). Aby bolo možné získané dáta, informácie a znalosti kombinovať a fúzovať, je nevyhnutné dosiahnuť **sémantickú interoperabilitu** [77].

V rámci Internet of Things sú vytvorené dva hlavné smery:

1. Spotrebiteľský IoT (**Consumer IoT** — angl. skratka **CIoT**);

2. Priemyselný IoT (**Industrial IoT** — angl. skratka **IIoT**).

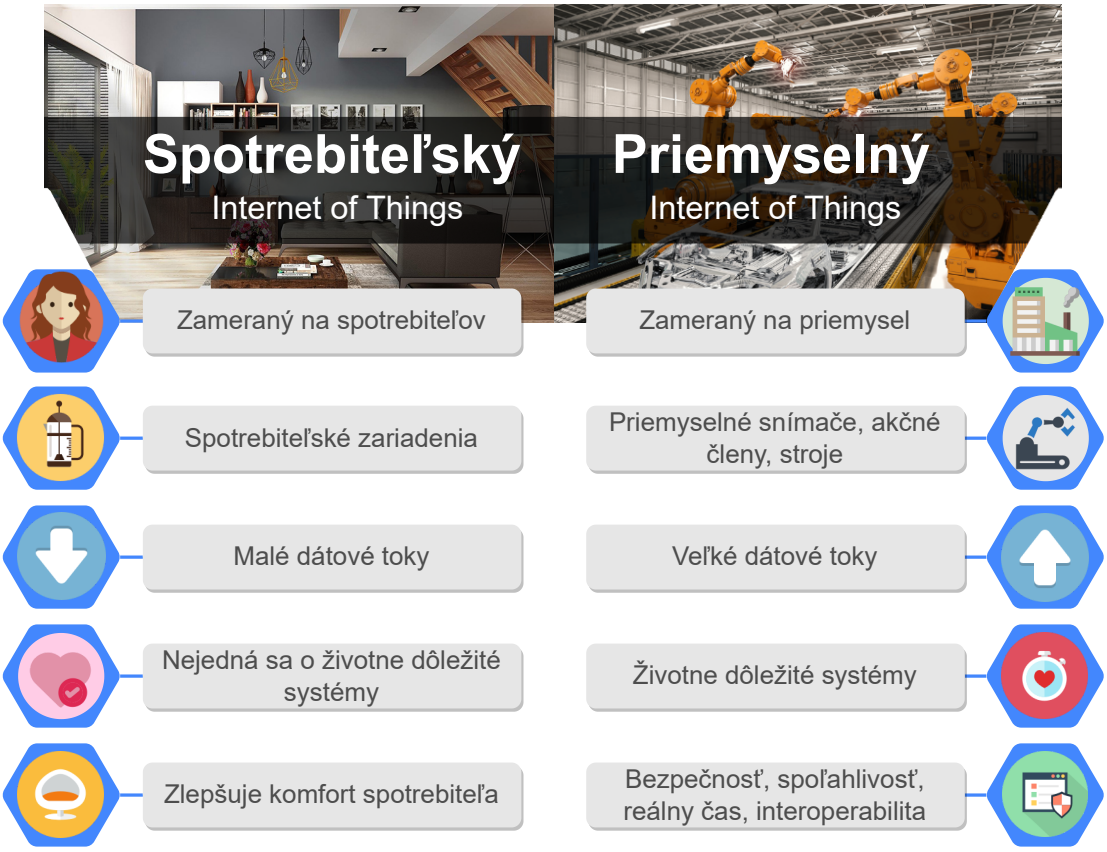
Spotrebiteľský IoT sa zameriava na spotrebiteľské zariadenia, domáce elektrospotrebiče, osobné telekomunikačné zariadenia a podobne. **Cieľom CIoT je zjednodušenie každodenného života, zvýšenie komfortu a užívateľského zážitku.** Príkladom môže byť automatizácia domácnosti, inteligentné práčky, chladničky, televízory, osvetlenie alebo nositeľná elektronika.

Priemyselný IoT sa zameriava na kľúčové alebo *kritické* úlohy. Ideovo vychádza z komunikácie M2M a rozširuje ju napr. o možnosť analýzy dát (spravidla v cloude) a o ďalšie aspekty IoT. V prípade IIoT sa prepájajú zariadenia a systémy, ktoré sa využívajú v rôznych priemyselných odvetviach a službách (dopravný priemysel, energetický priemysel, chemický priemysel, zdravotníctvo atď.). Z uvedeného vyplýva, že IIoT je neoddeliteľnou súčasťou digitálnych podnikov. **Hlavným cieľom IIoT je efektívnejšie využívanie zdrojov, zníženie prevádzkových nákladov, zvýšenie pracovnej produktivity a bezpečnosti pracovníkov, predchádzanie výpadkov pomocou monitorovania a včasnej (alebo prediktívnej) údržby, a tým dosiahnutie výrazných úspor a vrátenie investíc.** Predpokladá sa, že tento segment IoT bude prevládajúci [77]. Porovnanie segmentov možno vidieť na obrázku č. 23.

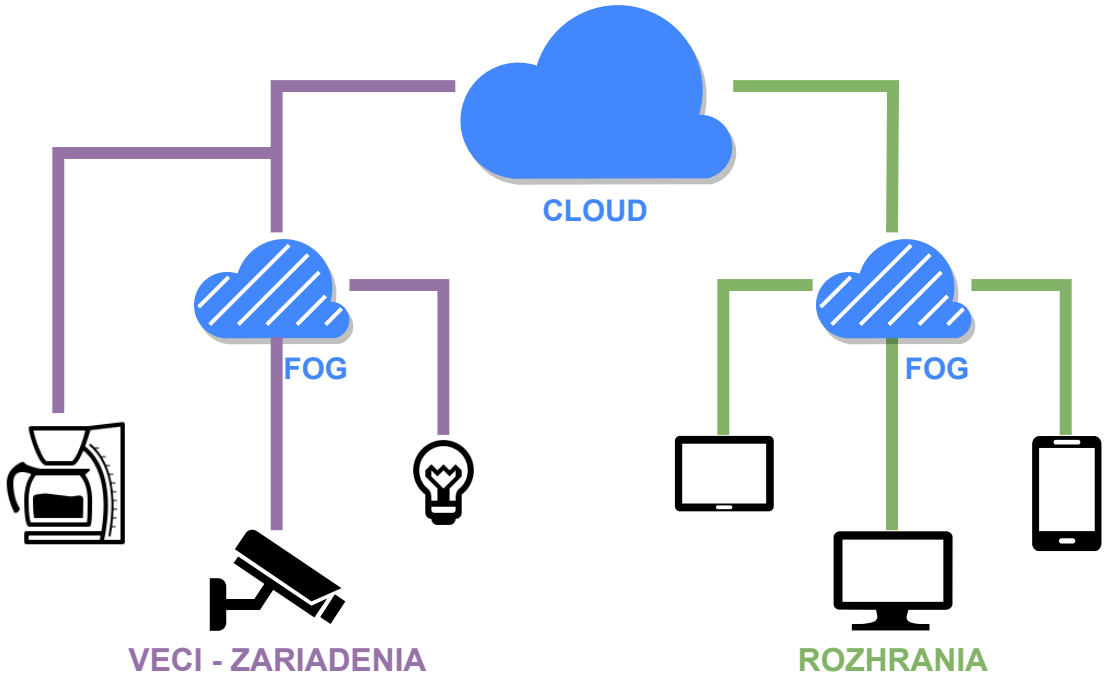
Prepojenie medzi jednotlivými prvkami systému IoT závisí od účelu tohto systému. Existuje niekoľko spôsobov prepojenia a spôsobov komunikácie (obrázok č. 24). Komunikácia môže teda prebiehať (väčšinou v rôznych kombináciách) nasledovne:

- medzi zariadeniami navzájom (tzv. fog computing);
- od zariadení do cloudu;
- medzi cloudmi.

Komunikácia medzi zariadeniami (vecami) je použitá v systémoch, kde využitie cloudu pre ukladanie, vyhodnocovanie a následné zdieľanie dát nie je pre dané riešenie vyhovujúce. Toto môže byť napríklad v prípadoch, kedy nie je k dispozícii dostatočná kapacita internetovej linky pre zaslanie všetkých dát alebo zaslanie veľkého množstva dát by bolo finančne náročné. Niekedy je nutné, aby IoT systém fungoval aj v prípade obmedzeného pripojenia do cloudu. Pri systémoch pracujúcich v reálnom čase je doba pre zaslanie dát do cloudu a späť neprijateľná. Pre niektoré systémy teda nie je centralizované cloudové riešenie vhodné a namiesto neho sa využíva decentralizovaný **fog computing**. V ňom spolu zariadenia (veci) komunikujú priamo medzi sebou (peer-to-peer). Fog computing dopĺňa cloud computing o možnosť realizovať zber, ukladanie,



Obrázok č. 23: Porovnanie spotrebiteľského a priemyselného IoT



Obrázok č. 24: Spôsoby komunikácie v IoT systéme

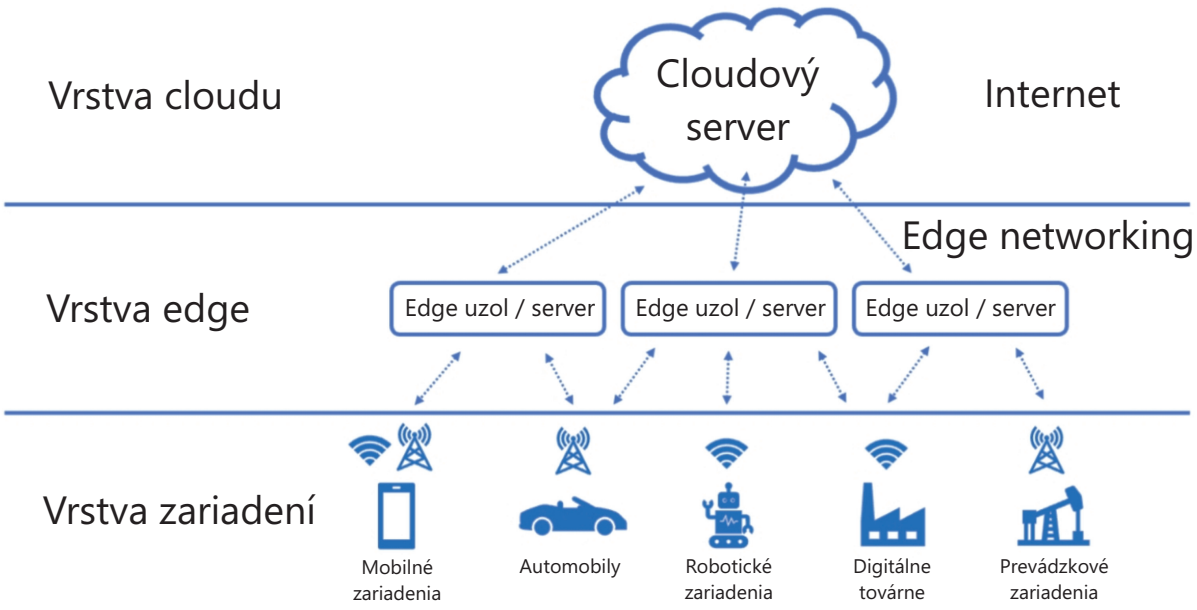
analýzu a zdieľanie dát bližšie k zariadeniam a ich dátam (lokálne spracovanie). Týmto sa umožňuje dosiahnutie lepšej škálovateľnosti, spoľahlivosti, rýchlejšej odozvy a tiež zníženia nákladov. Tento prístup sa využíva najmä v IIoT, kde je dôraz na spoľahlivosť a definované odozvy systémov. Výsledky spracovania dát sú následne odovzdávané do cloudu [77].

Komunikácia od zariadenia do cloudu sa často využíva v rámci CIoT, kde sa nekladie taký dôraz na rýchlosť odozvy. Prípadný výpadok spojenia nemá v spotrebiteľských systémoch závažné dopady.

Komunikácia medzi cloudmi sa používa v prípade nutnosti zdieľania dát medzi rôznymi doménami – teda napríklad medzi privátnym a verejným cloudom.

3.3 Edge computing

Pokiaľ majú byť nové riešenia IoT implementované predpokladaným tempom, je nutné využívať nové technológie, ktoré sú založené na nových myšlienkových princípoch. Takáto filozofia je v súlade s myšlienkou spracovania dát *na okraji* — **edge computing**. Tento koncept zahŕňa spracovanie dát už na mieste vstupu dát do siete, čím sa dosiahne zníženie dátového toku a najmä latencie, čo je kľúčové pre riešenia v rámci IIoT. Do cloudového prostredia sa tak odosielajú iba **redukované relevantné dáta**. Týmto sa znižuje množstvo dátového odpadu, ktorý by cloud musel spracovávať.



Obrázok č. 25: Edge computing v kontexte sieťovej komunikácie [92]

Edge computing teda opisuje spracovanie údajov v blízkosti zdroja týchto dát, pri-

čom nemusia byť odosielané do vzdialeného cloudu alebo iných centralizovaných systémov na spracovanie (obrázok č. 25). Jednou z motivácií často býva prinesenie umelej inteligencie (tzv. *edge intelligence*), spracovanie a analýza údajov bližšie k inteligentným snímačom a akčným členom alebo priamo do nich. Odstránením vzdialenosti a času potrebného na odosielanie údajov do centralizovaných systémov je možné zlepšiť výkonnosť a odozvu navrhnutého systému [4].

Edge computing je podobný fog computing a tieto dva termíny sa niekedy zamieňajú. Edge zariadenie môže byť napríklad sieťová brána alebo iný priemyselný prvok, ktorý dokáže ponúknuť výpočtový výkon. Fog computing umiestňuje inteligenciu do lokálnej siete, pričom edge computing sa ju snaží umiestňovať priamo do samostatných koncových zariadení.

Edge computing môže byť považovaný za súčasť inteligentných technológií, najmä keď sa používa v kontextoch, kde podporuje pokročilé spracovanie dát, autonómne rozhodovanie a interakciu v reálnom čase blízko zdroju dát. Edge computing posúva výpočtové zdroje a spracovanie dát bližšie k miestam, kde dáta vznikajú (napríklad IoT zariadenia, senzory, aktuátory atď.), čo minimalizuje latenciu, znižuje potrebu prenosu veľkých objemov dát do vzdialených dátových centier a umožňuje rýchlejšie a efektívnejšie rozhodovacie procesy. Edge computing umožňuje lokálne vykonávanie modelov výpočtovej inteligencie a strojového učenia, čo zlepšuje schopnosť systémov rýchlo reagovať na zmeny a učiť sa z nových informácií bez potreby neustáleho spojenia s cloudom. Edge computing tiež umožňuje spracovanie dát takmer v reálnom čase tým, že redukuje oneskorenie spôsobené prenosom dát medzi zariadením a cloudom. Toto je kriticky dôležité pre aplikácie vyžadujúce okamžité spracovanie a reakciu, ako sú autonómne vozidlá, robotika, digitálne továrne, inteligentné mestá a iné systémy využívajúce pokročilú automatizáciu.

3.4 Kontajnerizácia

Ak chceme na serveri nasadiť softvérovú aplikáciu, je potrebné mať na danom serveri operačný systém a prostredie, v ktorom bude táto aplikácia bežať. Ak by na serveri bežala len jedna softvérová aplikácia, tak by s veľkou pravdepodobnosťou nevyužila plný poskytovaný výkon servera. Logickým dôsledkom je nasadenie viacerých aplikácií na jeden server. Tu je však možné naraziť na problém, že tieto aplikácie využívajú rôzne verzie istej knižnice, prípadne sa môžu tieto aplikácie navzájom ovplyvňovať [4].

Jedným z riešení by bolo vytvorenie virtuálnych strojov na serveri, čo by však bolo neefektívne. Ide o to, že každý virtuálny stroj by si ukrojil istý výpočtový výkon zo servera len na jeho samostatný beh a nielen na beh aplikácie. Tento problém rieši tzv. **kontajner**. Tak ako v reálnom svete existuje kontajner na prepravu softvéru, tak v digi-

tálnom svete existuje kontajner na *univerzálnu prepravu softvéru* [5].

V reálnom svete je kontajner vyrobený tak, aby sa dal ľahko prepraviť v rôznych prepravných prostriedkoch, ako je kamión, loď, lietadlo alebo vlak. Cieľom kontajnera v digitálnom svete je zabaliť aplikáciu tak, aby sa dal čo najjednoduchšie prepraviť na rôzne systémy. Stručne povedané, kontajner je samostatný balík softvéru, ktorý obsahuje všetky potrebné závislosti, aby aplikácia mohla byť spustená a bežať nezávisle od prostredia, v ktorom bola vytvorená [4].

Hlavnou myšlienkou kontajnerizácie je **oddelenie aplikácie a jej závislostí od operačného systému**. Medzi spomínané závislosti patrí kód, runtime, systémové nástroje, knižnice a nastavenia [4].

Kontajner by mal obsahovať len jednu aplikáciu, vďaka čomu je možné získať **mikroservisovo orientovanú architektúru**, ktorá je v súčasnej dobe veľmi populárna [5]. Mikroservisová architektúra umožňuje jednoduchým spôsobom udržiavať a škálovať softvérové aplikácie.

Kontajnery by mali dodržiavať tieto tri princípy [4]:

1. **Štandardnosť** — Kontajnery sú vytvorené podľa štandardu, ktorý opisuje, že sú kompletne oddelené od systému, pomocou ktorého boli vytvorené, vďaka čomu sú prenositeľné na iné operačné systémy. Všetky závislosti, ako programovací jazyk, knižnice a iné, sú súčasťou kontajnera.

2. **Jednoduchosť** — Kontajner by mal obsahovať iba jednu softvérovú aplikáciu. Viac aplikácií v kontajneri by sa mohlo navzájom ovplyvňovať, čo by malo v konečnom dôsledku priniesť aj zložitosť pri identifikovaní problémov. Kontajnery zdieľajú jadro operačného systému a jeho zdroje, vďaka čomu nie je potrebný vlastný operačný systém pre každý kontajner.

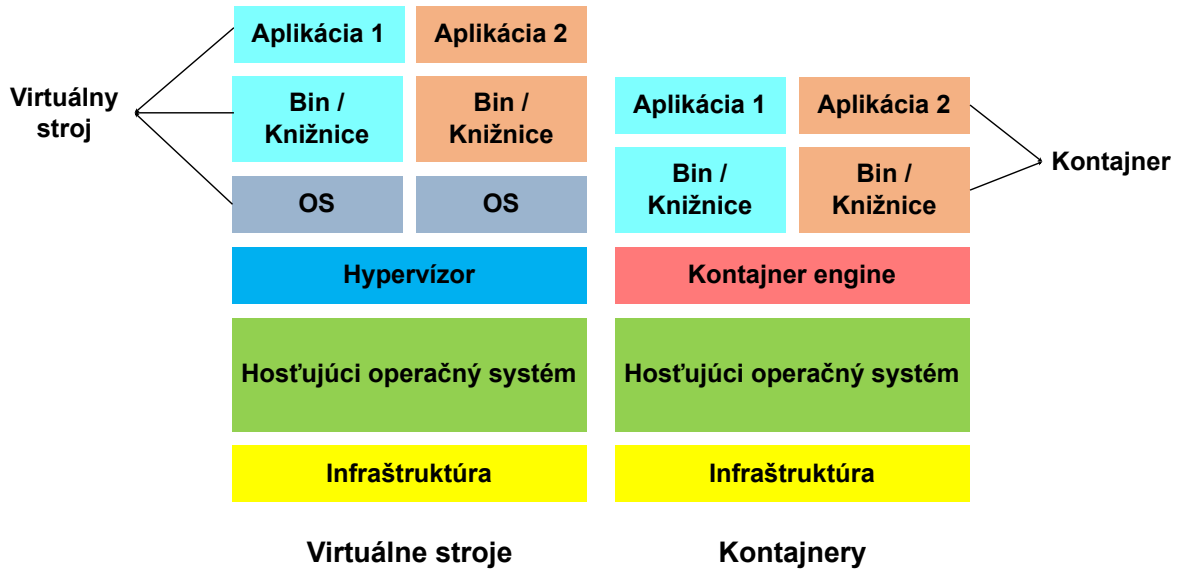
3. **Izolovanosť** — Kontajnery virtualizujú centrálny procesor, pamäť, disk a sieť na úrovni operačného systému. Kontajner je vďaka tomu izolovaný a nevznikajú žiadne ďalšie konflikty medzi internými procesmi kontajnera a externými procesmi hostiteľského operačného systému. Hostiteľ môže komunikovať s kontajnerom iba prostredníctvom jadra, čo poskytuje vyššiu bezpečnosť.

3.4.1 Porovnanie virtuálneho stroja a kontajnera

Virtuálny stroj (angl. skratka VM — virtual machine) je obraz (angl. image), ktorý sa správa ako reálny počítač a jeho vnútro tvorí operačný systém. Vďaka tomu sa správa ako sandbox a je oddelený od hostiteľského operačného systému. Na jednom hostiteľskom stroji môžu bežať viaceré VM, o ktorých beh sa stará **hypervízor**. VM funguje tak, že si

vyhradí virtuálny hardvér, ako je centrálny procesor, pamäť, úložiská a sieťové rozhrania, ktoré sú následne priradené na reálny hardvér.

Prvý problém VM oproti kontajnerom spočíva v tom, že VM potrebuje vlastný operačný systém, čo spôsobuje **vyššie nároky na výkon hardvéru**. VM potrebuje niekedy až desiatky GB miesta v porovnaní s niekoľkými MB, ako je to v prípade kontajnerov. Pri kontajneri sa spúšťa iba samostatná softvérová aplikácia, avšak pri VM sa spúšťa jadro a viacero systémových služieb (obrázok č. 26). Druhým problémom je čas potrebný na inicializáciu VM, keďže VM si potrebuje vyhradiť vlastný hardvér a spustiť operačný systém. Naproti tomu kontajner sa dokáže inicializovať v priebehu niekoľkých sekúnd. Kontajnery sú kompatibilné s rôznymi novšími technológiami, ako sú Continues Integration/Continues Delivery (CI/CD) a DevOps, ktoré pomáhajú znižovať zložitosť a zvyšovať efektivitu nasadzovania a údržby aplikácií [93]. Oproti tomu sú VM určené pre tradičné a monolitické IT architektúry [4].



Obrázok č. 26: Porovnanie virtuálneho stroja a kontajnera [4]

Kontajnerizácia je technológia, ktorá umožňuje balenie softvéru spolu so všetkými jeho závislosťami do kontajnerov, čo zjednodušuje nasadenie, škálovanie a správu aplikácií v rôznych výpočtových prostrediach. Z hľadiska nasadzovania inteligentných technológií do praxe je táto flexibilita kľúčová, nakoľko sa vyžaduje rýchla adaptácia na meniace sa požiadavky a podmienky. V kontexte inteligentných technológií teda kontajnerizácia slúži ako kľúčová podporná infraštruktúra, ktorá umožňuje lepšiu dostupnosť, efektívnosť a flexibilitu pri vývoji a nasadení inteligentných riešení. Aj keď sama o sebe neposkytuje „inteligenciu“, je nevyhnutná pre efektívne využívanie a škálovanie inteligentných technológií v rôznych aplikáciách a prostrediach.

3.4.2 Docker

Docker (Docker engine) patrí medzi najznámejšie otvorené platformy pre vývoj a **kontajnerizáciu aplikácií**.

Docker využíva klient-server architektúru. Vývojári používajú klienta na komunikáciu s docker daemonom, ktorý zastavuje, štartuje a vykonáva rôzne ďalšie operácie nad kontajnermi. Klient a daemon môžu bežať na rovnakom systéme, ale môžu bežať aj na rozdielnych systémoch. Komunikácia medzi daemonom a klientom sa väčšinou rieši cez REST API.

Dôležitou súčasťou architektúry je tzv. *docker registry*, ktorý tvorí úložisko pre **docker obrazy**. K dispozícii sú privátne a verejné registre. Ako už hovorí názov, privátne registre slúžia na ukladanie obrazov, ktoré sú dostupné len pre konkrétnych špecifických používateľov. Docker obrazy sú read-only šablóny, ktoré slúžia na vytvorenie **docker kontajnerov**. Docker kontajner predstavuje špecifickú bežiacu inštanciu docker obrazu. Kontajner je možné vytvoriť, zapnúť, vypnúť, odstrániť alebo pripojiť k sieťam a úložiskám. Docker ponúka aj tzv. *Docker swarm*, ktorý slúži na jednoduché škálovanie kontajnerov podľa potreby.

Medzi ďalšie nástroje pre kontajnerizáciu patrí napríklad Containerd, Podman alebo Rocket (Rkt). Je však potrebné uviesť, že sú zásadne menej rozšírené ako Docker.

Containerd je minimalistický, stabilný a spoľahlivý runtime pre kontajnery. Je ideálny pre vývojárov, ktorí vyžadujú len základné funkcionality. Nevýhodou je menší ekosystém a absencia grafického rozhrania pre správu. Trvá teda dlhšie sa naučiť s týmto nástrojom pracovať.

Podman je open-source nástroj pre kontajnerizáciu, ktorý je postavený na *daemonless* architektúre. Daemon alebo démon je označenie programu, ktorý je spustený dlhodobo a nie je v priamom kontakte s používateľom. Znamená to, že v prípade Podmanu démon pre riadenie kontajnerov nie je potrebný, čo znižuje bezpečnostné riziká a riziká zlyhania. Nevýhodou je tiež strmá krivka učenia a menší ekosystém.

Rocket alebo **Rkt** je jednoduchý nástroj pre kontajnerizáciu, ktorý kladie dôraz na bezpečnosť. Takisto nevyužíva centrálneho démona.

3.5 Počítačom generovaná realita

Ďalšou z digitálnych technológií je **počítačom generovaná realita** (**PGR**). PGR, ako moderná technológia, je v Industry 4.0 využitá pri virtualizácii efektívneho projektovania optimálnych výrobných štruktúr a pracovných operácií s ich efektívnym ergonomickým vyhodnotením a návrhom. V digitálnych podnikoch sa v súčasnosti hľadajú nové formy monitorovania, riadenia, diagnostiky a vizualizácie procesov. Práve počítačom generovaná realita takéto formy prináša [56].

Rozlišujeme **virtuálnu** (angl. virtual), **rozšírenú** (angl. augmented) a **zmiešanú** (angl. mixed) **realitu**, pričom v angličtine pre tieto *reality* existuje súhrnné označenie **extended reality** [62]. V slovenčine zatiaľ nebol prijatý všeobecne uznávaný preklad. Logicky sa ponúka názov *rozšírená realita*, ktorý však označuje *augmented* realitu, čo by striktne vzaté malo byť preložené ako *dopĺňaná* realita. Na pracovisku autora učebnice bol prijatý názov **počítačom generovaná realita**, ktorý bol zavedený aj v zdroji [56] a bude využívaný aj v predloženej učebnici. Možno sa stretnúť aj s prekladom **nadstavbová realita** [62].

Všeobecne prijímanou skratkou pre PGR je v angličtine **XR** (z angl. eXtended reality).

Takisto momentálne nepanuje všeobecná zhoda pri rozlišovaní rozšírenej a zmiešanej reality. Existujú dve hlavné definície, ktoré sú predstavené v nasledujúcom texte.

3.5.1 Definícia PGR podľa The Foundry

Definície od spoločnosti The Foundry [47], ktorá sa venuje vývoju softvérov pre 3D modelovanie a textúrovanie, sa budeme držať aj v predloženej učebnici. Táto definícia sa čoraz viac začína presadzovať aj v priemyselnej praxi [56].

Virtuálna realita (**VR**) simuluje fyzickú prítomnosť na miestach v reálnom svete alebo v zmyslenom svete, pričom dovoľuje užívateľovi interakciu s týmto svetom a v tomto svete. Neoddeliteľnou zložkou virtuálnej reality sú zážitkové vnemy. Pod týmto pojmom si môžeme predstaviť vizuálne, zvukové a prípadne aj ďalšie vnemy (napr. haptická odozva). Dokonalé virtuálne prostredie by ovplyvňovalo všetkých päť zmyslov, avšak nie je to nutnou podmienkou [62]. Zariadeniami pre zobrazovanie virtuálnej reality sú náhlavné súpravy, pričom z angličtiny sa ujal aj v slovenčine názov *headsety*. Headset môže byť spojený káblom alebo bezdrôtovo s počítačom (napr. séria zariadení HTC Vive), smartfónom (napr. Samsung Gear VR) alebo pracuje samostatne, čo znamená, že má integrovanú výpočtovú jednotku (napr. séria zariadení Meta Quest), a teda nepotrebuje výpočtovú jednotku externú v podobne počítača alebo smartfónu. Pokročilé headsety dokážu snímať aj ovládače, ktoré má užívateľ v rukách, bez toho, aby bola nutná inštalácia ďalších senzorov v miestnosti. Snímanie prebieha pomocou kamier umiest-

nených v headsete, pričom priekopníkom tejto technológie bola spoločnosť Microsoft.

Obrázok č. 27: Virtuálna, rozšírená a zmiešaná realita podľa The Foundry — príklad prevzatý z literatúry [46]

Rozšírená realita (AR) je živý, priamy alebo nepriamy náhľad na fyzický, reálny svet, ktorý je doplnený (rozšírený) počítačom generovanými prvkami, ako sú zvuk, video, grafika alebo GPS dáta. Rozšírená realita je vrstva obsahu nad reálnym svetom, pričom tento obsah **nie je ukotvený** k tomuto svetu alebo jeho časti. Prvky reálneho sveta a počítačom generovaný obsah nemôžu medzi sebou reagovať. V praktických aplikáciách je zmyslom rozšírenej reality najmä doplniť snímaný obraz o dodatočné informácie [83].

Zmiešaná realita (MR) je spojenie reálneho a virtuálneho sveta, čím vzniká nové prostredie a vizualizácia, kde fyzické a digitálne objekty koexistujú a **interagujú medzi sebou** v reálnom čase, čo je jej kľúčovou charakteristikou. Zmiešaná realita je vrstva umelého (digitálneho) obsahu v reálnom svete, ktorá je **ukotvená** a interaguje s reálnym svetom. Črtou zmiešanej reality je nutnosť pokročilého vnímania prostredia a objektov zariadením pre jeho podporu [56]. Poznáme dva typy zariadení pre podporu zmiešanej reality:

- **Mobilné zariadenia** — Prvou možnosťou je sledovanie zmiešanej reality na displeji mobilného zariadenia, pričom do tejto kategórie zaraďujeme najmä smartfóny a tablety. Samozrejme, tablety sú v tomto smere výhodnejšie, keďže ponúkajú väčšiu obrazovku. Snímanie prostredia prebieha pomocou kamery mobilného zariadenia, pričom v súčasnosti pri pokročilých modeloch ide o sústavu kamier a iných senzorov (time-of-flight senzor alebo dokonca lidar). Knižnice pre vývoj

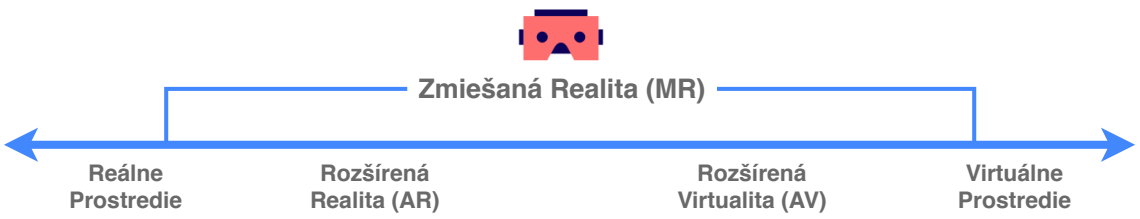
mobilných aplikácií pre zmiešanú realitu sú napríklad Android ARCore alebo Apple ARKit. Samozrejme, pomocou nich vieme vyvíjať aj aplikácie rozšírenej reality.

- **Headsety** — Pomocou headsetu môžeme sledovať počítačom doplnené prvky v realite prostredníctvom opticky priesvitného zobrazovača. Takýto headset ponúka kvalitatívne úplne odlišný, a samozrejme kvalitnejší, zážitok ako mobilné zariadenie. Prakticky jedinou a relevantnou sériou dostupných headsetov je Microsoft HoloLens [59], pričom prvá verzia bola predstavená v roku 2015. Nástupca bol predstavený v roku 2019 pod názvom Microsoft HoloLens 2 [56]. Tretia verzia je momentálne v roku 2024 stále vo vývoji. V priemysle (napr. v spoločnosti Volkswagen) sa využíva headset Daqri Smart Glasses [49].

Niekedy sa chybne v neodborných zdrojoch uvádza, že rozdiel medzi rozšírenou a zmiešanou realitou je v tom, že rozšírenú realitu sledujeme cez obrazovku mobilného zariadenia a zmiešanú realitu cez headset. Takáto interpretácia však nemá oporu v žiadnej definícii, ktorá by bola autorovi učebnice známa.

3.5.2 Definícia PGR podľa Milgrama a Kishiho

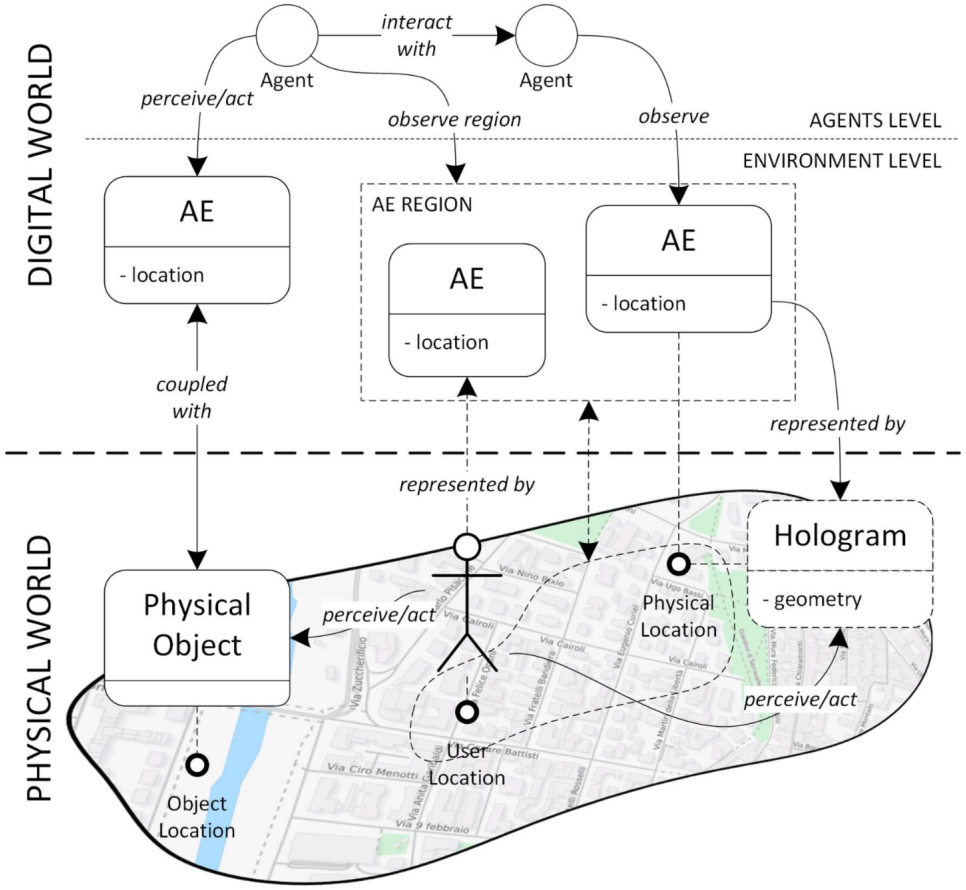
Nasledujúca definícia sa často vyskytuje skôr vo vedeckých kruhoch ako v priemyselnej praxi. V roku 1994 autori Milgram a Kishiho predstavili prechod medzi reálnym a virtuálnym prostredím — **tzv. virtuálno-realitné kontinuum** (angl. reality-virtuality continuum) [70]. Toto kontinuum definuje rôzny mix reálneho a virtuálneho sveta. Zmiešanú realitu teda chápu ako čokoľvek medzi skutočnou realitou a plnou virtuálnou realitou (obrázok č. 28). Medzi realitou a virtuálnou realitou rozlišuje *rozšírenú realitu* (angl. augmented reality), ktorá je prakticky zhodná s rozšírenou realitou podľa The Foundry, a *rozšírenú virtualitu* (angl. augmented virtuality). Možno povedať, že rozšírená virtualita korešponduje so zmiešanou realitou podľa definície The Foundry.



Obrázok č. 28: Virtuálno-realitné kontinuum podľa Milgrama a Kishiho [56]

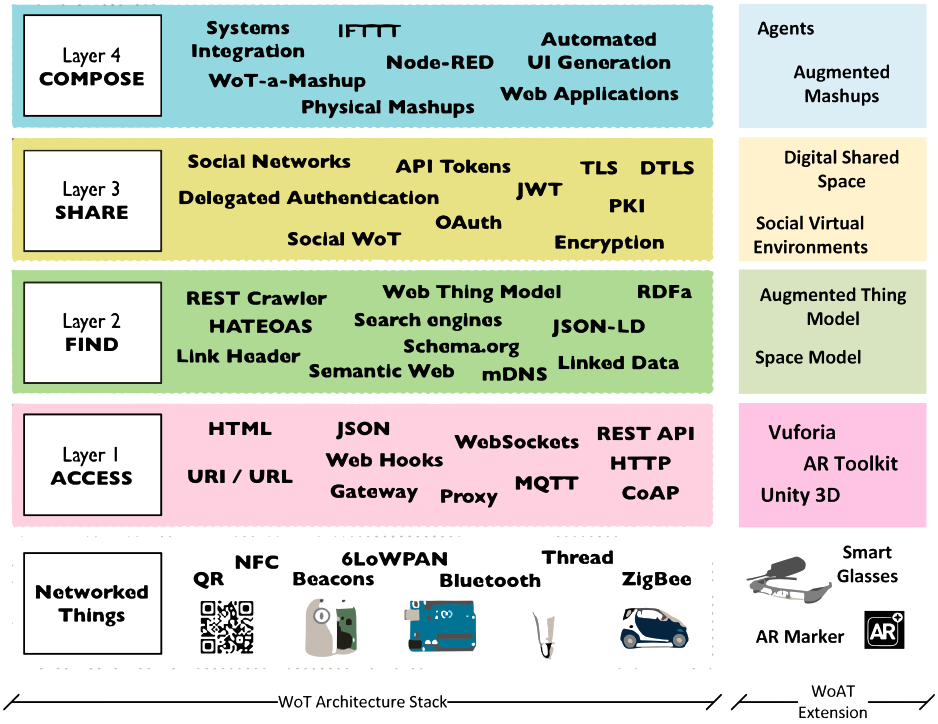
3.5.3 Vybrané aplikácie počítačom generovanej reality

V nasledujúcom texte sú uvedené vybrané výskumné projekty a vedecké práce, ktoré sa zaoberajú možnosťou využitia rozšírenej alebo zmiešanej reality pre oblasť ovládania a monitorovania mechatronických systémov v sieťach Internet of Things. Výber bol inšpirovaný publikáciou [56].



Obrázok č. 29: Reprezentácia rozšíreného sveta Augmented World [18]

V publikáciách [87], [28] sa hovorí o využití štandardných komunikačných protokolov, ktoré by mali byť používané pri návrhu IoT systému s implementáciou webových štandardov. Týmto vzniká tzv. Web of Things (slovensky Web vecí). Web of Things je navrhnutý pre jednoduchú integráciu systémov do súčasného webu. Ide teda o myšlienku vytvorenia spoločnej aplikačnej úrovne pre IoT založenej na webových technológiách a protokoloch. Následne bola táto myšlienka v práci [18] rozšírená o pojem *Augmented Worlds*. Koncept rozšíreného sveta Augmented World môže byť definovaný ako softvérová aplikácia, ktorá do okolitého fyzického prostredia (napr. mesto, budova, miestnosť) pridáva počítačové objekty, s ktorými môžu používatelia alebo softvéroví agenti interagovať (obrázok č. 29). Kombináciou Web of Things a Augmented World vznikol koncept Web of Augmented Things (obrázok č. 30).

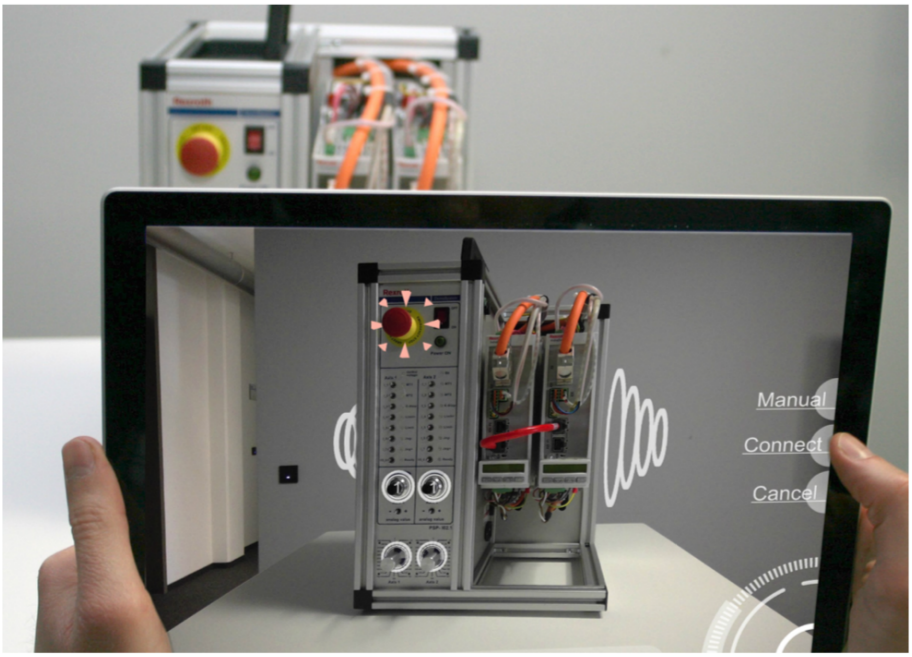


Obrázok č. 30: Vrstvy Web of Things rozšírené na Web of Augmented Things [18]

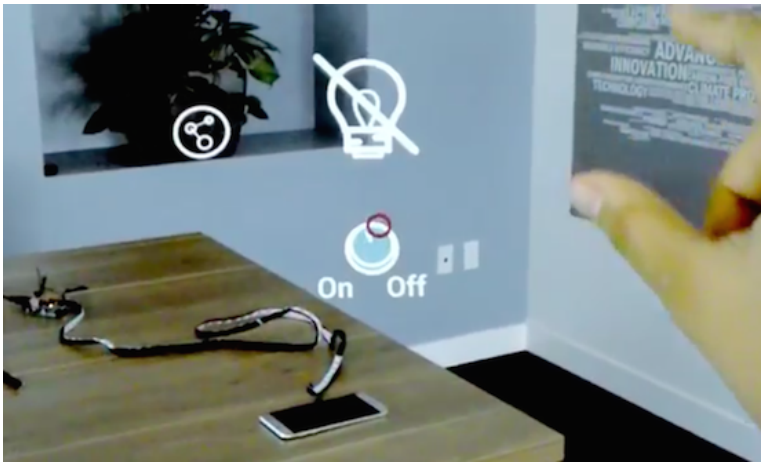
V práci [80] bol prezentovaný koncept *rozšírených vecí* (Augmented Things). Myšlienkou tejto práce je vytvorenie databázy digitálnych kópií reálnych objektov (typicky to môže byť spotrebná elektronika) a priradenie rôznych informácií k nim. Môže ísť napríklad o informácie o údržbe, návod na použitie a podobne. Po nasnímaní reálneho objektu, ktorého digitálna kópia je v databáze rozšírených vecí, prostredníctvom mobilného zariadenia sa na jeho obrazovke v rozšírenej realite zobrazia informácie o tomto reálnom objekte (obrázok č. 31).

Blízky zameraniu moderných HMI pre Industry 4.0 je koncept autora Phillipe Lewickeho [64]. Vytvoril demonštračnú aplikáciu, pomocou ktorej bolo možné ovládať inteligentnú žiarovku Phillips Hue pomocou headsetu Microsoft HoloLens. Pomocou HoloLensu bolo možné v rozšírenej realite vyberať jednoduchým gestom farbu svetla danej žiarovky. Autor si uvedomil, že dnešné riešenia umožňujú ovládať žiarovky prostredníctvom mobilnej aplikácie, v ktorej je následne potrebné nájsť konkrétnu miestnosť a žiarovku, ktorú chce ovládať. Opisovaný koncept však nebol ďalej rozpracovaný.

Na obrázku č. 32 je možné vidieť koncept dizajnéra Iana Sterlinga [85] a inžiniera Swaroopa Pala. Tento koncept demonštruje ovládanie inteligentných zariadení pomocou gest. Využitý je rovnako ako v predošlom prípade Microsoft HoloLens. Úlohou bolo poskytnúť užívateľské rozhranie aplikácií Android Music Player a mikrokontroléru Arduino so zapojeným ventilátorom so svetlom. Podobne ako v predošlom prípade nejde o ucelený systém, ale jednoúčelovú demonštračnú aplikáciu.

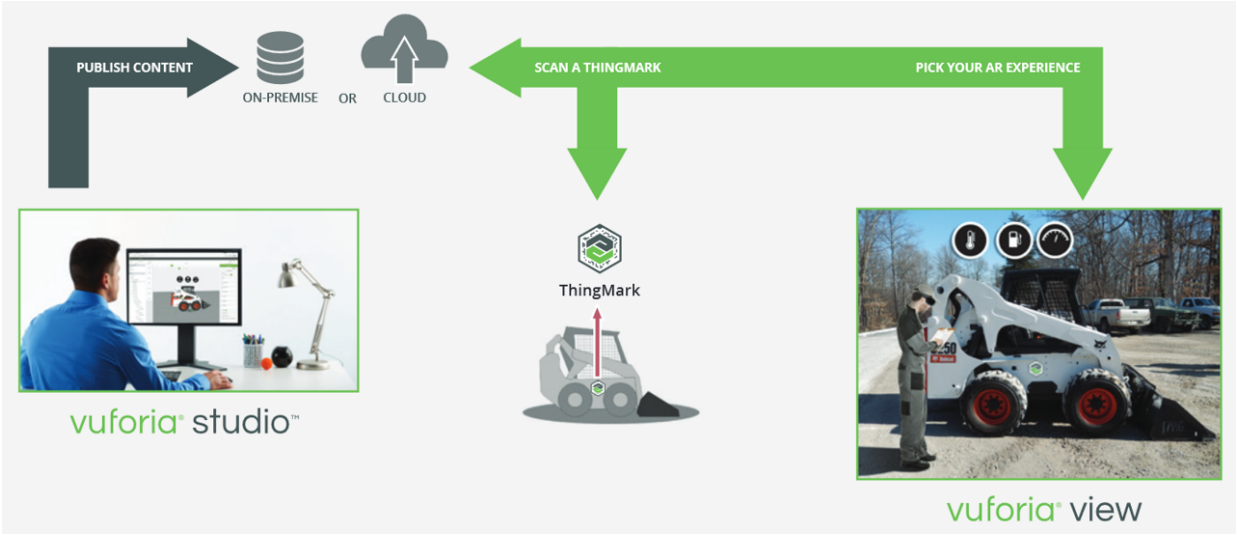


Obrázok č. 31: Doplňujúce informácie o produkte podľa konceptu *rozšírených vecí* [80]



Obrázok č. 32: Ovládanie mikrokontroléra Arduino pomocou Microsoft HoloLens [85]

Komplexným komerčným programovým systémom pre diagnostiku a ovládanie mechatronických systémov je Vuforia Studio [26], ktorý sa pôvodne nazýval ThingsWorx Studio. K premenovaniu prišlo po kúpe knižnice pre rozšírenú a zmiešanú realitu Vuforia technologickou spoločnosťou PTC. Takáto akvizícia bola logickým krokom, nakoľko spoločnosť PTC veľmi pružne reagovala na vznik konceptu Industry 4.0 a IIoT. Vuforia Studio využíva svoj uzavretý nástroj, kde je možné vkladať 3D a 2D objekty, ktoré sa zobrazia v rozšírenej realite po rozpoznaní daného zariadena. Táto technológia nerozpoznáva zariadenia priamo, ale pomocou vlastných 2D značiek *ThingMark*, ktoré sú vlastne konvenčnou technológiou podobnou QR kódom. Obsah sa následne vizualizuje pomocou aplikácie Vuforia View. Ilustráciu funkcionality možno vidieť na obrázku č. 33.

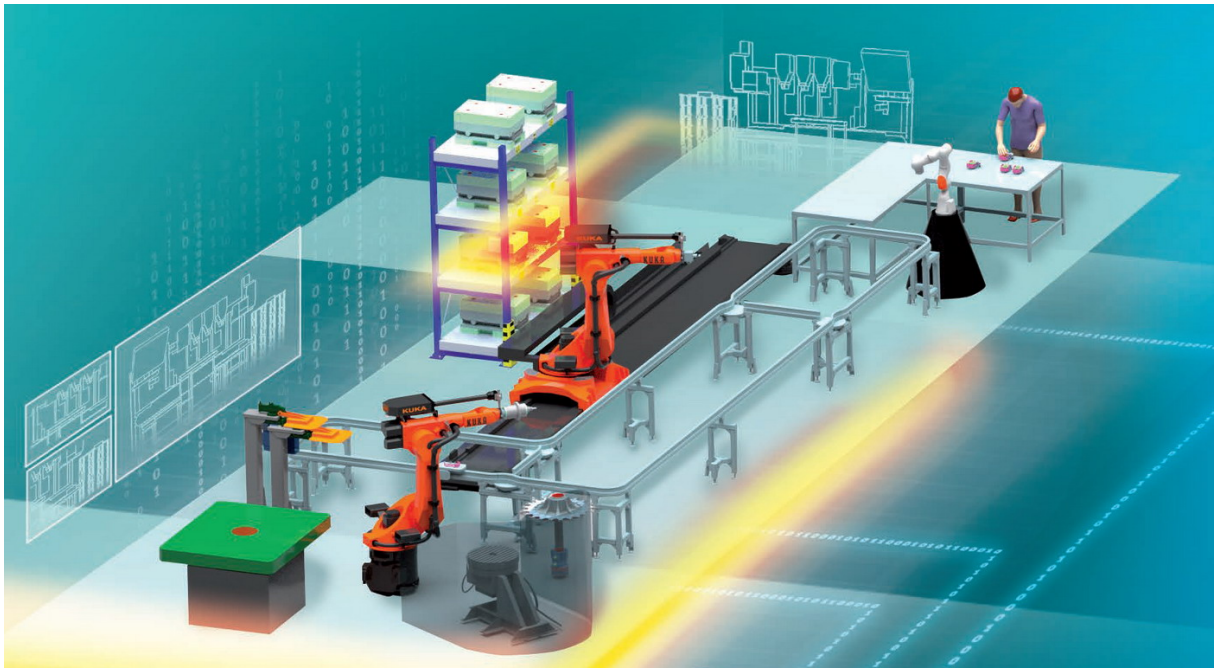


Obrázok č. 33: Ilustrácia funkcionality PTC Vuforia Studio [26]

Spoločnosť ŠKODA AUTO predstavila projekt Smart Maintenance, ktorý využíva rozšírenú realitu pre úlohy údržby. Využívaný je headset Microsoft HoloLens. Ide o pomerne jednoduchú softvérovú aplikáciu, ktorá pomocou kamier v zariadení rozoznáva kovovú trubice s úchytmi. Cieľom je kontrola vzdialeností úchytov, ktoré sa pravdepodobne časom môžu odchýliť. Technikovi sa v prípade detegovania trubice reálny objekt prekryje digitálnou trubicou, ktorá má úchyty na správnom mieste. Na základe vizuálnej informácie je možné jednoducho identifikovať prípadný posun a úchyt potom opraviť tak, aby sedel s pozíciou virtuálneho náprotivku. Takýto spôsob údržby uľahčuje a urýchľuje technikom prácu, nakoľko sú zbavení potreby neustáleho premeriavania vzdialenosti. Pre aplikáciu bol vyvinutý vlastný 3D engine. Po reálnom vyskúšaní aplikácie v rámci riešenia práce [63] je však možné konštatovať, že aplikácia zle reagovala na svetelné podmienky a taktiež zobrazenie utrpelo limitáciami headsetu HoloLens. Hologramy boli príliš bledé a správne nekopírovali objekty. Zorné pole bolo obmedzené. Reálne využitie prezentovaného riešenia je teda otázne.

3.5.4 Spolupráca a výmena informácií s priemyselnými partnermi

Pre zefektívnenie výskumu v predmetnej oblasti autor učebnice spolu s pracovníkmi Ústavu automobilovej mechatroniky FEI STU BA dlhodobo realizuje spoluprácu s odborníkmi pracovníkmi v Českej a Slovenskej republike s cieľom vzájomnej výmeny informácií a rozvíjania spolupráce [63].



Obrázok č. 34: Ilustrácia testbedu pre Industry 4.0 [95]

Zaujímavé riešenia ponúka tzv. testbed pre Industry 4.0 v Českom inštitúte informatiky, robotiky a kybernetiky ČVUT v Prahe (obrázok č. 34). Testbed je experimentálne vývojové technologické pracovisko, pričom v súčasnej dobe takéto pracoviská pre Industry 4.0 vznikajú aj na Slovensku. Testbed v Prahe ponúka ukážky automatizovanej a digitalizovanej výroby a možnosť otestovania nových riešení pre digitálne továrne, overovanie ich kompatibility, funkčnosti a efektívnosti. Ponúka tiež možnosti simulácie a optimalizácie výrob a ďalších vnútropodnikových procesov [95].

Pracovisko je tvorené niekoľkými zariadeniami, ktoré demonštrujú koncept Industry 4.0:

- dopravníkový pás;
- obrábací stroj;
- skrutkovacie rameno.

Každé z týchto zariadení má aj istú formu digitálnej reprezentácie, s ktorou komunikuje a pomocou ktorej riadi chod prevádzky.

Dopravníkový pás je vybavený niekoľkými robotickými ramenami, miestami pre montáž a priemyselnými vozíkmi, ktoré s výrobnými dielmi cestujú po dráhe. Virtuálna reprezentácia pásu dostáva dáta o tom, aké súčiastky sú naložené na vozíku, na ktorú z nich čaká ktorá montážna stanica a v akom stave je montovanie produktu. V momente, keď príde vozík na rázcestie na páse, odošle informáciu digitálnemu dvojčaťu, ktoré sa na základe dostupných dát rozhodne, ktorým smerom vozík odošle. Cez softvérovú aplikáciu na riadiacom počítači je možné zároveň nastaviť, aké komponenty sa budú montovať. Riešenie bolo demonštrované na montáži rôznofarebných modelov tatroviek, pričom bolo možné vyberať farebné kombinácie, ktoré výrobná linka vyprodukuje. V tomto prípade tzv. digitálne dvojča držalo aktuálny stav linky a riadilo výrobu formou smerovania vozíkov. Diagnostika a vizualizácia systému ešte neboli v danom čase implementované [63].

Ďalším dôležitým prvkom pracoviska je **obrábací stroj**. Ide o robotické rameno s frézou, ktorá obrába kovový materiál do požadovaného tvaru na základe vopred danej predlohy. Obrábací stroj je taktiež napojený na vzdialený systém s cieľom simulovať správanie sa obrábaného materiálu. Dáta z ramena sú odosielané do softvéru pre multifyzikálne simulácie Ansys, v ktorom bol vytvorený model celého procesu. Ide o zjednodušený model s tými časťami, ktoré sú pre simuláciu nevyhnutné. Zjednodušenie prebehlo z dôvodu potreby rýchlejších výpočtov. Matematický model rovníc bol ďalej redukovaný do nižších rádov, nakoľko sú dáta zbierané v reálnom čase a proces obrábania sa riadi aj podľa výstupov zo simulácie. Dôsledkom tohto prepojenia a simulácia boli dosiahnuté lepšie výsledky obrábania a obrobený predmet je parametricky lepší v prípade využitia simulačného modelu z Ansysu [63].

Posledným skúmaným zariadením je **skrutkovacie rameno**. Rameno má takisto digitálnu reprezentáciu zbierajúcu dáta o zariadení. K týmto dátam je možné vzdialene pristupovať napríklad aj formou mobilnej aplikácie v smartfóne alebo tablete, pričom na ňom bolo riešenie aj prezentované. Aplikácia je vytvorená pre technikov vo výrobe a jej účelom je pomôcť pri diagnostike stavu zariadenia a následnom riešení problému. Poskytuje manuál k zariadeniu a je možné využiť 3D vizualizáciu alebo zobrazenie v rozšírenej realite. Poškodené komponenty sú farebne odlíšené. Rozšírená realita využíva kameru tabletu. Tú je potrebné nasmerovať tak, aby zariadenie padlo presne do predlohy, ktorá je vyznačená v softvérovej aplikácii na tablete. Toto vizualizačné riešenie bolo vyvinuté v spolupráci s českým herným štúdiom, ktoré má v danej oblasti dlhoročné skúsenosti. Riešenie fungovalo plynule a vyzeralo byť využiteľné aj v praxi [56].

Ako už bolo uvedené, súčasný nastupujúci trend Internet of Things má dosah nielen v aplikáciách pre domácnosti, inteligentné budovy a v službách, ale aj signifikantný

vplyv na priemysel a výrobu. Aplikáciu princípov IoT v priemysle nazývame Industrial Internet of Things (IIoT). V tomto prípade v úlohe prepojených zariadení vystupujú jednotlivé strojové časti, senzory a aktuátory. Prepojenie zariadení by malo byť najmä bezdrôtové a malo by priniesť nové možnosti ich vzájomnej interakcie a tiež priniesť nové možnosti ich diagnostiky, ovládania a zabezpečenie pokročilých služieb.

Obrázok č. 35: Testovanie rozšírenej a zmiešanej reality pre údržbu rezacieho stroja [56]

V rámci riešenia projektov na pracovisku autora učebnice prebehlo viacero stretnutí a diskusií s priemyselnými partnermi, ktorí vyslovili požiadavky na prepojenie sietí Internet of Things s rozšírenou alebo zmiešanou realitou [83]. Je zrejmým faktom, že integrácia technológií rozšírenej a zmiešanej reality do výrobných procesov v digitálnych továrňach je neodvratná a je nutné sa tejto problematike venovať aj na univerzitnej pôde v rámci aplikovaného výskumu. Nižšie uvedené myšlienky získané počas spomínaných diskusií s priemyselnými partnermi uvádzame bez presných názvov jednotlivých spoločností pre zachovanie anonymity a prípadného priemyselného tajomstva.

1. **Britská spoločnosť zaoberajúca sa implementáciou princípov Industry 4.0**
   Jedným z moderných trendov v priemyselnej oblasti je využívanie čoraz výkonnejších, odolnejších a dostupnejších mobilných zariadení, ako je smartfón alebo tablet. Takéto zariadenia umožňujú využitie moderných technológií, ktoré sa doteraz v priemysle príliš nevyužívali. Máme na mysli práve rozšírenú a zmiešanú realitu a jej aplikácie pre ovládanie a monitorovanie požadovaných zariadení. Použitím

rozšírenej alebo zmiešanej reality vzniká kvalitatívne nový a lepší spôsob riešenia HMI. Pri konvenčných prístupoch je nevyhnutné si na zobrazovacom zariadení vybrať konkrétne zariadenie (snímač, aktuátor...), t. j. musíme vedieť jeho konkrétne umiestnenie vo výrobnej hale alebo jeho ID. Po zvolení zariadenia sú na zobrazovacej jednotke zobrazované požadované dáta (napríklad v tvare grafu alebo tabuľky). Pri využití aplikácie rozšírenej alebo zmiešanej reality je možné v rámci prostredia výrobnej haly operatívne vyhľadávať jednotlivé snímače a cez zobrazovaciu jednotku interaktívne zobraziť požadované hodnoty alebo meniť nastavenia a parametre daného zariadenia. Zaujímavou by bola pokročilá funkcionalita, ktorá by v prostredí umožňovala zobraziť, s ktorým zariadením je vybrané zariadenie prepojené, resp. si preposiela dáta. Práve lokalizácia a identifikácia jednotlivých snímačov a aktuátorov je v súčasnosti otvorený problém, ktorý je možné riešiť viacerými prístupmi [83].

2. **Slovenská spoločnosť zaoberajúca sa diagnostikou pneumatík**
   Využitím rozšírenej alebo zmiešanej reality v diagnostike rôznych zariadení je takisto otvorenou otázkou, ktorou sa zaoberá viacero spoločností. Počas stretnutí jeden z priemyselných partnerov formuloval požiadavku na systém diagnostiky chýb v pneumatikách prostredníctvom headsetu alebo mobilného zariadenia pre zmiešanú realitu. Tento systém by zároveň mal umožňovať zobrazovať diagnostické informácie z rôznych zariadení v továrni, čo je podobná požiadavka ako v predošlom bode [83].

3. **Slovenský výrobca pokročilých rezacích strojov**
   Využitie rozšírenej alebo zmiešanej reality na údržbu a ovládanie zložitých strojov a zariadení, ktorých rozloha dosahuje niekoľko desiatok metrov, je v súčasnosti takisto otvorenou tematikou, ktorá si vyžaduje komplexný multidisciplinárny prístup. Medzinárodní lídri v oblasti technológií delenia materiálov takéto riešenia už začali implementovať. Diskusia o možnostiach implementácie systémov údržby a diagnostiky s využitím počítačom generovanej reality preto prebehla aj so slovenskou spoločnosťou z tejto oblasti (obrázok č. 35).

4. **Ovládanie sofistikovaných priemyselných zariadení s obmedzeným prístupom**
   Ďalší z priemyselných partnerov videl využitie rozšírenej alebo zmiešanej reality v ovládaní a diagnostike rôznych sofistikovaných zariadení, ku ktorým ma prístup iba obmedzená skupina zamestnancov. Odpadá tak nutnosť realizácie fyzických ovládacích panelov, ku ktorým môže mať prístup aj bežný zamestnanec. Požiadavkou je, aby zariadenie mohol ovládať len zamestnanec, ktorý má prístup k mobilnému zariadeniu (smartfón/tablet) s aplikáciou rozšírenej alebo zmiešanej rea-

lity. Okrem bezpečnosti prináša takáto aplikácia aj výhody opísané v predošlých bodoch [83].

Záujem priemyselných partnerov o implementáciu rozšírenej alebo zmiešanej reality do priemyselných procesov je dôkazom, že predmetná problematika je moderná a v súlade s integráciou moderných digitálnych technológií do priemyslu (kybernetickofyzikálne systémy) pod taktovkou Industry 4.0.

3.6 Ďalšie technológie a softvérové nástroje využívané v učebnici

V tejto podkapitole budú v krátkosti charakterizované ďalšie technológie a softvérové aplikácie využívané v učebnici v rámci edukačných prípadových štúdií.

3.6.1 Komunikačný protokol MQTT

V systémoch Internet of Things sa často využíva **protokol MQTT** (**Message Queue Telemetry Transport**). Ide o sieťový, komunikačný protokol, ktorý využíva na výmenu dát medzi zariadeniami princíp *publish — subscribe*. Tento protokol je navrhnutý takým spôsobom, aby bol jednoduchý a nenáročný na implementáciu. Aj toto je dôvod, prečo je využívaný v IoT, kde jednotlivé *veci* môžu byť limitované výkonom a napájaním z batérie. Vlastnosťou MQTT je to, že jeden server (tzv. **broker**) môže obsluhovať väčšie množstvo klientov, ktorí so serverom komunikujú a sú s ním v spojení [56]. Vďaka tomu je možné komunikačný protokol MQTT využívať v obmedzených podmienkach a taktiež v sieťach, ktoré disponujú nižšou šírkou pásma, obmedzenými možnosťami spracovania, menšími kapacitami pamäte a vysokou latenciou. Samotný návrh protokolu MQTT minimalizuje požiadavky na šírku pásma siete a zabezpečuje spoľahlivosť doručenia pri posielaní dát medzi serverom (brokerom) a klientom [83].

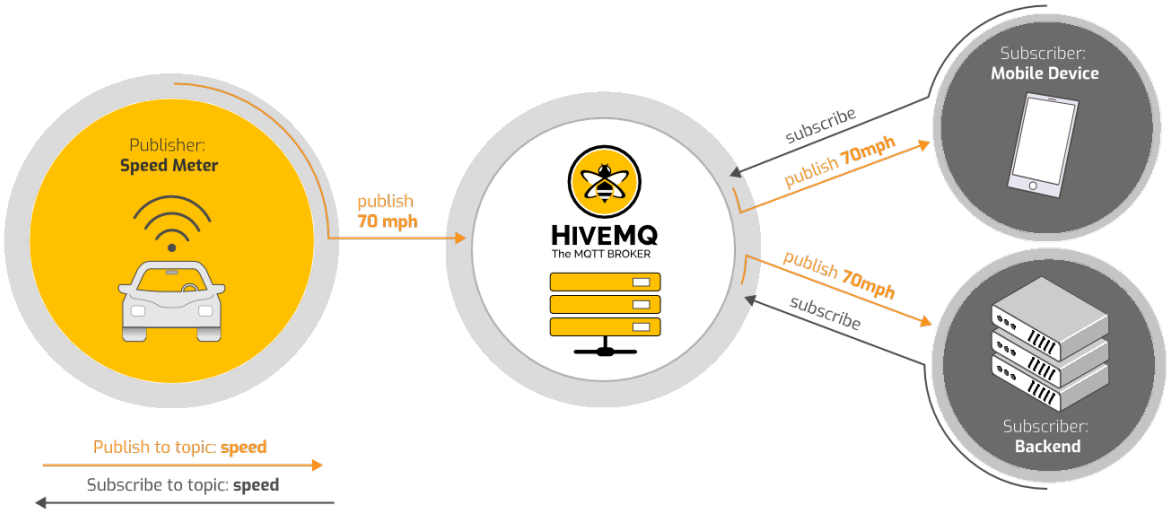
PRINCÍP KOMUNIKÁCIE

Protokol MQTT používa návrhový vzor *publisher-subscriber* na báze zasielania správ (*messages*), pričom tieto správy sú priradené témam (*topic*). Princíp komunikácia prostredníctvom protokolu MQTT je založený na publikovaní správy publisherom a odoberaní tém subscriberom. Publisheri a aj subscriberi sú pripojení k tzv. sprostredkovateľovi (*broker*), ktorý predstavuje rozhranie, ku ktorému je jednoduché sa pripojiť. Z tohto dôvodu je pridávanie ďalších subscriberov jednoduché [56]. Príklad je možné vidieť na obrázku č. 36.

Publisher (na obrázku č. 36 ide o automobil) odošle brokeru hodnotu. Ten rozošle túto hodnotu všetkým subscriberom, ktorí na ňu čakajú. Teda tým, ktorí sa prihlásili na odber tohto typu správ (na správy danej témy — topicu).

Všetky správy sú publikované v tzv. témach. Tieto témy nepotrebujeme vopred definovať, stačí správu rovno publikovať. Témy majú podobnú štruktúru ako súborové systémy. Jednotlivé dáta sú kategorizované (delené) s pomocou znaku `/`. Ak máme napríklad viacero počítačov v renderovacej farme, ktoré pôsobia ako publisher a zasielajú správy o teplote grafických kariet, tak hierarchia tejto témy môže vyzerať nasledovne:

```
sensor/COMPUTER_NAME/temperature/GRAPHICSCARD_NAME
```



Obrázok č. 36: Fungovanie komunikačného protokolu MQTT — príklad prevzatý z literatúry [40]

Odberatelia si dokážu vytvoriť odber (*subscription*) na konkrétnu tému. V tomto prípade budú dostávať iba správy s týmito dátami. Je však možné vytvoriť odber obsahujúci viacero informácií. Toto je možné nastaviť pomocou špeciálnych znakov [83]:

- Znak + sa používa pre získanie viacerých informácií z rovnakej úrovni. Ak teda napríklad chceme v predošlom príklade získať teplotu všetkých grafických kariet v zvolenom počítači, tak hierarchia odberateľa bude vyzerať takto:

  ```
  sensor/COMPUTER_NAME/temperature/+
  ```

  Ak chceme získavať takúto informáciu zo všetkých počítačov, ktoré publikujú do tejto témy, využijeme nasledovný zápis:

  ```
  sensor/+/temperature/+
  ```

  Vždy je však potrebné definovať celú tému. Myslím sa tým, že v našom príklade by reťazce ako `sensor/+/temperature` alebo `sensor/+` nefungovali.

- # sa využíva na označenie všetkých zostávajúcich úrovní hierarchie. Musí byť teda použitý výhradne na konci reťazca. Ak chceme z daného počítača z nášho príkladu

všetky údaje, ktoré poskytuje, čo nemusí byť len teplota grafickej karty, tak zápis bude vyzerať nasledovne:

```
sensor/COMPUTER_NAME/#
```

V tomto prípade je jedno, aké ďalšie informácie počítač poskytuje alebo koľko úrovní informácia má. Pomocou vyššie uvedeného zápisu budú vyžiadané všetky.

QUALITY OF SERVICE

Komunikačný protokol MQTT ponúka tri úrovne kvality služby QoS, ktoré definujú, ako veľmi sa broker a klient snažia, aby bola správa prijatá korektne. Úrovne QoS sú nasledovné [83]:

- **QoS 0:** broker / klient doručí správu raz a bez potvrdenia o doručení.
- **QoS 1:** broker / klient doručí správu aspoň raz a je potrebné potvrdenie o doručení.
- **QoS 2**: broker / klient doručí správu presne raz s využitím štvorkrokovej inicializácie komunikácie.

Publisherom má správa danú maximálnu úroveň QoS, ktorú si vie odberateľ vyžiadať. Znamená to, že ak je správa publikovaná na úrovni QoS 1, tak si ju odberatelia vedia vyžiadať na úrovni QoS 0 a QoS 1. Ak si ju odberateľ vyžiada na vyššej úrovni (napr. v našom prípade QoS), tak odovzdanie informácie prebehne, ale len na najvyššej publikovanej úrovni.

RETAINED MESSAGES

Každú zo správ je možné nastaviť takým spôsobom, aby po jej odoslaní všetkým odberateľom zostala v pamäti brokera. Toto sa deje pomocou tzv. *retain* statusu (angl. *flagu*). Ak príde nový klient, ktorý sa stane odberateľom danej témy, ktorá obsahuje túto správu, tak po vyžiadaní tejto témy bude poskytnutá aj uložená správa.

ZÁVETY

Keď sa klient pripojí k brokerovi, tak ho dokáže informovať, že má pripravenú tzv. *závet* (angl. *will*). Závet je správa, ktorá sa publikuje v tom prípade, keď sa daný klient nečakane odpojí. Závet má presne takú istú štruktúru ako ostatné (bežné) správy. Takisto má svoju *tému*, *retain status* a aj *QoS*. Niekedy môžeme nájsť aj termín *testament*.

MOSQUITTO

Mosquitto poskytuje klientske a serverové implementácie komunikačného protokolu MQTT. Môže byť implementovaný na zariadeniach, ktoré majú nízky výkon, medzi ktoré radíme aj mikrokontroléry pre Internet of Things. Mosquitto je možné využiť všade tam, kde je potrebné nenáročné zasielanie správ — najmä na zariadeniach s obmedzenými výpočtovými a pamäťovými prostriedkami.

Projekt Mosquitto je členom nadácie *Eclipse Foundation*. Pozostáva z troch častí [83]:

- hlavný server **mosquitto**;
- klientske nástroje **mosquitto_sub** a **mosquitto_pub** slúžiace ako jeden zo spôsobov pre komunikáciu so serverom MQTT;
- klientska knižnica napísaná v jazyku C / C++ umožňujúca využitie vo vlastných aplikáciách, resp. pre výskumné činnosti, ktoré súvisia s MQTT protokolom.

Mosquitto sa okrem akademickej obce používa aj v rôznych open source projektoch — napríklad openHAB pre domácu automatizáciu. Je tiež integrovaný aj do rôznych komerčných produktov.

Medzi ďalšie komunikačné protokoly a prostriedky využívané v IoT riešeniach patrí napríklad CoAP, AMQP, DDS alebo WebSockety.

CoAP (**Constrained Application Protocol**) je špecializovaný aplikačný protokol navrhnutý pre jednoduché zariadenia a sieťové prostredia. Je založený na komunikačnom modeli *request-response*, podobne ako HTTP, ale je optimalizovaný pre zariadenia s nízkym výkonom. Výhodou je podpora multicastu (dôležit8 pre aplikácie IoT a M2M), jednoduchá integrácia s webovými technológiami a asynchrónna komunikácia. Je menej rozšírený ako MQTT a môže byť menej efektívny v niektorých scenároch pre potrebu potvrdzovania správ.

AMQP (**Advanced Message Queuing Protocol**) je vysoko flexibilný a škálovateľný aplikačný protokol. Je zameraný na spoľahlivý a bezpečný prenos. Umožňuje komplexnú správu frontu správ. Nevýhodou je väčšia réžia v porovnaní s MQTT a nutnosť komplexnej administrácie.

DDS (**Data Distribution Service**) je štandard (niekedy tiež definovaný ako middleware) pre M2M komunikáciu, ktorý je zameraný na vysoký výkon, interoperability, reálny čas a škálovateľnosť. Využíva sa model *publish-subscribe*. Výhodou je vysoká priepustnosť, nízka latencia, robustná a flexibilná konfigurácia. Je však zložitejší na nastavenie a správu. Môže využívať viac zdrojov. Využíva sa napríklad v middlewari (meta operačnom systéme) pre robotické a mechatronické systémy Robot Operating System 2.

WebSocket poskytuje plnohodnotný dvojcestný komunikačný kanál cez TCP spojenie. Ide o vhodnú možnosť pre webové aplikácie, ktoré potrebujú rýchlu a interaktívnu komunikáciu. Výhodou je efektívna dvojcestná komunikácia, jednoduchá integrácia s webovými aplikáciami a podpora v moderných webových prehliadačoch. Nevýhodou z hľadiska IoT aplikácií je fakt, že nie je navrhnutý špecificky pre minimalizáciu réžie a môže byť náročnejší na implementáciu v niektorých IoT zariadeniach.

Každý z týchto protokolov a prostriedkov má svoje miesto v IoT a M2M ekosystéme a správnosť výberu závisí od konkrétnych potrieb projektu. Zatiaľ čo MQTT je vynikajúci pre jednoduché a efektívne správy v IoT scenároch, protokoly ako AMQP a DDS môžu ponúknuť výhody v spoľahlivosti a výkone pre podnikové a aplikácie reálneho času.

3.6.2 Node-RED

Node-RED je *flow-based* programovací nástroj, ktorý bol vyvinutý tímom IBM Emerging Technology Services. Momentálne je súčasťou nadácie OpenJS Foundation a ide o open-source nástroj.

FLOW-BASED PROGRAMOVANIE

Za autora myšlienky flow-based programovania sa považuje J. Paul Morrison, pričom jej pôvod sa datuje do 70-tych rokov minulého storočia. Je to spôsob opísania funkcionality alebo správania aplikácie ako siete čiernych skriniek (angl. *black box*) alebo uzlov (angl. *nodes*), ako sú nazývané aj v Node-RED. Každý z uzlov má jasne definovaný účel. Dostane nejaké dáta, spracuje ich a následne odovzdáva ďalej. Sieť zodpovedá za tok dát medzi uzlami [41].

Ide o model, ktorý je veľmi vhodný do vizuálnej reprezentácie a je prístupnejší širšiemu okruhu užívateľov. Ak sa niekto pozerá na takúto vizuálnu reprezentáciu programu, tak dokáže porozumieť celému toku bez toho, aby musel rozumieť všetkým riadkom kódu v každom z uzlov [83].

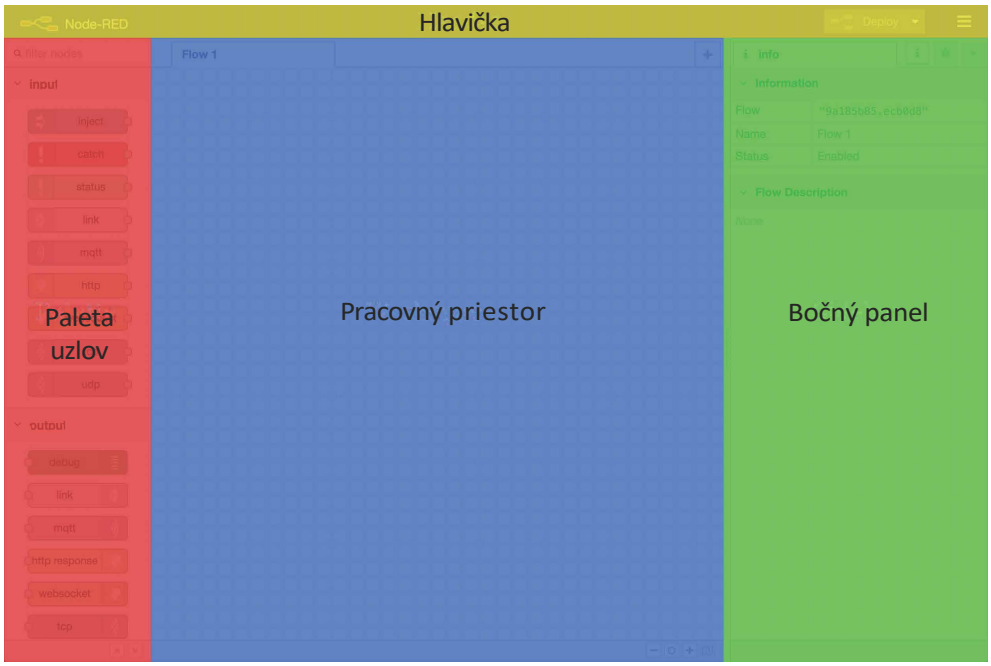
EDITOR NODE-RED

Node-RED pozostáva z runtime založenom na Node.js a vizuálneho editora. Tvorba programu prebieha v prehliadači pretiahnutím predpripravených uzlov z palety do pracovného priestoru. Potom je potrebné tieto uzly prepájať medzi sebou. Aplikácia sa následne nasadzuje do produkcie automaticky pomocou tlačidla *deploy*.

Paletu uzlov je možné jednoduchým spôsobom rozšíriť doinštalovaním nových uzlov, ktoré vytvára programátorská komunita. Toky, ktoré sú vytvorené v pracovnom priestore, je následne možné exportovať a zdieľať ako súbory JSON [41].

Runtime je postavený na **Node.js** (open-source a multiplatformový JavaScript run-

time engine) a plne využíva jeho udalosťami riadený, neblokujúci model. Preto je runtime Node-RED (resp. aplikácie vytvorené v Node-RED) ideálnym kandidátom na beh na okraji siete (edge computing) s použitím lacného hadvéru (napríklad mikropočítače Raspberry Pi) a takisto aj na beh v cloude. Vytvorené toky sa ukladajú v Node-RED vo formáte JSON, ktoré sú ľahko importovateľné a exportovateľné pre zdieľanie s ostatnými vývojármi [66].



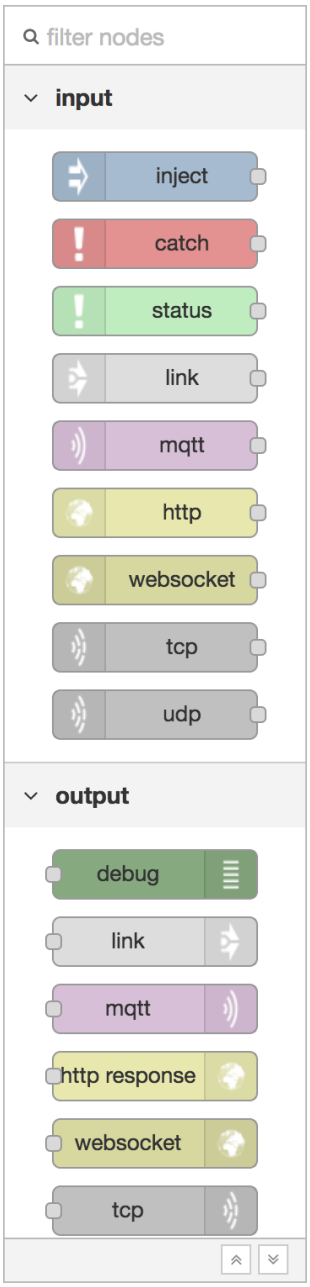
Obrázok č. 37: Okno editora Node-RED a jeho hlavné komponenty [42]

Na obrázku č. 37 je zobrazené prostredie editora Node-RED, ktorý sa skladá zo štyroch častí [83]:

- hlavička (angl. *header*)
- pracovný priestor (angl. *workspace*)
- paleta uzlov (angl. *palette*)
- bočný panel (angl. *sidebar*)

PALETA UZLOV

Paleta uzlov (obrázok č. 38) obsahuje všetky nainštalované uzly, ktoré sú ihneď dostupné na použitie. Sú usporiadané do viacerých kategórií. Ak existujú nejaké programové podtoky (*subflows*), tak sú zobrazované v kategórii hornej časti palety. Kategórie je možné zobraziť alebo skryť prostredníctvom kliknutia na ich záhlavie. Nižšie sú opísané uzly, ktoré sú základnými stavebnými blokmi pre tvorbu tokov [83].



Obrázok č. 38: Paleta uzlov [42]

Uzol typu **Inject** (obrázok č. 39) je možné použiť na manuálne spustenie toku. Manuálne spustenie toku je realizované kliknutím na tlačidla v editore Node-RED. Inject je takisto možné použiť na automatické spúšťanie tokov v pravidelných intervaloch. Odosielaná správa môže mať nastavenú svoju tému a hodnotu [56].

Obrázok č. 39: Uzol Inject [42]

Uzol typu **Debug** (obrázok č. 40) je možné použiť na zobrazenie správ v bočnom paneli v Node-RED editore. Tento panel poskytuje štruktúrované zobrazenie odosielaných správ, čo značne uľahčuje ich skúmanie. Každá správa obsahuje ID zdrojového uzla a časovú informáciu. Tlačidlo na uzle môže byť využité na zapnutie alebo vypnutie jeho výstupu. Tento uzol je takisto možné nakonfigurovať na posielanie všetkých správ do logu [83].

Obrázok č. 40: Uzol Debug [42]

Uzol typu **Function** (obrázok č. 41) slúži na definovanie vlastného kódu napísaného v jazyku JavaScript nad obsahom správy, ktoré prechádzajú cez tento uzol.

Obrázok č. 41: Uzol Function [42]

Uzol typu **Change** (obrázok č. 42) je možné využiť na zmenu vlastností správy a na nastavenie vlastností kontextu bez toho, aby bolo nutné využiť uzol typu Function. Každý uzol je možné nakonfigurovať s viacerými operáciami aplikovanými v danom poradí. Dostupnými operáciami sú [83]:

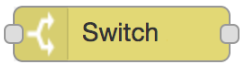
- Set — umožňuje nastaviť vlastnosť. Hodnota môže mať rôzny typ alebo môže byť prevzatá z existujúcej správy alebo vlastnosti kontextu.
- Move — umožňuje premenovať alebo presunúť vlastnosť.
- Change — umožňuje vyhľadávanie alebo nahrádzanie časti vlastností správy.
- Delete — umožňuje odstránenie vlastnosti.

Pri nastavovaní vlastnosti môže byť výsledná hodnota tiež výsledkom výrazu typu JSONata. JSONata je deklaratívny dopytovací a transformačný jazyk pre dáta typu JSON [56].

Obrázok č. 42: Uzol Change [42]

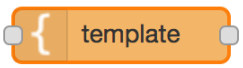
Uzol typu **Switch** (obrázok č. 43) slúži na smerovanie správ do rôznych vetiev toku prostredníctvom vyhodnotenia sady pravidiel. Existujú štyri typy takýchto pravidiel [83]:

- *Value rules* — sú vyhodnocované oproti nastavenej vlastnosti správy.
- *Sequence rules* — tieto pravidlá môžu byť využité v sekvenciách správ, ktoré sú generované napríklad pomocou uzla typu Split.
- Výraz JSONata — vyhodnocuje sa oproti celej správe. Platí, ak výraz vráti logickú hodnotu *true*.
- *Otherwise* — využije sa v prípadoch, ak neplatí žiadne z prechádzajúcich pravidiel.



Obrázok č. 43: Uzol Switch [42]

Uzol typu **Template** (obrázok č. 44) slúži na generovanie textu pomocou vlastností správy s využitím šablóny. Na generovanie sa využíva šablónovací jazyk Mustache.

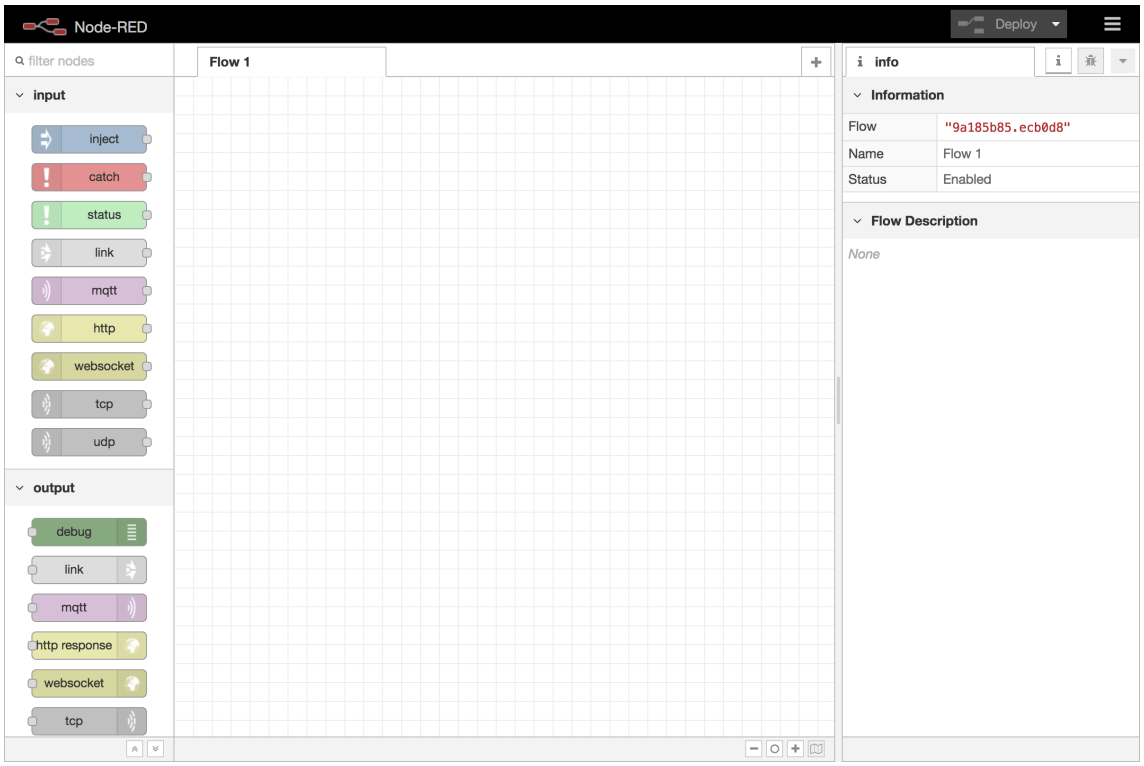


Obrázok č. 44: Uzol Template [42]

V Node-RED existuje aj nástroj *Palette Manager* slúžiaci na inštaláciu dodatočných uzlov z repozitára komunity.

PRACOVNÝ PRIESTOR NODE-RED

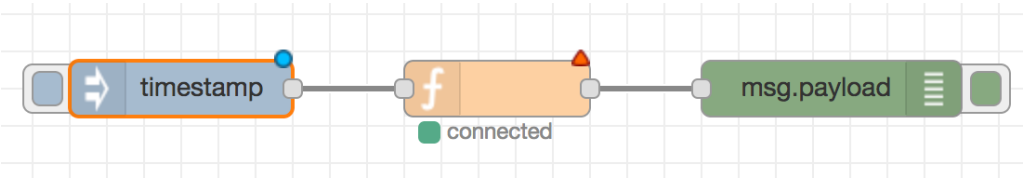
Hlavný pracovný priestor Node-RED (obrázok č. 45) je miestom, kde sa vyvíja samotný program pomocou ťahania uzlov z palety a ich následným prepájaním, čím vzniká programový tok. Tento pracovný priestor využíva karty v hornej časti. Podľa nich je možné si toky rozdeľovať kontextovo a získať lepší prehľad o programe.



Obrázok č. 45: Predvolený pracovný priestor editora Node-RED [42]

Ako už bolo uvedené, uzly do pracovného priestoru ťaháme z palety. Uzly sú navzájom prepájané cez ich porty. Uzol môže mať najviac jeden vstupný port, výstupných portov však môže byť mnoho. Port môže mať štítok, ktorý je zobrazený pri podržaní kurzora myši nad ním [42].

Niektoré typy uzlov (obrázok č. 46) zobrazujú stavovú správu a ikonu, ktorá je umiestnená pod uzlom. Toto vyjadruje stav uzla. Ak nastali v uzle nejaké zmeny, zobrazí sa nad ním modrý krúžok. Ak sú v jeho konfigurácii chyby, zobrazuje sa nad ním červený trojuholník. Niektoré typy uzlov obsahujú tlačidlo na pravom alebo ľavom okraji. Tieto tlačidlá umožňujú interakciu s uzlom v rámci editora. Uzly typu *Inject* a *Debug* sú jedinými vstavanými uzlami, ktoré majú takéto tlačidlá.



Obrázok č. 46: Stav uzla [42]

Existujú aj uzly s konfiguráciou. Sú to špeciálne typy uzlov, ktoré držia opakovateľne použiteľnú konfiguráciu, ktorú môžu zdieľať bežné uzly v toku. Napríklad uzly typu MQTT využívajú vstupné a výstupné uzly na konfiguráciu brokera MQTT pre repre-

zentáciu zdieľaného pripojenia. Konfiguračné uzly sa pridávajú prostredníctvom dialógového okna pre editovanie takého typu uzla, ktoré konfiguračný uzol vyžadujú [83]. Obsahuje výber z dostupných konfiguračných uzlov vyžadovaného typu alebo pridanie novej inštancie.

BOČNÝ PANEL NODE-RED

Tento panel obsahuje subpanely, ktoré užívateľovi poskytujú mnoho užitočných nástrojov [83]:

- informácie o uzloch a ich opis;
- zobrazenie správ posielaných do uzlov typu Debug;
- zobrazenie obsahu kontextu;
- správa konfiguračných uzlov.

Panely sú otvárané kliknutím na ich ikonu v hornej oblasti bočného panela alebo rozbaľovacom zozname. Postrannú lištu je možné zmeniť potiahnutím jej okraja cez pracovný priestor.

Pomocou Node-RED dokážeme vizuálne naprogramovať prepojenia medzi rozličnými vstupmi a výstupmi. Jedným z prípadov použitia môže byť napríklad situácia, keď vstupným uzlom je MQTT a výstupným uzlom je databáza (resp. zápis údajov do nej). Ak chceme vytvárať prepojenia s externými systémami, je najprv potrebné vytvoriť si na ne konekcie.

3.6.3 OpenPLC

OpenPLC je sada softvérových nástrojov, ktoré umožňujú **tvorbu a beh programov pre PLC** realizovaných podľa **normy IEC 61131-3**. Ide o ľahko využiteľné a open-source riešenie podporujúce všetkých päť programovacích jazykov danej normy [66].

OpenPLC pozostáva z **editora** a **runtimu**. Editor slúži na samotnú tvorbu PLC programu. Runtime je zodpovedný pre spustenie a beh PLC programu.

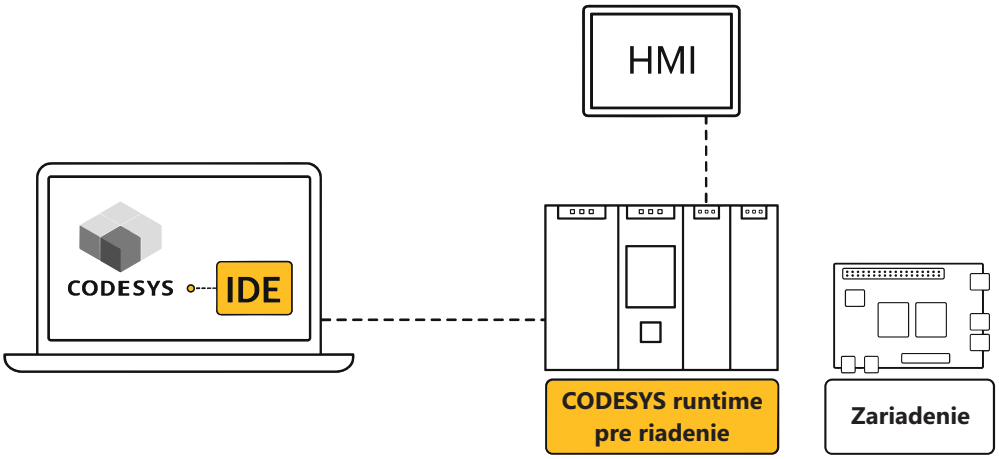
Nevýhodou OpenPLC je fakt, že aktuálne nepodporuje moderný komunikačný protokol OPC Unified Architecture, ale len protokol Modbus.

3.6.4 Codesys

Codesys (controller development system) je **automatizačný softvér** a ide o komplexný softvérový balík, ktorý je určený pre riadiace systémy. Jeho dôležitou vlastnosťou je, že je **nezávislý od výrobcov riadiaceho hardvéru**. Je podobne ako OpenPLC zložený z dvoch častí (obrázok č. 47) — integrovaného vývojového prostredia (IDE) a runtimu [66].

Codesys IDE je programovacie prostredie, ktoré je v súlade so štandardom IEC 61131-3, čo znamená, že na vývoj PLC programu ponúka 5 programovacích jazykov. Poskytuje všetky potrebné funkcie a nástroje na efektívne vytváranie PLC programov, vrátane rôznych knižníc, funkcií a modulov, ktoré uľahčujú proces programovania. Okrem toho je možné využiť rôzne typy simulátorov a debuggerov, ktoré umožňujú testovať a ladiť programy pred ich nasadením do reálneho riadiaceho systému [76].

Codesys Control runtime zabezpečuje beh PLC programov a HMI aplikácií vyvinutých pomocou Codesys IDE. Runtime sa zvyčajne nahrá do PLC, ale dá sa nahrať na akékoľvek zariadenie, ktoré je kompatibilné so štandardom IEC 61131-3. Výhodou teda je, že nemusíme vlastniť fyzické PLC zariadenie. Súčasť **Codesys Control Win** poskytuje virtuálne softPLC, ktoré beží na bežnom kancelárskom PC s operačným systémom Windows. Ďalej existuje napríklad *Codesys Control for Raspberry Pi*, ktorý sa dá nahrať do mikropočítačov Raspberry Pi alebo open-source PLC založených na tomto mikropočítači (napríklad séria Revolution Pi). Nevýhodou je, že runtime nie je úplne zadarmo. Bezplatne dokáže bežať len 2 hodiny, čo je pre edukačné účely plne postačujúce.



Obrázok č. 47: Codesys IDE a runtime [6]

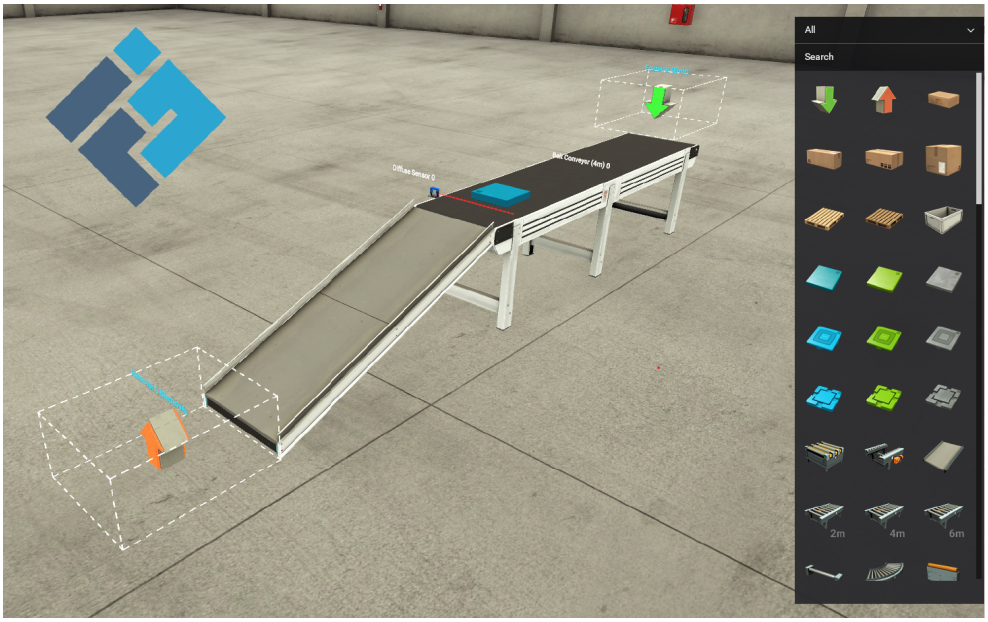
3.6.5 Factory I/O

Factory I/O je softvérový nástroj pre **návrh a simuláciu priemyselných procesov**. Umožňuje modelovanie a simuláciu virtuálnych 3D tovární (obrázok č. 48). Slúži ako nástroj pre výučbu a školenia, kde sa študenti a pracovníci môžu naučiť navrhovať riadiace algoritmy a programovať automatizačné systémy.

Tento softvérový nástroj poskytuje prostredie pre vytváranie virtuálnych výrobných liniek alebo komplexných tovární s rôznymi typmi zariadení, ako sú dopravníkové pásy, robotické ramená, senzory, motory a iné aktuátory. Zariadenia sa dajú konfigurovať, programovať a ovládať priamo pomocou PLC, pre čo slúžia rôzne ovládače (*I/O Drivers*)

[76]. Podporované sú rôzny typy PLC od výrobcov Allen-Bradley a Siemens. Ďalej je možná komunikácia pomocou Modbus TCP a OPC UA / DA, pričom v tomto prípade je možné využiť fyzické PLC od ľubovoľného výrobcu alebo softPLC. Je tiež možné si naprogramovať vlastný ovládač pomocou dostupného SDK.

Simulácia vo Factory I/O umožňuje testovanie a ladenie automatizačných procesov bez potreby fyzického hardvéru a zariadení. Dajú sa navrhnúť a overiť rôzne scenáre, skúmať interakcie medzi zariadeniami a vytvárať rôzne situácie v priemyselnom prostredí [76]. Dostupných je viac než 20 pripravených scén inšpirovaných typickými priemyselnými aplikáciami. Samozrejmosťou je možnosť prípravy vlastných komplexných scenárov.



Obrázok č. 48: Ukážka systému vo Factory I/O [98]

3.6.6 Multiplatformový vývoj mobilných aplikácií

Multiplatformový vývoj mobilných aplikácií je v dnešnej dobe čoraz populárnejší. V rámci tohto vývoja je pomocou jedného kódu možné aplikáciu nasadiť na rôzne operačné systémy (napríklad Android a iOS). Na opačnej strane stojí vývoj pre konkrétnu mobilnú platformu — pre konkrétny operačný systém, pričom takýto vývoj voláme ako **natívny** [76].

Medzi **výhody natívneho vývoja** možno zaradiť [30]:

- užívateľské rozhranie v danej aplikácií je vždy natívne a v súlade s grafickými zvyklosťami danej platformy;
- vyšší výkon aplikácie;

- prístup k všetkým hardvérovým a softvérovým zdrojom;
- lepšia kompatibilita.

Medzi **nevýhody natívneho vývoja** patrí:

- špecifická verzia danej aplikácie musí byť vyvinutá pre každú platformu (operačný systém);
- pre vývoj na konkrétnu platformu je nutné ovládať väčšinou špecifický programovací jazyk odlišný od inej platformy (napr. pre iOS je to Swift a pre Android Kotlin/Java);
- súhrnne je vývoj na rôzne platformy dlhší a drahší.

Samozrejme, multiplatformový vývoj nie je samospasný a má svoje plusy a mínusy. Medzi **výhody multiplatformového vývoja** možno zaradiť:

- jednotný vývoj pre viaceré platformy;
- nie je nutné poznať viaceré programovacie jazyky pre nasadenie aplikácie na rôzne platformy (operačné systémy);
- kratší vývojový čas a nižšie náklady.

Medzi **nevýhody multiplatformového vývoja** patrí:

- často limitovaná používateľská skúsenosť (angl. *user experience*);
- obmedzený výkon aplikácie;
- obmedzený prístup k hardvérovým a softvérovým zdrojom;
- problémy s kompatibilitou medzi jednotlivými zariadeniami.

Multiplatformový vývoj, resp. knižnice na vývoj multiplatformových aplikácií, je možné rozdeliť podľa rôznych kritérií. Jedným z nich je rozdelenie na **kroskompilované** (angl. *cross-compiled*) a **hybridné** aplikácie [30]. Existujú tiež tzv. **progresívne webové aplikácie** (PWA) [25], ktorým sa bližšie v tejto učebnici nevenujeme.

Kroskompilované aplikácie (niekedy tiež angl. *native-like* aplikácie) sú napísané v jednom programovacom jazyku a následne v čase kompilácie sa prekladajú do jazykov daných pre špecifickú platformu. Toto umožňuje programátorom naprogramovať aplikáciu v jednom jazyku a napokon vytvoriť plne natívnu aplikáciu pre vybrané platformy [99]. Medzi najznámejšie knižnice/nástroje takéhoto vývoja patrí **.NET Multiplatform App UI** (.NET MAUI — jazyk C#), **Flutter** (jazyk Dart), **RubyMotion** (jazyk Ruby) alebo **React Native** (jazyk JavaScript).

Hybridné mobilné aplikácie sú postavené na webových technológiách, ako je HTML, CSS a JavaScript. Namiesto toho, aby boli spúšťané v mobilnom prehliadači, tak sú spúšťané v natívnom kontajneri, čo im umožňuje prístup k hardvérovým zdrojom daného zariadenia. V dnešnej dobe je najrozšírenejší framework *Apache Cordova*, na ktorom je postavených viacero ďalších frameworkov. Nevýhodou aplikácií vyvinutých pod Apache Cordova a odvodených nástrojov je to, že sú pomalšie z hľadiska výkonu ako natívne [99]. Medzi populárne frameworky využívajúce Apache Cordova patrí napríklad **Ionic** alebo **Framework7**.

3.6.7 Angular a Ionic

Angular je open-source framework postavený na jazyku JavaScript vyvinutý spoločnosťou Google. Je to jeden z najznámejších frameworkov pre vývoj webových aplikácií. Ako programovací jazyk sa využíva TypeScript (nadstavba jazyka JavaScript od spoločnosti Microsoft prinášajúca najmä statické typovanie). Kód napísaný v TypeScripte sa automaticky kompiluje na JavaScript, ktorý následne beží kdekoľvek, kde beží aj JavaScript [76].

Angular poskytuje robustnú štruktúru, ktorá pomáha pri organizácii zdrojového kódu, a ponúka množstvo nástrojov pre rýchle a efektívne vytváranie interaktívnych používateľských rozhraní. Takisto zahŕňa funkcie, ako je správa stavu aplikácie, routovanie, validácia formulárov, integrácia s externými knižnicami a pod. Mnoho funkcionalít v Angulari, ako napríklad správa formulárov, HTTP volania, animácie a routovanie sú implementované pomocou **RxJS**. RxJS je funkcionálna a reaktívna programovacia knižnica, ktorá poskytuje možnosti práce s asynchrónnymi udalosťami a tokmi dát. Je založená na konceptoch pozorovateľ (*observable*), správca (*operator*) a odberateľ (*subscriber*), ktoré umožňujú deklaratívne manipulovať s udalosťami a dátami [76].

Ionic je knižnicou pre vývoj hybridných multiplatformových aplikácií a súpravou nástrojov mobilného UI s otvoreným zdrojovým kódom. Skonštruovaný je tak, aby bol rýchly, napriek tomu, že ide o hybridný framework. Aplikácie vyvinuté v Ionicu môžu byť nasadené na mobilných platformách Android a iOS. Mobilné aplikácie je možné vyvíjať pomocou JavaScript frameworkov **React**, **Angular** alebo **Vue.js**. Ionic umožňuje prístup k širokej škále pluginov, ktoré umožňujú prístup k natívnym funkciám na mobilných platformách, ako sú fotoaparát, geolokácia, notifikácie a ďalšie. Taktiež poskytuje rozhranie pre pridanie vlastných pluginov a rozšírení, čo umožňuje prispôsobenie a rozšírenie funkcionalít aplikácie.

V edukačnej prípadovej štúdii v tejto učebnici sa využíva Ionic v kombinácii s Angularom.

Obrázok č. 49: Architektúra Ionic aplikácie [76]

3.6.8 3D engine Unity

Pod **3D enginom** alebo **herným enginom** chápeme softvérový framework navrhnutý pre tvorbu a vývoj videohier alebo iných interaktívnych aplikácií. Medzi funkcie, ktoré ponúka 3D engine, patrí renderovací engine pre 2D/3D objekty, fyzikálny engine s detekciou kolízií, zvuk, možnosť skriptovania, animácia, umelá inteligencia, komunikácia cez sieť, streamovanie, manažment pamäte a vlákien, podpora lokalizácie a podobne [59].

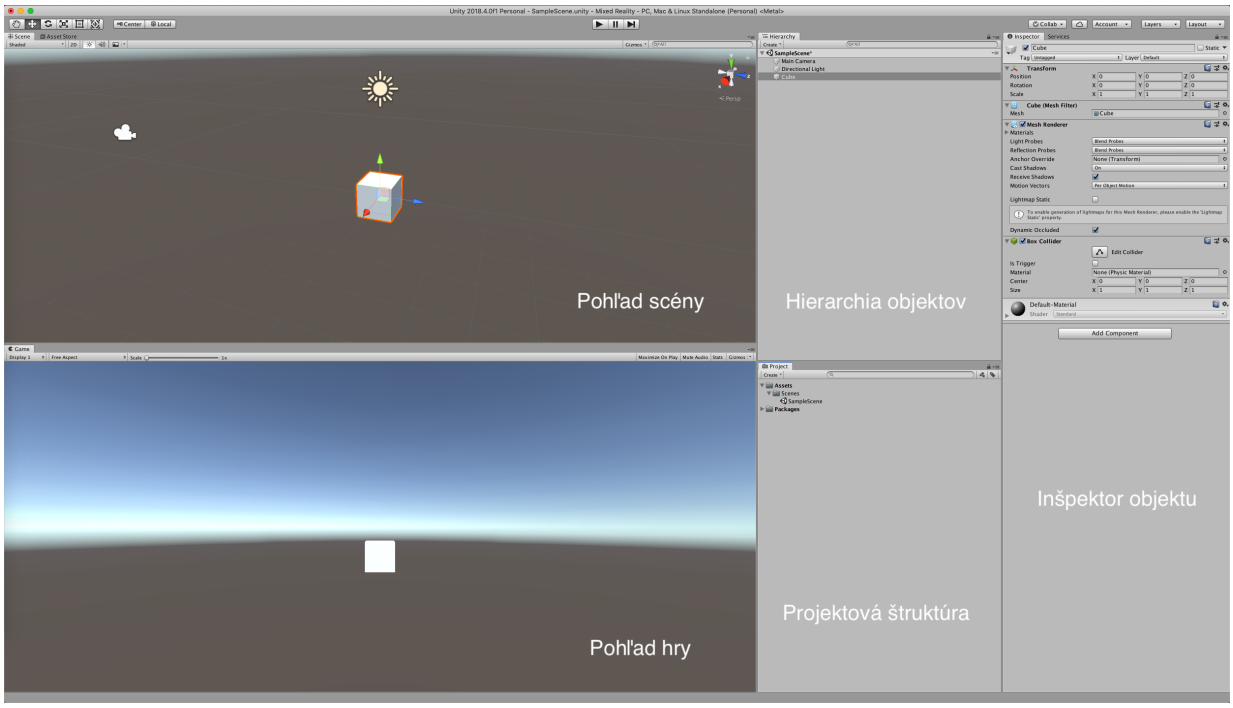
Unity je **multiplatformový 3D engine**, ktorý bol pôvodne určený len pre tvorbu 2D a 3D hier, ale jeho pôsobenie sa rozšírilo do rôznych oblastí, ako sú napríklad produktové 3D vizualizácie a animácie, animované filmy, virtuálna filmová produkcia, architektúra a v neposlednom rade aj priemyselné aplikácie. Podporuje viac než 20 platforiem, kde môže bežať výstupná aplikácia. Engine podporuje skriptovacie API v jazyku C#, vizuálne skriptovanie, tvorbu pluginov, ale aj samostatný vývoj aplikácie [83].

Na domovskej webovej stránke je možné stiahnuť tento editor zadarmo za istých podmienok pre Windows, Linux aj macOS.

PROSTREDIE EDITORA

Na úvodnej obrazovke editora (obrázok č. 50) je možné vidieť rozloženie prostredia na viacero panelov — pohľad na scénu (angl. *Scene View*), pohľad na hru (angl. *Game View*), projektová štruktúra, hierarchia objektov a inšpektor (detaily) týchto objektov. Týchto panelov si je možné zobraziť viac, čo závisí od toho, čomu je potrebné sa aktuálne veno-

vať a či je daný panel potrebný.



Obrázok č. 50: Rozhranie editora Unity [83]

POHĽAD SCÉNY

Pohľad scény (*Scene View*) je interaktívne zobrazenie, kde sa tvorí rozhranie a prostredie samotnej aplikácie. Využíva sa na výber a nastavenie pozície scény, objektov, kamier, svetiel a ostatných typov herných objektov.

POHĽAD HRY

Pohľad hry (*Game View*) vykresľuje pohľad z nastavenej kamery v aplikácii. Ide o reprezentáciu toho, ako bude vyzerať finálny build aplikácie. Je teda potrebné mať v scéne aspoň jednu kameru.

PROJEKTOVÁ ŠTRUKTÚRA

V tomto zobrazení je možné pristúpiť k súborov projektov a ich manažovať. Nachádzajú sa tu rôzne typy súborov, ako sú materiály, pluginy, prefabrikáty, scény, skripty alebo akékoľvek iné súbory, ktoré sa majú používať v projekte.

HIERARCHIA OBJEKTOV

Okno hierarchie obsahuje zoznam objektov typu GameObject v aktuálne otvorenej scéne. Niektoré z nich môžu byť inštanciami vytvorenými zo súborov z projektovej štruktúry

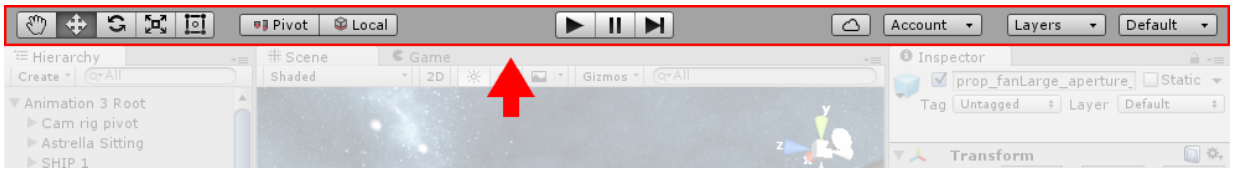
a iné zasa prefabrikáty, čo sú vlastné objekty, ktoré tvoria väčšinu scény. Keď sú objekty pridávané alebo mazané zo scény, tak sa rovnako pridajú alebo zmažu aj z okna hierarchie.

INŠPEKTOR OBJEKTU

Projekty v Unity editore sú tvorené viacerými objektmi typu GameObject, ktoré obsahujú zvuky, skripty, vrstvy a iné grafické elementy — napríklad svetlá. Okno inšpektora objektu slúži teda na zobrazenie podobných informácií o aktuálne vybranom objekte vrátane všetkých priradených komponentov a ich vlastností. Tieto vlastnosti je možné aj editovať.

PANEL S NÁSTROJMI

Panel s nástrojmi pozostáva zo siedmych základných nastavení (obrázok č. 51). Nájdeme tu napríklad nástroje na transformovanie scény a zmeny pohľadu, Gizmo nástroj, nástroje na spustenie, pozastavenie a zastavenie behu aplikácie a podobne [83].



Obrázok č. 51: Panel s nástrojmi v rámci editora Unity [83]

Unity obsahuje aj framework pre multiplatformový vývoj rozšírenej a zmiešanej reality s názvom **AR Foundation** [15]. Tento framework predstavuje rozhranie pre vývojárov Unity, ktoré môžu používať, ale AR Foundation neimplementuje žiadne funkcie rozšírenej alebo zmiešanej reality. Ak chceme používať AR Foundation na cieľovom zariadení, potrebujeme aj samostatné balíky pre cieľové platformy oficiálne podporované systémom Unity. Ide o **ARCore XR Plugin** pre Android, **ARKit XR Plugin** na iOS, **Magic Leap XR Plugin** pre Magic Leap a **Windows XR Plugin** pre Microsoft HoloLens [69]. V jednej z edukačných prípadových štúdii v tejto učebnici je využitý ARCore XR Plugin.

Medzi ďalšie známe 3D enginy patrí napríklad Unreal Engine alebo CryEngine. Unreal Engine ponúka veľmi vysokú grafickú kvalitu a robustné prostredie pre tvorbu 2D a 3D aplikácií. Je však náročnejší na naučenie ako Unity a takisto omnoho náročnejší na hardvér. Ako plne open-source alternatíva je v ponuke Godot Engine, ktorý sa aktuálne rýchlo rozvíja a získava obľubu u začínajúcich herných vývojárov.

4 Edukačné prípadové štúdie

Dnešný moderný priemysel si vyžaduje odborníkov z mnohých akademických a praktických oblasti. Od vzdelávacích inštitúcií (najmä vysokých škôl a univerzít) sa od praxe očakáva, že začlenia nové trendy a metódy konceptu Industry 4.0 do súčasných učebných osnov. Takto by sa zabezpečilo, aby budúci absolventi neboli prekvapení meniacimi sa očakávaniami v odvetví. Kyberneticko-fyzikálne systémy sú jedným z činiteľov zmien vo vzdelávaní moderných digitálnych inžinierov a technických odborníkov.

Vzdelávanie v oblasti automatizácie (operačné technológie — OT) a v oblasti informatiky (informačné technológie — IT) sa až do nedávnej doby realizovalo a často ešte aj v súčasnosti realizuje striktne oddelene a v rôznych študijných programoch. Konvergencia IT a OT, ktorá je opísaná v predošlých častiach učebnice, však prináša nutnosť nových pohľadov a prístupov k tvorbe učebných osnov. V dnešnej dobe je nevyhnutné vytvárať multidisciplinárne študijné programy a synergicky kombinovať vedomosti z oblasti informačno-komunikačných technológií, automatického riadenia a mechatroniky.

Transformácia dát do digitálnej podoby, ktorú priniesol koncept Industry 4.0, bola jedným z hlavných faktorov, ktoré umožnili vznik konceptu Kvalita 4.0. Použitie tohto konceptu znižuje riziko chýb a odstraňuje bariéry interoperability, spolupráce. Medzi základné nástroje konceptu Kvalita 4.0 patrí správa a analýza veľkých objemov dát (Big Data), využívanie moderných digitálnych technológií (Internet of Things, cloud computing, počítačom generované realita...), ako aj aplikácie počítačového a strojového videnia s podporou hlbokého učenia.

Na výučbu interdisciplinárnych vedomostí v študijných programoch zameraných na digitálne inžinierstvo je potrebné aktualizovať učebné osnovy. Do vyučovacieho procesu je nutné začleniť komplexné moduly, ako aj komponenty vychádzajúce z praxe. Vzdelávanie inžinierov pre Industry 4.0 musí byť zamerané na zručnosti vedúce k digitalizácii vo výrobnom sektore. Koncept Industry 4.0 si vyžaduje nové, multifunkčné profesie, v ktorých si títo noví odborníci budú musieť rozšíriť svoje chápanie informačných technológií a zaužívaných postupov. Podľa správy „High-tech skills and leadership for Europe“ bol v roku 2020 v Európe deficit 500 000 expertov na IT [16].

Švédska iniciatíva Ingenjör4.0 predstavila pozoruhodný vzdelávací koncept [44]. Ide o jedinečný program nadobúdania zručností, ktorý je založený na webových moduloch vyvinutých v spolupráci viacerých švédskych univerzít. Moduly možno kombinovať podľa potrieb účastníkov vzdelávacieho kurzu, čo uľahčuje prispôsobenie kvalifikácie na základe konkrétnych potrieb danej spoločnosti, študijnej skupiny a jednotlivca. Ingenjör4.0 umožňuje profesionálom v danom odvetví jedinečné, inovatívne, obsiahle a celoživotné vzdelávanie. Nakoľko cyklus morálnej životnosti využívaných technológií

je čím ďalej kratší, každý inžinier (a to nielen vo vývoji, ale aj vo výrobe) by mal byť počas svojho pracovného produktívneho života opakovane preškoľovaný. Neodmysliteľnú časť rýchlo sa meniacich technológií tvoria práve technológie, ktoré súvisia s digitalizáciou výrobkov, výroby a ďalších súvisiacich procesov (servis, logistika, predaj atď.).

Paradoxné je, že stále existuje veľké množstvo manažérov, ktorí neveria v deklarovaný prínos digitálneho podniku. Jedna z príčin pochybností a neporozumenie vnímania digitálnych podnikov spočíva v nedostatku kvalitných realizátorov požiadaviek na našom trhu a tiež nízke povedomie odbornej verejnosti — najmä zo strany slovenských spoločností.

Táto učebnica sa preto v nasledujúcej kapitole zaoberá opisom **prípadových štúdií využiteľných na výučbu expertov pre oblasť digitalizácie** výrobných procesov s dôrazom na moderné formy komunikácie vo výrobných systémoch (od úrovni zariadení až po úroveň cloudu) s využitím virtuálnych modelov liniek, softPLC a voľné dostupných platforiem pre IIoT (napríklad Node-RED).

Nasledujúca kapitola sa venuje problematike modelovania, simulácie a riadenia diskrétnych udalostných systémov. Riadený udalostný systém v praxi pozostáva napríklad zo série výrobných liniek. V univerzitnom prostredí pre účely vzdelávania sa využívajú rozličné zmenšené modely (napr. od firmy Fischertechnik [23] alebo LEGO [73]). Zaujímavou alternatívou sú tiež virtuálne modely udalostných systémov, ktoré je možné zostaviť napríklad pomocou Factory I/O, pričom táto softvérová aplikácia je využívaná v predloženej učebnici.

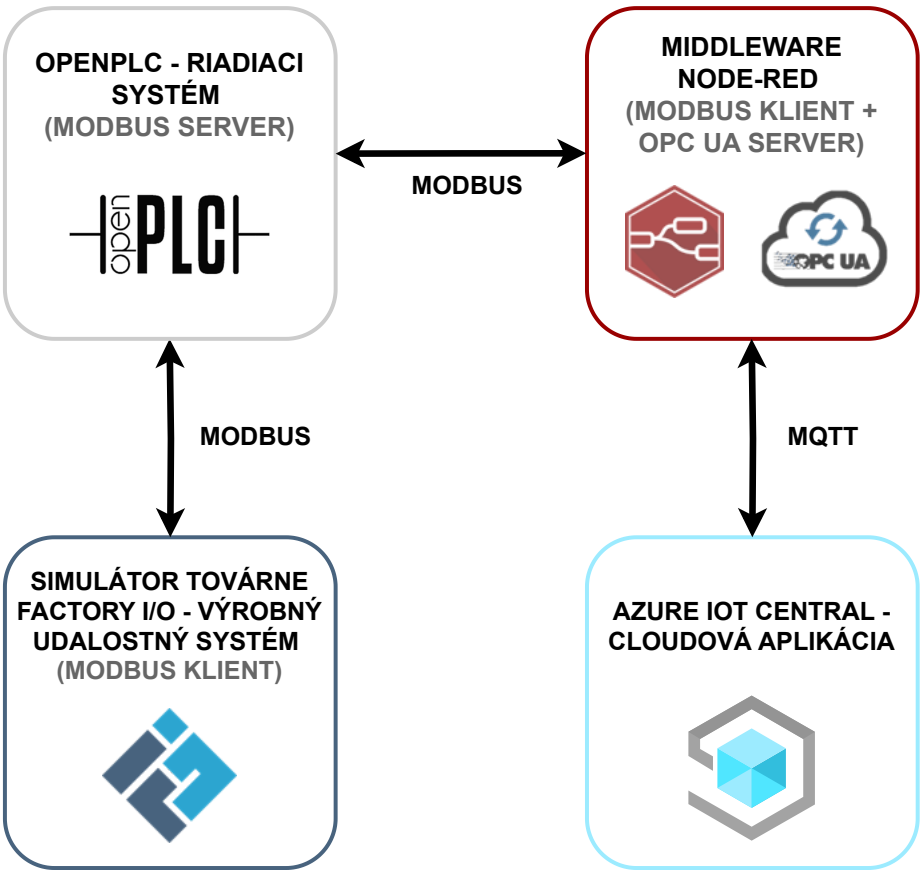
Využitie virtuálneho, teda simulovaného, udalostného systému prináša so sebou viacero výhod. Nakoľko opísané prípadové štúdie by mali byť využiteľné pre oblasť vzdelávania v odbore inžinierskej a aplikovanej informatiky, je potrebné zabezpečiť, aby tieto materiály boli pre záujemcov ľahko dostupné. Virtuálny model systému nie je potrebné kupovať, servisovať a tiež nezaberá žiadny fyzický priestor. Je ho tým pádom možné ponúknuť širšej palete záujemcov o vzdelávanie.

Na to, aby bolo možné systém riadiť, je potrebný riadiaci systém. V prípade diskrétnych udalostných systémov, čo výrobné systémy vo svojej podstate sú, sa používajú programovateľné logické kontroléry (PLC). Tieto predstavujú malé priemyselné počítače obsahujúce vstupno-výstupné piny. Do nich je možné zapojiť príslušné senzory (snímače) alebo aktuátory (napr. riadené pohony a pod.). Chod PLC je možné simulovať aj na počítači (softPLC), a teda pre beh uvedených edukačných prípadových štúdií nie je nevyhnutné vlastniť hardvérové PLC, čo je pre vzdelávacie účely výhodné. Chod PLC realizuje PLC runtime engine, čo je väčšinou štandardná súčasť editorov PLC programov, ako sú napríklad TIA Portal od spoločnosti Siemens, Codesys alebo OpenPLC.

PLC runtime a Factory I/O je možné spojiť pomocou vhodného komunikačného protokolu.

Opísané edukačné prípadové štúdie boli vytvorené na Fakulte elektrotechniky a informatiky Slovenskej technickej univerzity v Bratislave na Ústave automobilovej mechatroniky najmä v rámci tematických diplomových prác v študijných odboroch kybernetika a informatika [66], [98], [76] a [69].

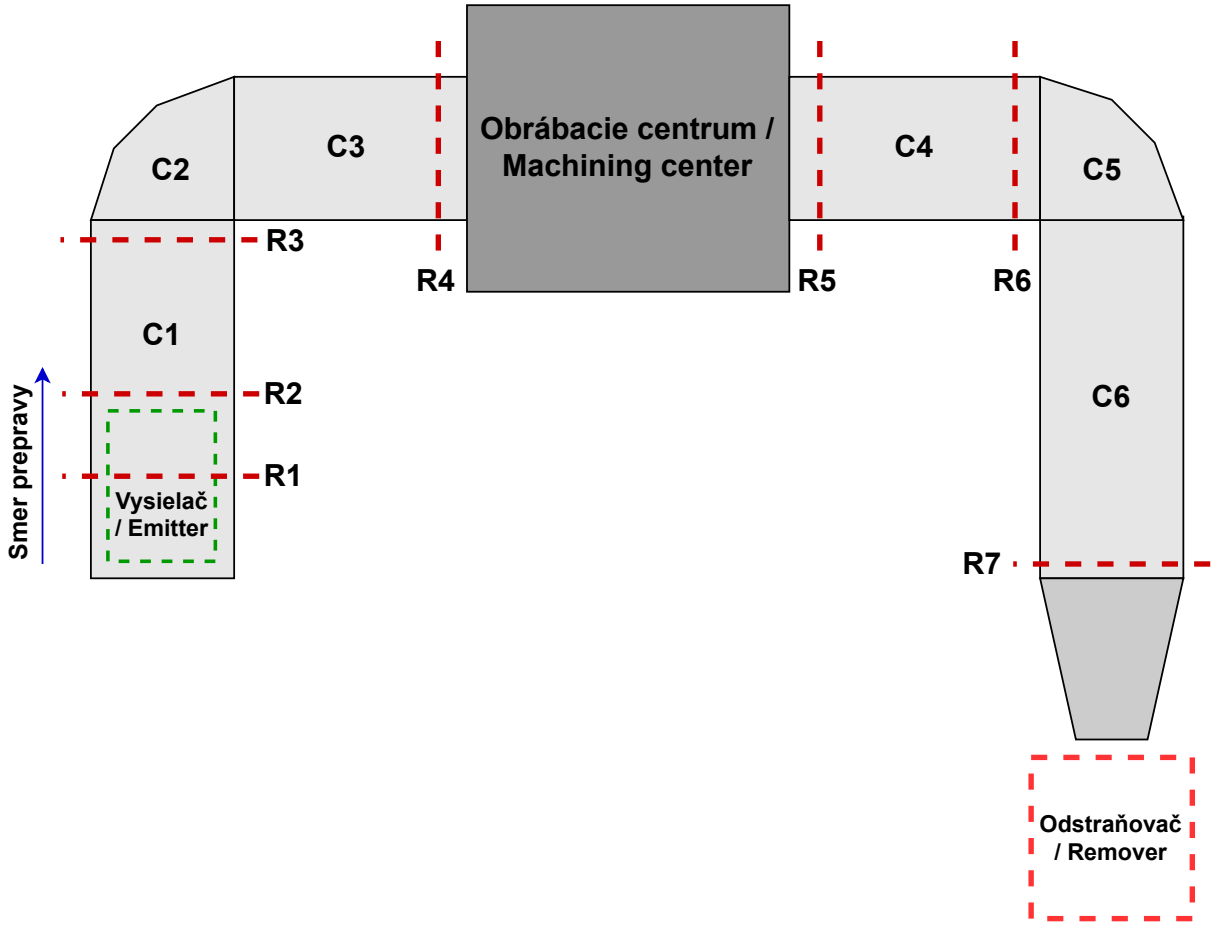
4.1 Prvá prípadová štúdia: Prepojenie PLC s cloudovou aplikáciou Microsoft Azure IoT Central



Obrázok č. 52: OpenPLC v prepojení s Node-RED a Microsoft Azure

V rámci prvej prípadovej štúdie (obrázok č. 52) bolo cieľom namiesto platených PLC editorov a runtime demonštrovať využitie voľne dostupného open-source softvérového systému OpenPLC. OpenPLC nepodporuje moderný komunikačný štandard OPC UA, ale len konvenčný protokol Modbus. OpenPLC síce dokáže komunikovať s Factory I/O pomocou tohto konvenčného protokolu a realizovať pomocou neho riadiace procesy. Cieľom prípadovej štúdie je však dáta posielať aj do cloudového prostredia a prípadne tieto dáta poskytnúť ďalším klientom cez OPC UA. Preto je nutné využiť prostredník (middleware) Node-RED, ktorý nie je potrebný k riadeniu, ale je využitý pre možnosť zdieľania dát do cloudu cez protokol MQTT a tiež cez protokol OPC UA pre server. K OPC UA serveru je možné pripojiť širokú paletu klientov. V tomto konkrétnom prípade budeme testovať OPC UA klienta UAExpert. Táto prípadová štúdia je opísaná s využitím diplomovej práce [66].

4.1.1 Špecifikácia a správanie diskrétneho udalostného systému

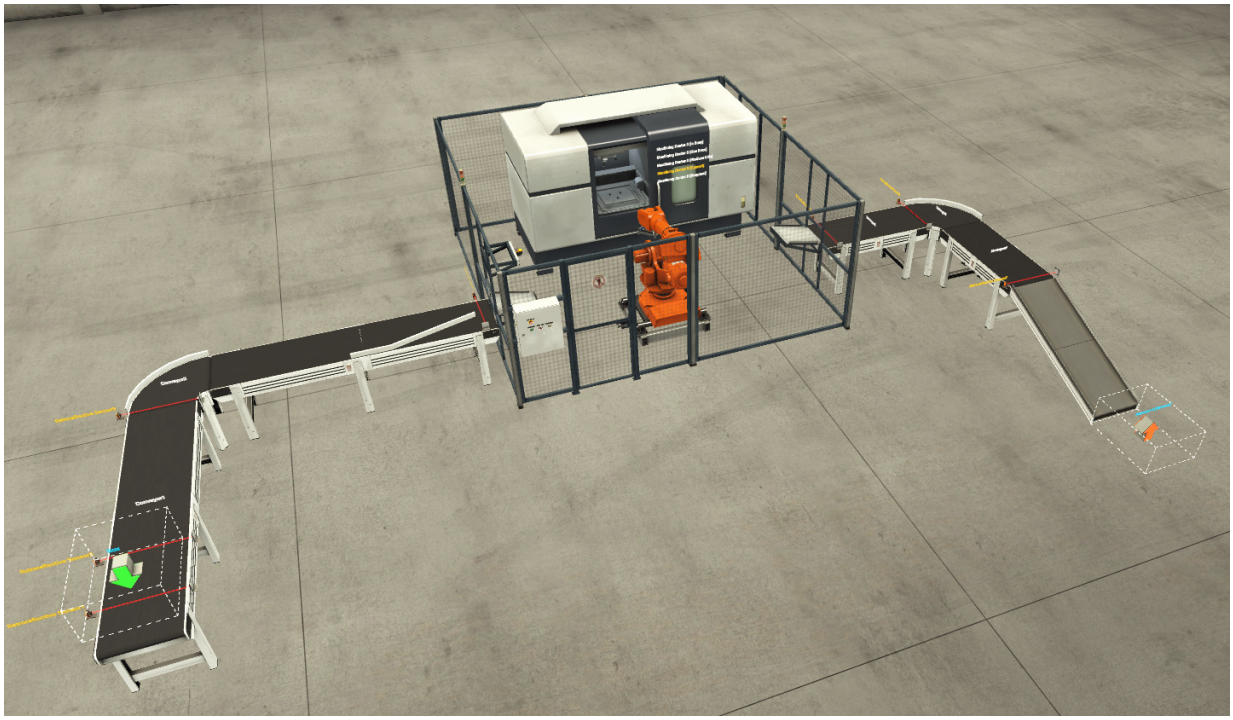


Obrázok č. 53: Schéma modelového výrobného systému [66]

Diskrétne udalostný systém je možné vo všeobecnosti zadefinovať ako systém, ktorý môže nadobúdať viacero stavov, pričom prechody medzi jednotlivými stavmi sú vyvolávané udalosťami (angl. event-driven system). V tomto prípade je uvažovaný **virtuálny model** diskrétneho udalostného systému vo forme **výrobného systému**, ktorý pozostáva z pásových dopravníkov a obrábacieho centra. Model sa skladá z viacerých častí (dielov), ktoré by dynamicky mali reagovať na udalosti, ktoré v systéme nastávajú. Na **vstupnom vysielači** (**emitter**) sú v istých časových intervaloch generované **kusy materiálu** (**polovýrobky**), ktoré je potrebné dopraviť do **obrábacieho centra**. V obrábacom centre sa z nich procesom obrábania stávajú **výrobky**, ktoré sú pripravené na expedíciu zákazníkom. Premiestňovanie materiálu a výrobkov zabezpečujú **pásové dopravníky** (výrobné pásy). Tých máme konkrétne šesť (C1 — C6). Funkčnosť pásových dopravníkov zabezpečuje sedem **svetelných senzorov** (R1 — R7). Pomocou emittera je vyslaný materiál na pás, kde ho zachytí svetelný senzor R1 a následne sa spustí pásový dopravník C1. Vtedy, keď daný produkt zaznamená senzor R3, spustí sa rohový pás

C2 a následne aj pás C3, ktorým materiál putuje do obrábacieho centra. Tu sa materiál spracuje na konečný produkt, ktorý postupuje na pás C4, ktorý sa spustí po zaznamenaní senzorom R5. Následne senzorom R6 sú spustené pásy C5 a C6. Finálny výrobok je zaznamenaný senzorom R7, ktorý by mal tiež zabezpečovať počítanie počtu hotových výrobkov. Je vhodné použiť zarovnávač, ktorým je docielené to, aby produkt správne vstúpil do obrábacieho centra. Je potrebné tiež zabezpečiť, aby vo výrobnom systéme nedochádzalo ku kolíziám medzi kusmi materiálu a tiež medzi finálnymi (hotovými) výrobkami.

Na obrázku č. 54 je možné vidieť celkový pohľad na virtuálny model výrobnej linky.

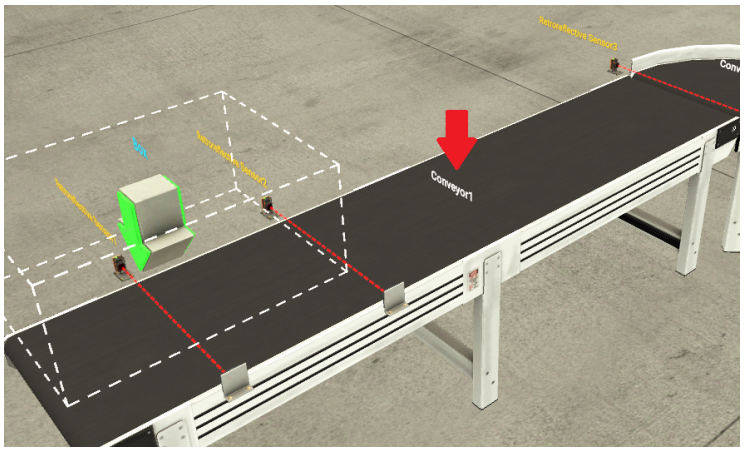


Obrázok č. 54: Celkový pohľad na virtuálnu výrobnú linku [66]

4.1.2 Komponenty použité na tvorbu modelu linky

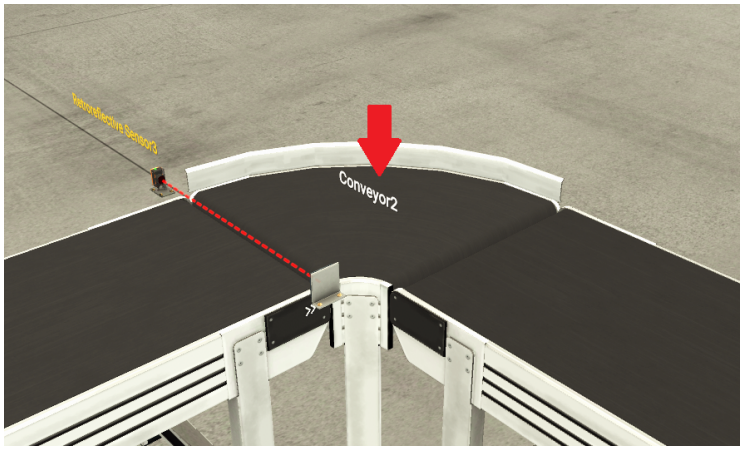
Na vytvorenie uvedeného modelu výrobnej linky vo Factory I/O bolo potrebných viacero dielov. V nasledujúcich bodoch je možné vidieť ich charakteristiku.

1. **Pásový dopravník (angl. Belt conveyor)** — tento komponent sa používa na prepravu ľahkého nákladu. Pásové dopravníky sú v dostupných dĺžkach 2, 4 a 6 metrov a v analógovom (vieme nastavovať rýchlosť dopravníka) a digitálnom variante.



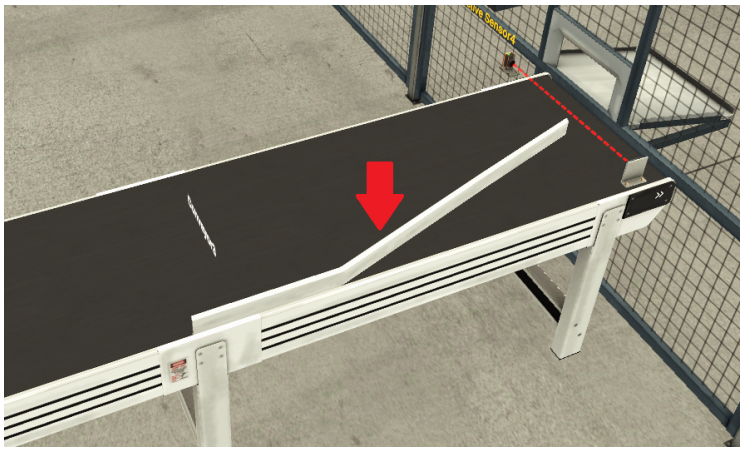
Obrázok č. 55: Pásový dopravník [66]

2. **Zakrivený pásový dopravník (angl. Curved belt conveyor)** — využíva sa na prepravu ľahkého nákladu a je takisto dostupný v analógových a digitálnych variantoch.



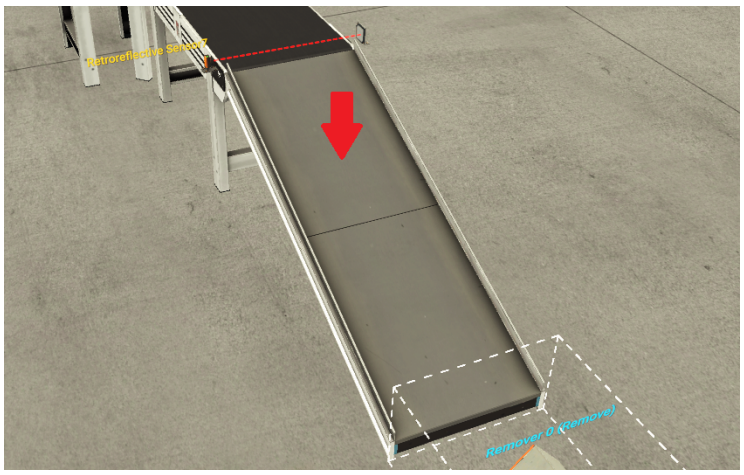
Obrázok č. 56: Zakrivený pásový dopravník [66]

3. **Zarovnávač (angl. Aligner)** — ide o kovovú konštrukciu, ktorá je pripevnená k dopravníku, aby sa predišlo pádu výrobku pri preprave. K dispozícii sú štyri druhy v rôznych farbách.



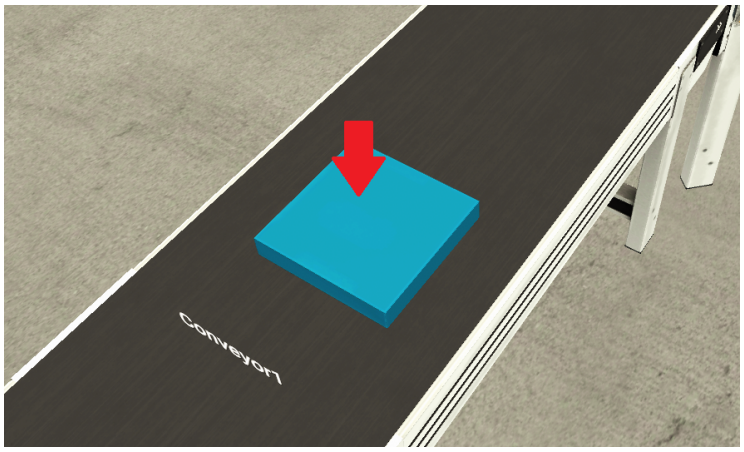
Obrázok č. 57: Zarovnávač [66]

4. **Kĺzačka (angl. Chute conveyor)** — slúži zväčša na expedíciu položiek z pásových dopravníkov.



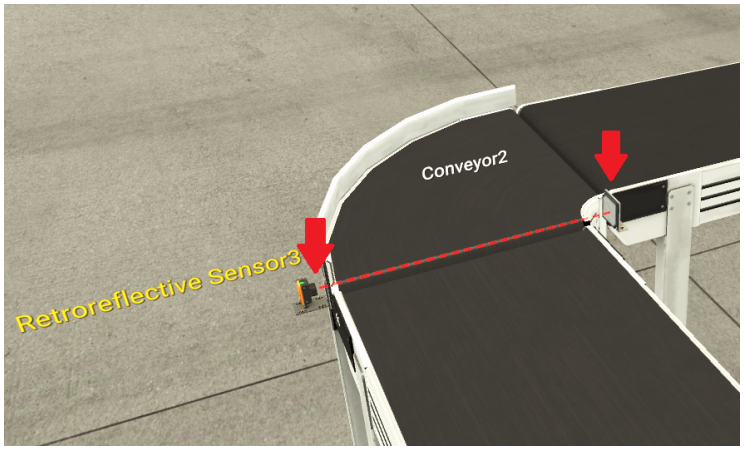
Obrázok č. 58: Kĺzačka [66]

5. **Neopracovaný materiál (angl. Raw Material)** — ide o kovový alebo plastový materiál na výrobu vrchnákov alebo podstavcov. V opisovanom prípade je chápaný ako polovýrobok, ktorý treba opracovať (obrobiť) na hotový výrobok.



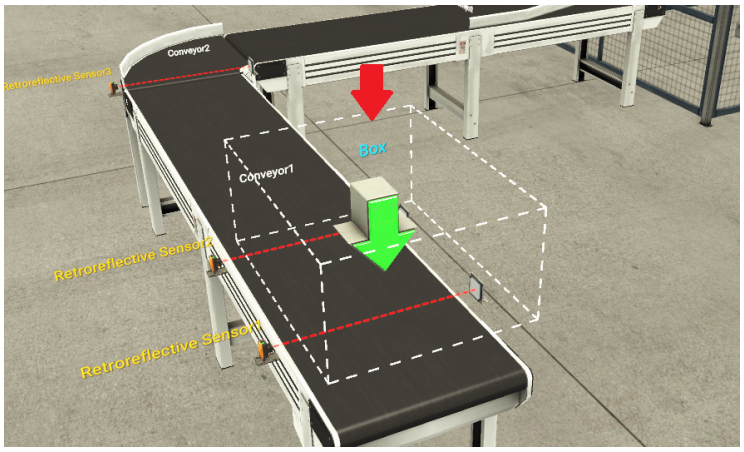
Obrázok č. 59: Neopracovaný materiál [66]

6. **Retroreflexný senzor a odrazka (angl. Retroreflective Sensor and Reflector)** — využíva sa spoločne s odrazkou, pričom je vybavený dvomi LED diódmi, ktoré indikujú správe nastavenie.



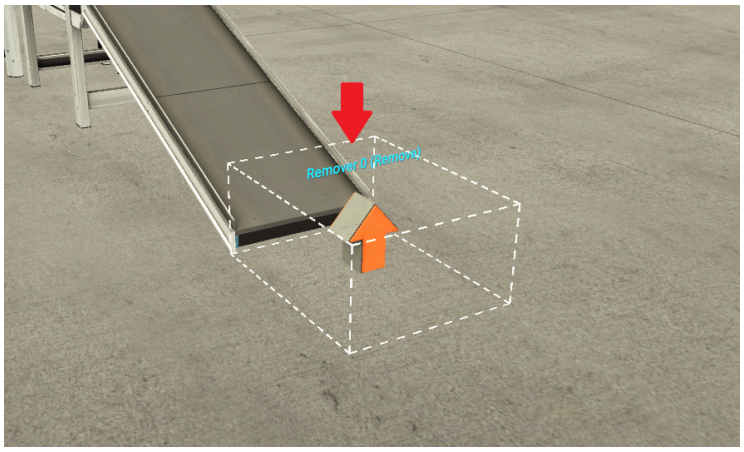
Obrázok č. 60: Retroreflexný senzor a odrazka [66]

7. **Vysielač (angl. Emitter)** — ide o vstupný bod výrobnej linky, ktorý zabezpečuje prísun výrobných súčiastok (neopracovaných materiálov) na ňu. Neopracované materiály sa automaticky generujú v časových intervaloch, ktoré súvisia s užívateľským nastavením emittera.



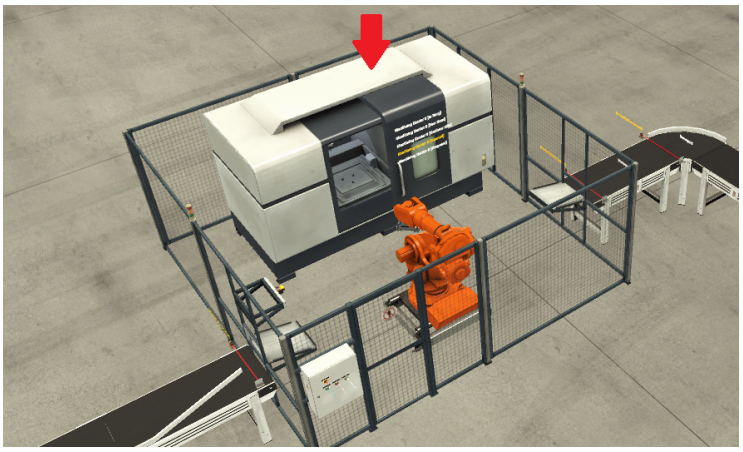
Obrázok č. 61: Vysielač [66]

8. **Odstraňovač (angl. Remover)** — odstráni jednu alebo viaceré položky zo scény.



Obrázok č. 62: Odstraňovač [66]

9. **Obrábacie / výrobné centrum (angl. Machining Center)** — ide o robot slúžiaci na výrobu podstavcov.



Obrázok č. 63: Obrábacie centrum [66]

4.1.3 Riadenie diskrétneho udalostného systému

Pre riadenie diskrétneho udalostného systému sa v tejto prípadovej štúdii využil open-source editor a runtime OpenPLC. V OpenPLC využívame jazyk Ladder logic (tzv. rebríkový diagram).

V hlavnom okne OpenPLC sa v hornej časti zobrazí vstupné pole, ktoré umožňuje vytvárať premenné pre PLC program. Programy PLC sú cyklické, čo znamená, že sa začnú vykonávať prvým príkazom, skončia posledným príkazom, istý čas počkajú (aby sa udržal rovnaký takt) a následne sa príkazy opakujú. Predvolená hodnota čakania je 20 ms. Bola ponechaná predvolená hodnota, aby program nespotreboval 100 % CPU zariadenia, keďže toto PLC chápeme len ako softPLC a pre demonštračné účely je to postačujúce. Na obrázku č. 64 sú využívané globálne premenné **RetroreflectiveSensor** od 1 po 7, ktoré snímajú produkt na páse, na základe ktorého je vypínaný alebo zapínaný pás, prípadne sa ráta počet polovýrobkov alebo výrobkov. Hodnoty výrobkov a následne prepočty sú riešené cez globálne premenné **Result** 1 až 3. **Result1** počíta, koľko polovýrobkov bolo pustených na pás. **Result2** hovorí, koľko je hotových finálnych (koncových) výrobkov a **Result3** udáva, koľko výrobkov je aktuálne vo výrobe. Pre beh výrobných pásov sa využívajú premenné **Conveyor 1 až 6**. Ďalej sa využívajú rôzne časovače a matematické funkcie. Časovačmi napríklad docieľujeme, aby nedošlo ku kolíziám. Senzormi tiež vyhodnocujeme dobu výrobkov na páse, aby nedošlo k ďalším potenciálnym kolíziám.

#	Name	Class	Type	Location	Initial Value
1	RetroreflectiveSensor1	Local	BOOL	%QX0.0	true
2	RetroreflectiveSensor2	Local	BOOL	%QX0.1	true
3	RetroreflectiveSensor3	Local	BOOL	%QX0.2	true
4	RetroreflectiveSensor4	Local	BOOL	%QX0.3	true
5	RetroreflectiveSensor5	Local	BOOL	%QX0.4	true
6	RetroreflectiveSensor6	Local	BOOL	%QX0.5	true
7	RetroreflectiveSensor7	Local	BOOL	%QX0.6	true
8	Box	Local	BOOL	%QX0.7	
9	Result1	Local	INT	%MW0	0
10	Result2	Local	INT	%MW1	0
11	Result3	Local	INT	%MW2	0
12	Conveyor1	Local	BOOL	%QX1.0	
13	Conveyor2	Local	BOOL	%QX1.1	
14	Conveyor3	Local	BOOL	%QX1.2	
15	Conveyor4	Local	BOOL	%QX1.3	
16	Conveyor5	Local	BOOL	%QX1.4	
17	Conveyor6	Local	BOOL	%QX1.5	
18	Robot	Local	BOOL	%QX1.6	true
19	Factoryio_stop	Local	BOOL	%QX1.7	
20	Factoryio_start	Local	BOOL	%QX2.0	
21	VarTime	Local	TIME		
22	VarTime0	Local	TIME		
23	VarTime1	Local	TIME		
24	SR0	Local	SR		
25	TOF0	Local	TOF		
26	TOF1	Local	TOF		
27	SR1	Local	SR		
28	TOF2	Local	TOF		
29	TOF3	Local	TOF		

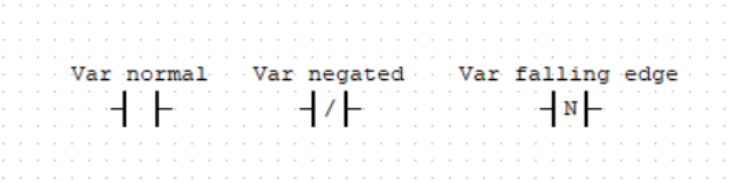
Obrázok č. 64: Premenné v riadiacom programe [66]

V strednej časti editora OpenPLC tvoríme diagram. Diagram tvoríme s pomocou panelu s nástrojmi. Program je „skenovaný“ alebo spúšťaný procesorom zľava doprava a zhora nadol. Na zostavenie nášho programu využívame nasledujúce elementy:

1. **Power rails** — prípad prázdnej priečky s ľavou a pravou napájacou lištou, kde sa zohľadňuje stav ľavej napájacej lišty po celú dobu a pre pravú lištu nie je definovaný žiadny stav.

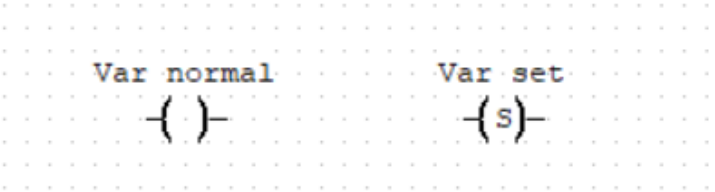
Obrázok č. 65: Power rails [66]

2. **Contact** — ide o prvok (obrázok č. 66), ktorý predstavuje vstupy. Ako vstup si je možné predstaviť napríklad tlačidlo alebo senzor. Tieto kontakty „prepúšťajú signál“ podľa toho, či vstup má byť 1 (normálne otvorené — prvý symbol na obrázku č. 66) alebo 0 (normálne zatvorené — druhý symbol na obrázku č. 66). Tiež je možné testovať nábežnú alebo dobežnú hranu (posledný symbol na obrázku č. 66).



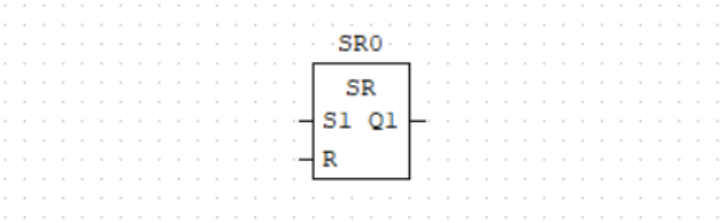
Obrázok č. 66: Contacts [66]

3. **Coils** — Cievky (obrázok č. 67) nám predstavujú výstupy (aktuátory), ktoré chceme nastavovať. Buď neustále kopírujú hodnotu vstupu (prvý symbol na obrázku č. 67) alebo sa v prípade prichádzajúceho signálu 1 nastavia na hodnotu 1 a už takto zostanú — tzv. SET (druhý symbol na obrázku č. 67). Hodnota vtedy ostáva na 1, pokiaľ nenastane v programe príkaz RESET.



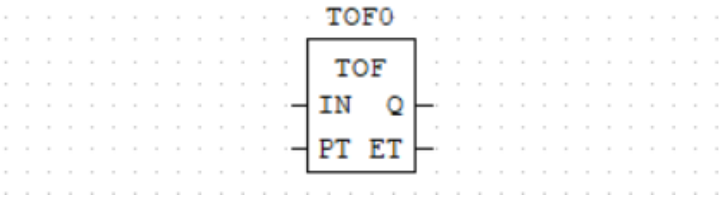
Obrázok č. 67: Coils [66]

4. **SR** — Ide o funkčný blok (set — reset), ktorý udržuje svoj výstup v jednom z dvoch stabilných stavov, teda TRUE alebo FALSE. Výstup možno nastaviť alebo resetovať s využitím TRUE signálu do Set alebo Reset vstupu. Ak sú oba vstupy TRUE, tak výstup je TRUE.



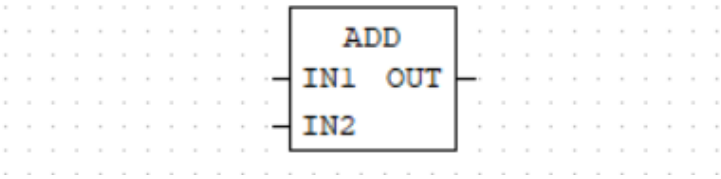
Obrázok č. 68: Set — reset [66]

5. **TOF** — Je to funkčný blok, ktorý spôsobí oneskorené vypnutie danej vetvy po tom, ako prestane „tiecť“ signál (po tom ako sa objaví 0).



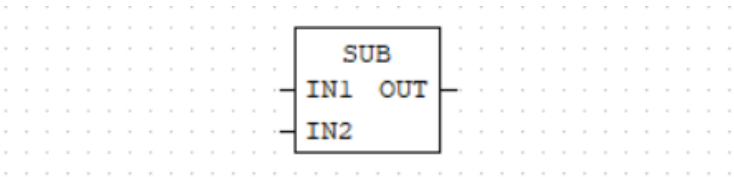
Obrázok č. 69: TOF [66]

6. **ADD** — Aritmetická funkcia, ktorá pridá vstup IN1 k vstupu IN2.



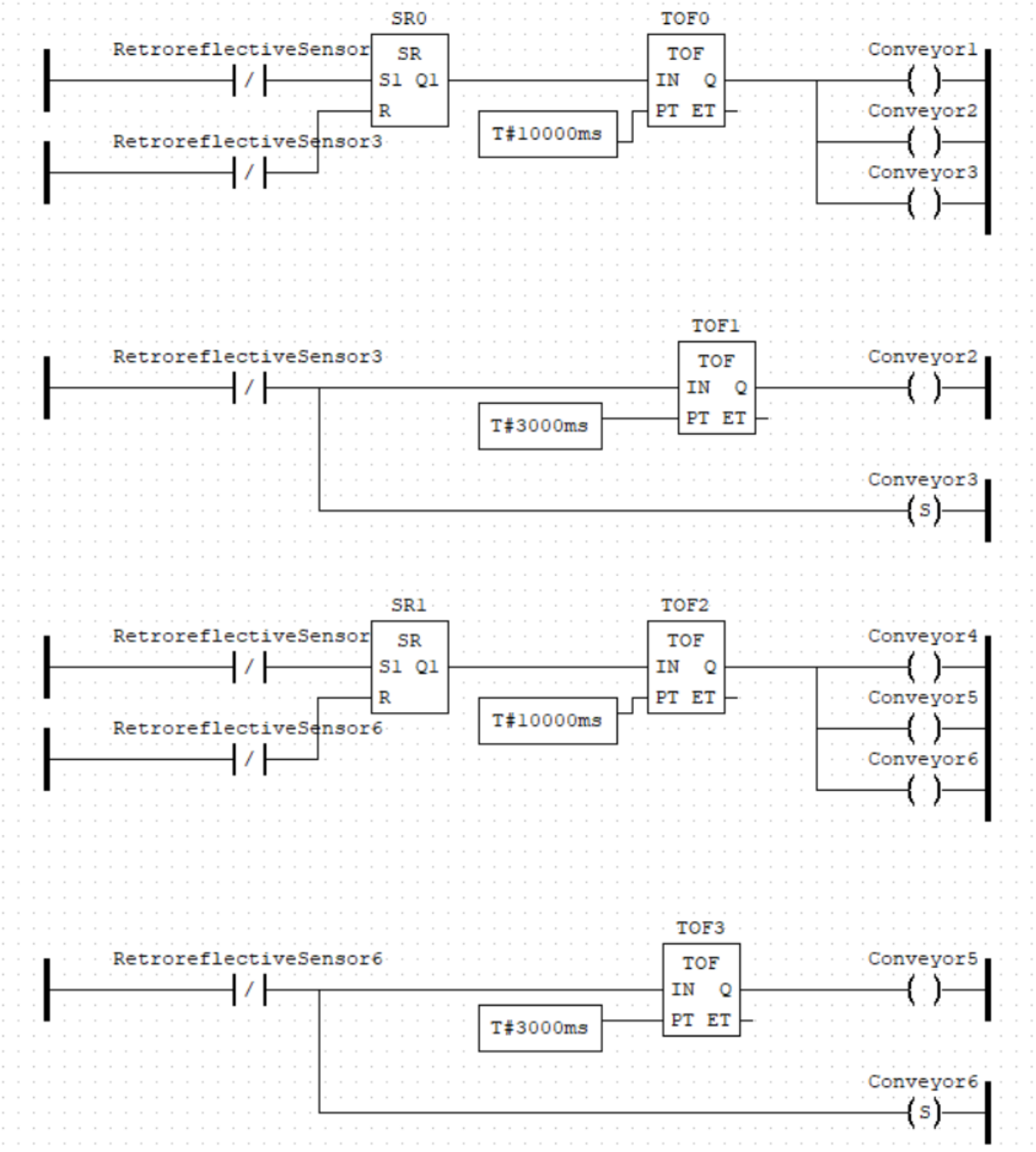
Obrázok č. 70: ADD [66]

7. **SUB** — Aritmetická funkcia, kde SUB odpočíta IN2 od IN1 a uloží výsledok do cieľa.



Obrázok č. 71: SUB [66]

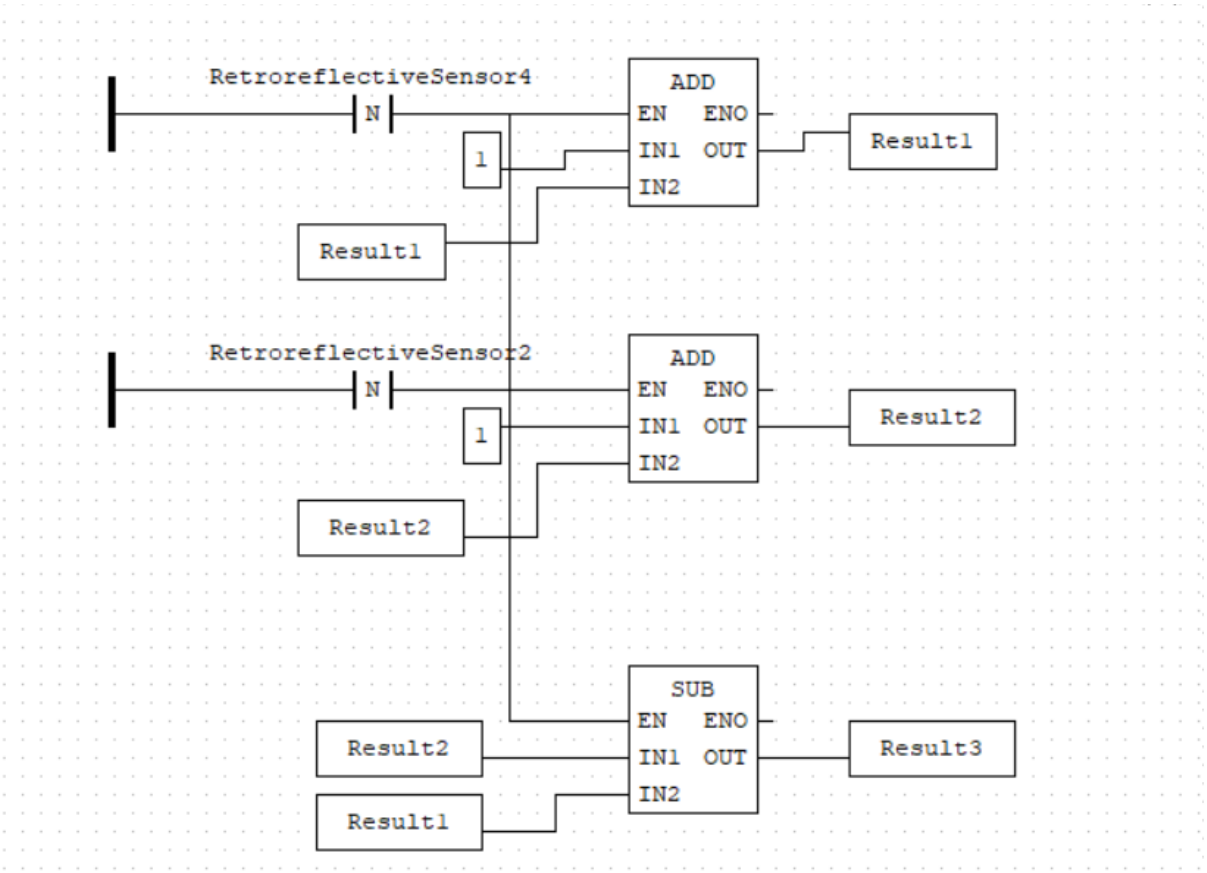
Na záver možno konštatovať, že uvedený rebríkový diagram (rebríková schéma) zabezpečuje chod celého výrobného systému. Obsahuje 11 priečok, 11 kontaktov, 15 cievok, 3 SR funkčné bloky, 6 TOF funčných blokov, 2 ADD aritmetické funkcie a 1 SUB aritmetickú funkciu.



Obrázok č. 72: Rebríkový diagram — časť 1. [66]

Prvá časť rebríkového diagramu zabezpečuje chod prvých troch pásových dopravníkov. Ak **RetroreflectiveSensor1** a **RetroreflectiveSensor3** zaznamená materiál, tak dopravníky **Conveyor1**, **Conveyor2** a **Conveyor3** sa spustia na dobu 10 sekúnd a materiál putuje po dopravníkoch smerom k stroju. Avšak ak **RetroreflectiveSensor1** a **RetroreflectiveSensor3** nezaznamená materiál na dopravníkovom páse dlhšie ako 10 sekúnd, pásy sa vypnú. Existuje ďalšia podmienka, a to taká, že ak **RetroreflectiveSensor3** zachytí materiál, tak **Conveyor2** a **Conveyor3** sa spustí na 3 sekundy. Je to z toho dôvodu, že

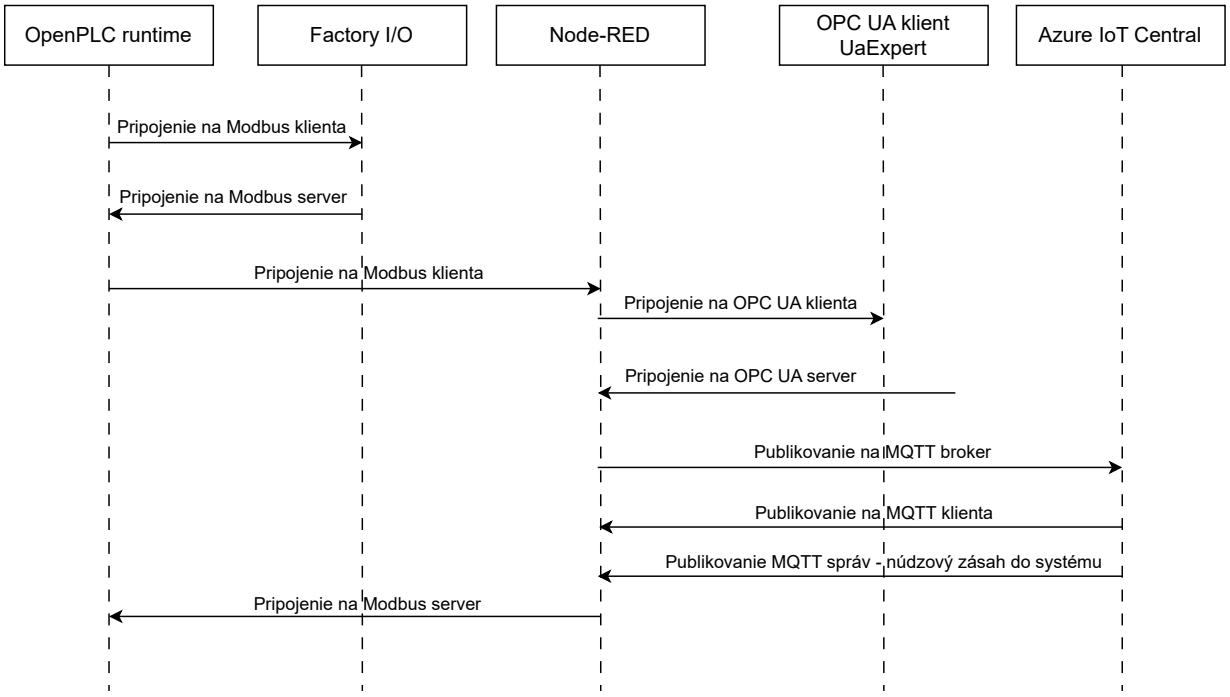
môže ísť o posledné kusy a **RetroreflectiveSensor1** by už materiál nezachytil a nastala by kolízia. Rovnaký postup nastáva aj pri druhej časti s **RetroreflectiveSensor5** a **RetroreflectiveSensor6** a **Conveyor4**, **Conveyor5** a **Conveyor6**.



Obrázok č. 73: Rebríkový diagram — časť 2. [66]

Druhú časť rebríkového diagramu tvoria bloky a aritmetické funkcie. **RetroreflectiveSensor2** vchádza do funkcie ADD a slúži na počítanie kusov surového materiálu (polovýrobkov), ktoré sa položili na daný pás a hodnotu zapíše do premennej **Result1**. **RetroreflectiveSensor7** slúži na počítanie už vyrobených podstavcov (výrobkov) a hodnotu je zapisovaná do premennej **Result2**. Aritmetická funkcia SUB potom tieto dva premenné **Result1** a **Result2** od seba odčíta a táto hodnota určuje, koľko výrobkov/kusov materiálu sa momentálne nachádza na pásových dopravníkoch.

4.1.4 Komunikácia jednotlivých subsystémov



Obrázok č. 74: Sekvenčný diagram komunikácie [66]

Komunikácia prebieha najprv medzi OpenPLC runtime a simulátorom Factory I/O. Komunikácia je zabezpečená za pomoci Modbus protokolu a to z toho dôvodu, že OpenPLC nepodporuje modernejšie protokoly (napr. OPC UA). OpenPLC sa v tomto prípade správa ako Modbus server a Factory I/O sa správa ako Modbus klient. Nakoľko chceme dáta posielať do cloudovej aplikácie, prípadne ich poskytnúť ďalším klientom pomocou OPC UA servera, potrebujeme prostredníka (middleware) vo forme Node-REDu. Node-RED nie je teda potrebný pre samotné riadenie výroby, ale je využívaný pre možnosť zdieľania dát pre OPC UA server a do cloudovej aplikácie. Z Node-REDu do cloudu Microsoft Azure a naspäť je komunikované pomocou komunikačného protokolu MQTT.

4.1.5 Prepojenie OpenPLC runtime s Node-RED

Aby bolo možné správne prepojiť OpenPLC runtime s middlewarom Node-RED, tak je potrebné pochopiť, ako funguje protokol Modbus a čo je v opisovanom prípade server a čo je klient. OpenPLC dokonca dokáže súčasne fungovať aj ako server a aj ako klient, čo však aktívne využívať nebudeme.

Modbus ponúka 4 typy prenášaných dát [66].

- **Discrete Input** — Jeden bit (BOOL), ktorý sa používa na binárny vstup (napríklad zo senzorov). V opisovanom prípade sú to adresy typu %IX. Dokáže ho zapisovať len Modbus server.

- **Coil** — Jeden bit (BOOL), ktorý sa väčšinou používa na binárny výstup. V opisovanom prípade sú to adresy typu %QX. Dokáže ho zapisovať nielen server, ale aj klient.
- **Input Register** — 16-bitový register určený len na čítanie. Je to niečo ako Discrete Input, akurát nejde o premennú typu BOOL, ale ide o INT s veľkosťou 16-bit, ktorý môže byť unsigned alebo signed.
- **Holding Register** — 16-bitový register určený na čítanie a aj zapisovanie. Je to niečo ako Coil, akurát nejde o premennú typu BOOL, ale ide o INT s veľkosťou 16-bit, ktorý môže byť unsigned alebo signed.

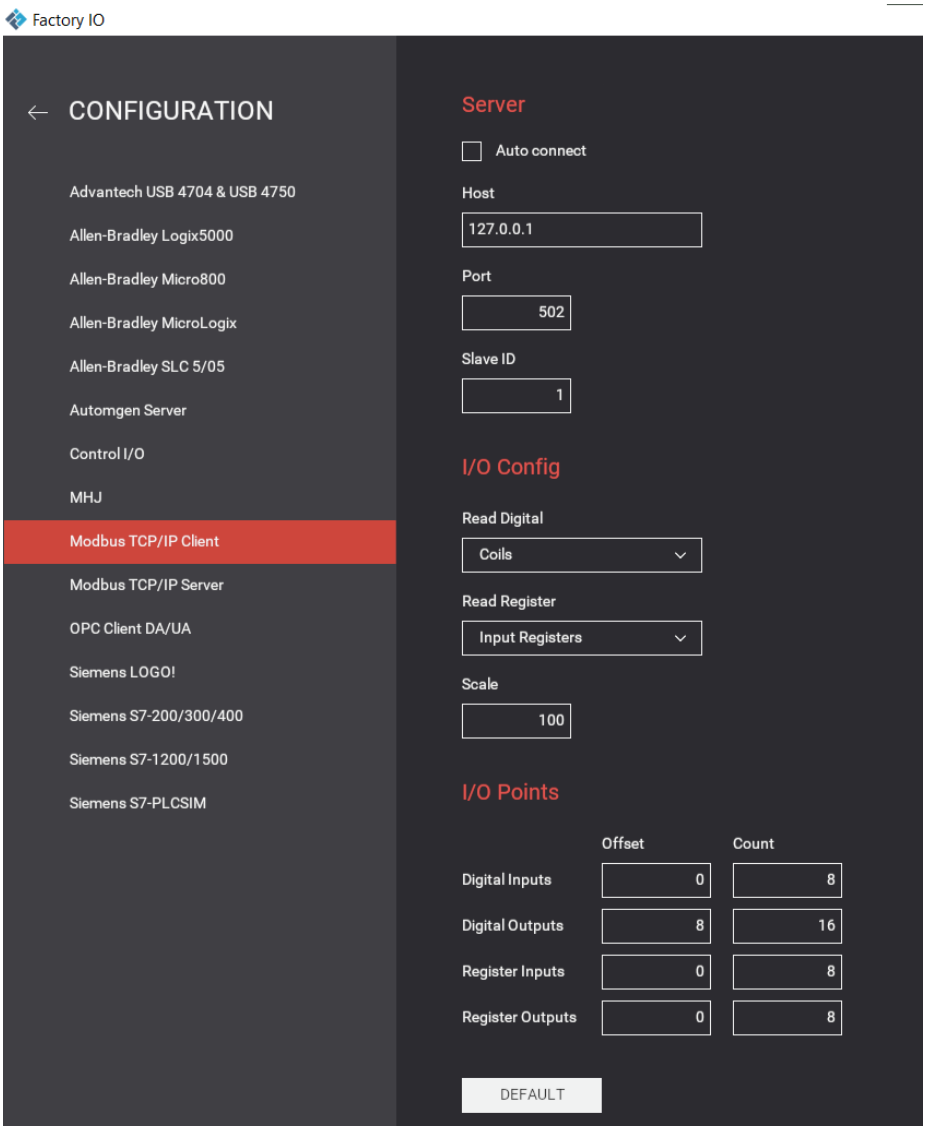
Ako je to teda s OpenPLC a jeho fungovaním? Ako už bolo uvedené, OpenPLC dokáže fungovať ako Modbus klient a aj ako Modbus server. V týchto módoch funguje súčasne, avšak pre každý je vyhradená iná adresa.

Pre nami využívaný mód fungovania OpenPLC ako servera má vyhradené adresy uvedené v tabuľke č.1.

Tabuľka č. 1: Premenné Modbus servera v OpenPLC Engine [43]

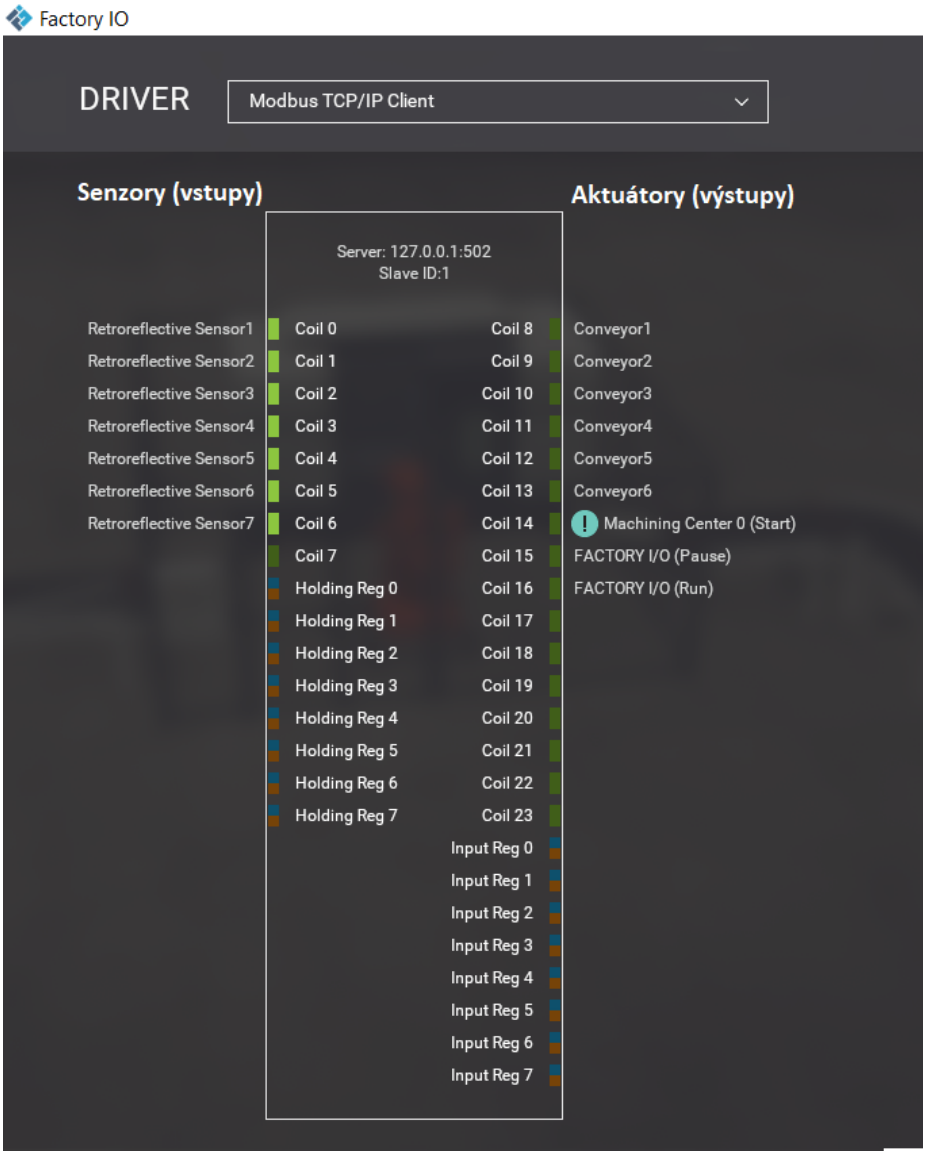
Modbus typ dát	**Využitie**	**PLC adresy**	**Modbus adresa**	**Veľkosť dát**	**Rozsah**	**Prístup**
Discrete Output Coils	digitálne výstupy	%QX0.0 — %QX99.7	0 — 799	1 bit	0 alebo 1	RW
Discrete Input Coils	digitálne vstupy	%IX0.0 — %IX99.7	0 — 799	1 bit	0 alebo 1	R
Analog Input Registers	analógové vstupy	%IW0 — %IW1023	0 — 1023	16 bitov	0 — 65535	R
Analog Output Holding Registers	analógové výstupy	%QW0 — %QW1023	0 — 1023	16 bitov	0 — 65535	RW

Následne je potrebné nastaviť Factory I/O ako Modbus klienta (obrázok č. 75). Nastavujeme *localhost*, kde beží OpenPLC, čo je 127.0.0.1. Ďalej je potrebné nastaviť, že digitálne vstupy využívame na Coiloch. Logickejšie by bolo využiť Inputy, ale nakoľko Modbus klient dokáže zapisovať len do Coils, tak pre vstupy je nutné využiť Coils. Ako posledné treba nastaviť I/O Points, pričom tu nastavujeme vstupy a výstupy podľa toho, koľko priestoru potrebujeme.



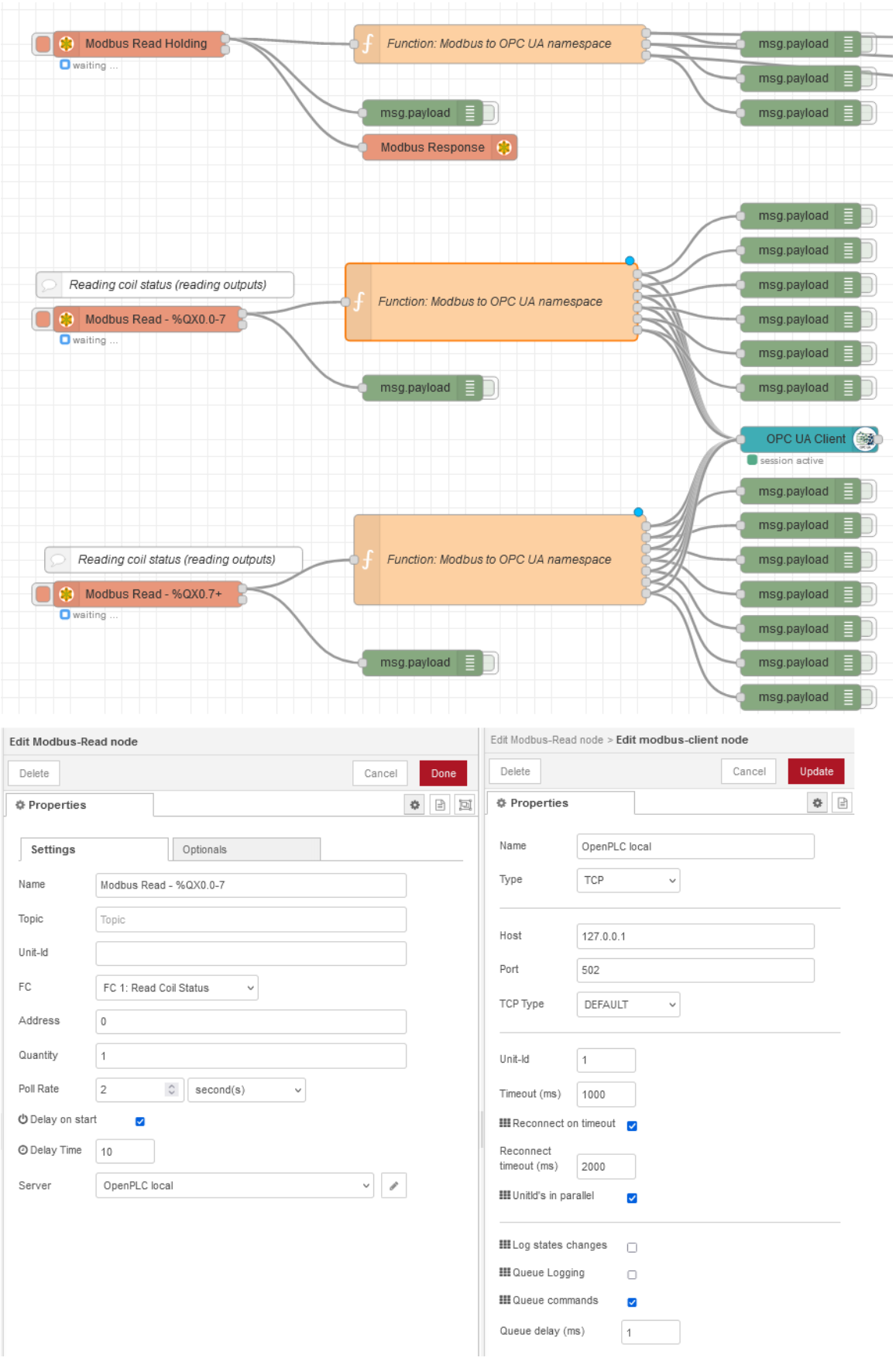
Obrázok č. 75: Factory I/O — Modbus klient [66]

Ako posledné je vo Factory I/O treba priradiť správne k súčastiam Factory I/O vstupné a výstupné premenné (obrázok č. 76). Adresy vo Factory I/O musia byť zhodné s adresami v OpenPLC.



Obrázok č. 76: Factory I/O — Modbus klient 2 [66]

Do OpenPLC runtime je potrebné nahrať náš program z OpenPLC editora, ktorý spustíme. Prejdeme teda na obrázok č. 77, kde je ukázané, ako sa načítavajú hodnoty z OpenPLC runtime a ako sa napájame cez protokol Modbus. Možnosť komunikácie cez Modbus sme získali nainštalovaním knižnice *node-red-contrib-modbus 5.14.1*.



Obrázok č. 77: Node-RED ako Modbus klient [66]

Cez uzol, ktorý bol nazvaný **Modbus Read — %QX0.0-7**, sú načítavané **vstupy** (**zo senzorov**) a cez uzol **Modbus Read — %QX0.7+** sú načítavané **výstupy** (**aktuátory**). Začína sa vstupmi od adresy 0, teda %QX0.0. Výstupy sú realizované od adresy %QX1.0. Uzol Modbus číta vždy celý bajt, čo v tomto prípade je v poriadku, keďže je presne 8 vstupných premenných. Premenné sú typu BOOL (true / false). Modbus Read funguje ako klient a je potrebné ho napojiť na server. My ho napojíme na server, ktorý bol nazvaný OpenPLC local a nastavili sme príslušnú adresu 127.0.0.1 a port 502. Ďalej určíme, že budeme čítať coils, čo je štandardný príkaz Modbus protokolu (*FC1: Read Coil Status*). V prípade našich vstupných premenných nastavíme adresu na 0, keďže načítavame od %QX0.0 (v prípade %QX1.0 by to bolo teda 8). Kvantita bola nastavená na 1, nakoľko čítame 1 bajt. Poll rate je nastavené na 2 sekundy, čo znamená, že hodnota sa načítava každé 2 sekundy.

Podobný postup bol použitý aj na **Modbus Read — %QX0.7+**, kde sú však čítané **výstupné premenné** od PLC adresy %QX1.0 a naša Modbus adresa je 8.

V PLC programe sú aj 3 hodnoty typu INT, ktoré sú uložené v registroch, ktoré majú iné adresy ako coils (tie sú typu BOOL). Jedná sa napríklad o počet vyrobených výrobkov. Tieto hodnoty sú načítavané pomocou uzla **Modbus Read Holding**.

4.1.6 OPC UA server a klient

Vzhľadom na to, že chceme hodnoty z výrobného systému ponúknuť aj ďalším užívateľom, ktorí môžu disponovať svojimi OPC UA klientmi, bolo nutné v Node-REDe realizovať OPC UA server, ktorý by tieto dáta poskytoval.

Ako už bolo uvedené, OpenPLC runtime nedokáže fungovať ako OPC UA server, nakoľko ide o open-source bezplatný nástroj, ktorého vývoj je pomerne pomalý. Pokročilejšie funkcionality sú výsadou pokročilejších PLC riešení, ako je napríklad Codesys alebo Siemens TIA Portal. Z toho dôvodu bude OPC UA server vytvorený pomocou Node-REDu, do ktorého dáta prichádzajú z OpenPLC runtime pomocou protokolu Modbus.

Na vytváranie OPC UA servera bola využitá knižnica *node-red-contrib-opcua 0.2.256*.

Pre **vytvorenie servera** je nutné využiť **uzol OPC UA server**, kde sa nastaví port (v opisovanom prípade 53880) a voliteľne aj jeho názov. Využívame predvolený názov. Je tiež možné nastaviť aj možnosti autentifikácie, nakoľko už bolo uvádzané v predošlých kapitolách, že komunikačný štandard OPC UA podporuje viacero bezpečnostných profilov. V tomto prípade pre jednoduchosť (a nakoľko fungujeme na localhoste) využívame anonymný prístup.

S vytvorením OPC UA servera súvisí aj vytvorenie **adresného priestoru** (angl. address space), teda premenných, ktoré na úvod budú prázdne alebo budú mať preddefi-

novanú hodnotu, pričom neskôr tieto premenné budú naplnené hodnotami, ktoré OpenPLC runtime posiela pomocou Modbus protokolu.

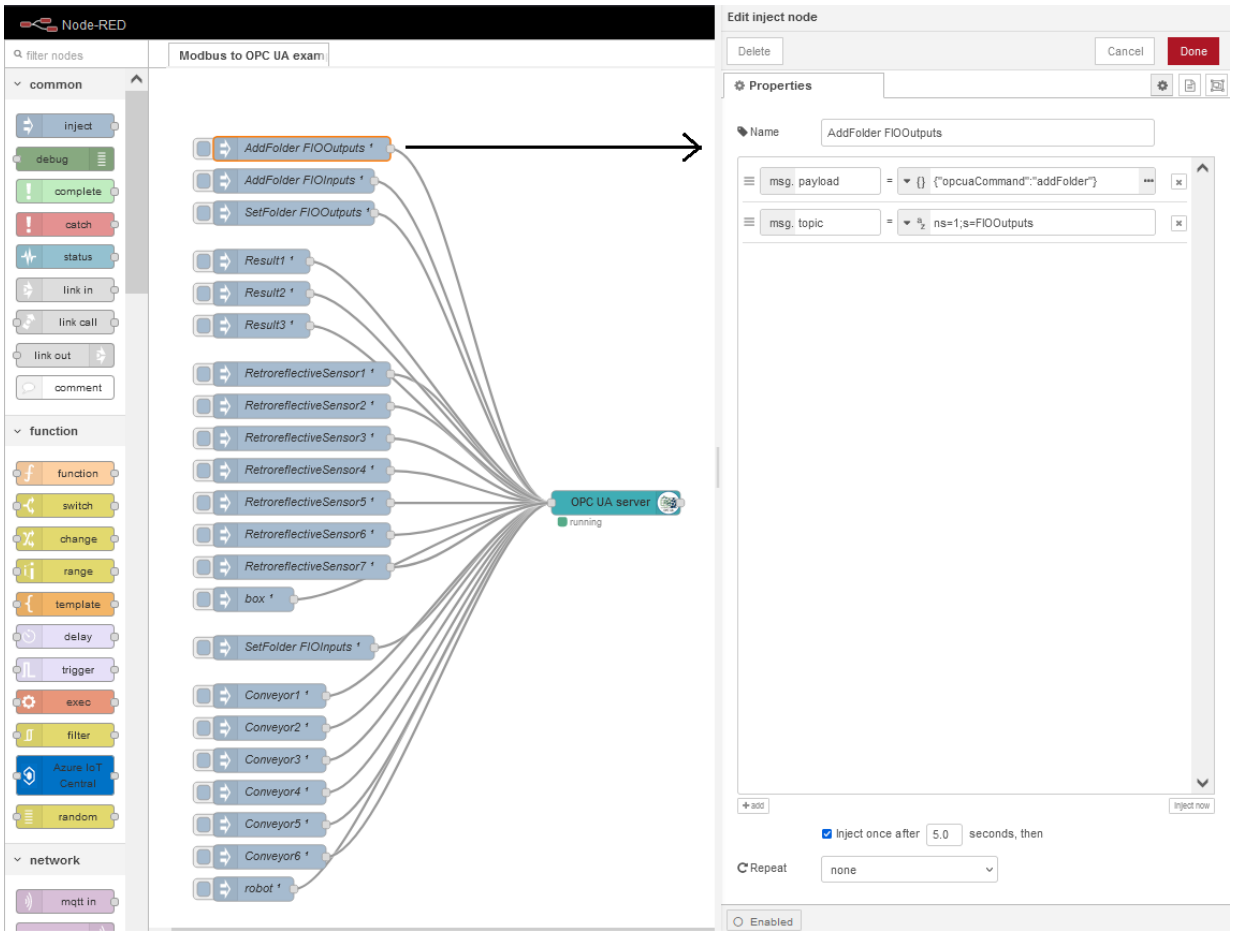
Aby sme vstupy a výstupy mali v adresnom priestore oddelené prehľadným spôsobom, tak boli vytvorené v adresnom priestore priečinky *FIOOutputs* a *FIOInputs*.

Treba poznamenať, že jednotlivé direktívy, ktoré sa posielajú uzlu OPC UA servera, sú posielané pomocou uzla typu *Inject*. Je potrebné teda nastaviť, aby sa tieto príkazy vykonali automaticky po spustení Node-RED programu, pričom je logické, že časovanie treba nastaviť tak, aby sa najprv spustili príkazy vytvárajúce priečinky a až potom sa vytvorili samotné premenné. Takto je úspešne vytvorený OPC UA server so želaným adresným priestorom.

My budeme využívať tieto direktívy (podporovaných je však viacero):

- **addFolder** — vytvára priečinok v adresnom priestore;
- **setFolder** — nastavuje priečinok v adresnom priestore;
- **addVariable** — pridáva premennú do nastaveného priečinka.

Direktívy sú viazané na *msg.payload* správ v Node-RED a samotný obsah na *msg.topic* správ. Toto je možné vidieť aj na obrázku č. 78, kde je ukázané vytváranie priečinka *FIOOutputs*. Obsahom *msg.payload* je príkaz na pridanie priečinka a v *msg.topic* vidíme definovaný namespace a názov priečinka.



Obrázok č. 78: Vytváranie OPC UA servera a jeho adresného priestoru v Node-RED [66]

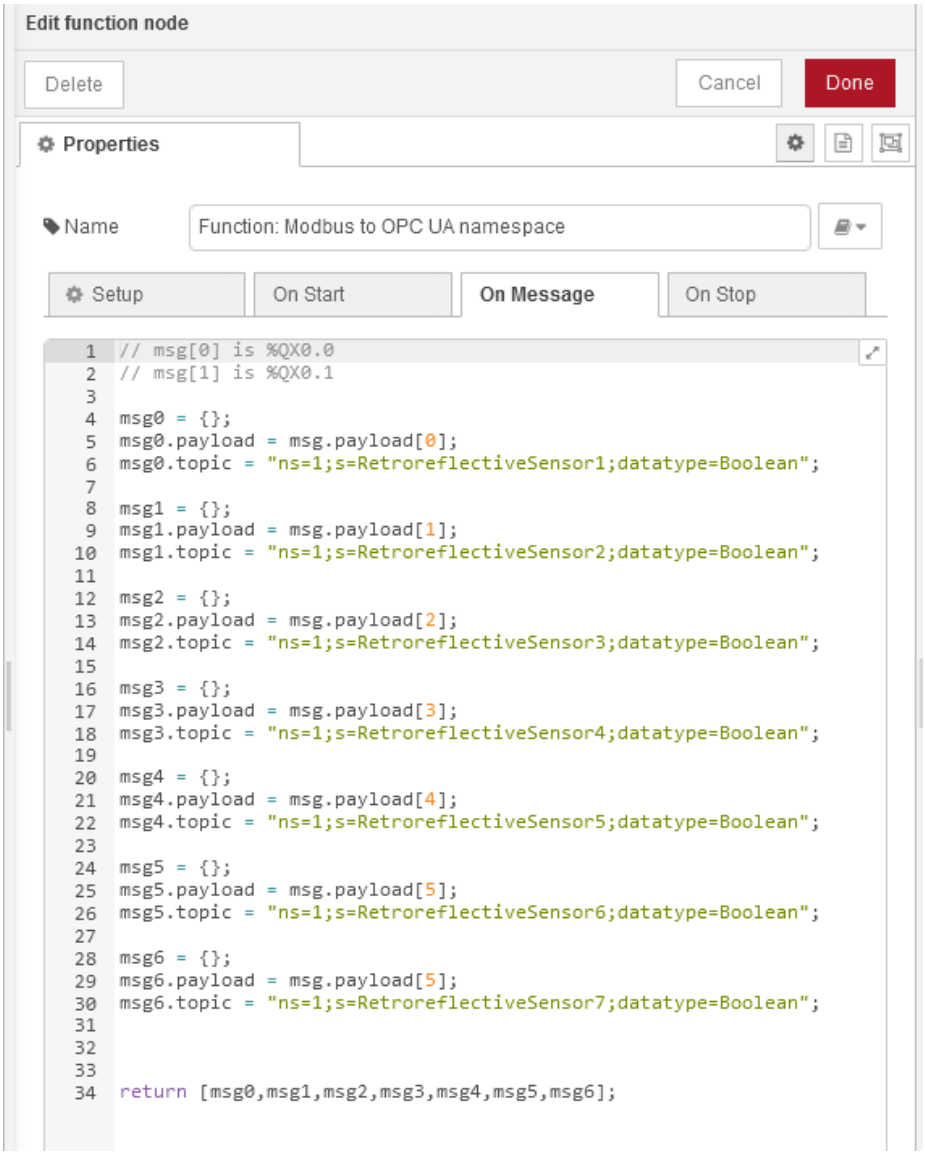
Pridanie premennej *RetroreflectiveSensor1* by vyzeralo nasledovne, pričom je možné vidieť, že premennej nastavujeme aj dátový typ:

- `msg.payload: {"opcuaCommand":"addVariable"}`
- `msg.topic: ns=1;s=RetroreflectiveSensor1;datatype=Boolean`

Na to, aby bolo možné napĺňať premenné dátami, sme potrebovali vytvoriť vlastnú funkciu v jazyku JavaScript, nakoľko v Node-RED nie je nutné využívať len vstavané uzly, ale dokážeme si vytvárať aj vlastné.

Všimnime si na obrázku č. 77 uzol s názvom **Function: Modbus to OPC UA namespace**, teda uzol predstavujúci našu vlastnú funkciu. Konkrétne bude opísaný ten uzol, ktorý je spojený s uzlom *Modbus Read — %QX0.0-7*. Uzol *Modbus Read — %QX0.0-7* posiela do uzla *Function: Modbus to OPC UA namespace* pole 8 hodnôt (typu BOOL). Na obrázku č. 79 je možné vidieť obsah uzla. Pre každú hodnotu získanú z Modbusu sme pripravili prázdnu premennú (*msg0,msg1,msg2,...*), kde sú ukladané jednotlivé prvky poľa. Do ich payloadu ukladáme samotnú hodnotu (typu BOOL). Zaujímavejšou časťou

je však *msg.topic*. Tu podľa dokumentácie OPC UA klienta nainštalovaného v Node-REDe je nutné definovať, do ktorej premennej v adresnom priestore OPC UA premenná pôjde. Máme teda zadefinované ID namespacu (skrátene *ns*), názov premennej a dátový typ, ktorý sa zhoduje so vstupnými premennými získanými z Modbus servera.

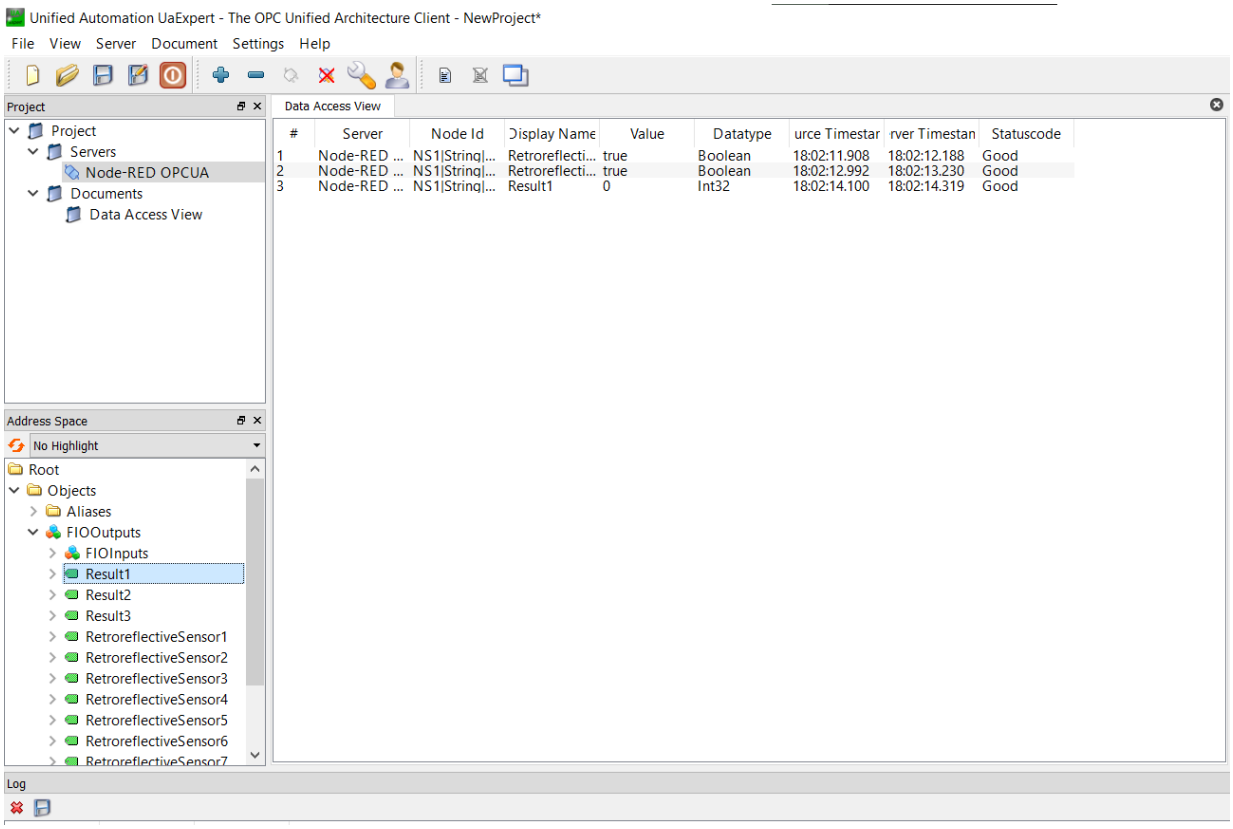


Obrázok č. 79: Funkcia na napĺňanie OPC UA servera hodnotami získanými pomocou Modbus protokolu z OpenPLC — *Function: Modbus to OPC UA namespace* [66]

Následne je uzol *Function: Modbus to OPC UA namespace* spojený s uzlom **OPC UA client**, ktorý hodnoty uloží do OPC UA servera.

Hodnoty z OPC UA servera dokážeme poskytovať aj iným OPC UA klientom, ako je klient nainštalovaný v Node-RED. Teoreticky by mohlo ísť o klienta, ktorý nemusí bežať len na počítači, ale aj na smartfóne, tablete, v cloudovom prostredí a podobne. Pre otestovanie sme využili OPC klienta **UaExpert**. UaExpert je navrhnutý ako všeobecný

testovací klient, ktorý podporuje funkcie OPC UA, ako sú DataAccess, Alarms & Conditions, Historical Access a iné. Ide o multiplatformový testovací klient OPC UA naprogramovaný v C++. Využíva grafickú knižnicu GUI QT. Na obrázku č. 80 je možné vidieť, že sa v priečinkoch vytvorili naše premenné, ktoré dokážeme prehľadne a pohodlne zobraziť v tomto klientovi.



Obrázok č. 80: Ukážka OPC UA klienta UaExpert [66]

4.1.7 Aplikácia v cloude Microsoft Azure

Jedným z hlavných cieľov tejto prípadovej štúdie je realizácia cloudovej aplikácie, pomocou ktorej by bolo možné monitorovať diskrétny udalostný výrobný systém — sledovať hodnoty vybraných veličín, tieto dáta vhodne vizualizovať, účelne ich spracovať a prípadne do výrobného systému aj zasahovať.

Na začiatku bolo potrebné určiť, čo chceme, aby sa v cloudovej aplikácii nachádzalo. Určené boli teda **funkcionálne požiadavky**:

- zobrazovanie načítaných hodnôt a akčných tlačidiel v prehľadnom paneli (angl. dashboard);
- komunikácia s Node-RED pomocou protokolu MQTT;
- zobrazovanie počtu vyrobených (finálnych/hotových) výrobkov odovzdaných do expedície;
- zobrazovanie počtu polovýrobkov (kusov surového materiálu), ktoré prišli do výroby;
- zobrazovanie počtu polovýrobkov/výrobkov, ktoré aktuálne sú na dopravníkoch (respektíve vo výrobnom systéme ako takom);
- grafické znázornenie aktuálnej teploty vo výrobnej hale;
- jednoduché spracovanie hodnôt aktuálnej teploty vo výrobnej hale za účelom vyvolania alarmu, ak teplota vystúpi nad istú hodnotu;
- možnosť núdzového zásahu do systému — pozastavenie a spustenie výrobnej linky.

Vyššie definované charakteristiky majú za úlohu opísať to, čo vytváraný aplikačný systém (cloudová aplikácia) bude schopná realizovať. Je však nevyhnutné identifikovať aj druhú skupinu požiadaviek, ktoré sa nebudú venovať funkcionalitám aplikácie. Pôjde o kvalitatívne, teda **nefunkcionálne požiadavky**:

- cloudová aplikácia by mala disponovať jednoduchým a intuitívnym používateľským rozhraním;
- jednotlivé zobrazované veličiny by mali byť súčasťou vhodného objektového modelu, pričom sa predpokladá vhodné využitie súčastí cloudového systému Microsoft Azure IoT Central;
- aplikácia by mala umožňovať jednoduchú rozšíriteľnosť o zobrazovanie ďalších veličín, prípadne možnosť pridania ďalších výrobných liniek, kde každá by mala svoj panel;

- z pohľadu jazykovej internacionalizácie by mala aplikácia využívať angličtinu.

Po návrhu aplikácie bolo potrebné prejsť k samotnej implementácii. Najefektívnejším spôsobom je využitie služby typu PaaS (Platform as a Service), teda využitie pripravených vývojárskych nástrojov na vytváranie vlastných cloudových aplikácií. Pôvodným plánom bolo spojiť dokopy viac služieb typu PaaS, napríklad Azure IoT Hub pre monitorovanie a pripájanie zariadení, Power BI pre vizualizáciu údajov, Azure Stream Analytics na analýzu údajov a napríklad Azure Functions pre spätný zásah do diskrétneho udalostného systému. Počas tvorby prípadovej štúdie bolo zistené, že efektívnym riešením bude využitie pomerne novej komplexnej služby typu aPaaS (application Platform as a Service) **Azure IoT Central**, ktorá v sebe už spája vyššie uvedené funkcionality.

Microsoft Azure IoT Central sa využíva na pripojenie a správu zariadení vo veľkom rozsahu a poskytuje spoľahlivé údaje pre obchodné štatistiky. Je ho možné tým pádom zaradiť medzi ERP (angl. enterprise resource planning) systémy. Zahŕňa v sebe viacero služieb typu PaaS, čím umožňuje vytvoriť jednoducho konfigurovateľné, komplexné a bezpečné IoT riešenia. Webové používateľské rozhranie umožňuje rýchlo prepojiť zariadenia, monitorovať stav zariadení, vytvárať pravidlá a spravovať milióny zariadení a ich údaje počas celého životného cyklu.

Ďalšie výhody Azure IoT Central sú:

- rýchla konektivita medzi IoT zariadeniami a cloudom;
- centralizovaná správa pre jednoduchosť rekonfigurácie a aktualizácie zariadení;
- vizualizácie a analýzy pre pochopenie údajov prichádzajúcich z IoT;
- rozšíriteľnosť.

Azure IoT Central funguje podobne ako Azure IoT Hub na báze dvojčiat zariadení (angl. device twin), pričom každé toto zariadenie vychádza z istej šablóny (angl. template). **Device template** je takzvaný návrh, ktorým definujeme charakteristiky a správanie typu zariadenia. Tieto zariadenia následne pripájame k našej aplikácii. Vieme napríklad definovať telemetriu, ktorú zariadenie odosiela, takže IoT Central môže vytvárať vizualizácie, ktoré využívajú správne fyzikálne jednotky a typy údajov. Nami vytvorená šablóna má názov **template-factoryio**. V nej sa nachádzajú modely zariadenia, ktoré si definujeme pre integrovanie s našou aplikáciou. Každý model má jedinečné ID a každému modelu tiež implementujeme jeho schopnosti, prípadne sémantický typ. Sémantický typ umožňuje IoT Centralu urobiť akýsi predpoklad o tom, ako s hodnotou zaobchádzať.

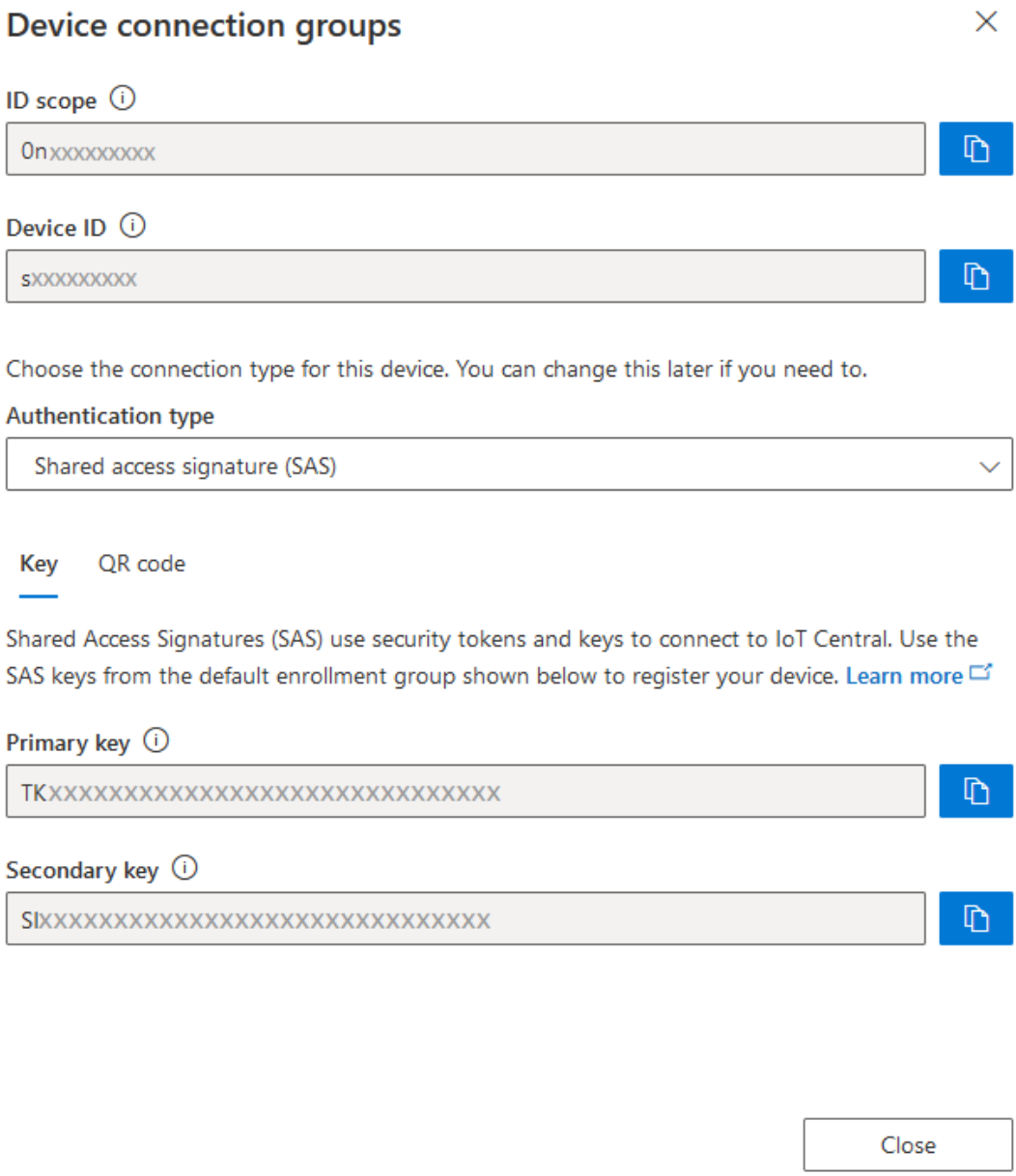
Schopnosti, ktoré môžeme naším modelom priradiť, sú:

- *properties* — dátové polia, ktoré predstavujú stav vášho zariadenia;
- *telemetry* — telemetria (merania) zo senzorov;
- *command* — metódy, ktoré môžu používatelia vášho zariadenia vykonávať na zariadení (napr. akčné zásahy do systému).

Štruktúra nášho zariadenia s názvom *conveyor-system1* je nasledovná, pričom prvý údaj udáva **systémový názov** a druhý údaj udáva **typ schopnosti**:

- turnOn — *command*;
- turnOff — *command*;
- produced — *telemetry*;
- entered — *telemetry*;
- on_line — *telemetry*;
- temperature — *telemetry*;
- location — *property*.

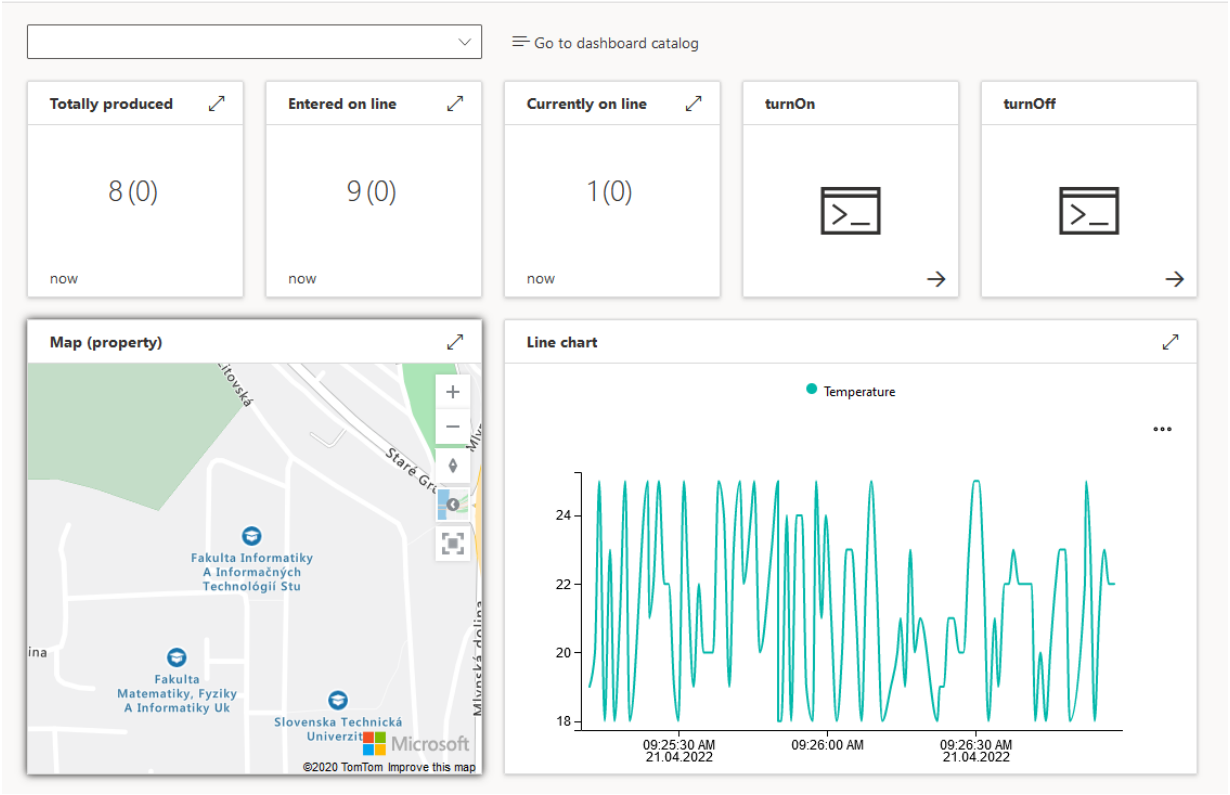
Pripojenie zariadenia je riešené pomocou autentifikácie typu SAS (angl. skratka Shared access signature). Náš model zariadenia v cloudovej aplikácii prepájame s Node-RED s pomocou údajov ID scope, Device ID a Primary key (obrázok č. 81).



Obrázok č. 81: Device connection groups [66]

Následne si je možné prezerať dáta, ktoré prijíma cloudová aplikácia. Pozrime sa na obrázok č. 82, kde je možné vidieť 3 hodnoty, ktoré cloud prijíma — *Totally produced*, *Entered on line a Currently on line*. Hodnota *Totally produced* udáva, koľko výrobkov bolo vyrobených a odovzdaných do expedície. Hodnota *Entered on line* udáva počet polovýrobkov (kusov surového materiálu) položených na pás. Hodnota *Currently on line* udáva počet aktuálnych polovýrobkov a výrobkov na páse. Ďalej je k dispozícii mapa, ktorá dokáže zobrazovať, kde sa daná výroba odohráva. Vedľa nej je umiestnený graf, ktorý zobrazuje teplotu ovzdušia v danej továrni. Nakoľko opisovaná výrobná linka nie je reálna, tak teplota je len simulovaná pomocou generovania náhodných hodnôt v pros-

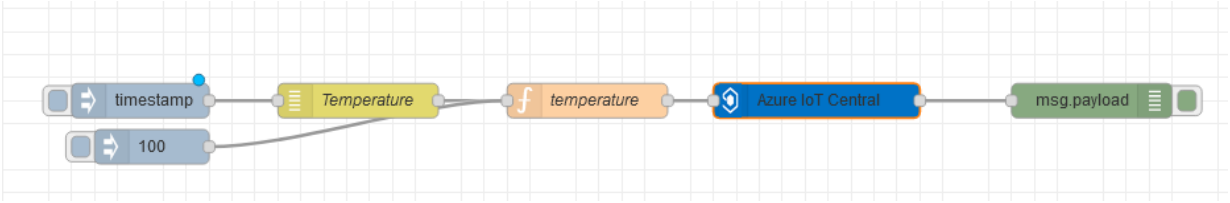
tredí Node-RED v rozmedzí 18 až 25. Vieme z Node-REDu tiež vyslať signál počas behu aplikácie, že teplota ovzdušia je 100 °C. V tomto prípade máme realizovanú na strane Azure IoT Central **udalosť** (**alarm**), ktorá zabezpečí zaslanie e-mailu poverenému pracovníkovi. Týmto sa demonštruje základná forma **spracovania a vyhodnotenia dát**, na základe ktorého je vykonaná udalosť. Udalosti sa v Azure IoT Central môžu vyberať z preddefinovaných typov alebo je možné naprogramovať vlastnú funkciu pomocou Azure Functions (serverless architektúra).



Obrázok č. 82: Dashboard v Azure IoT Central [66]

Ako prebieha v Node-RED posielanie dát, bude stručne opísané práve na príklade generovania hodnôt teploty (obrázok č. 83). Začína sa využitím uzla *timestamp*, ktorý zabezpečuje cyklický beh daného programového toku. Ďalej využívame *random* uzol (na obrázku s názvom *Temperature*), ktorý generuje celé náhodné čísla od 18 po 25. Tento uzol vchádza do vlastnej funkcie *temperature*, kde do *msg.payload* našej správy sa vkladá identifikátor, ktorý je zhodný s identifikátorom premennej *temperature* v cloude na danom modeli (dvojčati) zariadenia. Je tu pripravený aj uzol *inject 100*, ktorého manuálnym zatlačením je možné zaslať hodnotu teploty 100 °C, aby bolo možné simulovať požiar vo výrobnej hale. Celá vetva nakoniec vchádza do uzla *Azure IoT Central*, kde je definované ID zariadenia na cloude a kľúč k prístupu. Okrem tohto je tu definovaný

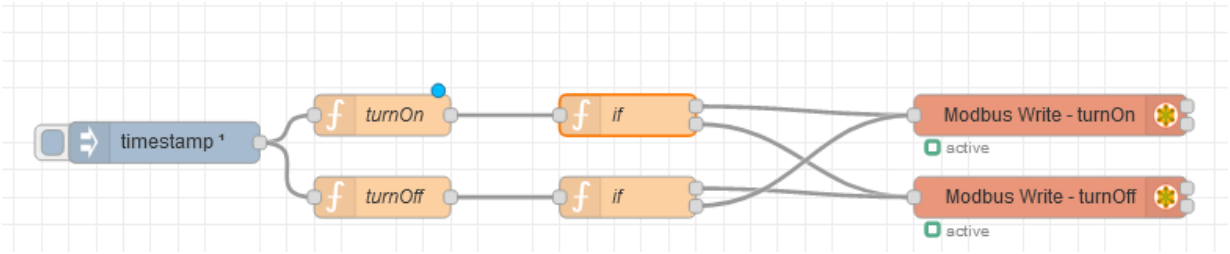
aj komunikačný protokol, v našom prípade ide o MQTT. Je možné komunikovať aj cez AMQP alebo HTTPS. Používaná je knižnica *azure-iot-central 1.6.0*.



Obrázok č. 83: Tok posielanie dát v Node-RED (teplota ovzdušia) [66]

Prostredníctvom cloudovej aplikácie je možné do opisovaného virtuálneho výrobného systému aj **zasahovať**. Toto je demonštrované pomocou implementácie možnosti núdzovo **pozastaviť a spustiť linku**. Na obrázku č. 82 je možné vidieť v pravom hornom rohu tlačidlá, ktoré to zabezpečujú.

V prvom rade je nutné v uzle Azure IoT Central nastaviť, na aké príkazy (commandy) tento uzol počúva. V tomto prípade to budú *turnOff* a *turnOn*. Ďalším dôležitým prvkom sú JavaScript funkcie, ktoré budú na tieto príkazy z cloudovej aplikácie odpovedať (obrázok č. 84). Tieto funkcie musia byť registrované v kontexte opisovaného Node-RED toku (angl. *flow context*). Toto sa deje (v prípade pozastavenie linky) pomocou príkazu `flow.set('turnOff',turnOff);`. O zvyšok sa stará uzol Azure IoT Central. Za JavaScript funkciami máme napojené ďalšie uzly, ktoré nám zabezpečia pozastavenie alebo spustenie linky. Príkaz sa zasiela do OpenPLC runtime pomocou komunikačného protokolu Modbus, a preto využívame uzly typu *Modbus Write*.



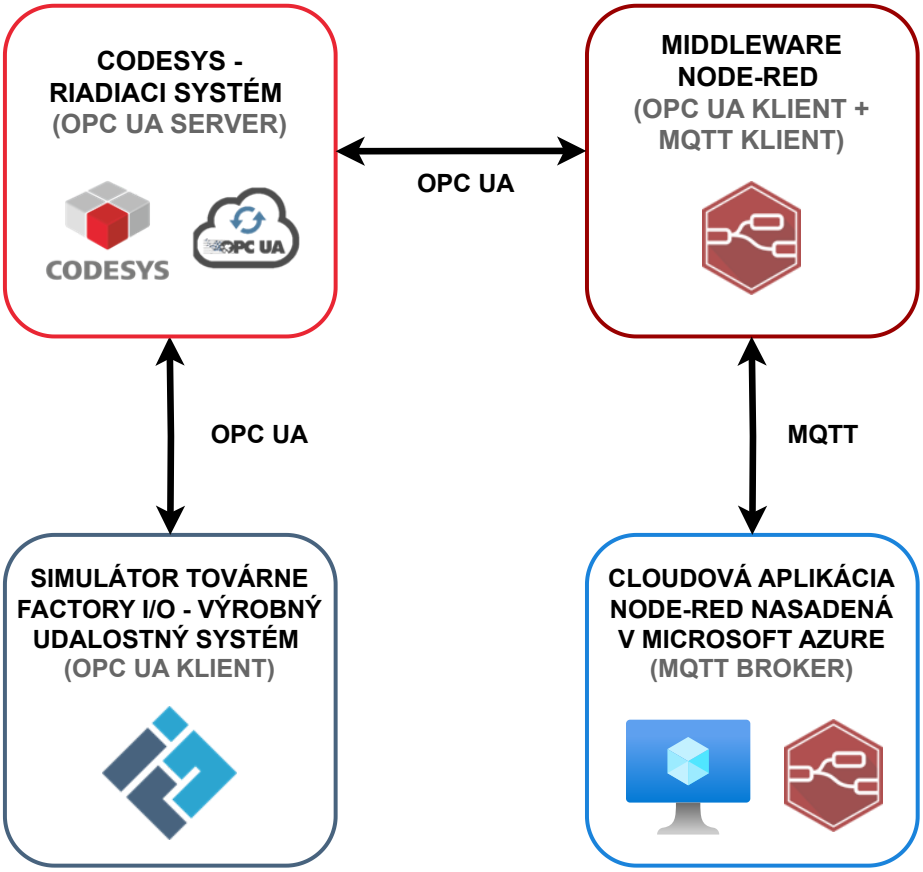
Obrázok č. 84: Tok v Node-RED zabezpečujúci pozastavenie a spustenie výrobného systému [66]

Videá k tejto edukačnej prípadovej štúdii je možné nájsť na tejto adrese:

- `https://bit.ly/pripad1`

Ďalšie informácie, návody a programové projekty je možné nájsť na e-learningovej webovej stránke `https://elearning.mechatronika.cool/?p=8342` .

4.2 Druhá prípadová štúdia: Prepojenie PLC s cloudovou aplikáciou realizovanou pomocou Node-RED



Obrázok č. 85: Codesys v prepojení s OPC UA a MS Azure

V druhej edukačnej prípadovej štúdii (obrázok č. 85) sme sa rozhodli namiesto neplateného a open-source PLC editora využiť komplexné riešenie pre automatizáciu zvané Codesys. Toto riešenie zabezpečí spojenie s Factory I/O pomocou moderného komunikačného protokolu a štandardu OPC UA. OPC UA server bude bežať prostredníctvom runtime Codesys a Factory I/O bude figurovať ako OPC UA klient. Pre účely riadenia nie je teda potrebný prostredník vo forme Node-RED. Node-RED však budeme využívať pre odchytávanie dát a ich zasielanie do cloudu a takisto bude využívaný pre zabezpečenie núdzového zastavenia a spustenia linky, ktoré môže užívateľ realizovať cez cloudovú aplikáciu. Cloudová aplikácia bude takisto realizovaná pomocou Node-RED, ktorý ponúka aj možnosti vizualizácie a realizácie používateľského rozhrania. Node-RED bude teda nasadený aj v cloudovom prostredí Microsoft Azure. Komunikáciu zabezpečí protokol MQTT. Codesys je možné voľne využívať pre akademické účely, obmedzením je však to, že runtime dokáže v voľnom režime bežať nepretržite len 2 hodiny, čiže vo výrobe je bezplatná verzia nepoužiteľná. Pre akademické použitie

však vo väčšina prípadov toto obmedzenie nie je problémom. Táto prípadová štúdia je opísaná s využitím diplomovej práce [66].

Špecifikácia a správanie diskrétneho udalostného systému je rovnaké ako v prvej prípadovej štúdii (kapitola 4.1.1 — obrázok č. 53).

4.2.1 Riadenie diskrétneho udalostného systému

Rovnako ako v prvej prípadovej štúdii aj tu je potrebné riadiť diskrétny udalostný systém. V prvej prípadovej štúdií sa využíval open-source editor a runtime OpenPLC. Tento bol nahradený Codesysom, kde používame programovací jazyk Structured text (tzv. štruktúrovaný text) na deklaráciu premenných. Tento Structured text je porovnateľný s programovacím jazykom C alebo Pascal. Riadiaci program je písaný v rebríkovej schéme podobne ako v prvej edukačnej prípadovej štúdii.

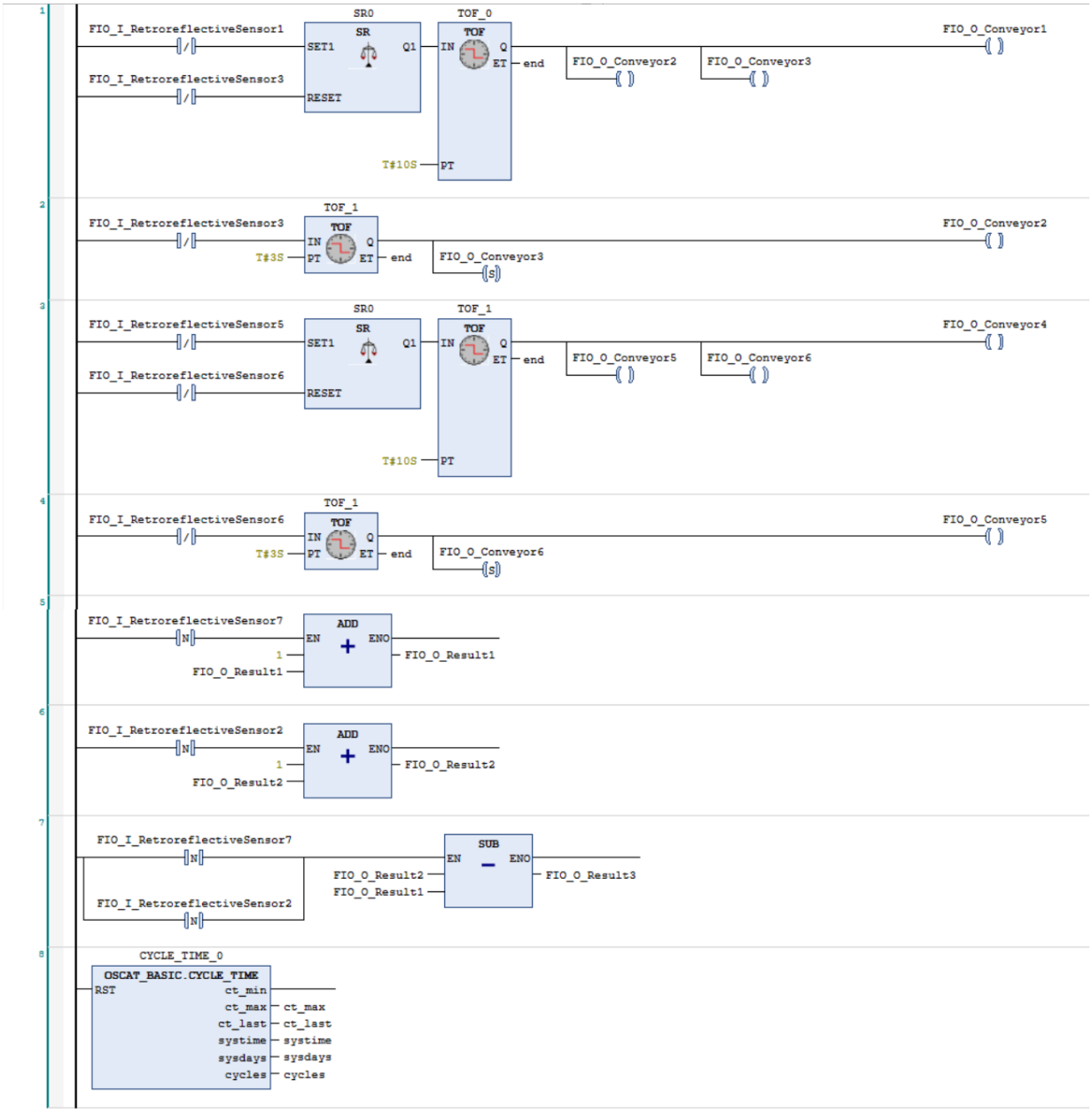
```
1  PROGRAM PLC_PRG
2  VAR
3      FIO_I_RetroreflectiveSensor1    : BOOL := TRUE;
4      FIO_I_RetroreflectiveSensor2    : BOOL := TRUE;
5      FIO_I_RetroreflectiveSensor3    : BOOL := TRUE;
6      FIO_I_RetroreflectiveSensor4    : BOOL := TRUE;
7      FIO_I_RetroreflectiveSensor5    : BOOL := TRUE;
8      FIO_I_RetroreflectiveSensor6    : BOOL := TRUE;
9      FIO_I_RetroreflectiveSensor7    : BOOL := TRUE;
10     FIO_O_Result1                   : INT;
11     FIO_O_Result2                   : INT;
12     FIO_O_Result3                   : INT;
13     FIO_Factoryio_stop              : BOOL;
14     FIO_Factoryio_start             : BOOL := TRUE;
15     FIO_O_Conveyor1                 : BOOL;
16     FIO_O_Conveyor2                 : BOOL;
17     FIO_O_Conveyor3                 : BOOL;
18     FIO_O_Conveyor4                 : BOOL;
19     FIO_O_Conveyor5                 : BOOL;
20     FIO_O_Conveyor6                 : BOOL;
21         TOF_0: TOF;
22         end: TIME;
23         SR0: SR;
24         TOF_1: TOF;
25         CYCLE_TIME_0: OSCAT_BASIC.CYCLE_TIME;
26         ct_max: TIME;
27         ct_last: TIME;
28         systime: TIME;
29         sysdays: INT;
30         cycles: DWORD;
31 END_VAR
```

Obrázok č. 86: Codesys — deklarácia premenných [66]

V hlavnom okne Codesysu v hornej časti máme vstupné pole, kde sa deklarujú premenné v textovej forme. Hlavné premenné v riadiacom programe sú rovnaké ako v prvej prípadovej štúdii (obrázok č. 64) a tieto premenné sa rovnako správajú (kapitola 4.1.3).

Avšak boli pridané aj ďalšie premenné, ktoré súvisia s meraním času behu riadiaceho systému (obrázok č. 86).

V spodnej časti prostredia Codesys tvoríme rebríkový diagram. Diagram sa rovnako tvorí cez panel s nástrojmi. Pre zostavanie programu boli použité rovnaké komponenty ako na obrázkoch č. 65 až 71. Program vyzerá ako na obrázku č. 87.



Obrázok č. 87: Codesys — rebríkový diagram [66]

Navyše oproti prvej edukačnej prípadovej štúdii je prítomný blok CYCLE_TIME, ktorý je typu "Function module". Tento CYCLE_TIME bol vložený za pomoci externej knižnice OSCAT_BASIC verzie 3.3.4.0. Používa sa preto, aby sa mohlo merať, ako dlho riadiaci systém beží. Jeho parametrami sú:

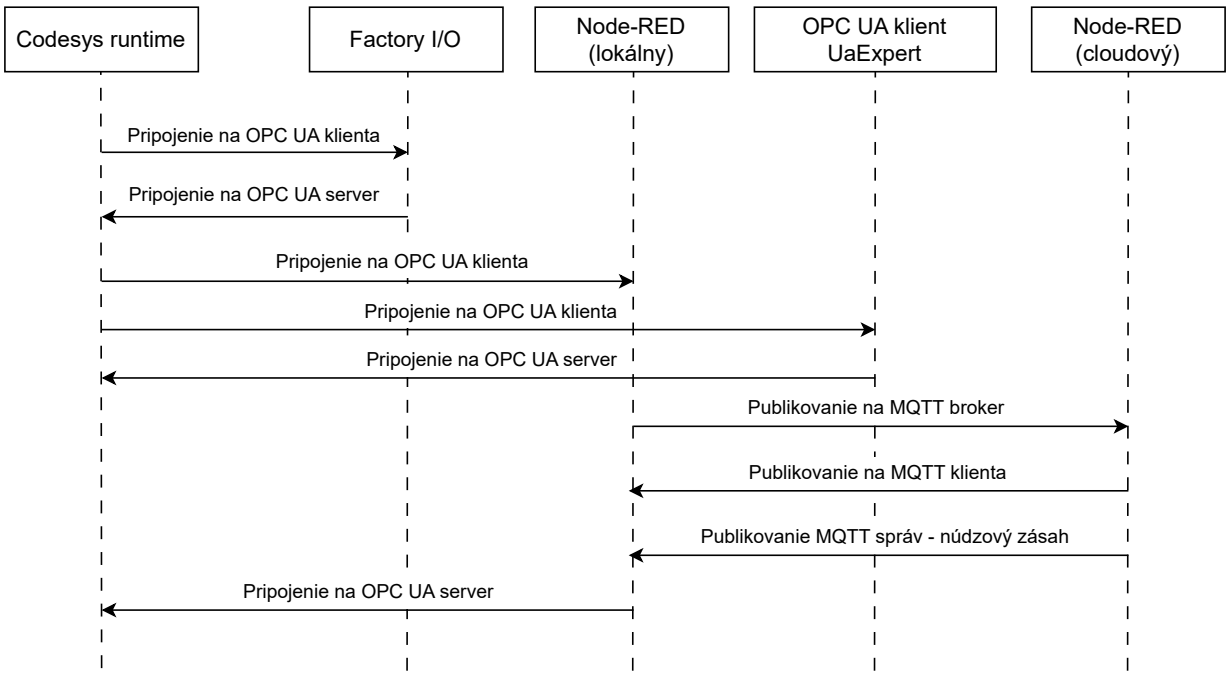
Vstup:

- booleovská premenná Reset.

Výstup:

- *ct_min* typu time — minimálny čas meraného cyklu;
- *ct_max* typu time — maximálny čas meraného cyklu;
- *ct_last* typu time — nedávno nameraný čas cyklu;
- *systime* typu time — trvanie od posledného spustenia;
- *sysdays* typu int — počet dní od posledného začiatku;
- *cycles* typu dword — počet cyklov od posledného spustenia.

4.2.2 Komunikácia jednotlivých subsystémov



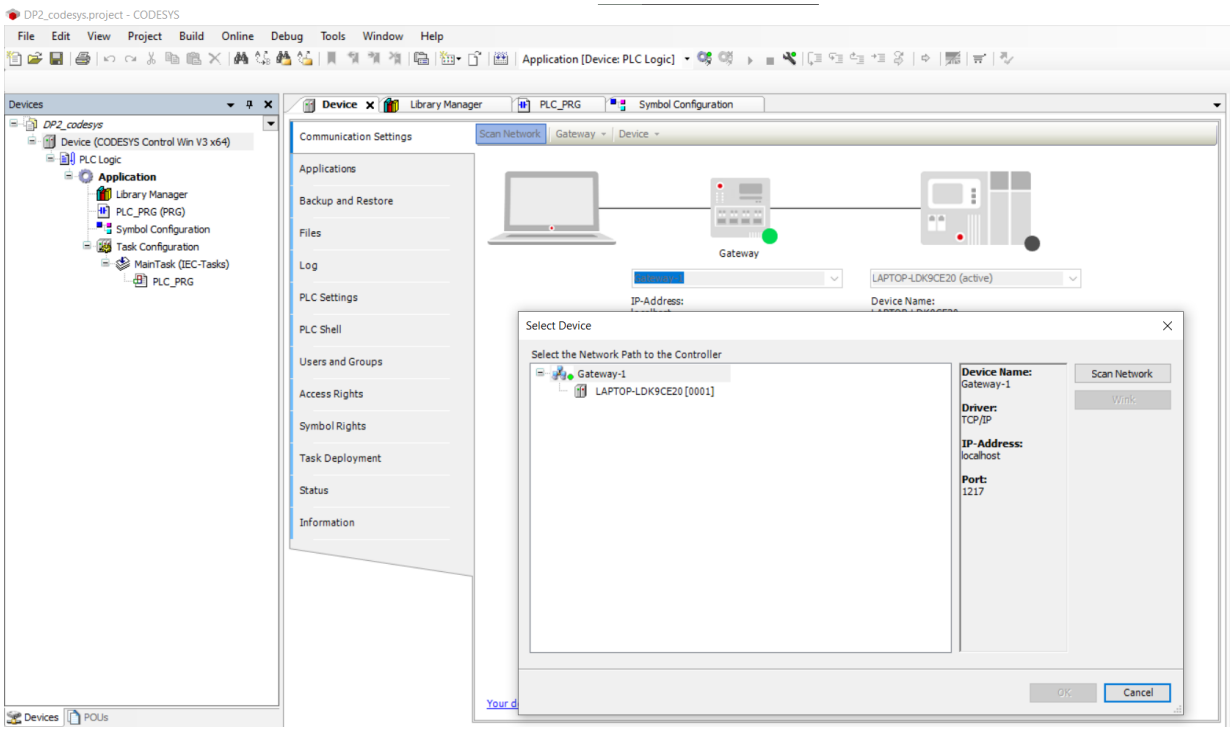
Obrázok č. 88: Sekvenčný diagram pre edukačnú prípadovú štúdiu č. 2 [66]

Najdôležitejší prvok komunikácie je v tomto prípade komunikácia medzi Codesys a Factory I/O, keďže zabezpečuje riadiaci proces. Komunikáciu zabezpečuje moderný protokol OPC UA. Codesys sa teraz bude správať ako OPC UA server a Factory I/O ako OPC UA klient. Vďaka existencii OPC UA servera si je možné premenné prezerať v ľubovoľnom OPC UA klientovi, napríklad v UaExperte. Node-RED budeme opäť

v tomto prípade využívať na zdieľanie dát do cloudovej aplikáciee, ale na samostatné riadenie nie je potrebný. S cloudovou aplikáciou budeme komunikovať cez protokol MQTT.

4.2.3 OPC UA server v Codesys runtime

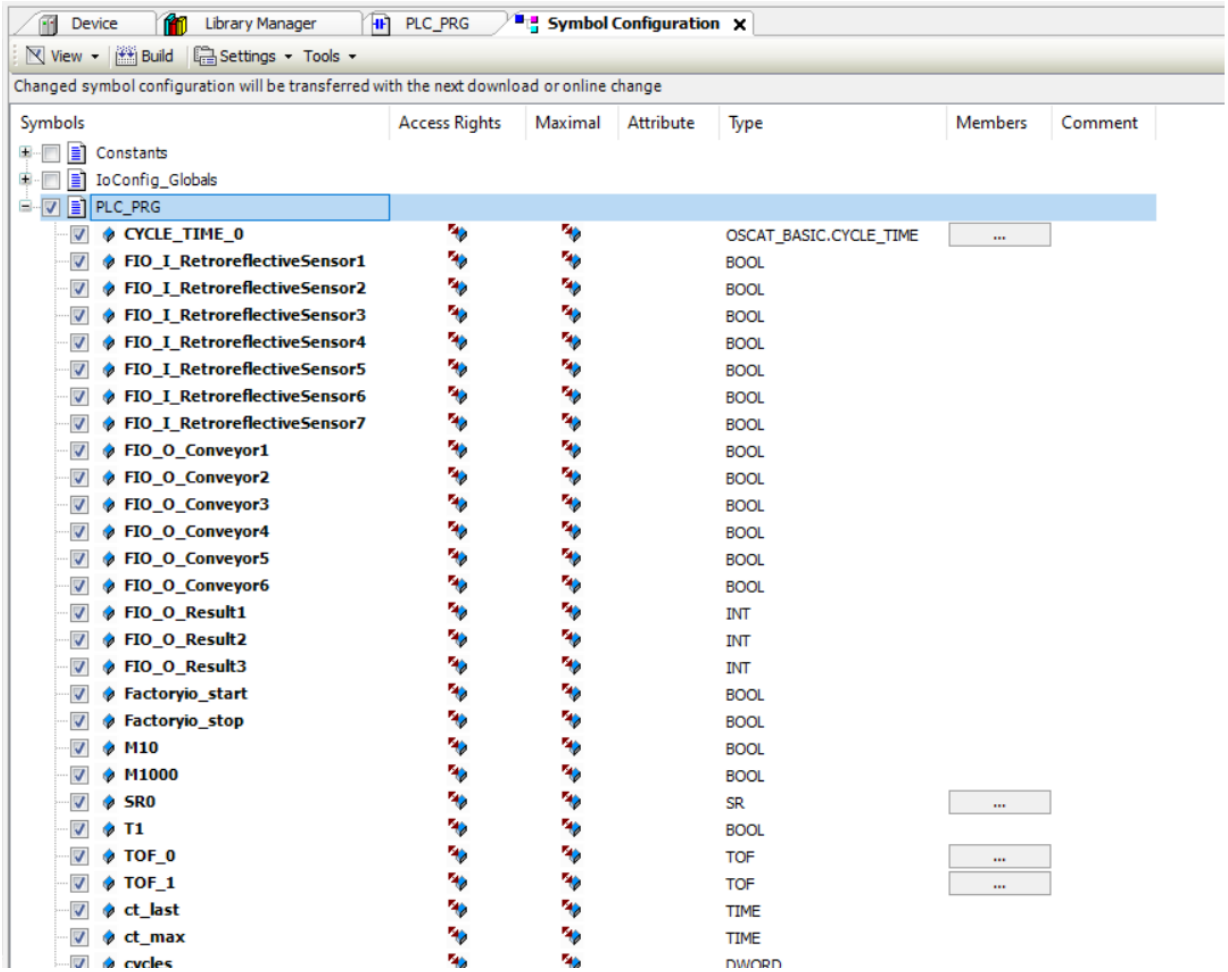
S Codesysom sa pracuje pomerne jednoducho. Je potrebné si uvedomiť, že je žiaduce premenné pomenovať tak, aby ich Factory I/O ľahko identifikoval a vyfiltroval tie, ktoré potrebuje. Celý OPC UA adresný priestor totiž obsahuje aj veľké množstvo rôznych konfiguračných a stavových premenných, ktoré prichádzajú s využitím Codesys runtime. Preto všetky užitočné premenné majú prefix (predponu) *FIO*. Toto sa realizuje preto, aby sme vedeli rozoznať naše (užitočné) premenné a boli ľahšie čitateľné. A tiež preto, aby sme ich vedeli ľahko získať vo Factory I/O. V konfigurácii symbolov (angl. Symbol configuration) Codesys projektu tieto premenné vyberieme, čím vlastne zadáme, že budú ponúkané OPC UA serverom. Následne inicializujeme Codesys Control Win PLC, čo je softPLC bežiace pod Windowsom. Následne v Codesys projekte sa na tento runtime pripojíme a uploadneme (nahráme) doňho program. Pripojenie prebieha cez naskenovanie siete a výber riadiacej jednotky, čo môže byť spomínané softPLC alebo aj bežná hardvérová PLC jednotka (obrázok č. 89).



Obrázok č. 89: Skenovanie siete [66]

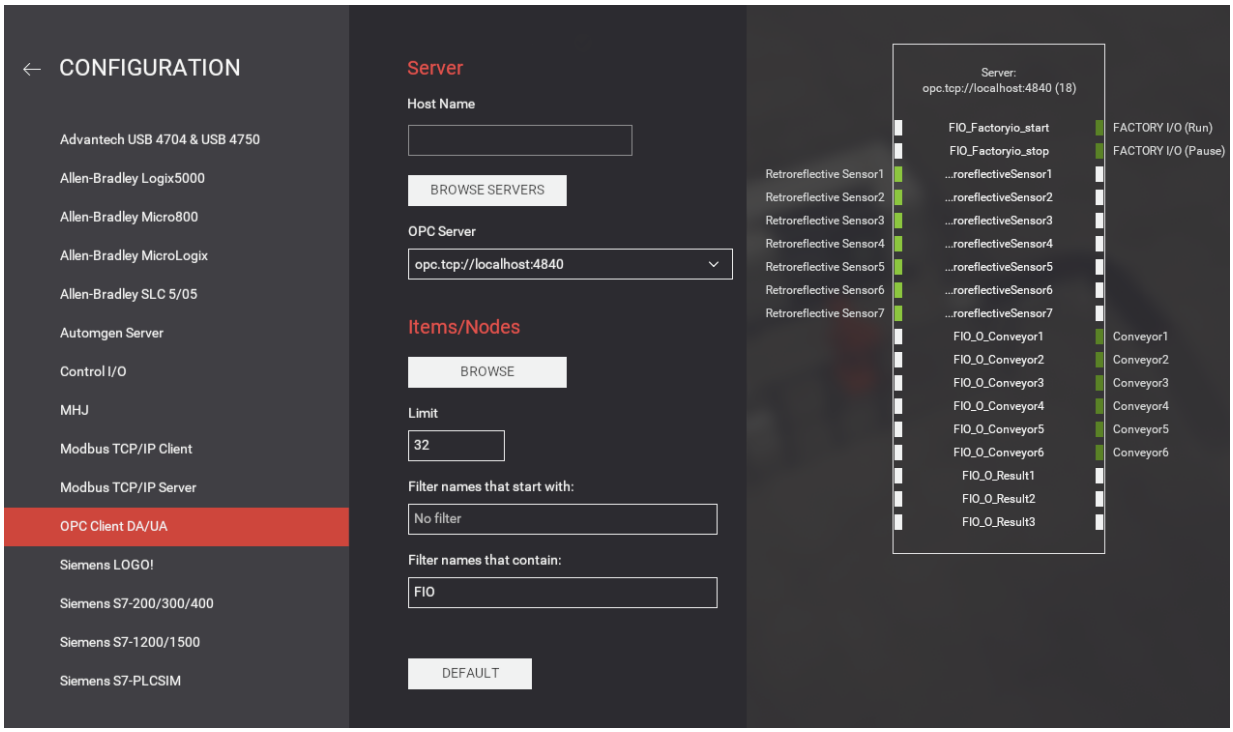
V konfigurácii symbolov môžeme kliknutím vybrať celý náš program PLC_PRG, čím sa označia všetky premenné (obrázok č. 90). Takto sú rovno v OPC UA serveri

dostupné všetky premenné, ktoré využívame. Následne vykonáme build programu a spustíme ho.



Obrázok č. 90: Konfigurácia premenných určených pre OPC UA server [66]

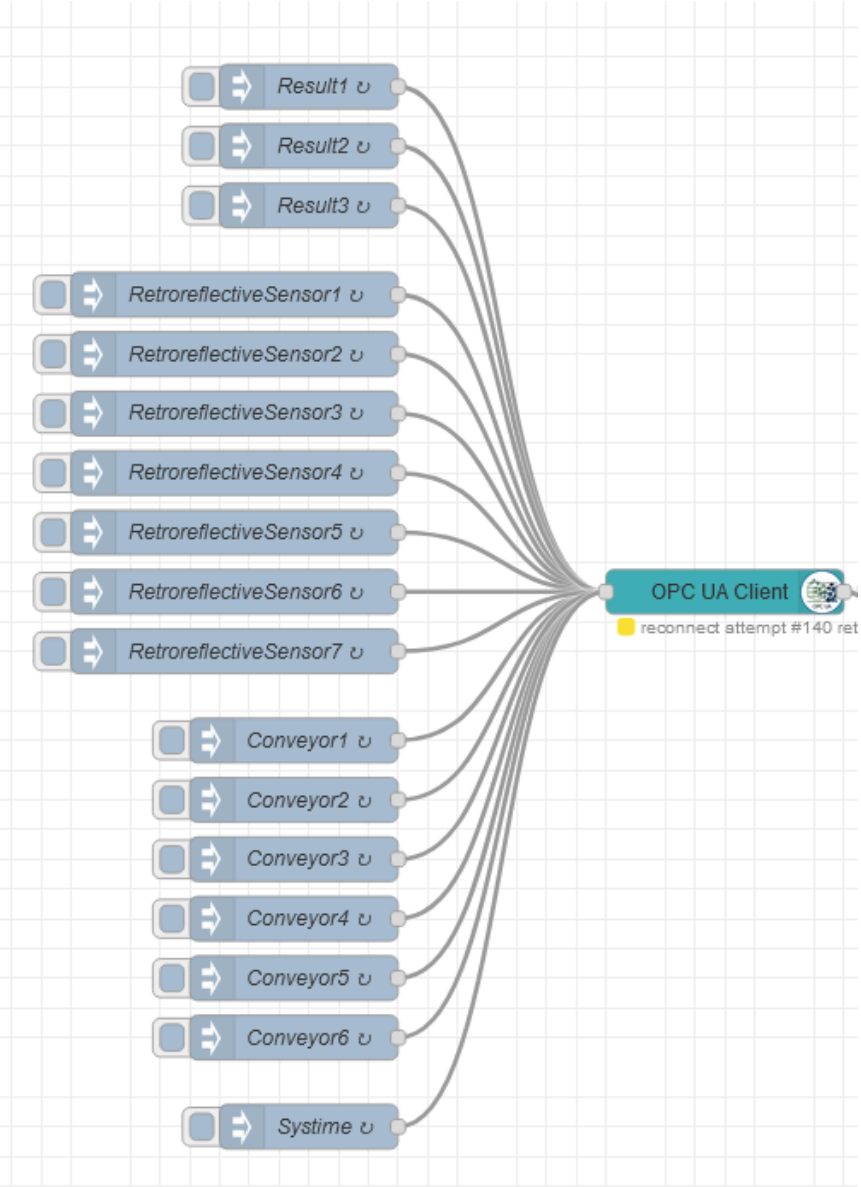
Vo Factory I/O otvoríme našu scénu (obrázok č. 54) a v konfigurácii nastavíme driver OPC Client DA/UA. Zadáme adresu *opc.tcp://localhost:4840* ako server a ako filter premenných si nastavíme "FIO", čím získame užitočné premenné z adresného priestoru OPC UA, ako bolo spomínané. Priradíme všetky premenné a komunikáciu medzi Factory I/O a Codesys máme zrealizovanú (obrázok č. 91.



Obrázok č. 91: OPC UA Client Factory I/O [66]

4.2.4 Prepojenie runtimu Codesys s lokálnym Node-REDom

Po vytvorení a rozbehnutí OPC UA servera pomocou Codesys chceme dáta pomocou protokolu OPC UA dostať do prostredníka Node-RED, ktorý beží na localhoste podobne ako OPC UA server. Túto funkcionalitu bude poskytovať uzol OPC UA Client, kde ako server bude nastavená rovnaká adresa ako v predošlej kapitole (opc.tcp://localhost:4840). Takto Node-RED figuruje ako OPC UA klient. Sú prijímané všetky premenné, s ktorými chceme pracovať (obrázok č. 92). Naľavo v tomto obrázku je možné vidieť uzly typu *Inject*, ktoré sa každú sekundu dopytujú na jednotlivé premenné, pričom názov premennej je definovaný v *msg.topic* danej správy. Názov premennej je v Codesys OPC UA adresnom priestore pomerne komplexný. Názvy a aj samotné dáta si je možné prehľadne prezrieť napríklad v OPC UA klientovi UAExpert. Tam sa je možné dočítať, že ID jednej z premenných je:
ns=4;s=|var|CODESYS Control Win V3 x64.Application.PLC_PRG.FIO_I_RetroreflectiveSensor1
a na základe tohto ID sa je možné v Node-REDe na premennú dopytovať.



Obrázok č. 92: Lokálny Node-RED — OPC UA klient [66]

Je potrebné si objasniť dôvod behu Node-RED na localhoste. Podobne ako v predošlej prípadovej štúdii je cieľom využívať cloud pre monitorovanie diskrétneho udalostného systému a núdzový zásah do neho. Nakoľko OPC UA server beží lokálne a nemáme verejnú (statickú) IP adresu, tak pre odoslanie údajov do cloudového prostredia a prijímanie údajov z cloudového prostredia je využitý lokálny Node-RED.

Do cloudovej aplikácie budeme naše užitočné premenné posielať pomocou protokolu MQTT. Postup je taký, že použijeme uzol *MQTT-in* (funguje ako subscriber) a *MQTT-out* (funguje ako publisher). Tieto uzly sa budú pripájať na broker (server), ktorý je realizovaný v cloudovom prostredí. Konkrétne ide o Aedes MQTT broker. Naše užitočné premenné budeme v lokálnom Node-RED čítať za pomoci OPC UA klienta, z ktorého sú pomocou vlastných funkcií *Filter* vyberané jednotlivé údaje (premenné) a

posielame ich pomocou MQTT do cloudovej aplikácie. Pri posielaní údajov cez MQTT je nutné im nastaviť príslušný *MQTT topic*. Tento vyzerá napríklad takto: *inputs/FIO_I_RetroreflectiveSensor1*.

4.2.5 Node-RED dashboard v cloude Microsoft Azure

V prvej prípadovej štúdii bol dashboard (grafické rozhranie) pre monitorovanie a núdzový zásah do systému realizovaný pomocou aPaaS služby Azure IoT Central. V druhej prípadovej štúdii bol zvolený iný prístup. Aj Node-RED poskytuje možnosť realizácie grafického rozhrania, preto sa ako vhodná možnosť ukázala nasadiť Node-RED aj do cloudu a pomocou neho vytvoriť grafické rozhranie .

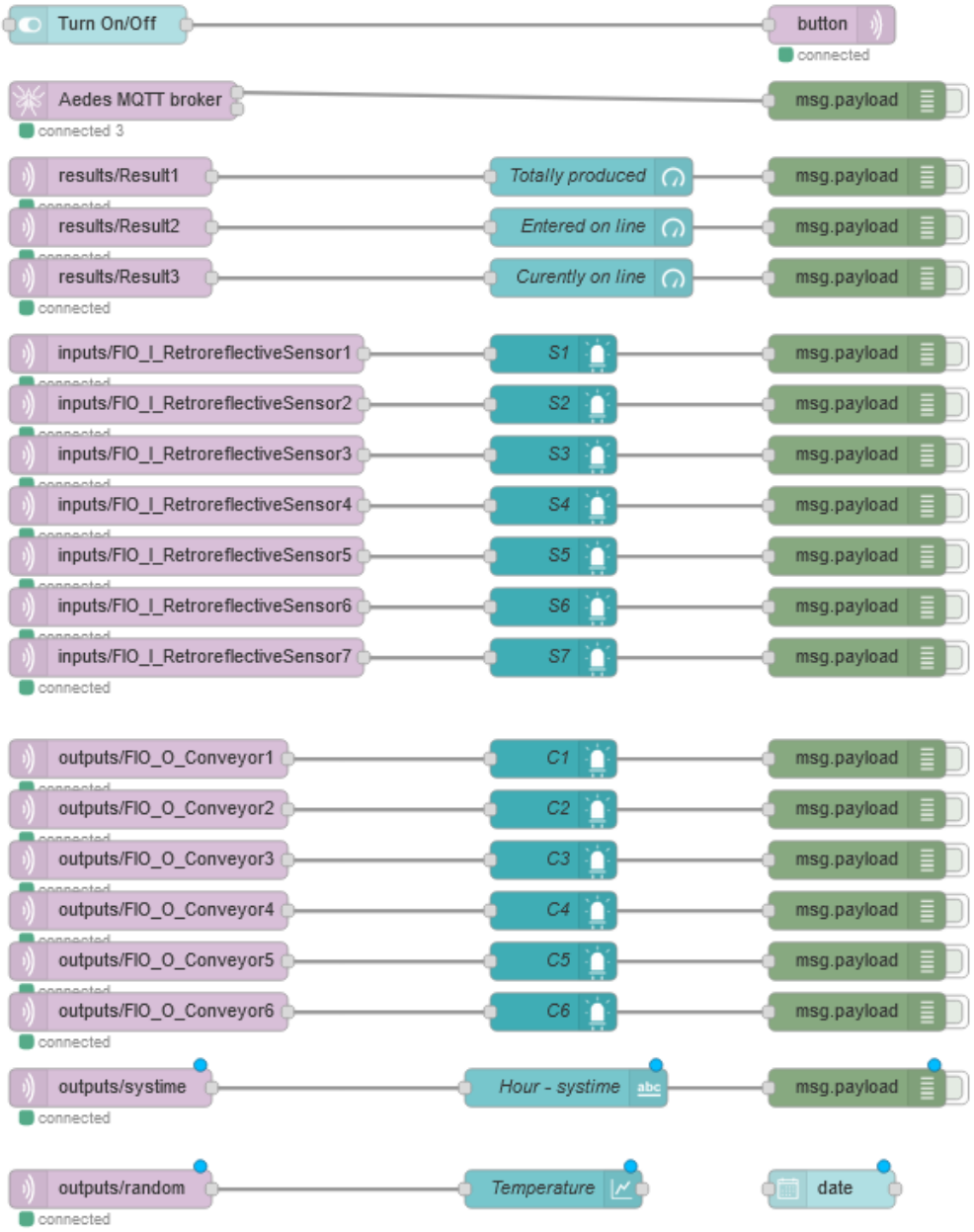
Nasadenie Node-RED do cloudového prostredia je pomerne jednoduché, nakoľko ide vlastne o aplikáciu bežiacu na runtime Node.JS. Využitý bol teda virtuálny stroj, konkrétne IaaS službu Azure Virtual Machine. Azure Virtual Machine je jedným z niekoľkých typov škálovateľných výpočtových zdrojov, ktoré Azure ponúka. Predtým ako ho vytvoríme, je potrebné definovať niekoľko záležitostí. Ide napríklad o názov aplikácie, miesto, kde budú zdroje uložené, veľkosť virtuálneho stroja, operačnej pamäte, maximálny počet virtuálnych počítačov, operačný systém, konfiguráciu a súvisiace zdroje. Ďalej poskytuje flexibilitu virtualizácie bez toho, aby sme boli nútení kupovať a udržiavať fyzický hardvér, na ktorom je spustený.

Vybraný bol virtuálny stroj triedy *Standard_B1s*. Poskytuje 1 GB operačnej pamäte. Využili sme operačný systém na báze Linuxu — Ubuntu Server 20.04 (Focal). Inštalácia Node-RED prebehla bez akýchkoľvek problémov.

Dáta teda posielame z lokálneho Node-REDu do Node-REDu v cloudovom prostredí pomocou komunikačného protokolu MQTT. V cloudovom Node-REDe dáta čítame, čiže prijímame pomocou protokolu MQTT, a následne ich zobrazujeme v grafickom rozhraní. Na zobrazenie dát v grafickom rozhraní využívame knižnicu *node-red-dashboard 3.1.6* a *node-red-contrib-ui-led 0.4.11*, nakoľko uzly, ktoré nám to umožňujú, nie sú súčasťou základnej inštalácie Node-REDu a je potrebné ich nainštalovať samostatne. Tabuľka č. 2 bližšie opisuje obrázok č. 93, kde za pomoci týchto uzlov je možné zobrazovať dáta v grafickom rozhraní.

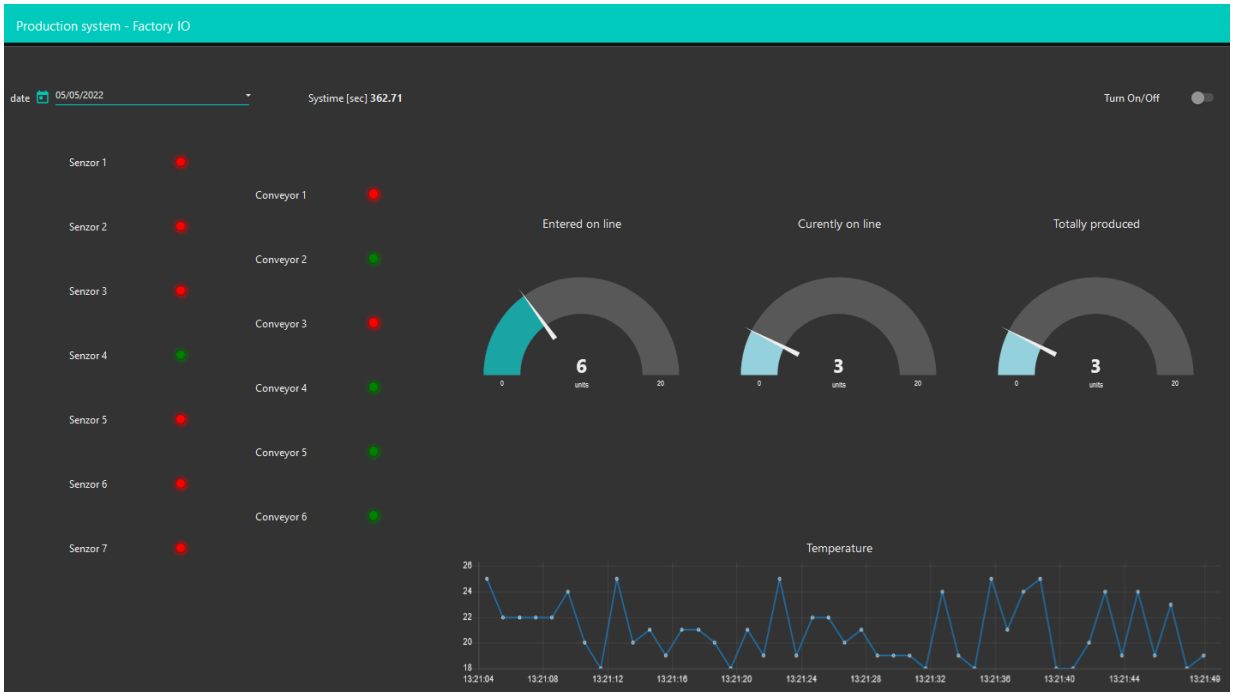
Tabuľka č. 2: Premenné v lokálnom Node-REDe [66]

Názov uzla	Nami pomenovaný uzol
led	S1-S7 , C1-C6
gauge	Totally produced, Entered on line, Curently on line
text	Systime
chart	Temperature
button	Turn On/Off
date picker	date



Obrázok č. 93: Node-RED v cloude [66]

Naše výsledné grafické rozhranie (dashboard) je vidieť na obrázku č. 94). Je dôležité uviesť, že premenné je možné nielen monitorovať, ale opäť je možné aj núdzovo zasahovať do výrobného procesu.



Obrázok č. 94: Dashboard v Node-RED [66]

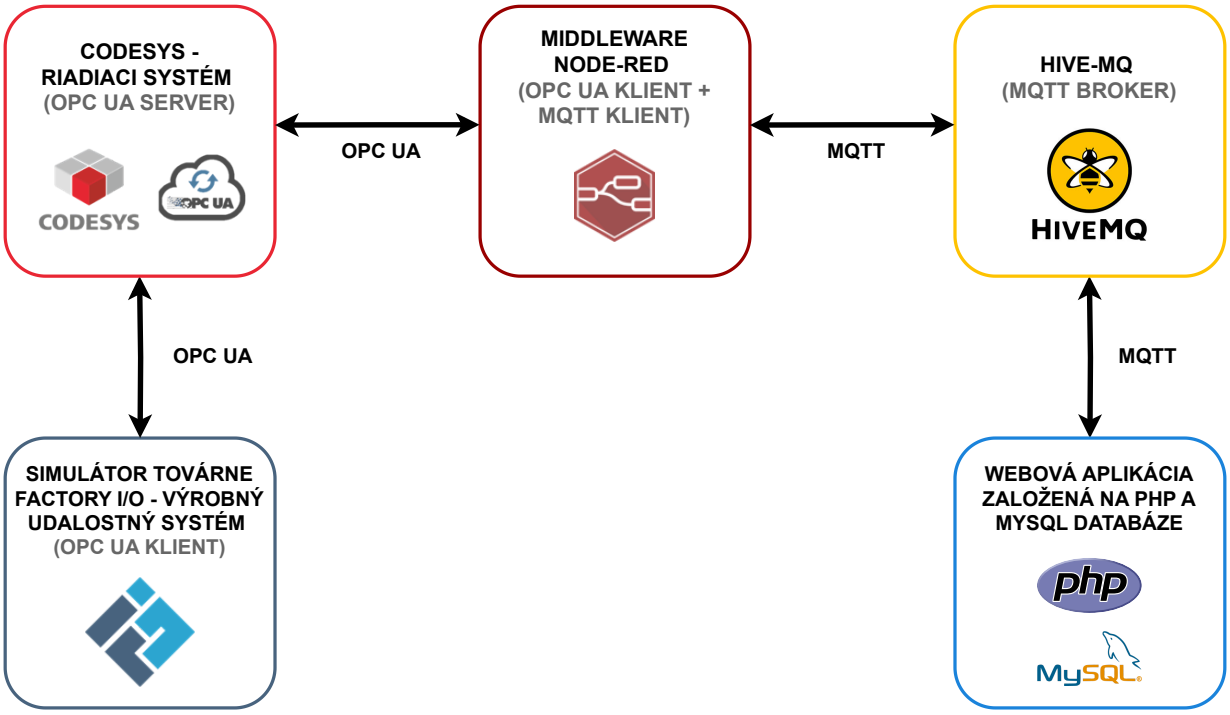
Videá k tejto edukačnej prípadovej štúdii je možné nájsť na tejto adrese:

- `http://bit.ly/3OhmDyU`

Ďalšie informácie, návody a programové projekty je možné nájsť na e-learningovej webovej stránke `https://elearning.mechatronika.cool/?p=8344` .

4.3 Tretia prípadová štúdia: Prepojenie PLC s webovou aplikáciou a databázou

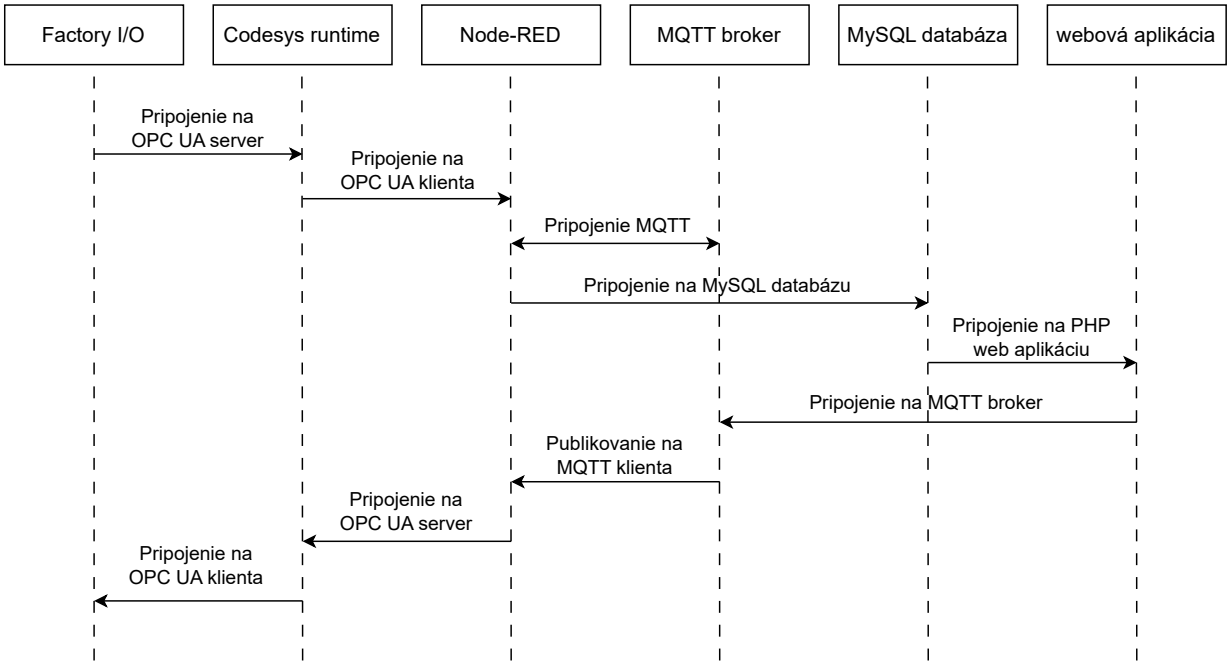
Aj cieľ tretej prípadovej štúdie bol z oblasti digitalizácie diskrétnych udalostných systémov a takisto bola vytýčená úloha návrhu riadenia pre virtuálny model výrobného systému realizovaného vo Factory I/O. Model je riadený prostredníctvom PLC enginu Codesys, pričom monitorovanie udalostného systému je zabezpečené webovou aplikáciou s príslušnou databázou. Pomocou webovej aplikácie je možné do systému aj zasahovať (napr. núdzové zastavenie). Táto prípadová štúdia je opísaná s využitím diplomovej práce [98].



Obrázok č. 95: Schéma komunikácie jednotlivých častí prípadovej štúdie

V prípadovej štúdii bol prepojený Codesys a simulátor Factory I/O pomocou moderného komunikačného protokolu OPC UA. Codesys, ktorý zabezpečoval riadenie virtuálnej továrne, slúžil na naprogramovanie riadenia, ako PLC engine a tiež ako OPC UA server. Vo Factory I/O bol navrhnutý model diskrétneho udalostného systému. Konkrétne išlo o linku, ktorá triedila prichádzajúci materiál na základe nastavených kritérií a následne ho spracoval v obrábacom centre. Factory I/O figuroval ako OPC UA klient. Dáta z riadenia boli posielané do prostredníka Node-RED, ktorý tiež figuroval ako OPC UA klient. Node-RED zachytával dáta z OPC UA servera, spracoval ich a zaslal s konkrétnou témou na MQTT broker pomocou komunikačného protokolu MQTT. Na MQTT broker bola zaslaná požiadavka na prijímanie dát s našou témou. Prijaté dáta boli následne uložené do databázy, odkiaľ sú dopytované pomocou SQL dopytov v pro-

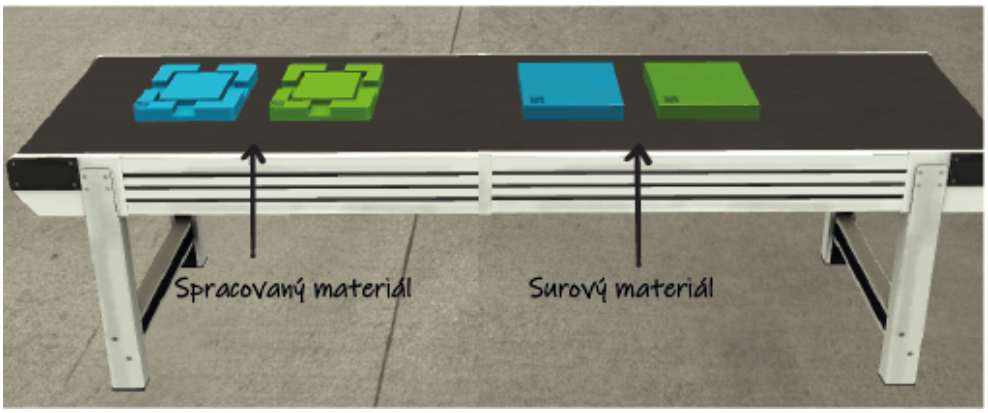
gramovacom jazyku PHP a zobrazené v PHP webovej aplikácii.



Obrázok č. 96: Priebeh komunikácie jednotlivých častí prípadovej štúdie

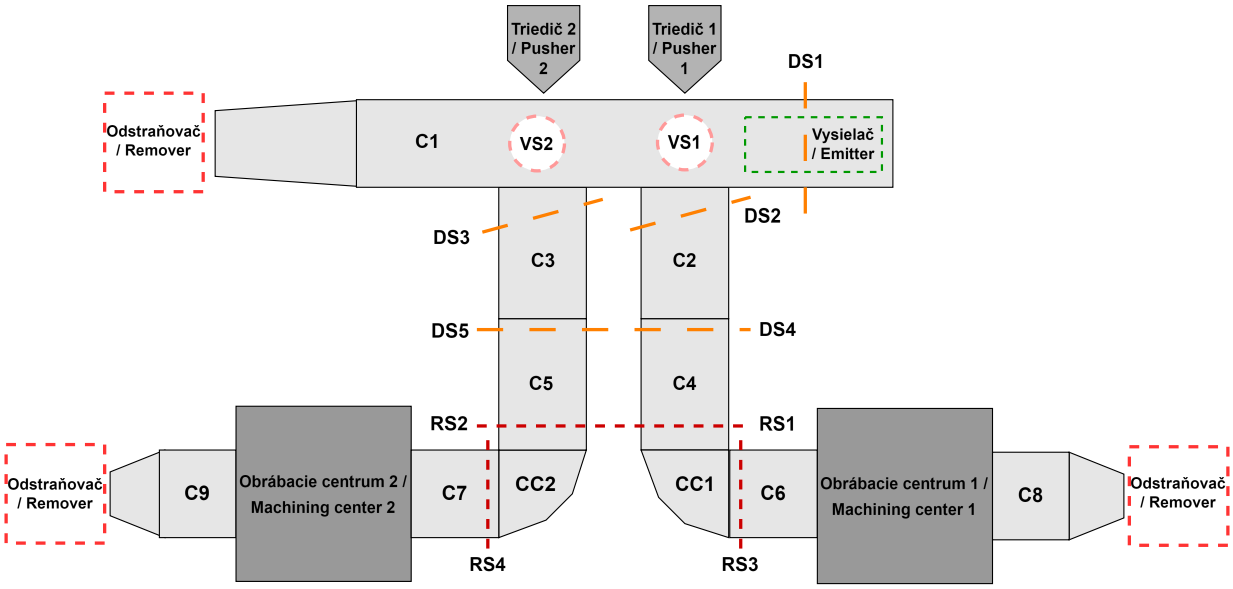
4.3.1 Špecifikácia a správanie diskrétneho udalostného systému

V prvom kroku bolo potrebné vo Factory I/O navrhnúť vhodný virtuálny model diskrétneho udalostného systému vo forme triediacej linky. Úlohou navrhnutého systému je vygenerovať dva rôzne materiály (teda polovýrobky), tieto roztriediť a poslať do správnej vetvy systému. Následne správne rozdelené materiály dopraviť k obrábaciemu centru, kde je materiál opracovaný a poslaný ďalej na expedíciu.



Obrázok č. 97: Ukážka materiálov [98]

Schému diskrétneho udalostného systému môžeme vidieť na obrázku č. 98.



Obrázok č. 98: Schéma udalostného systému

Legenda značiek z obrázka č. 98:

- C1-C9 -> dopravníky;
- CC1, CC2 -> zakrivené dopravníky;
- VS1, VS2 -> vizuálne senzory;
- DS1-DS5 -> difúzne senzory;
- RS1-RS3 -> retroreflektívne senzory;
- Pusher1, Pusher2 -> mechanické triediče;
- Machining Centre1, Machining Centre2 -> obrábacie centrá.

4.3.2 Komponenty použité na tvorbu modelu linky

Najprv je potrebné uviesť charakteristiku prvkov Factory I/O, ktoré sú v tejto prípadovej štúdii nové a neboli využité v prvej a druhej prípadovej štúdii.

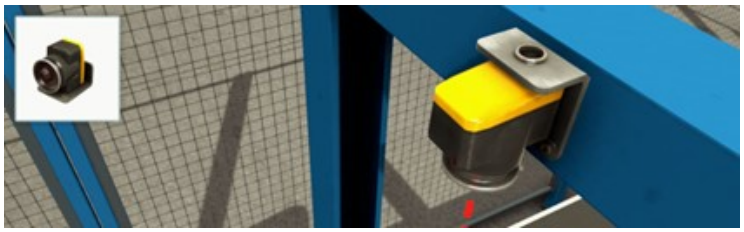
1. **Triedič (angl. Pusher)** — Pneumatický triedič v tvare piestu vybavený dvoma jazýčkovými snímačmi, ktoré indikujú prednú a zadnú hranicu. Obsahuje aj servoventil, ktorý je možné využiť na nastavenie polohy piestu. Ovládanie sa môže vykonávať pomocou digitálnych alebo analógových hodnôt podľa zvolenej konfigurácie.

Obrázok č. 99: Pneumatický triedič [98]

2. **Difúzny senzor (angl. Diffuse sensor)** — Difúzny fotoelektrický senzor, ktorý dokáže rozpoznať akýkoľvek pevný objekt.

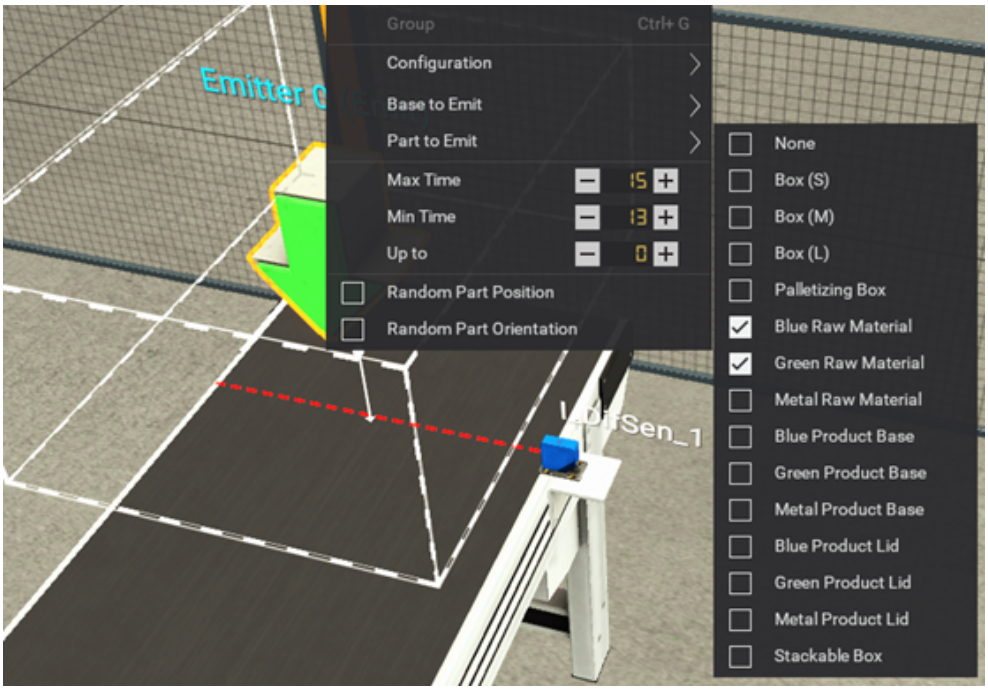
Obrázok č. 100: Difúzny senzor [98]

3. **Senzor videnia / vizuálny senzor (angl. Vision sensor)** — Vizuálny senzor rozpoznáva suroviny, viečka výrobkov a základne výrobkov a ich príslušné farby.



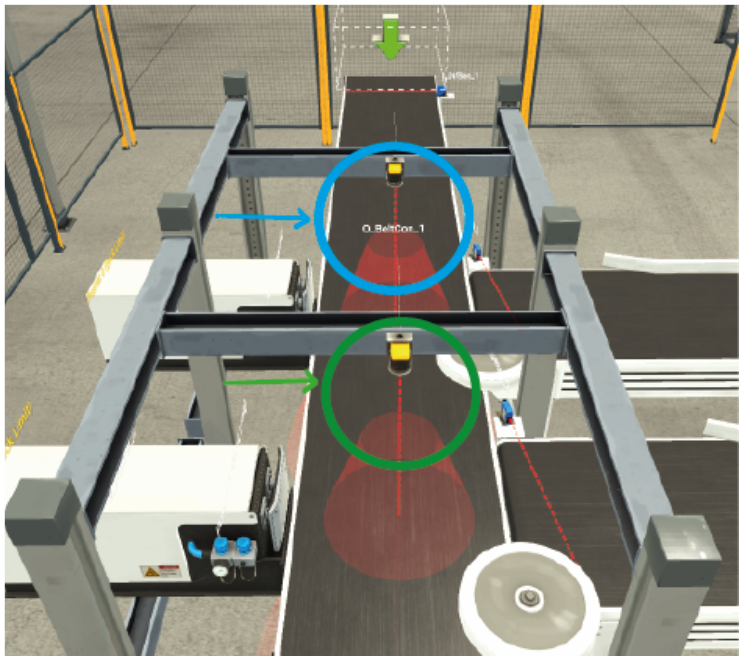
Obrázok č. 101: Vizuálny senzor [98]

Najprv boli vytvorené systémy pásových dopravníkov (C1-C9, CC1-CC2), po ktorých sa materiály budú prepravovať. Ďalej boli využité obrábacie centrá (*Machining centre* na obrázku č. 98) v oboch vetvách systému. Na generovanie materiálov sme použili vysielač (*vysielač / emitter* na obrázku č. 98). Na obrázku č. 102 je možné vidieť nastavenia, ktoré sme zvolili pre vysielač. Nastavené bolo generovanie dvoch typov materiálov – *Green Raw Material* a *Blue Raw Material* a tiež interval generovania. Pomocou *Max Time* a *Min Time* bolo určené, aby sa materiály generovali každých 13-15 sekúnd. *Up to* určuje, koľko materiálov má byť vygenerovaných. My sme nechali hodnotu 0, čo znamená, že nebola určená žiadna hranica.



Obrázok č. 102: Nastavenia pre vysielač (emitter) [98]

Ďalej bol navrhnutý subsystém na triedenie materiálu. To bude zabezpečovať obrazový senzor a triediť sa bude podľa farby neopracovaného materiálu. Každému senzoru sme nastavili, na ktorý druh materiálu má reagovať (obrázok č. 103). Po detekcii materiálu sa spustí pneumatický triedič (*pusher* na obrázku č. 98), ktorý zatlačí materiál na správny dopravníkový pás a ten ho dopraví k obrábaciemu centru.



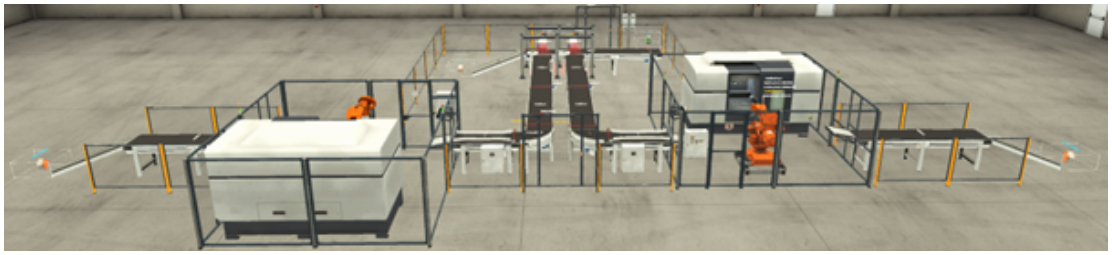
Obrázok č. 103: Senzory videnia [98]

Následne sme virtuálny systém doplnili o retroreflexné senzory a odrazky (podobne ako v prvej štúdii — obrázok č. 60). Tieto senzory pomáhajú riadiť chod systému. Vďaka nim vieme detegovať, kde sa materiál nachádza a spustiť dopravníky podľa potreby.

Okrem riadenia bolo potrebné realizovať aj zobrazovanie údaja o počte materiálov v systéme. Na zobrazovanie týchto údajov nám slúžia displeje priamo vo virtuálnom modeli. Jeden je umiestnený na začiatku systému, zvyšné dva pri obrábacích centrách.

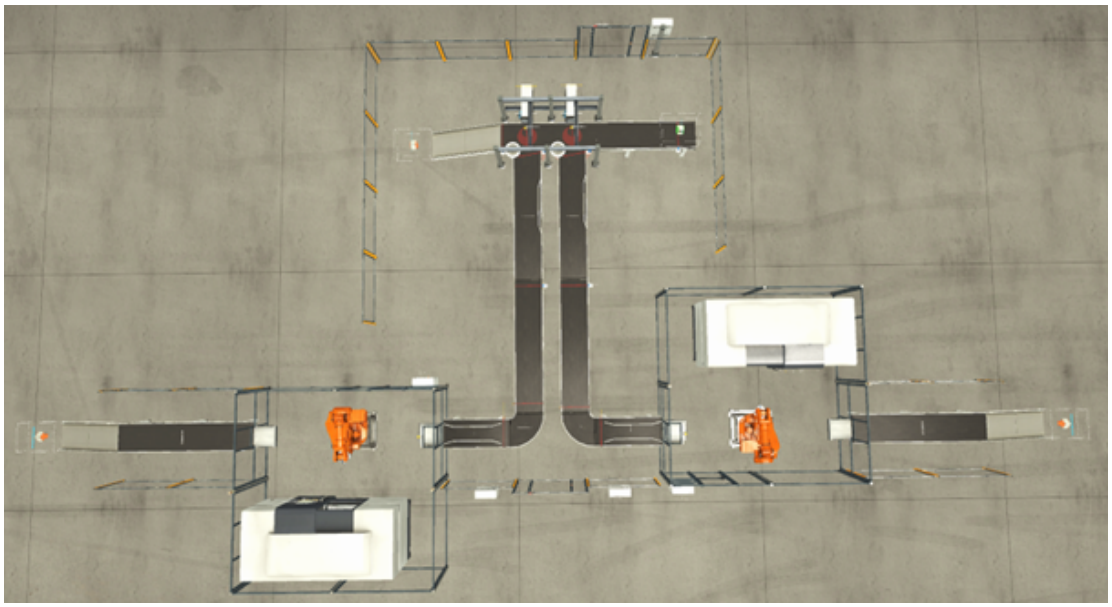
Na záver každej vetvy bol umiestnený odstraňovač (*remover* na obrázku č. 98), ktorý zabezpečí odstránenie spracovaných materiálov (výsledných produktov), ktoré vyjdú z obrábacieho centra. Týmto modelujeme expedíciu tovaru zákazníkom.

Na nasledujúcich obrázkoch č. 104 — 106 sú zobrazené rôzne pohľady na vytvorený kompletný model virtuálneho diskrétneho udalostného systému vo Factory I/O.



Obrázok č. 104: Pohľad na virtuálny výrobný systém (uhol č. 1) [98]

Obrázok č. 105: Pohľad na virtuálny výrobný systém (uhol č. 2) [98]



Obrázok č. 106: Pohľad na virtuálny výrobný systém (uhol č. 3) [98]

4.3.3 Riadenie diskrétneho udalostného systému

Riadenie udalostného systému je navrhnuté v softvéri Codesys. Pre riadiaci program sme si zvolili PLC jazyk zvaný štruktúrovaný text (angl. Structured text). Pomocou tohto jazyka bol vytvorený riadiaci program, ktorý zabezpečuje chod celého virtuálneho výrobného systému.

Riadenie začína spustením systému a vygenerovaním materiálu vysielačom. Akonáhle difúzny senzor umiestnený priamo pod vysielačom deteguje materiál, dopravníkový pás (C1) sa pohne a počet vygenerovaných materiálov sa zvýši o hodnotu 1. Materiál putuje dopravníkovým pásom k obrazovému senzoru. V prípade, že senzor zaznamenal správny materiál, aktivuje sa mechanický triedič, ktorý posunie materiál na nasledujúci správny dopravníkový pás (C2 alebo C3). Prvý dopravník (C1) sa vypne a počet materiálov danej farby v systéme sa zvýši o 1.

V prípade, že by vysielač vygeneroval iný materiál, ako je určený pre túto časť linky, je popod obrazové senzory priamo dopravený na kĺzačku a pomocou nej k odstraňovaču.

Sieť dopravníkov (C2-C9, CC1, CC2), ktorá začína po dopravení materiálu triedičom, je riadená s využitím senzorov a informácií, či je obrábacie centrum práve vyťažené alebo čaká na materiál. Obrábacie centrum je počas celej simulácie zapnuté a čaká na informáciu, či bol materiál doručený. Na toto slúži obrazový senzor, ktorý je súčasťou obrábacieho centra.

Premenné a funkcie programu boli deklarované v hornej časti súboru PLC_PRG. Pri premenných si je dôležité uvedomiť, že premenné budú posielané do Factory I/O, preto ich treba dôkladne pomenovať, aby ich vedel ľahko identifikovať a filtrovať. Typ premennej v Codesyse priradíme pomocou „:“.

Syntax: *nazov_premennej : typ_premennej*.

Každej premennej bol zadefinovaný typ, prípadne aj hodnota. V riadiacom programe využívame premenné typu BOOL, UINT a funkciu CTU.

Dátový typ premennej BOOL pozná iba dve hodnoty — 0 a 1, prípadne TRUE a FALSE. Dátový typ UINT je neznamienkový integer, čo znamená, že je to premenná, do ktorej vieme uložiť hodnoty od 0 po 65 535.

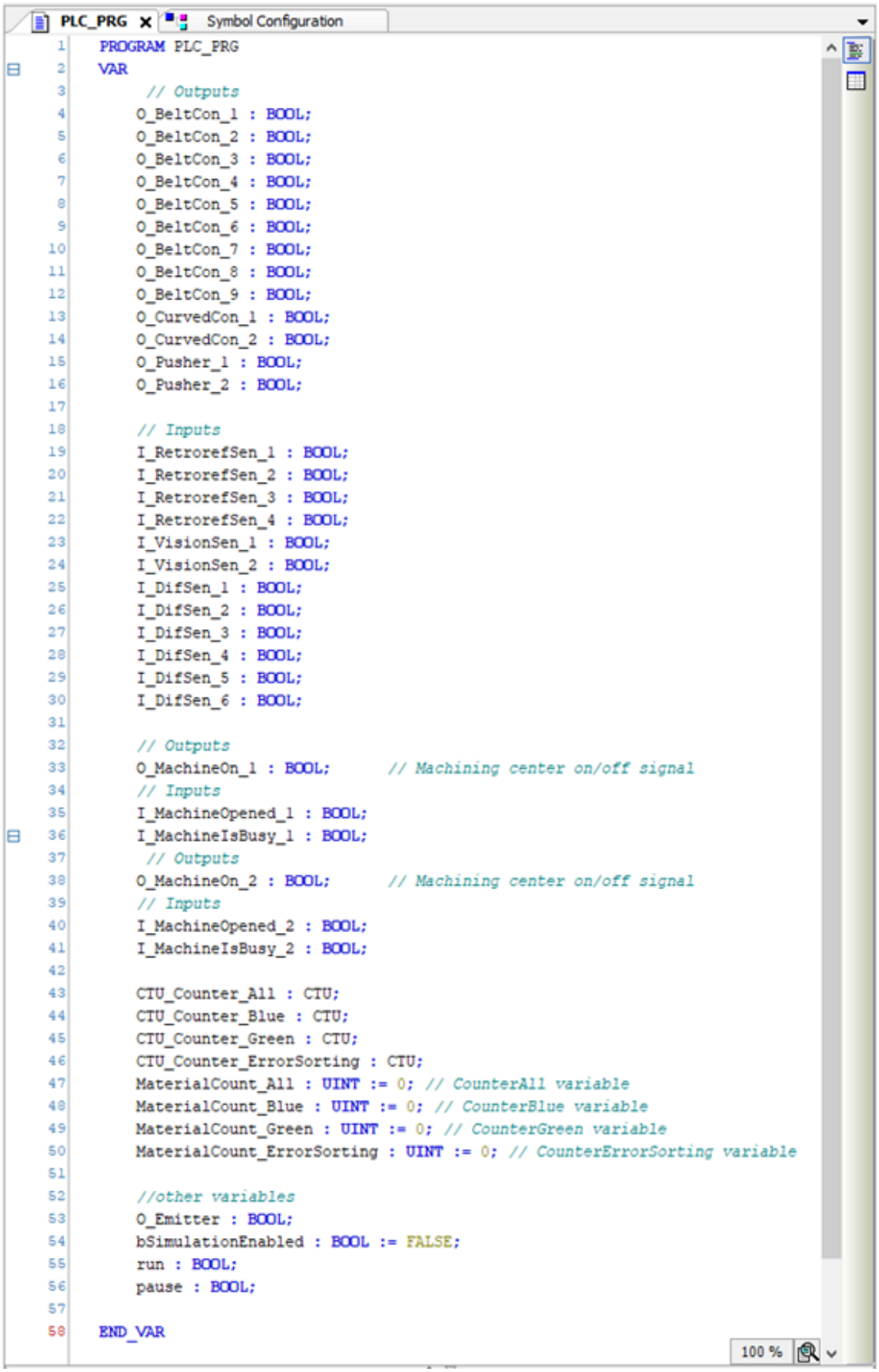
Funkciu CTU poskytuje knižnica Standard Library. Táto funkcia zabezpečuje inkrementáciu. Vstupmi funkcie sú CU, RESET a PV. Výstupmi funkcie sú Q a CV.
Vstupy funkcie CTU:

- CU je typu BOOL, v prípade TRUE zvýši hodnotu uloženú v CV o 1.
- RESET je typu BOOL, v prípade TRUE sa vyresetuje hodnota uložená v CV.
- PV je typu WORD (rovnaká veľkosť ako UINT), určuje hornú hranicu pre inkrementáciu CV.

Výstupy funkcie CTU:

- Q je typu BOOL, defaultne FALSE, na TRUE sa nastaví, keď CV >= PV.
- CV je typu WORD, ukladá sa tu hodnota po inkrementovaní.

Na obrázku č. 107 môžeme vidieť premenné, ktoré sme použili pri písaní riadiaceho programu. Na riadkoch 47-50 a 54 sa nachádzajú nielen zadefinované typy premenných ale aj priradené hodnoty danej premennej.



Obrázok č. 107: Premenné riadiaceho programu [98]

RIADIACI PROGRAM REALIZOVANÝ V CODESYSE

Riadiaci program slúži na riadenie chodu virtuálneho modelu linky. Na tvorbu programu bol zvolený štruktúrovaný text. Na riadenie využívame funkcie, podmienky a cykly.

Hodnotu do premennej typu BOOL je možné zapísať dvomi spôsobmi, prvým je priradenie a druhým spôsobom je využitie SET a RESET. Na obrázku č. 108 môžeme vidieť obidva spôsoby zapísania premennej.

```
10  O_BeltCon_8 S= TRUE;
11  O_BeltCon_9 S= TRUE;
12  O_Pusher_1 := FALSE;
13  O_Pusher_2 := FALSE;
```

Obrázok č. 108: Ukážka zápisu hodnoty do premennej [98]

Rozdiel medzi týmito zápismi je v tom, že pri využití „:=“ bude hodnota priradená len na dobu trvania podmienky. Pri použití „S=“ sa nastaví hodnota na dobu, pokiaľ túto hodnotu neupravíme pomocou „R=“. Na obrázku č. 109 je možné vidieť ukážku prepísania hodnoty pomocou SET a RESET. Útržok kódu slúži na riadenie dopravníka. Ak je premenná *O_Pusher_1* nastavená na hodnotu TRUE, tak sa premenná *O_Belt_Con_2* nastaví na hodnotu TRUE. Toto nastavenie trvá, pokiaľ nie je premenná *I_DifSen_4* nastavená na TRUE. Ak je hodnota premennej *I_DifSen_4* TRUE, tak sa hodnota premennej *O_BeltCon_2* zmení na FALSE. Toto nastavenie trvá, pokiaľ nie je znovu splnená prvá podmienka. (*Riadime sa pravdivostnou tabuľkou pre SET a RESET.*)

```
38  //run and stop O_BeltCon_2
39  IF O_Pusher_1 THEN
40      O_BeltCon_2 S= TRUE;
41  END_IF
42  IF I_DifSen_4 THEN
43      O_BeltCon_2 R= TRUE;
44  END_IF
```

Obrázok č. 109: Ukážka zápisu hodnoty do premennej [98]

Na nasledujúcom obrázku č. 110 je možné vidieť ukážku kódu, ktorý slúži na riadenie prvého dopravníka a následne na triedenie materiálu podľa farby.

```
13  //run O_BeltCon_1
14  IF I_DifSen_1 THEN
15      O_BeltCon_1 S=TRUE;
16  END_IF
17
18  //stop O_BeltCon_1
19  IF I_VisionSen_1 OR I_VisionSen_2 THEN
20      O_BeltCon_1 R= TRUE;
21  END_IF
22
23  //sorted blue
24  IF I_VisionSen_1 THEN
25      O_Pusher_1 := TRUE;
26  END_IF
27
28  //sorted green
29  IF I_VisionSen_2 THEN
30      O_Pusher_2 := TRUE;
31  END_IF
```

Obrázok č. 110: Ukážka kódu na riadenie dopravníka a triedenie [98]

Obrázok č. 111 je ukážkou kódu, ktorý slúži na riadenie dopravníkového systému na spracovanie modrého materiálu. Kód riadi dopravníky od triediča až po obrábacie centrum. Na riadenie dopravníkového systému so zeleným materiálom sme použili totožný kód s upravenými názvami premenných podľa potreby.

```
33  //MACHINING FOR BLUE BRANCH
34  //run and stop O_BeltCon_2
35  IF O_Pusher_1 THEN
36      O_BeltCon_2 S= TRUE;
37  END_IF
38
39  //run O_BeltCon_4
40  IF I_DifSen_2 THEN
41      O_BeltCon_4 S=TRUE;
42      O_BeltCon_2 S= TRUE;
43  END_IF
44
45  IF I_DifSen_4 THEN
46      O_BeltCon_2 R= TRUE;
47  END_IF
48
49  IF NOT I_RetrorefSen_1 AND I_MachineOpened_1 AND NOT I_MachineIsBusy_1 THEN
50      O_CurvedCon_1 S= TRUE;
51      O_BeltCon_6 S= TRUE;
52      O_BeltCon_4 S=TRUE;
53  END_IF
54
55  IF NOT I_RetrorefSen_1 AND I_MachineIsBusy_1 THEN
56      O_CurvedCon_1 R= TRUE;
57      O_BeltCon_6 R= TRUE;
58      O_BeltCon_4 R=TRUE;
59  END_IF
60
```

Obrázok č. 111: Riadiaci kód pre modrú vetvu [98]

Obrázok č. 112 obsahuje kód, ktorý sme použili na zaznamenávanie počtu mate-

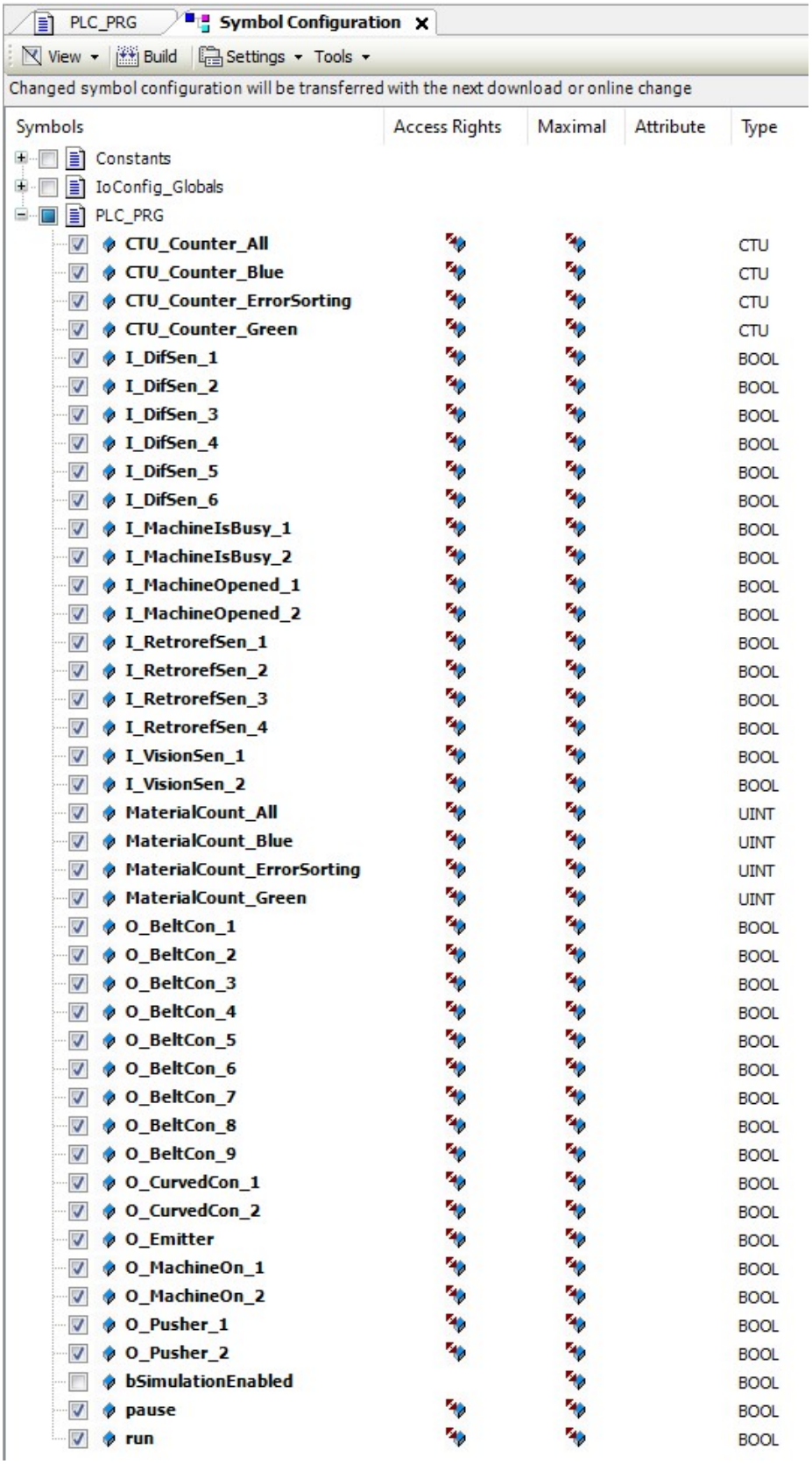
riálov v danom systéme. Hodnota je inkrementovaná pri zmene hodnoty konkrétneho senzora. Funkcia sleduje nábežnú hranu signálu, pretože pokiaľ by sa hodnota inkrementovala počas celej doby, kedy senzor deteguje materiál a má zmenenú hodnotu, tak by sa konečná hodnota nezvýšila o 1 ale o viac — potenciálne až nekonečno.

```
89  // Check if a material is detected and increment the counter
90  //Counter ALL
91  CTU_Counter_All(PV := 200, CU := NOT I_DifSen_1 );
92  IF I_DifSen_1 THEN
93      MaterialCount_All:=(CTU_Counter_All.CV);
94  END_IF
95
96  //Counter BLUE
97  CTU_Counter_Blue(PV := 100, CU := NOT I_VisionSen_1 );
98  IF I_VisionSen_1 THEN
99      MaterialCount_Blue:=(CTU_Counter_Blue.CV);
100 END_IF
101
102 //Counter GREEN
103 CTU_Counter_Green(PV := 100, CU := NOT I_VisionSen_2 );
104 IF I_VisionSen_2 THEN
105     MaterialCount_Green:=(CTU_Counter_Green.CV);
106 END_IF
107
108 //Counter error sorting
109 CTU_Counter_ErrorSorting(PV := 100, CU := NOT I_DifSen_6 );
110 IF I_DifSen_6 THEN
111     MaterialCount_ErrorSorting:=(CTU_Counter_ErrorSorting.CV);
112 END_IF
```

Obrázok č. 112: Ukážka kódu slúžiaceho na inkrementáciu premenných [98]

4.3.4 Prepojenie Factory I/O s Codesys pomocou OPC UA

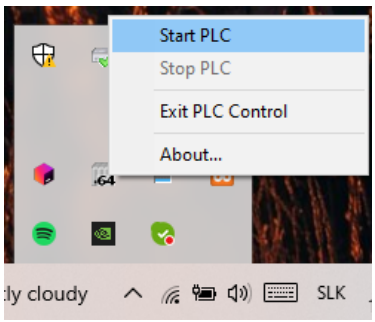
Po napísaní riadiaceho programu nasleduje jeho prepojenie s virtuálnym modelom systému vo Factory I/O, pričom toto bude prebiehať prostredníctvom komunikačného protokolu OPC UA. Na prepojenie premenných z Codesysu a Factory I/O je nutné do štruktúry Codesys projektu pridať objekt *Symbol Configuration*, kde treba označiť všetky premenné, ktoré chceme využiť vo Factory I/O. Následne realizujeme *Build* pre *Symbol Configuration* (v prípade neskoršej úpravy premenných je potrebné vždy znovu spustiť *Build* pre *Symbol Configuration*).



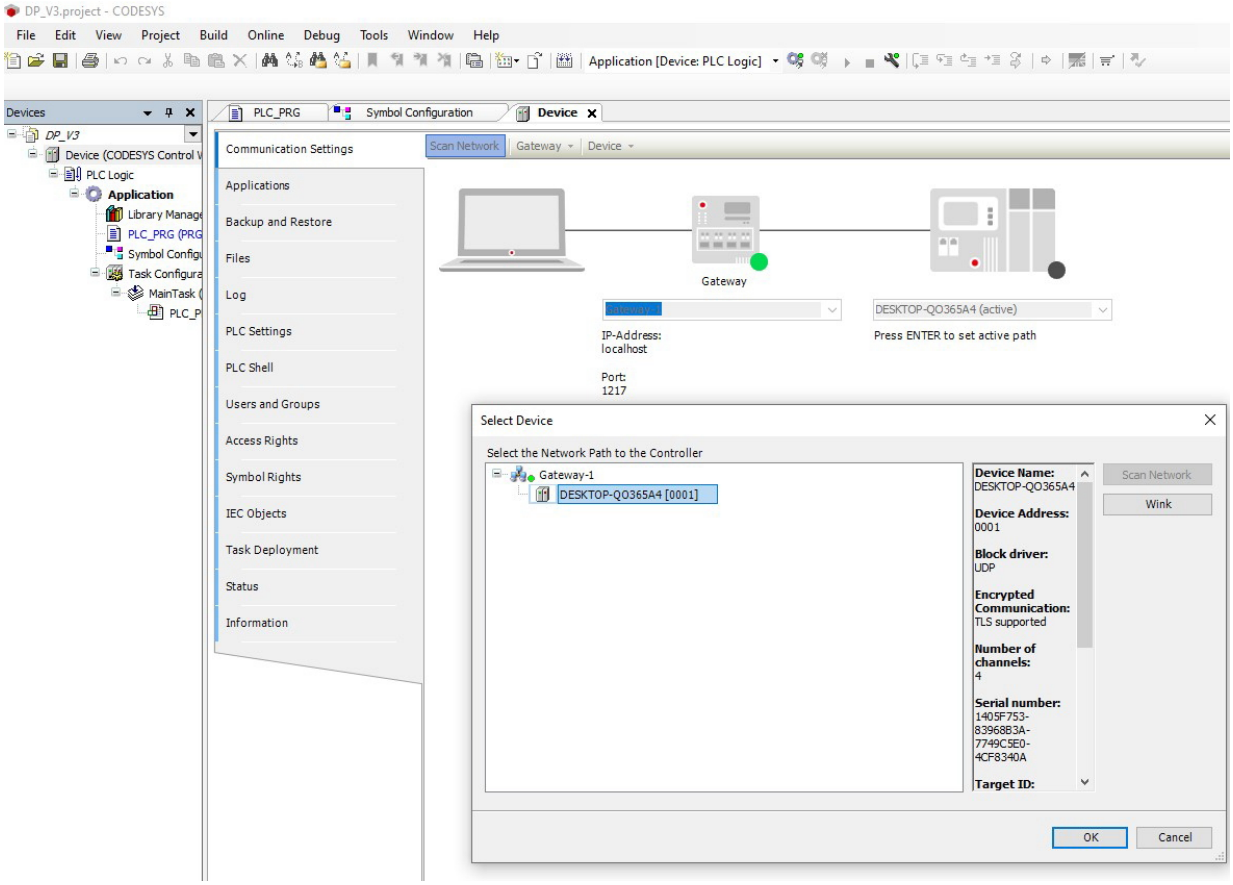
Obrázok č. 113: Objekt Symbol Configuration [98]

Po príprave premenných je potrebné urobiť build pre celý projekt. Je potrebné zap-

núť PLC server (obrázok č. 114), v možnosti *Device* zvoliť *Communication Settings*, spustiť *Scan Network* a vybrať zariadenie (obrázok č. 115). Následne je treba spustiť *Online->Login a Debug->Start*.



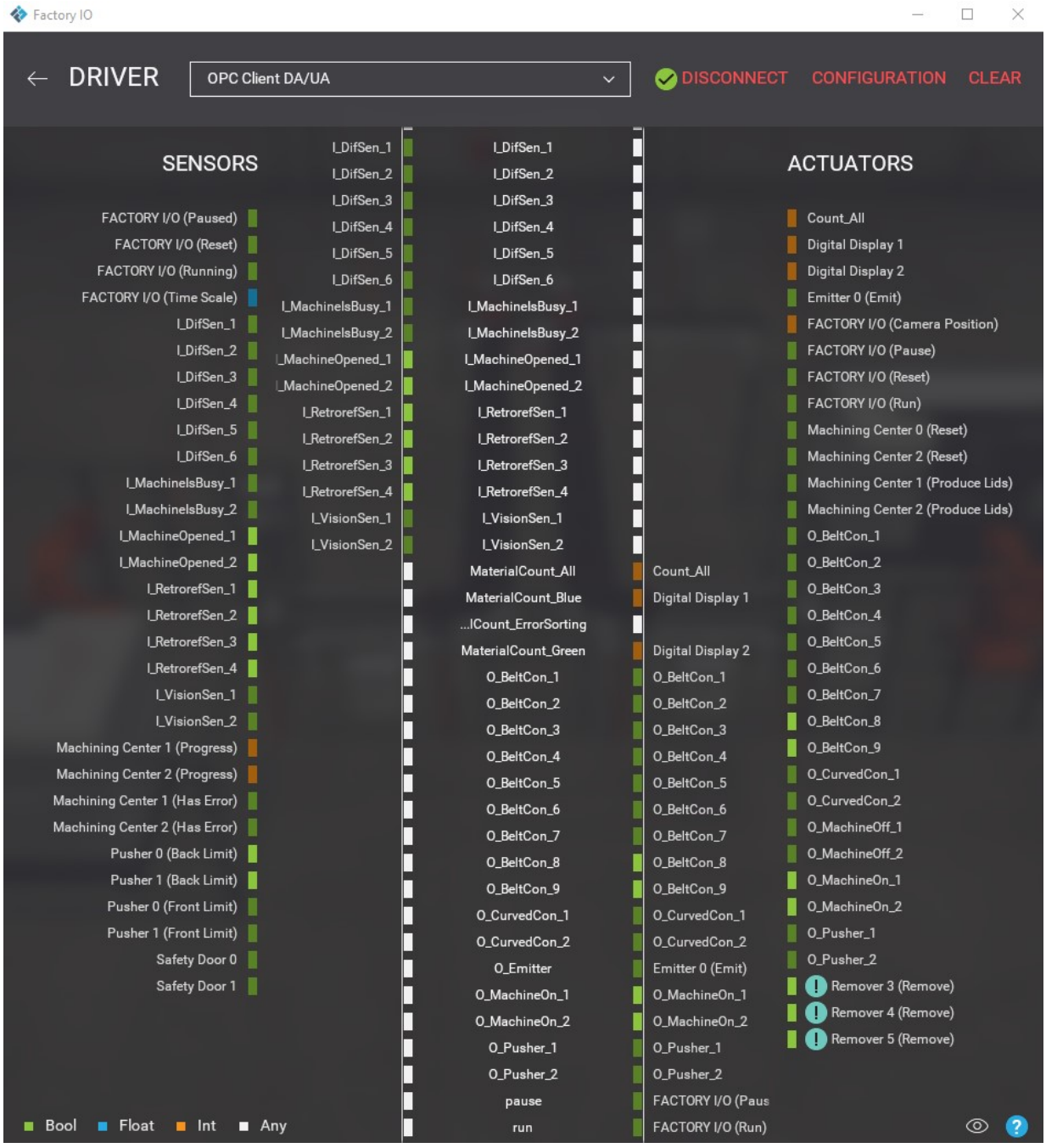
Obrázok č. 114: Zapnutie PLC servera [98]



Obrázok č. 115: Skenovanie siete a zvolenie zariadenia [98]

Po príprave Codesysu a spustení OPC servera je potrebné otvoriť Factory I/O, kde sme si zvolili scénu s naším pripraveným modelom diskrétneho udalostného systému. V konfigurácii je potrebné nastaviť ovládač OPC Client DA/UA, kde ako OPC UA Server zadáme *opc.tcp://localhost:4840* (v našom prípade sme použili *OPCUAServer@DESKTOP-*

QO365A4 (UA)). Ako filter pre premenné sme použili PLC. Po nastavení konfigurácie sme napárovali premenné z riadiaceho programu a Factory I/O (obrázok č. 116).



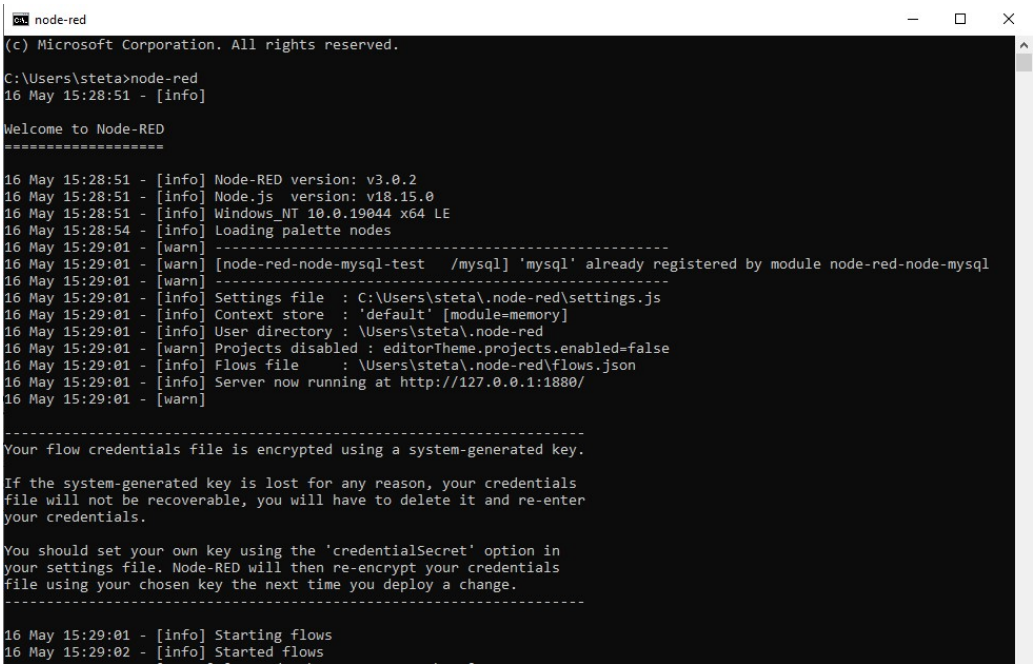
Obrázok č. 116: Párovanie premenných [98]

Po prepojení a správnom nakonfigurovaní sa riadiaci program nahral a je možné spustiť virtuálny model systému. Ako už bolo uvedené, komunikáciu zabezpečuje moderný komunikačný protokol OPC UA.

4.3.5 Node-RED a práca s dátami pomocou OPC UA a MQTT

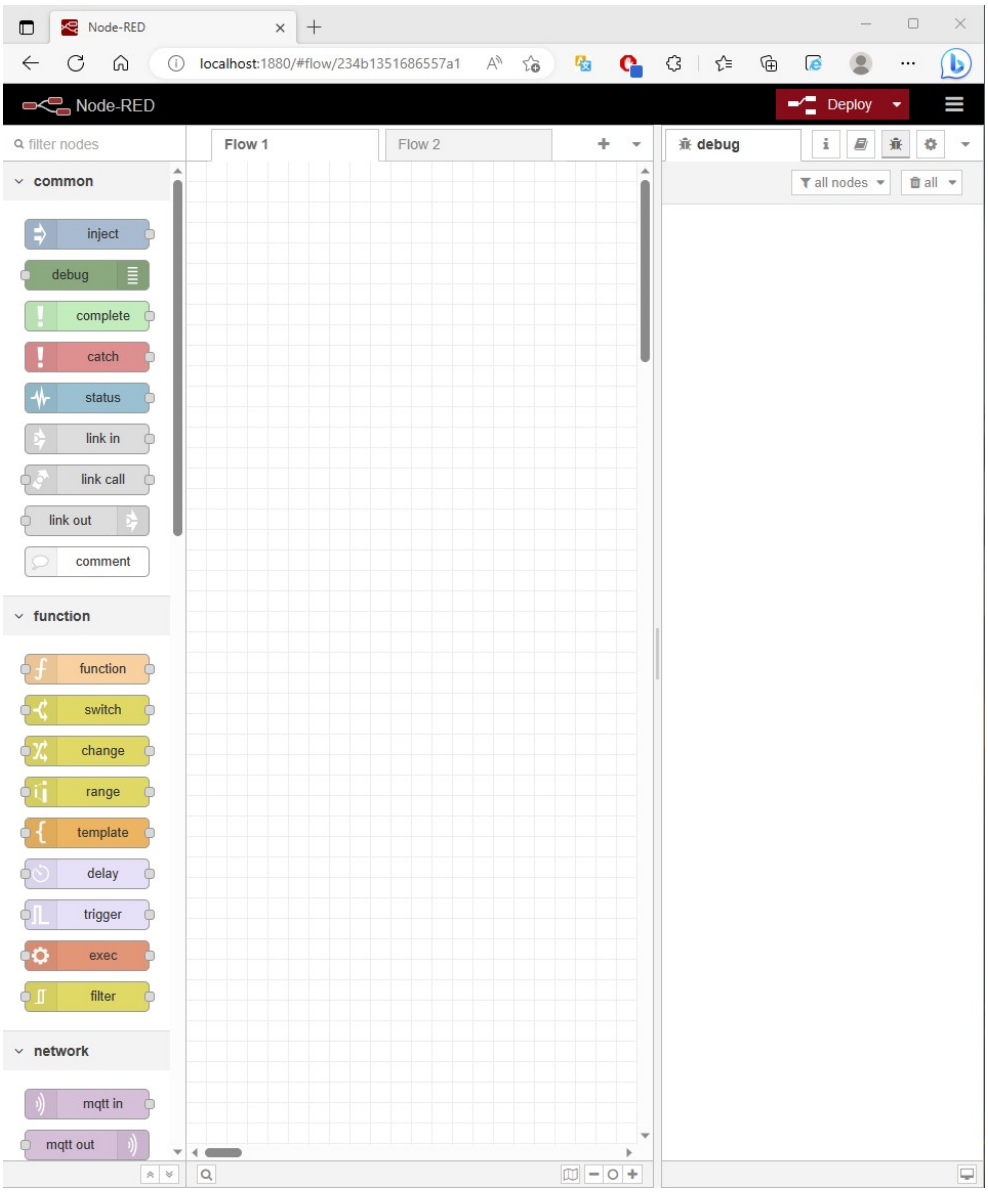
Middleware Node-RED neslúži v tejto prípadovej štúdii na riadenie virtuálneho modelu systému, ale funguje ako prostredník medzi OPC UA serverom a webovou aplikáciou. Slúži na preposielanie dát, v tomto prípade do webovej aplikácie. Webová aplikácia totiž nevie v opisovanom prípade priamo komunikovať s OPC UA serverom, nakoľko by tento OPC UA server musel figurovať na verejnej IP adrese, ktorú v tomto prípade nemáme k dispozícii.

Pre potreby tejto edukačnej prípadovej štúdie bol zvolený lokálne bežiaci Node-RED. Ten je potrebné nainštalovať do zariadenia. Prerekvizivitou lokálne bežiaceho Node-REDu je podporovaná verzia Node.JS, tú je teda tiež potrebné nainštalovať. Po splnení všetkých krokov a nainštalovaní je možné Node-RED spustiť. Spúšťame ho pomocou príkazu ***node-red***, ktorý zadáme v konzole (cmd.exe) alebo PowerShell, nakoľko pracujeme s operačným systémom Windows. Na obrázku č. 117 sa nachádza výpis po spustení Node-RED.



Obrázok č. 117: Príkazový riadok po spustení lokálneho Node-RED [98]

Do editora Node-RED sa potom dostaneme tak, že v prehliadači otvoríme adresu ***http://localhost:1880***.



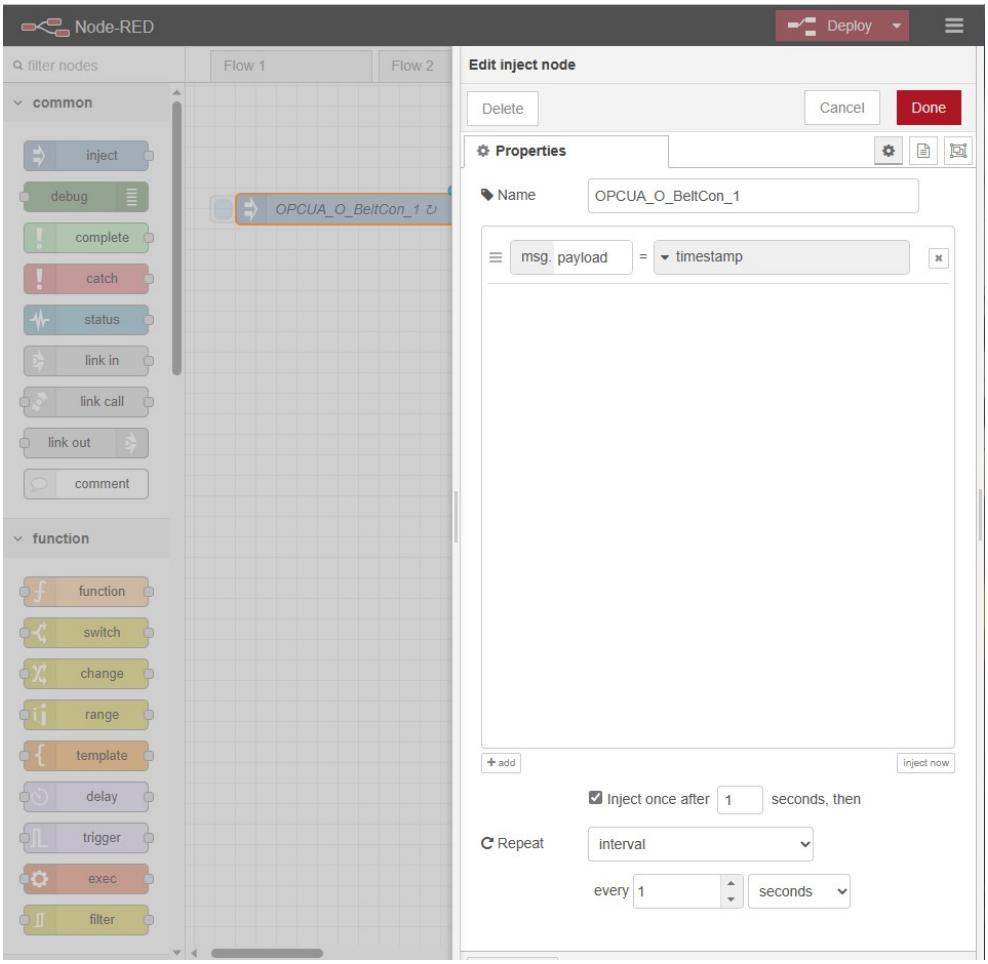
Obrázok č. 118: Prázdny Node-RED editor [98]

Programové toky v Node-REDe tvoríme prepájaním uzlov. Tvorbu tokov pre túto edukačnú prípadovú štúdiu môžeme rozdeliť do troch častí – čítanie dát z OPC UA servera a následné posielanie dát na MQTT broker, čítanie dát z MQTT brokera a ukladanie do databázy a treťou časťou je zasielanie dát späť na OPC UA server. Túto časť si však rozoberieme neskôr, pretože je to súčasť spätnej komunikácie z webovej aplikácie, ktorá slúži na manuálne (núdzové) riadenie nášho virtuálneho systému.

ČÍTANIE DÁT Z OPC UA SERVERA A NÁSLEDNÉ POSIELANIE DÁT NA MQTT BROKER

Prvý uzol, ktorý bol použitý v toku, je uzol *Inject*. Slúži na vloženie správy do toku alebo (inak povedané) aktiváciu danej vetvy uzlov. Na vkladanie sme použili nastave-

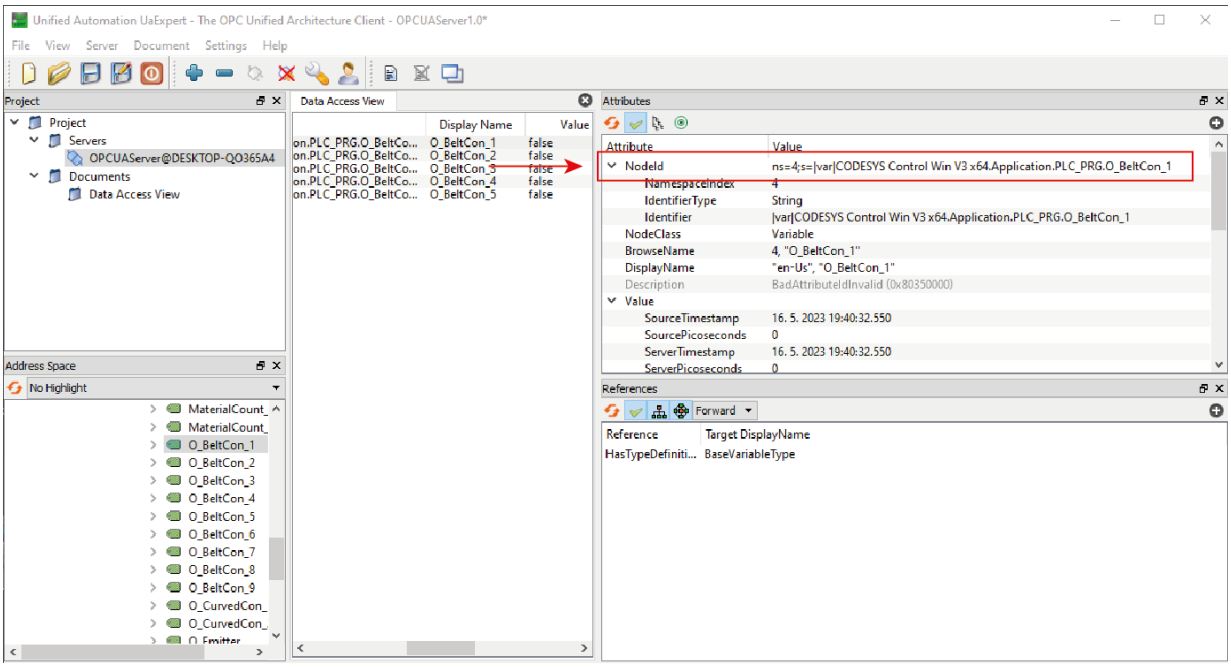
nie intervalu. Nastavenie uzla je možné vidieť na obrázku č. 119. Nastavený bol názov premennej do poľa *Name* a v dolnej časti bol nastavený interval na vkladanie správy. Prvé vloženie nastane jednu sekundu po pripojení a následne sa budú správy vkladať opakovane v intervale jednej sekundy. Toto nastavenie bolo použité v uzloch pre každú premennú, ktorej správu chceme prijímať z OPC UA servera.



Obrázok č. 119: Nastavenie uzla Inject [98]

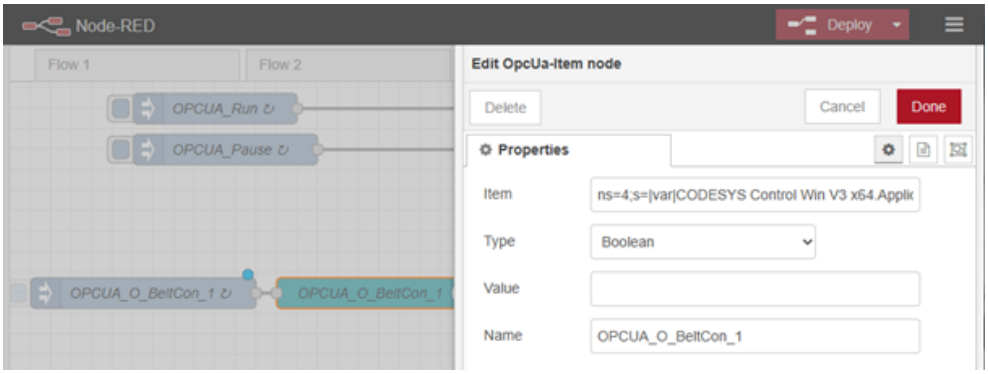
Nasledujúce dva uzly využívame z knižnice *node-opcua*, ktorá ponúka uzly na komunikáciu pomocou komunikačného protokolu OPC UA. Knižnicu nainštalujeme pomocou voľby *Manage Palette*, kde si vyhľadáme ***node-red-contrib-opcua*** a spustíme inštaláciu. Následne sa uzly pre komunikáciu OPC UA pridajú do palety editora.

Uzol *OpcUa-Item* sme využili na definovanie položky, názvu a typu premennej. Do poľa *Item* sme vložili charakteristiku premennej, ktorej správu chceme prijať. Na získanie tohto údaja (charakteristiky) sme podobne ako v prvej prípadovej štúdii využili softvér (OPC UA klient) UaExpert, ktorým sme sa pripojili na OPC UA server vytvorený pomocou Codesysu a do stredného okna sme umiestnili všetky premenné, ktoré potrebujeme. Dôležitý údaj pre nás je *NodeId* každej premennej (obrázok č. 120).



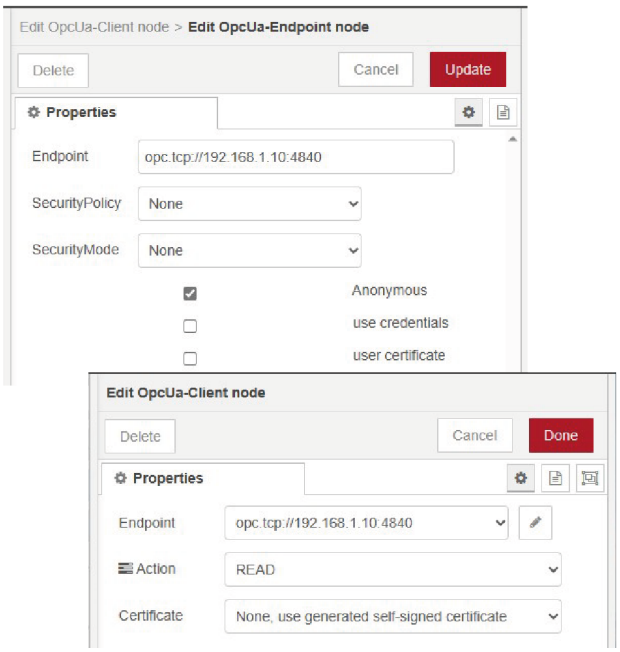
Obrázok č. 120: NodeId v programe UaExpert [98]

Následne je potrebné vyplniť typ premennej, ktorej správu budeme prijímať a jej názov. Nastavenia tohto uzla vidíme na obrázku č. 121.

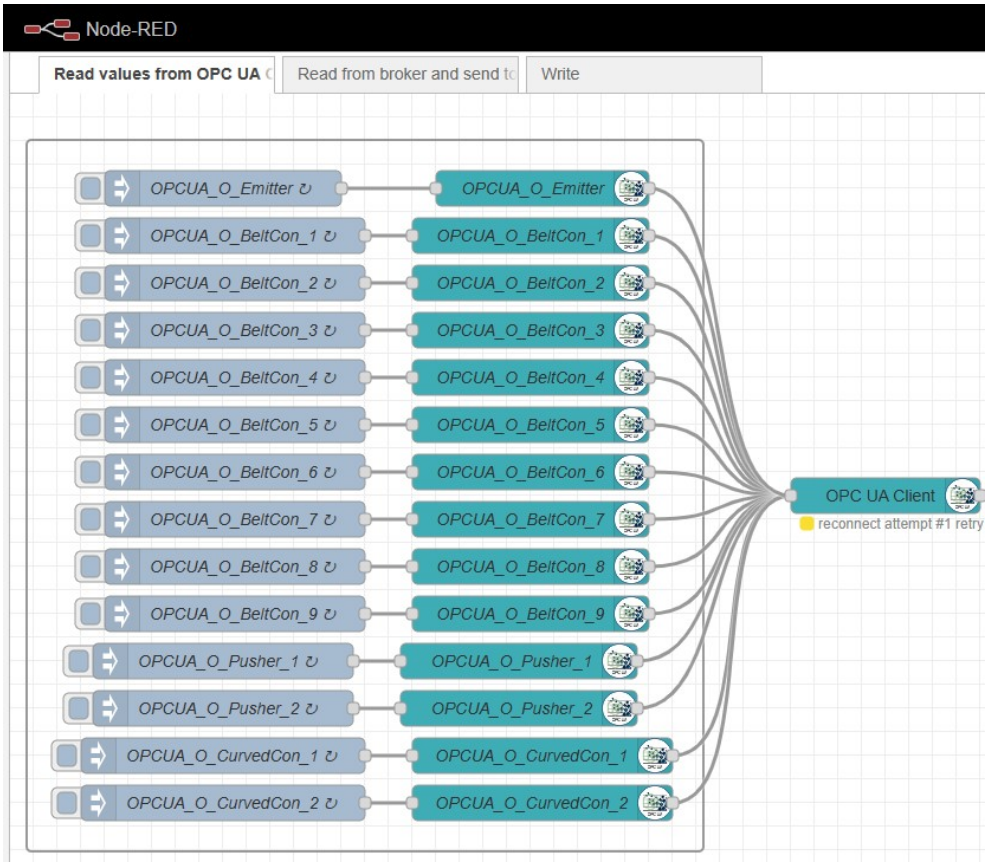


Obrázok č. 121: Uzol OpcUa Item s nastavením [98]

Uzol *OpcUa-Client* sme využili na pripojenie sa k OPC UA serveru. Ako endpoint sme zadali ***opc.tcp://192.168.1.10:4840*** – čo zodpovedá LAN IP zariadenia. V možnosti Action bol zvolaný READ, pretože chceme správy z OPC UA servera čítať (obrázok č. 122). Po správnej konfigurácii sme uzly prepojili.



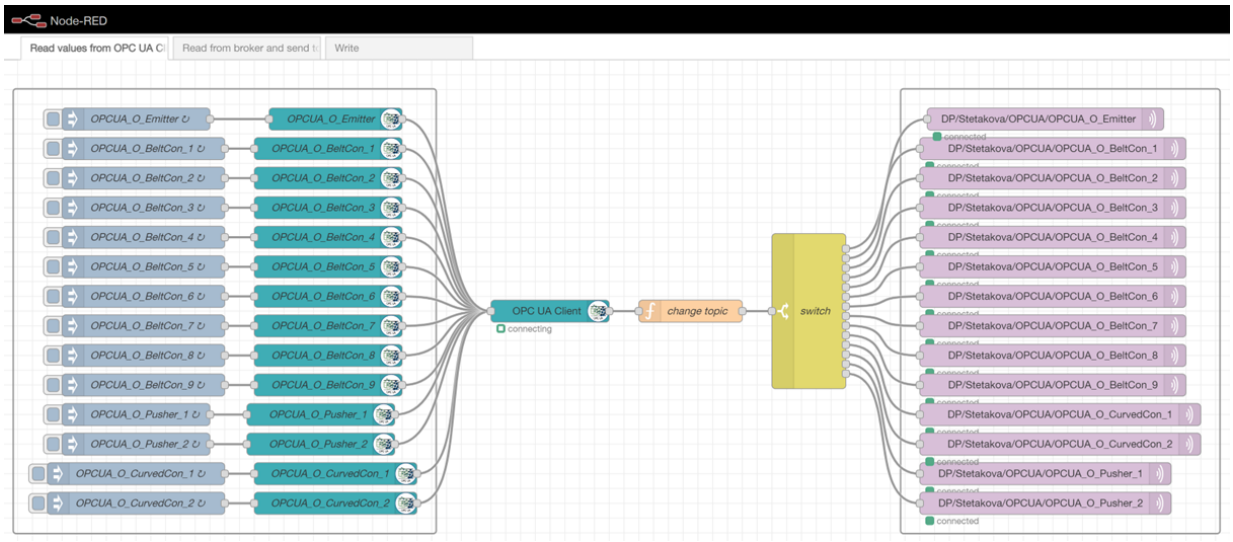
Obrázok č. 122: Nastavenie uzla OpcUa-Client a endpointu [98]



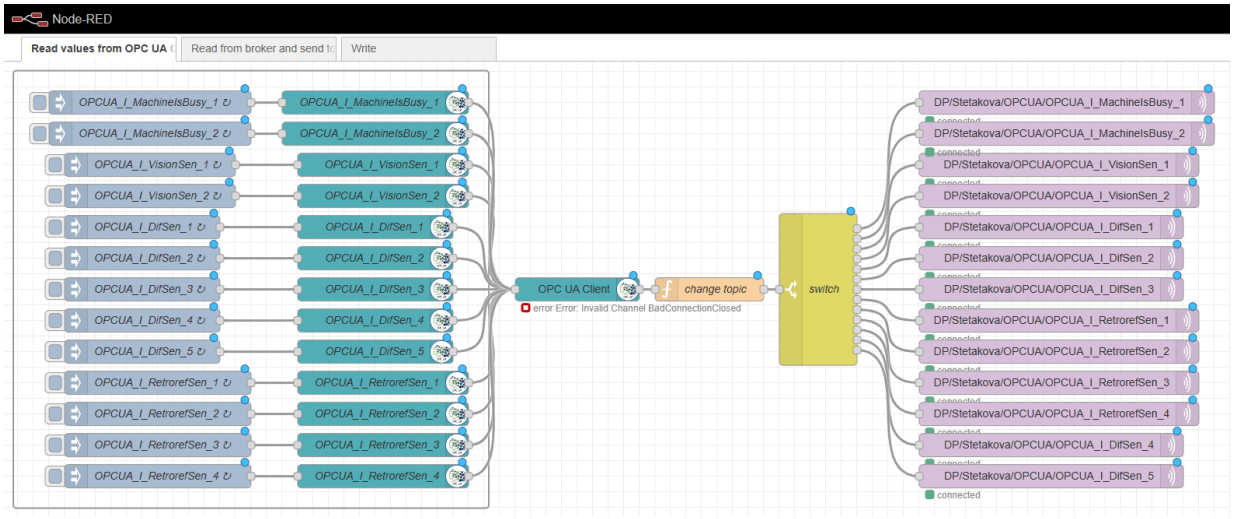
Obrázok č. 123: Prepojenie uzlov pre čítanie z OPC UA servera [98]

POSIELANIE A PRIJÍMANIE DÁT POMOCOU MQTT

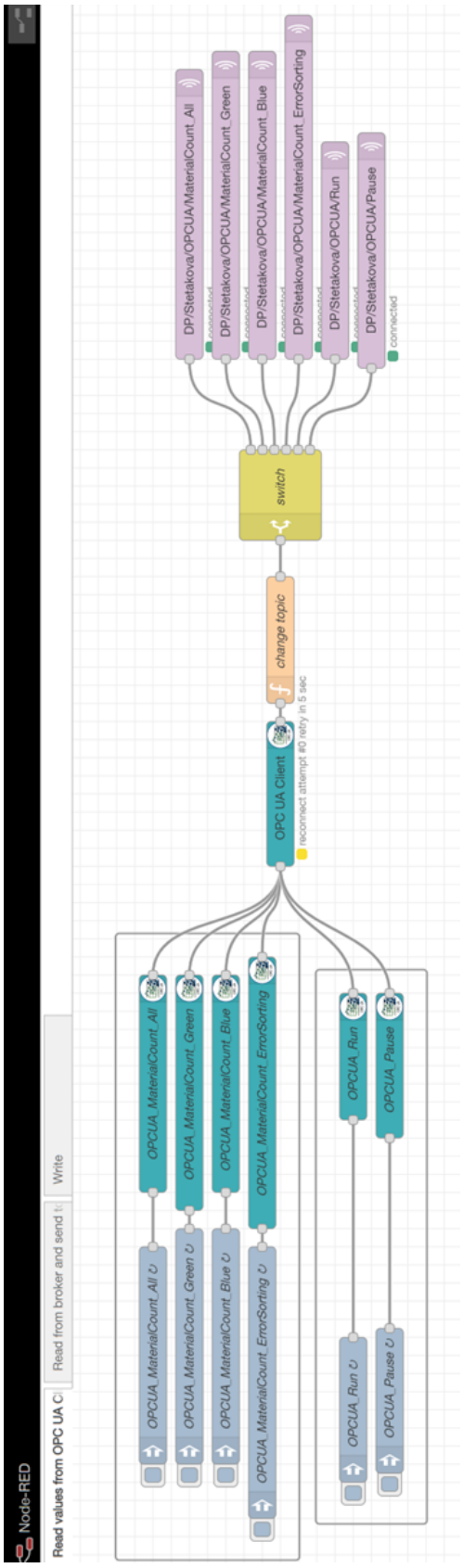
Nasleduje posielanie údajov na MQTT broker pomocou Node-RED uzla *mqtt out*. Avšak na to, aby sme ich mohli posielať so správnou témou (topic), sme využili na úpravu uzly *Function* a *Switch*. Uzol *Function* obsahuje kód napísaný v jazyku JavaScript, ktorý slúži na úpravu štruktúry správy a zmenu témy. Po úprave témy sme využili uzol *Switch*, ktorý zabezpečuje presmerovanie správy do *mqtt out* uzla so správnou témou. V uzle *mqtt out* sme ako server (broker) pre MQTT využili *broker.hivemq.com* a port 1883, čo je voľne dostupný broker aj pre testovacie účely. Po prepojení všetkých uzlov nám vznikla prvá časť tvorby toku dát. Pre lepší prehľad sme premenné rozdelili do skupín podľa vstupu, výstupu, premennej typu INT a riadiacich premenných.



Obrázok č. 124: Načítanie správ a ich posielanie na MQTT broker (výstupy) [98]



Obrázok č. 125: Načítanie premenných a ich posielanie na MQTT broker (vstupy) [98]



Obrázok č. 126: Načítanie premenných a ich posielanie na MQTT broker (INT a riadiace premenné) [98]

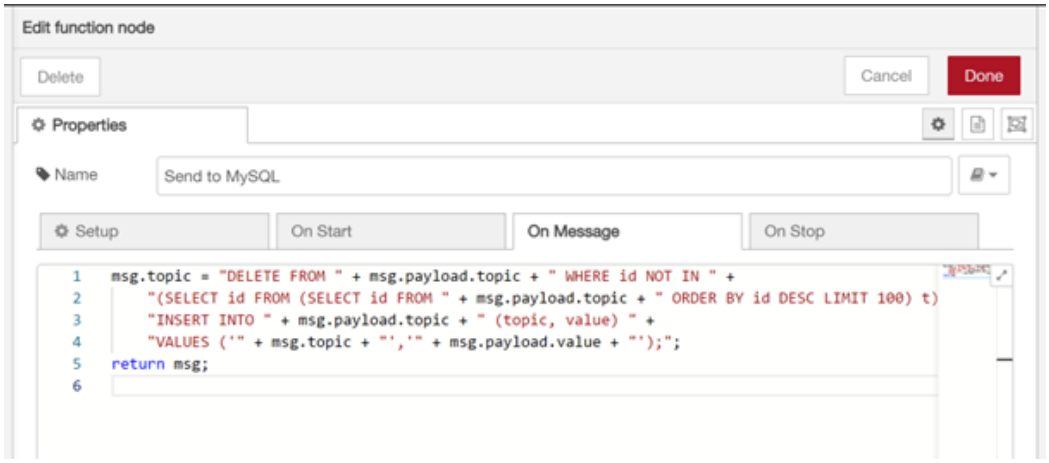
ČÍTANIE DÁT Z MQTT BROKERA A UKLADANIE DO DATABÁZY

Druhá časť zabezpečuje zasielanie dopytu na MQTT broker s názvami tém, ktorých správy je žiadané prijímať. Následne sa tieto dáta spracujú a pomocou funkcie sa vytvorí dopyt (angl. query) na zápis údajov do databázy.

Na zasielanie dopytu a následné prijímanie správy konkrétnej témy bol využitý uzol *mqtt in*. Do nastavení bola zadaná adresa MQTT brokera, ktorého dáta prijímame a port. Nastavili sme dopytovanú tému a QoS (Quality of Service) level na 2. Toto znamená, že každý paket je doručený do cieľa presne jedenkrát.

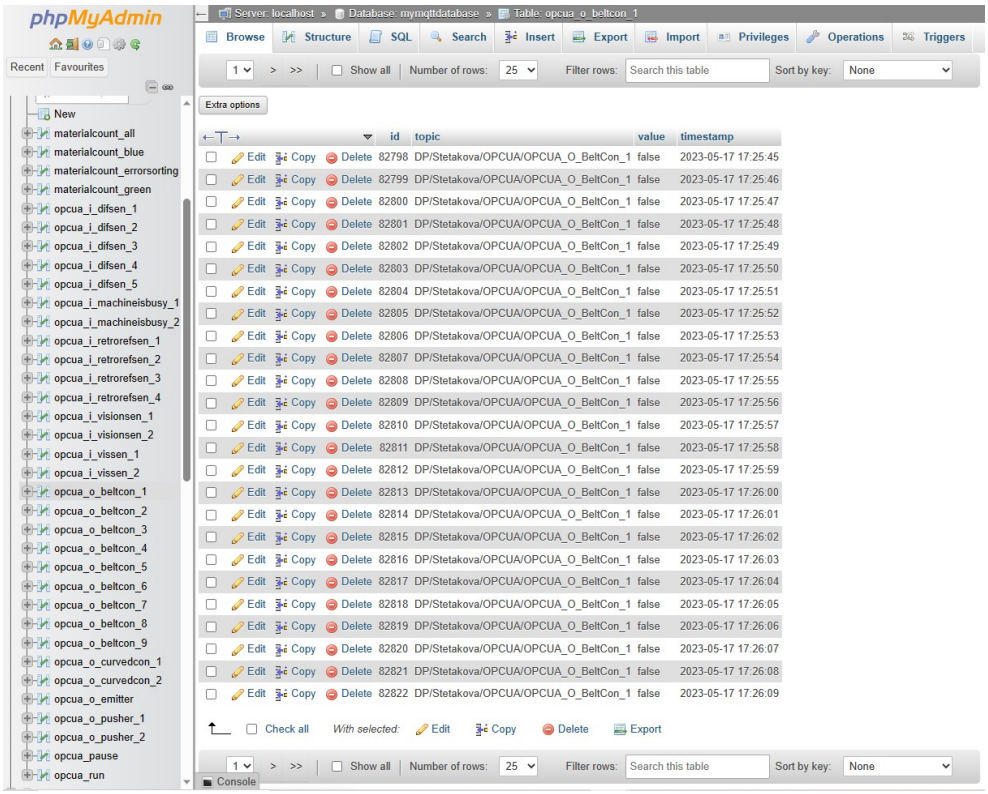
Následne sme využili uzol *Function*. Napísali sme vlastný kód, ktorý do *msg.topic* vloží SQL dopyt (SQL query) pre MySQL databázu. Tento dopyt po odoslaní správy skontroluje počet záznamov a v prípade, že je počet záznamov viac ako 100, tak vyberie id prvých 100 záznamov z tabuľky a odstráni ich, čo zabezpečí zbytočné neplnenie databázy starými záznamami. Vždy teda zostane v tabuľke 100 najaktuálnejších záznamov. Po odstránení ďalej query zabezpečí vloženie nového záznamu do konkrétnej tabuľky s hodnotami správy, ktoré boli získané z MQTT brokera.

Na záver bol tok ukončený pripojením k uzlu *mysql*, ktorý zabezpečí pripojenie k databáze a vykonanie SQL dopytu.

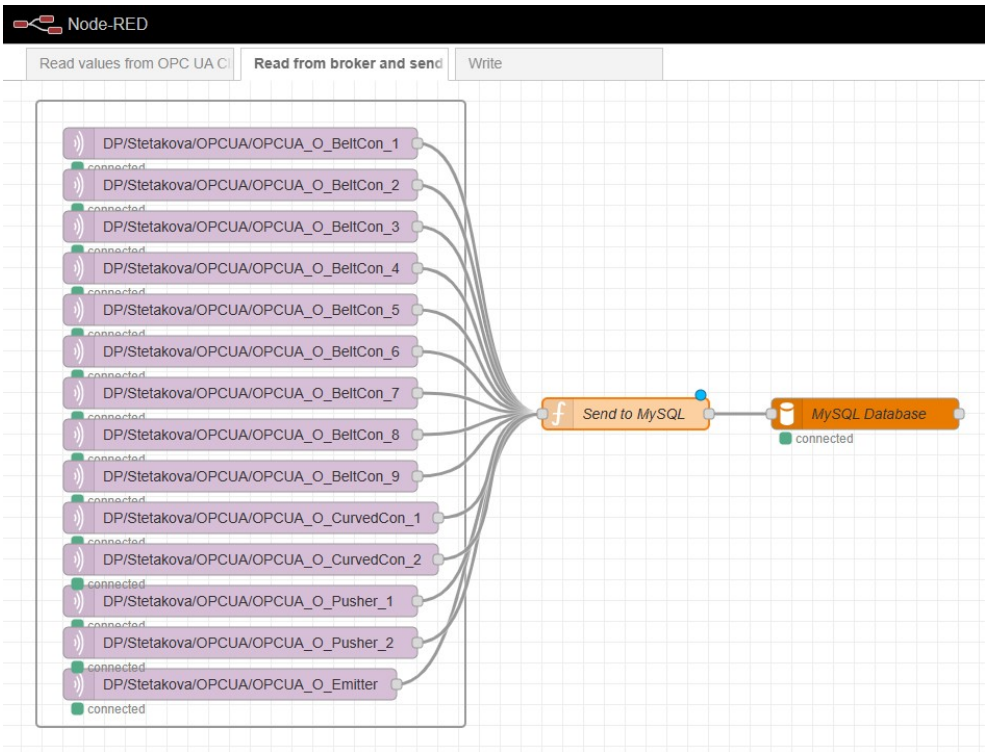


Obrázok č. 127: Uzol Function s kódom na vytvorenie query [98]

Pred spustením tejto časti je však potrebné vytvoriť si MySQL databázu a tabuľky v nej. V rámci prípadovej štúdie sme na vytváranie databázy a tabuliek využili PHP skript, vďaka ktorému sme sa vyhli manuálnemu vytváraniu každej tabuľky. Pre každú premennú bola vytvorená osobitná tabuľka so stĺpcami pre id, názov, hodnotu a časovú stopu (angl. timestamp).



Obrázok č. 128: Ukážka štruktúry databázy [98]



Obrázok č. 129: Tok pre načítanie dát a uloženie do databázy [98]

4.3.6 PHP webová aplikácia

Webová aplikácia slúži na zobrazenie posielaných údajov, ale je ju tiež možné použiť na manuálne ovládanie nášho virtuálneho modelu diskrétneho udalostného systému.

Vytvorili sme nový PHP projekt. Na vytvorenie PHP webovej aplikácie, ktorá dokáže prijímať a posielať dáta cez MQTT protokol je potrebné nainštalovať *php-mqtt/client*. Nainštalovali sme ďalej *composer*, ktorý slúži na správu knižníc a iných balíčkov pre PHP. Balíček *php-mqtt/client* je možné nainštalovať pomocou príkazu `composer require php-mqtt/client`, ktorý treba zadať do príkazového riadku s tým, aby ako pracovný priečinok bol nastavený priečinok s projektom. Po nainštalovaní je možné používať funkcie balíčka na odoberanie (subscribe) a publikovanie (publish) správ.

Programový modul č. 1 Kód na pripojenie k databáze [98]

```php
<?php

$host = "localhost";
$username = "root";
$password = "";
$dbname = "myMQTTdatabase";

// Create connection
$conn = mysqli_connect($host, $username, $password, $dbname);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

?>
```

Potom sme vytvorili PHP skript s kódom, ktorý zabezpečuje dopyt z databázy. Pripravili sme si prístupové údaje na pripojenie k databáze: názov servera, používateľské meno a heslo. Na pripojenie sme použili *mysqli* (programový modul č. 1). Následne sme vytvorili dopyt (programový modul č. 2), ktorým sme vybrali z tabuľky potrebné dáta a uložili sme ich do poľa.

Na ďalšie spracovanie týchto dát bol vytvorený skript v jazyku JavaScript (programový modul č. 3). Tu boli využité funkcie a AJAX, ktoré zabezpečili aktualizovanie prichádzajúcich údajov z databázy bez potreby znovunačítania webovej aplikácie. Po načítaní webovej aplikácie sa vygenerujú tabuľky aj s hodnotami z databáz. Následne

Programový modul č. 2 Ukážka dopytu na získanie údajov z databázy [98]

```php
<?php
require_once 'config.php';

// Query the data
$sql = "SELECT * FROM (SELECT * FROM OPCUA_O_Emitter ORDER BY timestamp DESC LIMIT 1) c1
        UNION ALL
        SELECT * FROM (SELECT * FROM OPCUA_I_DifSen_1 ORDER BY timestamp DESC LIMIT 1) c2
        UNION ALL
        SELECT * FROM (SELECT * FROM OPCUA_O_BeltCon_1 ORDER BY timestamp DESC LIMIT 1) c3";

$result = mysqli_query($conn, $sql);


// Handle errors in the query
if (!$result) {
    die('Query error: ' . mysqli_error($conn));
}

// Convert the data to an array
$data = array();

while ($row = mysqli_fetch_assoc($result)) {
    $data[] = $row;
}
// Close the database connection
mysqli_close($conn);

// Return the data as JSON
header('Content-Type: application/json');
echo json_encode($data);

?>
```

tiež funkcia zabezpečí aktualizáciu hodnoty prijatej z databázy každú sekundu.

Programový modul č. 3 Funkcia v jazyku Javascript zabezpečujúca aktualizáciu údajov [98]

```
// Function to update the table data
    function updateTableData() {
        $.ajax({
            url: 'getData/get_data_common.php',
            type: 'post',
            dataType: 'json',
            success:function(response) {
                // Iterate over the response data and update values
                $.each(response, function(index, record) {
                    const topic = record.topic;
                    const myArray = topic.split("/");
                    const variable = myArray[3];
                    const value = record.value;

                    // Update the circle with the new value
                    updateCircle(variable, value);
                });
            }
        });
    }

    // Update the table data periodically
    setInterval(function(){
        updateTableData();
    }, 1000);
```

Na zobrazenie tabuliek sme vytvorili PHP súbor s HTML značkami, do ktorých sme vkladali vygenerované dáta na základe ID. Táto časť slúži na prijímanie a zobrazovanie údajov.

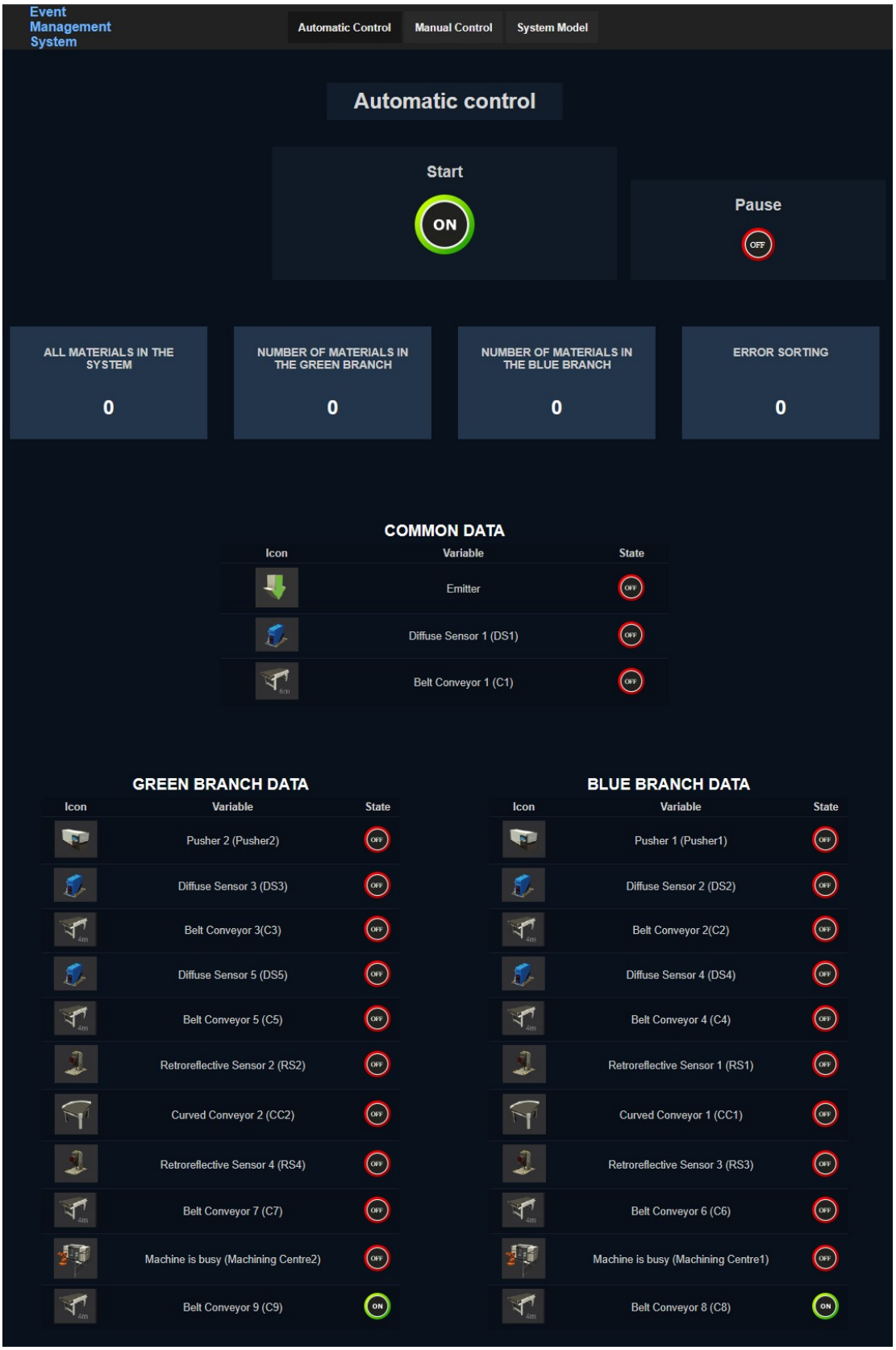
Následne bola vytvorená časť, ktorá nám zabezpečuje spätnú komunikáciu s MQTT brokerom. Na tieto účely sme z balíka *php-mqtt/client* využili funkciu *publish*. V jazyku JavaScript sme vytvorili funkcie *onClick*, ktoré prislúchajú k tlačidlám a ako odozva na kliknutie sa vytvorí správna správa a spustí sa PHP skript. Tento pomocou *publish* danú správu pošle na broker cez MQTT protokol. V programovom module č. 4 je možné vidieť súbor *publish.php* s pripojením na MQTT broker a následné odoslanie správy.

Programový modul č. 4 Pripojenie na MQTT broker a odoslanie správy [98]

```php
<?php

require_once 'vendor/autoload.php';

use PhpMqtt\Client\MqttClient;

// Get the JSON object sent in the POST request
$data = json_decode(file_get_contents('php://input'), true);

// Extract the topic and value from the JSON object
$topic = $data['topic'];
$value = $data['value'];

$client = new MqttClient('broker.hivemq.com', 1883);
$client->connect();
$client->publish($topic, $value);
$client->disconnect();

// Redirect back to the index page
header("Location: index.php");
die();
?>
```

Na záver bola webová aplikácia naštýlovaná pomocou CSS súboru.

Finálna webová aplikácia obsahuje **tri podstránky**. Prvá podstránka **Automatic Control** (obrázok č. 130) slúži na zobrazenie stavov premenných. Vidíme tu tri tabuľky, karty s hodnotami určujúce počet materiálov v systéme. Ďalej sú tu dve osobitné karty. Jedna zobrazuje stav továrne, teda či je virtuálny model systému zapnutý alebo vypnutý. Druhá zobrazuje, či je továreň v stave *Pause*. Druhá podstránka **Manual Control** (obrázok č. 131) slúži užívateľovi na prípadné manuálne zásahy do systému.

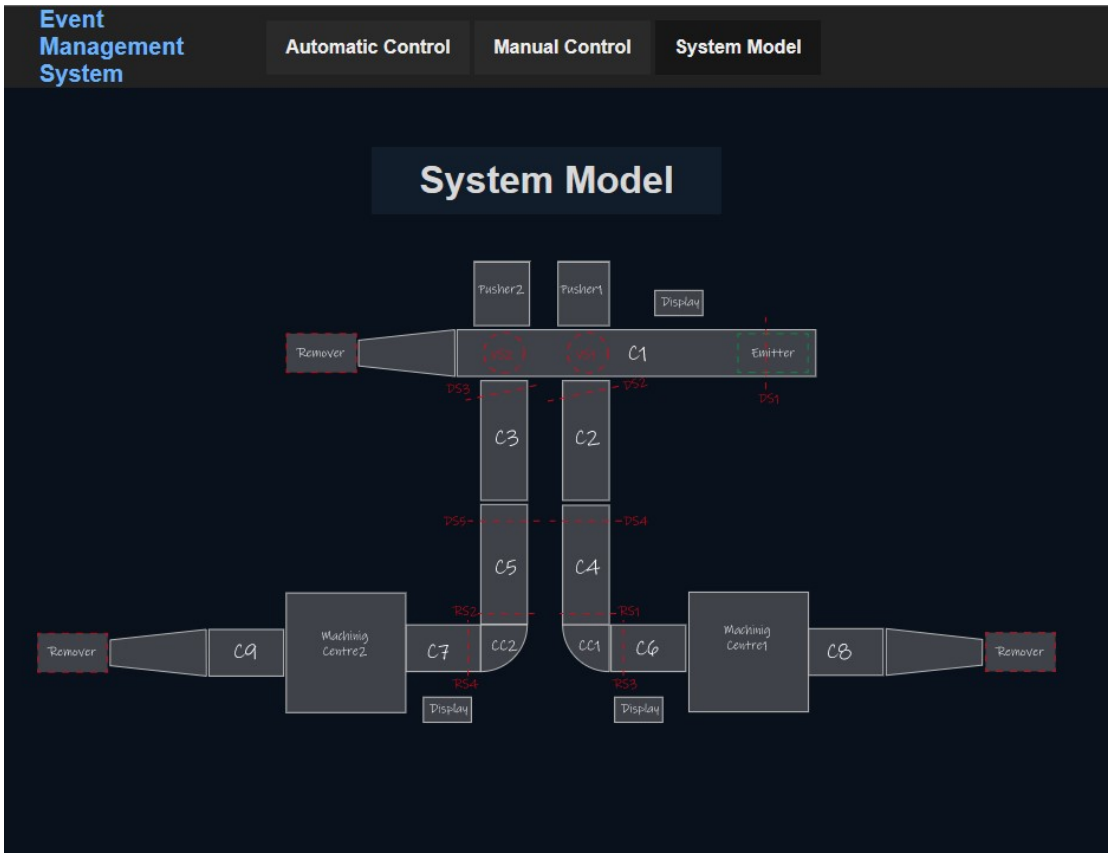


Obrázok č. 130: Ukážka webovej aplikácie — Automatic Control [98]

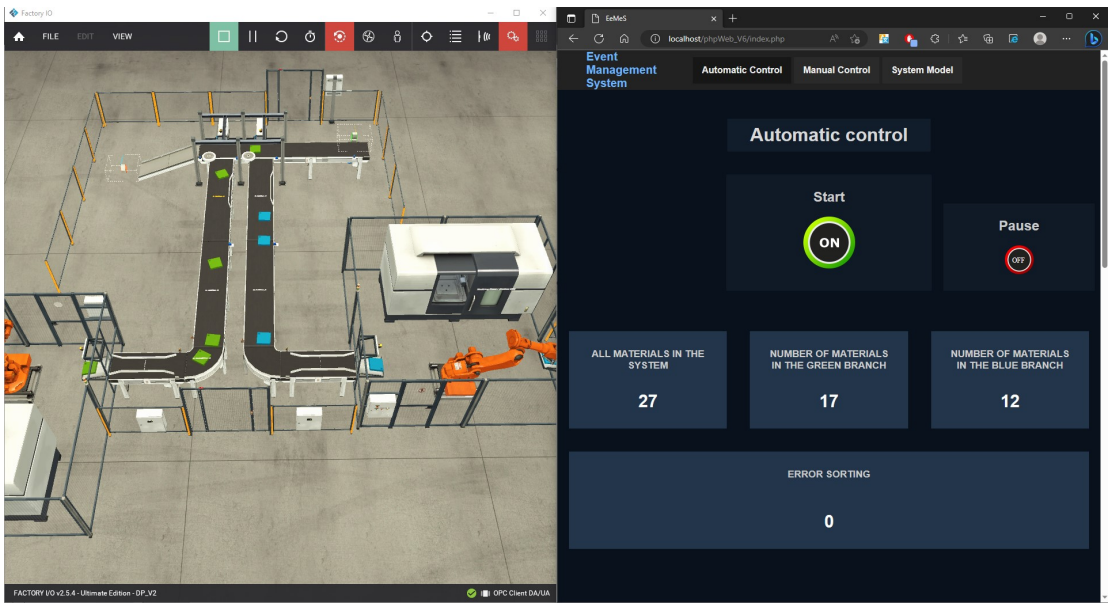
Obrázok č. 131: Ukážka webovej aplikácie — Manual Control [98]

Nachádzajú sa tu tiež dve osobitné karty, ktoré v tomto prípade slúžia nielen na zobrazenie stavu, ale tiež aj na riadenie spustenia alebo pozastavenia továrne. Okrem toho sa tu nachádzajú tri tabuľky, ktoré obsahujú zobrazenie stavu daného prvku a dve tlačidlá, ktoré zabezpečujú zapnutie alebo vypnutie. Toto riadenie je zabezpečené spätnou komunikáciou, ktorá bude vysvetlená v nasledujúcej podkapitole. Tretia pod-

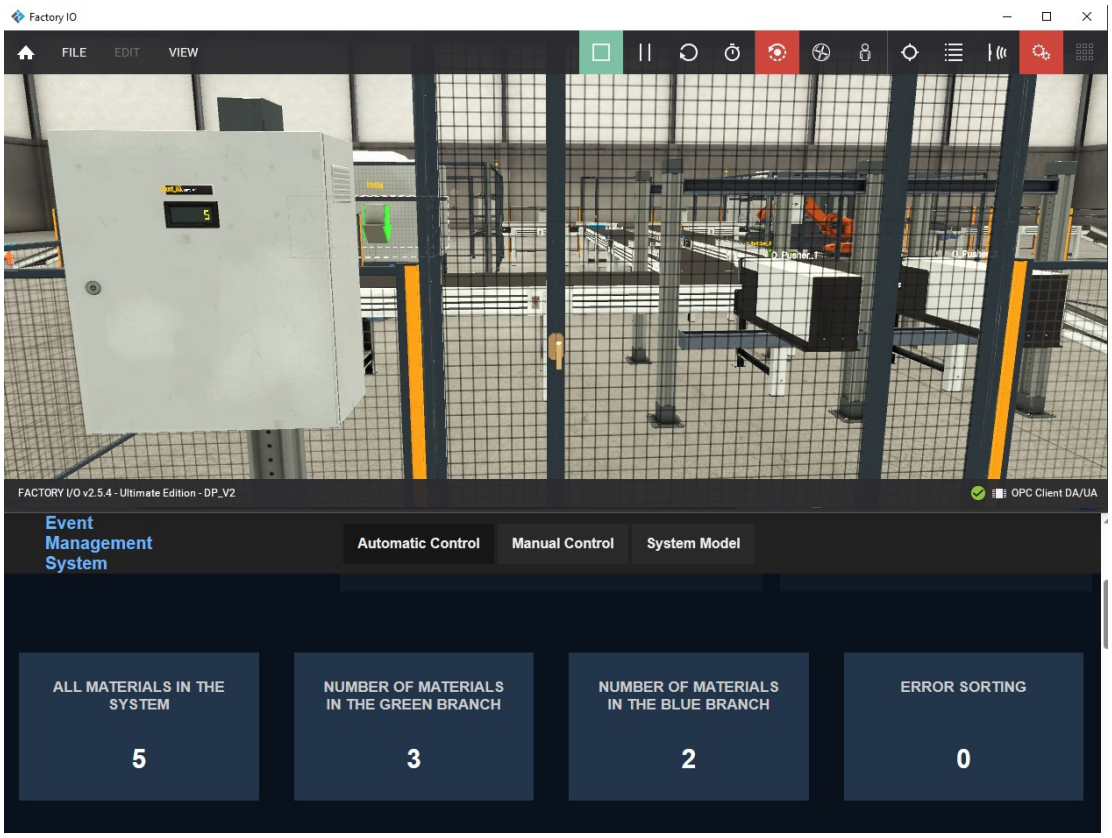
stránka obsahuje **schému modelu výrobného systému** (obrázok č. 132). Toto môže slúžiť na lepšiu orientáciu používateľa.



Obrázok č. 132: Ukážka webovej aplikácie — System Model [98]



Obrázok č. 133: Ukážka prepojenia webovej aplikácie s virtuálnou továrňou [98]

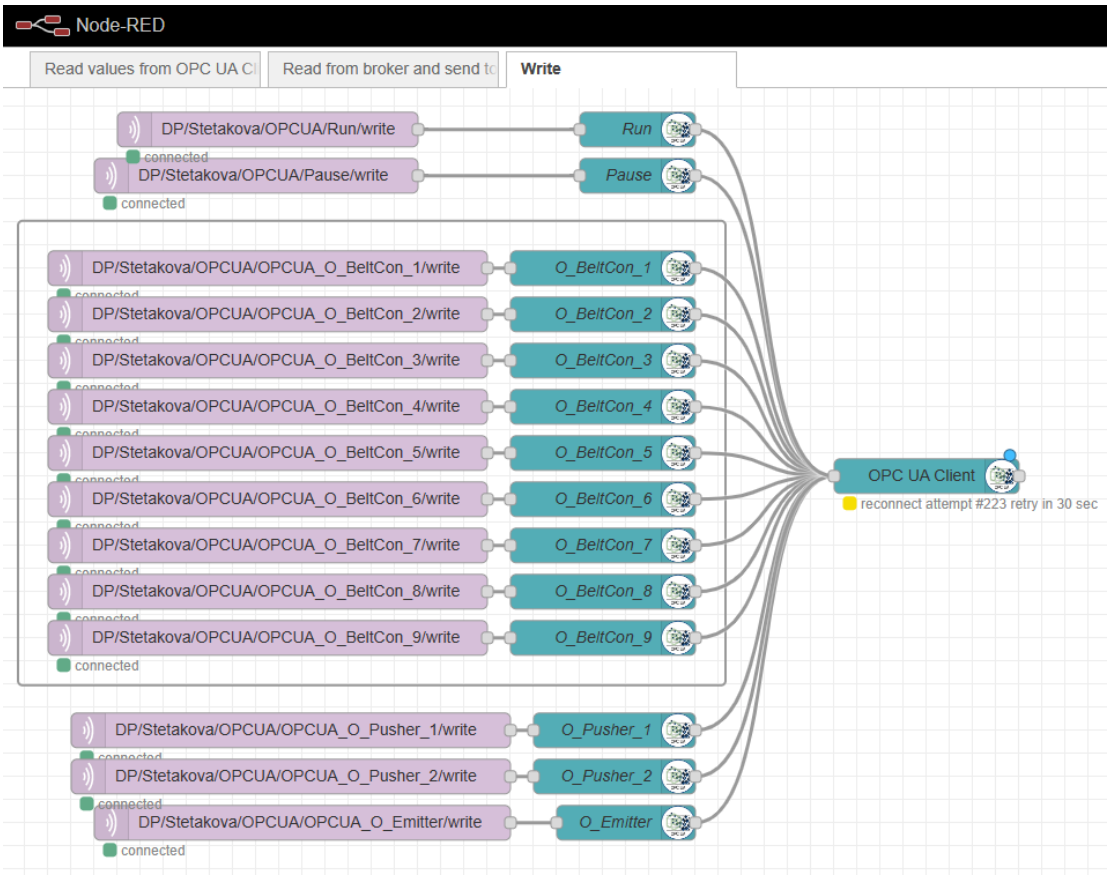


Obrázok č. 134: Ukážka prepojenia webovej aplikácie s virtuálnou továrňou [98]

4.3.7 Spätná komunikácia v Node-RED s OPC UA serverom

Spätná komunikácia zabezpečuje interakciu medzi webovou aplikáciou a OPC UA serverom prostredníctvom middlewaru Node-RED.

Prvú časť spätnej komunikácie zabezpečuje webová aplikácia pomocou balíka *php-mqtt/client*, z ktorého sme použili funkciu *publish*. Najprv sme sa pripojili k MQTT brokeru (v opisovanej štúdii ide o HiveMQ) a následne publikovali správu. Tieto publikované správy boli zaslané na MQTT broker, odkiaľ ich potrebujeme získať a poslať ďalej. Na zabezpečenie čítania z MQTT brokera sme znovu použili prostredník Node-RED, kde sme vytvorili nový programový tok (obrázok č. 135). Využili sme uzol *mqtt in*, ktorému sme určili, ktoré témy chceme odoberať. Následne bol vybraný uzol *OpcUa Item*, kde boli vyplnené údaje o premennej v riadiacom programe. Do Item sme vložili NodeId danej premennej, správne sme určili typ premennej a vyplnili sme názov — ten slúži len na označenie nášho uzla pre prehľadnosť. Potom sme využili uzol *OpcUa Client* a nakonfigurovali sme ho. Vyplnili sme endpoint (localhost sme nahradili LAN IP zariadenia) a ako Action sme zvolili WRITE. Vďaka vyplnenému NodeId si následne server priradí hodnoty pre správnu premennú.



Obrázok č. 135: Spätná komunikácia s OPC UA Serverom v Node-RED [98]

V poslednej fáze bolo riešenie otestované. Nosným prvkom systému je komunikácia, preto bola hlavným predmetom testovania práve celková komunikácia medzi jednotlivými zložkami celého systému. Po otestovaní komunikácie bola testovaná webová aplikácia vrátane jej responzivity na rôznych zariadeniach a rozmeroch obrazoviek. Výsledkom testovania bolo potvrdenie funkčnosti riadenia a komunikácie a aj responzivita webovej aplikácie. Z toho plynie, že návrh a následná implementácia riadiaceho a monitorovacieho systému boli funkčné.

Video k tejto edukačnej prípadovej štúdii je možné nájsť na tejto adrese:

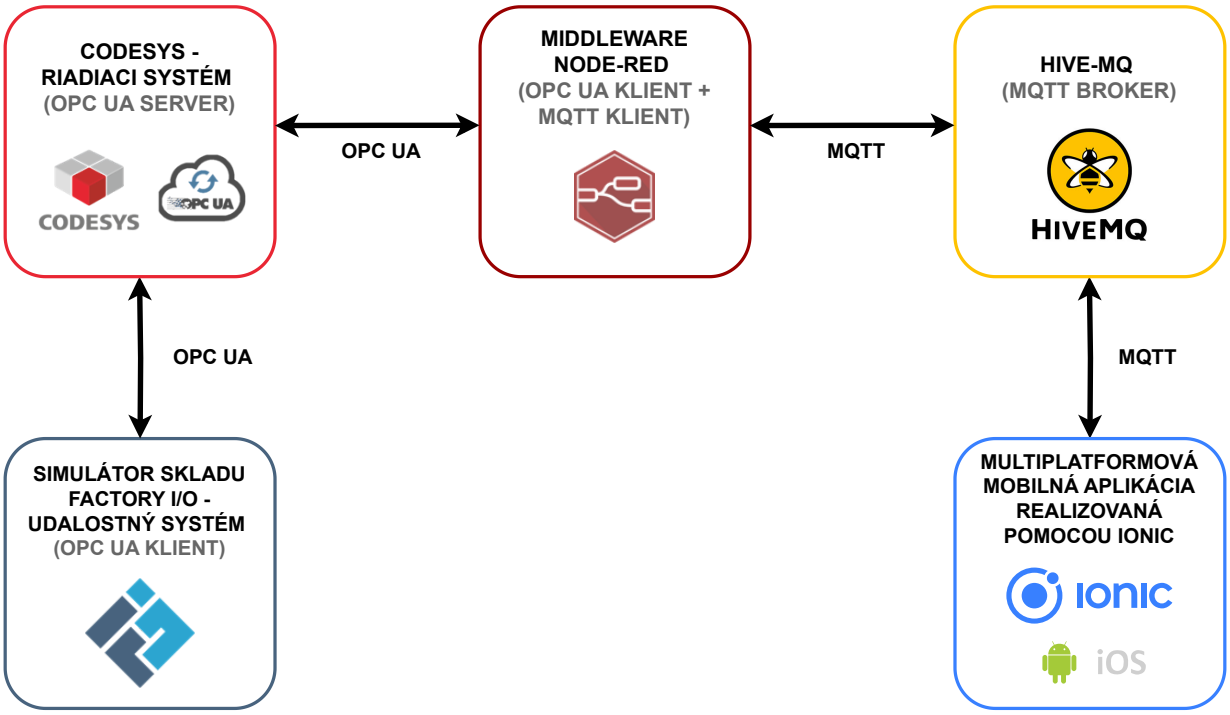
- `https://youtu.be/k4RjiVWXSW0`

Ďalšie informácie, návody a programové projekty je možné nájsť na e-learningovej webovej stránke `https://elearning.mechatronika.cool/?p=8348` .

4.4 Štvrtá prípadová štúdia: Prepojenie PLC s mobilnou aplikáciou

V tejto edukačnej prípadovej štúdii je uvedené riešenie pre automatizáciu virtuálneho skladového systému, ktorý je riadený prostredníctvom softPLC (realizovanom v Codesys) a mobilnej aplikácie. Virtuálny skladový systém bol realizovaný vo Factory I/O, ktorý bude figurovať ako OPC UA klient. Na prepojenie mobilnej aplikácie s virtuálnym skladovým systémom je využívaný middleware Node-RED. Node-RED bude figurovať ako OPC UA klient. S mobilnou aplikáciou bude Node-RED komunikovať cez MQTT protokol. Táto prípadová štúdia je opísaná s využitím diplomovej práce [76].

Architektúra celého systému je navrhnutá na základe technológií, ktoré sú ľahko dostupné. Návrh je znázornený na obrázku č. 136.



Obrázok č. 136: Architektúra aplikačného systému štvrtej prípadovej štúdie

Medzi tri hlavné komponenty patrí virtuálny skladový systém, softPLC a mobilná aplikácia. Na počítači s operačným systémom Windows máme nainštalovaný softvér Codesys a simulátor Factory I/O. Factory I/O poskytuje vopred pripravenú scénu so skladovým systémom, ktorý máme za úlohu riadiť. Na riadenie skladového systému využívame Codesys, ktorý okrem vývojového prostredia pre PLC program zabezpečuje aj softPLC bežiace na počítači. Codesys runtime (figuruje ako OPC UA server) a Factory I/O budú komunikovať prostredníctvom OPC UA protokolu modelom klient-server.

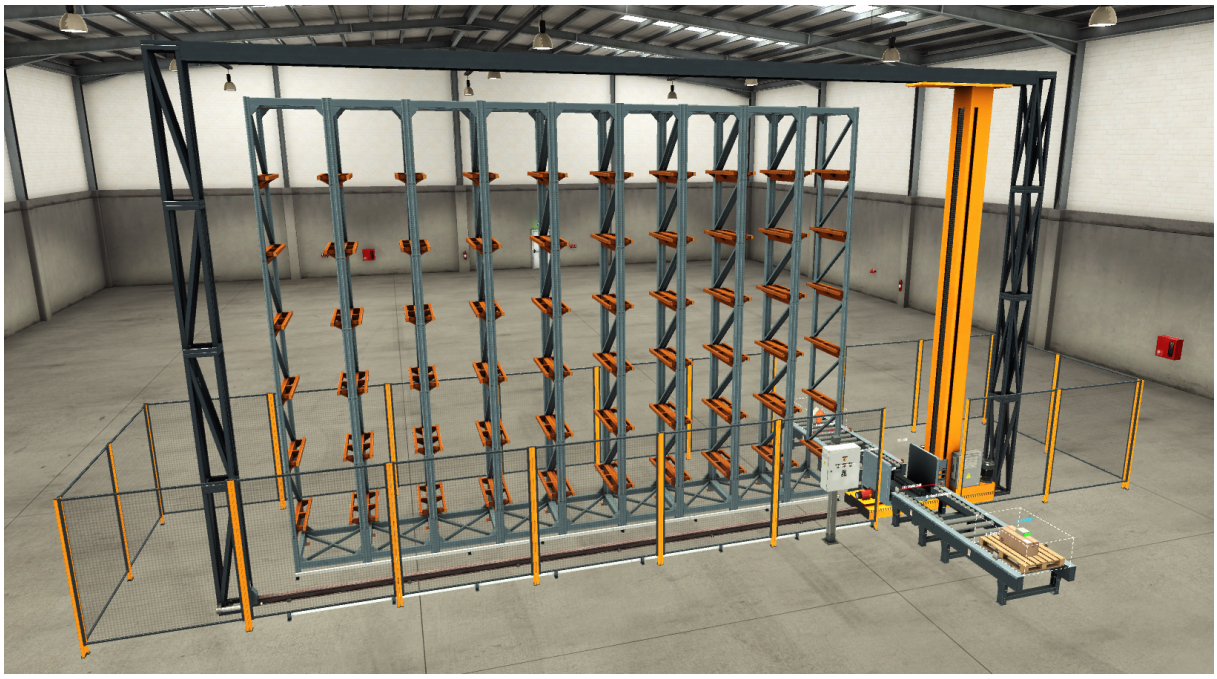
Aby sme mohli do tejto zostavy pridať aj mobilnú aplikáciu, využívame middleware

Node-RED. V Node-REDe máme ďalšieho OPC UA klienta, ktorý sa napája na OPC UA server. Dáta medzi mobilnou aplikáciou a Node-REDom sa vymieňajú prostredníctvom protokolu MQTT. Využije sa HiveMQ, čo je MQTT broker ktorý spracúva údaje a posiela ich ďalej k daným MQTT klientom.

4.4.1 Špecifikácia a správanie diskrétneho udalostného systému

Využívaným diskrétnym udalostným systémom je virtuálny automatizovaný skladový systém od tvorcov softvéru Factory I/O (obrázok č. 137).

Model pozostáva z elektrického rozvádzača s tlačidlami, vysielača (angl. emitter), dopravníkového pásu (ide o model s valcami pre ťažšie predmety), nakladacieho pásu, retroreflexných senzorov, aktuátorov, koľajnicového stohovacieho žeriavu a skladu. Koľajnicový stohovací žeriav obsahuje vozík, vertikálnu platformu, dve vidlice a skladové miesto.



Obrázok č. 137: 3D model skladového systému [76]

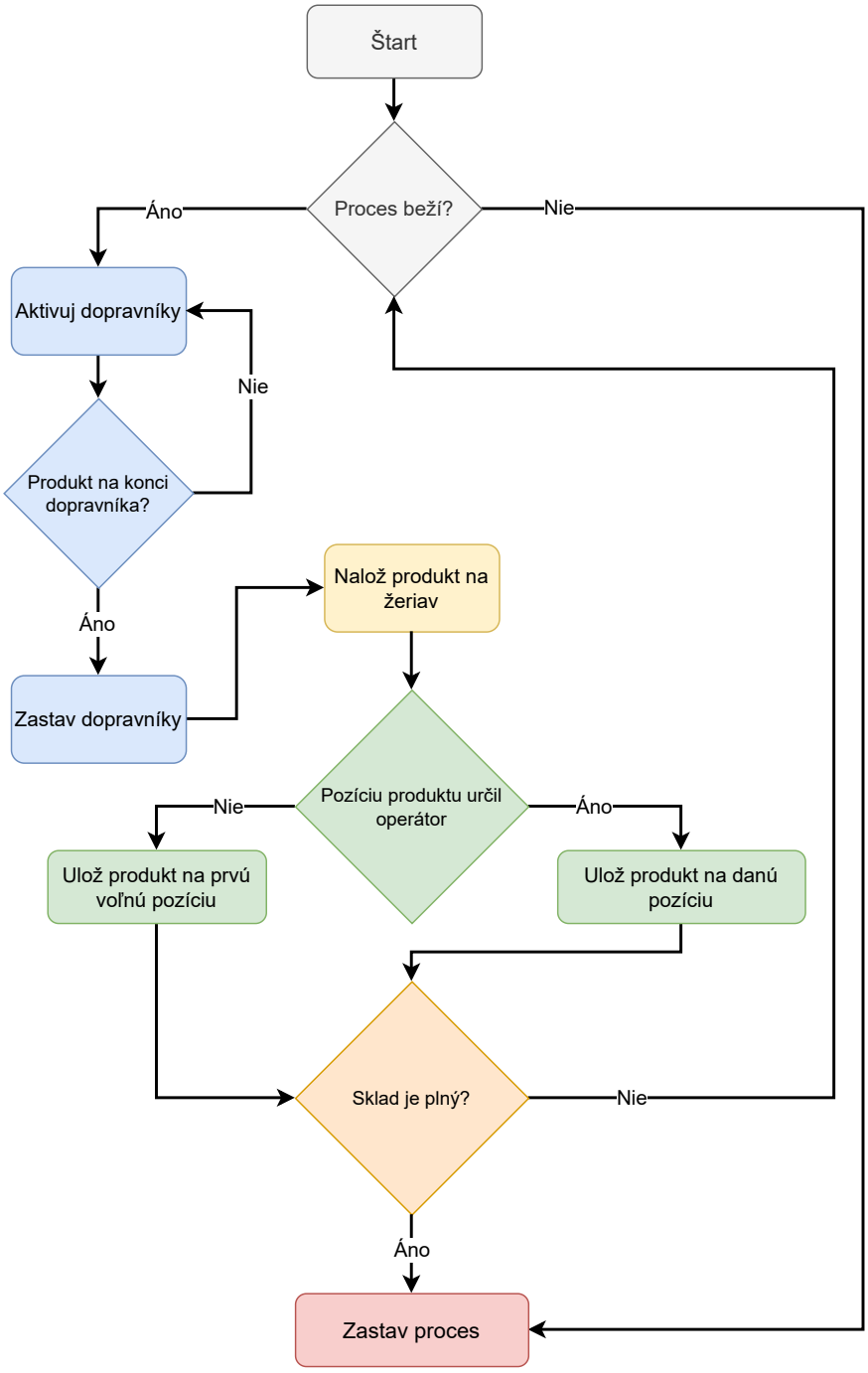
Na vertikálnej platforme a na vozíku sa nachádzajú dva laserové diaľkomery, ktoré merajú horizontálnu a vertikálnu polohu platformy. Stohovací žeriav je možné ovládať digitálnymi, numerickými a analógovými hodnotami podľa zvolenej konfigurácie. Pre jednoduchosť a názornosť sú v našom prípade nastavené numerické hodnoty. Polohu žeriavu určuje premenná s názvom *TargetPosition*. Jeho začiatočná hodnota je číslo 55. Celkový počet buniek v sklade je 54. Toto znamená, že cieľová bunka v sklade a zároveň aj poloha žeriavu môže byť definovaná celočíselnou hodnotou medzi 1 a 54. Ak je táto hodnota nastavená na nulu, stohovací žeriav sa zastaví v aktuálnej polohe. Ak je však

vyššia ako 54, žeriav sa vráti do začiatočnej polohy, teda na číslo 55.

Cieľom riadenia je pomocou spomenutých dielov výrobok umiestniť do skladu. Proces začína vygenerovaním výrobku z vysielača (emitter). Akonáhle sa produkt dostaví na pás, retroreflexný senzor ho rozpozná a čaká sa na operátora, aby zatlačil tlačidlo Štart. Stlačením tohto tlačidla sa aktivuje valcový pás a produkt sa posunie k nakladaciemu pásu. Na konci nakladacieho pásu je ďalší senzor, ktorý keď rozpozná výrobok, tak zastaví oba pásy a nasleduje naloženie výrobku na žeriav. Akonáhle žeriav naloží výrobok do vidlice, nasleduje transport výrobku do skladu. Ako už bolo spomenuté, žeriav je v tomto prípade ovládaný numerickými hodnotami. Do premennej aktuátora *TargetPosition* zašleme hodnotu z intervalu 1 až 54 a týmto určíme polohu žeriavu a pozíciu výrobku v sklade.

Jednou z požiadaviek tejto časti riadenia je, aby výber skladového miesta bol automatický alebo manuálny. Pokiaľ operátor neurčí pozíciu produktu v sklade sám, program, pomocou ktorého riadime diskrétny udalostný systém, automatický určí prvú voľne dostupnú pozíciu v sklade a tam výrobok umiestni. Keď žeriav produkt uskladní, vráti sa späť na začiatočnú polohu. Tento proces opakujeme, pokiaľ sa všetky bunky v sklade neobsadia alebo pokiaľ operátor nezatlačí tlačidlo Stop.

Pre lepšiu názornosť sme správanie skladového systému vyjadrili pomocou vývojového diagramu — obrázok č. 138.

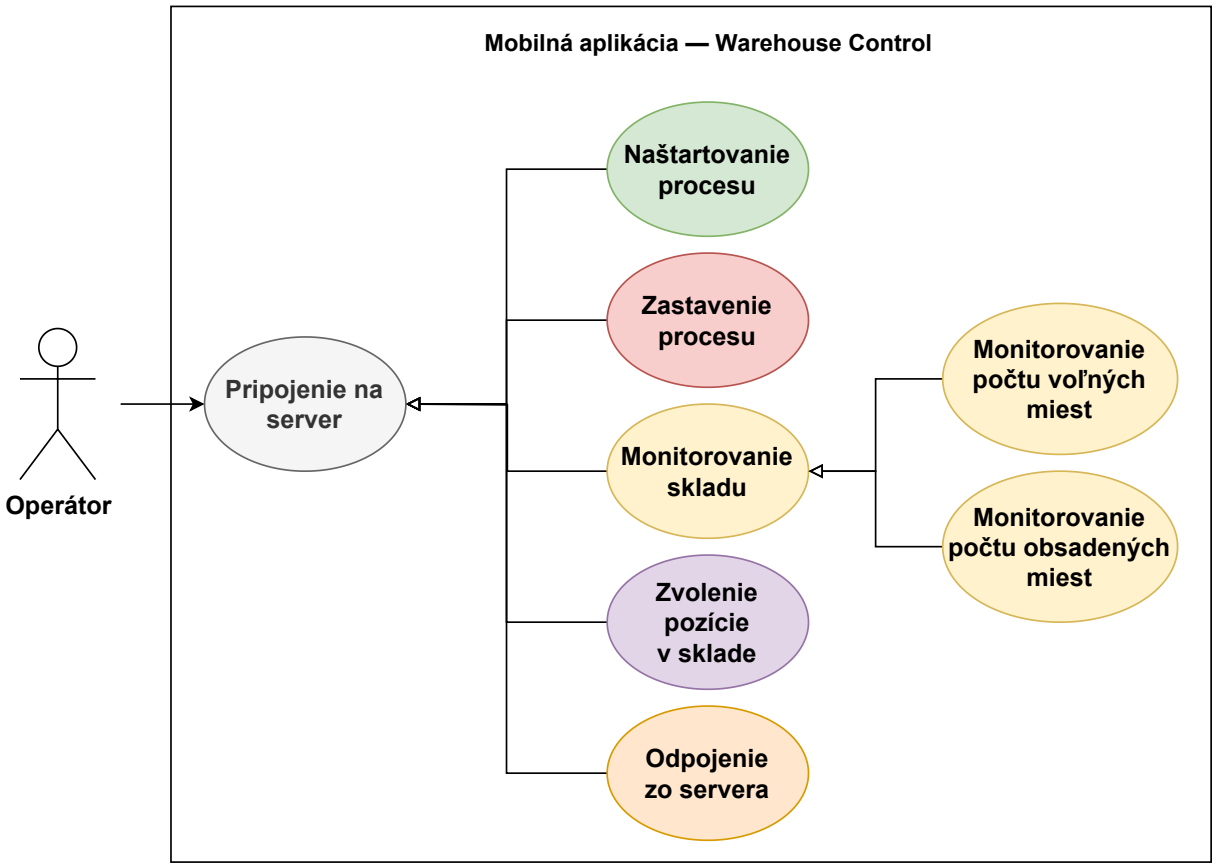


Obrázok č. 138: Vývojový diagram skladového systému [76]

4.4.2 Špecifikácia mobilnej aplikácie

Ako už bolo spomínané, operátor riadi a monitoruje sklad s využitím mobilnej aplikácie. To znamená, že v mobilnej aplikácii je potrebné mať možnosť naštartovania procesu stlačením tlačidla Štart, zastavením procesu stlačením tlačidla Stop a proces celkom ukončiť odpojením sa zo servera. Taktiež má operátor možnosť monitorovať počet voľných miest v sklade, ako aj počet produktov, ktoré sa v sklade nachádzajú. Poslednou požiadavkou je, aby mal možnosť si vybrať, na akú pozíciu chce produkt do skladu

umiestniť. Diagram prípadov použitia, ktorý je znázornený na obrázku č. 139, opisuje všetky požiadavky týkajúce sa mobilnej aplikácie.

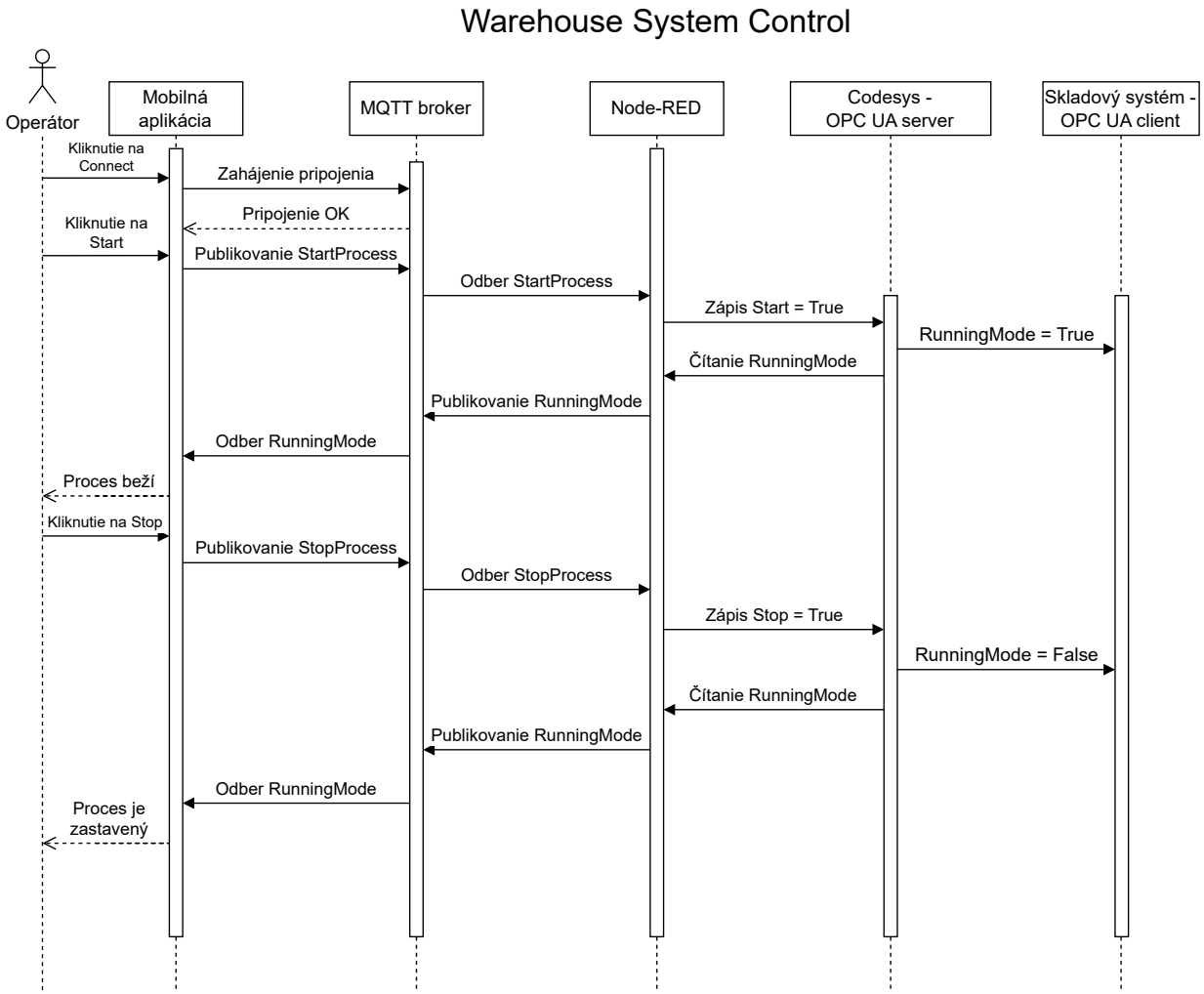


Obrázok č. 139: Diagram prípadov použitia mobilnej aplikácie [76]

4.4.3 Komunikácia jednotlivých subsystémov

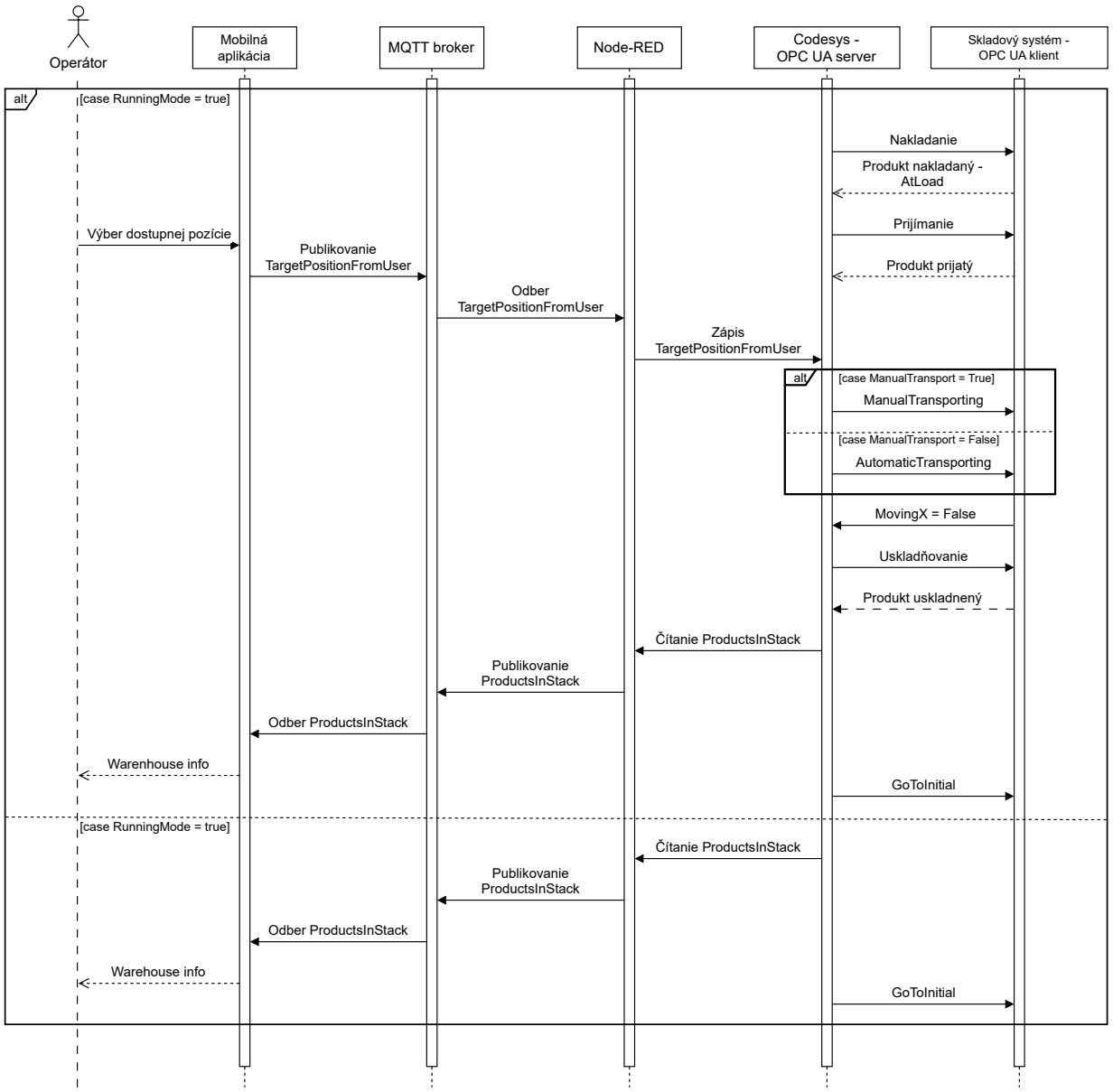
Veľmi podstatným prvkom v celom systéme je komunikácia. Na základe dostupných technológií a možností sme navrhli také riešenie, ktoré zabezpečí rýchly prenos údajov medzi systémami a umožní operátorovi sledovať aktuálny stav skladových priestorov.

Proces začína od operátora, ktorý sa najprv pripojí na MQTT broker HiveMQ. Akonáhle sa mu to podarí, má možnosť naštartovať proces zatlačením tlačidla Start. Hodnota z tlačidla sa najprv zašle na MQTT broker a MQTT klient v Node-RED túto zmenu zachytí tým, že vykoná subscribe na danú tému (*topic*), pod ktorou je hodnota zapísaná/vysielaná. Následne sa táto hodnota prepošle prostredníctvom OPC UA klienta v Node-REDe na OPC UA server v Codesyse. Akonáhle OPC UA server v Codesyse túto hodnotu zapíše, prepošle ju ďalej k OPC UA klientovi vo Factory I/O. Táto časť komunikácie je znázornená na obrázku č. 140.



Obrázok č. 140: Sekvenčný diagram — prípad č.1 [76]

Na obrázku č. 141 je znázornená komunikácia v dvoch prípadoch: keď proces beží a keď je zastavený. Ak proces beží, operátor má možnosť vybrať si pozíciu v sklade, kam chce výrobok uskladniť. Akonáhle skladový systém vo Factory I/O tento výrobok uskladní, zašle o tom informáciu späť operátorovi. V prípade, že proces nebeží, operátor vidí iba to, aký je aktuálny stav v sklade a má možnosť proces opäť naštartovať.



Obrázok č. 141: Sekvenčný diagram — prípad č.2 [76]

4.4.4 Riadenie diskrétneho udalostného systému

Riadenie skladového systému prebieha prostredníctvom softPLC v Codesyse. Ako už bolo spomenuté, Codesys je softvérový systém zložený z dvoch častí: integrované vývojové prostredie (IDE) a runtime. Integrované vývojové prostredie sme využili na implementáciu programu skladového systému.

Codesys Control Win (runtime) poskytuje virtuálne softPLC, ktoré beží na lokálnom OPC UA serveri. Beh programu sa na každé 2 hodiny vypne a je potrebné ho znovu naštartovať, čo je spôsobené licenciou, ktorá je v tomto móde zadarmo.

Z vývojového diagramu na obrázku č. 138 je možné vidieť, že celý proces skladovania je rozdelený do niekoľkých krokov. Jednotlivé kroky v procese sčasti závisia od operátora. Z tohto dôvodu riadenie členíme na **manuálne** a **automatické**. Manuálne

v tomto prípade znamená, že operátor môže naštartovať proces, ukončiť a vybrať si miesto v sklade, kam chce výrobok uskladniť.

Aplikácia v Codesyse je zložená z objektov rôznych typov. Zdrojový kód pre program v PLC je objekt typu *programovacia organizačná zložka* (*POU*). Nakoľko opisovaný proces čiastočne závisí od operátora (operátor je ten, ktorý proces naštartuje stlačením tlačidla Štart), boli vytvorené dva programy (dve POU): *Control* a *Main*.

Control POU je pomocný program, ktorý beží neustále popri hlavnom programe na ovládanie vykonávania sekvencií hlavného programu na základe vstupu od operátora.

Main POU je hlavný program, ktorý je zložený z niekoľkých krokov, ktoré riadia proces v sklade. Aby všetko toto bolo možné implementovať, prvým krokom bolo definovanie a inicializácia globálnych premenných.

Globálne premenné v PLC sú bežné premenné, ktoré sú rozpoznané v rámci celého projektu, teda je možné k ním pristupovať, čítať a meniť ich hodnoty odkiaľkoľvek v projekte. V rámci projektu boli vytvorené dva zoznamy globálnych premenných: **FIO** a **ControlPou**. Zoznam s názvom FIO obsahuje všetky globálne premenné, ktoré reprezentujú senzory a aktuátory v opisovanom skladovom systéme (obrázok č. 142). Zoznam premenných s názvom ControlPou (obrázok č. 143) obsahuje globálne premenné, ktoré budú využívané na prepájanie medzi manuálnym a automatickým riadením a taktiež na vizualizáciu aktuálneho stavu procesu v mobilnej aplikácii.

Globálna premenná *RunningMode* v zozname ControlPou je boolovského typu, ktorej hodnota bude spúšťať a zastavovať proces v hlavnom programe. Premenná s názvom *ManualTransport* je tiež boolovského typu, ktorej hodnota prepája riadenie medzi manuálnym a automatickým v hlavnom programe. Premenná *TargetPositionFromUser* ukladá hodnotu, ktorú operátor zašle prostredníctvom mobilnej aplikácie, keď chce určiť pozíciu výrobku v sklade. Je dátového typu WORD, čo je 16-bitové celé číslo bez znamienka (UInt16). Premenná *ProductsInStack* je pole veľkosti 54, ktoré reprezentuje sklad, presnejšie skladové miesta. Taktiež je dátového typu WORD. Na začiatku je pole vyplnené nulami. Nuly reprezentujú prázdne miesta v sklade. Akonáhle sa nejaký výrobok uskladní do skladu, jeho pozíciu zapíšeme do tohto poľa. Využívame ho z dôvodu, aby operátor mohol v mobilnej aplikácii vidieť, ktoré miesta v sklade sú voľné a ktoré sú už obsadené. A taktiež v prípade, keď operátor neurčí danú pozíciu, aby program automaticky vyhľadal prvé voľné miesto a tam výrobok uskladnil.

FIO

```
VAR_GLOBAL
    //sensors
    AtEntry:BOOL;
    AtExit:BOOL;
    AtLeft:BOOL;
    AtLoad:BOOL;
    AtMiddle:BOOL;
    AtRight:BOOL;
    AtUnload:BOOL;
    Auto:BOOL;
    EmergencyStop:BOOL;
    Maual:BOOL;
    MovingX:BOOL;
    MovingZ:BOOL;
    Reset:BOOL;
    Start:BOOL;
    Stop:BOOL;

    //actuators
    EntryConveyor:BOOL:=FALSE;
    ExitConveyor:BOOL:=FALSE;
    ForksLeft:BOOL:=FALSE;
    ForksRight:BOOL:=FALSE;
    Lift:BOOL:=FALSE;
    LoadConveyor:BOOL:=FALSE;
    ResetLight:BOOL:=FALSE;
    StartLight:BOOL:=FALSE;
    StopLight:BOOL:=FALSE;
    TargetPosition:WORD:=0;
    UnloadConveyor:BOOL:=FALSE;
END_VAR
```

Obrázok č. 142: Zoznam globálnych premenných pre riadenie skladového systému vo Factory I/O [76]

ControlPou

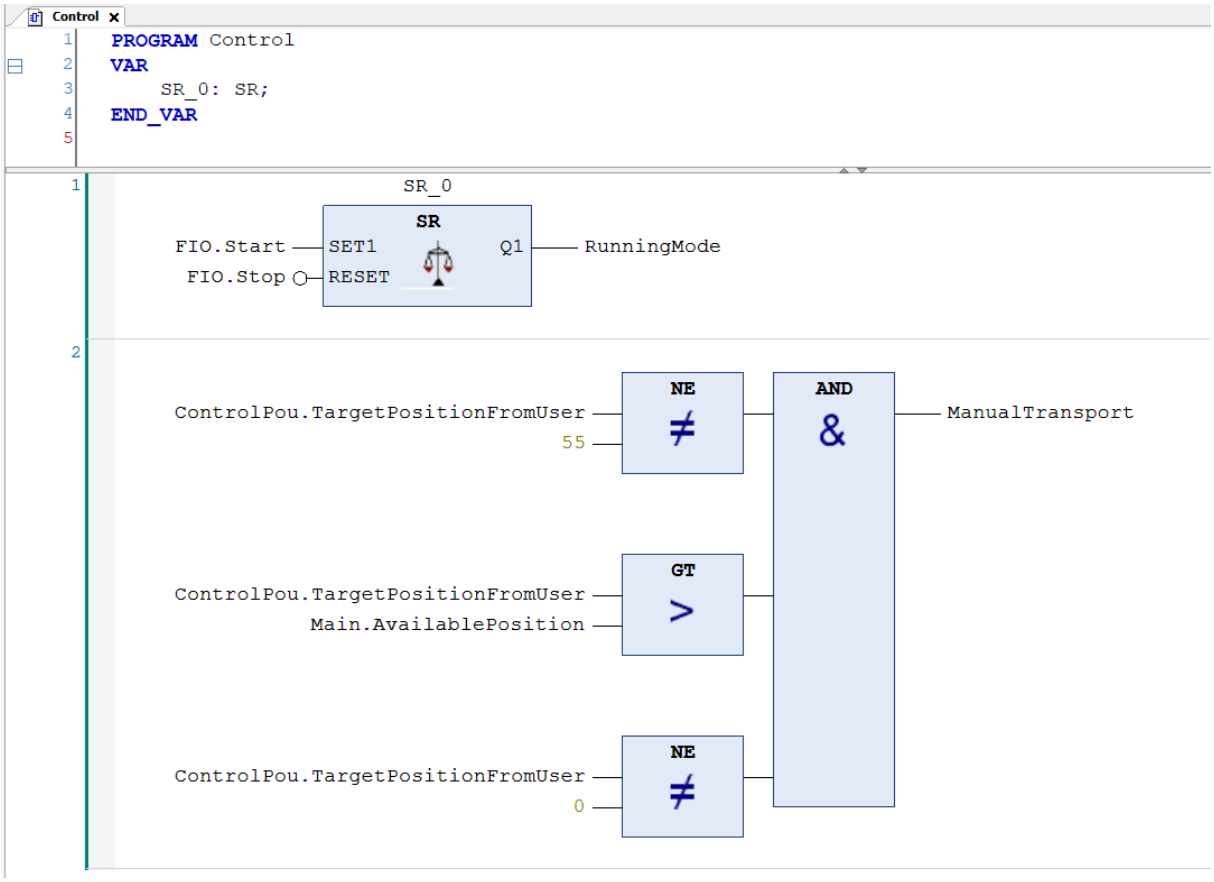
```
VAR_GLOBAL
    RunningMode: BOOL;
    ManualTransport: BOOL;
    //additional variable to control position
    TargetPositionFromUser: WORD := 0;
    ProductsInStack: ARRAY[0..53] OF WORD;
END_VAR
```

Obrázok č. 143: Zoznam globálnych premenných pre prepojenie medzi manuálnym a automatickým riadením [76]

POMOCNÝ PROGRAM — CONTROL

Na implementáciu pomocného programu Control (obrázok č. 144) sme využili programovací jazyk Function Block Diagram (FBD), nakoľko tu riadime dva procesy, ktoré závisia iba od vstupu od operátora, čo sa prehľadne dá v tomto jazyku znázorniť.

V prvom bloku programu je nastavovaná hodnota globálnej premennej *RunningMode*, ktorá je boolovského typu. Jej hodnota sa nastaví na základe hodnoty z tlačidla Štart alebo Stop. Na nastavenie hodnoty používame **SR blok**. SR blok má dva vstupy Set a Reset a výstup Q. Aktivácia výstupu Q je určená hodnotou zo vstupu Set, čo je v opisovanom prípade hodnota tlačidla Start. Deaktivácia výstupu je určená hodnotou zo vstupu Reset, čo je hodnota z tlačidla Stop. Tu ale bolo nutné do vstupu Reset poslať negovanú hodnotu, lebo predvolená hodnota tlačidla Stop je TRUE. Pod aktiváciou a deaktiváciou výstupu je možné rozumieť priradenie hodnôt TRUE a FALSE do premennej *RunningMode*. Hodnota z premennej *RunningMode* následne štartuje a zastavuje proces v hlavnom programe.



Obrázok č. 144: Pomocný program Control [76]

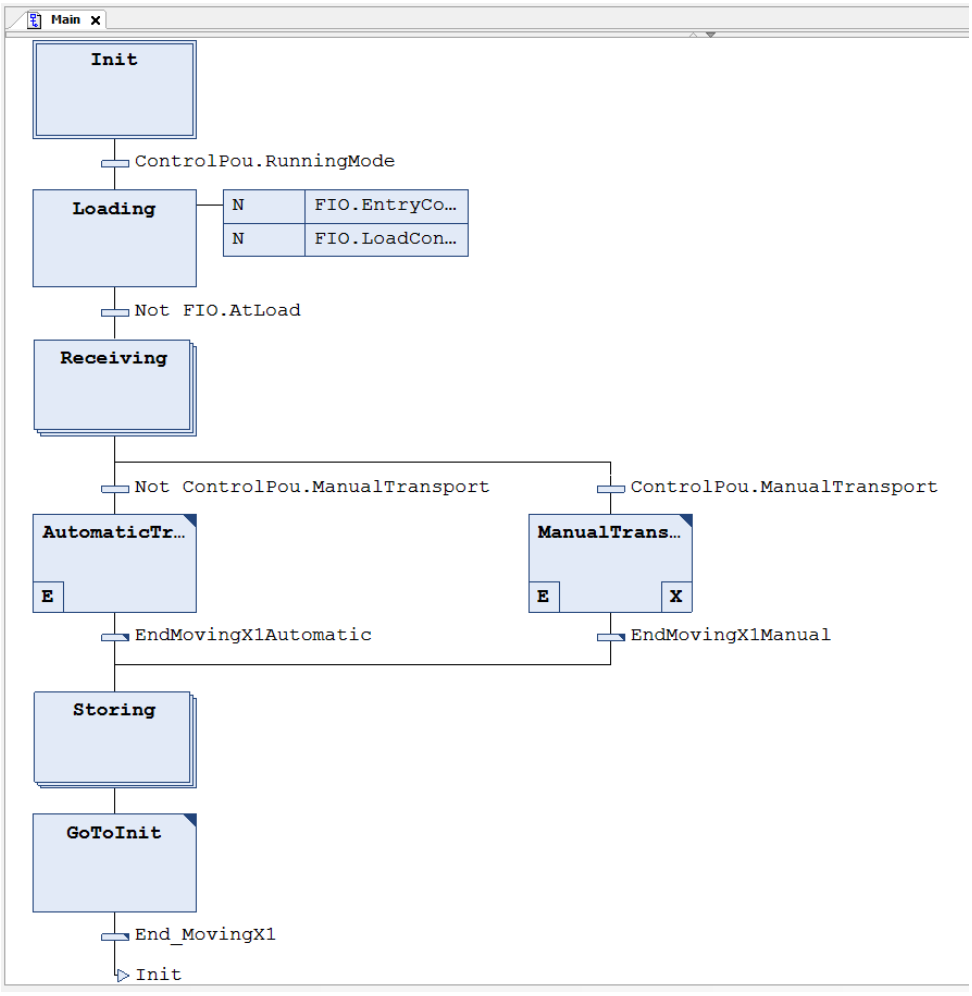
V druhom bloku programu je nastavovaná hodnota globálnej premennej *ManualTransport*, ktorá je boolovského typu. Na nastavenie hodnoty je využívaný boolovský

operátor AND s tromi vstupmi. Vstupy sú matematické operátory. Dva sú pre porovnanie nerovnosti a jeden operátor porovnania „väčší ako“. Tu ošetrujeme hodnoty, ktoré poslal užívateľ z mobilnej aplikácie a ak všetky operátory spĺňajú podmienku (teda majú výstupnú hodnotu TRUE) globálna premenná *ManualTransport* nadobudne hodnotu TRUE. V opačnom prípade jej hodnota bude FALSE. Pomocou tejto premennej sa bude ovládať prepínanie medzi automatickým a manuálnym riadením v hlavnom programe.

HLAVNÝ PROGRAM — MAIN

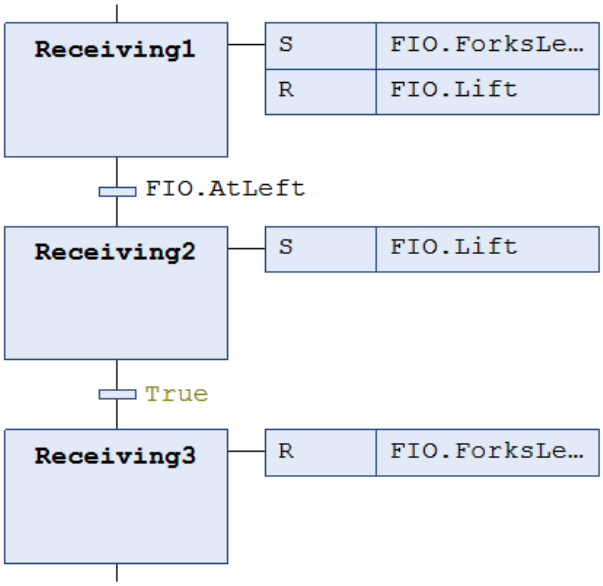
Riadenie hlavného programu možno rozdeliť do niekoľkých krokov. Tieto kroky predstavujú dané procesy v skladovom systéme, ktoré sa vykonávajú postupne jeden za druhým. Celý proces v skladovom systéme sa teda vykonáva sekvenčne. Z tohto dôvodu sme sa rozhodli na implementáciu programu využiť programovací jazyk Sekvenčný funkčný diagram (SFC).

Program sa skladá zo 7 blokov (obrázok č. 145).



Obrázok č. 145: Hlavný program Main [76]

Na začiatku je prítomný inicializačný blok *Init*, ktorý prejde do ďalšieho kroku, keď hodnota globálnej premennej *RunningMode* bude TRUE. V ďalšom kroku je prítomný blok *Loading*, ktorý aktivuje valcový a výrobný pás. Na aktiváciu pásov boli využité dve IEC akcie. Tieto akcie sa vykonajú iba, keď je blok *Loading* aktívny. V tomto prípade to znamená, že sa blok deaktivuje a obidva pásy sa zastavia, keď výrobok príde po koniec nakladacieho pásu, kde ho retroreflexný senzor *AtLoad* zdeteguje. Nasledovným krokom je blok *Receiving*. Tu bolo využité makro a jednotlivé bloky sme zabalili, nakoľko sa krok pre naloženie výrobku na žeriav skladá z troch častí (obrázok č. 146).



Obrázok č. 146: Makro pre Receiving Step [76]

Najprv sa aktivuje ľavá vidlica žeriavu (aktuátor *ForksLeft*), resp. vysunie sa dopredu, a nakoľko vidlica musí byť pod výrobkom, aktuátor *Lift* musí byť deaktivovaný. Na túto funkcionalitu opäť využívame dve IEC akcie. Akonáhle senzor na žeriave *AtLeft* zdeteguje vysunutie vidlice, prechádza sa do ďalšieho bloku a tu sa iba opäť aktivuje aktuátor *Lift*, aby výrobok zodvihol. Akonáhle je výrobok zdvihnutý, prejde sa do ďalšieho bloku a resetne sa ľavý aktuátor *ForksLeft*. Ďalším krokom je transport výrobku do skladu. Toto bolo vyriešené pomocou alternatívnej vetvy, ktorá obsahuje dva bloky *AutomaticTransporting* a *ManualTransporting*. Iba jeden z nich sa aktivuje, a to v závislosti od toho, aká je hodnota globálnej premennej *ManualTransport*, ktorú nastavujeme v pomocnom programe Control.

Na vstupe do bloku *AutomaticTransporting* je akcia *AutomaticTransport_entry*, ktorá bola implementovaná pomocou štruktúrovaného textu (Programový modul č. 5).

Najprv si je nutné v slučke skontrolovať, či sa hodnota premennej *AvailablePosition*

Programový modul č. 5 Akcia AutomaticTransport_entry [76]

```
1  //Loop to check if AvailablePosition is already in ProductsInStack array
2  REPEAT
3      found:=FALSE;
4      FOR i:=0 TO 53 DO
5          IF ControlPou.ProductsInStack[i] = AvailablePosition THEN
6              found:=TRUE;
7              EXIT;
8          END_IF
9      END_FOR
10     IF found THEN
11         AvailablePosition:=AvailablePosition+1; //If AvailablePosition already exists
12         // in array, increase by 1
13     END_IF
14 UNTIL NOT found
15 END_REPEAT
16
17 //Store AvailablePosition in ProductsInStack array
18 FOR i:=0 TO 53 DO
19     IF ControlPou.ProductsInStack[i] = 0 THEN //CheckForEmptySlotIn ProductsInStack array
20         ControlPou.ProductsInStack[i]:=AvailablePosition;
21         EXIT;
22     END_IF
23 END_FOR
```

už nachádza v poli *ProductsInStack*. Akonáhle je nájdená, premenná *AvailablePosition* sa zvýši o 1, skontroluje sa prvé voľne dostupné miesto v sklade (v poli *ProductsInStack*) a na tú pozíciu sa zapíše hodnota z premennej *AvailablePosition*. Po skončení tejto akcie nasleduje akcia *AutomaticTransport_active*, ktorá je tiež implementovaná pomocou štruktúrovaného textu (Programový modul č. 6).

Programový modul č. 6 Akcia AutomaticTransport [76]

```
1 FIO.TargetPosition := AvailablePosition;
2 ControlPou.ProductsInStack[AvailablePosition - 1] := AvailablePosition;
```

V tejto akcii je najprv do globálnej premennej *TargetPosition* priradená hodnota z premennej *AvailablePosition* a potom táto rovnaká hodnota je priradená aj do poľa *ProductsInStack* na danú pozíciu. Pre ukončenie tohto bloku, ako podmienku pre prechod do nasledovného kroku, máme k dispozícii funkčný blok *F_TRIG*, ktorého hodnota bude TRUE akonáhle sa vozík na žeriave zastaví na danej pozícii (resp. hodnota senzora pohybu *MovingX* bude FALSE).

Na vstupe do bloku *ManualTransporting* sa nachádza akcia *ManualTransport_entry*. Akcia je opäť implementovaná pomocou štrukturovaného textu (programový modul č. 7).

Programový modul č. 7 Akcia *ManualTransport_entry* [76]

```
1 IF ControlPou.ProductsInStack[ControlPou.TargetPositionFromUser - 1] = 0 THEN
2         ControlPou.ProductsInStack[ControlPou.TargetPositionFromUser - 1] :=
3     ControlPou.TargetPositionFromUser;
4 END_IF
```

V tejto akcii ošetríme, či hodnota premennej *TargetPositionFromUser* (teda hodnota, ktorú si zvolil operátor) sa nenachádza v poli *ProductsInStack*. Ak nie, tak ju tam pridáme. Následne sa vykoná akcia *ManualTransport_active*, v ktorej je iba presunutá hodnota z globálnej premennej *TargetPositionFromUser* do aktuátora *TargetPosition*. Na výstupe z bloku máme akciu *ManualTransport_exit*, kde je iba vynulovaná hodnota z globálnej premennej *TargetPositionFromUser*, aby sa resetla hodnota globálnej premennej *ManualTransport* na FALSE. Obidve akcie *ManualTransport_active* a *ManualTransport_exit* obsahujú jeden operátor MOVE na priradenie daných hodnôt.

Opäť, na ukončenie tohto bloku ako podmienku pre prechod do nasledovného kroku je využitý funkčný blok *F_TRIG*.

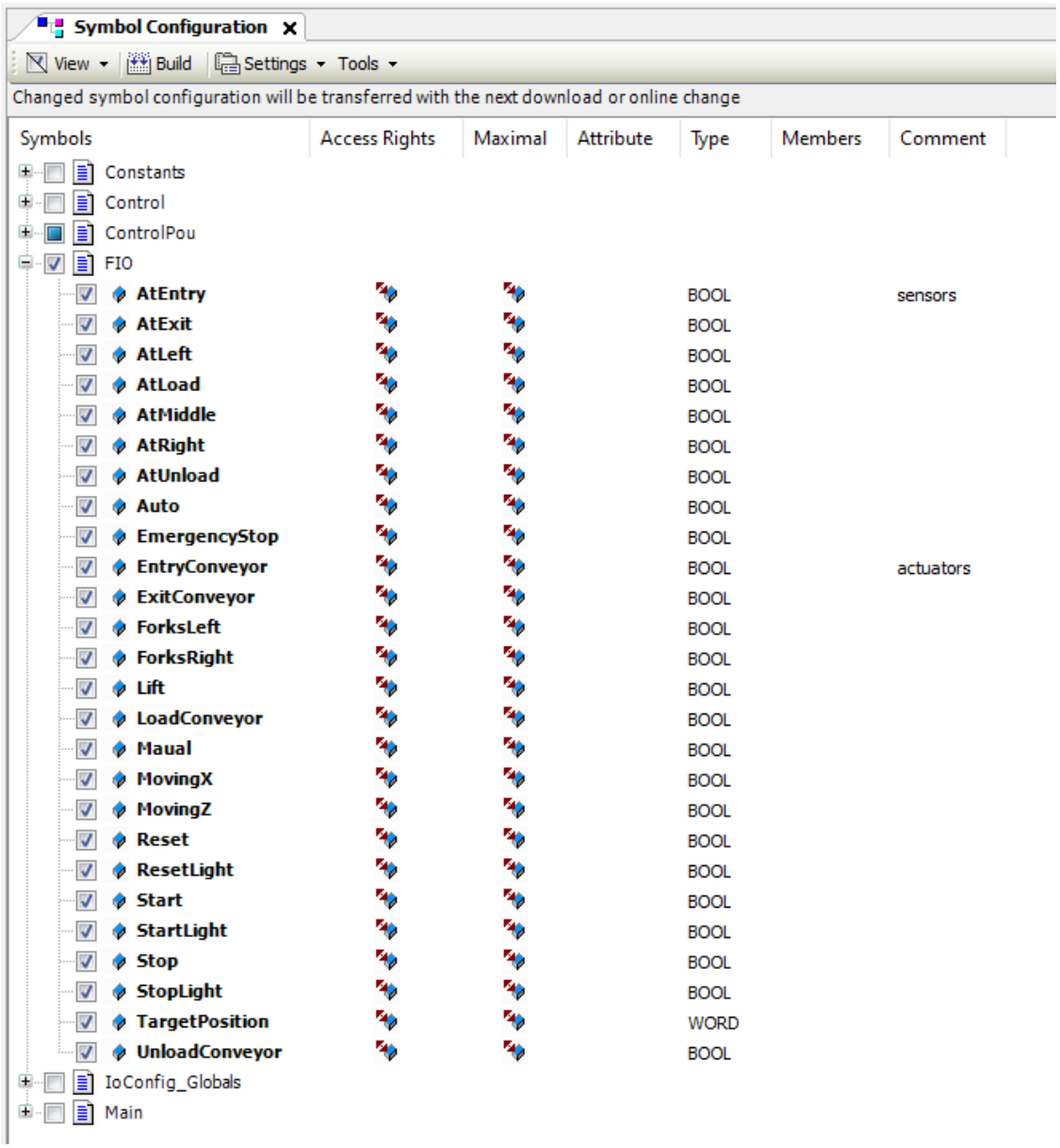
Ako ďalší krok je blok *Storing*, ktorý je inverzný od bloku *Receiving*. Akonáhle je

výrobok vložený do skladu, prejdeme do posledného bloku *GoToInitial*, ktorý má na vstupe akciu *GoToInitial_active*, kde je iba nastavená hodnota globálnej premennej *Target-Position* na 55, aby sa žeriav vrátil späť na začiatočnú pozíciu. Akonáhle sa vozík zastaví (hodnota senzora pohybu *MovingX* bude FALSE) opäť sa aktivuje inicializačný blok *Init* a proces sa opakuje.

4.4.5 Konfigurácia komunikácie medzi softPLC a Factory I/O

Ďalším krokom v implementácii riadenia virtuálneho skladového systému je konfigurácia komunikácie medzi softPLC (Codesys runtime) a Factory I/O. Komunikácia prebieha prostredníctvom protokolu OPC UA, nakoľko oba softvéry podporujú možnosť takejto komunikácie.

Komunikačný model OPC UA v tomto prípade je klient-server. OPC UA klient je Factory I/O a OPC UA server je Codesys runtime. Na konfiguráciu komunikácie je potrebné najprv vytvoriť adresný priestor v OPC UA serveri. Tu je žiadúce, aby adresný priestor obsahoval všetky globálne premenné, ktoré boli zadefinované v zozname FIO (obrázok č. 142). Na tento účel je treba pridať do aplikácie objekt *Symbol Configuration* (obrázok č. 147). Z tohto objektu je nutné vybrať zoznam FIO a označiť všetky premenné. Tieto premenné budú tak dostupné pre OPC UA klienta.

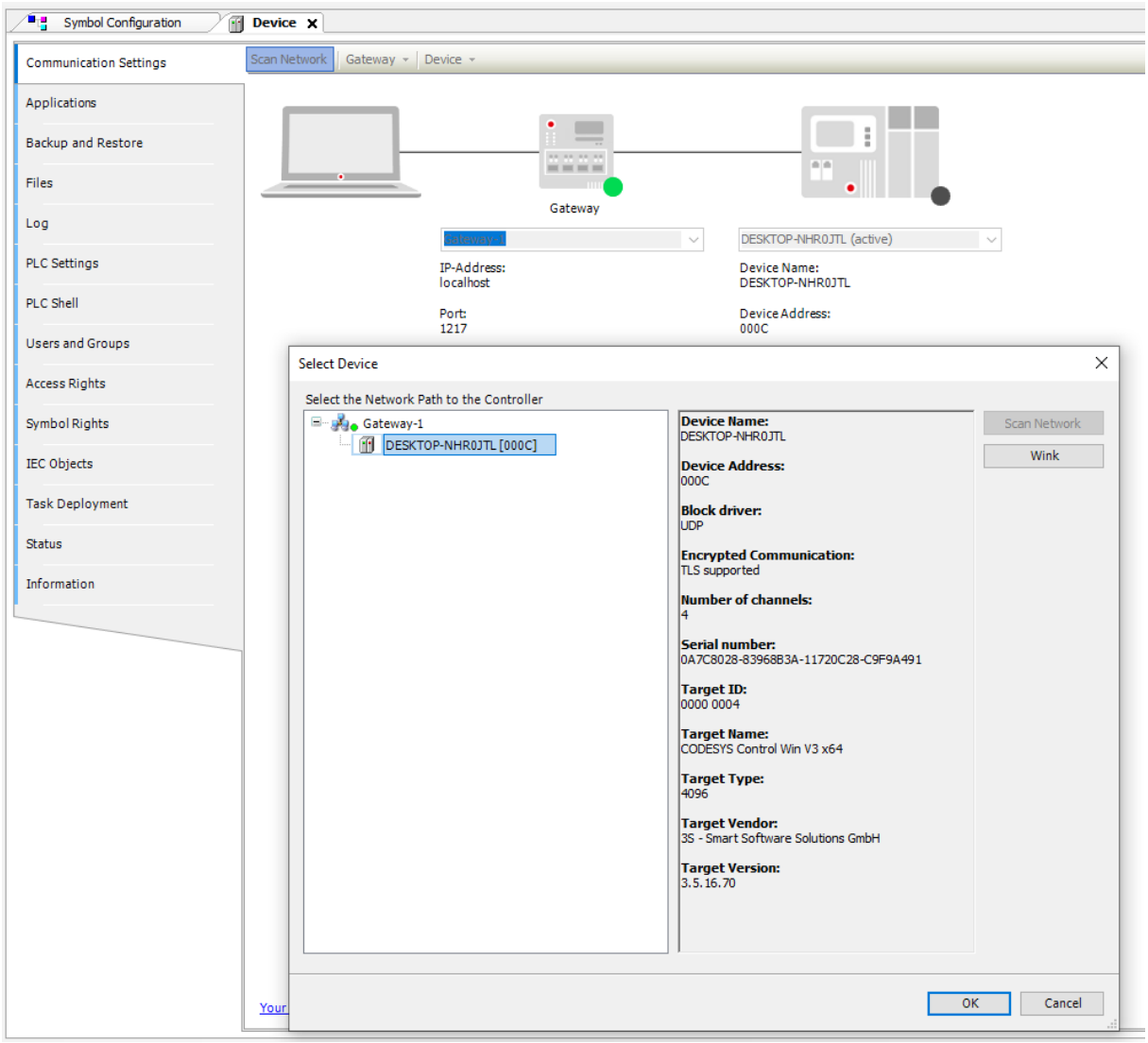


Obrázok č. 147: Objekt Symbol Configuration [76]

Následne nakonfigurujeme zariadenie, čo v tomto prípade bude softPLC. V časti *Communication settings* najprv naskenujeme sieť zatlačením tlačidla *Scan Network* a vyberieme zariadenie z ponuky na obrázku č. 148. Program, ktorý bol vytvorený, nahráme do softPLC zatlačením tlačidla *Build* a projekt následne spustíme.

Ďalším krokom je konfigurácia OPC UA klienta vo Factory I/O. V scéne *Automated Warehouse*, čo je model skladového systému, je potrebné z ponuky ovládačov vybrať *OPC Client DA/UA*. Následne, v časti *Configuration* je potrebné vybrať najprv UPC UA server, na ktorý sa chceme pripojiť, a do poľa pre filtrovanie uzlov z adresného priestoru, je potrebné zadať FIO (nakoľko chceme pristupovať k premenným iba z tohto zoznamu). Akonáhle sa pripojíme na daný OPC UA server, všetky senzory a aktuátory namapujeme

(napárujeme) na premenné, ktoré boli získané zo zoznamu FIO (obrázok č. 149). Týmto bola nakonfigurovaná OPC UA komunikácia medzi softPLC a skladovým systémom vo Factory I/O.



Obrázok č. 148: Výber zariadenia [76]

Obrázok č. 149: Mapovanie senzorov a aktuátorov vo Factory IO na OPC UA premenné [76]

4.4.6 Implementácia komunikácie v Node-RED

Ďalším krokom v implementačnej časti je vytvorenie spojenia medzi virtuálnym skladovým systémom a mobilnou aplikáciou. Preto sme využili middleware Node-RED. Na napojenie na OPC UA server z Node-REDu sú potrebné uzly, ktoré podporujú takýto typ komunikácie. Na tento účel bolo potrebné nainštalovať knižnicu *node-red-contrib-opcua*. Na napojenie sa na MQTT broker (v našom prípade je to cloudový broker HiveMQ), sú tiež potrebné uzly pre MQTT protokol. Node-RED predvolene podporuje takýto typ komunikácie, keďže je častou voľbou pre IoT riešenia, a teda nebolo potrebné nič inštalovať.

V Node-RED editore máme jeden tok (flow), ktorý v sebe obsahuje dva typy pre-

pojenia: *OPC UA — MQTT* a *MQTT — OPC UA*.

KOMUNIKÁCIA OPC UA — MQTT

Implementácia komunikácie *OPC UA — MQTT* je znázornená na obrázku č. 150.

Na prijímanie údajov z OPC UA servera je využitý uzol *OPC UA Client* s nasledovnými nastaveniami:

- Endpoint — URL na OPC UA server (v našom prípade ide o *//localhost:4840*);
- Action — pre príjem dát z OPC UA servera ako akciu nastavíme Subscribe.

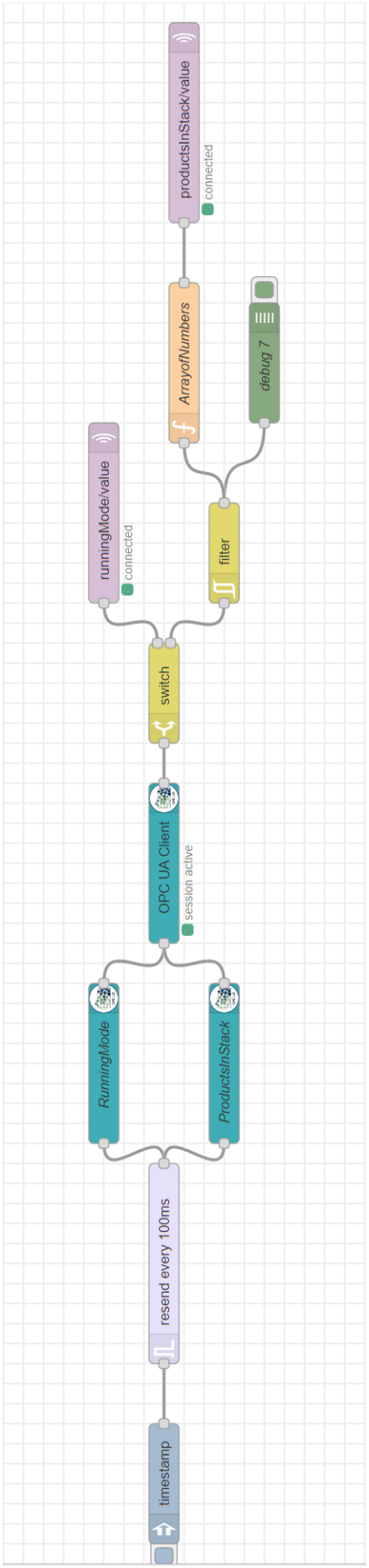
Vstupom do uzla *OPC UA klient* sú dva *OPC UA Item* uzly, ktoré predstavujú uzly/premenné, ku ktorým chceme pristúpiť z OPC UA servera. V tomto prípade, keďže potrebujeme informáciu o tom, či proces beží a aký je aktuálny stav v sklade, tak pristupujeme k premenným z OPC UA serveru *RunningMode* a *ProductsInStack*. *OPC UA Item* uzly majú tieto nastavenia:

- Item — adresa premennej, ktorá sa nachádza na OPC UA Serveri (pre premennú *RunningMode* je napríklad adresa: *ns=4;s=|var|CODESYS Control Win V3 x64.Application.ControlPou.RunningMode*)
- Type — dátový typ premennej (pre premennú *RunningMode* je dátový typ Boolean a pre *ProductsInStack* je typ UInt16)

Nakoľko bola zvolená v uzle *OPC UA client* akcia Subscribe, tak je potrebné nastaviť časový interval, ako často chceme prijímať údaje z OPC UA servera. V tomto prípade sme nastavili interval na 100 ms, aby boli údaje čo najaktuálnejšie. Následne výstup z uzla *OPC UA Client* zachytávame pomocou uzla *switch*, kde pre každý topic (topic je teda adresa premennej z OPC UA servera) nastavíme správny výstup, aby sme každú hodnotu, ktorú získame zo servera, vedeli zaslať na správny topic do MQTT brokera. Pre hodnotu premennej uzla *ProductsInStack* sme ešte predtým pridali uzol *filter*, ktorý bude posielať hodnoty na MQTT broker len v prípade, ak sa naozaj nejaká zmena v skladovom systéme udiala, čo šetrí dátové spojenie.

Hodnoty na MQTT broker zasielame pomocou uzlov *MQTT Publish*, ktoré majú nasledovné nastavenie:

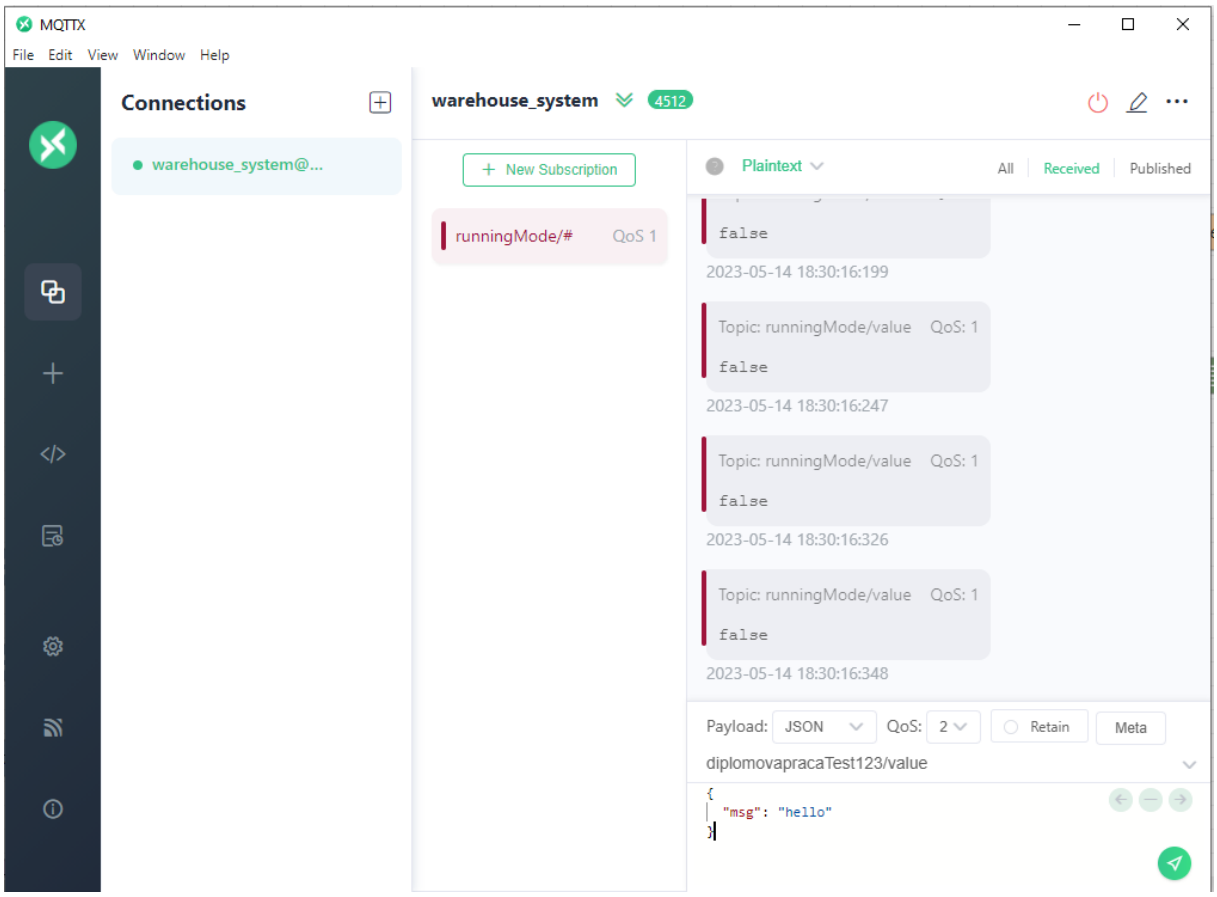
- Server — adresa servera/brokera HiveMQ (v našom prípade adresa je: *//broker.hivemq.com:1883*)
- Topic — pod akou témou (topicom) bude hodnota, ktorú posielame na broker (napr. pre hodnotu premennej *RunningMode* topic je: *runningMode/value*);



Obrázok č. 150: Implementácia komunikácie OPC UA — MQTT [76]

- QoS — nastavenie kvality služieb (pre oba uzly bolo nastaené QoS2, aby bolo isté, že hodnota bude doručená presne jeden raz);
- Security — tu bolo nastavené meno a heslo, aby dáta, ktoré posielame na broker, neboli verejné pre každého.

Pre uistenie, že sa hodnoty z OPC UA klienta zaslali na MQTT broker, bol urobený test pomocou softvéru MQTTX, kde bol vytvorený testovací MQTT klient, ktorý sa predplatil (subscribe) na topic *runningMode/value* pod daným menom a heslom. Z obrázka č. 151 je možné vidieť, že sa hodnoty naozaj poslali.

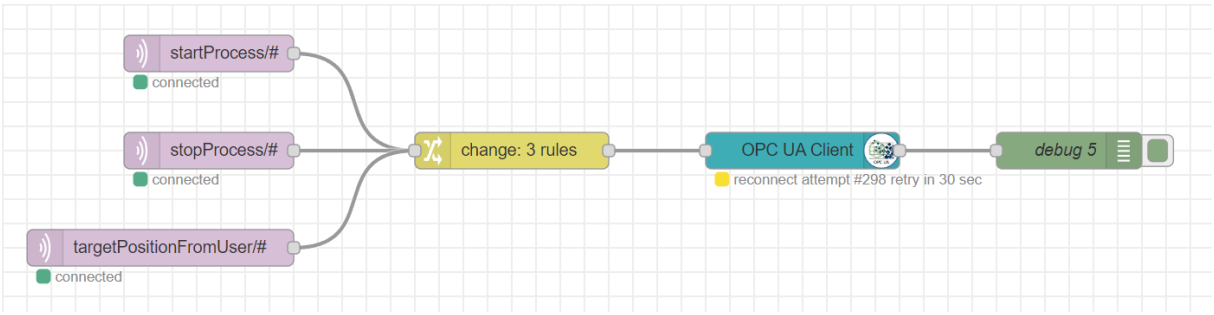


Obrázok č. 151: Testovanie komunikácie OPC UA — MQTT [76]

KOMUNIKÁCIA MQTT — OPC UA

Na posielanie dát z mobilnej aplikácie, využívame protokol MQTT s uzlom *MQTT Publish*. Dáta, ktoré operátor pošle z mobilnej aplikácie odchytávame v Node-RED pomocou MQTT uzla *Subscribe*. Hodnoty, ktoré posielame na OPC UA server, sú: *StartProcess* — na naštartovanie procesu, *StopProcess* — na zastavenie procesu a *targetPositionFromUser* — hodnota, ktorú operátor zvolil pre danú pozíciu výrobku v sklade. Ako je možné vidieť na obrázku č. 152, tak na začiatku máme tri *MQTT Subscribe* uzly, ktoré sa predplatia

(subscribe) na daný topic z MQTT brokera. Následne pomocou uzla *change* pre každý uzol správne nastavíme jeho topic. Aby sme totiž poslali údaje na server, potrebujeme vedieť adresu premennej, do ktorej hodnotu posielame. Akonáhle získame výstup z uzla *change*, presmerujeme ho na *OPC UA client*. V tomto prípade, keďže údaje zapisujeme, *OPC UA client* má ako Action nastavenú hodnotu WRITE.



Obrázok č. 152: Komunikácia MQTT — OPC UA [76]

4.4.7 Implementácia mobilnej aplikácie

Mobilná aplikácia bola realizovaná pomocou frameworku pre multiplatformový vývoj Ionic, o ktorom je uvedených viac informácií v predošlej časti tejto učebnice.

Na základe požiadaviek, ktoré boli definované a znázornené na obrázku č. 139, mobilná aplikácia obsahuje dve obrazovky, resp. dva komponenty: domovskú obrazovku (*HomeComponent*) a obrazovku WarehouseControl (*WarehouseControlComponent*).

Všetky technológie a zariadenia, ktoré boli využité pre vývoj mobilnej aplikácie, sme uviedli v tabuľkách č. 3, 4 a 5.

Tabuľka č. 3: Vývojové prostredia [76]

VÝVOJOVÉ PROSTREDIA	
Názov	Verzia
Visual Studio Code	1.78.2
Android Studio Dolphin	2021.3.1
Xcode	14.3

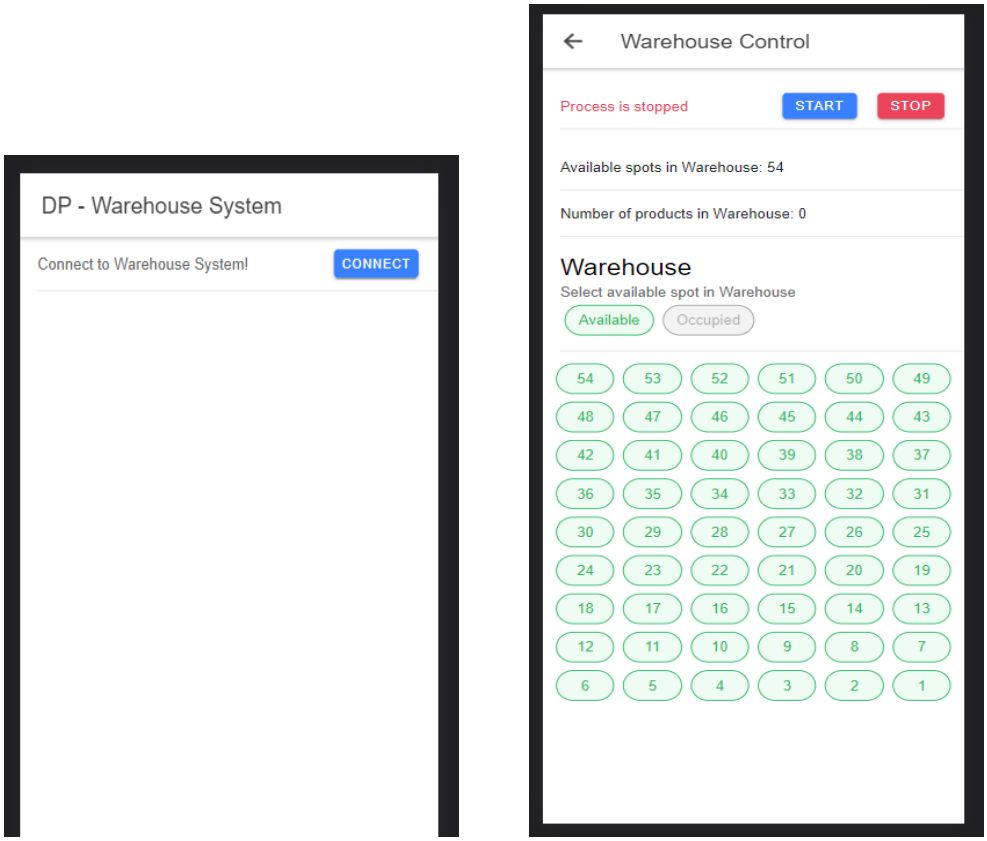
Tabuľka č. 4: Frameworky pre vývoj mobilnej aplikácie [76]

FRAMEWORK	
Názov	Verzia
Ionic	6.20.9
Angular	15.0.0

Tabuľka č. 5: Mobilné zariadenie pre vývoj mobilnej aplikácie [76]

MOBILNÉ ZARIADENIE		
Názov	OS	Opis
iPhone 7	**iOS 15.7.6**	fyzické zariadenie
Google Pixel 7	**Android 13**	fyzické zariadenie
Google Pixel 3a	**Android 13**	emulátor
iPhone 14 pro	**iOS 15.7.5**	emulátor

Domovská obrazovka (obrázok č. 153) sa načíta ako prvá a obsahuje iba jedno tlačidlo, pomocou ktorého sa operátor napojí na MQTT broker. Ak je konekcia úspešná, načíta sa obrazovka WarehouseControl.



(a) Domovská obrazovka (b) WarehouseControl obrazovka

Obrázok č. 153: Obrazovky v mobilnej aplikácii

Obrazovka WarehouseControl (obrázok č. 153 vpravo) obsahuje dve tlačidlá: *Start* a *Stop* a informáciu o tom, či proces beží. Ďalej tiež obsahuje informácie o počte voľných a obsadených miest v sklade. Ďalej obsahuje schematickú vizualizáciu skladového systému, kde má operátor možnosť si zvoliť miesto v sklade, kam chce výrobok uskladniť.

Nakoniec, má možnosť sa z MQTT brokera odpojiť pomocou tlačidla v ľavom hornom rohu. Ten ho zároveň vráti späť na domovskú obrazovku.

Schematická vizualizácia skladového systému bola vytvorená pomocou mriežky (*grid*), ktorá obsahuje 54 položiek. Položky sú vlastne voľne dostupné UI komponenty <*ion-chip*> obsahujúce rôzne možnosti štýlovania už vopred pripravené. My sme ich nastavili tak, aby sa dostupné miesta v sklade vyznačili zelenou farbou a obsadené miesta šedou farbou. Taktiež pre obsadené miesta bolo nastavené, aby boli deaktivované, teda aby sa na ne nedalo kliknúť.

IMPLEMENTÁCIA KOMUNIKÁCIE V MOBILNEJ APLIKÁCII

Na komunikáciu medzi mobilnou aplikáciou a Node-REDom, ako už bolo spomenuté, využívame protokol MQTT. Aby sme mohli nastaviť MQTT klienta v mobilnej aplikácii, bolo potrebné do Ionic projektu nainštalovať knižnicu *ngx-mqtt*. Výhodou tejto knižnice je, že ponúka metódy pre posielanie, prijímanie, napojenie a odpojenie sa z MQTT brokera. Metóda pre prijímanie dát z MQTT brokera má ako návratovú hodnotu Observable (pozorovateľ) typu *IMqttMessage*. Toto v princípe znamená: pozoruj správu z MQTT brokera a získaj hodnotu len vtedy, keď si odber predplatíš (vykonáš subscribe). Toto v opisovanom prípade zabezpečuje knižnica *RxJS*. Výhodou tohto prístupu k dátam je, že ich môžeme predtým, ako vykonáme zápis hodnôt do nejakej premennej, upraviť podľa toho, aký výstup chceme mať.

Pre lepšiu prehľadnosť, bola v rámci projektu vytvorená verejná trieda *MqttClientService*, ktorá obsahuje všetky metódy potrebné pre MQTT klienta. *Service* je špeciálny typ triedy, ktorý sa v Angulari bežne využíva na poskytovanie funkcionalít a dát v rámci aplikácie.

Najprv bol vytvorený súkromný objekt *connection*, ktorý obsahuje všetky nastavenia potrebné pre vytvorenie MQTT klienta (programový modul č. 8).

Na napojenie a odpojenie sa z MQTT brokera máme dostupné verejné metódy *connect*() a *disconnect*() (programový modul č. 9). Metóda *connect*() nám vracia hodnotu True alebo False v závislosti od toho, či sa nám podarilo napojiť sa na MQTT broker.

Metódu *connect*() voláme v metóde *establishConnection*(), ktorú máme v komponente *HomePage* (Domovská obrazovka). Vtedy, keď operátor stlačí tlačidlo Connect, zavolá sa metóda *establishConnection*(), ktorá následne zavolá metódu *connect*(). V prípade, že sa podarilo napojiť na MQTT broker, operátor bude presmerovaný na obrazovku Warehouse Control. V opačnom prípade sa zobrazí chybová hláška.

Ďalšou verejnou metódou je *observeMultiple*() (programový modul č. 10), ktorej vstupným argumentom je pole *topics*. Toto pole vlastne obsahuje dané témy (topics),

Programový modul č. 8 Objekt connection [76]

```
private connection = {
    hostname: 'broker.hivemq.com',
    port: 8000,
    path: '/mqtt',
    clean: true,
    connectTimeout: 4000,
    reconnectPeriod: 4000,
    username: 'meno',
    password: 'heslo',
  };
```

Programový modul č. 9 Metóda *connect*() a *disconnect*() [76]

```
connect(): boolean {
    try {
      this.mqttService?.connect(this.connection);
    } catch (error) {
      console.log('mqtt.connect error', error);
      return false;
    }
    return true;
  }

  disconnect(): void {
    this.mqttService.disconnect();
  }
```

ktoré chceme pozorovať z MQTT brokera HiveMQ. Akonáhle sa na danú tému zapíše nejaká hodnota v MQTT brokeri, vykonáme odber (subscribe) a získame hodnotu. Návratovou hodnotou tejto metódy je *Observable* typu pole *IMqttMessage*.

Programový modul č. 10 Metóda *observeMultiple*() [76]

```
observeMultiple(topics: string[]): Observable<IMqttMessage[]> {
    const observables = topics.map((topic) => this.mqttService.observe(topic));
    return combineLatest(observables);
  }
```

Metóda *observeMultiple*() je volaná v metóde *doSubscribe*() z komponentu *WarehouseControlComponent* (obrazovka Warehouse Control). Metóda *doSubscribe*() sa zavolá, akonáhle sa načíta obrazovka Warehouse Control, nakoľko hneď chceme získať údaje zo skladového systému. Vstupom do metódy *observeMultiple*() je pole tém (topics), čo v našom prípade sú témy *'runningMode/#'* a *'productsInStack/#'*. Akonáhle vykonáme odber (subcribe) na tieto témy, tak získame hodnoty o tom, či proces beží a aký je stav v sklade.

Nakoniec tu máme verejnú metódu *publishMessage*() (programový modul č. 11), ktorej vstupným argumentom je objekt *publishPayload* s nasledovnými členmi: *topic* — pod akou témou (topic) chceme hodnotu poslať na MQTT broker, *payload* — užitočná hodnota (obsah premennej) pre danú tému a *qos* na nastavenie kvality služby.

Programový modul č. 11 Metóda *publishMessage*() [76]

```
publishMessage(publishPayload: {topic: string; payload: string; qos: number;}): void {
    const { topic, qos, payload } = publishPayload;
    console.log(publishPayload);
    this.mqttService.unsafePublish(topic, payload, { qos } as IPublishOptions);
  }
```

Táto metóda je volaná v metódach *startProcess*(), *stopProcess*() a *sendValue*(), ktoré máme v komponente *WarehouseControlComponent* (obrazovka Warehouse Control).

Akonáhle operátor zatlačí tlačidlo Start, volá sa metóda *startProcess*(), ktorá následne zavolá metódu *publishMessage*(). Ako parameter do metódy *publishMessage*() posielame objekt *publishStartProcess* (programový modul č. 12).

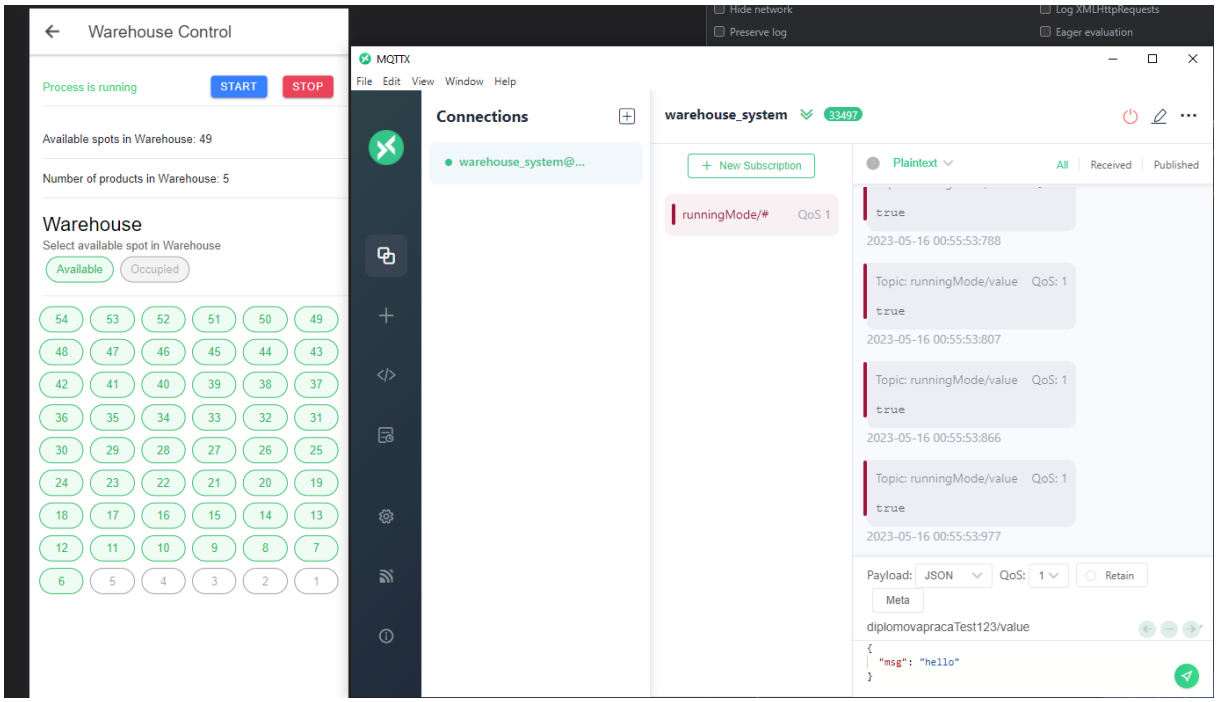
Rovnakým spôsobom pracujeme aj pre metódy *stopProcess*() a *sendValue*(), pričom posielame rôzne parametre pre príslušné témy (topics) v MQTT brokeri.

Na uistenie, že sa hodnoty z MQTT klienta z mobilnej aplikácie poslali na MQTT broker, otestovali sme spojenie opäť pomocou softvéru MQTTX, kde máme rovnakého

Programový modul č. 12 Objekt *publishStartProcess* [76]

```
publishStartProcess = {
    topic: 'startProcess/value',
    qos: 2,
    payload: 'true',
  };
```

testovacieho MQTT klienta (obrázok č. 151), ktorý sa prihlásil na odber (subscribe) na tému *runningMode/value* pod daným menom a heslom. Z obrázka č. 154 vidíme z mobilnej aplikácie (vľavo) informáciu, že proces beží a taktiež aj klient v MQTTX (vpravo), zobrazuje hodnotu z témy *runningMode/value* ako TRUE.

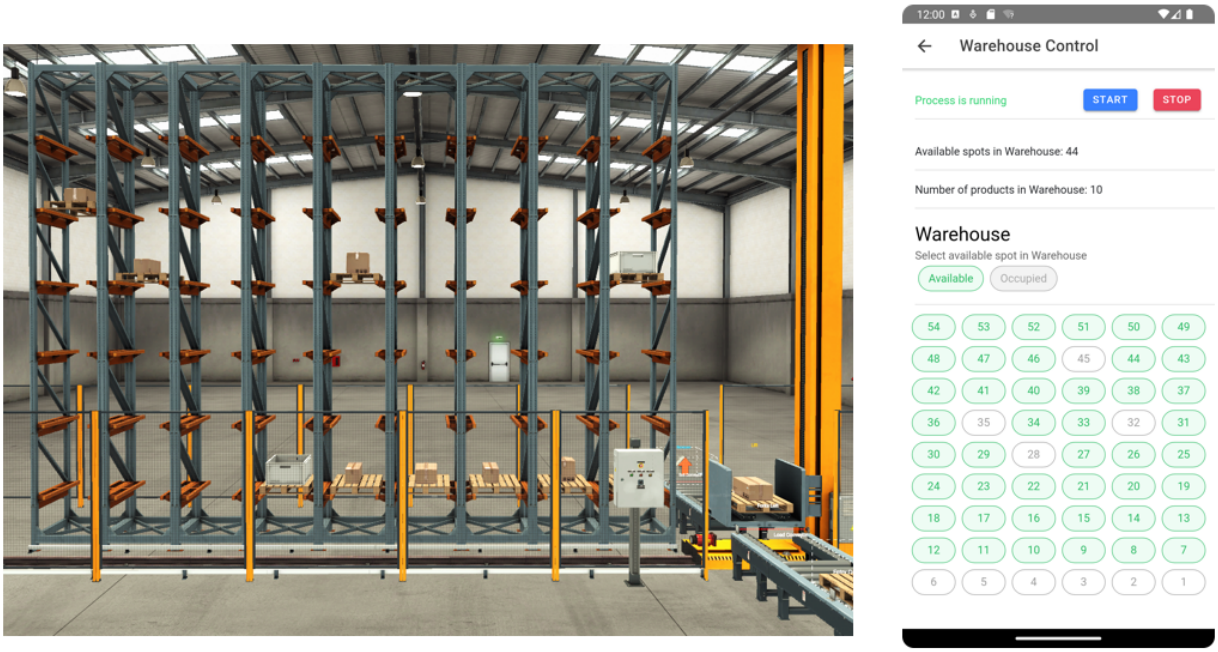


Obrázok č. 154: Testovanie komunikácie MQTT klient — MQTT broker [76]

4.4.8 Zhrnutie prípadovej štúdie

Akonáhle sa podarilo prepojiť komunikácie medzi jednotlivými technológiami, celý systém sme spustili, nechali bežať a sledovali výstupy. Na obrázku č. 155 vidíme konečný výstup prepojenia virtuálneho skladového systému vo Factory I/O a mobilnou aplikáciou realizovanou pomocou frameworku Ionic. Taktiež je vidno aj to, že oba aplikačné systémy navzájom medzi sebou komunikujú a vymieňajú aktuálne dáta.

Nakoľko bolo žiadané, aby konečný výsledok bol dostupný aj pre iné platformy mobilných zariadení, tak vďaka Ionicu a Capacitoru celý projekt je možné konfigurovať tak, aby sme mohli vytvoriť mobilné aplikácie pre platformy Android a iOS. Táto časť je re-



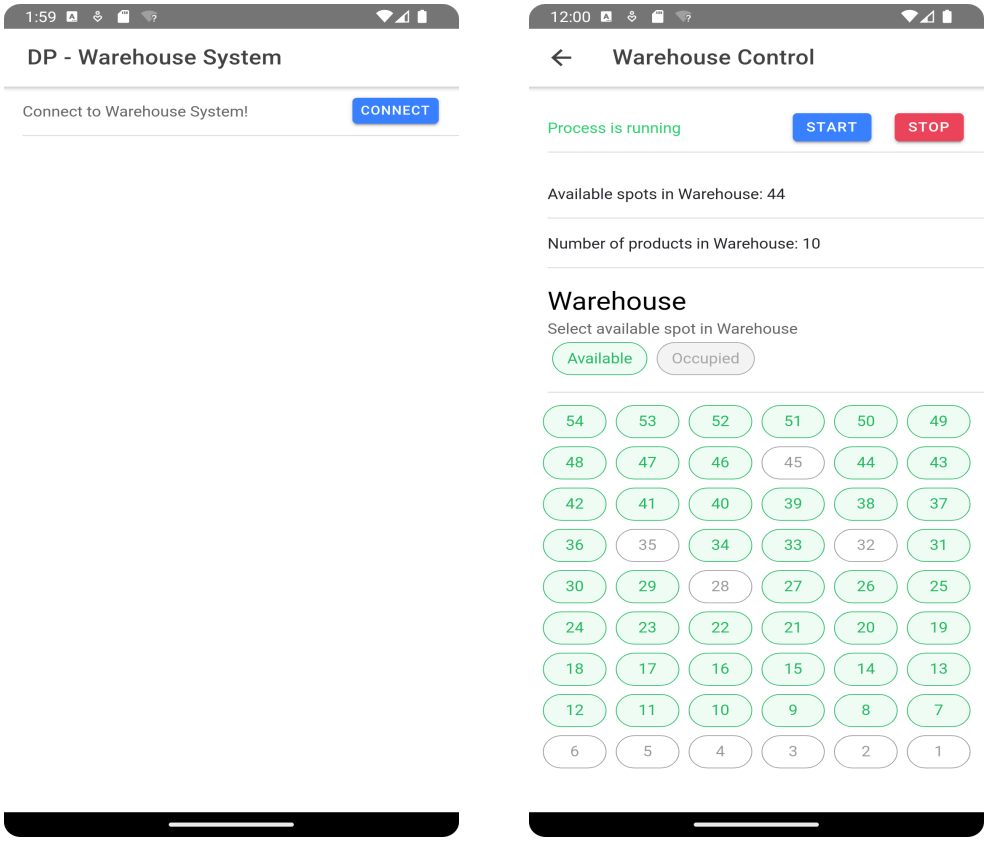
Obrázok č. 155: Prepojenie skladového systému s mobilnou aplikáciou [76]

latívne jednoduchá, nakoľko stačí iba spustiť nasledovné príkazy:

- *ionic capacitor copy android* — príkaz ktorý vytvorí verziu projektu pre platformu Android;
- *ionic capacitor copy ios* — príkaz ktorý vytvorí verziu projektu pre platformu iOS.

Projekt pre platformu Android sme spustili v Android Studiu, ktorý nám vygeneroval súbor *apk*. Funkcionalitu mobilnej aplikácie sme najprv otestovali na Android emulátore v Android Studiu (obrázok 156). Následne sme si aplikáciu nainštalovali aj na fyzické Android zariadenie a otestovali jej funkcionalitu.

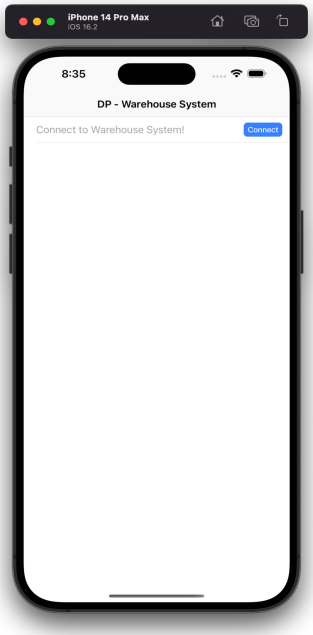
Projekt pre platformu iOS sme spustili v Xcode, ktorý vygeneroval súbor typu *ipa*. Funkcionalita pre iOS verziu bola najprv otestovaná na iOS emulátore v Xcode a potom aj na fyzickom iOS zariadení (obrázok č. 157).



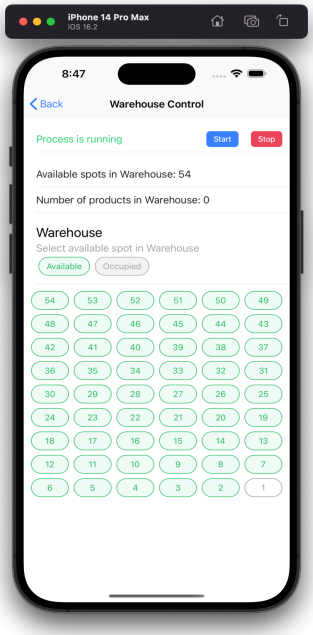
(a) Domovská obrazovka

(b) WarehouseControl obrazovka

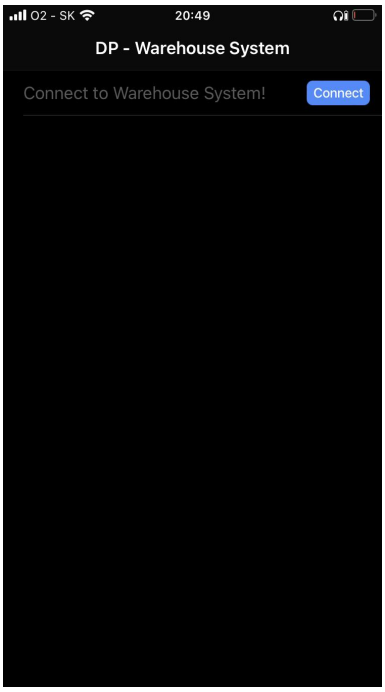
Obrázok č. 156: Obrazovky v mobilnej aplikácie v Android emulátore [76]



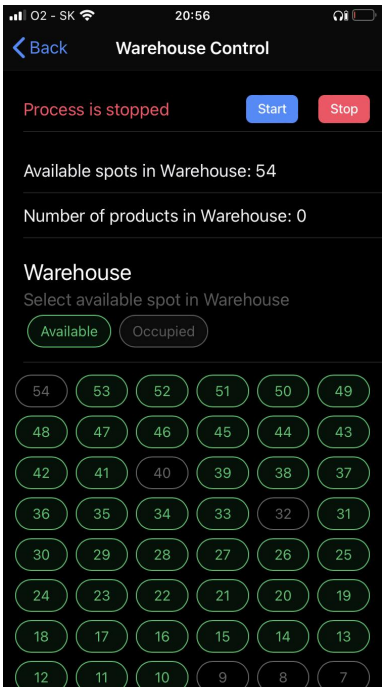
(a) emulátor



(b) emulátor



(c) fyzické zariadenie



(d) fyzické zariadenie

Obrázok č. 157: Obrazovky na iOS zariadení [76]

Predstavené riešenie je ďalej možné rozvíjať a vylepšovať. Napríklad, v skladovom systéme by mohla byť pridaná funkcionalita pre triedenie výrobkov, ktoré sa budú ukladať do skladu. V rámci tohto by sa mohla pridať funkcionalita v mobilnej aplikácii, aby si operátor mohol vybrať, či chce výrobky zoraďovať a zvoliť si možnosť, podľa akého

kľúča by ich triedil.

Taktiež sa v niektorých prípadoch v skladovom systéme môže stať, že výrobok, ktorý sa už v sklade nachádza, zo skladového miesta vypadne. Toto by sa dalo vylepšiť tým, že by sa pridal ďalší senzor alebo kamera, ktorá by snímala a detegovala spomínané situácie a oznamovala operátorovi v mobilnej aplikácii prípadnú poruchu.

Ďalším možným vylepšením je zakomponovanie databázového systému. Všetky informácie o skladovom systéme by sa ukladali do databázy, aby operátor mal všetky informácie k dispozícii kedykoľvek.

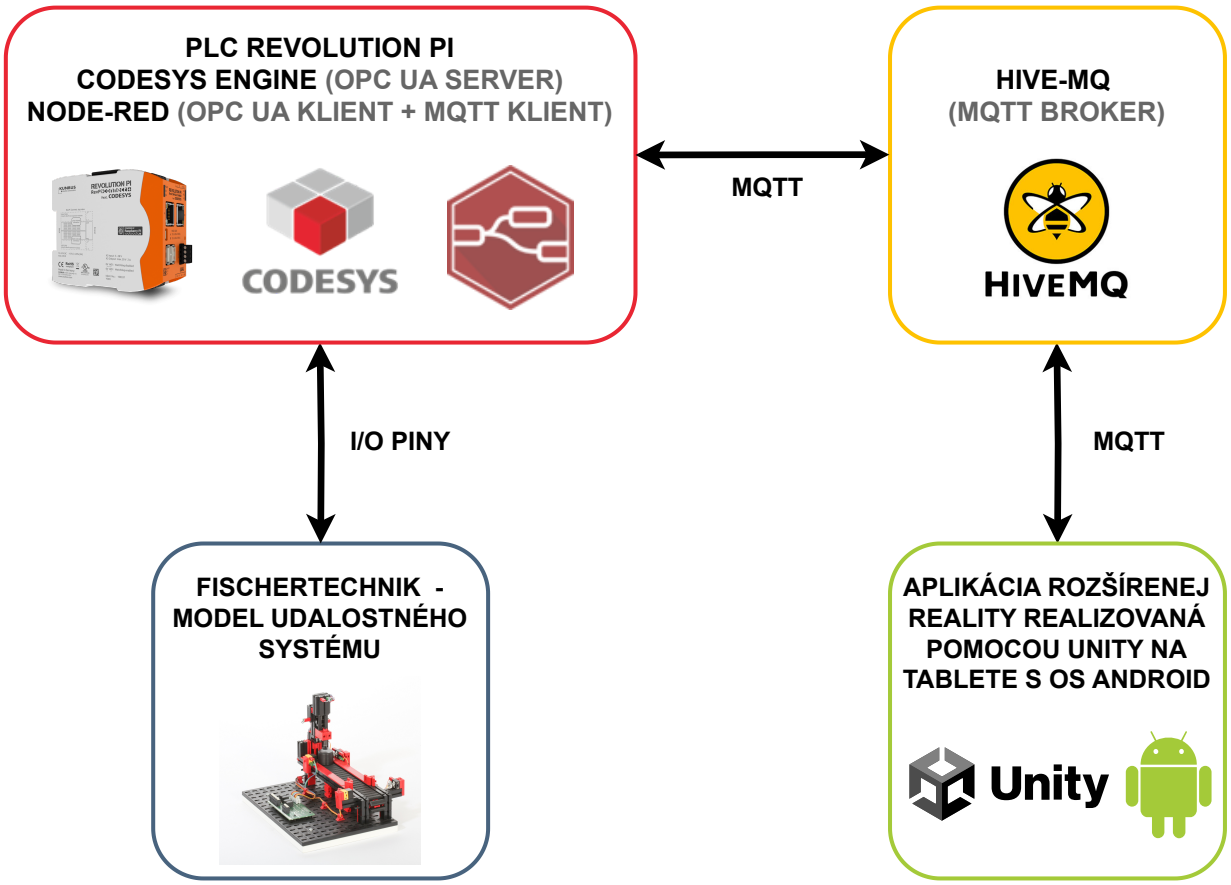
Videá k tejto edukačnej prípadovej štúdii je možné nájsť na adrese:

- `https://bit.ly/47S5L8Q`

Ďalšie informácie, návody a programové projekty je možné nájsť na e-learningovej webovej stránke `https://elearning.mechatronika.cool/?p=8350` .

4.5 Piata prípadová štúdia: Prepojenie PLC s aplikáciou rozšírenej reality

Táto prípadová štúdia sa zaoberá prepojením open-source PLC série Revolution Pi s aplikáciou rozšírenej reality bežiacej na tablete s operačným systémom Android. Takáto prípadová štúdia, ktorá prezentuje dynamicky sa rozvíjajúci segment počítačom generovanej reality, môže byť inšpiráciou pre moderné riešenia v priemysle. Implementácia rozšírenej alebo zmiešanej reality v digitálnych továrňach môže priniesť efektívnejšie monitorovanie a riadenie výroby, údržbu alebo moderné používateľské rozhrania zamerané na väčší komfort človeka, čo je jeden z pilierov konceptu Industry 5.0. Tento koncept bude v krátkosti predstavený v poslednej kapitole tejto učebnice.



Obrázok č. 158: Architektúra systému

Architektúra systému pre prepojenie PLC a rozšírenej reality sa skladá z niekoľkých komponentov, ktoré umožňujú prenos a spracovanie dát. Hlavným komponentom tohto systému je fyzický model dopravníkového pásu s dierovacím strojom od spoločnosti Fischertechnik, ktorý prijíma a posiela hodnoty z PLC cez vstupné a výstupné piny. Tieto hodnoty sú následne prenesené na OPC UA server (riešený cez Codesys runtime), kde sú uložené a ďalej spracované. Pre spracovanie týchto dát je využívaný middleware

Node-RED, ktorý slúži ako OPC UA klient a dokáže čítať a zapisovať údaje na OPC UA server. Node-RED dáta získané z OPC UA servera dokáže zasielať pomocou protokolu MQTT na broker HiveMQ. Pre demonštráciu možností využitia databázy je tiež jedna z premenných ukladaná do lokálnej databázy.

Android aplikácia (vyvinutá v 3D engine Unity) sa využíva pre zobrazenie dát v rozšírenej realite. Tieto dáta získava z MQTT brokera a vizualizuje ich užívateľovi. Takto je umožnené užívateľovi monitorovať stav modelu výrobného systému a tiež mať prehľad o aktuálnych hodnotách, ktoré sú v systéme sledované. Užívateľ má tiež možnosť zvoliť si manuálne ovládanie modelu výrobného systému. Hodnoty (zásahy) od užívateľa sa z Android aplikácie posielajú najprv na MQTT broker. Následne MQTT klient realizovaný v Node-RED hodnoty prijme a zašle ich na OPC UA server. PLC zachytí zmeny z OPC UA servera, hodnoty sa ďalej spracujú a PLC zapíše inštrukcie na model výrobného systému.

Celková architektúra systému (obrázok č. 158) pre daný prípadovú štúdiu je navrhnutá tak, aby umožnila rýchle a spoľahlivé prepojenie dát z PLC s aplikáciou rozšírenej reality. Táto prípadová štúdia je opísaná s využitím diplomovej práce [69].

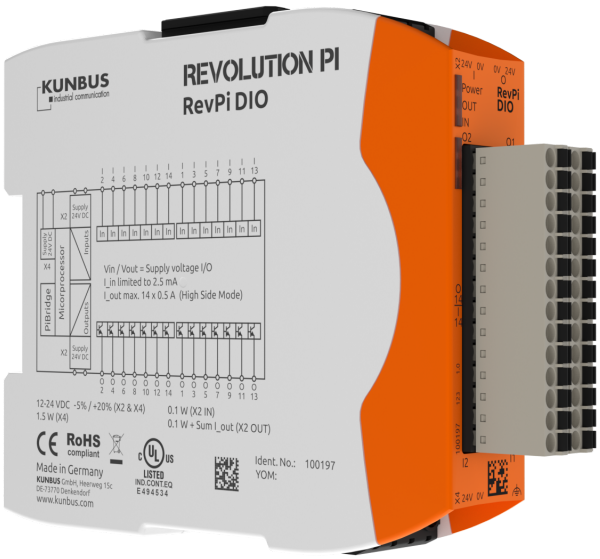
4.5.1 Špecifikácia a zostavenie laboratórneho modelu diskrétneho udalostného systému

Nakoľko sa na rozdiel od predošlých prípadových štúdií využíva fyzický laboratórny model diskrétneho udalostného systému, bolo nutné takýto model vybrať, resp. zostaviť. Po zostavení modelu ho bolo potrebné prepojiť s PLC, ktorý sa stal jeho riadiacim prvkom.

Medzi hardvérové komponenty patrí open-source PLC výpočtová jednotka **RevPi Connect + feat. CODESYS** (obrázok č. 159), digitálny input/output modul **RevPi DIO** (obrázok č. 160) a **napájací zdroj** na DIN lištu **MDR-60-24** (obrázok č. 161). Ako laboratórny model slúžila stavebnica **Fischertechnik** (obrázok č. 162) vo forme **dierovacieho (raziaceho) stroja a dopravníkového pásu**.

Je potrebné opísať, akým spôsobom sú jednotlivé komponenty systému zapojené. Vzhľadom na odporúčanie dokumentácie open-source PLC Revolution Pi bola riadiacia jednotka aj s ďalšími modulmi umiestnená na DIN lištu. Najprv je potrebné na ňu umiestniť napájací zdroj, následne RevPI DIO modul a potom výpočtovú jednotku RevPi Connect+, ktorá je prepojená tzv. PiBridge s DIO modulom. Fyzický model udalostného systému Fischertechnik obsahuje 2 DC motory, 2 fototranzistory, 2 tlačidlové spínače a 2 LED svetelné bariéry. Pre prepojenie PLC s hardvérovými súčasťami modelu obsahuje aj dosku plošných spojov (obrázok č. 163) s relé na prepólovanie motora, multipinový konnektor a ďalšie súčasti [24].

Obrázok č. 159: RevPi Connect+ feat. CODESYS [69]



Obrázok č. 160: RevPi DIO [69]

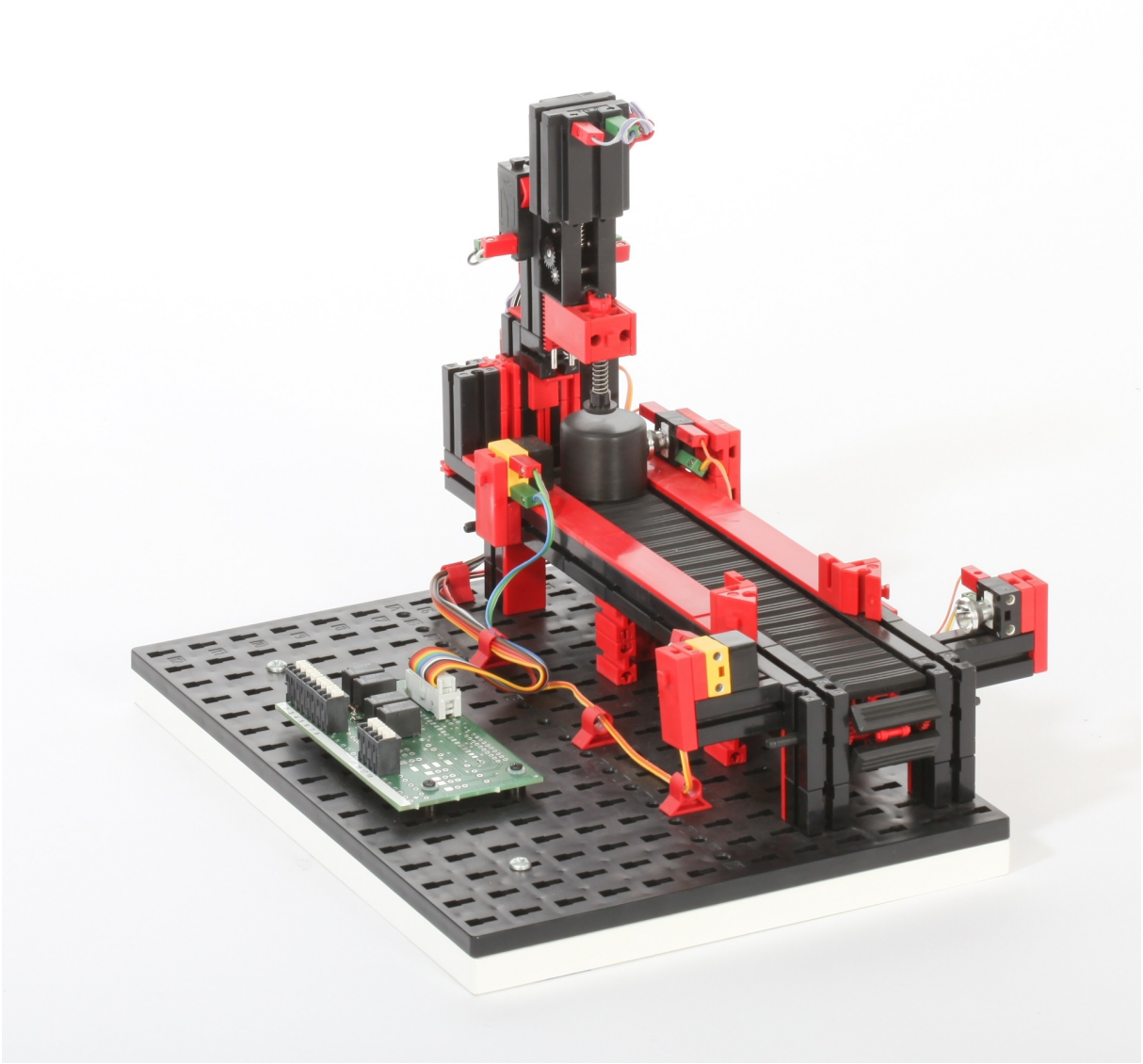
Obrázok č. 161: Napájací zdroj MDR-60-24 na DIN lištu [69]

Na obrázku č. 164 je možné vidieť vidieť finálne zapojenie modelu laboratórneho systému. Na overenie funkčnosti zapojenia bolo potrebné používať operačný systém Raspberry Pi OS upravený pre túto PLC jednotku. Do PLC je možné zapojiť monitor cez micro-HDMI port a taktiež klávesnicu a myš cez USB porty pre ovládanie. Operačný systém si vyžiada zadanie prihlasovacích údajov, ktoré sú dodané zo strany výrobcu.

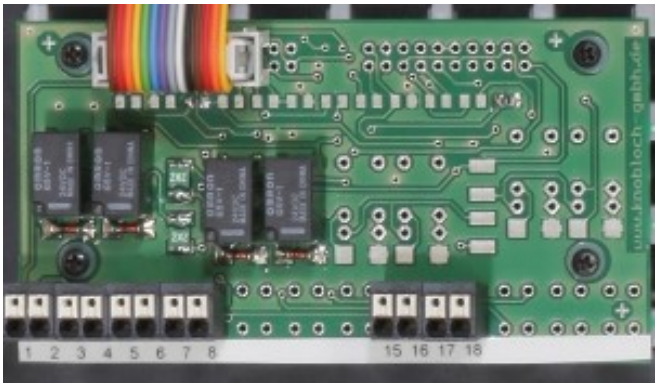
Na obrázku č. 165 je možné vidieť operačný systém Raspberry Pi OS s otvoreným terminálom a oknom s webovým prehliadačom. Vhodným spôsobom na rýchle otestovanie správnosti zapojenia je pomocou terminálu a príkazu *piTest*. Príkaz *piTest* s argumentom *-d* zobrazí zoznam použitých zariadení (modulov) pripojených k hlavnej výpočtovej jednotke PLC. Tento príkaz môže byť užitočný aj vtedy, ak zariadenie ešte nebolo nakonfigurované v programe PiCtory a chceme si iba overiť, že sú zariadenia správne zapojené a detegované. Konfiguráciu a jej význam v PiCtory opíšeme v nasledujúcej podkapitole. Ďalším potrebným krokom je pripojenie PLC k sieti. Samotný RevPi Connect+ má dva ethernetové porty na pripojenie k sieti pomocou RJ45 sieťového kábla. PLC bolo pripojené k rovnakému routeru ako PC, z ktorého sa bude nahrávať program do PLC. Následne bolo potrebné zistiť IP adresu PLC jednotky. Pomocou príkazu *ifconfig* v príkazovom riadku sme získali informácie o sieťových rozhraniach a ich IP adresách. Po zistení IP adresy sme sa mohli pripojiť k PLC pomocou SSH a pokračovať v ďalších úkonoch.

4.5.2 Komunikácia jednotlivých subsystémov a inštalácia

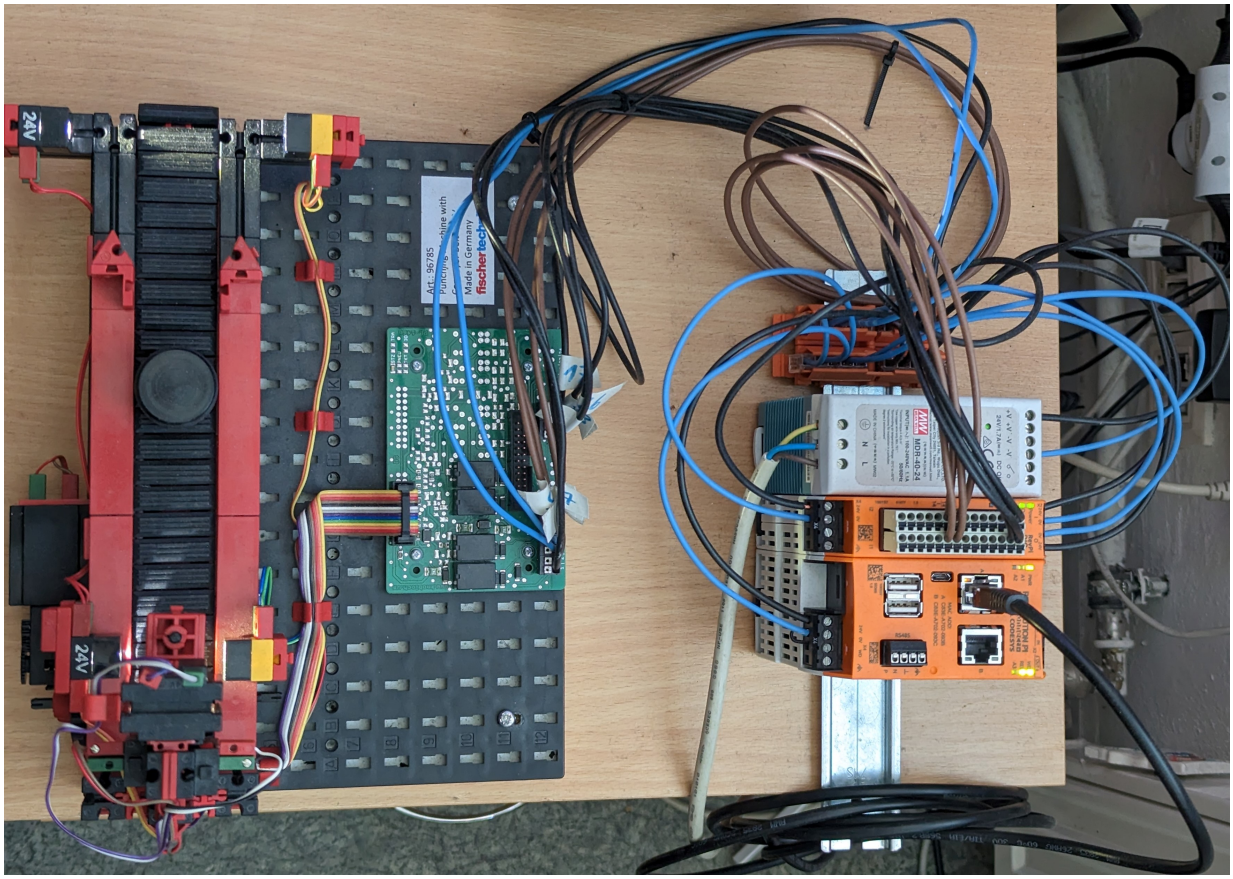
Po úspešnom zapojení a spustení PLC jednotky a jej pripojení na internet je možné pokračovať v konfigurácii. Aby sa bolo možné pripojiť k PLC, bolo potrebné zistiť jeho IP



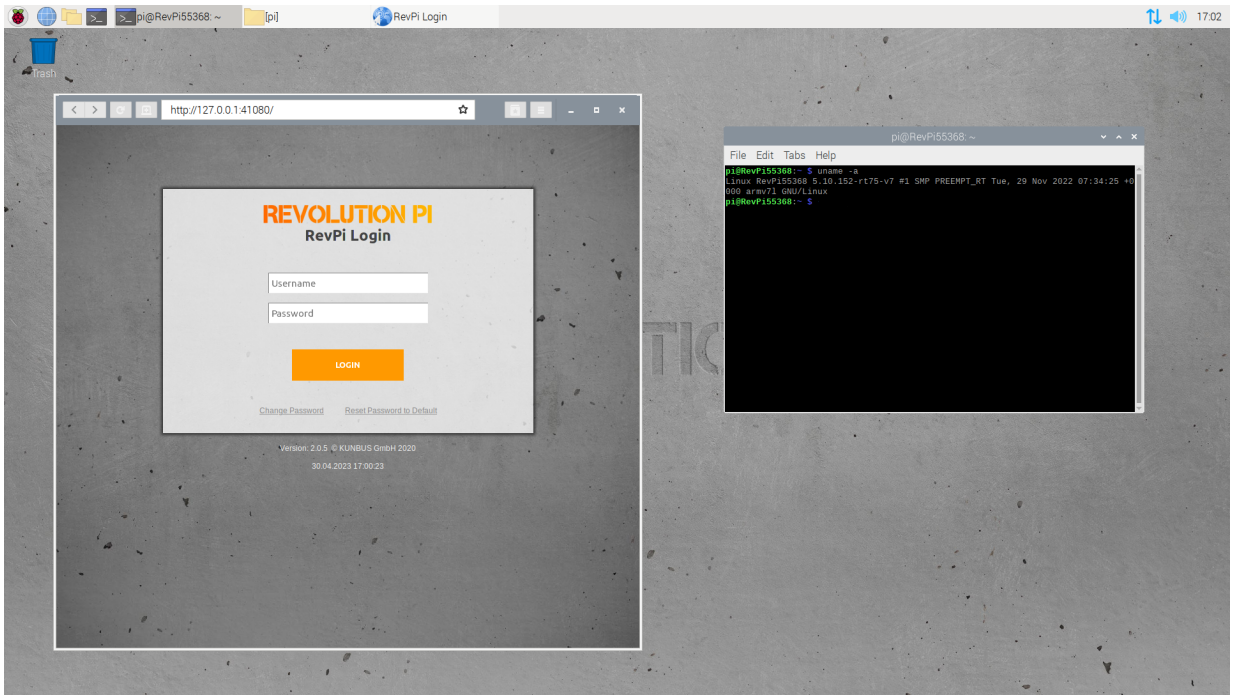
Obrázok č. 162: Fischertechnik dierovací stroj s dopravníkovým pásom [69]



Obrázok č. 163: Doska plošných spojov Fischertechnik [24]



Obrázok č. 164: Zapojenie laboratórneho systému [69]



Obrázok č. 165: Grafické rozhranie Raspberry Pi OS [69]

adresu, ktorú sme získali v predchádzajúcom kroku. Následne je možné využiť webovú aplikáciu **PiCtory**, ktorá umožňuje komunikáciu s PLC prostredníctvom webového prehliadača. Po zadaní IP adresy nášho PLC do internetového prehliadača sme v aplikácii PiCtory zostavili našu PLC zostavu aj vo virtuálnej podobe — teda výpočtovú jednotku aj s DIO modulom. Táto aplikácia umožnila okomentovať jednotlivé vstupné/výstupné piny, ktoré sme v našej prípadovej štúdii využívali. Rozhranie aplikácie PiCtory je znázornené na obrázku č. 166.

Na obrázku č. 166 je jasne vidieť, že orientácia v aplikácii PiCtory je intuitívna. Stačilo nájsť typy zariadení a modulov, ktoré sa využívali, a umiestniť ich do príslušnej časti aplikácie. Dôležité bolo umiestniť ich v rovnakom poradí, ako sú zapojené na DIN lište. Keď boli zvolené konkrétne zariadenia, bolo možné zmeniť názvy a pridať komentáre k jednotlivým vstupným/výstupným pinom. Pripravenú konfiguráciu si je potrebné uložiť a tiež uložiť ako štartovaciu konfiguráciu v časti *File*, aby táto vytvorená konfigurácia bola načítaná po každom reštarte systému.

V rámci opisovanej prípadovej štúdie boli využívané celkovo 4 vstupné a 4 výstupné piny. Výstupné piny boli označené `O_1` až `O_4` a slúžili na ovládanie motorov dopravníka a dierovacieho stroja. Vďaka nim bolo možné pohybovať pásovým dopravníkom dopredu a dozadu a posúvať dierovací stroj hore a dole. Vstupné piny boli označené číslami `I_11` až `I_14` a slúžili na čítanie stavu dvoch fotosenzorov umiestnených na krajoch dopravníka a tiež dvoch tlačidiel na dierovacom stroji. V tabuľke č. 6 je zhrnuté, ktoré vstupné a výstupné piny boli použité na ovládanie a čítanie konkrétnych častí fyzického modelu udalostného systému.

Tabuľka č. 6: Použité vstupné a výstupné piny

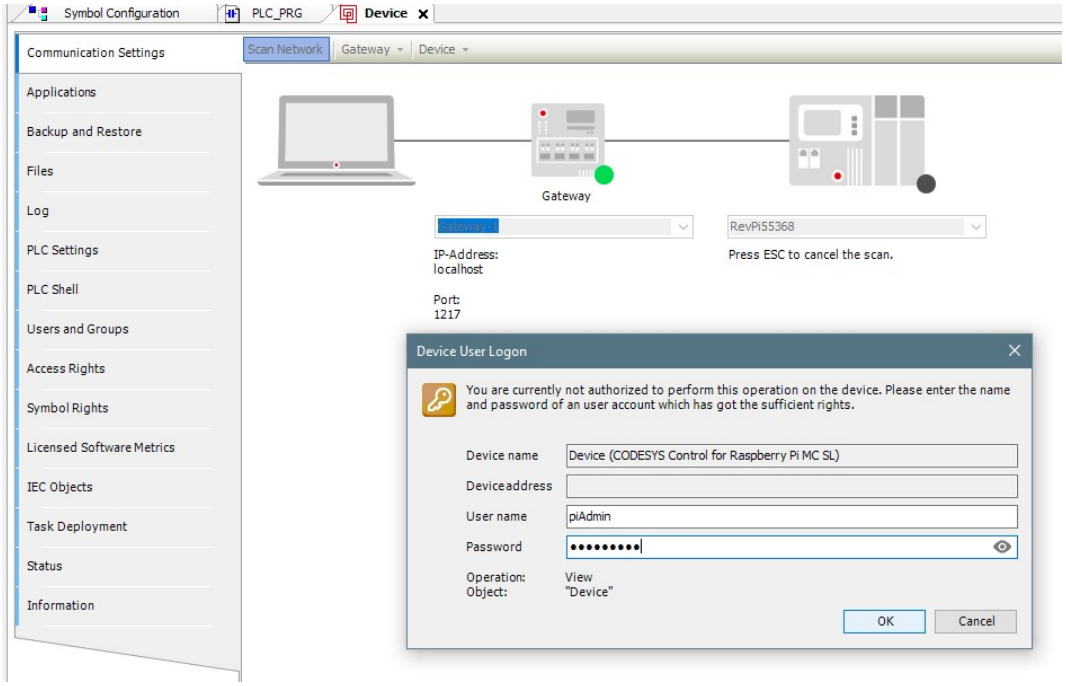
Číslo pinu	Využitie
I_11	Tlačidlo dolné na dierovacom stroji
I_12	Tlačidlo horné na dierovacom stroji
I_13	Fotosenzor na konci pásu
I_14	Fotosenzor pod dierovacím strojom
O_1	Dopravník — smer pohybu vľavo
O_2	Dierovací stroj — smer pohybu nahor
O_3	Dopravník — smer pohybu doprava
O_4	Dierovací stroj — smer pohybu nadol

4.5.3 Riadenie diskrétneho udalostného systému

PLC systém bol programovaný pomocou sady softvérových nástrojov Codesys, ktorá je kompatibilná s Revolution Pi. Konkrétne bola využitá verzia Codesys V3.5 SP18 Patch

Obrázok č. 166: Rozhranie PiCtory [69]

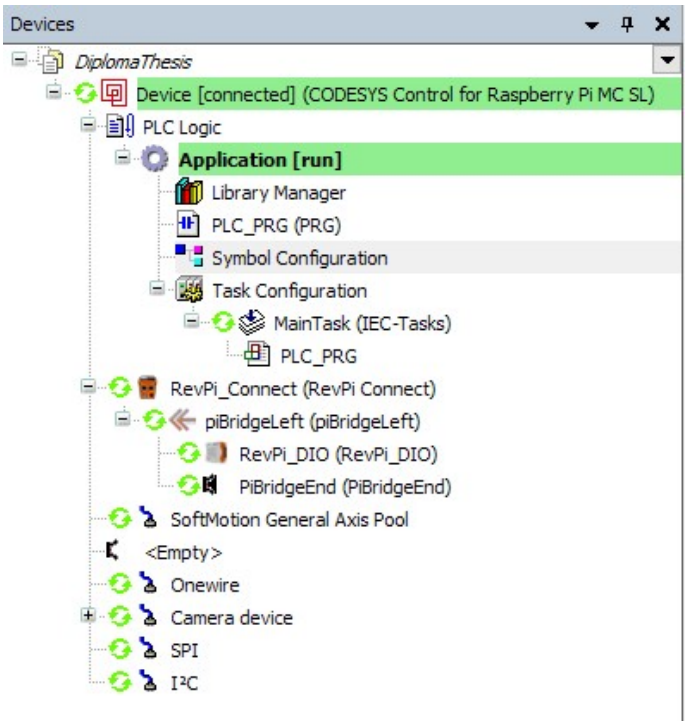
3, ktorá bola nainštalovaná na samostatnom počítači s operačným systémom Windows. Pre programovanie systému sme museli pripojiť náš PLC systém v Codesyse, ako je to možné vidieť na obrázku č. 167.



Obrázok č. 167: CODESYS pripojenie na PLC [69]

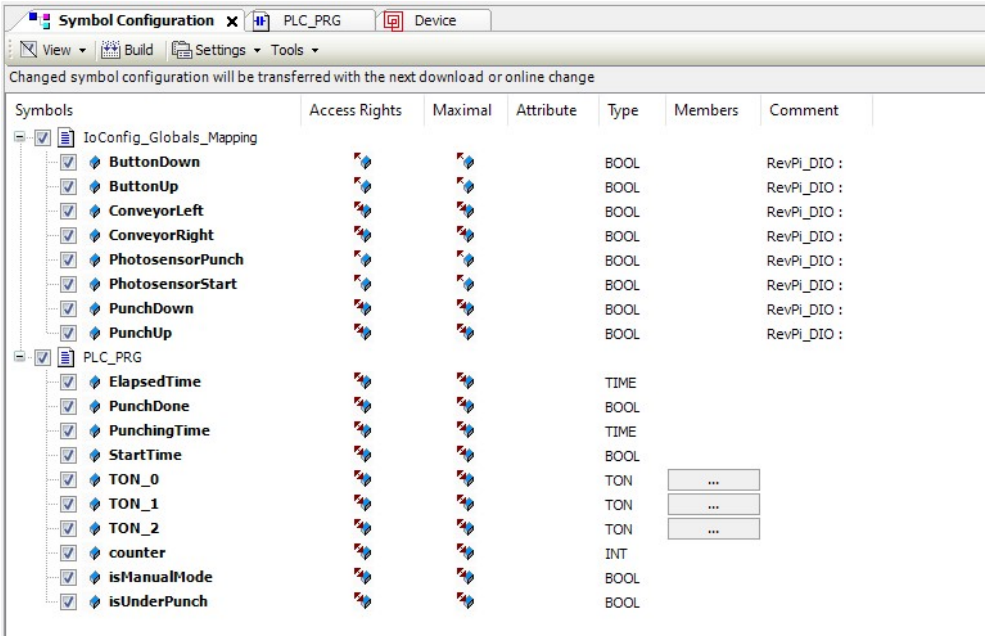
Po pripojení PLC k počítaču bolo potrebné v Codesyse zaradiť našu výpočtovú jednotku RevPi Connect+ a RevPi DIO modul do vytvoreného projektu. Tento krok sme realizovali s využitím vytvorenia konfigurácie zariadení v Codesyse. Konkrétne bolo potrebné vybrať pravým tlačidlom myši položku *Device* a následne vybrať *Add Device...* a zvoliť *RevPi Connect* z ponuky dostupných zariadení. Pod toto zariadenie sme ďalej pridali *piBridgeLeft* a modul *RevPi DIO*. Následne boli vyplnené potrebné informácie, ako napríklad IP adresa PLC, a konfiguráciu bolo ešte potrebné uložiť. Na obrázku č. 168 je možné vidieť, ako sa realizované pripojenie zobrazilo v strome zariadení v Codesyse. Tento krok umožnil komunikovať s PLC a nahrávať doňho programy, ktoré sme vytvorili v Codesyse.

Pri programovaní riadiaceho systému bolo potrebné určiť, aké funkcie bude vykonávať náš systém. S týmto súvisí funkčná špecifikácia vytvoreného programu v Codesyse. Vytvorili sme dve kategórie premenných — **globálne** a **lokálne**. Premenné pre čítanie hodnôt senzorov `I_11` až `I_14` a premenné pre ovládanie motorov laboratórneho systému `O_1` až `_O4` boli určené ako globálne premenné. Tieto sme evidovali priamo pri vstupoch a výstupoch DIO modulu. Možno ich vidieť v hornej časti obrázka č. 169, od *ButtonDown* až po *PunchUp*. Na druhej strane premenné ako počítadlá, časovače alebo pomocné premenné boli určené ako lokálne premenné priamo v PLC programe. Možno ich



Obrázok č. 168: CODESYS štruktúra projektu a zoznam zariadení [69]

vidieť v dolnej časti obrázku č. 169, od *ElapsedTime* až po *isUnderPunch*. V programovom module č. 13 je zobrazené definovanie lokálnych premenných.



Obrázok č. 169: CODESYS premenné [69]

Po úspešnom určení a deklarovaní všetkých potrebných premenných bol zostavený krátky kód, ktorý má na starosti riadenie fyzického modelu laboratórneho systému. Pre programovanie sme sa rozhodli použiť rebríkový diagram, ktorý je štandardne využí-

Programový modul č. 13 Lokálne premenné v Codesyse [69]

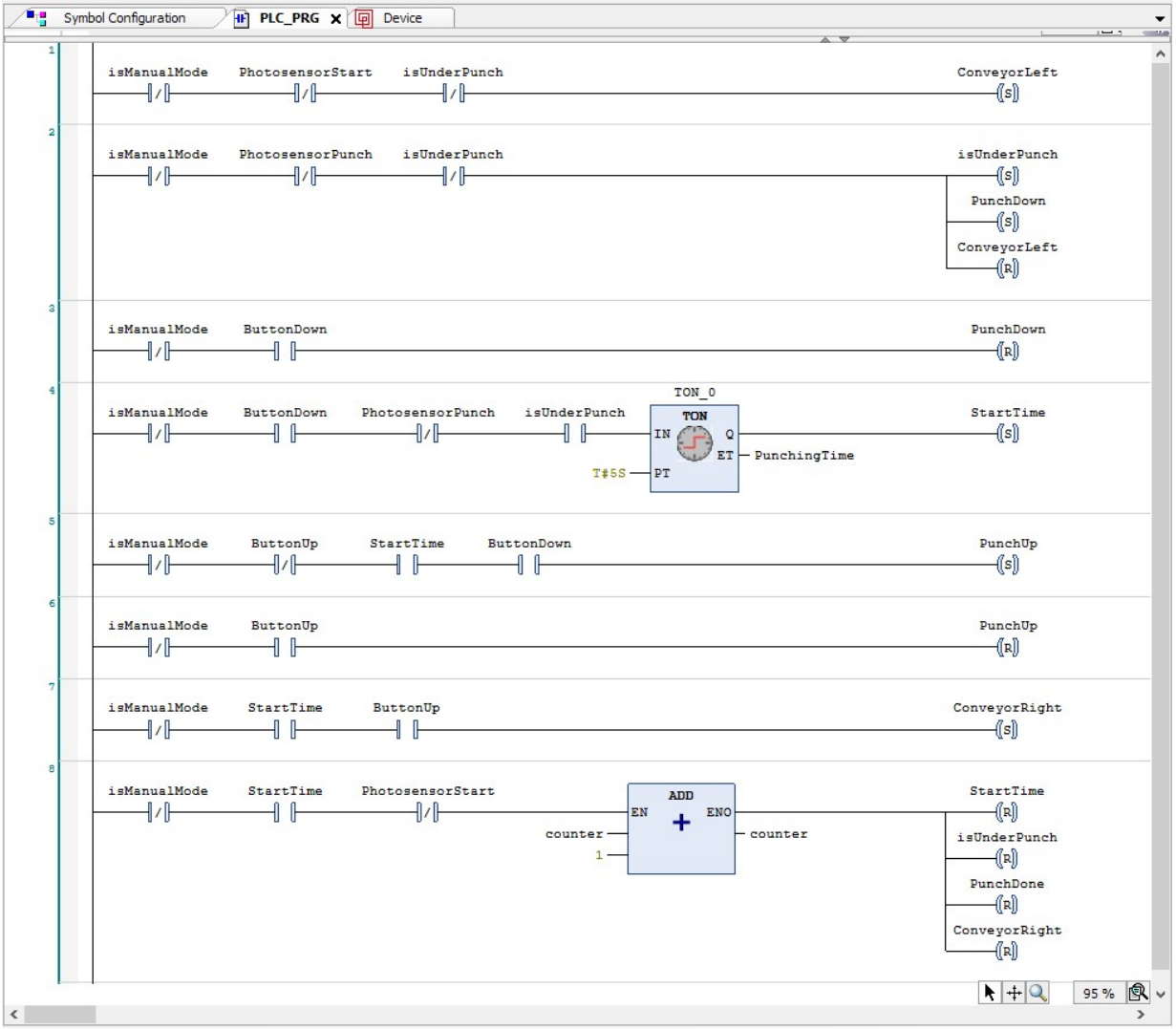
```
PROGRAM PLC_PRG
VAR
        TON_1: TON;
        StartTime: BOOL := 0;
        ElapsedTime: TIME;
        isUnderPunch: BOOL := 0;
        isManualMode: BOOL := 0;
        TON_0: TON;
        PunchingTime: TIME;
        TON_2: TON;
        PunchDone: BOOL;
        counter: INT;
END_VAR
```

vaný v PLC systémoch a umožňuje pomerne jednoduchú a prehľadnú vizualizáciu programu pomocou logických relácií.

V opisovanom programe bola implementovaná funkcia čakania na umiestnenie produktu na fotosenzor na začiatku dopravníkového pásu (*PhotosensorStart*). Po vložení výrobku na dohľad tohto fotosenzora sa spúšťa cyklus, v ktorom prejde produkt cez dierovací systém, pričom sa aktivuje fotosenzor pod dierovacím strojom (*Photosensor-Punch*). Následne sa dopravníkový pás zastaví a dierovací stroj sa posunie dole, kým sa neaktivuje senzor (*ButtonDown*), ktorý je v podobe tlačidla umiestneného na spodnej časti koľajníc dierovacieho stroja. Po tomto posune sa dierovací systém zastaví na dobu piatich sekúnd a následne sa vráti na svoju pôvodnú pozíciu na hornej časti koľajníc. Senzor (*ButtonUp*) slúži na detekciu horného limitu posunu dierovacieho systému. Po ukončení dierovania sa dopravník posúva v opačnom smere a produkt ide na začiatok dopravníka, kde sa cyklus ukončí a začína sa ďalší. Na obrázku č. 170 je zobrazený celý program, ktorý bol využitý na riadenie modelu laboratórneho systému, teda dopravníkového pásu s dierovacím strojom.

Na začiatku každého riadka v programe je umiestnená pomocná lokálna premenná s názvom *isManualMode*, ktorá určuje, či je laboratórny systém ovládaný manuálne alebo automaticky. Jej predvolenou hodnotou je FALSE, a teda program začína s automatickým riadením. Zmeniť hodnotu tejto premennej bude možné iba z mobilnej aplikácie pomocou tlačidla realizovaného v prostredí rozšírenej reality.

V prvom riadku programu je implementovaný kód, ktorý čaká na vloženie produktu na fotosenzor na začiatku pásu. Keď je produkt detegovaný, spustí sa motor dopravníko-



Obrázok č. 170: CODESYS program ovládajúci laboratórny systém [69]

vého pásu smerom vľavo. Premenná *isUnderPunch* sa používa ako pomocná premenná, ktorá zisťuje, či sa produkt nachádza pod dierovacím systémom. Jej predvolená hodnota je FALSE a k jej zmene príde len vtedy, keď produkt príde pod dierovací systém.

Druhý riadok programu zastavuje motor dopravníka (*ConveyorLeft*) a spúšťa dierovací stroj smerom nadol vtedy, keď fotosenzor pod dierovacím systémom deteguje produkt. Tiež sa tu nastavuje pomocná premenná *isUnderPunch* na hodnotu *true*, čo znamená, že produkt sa nachádza pod dierovacím systémom.

Tretí riadok programu zastavuje dierovací systém, ak je tlačidlový senzor na spodnej časti koľajníc aktivovaný a jeho hodnota sa zmení na hodnotu TRUE.

Vo štvrtom riadku programu sa overuje, či je zatlačené dolné tlačidlo, teda sa zisťuje, či je dierovací stroj dole, a taktiež musí byť aktivovaný fotosenzor pod dierovacím strojom a aktívna premenná *isUnderPunch*. V takom prípade sa spúšťa odpočet času, ktorý je nastavený na päť sekúnd. Táto premenná bola nazvaná *PunchingTime*.

Po uplynutí piatich sekúnd sa v ďalšom riadku programu spustí motor, ktorý posúva dierovací systém smerom hore. Tento riadok sa spustí len v tom prípade, keď pomocná premenná *StartTime* ukončila päť sekundový interval a tlačidlový senzor na hornej časti koľajníc nie je zatlačený, má hodnotu FALSE, zatiaľ čo opačný senzor (senzor v dolnej časti) je zatlačený a má hodnotu TRUE.

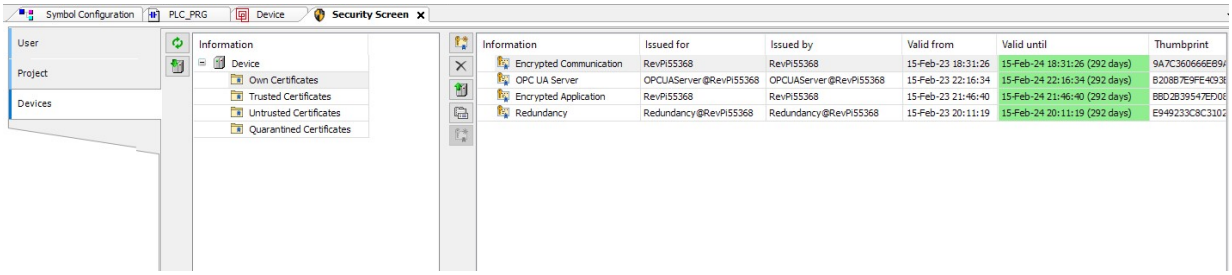
Keď sa dierovací systém pohybuje smerom hore, motor sa zastaví v tom momente, keď tlačidlový senzor na vrchnej časti koľajníc získa hodnotu TRUE a dierovanie sa ukončí. Tento proces je zobrazený na šiestom riadku.

Posledné dva riadky programu slúžia na posunutie výrobku dopravníkovým pásom smerom doprava a následne spustenie celého procesu odznova, keď sa hodnota fotosenzora na začiatku pásu zmení na FALSE. Hodnota pomocnej premennej *counter* sa v tomto momente zvýši o 1 a taktiež sa všetky ostatné pomocné premenné resetujú.

Pre vytvorenie PLC programu v Codesys boli dodržiavané konvencie pomenovania premenných. Tzv. *CamelCase* sme použili na pomenovanie premenných a funkčných blokov. Prvé písmeno prvého slova je v tomto prípade malé a prvé písmeno každého ďalšieho slova je veľké (napr. "myVariable"). *PascalCase* sme využili na pomenovanie programu a globálnych premenných. Prvé písmeno každého slova sa v tomto prípade píše veľkým písmenom (napr. "MyProgram"). *ALL_CAPS* sme použili na pomenovanie konštánt. Všetky písmená sa v tomto prípade píšu veľkými písmenami a slová sa oddeľujú podčiarkovník (napr. "MAXIMUM_VALUE").

4.5.4 Konfigurácia OPC UA, Node-RED, MQTT a MySQL

Pre správne fungovanie odosielania údajov na OPC UA server musí byť v Codesyse potvrdená možnosť s názvom *Support OPC UA Features* v časti *Symbol Configuration* v *Settings*. Taktiež je potrebné obnoviť licenciu pre OPC UA server, ktorá bude platná jeden rok od aktivácie. Aktivácia licencií sa vykonáva v okne *Security Screen*, čo je možné vidieť na obrázku č. 171 spolu so všetkými platnými licenciami. Po nahratí implementovaného programu do PLC sa od tejto chvíle odosielajú údaje na OPC UA server.



Obrázok č. 171: CODESYS OPC UA licencia [69]

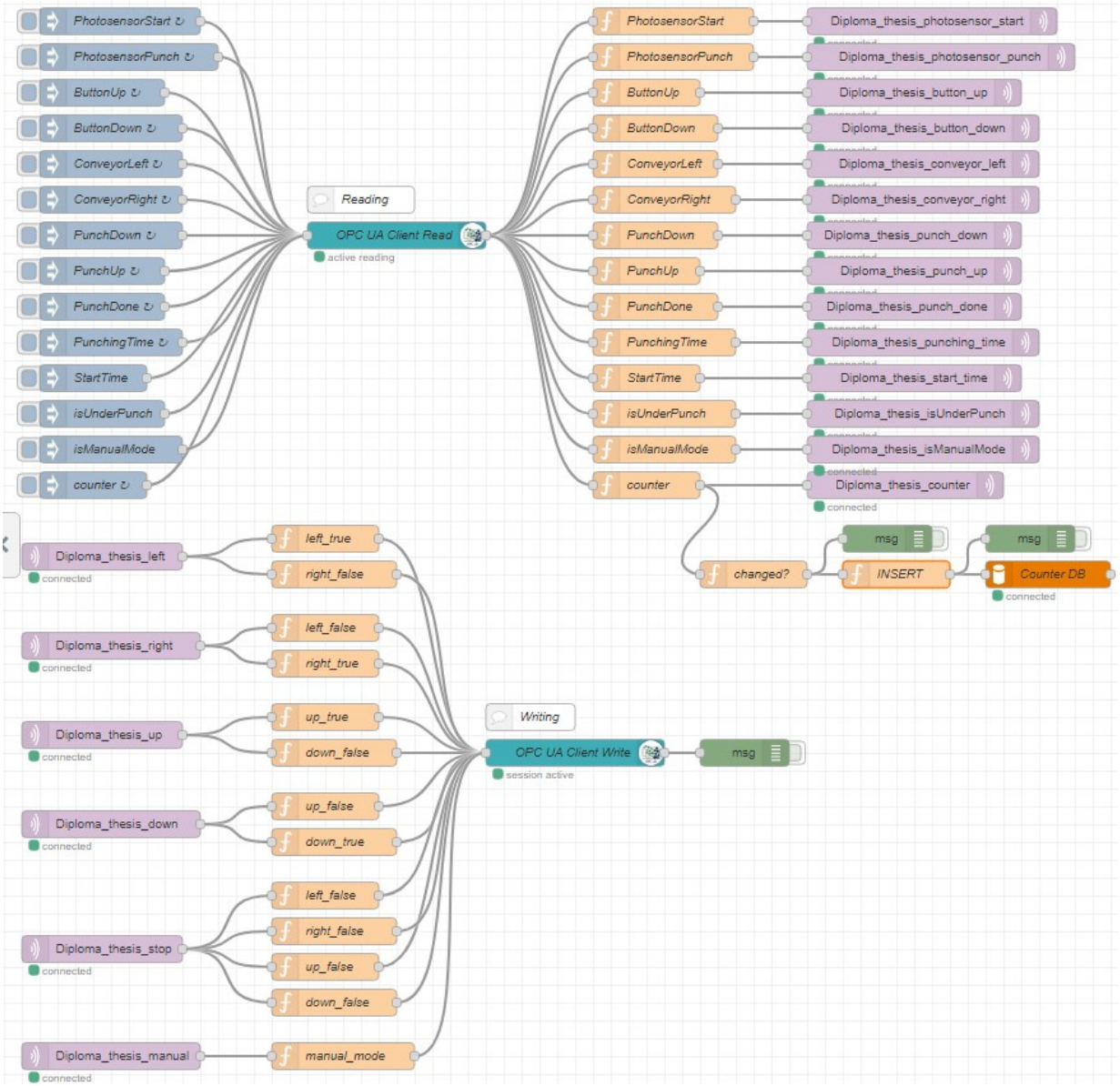
Pre overenie správneho fungovania OPC UA servera bol opäť využitý OPC UA klient s názvom UaExpert, odkiaľ sme sa pripojili na OPC UA server a mohli sme sledovať dáta produkované predmetným PLC. Potiahnutím zvolených premenných do okna *Data Access View* je možné monitorovať všetky premenné zadefinované v programe jednotlivých premenných, čo budeme potrebovať v nasledovnom kroku pri nastavovaní Node-RED toku.

Aby sme mohli získavať a pracovať s údajmi produkovanými opísaným fyzickým laboratórnym systémom, využili sme vývojový nástroj (middleware Node-RED), ktorý už bol nainštalovaný na opisovanom PLC vo výpočtovej jednotke. Na grafické prostredie v internetovom prehliadači Node-RED sa dostaneme pomocou IP adresy, na ktorej je pripojený náš systém a za koniec IP adresy je potrebné za dvojbodku pridať port 1880. V opisovanom prípade to bolo *http://192.168.0.100:1880/*. Tu nás uvíta uložený programový tok.

Na obrázku č. 172 je možné vidieť celý Node-RED program pre manipuláciu s dátami. V hornej časti sú uzly, ktoré čítajú premenné z OPC UA servera a tie sa ďalej triedia podľa *msg.topic* (čo je vlastne *Node Id* premennej spomínaný vyššie). Následne sa odosielajú na cieľovú destináciu, t. j. odosielajú sa na MQTT broker. Taktiež premenná *counter* sa ukladá do MySQL databázy. Triedenie sa vykonáva jednoduchým spôsobom pomocou Node-RED funkcií. V programovom module č. 14 je ukázané, ako sa tento krok realizuje pre premennú *PhotosensorStart*. Postup je podobný aj pre ostatné premenné. Po odchytení správnej premennej návratová hodnota putuje do uzla typu *mqtt out*. Tento uzol sa pripája na MQTT broker (server) *broker.hivemq.com:1883*.

Programový modul č. 14 Ukážka funkcií v Node-RED [69]

```
if (msg.topic=="ns=4;s=|var|CODESYS Control for Raspberry Pi MC
    SL.Application.IoConfig_Globals_Mapping.PhotosensorStart")
return msg;
```



Obrázok č. 172: Programový tok v Node-RED [69]

V grafickom prostredí Node-RED sme doinštalovali potrebné balíky *node-red-contrib-opcua* a taktiež *node-red-node-mysql*, ktoré umožňujú pracovať s uzlami pre OPC UA klienta a realizujú spojenie na MySQL databázu. Pomocou uzlov *OPC UA client* sme sa následne pripojili k OPC UA serveru.

Uzly s názvom *OPC UA Client Read* a *OPC UA Client Write* sú takmer rovnaké, jediný rozdiel majú v nastavení položky *Action*, pričom *OPC UA Client Read* má nastavené READ a *OPC UA Client Write* má nastavené WRITE pre správne čítanie (resp. zapisovanie) z/na OPC UA server. Ako *Endpoint* sme nastavili IP adresu a port, na ktorom sa nachádza opisovaný OPC UA server. V tomto prípade to bolo *opc.tcp://192.168.0.100:4840*. Položky *SecurityPolicy* a *SecurityMode* boli ponechané na *None* a *Anonymous*. Pri každej zmene uzla je dôležité nezabudnúť zatlačiť tlačidlo *Deploy*, aby sa zmeny uplatnili.

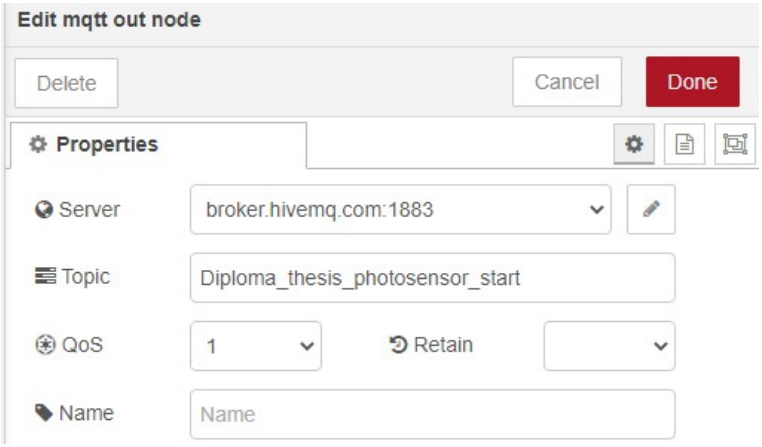
Na získanie dát z OPC UA servera do Node-REDu sme využili uzol *Inject*. Do tohto uzla sme dopísali *Node Id* do časti *msg.topic* a *Datatype* do časti *msg.datatype*. Názov jednotlivých inject uzlov sme pre prehľadnosť nastavili rovnaký, ako je aj názov premenných v Codesyse. Interval opakovania získania dát bol nastavený na 1 sekundu. Na obrázku č. 173 je možné vidieť konfiguráciu inject uzla pre premennú *Photosensor start*.

Cieľové úlohy prípadovej štúdie zahŕňali nielen získavanie dát, ale aj ich posielanie do Unity aplikácie pre rozšírenú realitu. Pre tento účel bol využitý MQTT broker HiveMQ, ktorý je voľne dostupný pre rôzne IoT zariadenia a umožňuje ich pripojenie. Aby bolo možné jednotlivé premenné odosielať na HiveMQ, vytvorené boli funkcie na ich selekciu a rozdelenie do rôznych MQTT tém (topicov). Všetky MQTT uzly s názvami tém môžeme vidieť fialovou farbou na obrázku č. 172. Konfigurácia uzla *MQTT out* pre odosielanie premennej s názvom *Photosensor start* je znázornená na obrázku č. 174.

Obrázok č. 173: Node-RED uzol typu inject [69]

4.5.5 Implementácia prepojenia PLC a rozšírenej reality

V tejto podkapitole bude opísané prepojenie medzi PLC a aplikáciou rozšírenej reality realizovanej pomocou 3D enginu Unity. Pre implementáciu bol využitý tablet s operačným systémom Android a podporou knižnice pre rozšírenú a zmiešanú realitu Android



Obrázok č. 174: Uzol MQTT out [69]

ARCore.

IMPLEMENTÁCIA KOMUNIKÁCIE S DATABÁZOU

V rámci prípadovej štúdie bolo rozhodnuté, že databáza bude realizovaná na externom serveri. Následne bolo potrebné vytvoriť tabuľku s vhodnými stĺpcami. Do databázy sa bude ukladať hodnota premennej *counter*, ktorá sa bude ukladať spolu s časovou pečiatkou zápisu (angl. timestamp). Táto hodnota udáva počet dierovaných položiek v celom simulovanom procese až po reštart systému, keď sa premenná resetuje na nulu.

Vytvorená bola tabuľka s názvom *counter*, ktorá obsahuje stĺpec *id* typu bigint označený ako primárny kľúč a ten sa zároveň automaticky zvyšuje. Ďalej je prítomný stĺpec *counter* typu int, a stĺpec *timestamp* typu timestamp.

Logika pre ukladanie do databázy sa realizuje v toku programu Node-RED. Najprv sa vo funkčnom uzle *counter* porovnáva hodnota globálnej premennej *prevNumber* ukladanej do *flow* objektu. Ak táto hodnota nebola nájdená, hodnota *prevNumber* sa nastaví na nulu, ako je to znázornené v programovom module č. 15. Hodnota ďalej putuje do nasledovného funkčného uzla s názvom *changed?*. V tomto uzle sa porovnáva, či sa hodnota zmenila, a teda či sa bude ukladať do databázy. Kód funkčného uzla je vidieť v programovom module č. 16. Ak sa hodnota zmenila, hodnota sa posiela do nasledujúceho funkčného uzla pomenovaného *INSERT*, kde sa upraví hodnota timestamp na vhodný formát pre ukladanie do MySQL databázy. Taktiež sa pripraví SQL dopyt na vloženie do databázy. Tento funkčný uzol je možné vidieť v programovom module č. 17. Po funkčnom uzle *INSERT* nasleduje už iba samotný uzol *mysql*, ktorý obsahuje konfiguračné údaje pre pripojenie na databázu. Prehľad zapísaných údajov v databáze je možné vidieť na obrázku č. 175.

Programový modul č. 15 Funkcia v Node-RED pre nastavenie globálnej premennej [69]

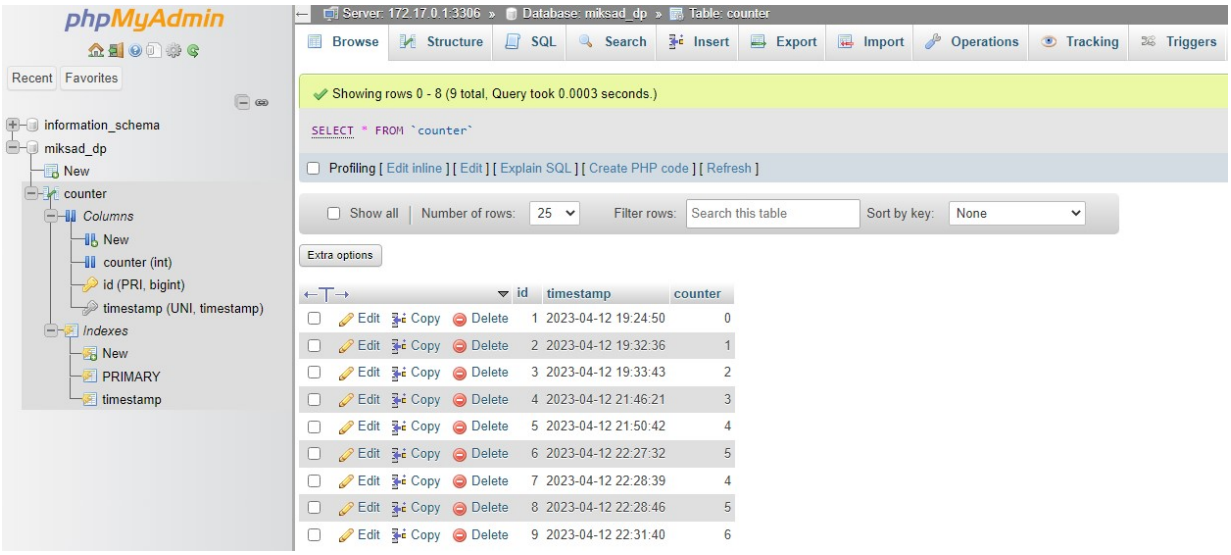
```
if (flow.get('prevNumber') === undefined)
    flow.set('prevNumber', 0);
if (msg.topic=="ns=4;s=|var|CODESYS Control for Raspberry Pi MC SL.Application.PLC_PRG.counter")
return msg;
```

Programový modul č. 16 Funkcia v Node-RED pre kontrolu zmeny globálnej premennej [69]

```
var prevNumber = flow.get('prevNumber');
// Check if the current value of the counter is different from the previous value
if (msg.payload !== prevNumber) {
    // If it is, create a new message object with the current value
    var newMsg = {
        counter: msg.payload,
        timestamp: msg.serverTimestamp
    };
    // Update the previous value stored in context and return new message object
    flow.set('prevNumber', msg.payload);
    return newMsg;
}
```

Programový modul č. 17 Funkcia v Node-RED pre nastavenie INSERT SQL dopytu [69]

```
var timestamp = msg.timestamp.toLocaleString
    ('en-US', { timeZone: 'Europe/Belgrade', hour12: false }).
    replace(/\//g, '-').replace(',', '').replace(' ', 'T');
var dateParts = timestamp.split(/[-T:]/);
var timestampString = dateParts[2] + '-' + dateParts[0] + '-' + dateParts[1] +
    ' ' + dateParts[3] + ':' + dateParts[4] + ':' + dateParts[5];
msg.topic = "INSERT INTO counter(counter, timestamp) VALUES (" +
    msg.counter + ", '" + timestampString + "');"
return msg;
```



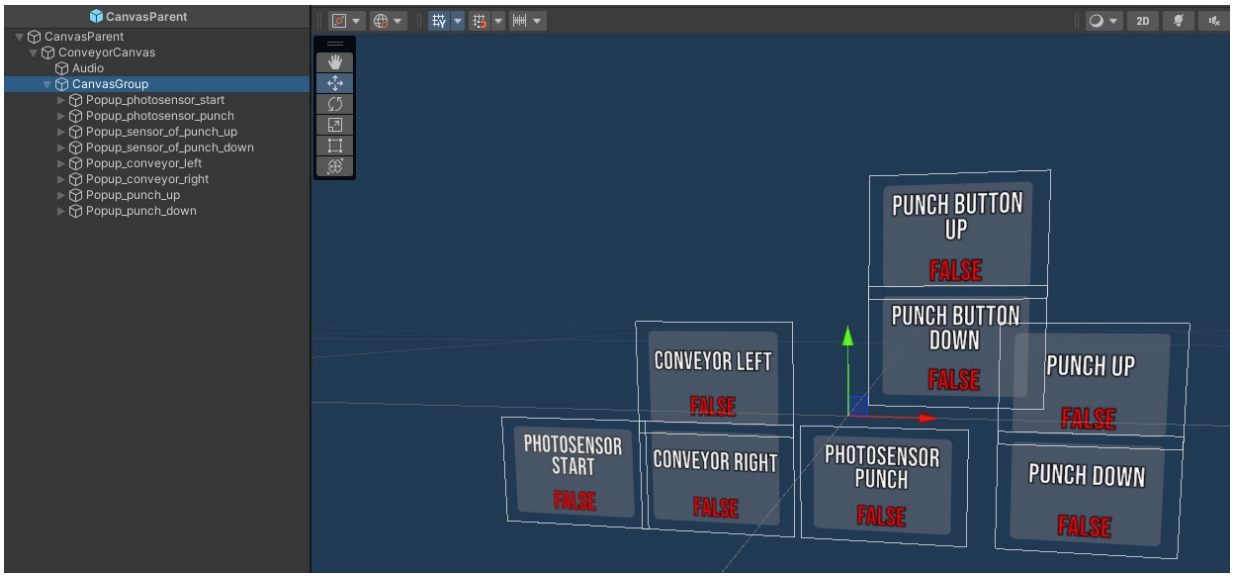
Obrázok č. 175: Hodnoty v databáze [69]

IMPLEMENTÁCIA APLIKÁCIE ROZŠÍRENEJ REALITY V 3D ENGINE UNITY

Ako posledný krok bolo potrebné preberať údaje z MQTT brokera a zobraziť ich v aplikácii rozšírenej reality vytvorenej v 3D engine Unity. Využitá bola verzia 2021.3.1f1 tohto enginu. Základ aplikácie bol vyvíjaný podľa oficiálneho návodu [89]. Na prácu s rozšírenou realitou využíva opisovaná aplikácia knižnicu/technológiu ARCore (SDK od spoločnosti Google), ktoré ponúka multiplatformové rozhrania API, a preto je to odporúčaný spôsob vytvárania aplikácií rozšírenej reality pomocou Unity.

V prostredí editora Unity sme vytvorili špeciálny asset *XR Reference Image Library* z knižnice ARCore, kam je možné pridávať ľubovoľný obrázok, ktorý bude slúžiť ako identifikátor pre vloženie objektov do scény. Pridaný bol do neho obrázok QR kódu, ktorý bude možné pomocou kamery mobilného zariadenia skenovať. Do uvedeného assetu možno uložiť ľubovoľný počet obrázkových značiek, ktoré chceme, aby bolo možné sledovať vo vytvorenej aplikácii v prostredí rozšírenej reality. Cieľom aplikácie bude naskenovať QR kód a po úspešnom naskenovaní zobraziť používateľské rozhranie v rozšírenej realite k modelu linky.

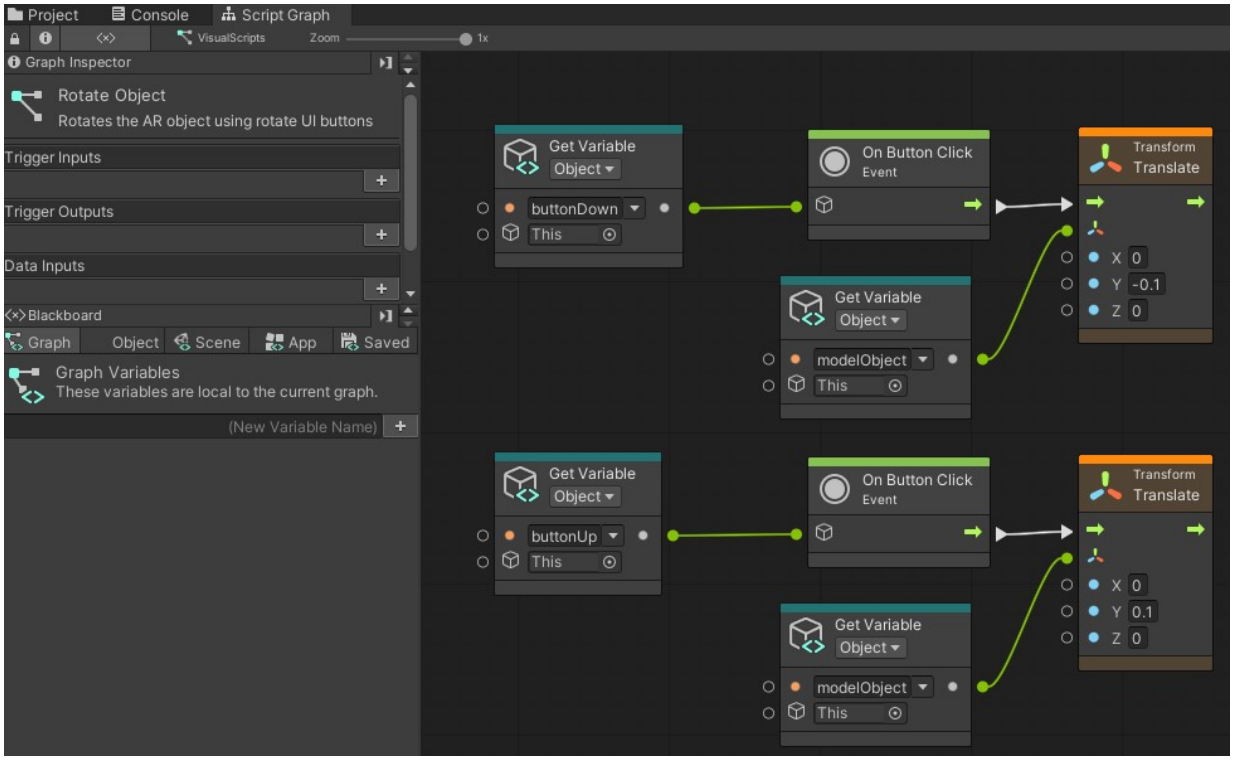
Následne bol v hierarchii Unity projektu vybraný herný objekt *AR Session Origin* a v časti Inspector bol pridaný komponent *AR Tracked Image Manager*. V tomto komponente je nutné do časti *Serialized Library* pridať vytvorený asset *XR Reference Image Library* a taktiež do časti *Tracked Image Prefab* je nutné vložiť vybraný prefabrikát. My sme si vytvorili prefabrikát s názvom *CanvasParent*, ktorý obsahuje popup okná pre každý údaj odosielaný z MQTT brokera. V tomto popup okne bude uvedený názov a hodnota prichádzajúcej premennej. Herný objekt *CanvasParent* je predvolene deaktivovaný a aktivuje sa až po naskenovaní vloženého QR kódu. Náhľad zobrazenia je možné vidieť na



Obrázok č. 176: Náhľad zobrazenia údajov v Unity [69]

obrázku č. 176.

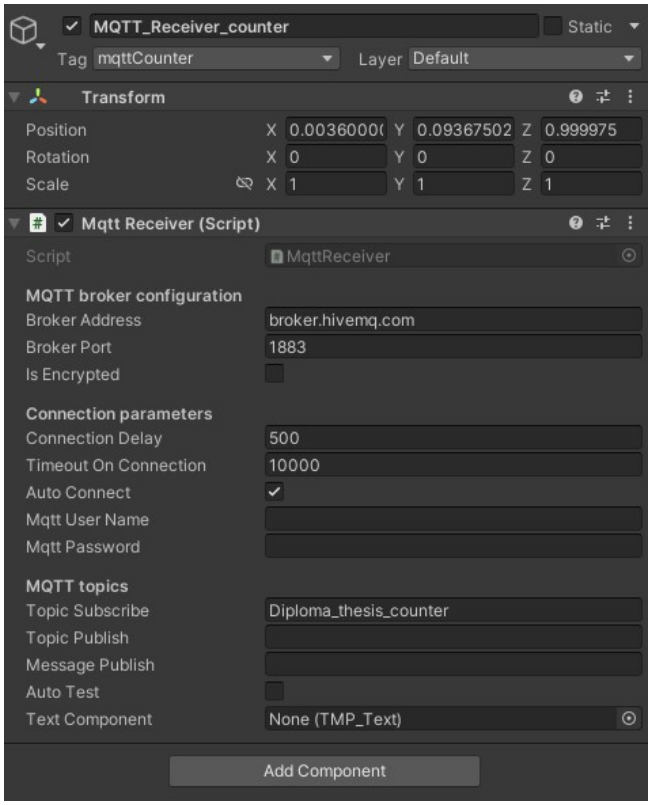
V tomto momente bolo možné urobiť build aplikácie a otestovať ho na Android zariadení — tablete. Po oskenovaní QR kódu sa v reálnom svete v prostredí rozšírenej reality pridal herný objekt, ale nebolo možné s ním vôbec interagovať a ani nerobil žiadnu funkciu.



Obrázok č. 177: Vizuálny skript na posúvanie objektu [69]

Do scény sme teda pridali tlačidlá na posúvanie herného objektu, aby sme zabezpečili možnosť mierne posúvať objekt pri situáciách, kedy sa objekt vytvorí na nesprávnom mieste, čiže buď vysoko, alebo príliš nízko. V tomto prípade by sa nedali prečítať hodnoty z popup okien. Na tento účel bolo vytvorené používateľské rozhranie nazývané prekrytie priestoru na obrazovke (angl. screen space overlay). Nazýva sa tak preto, lebo je vždy pred kamerou a viditeľný na obrazovke. Na naprogramovanie funkcií tlačidiel boli využité vizuálne skripty. V hierarchii projektu bol vytvorený nový prázdny herný objekt s názvom *VisualScripts*. V časti Inspector bol pridaný komponent *Script Machine* a následne sme vytvorili nový skriptovací graf. Skriptovací graf sme nazvali *Rotate Object* a v ňom bola následne vytvorená funkcionalita pre posúvanie objektu v scéne — obrázok č. 177.

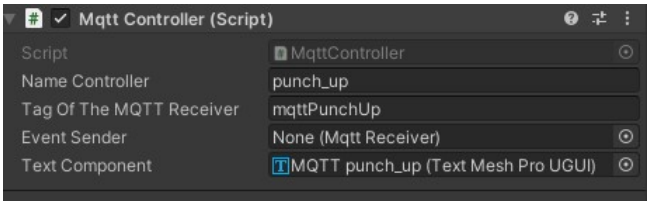
Podobne bol vytvorený skript pre obracanie objektu ku kamere zariadenia, ktorý bol nazvaný *DisplayPopup*. Tu bol využitý funkčný uzol *Look At*, do ktorého vstupujú uzly *On Update*, *Camera* a samotný herný objekt *canvasPopup*. Výstup tohto uzla sme pripojili na uzol *Rotate*. Týmto bolo zabezpečené, aby popup okno s monitorovacími údajmi vždy smerovalo na kameru zariadenia, a teda vždy bolo jasne čitateľné pre používateľa.



Obrázok č. 178: MQTT prijímanie správ v Unity [69]

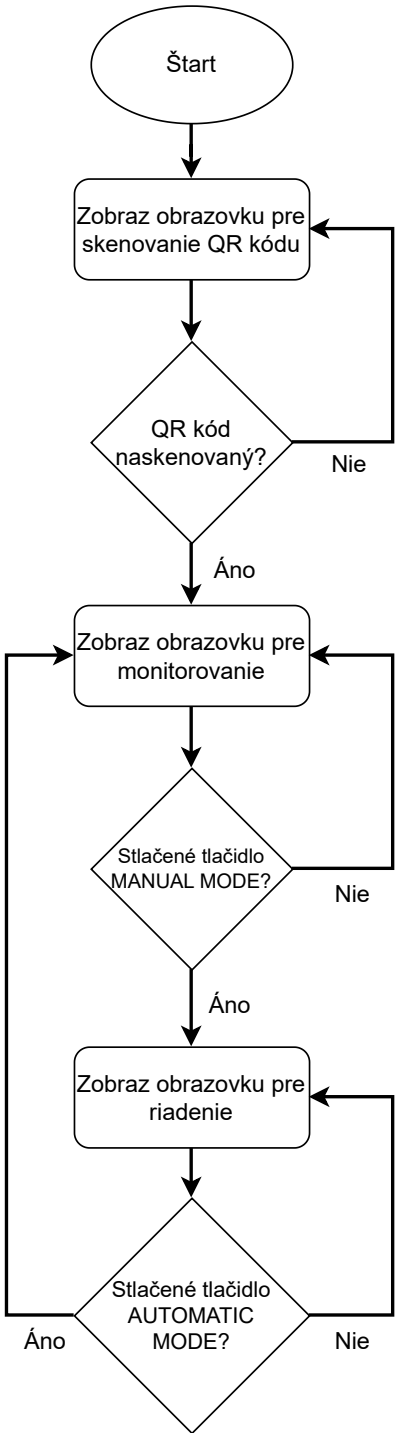
Na preberanie údajov z MQTT brokera sme využili knižnicu *M2MQTT for Unity* [90], ktorá obsahuje potrebné funkcie pre prácu s komunikačným protokolom MQTT

v 3D engine Unity. V hierarchii projektu bolo potrebné vytvoriť dva prázdne herné objekty *MQTT_Receivers* a *MQTT_Publishers*. Slúžili na prehľadnejšie ukladanie objektov pre odosielanie a prijímanie MQTT správ. Na obrázku č. 178 je možné vidieť jeden z mnohých objektov pre prijímanie dát z MQTT brokeru, kde vidíme adresu, port a tému, na ktorú sa pripájame pre prijímanie správ. Podobným spôsobom sme vytvorili aj objekty pre odosielanie správ na MQTT broker. Jediný rozdiel je v tom, že majú navyše vyplnené položky *Topic Publish* a *Message Publish*. Týmto herným objektom sme priradili značku (angl. *tag*) a názov tagu sme mohli použiť pre zobrazenie hodnoty v popup okne. Tieto súčasti sme pridali do skriptovacieho komponentu *MqttController*, ako je možné vidieť na obrázku č. 179.

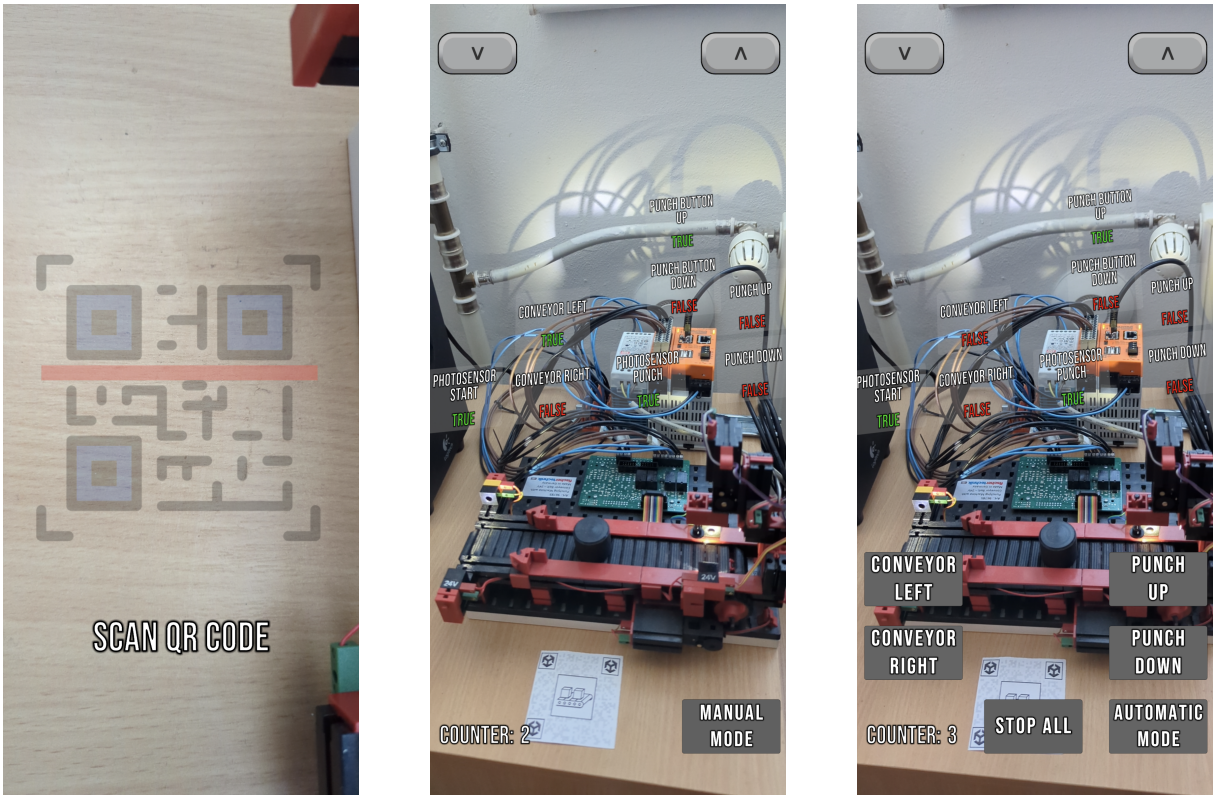


Obrázok č. 179: Vloženie dát z MQTT brokera do popup okna [69]

Vo finálnej aplikácii má používateľ k dispozícii tri obrazovky. Na začiatku sa zobrazí obrazovka s obrázkom QR kódu a výzva na skenovanie identifikátora. Po jeho oskenovaní sa zobrazí obrazovka pre monitorovanie modelu laboratórneho systému s tlačidlami pre mierne posúvanie popup okien a taktiež tlačidlo na manuálne ovládanie modelu laboratórneho systému. Po zatlačení tlačidla *Manual Mode* sa dopravníkový pás a aj dierovací stroj zastaví a na obrazovke sa zobrazia ďalšie tlačidlá na ovládanie laborórneho modelu. Na obrázku č. 180 je možné vidieť vývojový diagram aplikácie pre Android zariadenia. Na obrázku č. 181 je zase možné vidieť, ako môže vyzerať každá zo spomínaných obrazoviek.



Obrázok č. 180: Vývojový diagram aplikácie [69]



(a) Skenovanie QR kódu (b) Zobrazenie hodnôt (c) Manuálny režim riadenia

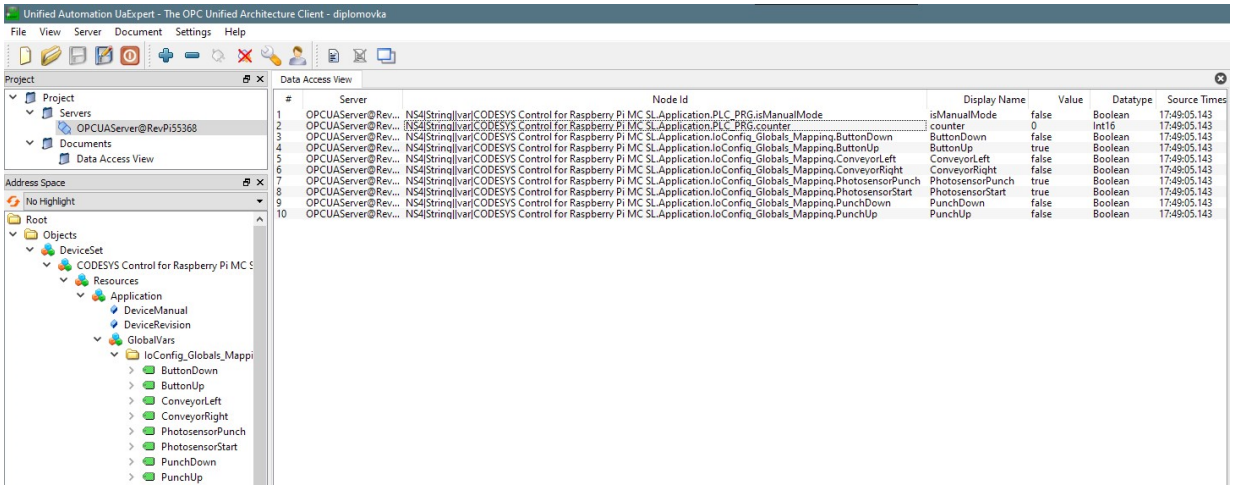
Obrázok č. 181: Obrazovky z Android aplikácie pre rozšírenú realitu [69]

4.5.6 Testovanie a validácia systému

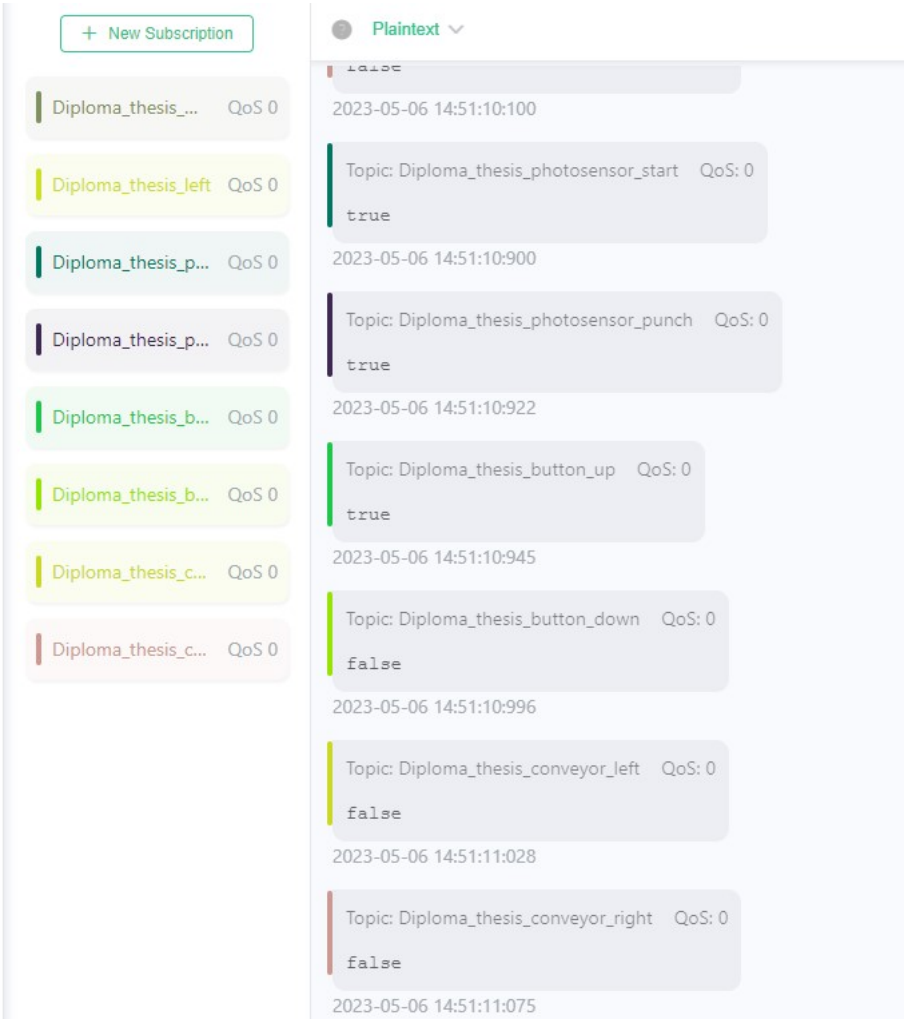
Po úspešnej implementácii softvérovej časti aplikácie v rozšírenej realite, ktorá prijíma údaje z MQTT brokeru, bolo potrebné prejsť k testovaniu a validácii celého systému. Vzhľadom na to, že sme pracovali s PLC systémom a OPC UA serverom, prvým krokom bolo otestovanie správneho fungovania týchto komponentov. Na tento účel bol použitý OPC UA klient — program UaExpert, ktorý umožňoval čítať a zapisovať hodnoty premenných priamo na server a sledovať ich stav. Testovanie v programe UaExpert je možné vidieť na obrázku č. 182.

Následne sme testovali spojenie medzi MQTT brokerom a aplikáciou v rozšírenej realite. K tomuto účelu bol využitý program MQTTX, ktorý umožňoval sledovať a analyzovať správy odosielané a prijímané z MQTT brokera, ako je možné vidieť na obrázku č. 183. Tento nástroj umožňoval zistiť, či aplikácia správne prijíma dáta zo servera. Prijímané dáta z MQTT brokera bolo možné otestovať aj priamo v Unity po spustení aplikácie, kde je možné v kóde vypisovať hodnoty do debug okna.

Výsledky testovania a validácie umožnili odhaliť a odstrániť chyby a nedostatky, ktoré sme identifikovali počas testovania, a zabezpečiť, že náš aplikačný systém bude schopný poskytnúť spoľahlivé a presné informácie používateľom.



Obrázok č. 182: OPC UA klient — UaExpert [69]



Obrázok č. 183: Hodnoty v programe MQTTX [69]

Možným vylepšením celého aplikačného systému by mohla byť detekcia modelu dopravníka inou metódou, ako je skenovanie QR kódu. Namiesto tejto metódy by mohlo byť využité rozpoznávanie obrazu, kde by sa priamo rozpoznával 3D tvar samotnej linky alebo jej časti.

Ďalším možným vylepšením by bol vývoj aplikácie rozšírenej reality na headset typu Microsoft HoloLens 2 alebo podobnej pokročilej alternatívy. Tento typ headsetu ponúka pokročilejšie možnosti pre rozšírenú a zmiešanú realitu, ako napríklad sledovanie očí a rúk v reálnom priestore pre ovládanie a zobrazovanie virtuálnych objektov. Týmto by sme vylepšili používateľskú skúsenosť a umožnili by sme používateľom pracovať s aplikáciou s väčšou efektivitou.

Video k tejto edukačnej prípadovej štúdii je možné nájsť na adrese:

- `https://youtu.be/c4AJ5AbAtro`

Ďalšie informácie, návody a programové projekty je možné nájsť na e-learningovej webovej stránke `https://elearning.mechatronika.cool/?p=8352` .

5 Vybrané aplikácie konceptu Industry 4.0

Modernými aplikáciami digitálnych a inteligentných technológií v rámci konceptu Industry 4.0 sa na Slovensku venuje viacero univerzitných pracovísk, či už vo forme výskumných projektov alebo v rámci modernizácie výučby smerom k tomuto trendu. Na Ústave automobilovej mechatronike Fakulty elektrotechniky a informatiky Slovenskej technickej univerzity v Bratislave (ÚAMT FEI STU) sú už približne od roku 2016 kontinuálne riešené grantové úlohy súvisiace najmä s oblasťou moderných metód automatického riadenia, počítačom generovanej reality, digitálnych dvojčiat, pokročilých komunikačných systémov, kontajnerizácie, cloud a edge computingu, strojového videnia, nových foriem HMI, metód modelovania, simulácie a riadenia udalostných a hybridných systémov s podporou Petriho sietí atď. Na tomto pracovisku sa výskumníci systematicky zaoberajú všetkými paradigmami Industry 4.0, teda interoperabilitou, virtualizáciou, decentralizáciou, real-time činnosťou, orientáciou na služby a modularitou. Takisto sa pracovisko venuje aj modernizácii výučby smerom k Industry 4.0 v rámci študijných programov vo všetkých troch stupňoch štúdia v študijnom odbore kybernetika. Pripravuje sa aj medziodborový študijný program v kombinácií odborov informatika a kybernetika, ktorý reaguje aj na trendy v oblasti konvergencie informačných a operačných technológií opísaných v tejto učebnici.

V nasledujúcej kapitole budú opísané vybrané aplikácie digitálnych a inteligentných technológií riešené na ÚAMT FEI STU.

5.1 Matematické modely pre digitalizáciu výrobných procesov s podporou Petriho sietí

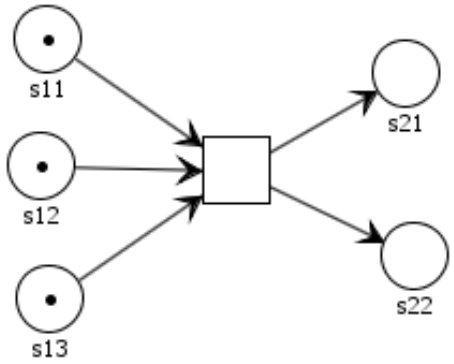
Pre vývoj moderných komplexných digitalizovaných výrobných procesov a ich projektovanie je v súčasnosti jednou z možností efektívne využívanie pokročilých štruktúr matematických modelov, ktoré sú založené napríklad na Petriho sieťach [32]. Petriho siete vzhľadom na svoju univerzálnosť pri modelovaní výrobných systémov, ktoré predstavujú najmä zložité diskrétne udalostné ale aj hybridné systémy, sú preto významnou oporou pri vytváraní nových platforiem digitalizovaných výrob. Kapitola bola vypracovaná najmä pomocou zdroja [56].

Diskrétne udalostné a hybridné systémy tvoria v súčasnosti význačnú skupinu rôznorodých systémov v Industry 4.0, ako sú systémy v automobilovom priemysle, výrobné systémy, zdravotníctve, službách a mnohé iné. Hybridné systémy sú kombináciou diskrétneho udalostného a spojitého systému. Nasledujúca časť sa zaoberá modernou alternatívou modelovania a riadenia takýchto systémov — prostredníctvom Petriho sietí vyšších úrovní. Petriho siete vyšších úrovní obsahujú silný matematický formalizmus, ktorý

znásobuje možnosti využitia modelu. Výsledkom je metóda a softvérový modul na podporu modelovania a riadenia udalostných a hybridných systémov s využitím časových Petriho sietí interpretovaných pre riadenie. Predložená metóda riadenia a podporný softvérový modul bol overený na riadení laboratórnych systémov pomocou mikrokontrolérov.

Ako je ilustrované na obrázku č. 5 v tejto učebnici, správanie diskrétneho udalostného systému možno charakterizovať diskrétnou stavovou premennou $x(t)$ (hodnoty $q_1, q_2 \dots q_n$) a zmenami jej hodnoty v čase. U diskrétnych udalostných systémoch sú pozorované len skokové zmeny stavov, ktoré sú spôsobené výskytom udalostí. Pri skúmaní diskrétnych systémov možno postupnosť okamihov $t_1, t_2 \dots t_n$ udalostí nahradiť aritmetickou postupnosťou $1, 2 \dots, n$. Systém je obvykle špecifikovaný konečným predpisom, na základe ktorého je možné príslušnú postupnosť stavov generovať, aj keď počet možných postupností je nekonečný. Jedným z nástrojov na modelovanie diskrétnych udalostných systémov sú **Petriho siete**.

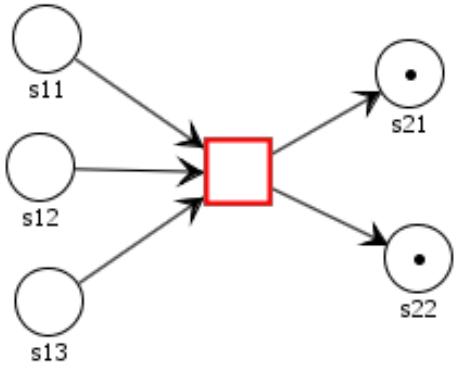
Predpokladajme, že systém rozložíme na subsystémy. Napríklad stav s_1 rozložíme na parciálne stavy s_{1_1} , s_{1_2} a s_{1_3}. Ďalej stav s_2 na s_{2_1} a s_{2_2}. Situácia je znázornená na obrázku č. 184.



Obrázok č. 184: Dekompozícia stavov na parciálne stavy [55]

Parciálnym stavom zodpovedajú malé kružnice. Tieto sa nazývajú **miesta** Petriho siete. Obdĺžnikom (alternatívne sa niekedy používa aj hrubá čiara) označujeme **prechody**. Prechody modelujú udalosti, ktoré môžu v systéme nastať. Aktuálny stav systému je reprezentovaný rozmiestnením **značiek (angl. tokens)**. Na obrázku značky, resp. tokeny, znázorňujeme zvyčajne ako malé čierne body. Prítomnosť tokenu v mieste v našom konkrétnom prípade modeluje skutočnosť, že daný aspekt stavu (napr. aktivácia) je v tomto okamihu splnený. Každý prechod má definované vstupné a výstupné miesta. Toto je v grafe znázornené orientovanými hranami. Týmto je deklarované, ktoré

aspekty stavu podmieňujú výskyt danej udalosti a ktoré aspekty sú výskytom tejto udalosti ovplyvnené. Udalosť môže byť uskutočnená len v prípade, ak sú všetky podmienky splnené. **Uskutočnením udalosti** (**spustením prechodu**) sa odstránia tokeny zo vstupných miest a umiestnia sa nové tokeny do výstupných miest.



Obrázok č. 185: Znázornenie situácie po spustení prechodu [55]

Z uvedeného vyplýva, že Petriho siete sú sofistikovanejším nástrojom ako napr. konečné automaty. Hlavnou myšlienkou Petriho sietí je reprezentovať stavy subsystémov oddelene. Týmto spôsobom vieme veľmi dobre modelovať a analyzovať distribuované aktivity systému. Distribuovateľnosť je jedna z hlavných charakteristík konkurentných systémov. Okrem klasických Petriho sietí poznáme aj **Petriho siete vyšších úrovní** (high-level Petri nets). Výhody spočívajú v opise časových vzťahov v systéme alebo napr. dátových typov. Poznáme stochastické Petriho siete, zovšeobecnené Petriho siete, hybridné Petriho siete a pod.

Rozvoj metód modelovania a riadenia digitalizovaných výrobných systémov v Industry 4.0, ktoré predstavujú diskrétne udalostné a hybridné systémy, patrí k moderným smerom v oblasti kybernetiky a mechatroniky. Na základe analýzy dostupnej literatúry a výskumných projektov na Slovensku a vo svete sa zistilo, že metódy modelovania a riadenia založené na využití Petriho sietí vyšších úrovní sú dostupné len ako teoretický koncept. V čase výskumu neexistovala softvérová aplikácia alebo prostredie, ktoré by umožňovalo modelovanie a riadenie diskrétnych udalostných a hybridných systémov s podporou Petriho sietí pomocou mikrokontrolérov. Dostupnosť a ich neustále sa zväčšujúci výkon ich predurčujú využiť nielen vo výskume v tejto oblasti, ale aj v priemyselnej praxi v rámci konceptu Industry 4.0, kde je miniaturizácia riadiacich systémov veľmi žiadaná.

Hybridný systém spája v sebe dve odlišné dynamiky — dynamiku diskrétne udalostnú a spojitú. Riadenie takýchto systémov prináša nové výzvy, nakoľko je potrebné zosúladiť

metódy riadenia udalostných systémov, kde je matematický formalizmus veľkým benefitom, s metódami riadenia spojitých systémov vychádzajúce z klasickej teórie riadenia. Korektným návrhom metodiky a softvérového modulu možno tieto prístupy synergicky kombinovať. Získa sa tak funkčný a originálny systém pre riadenie, ktorý dovolí zosúladiť metódy riadenia diskrétneho udalostného systému s metódami riadenia spojitého systému (napr. prostredníctvom PID regulátora). Efektívna kooperácia uvedených prístupov umožní riadiť hybridný systém.

Uvedená metóda je využiteľná napríklad pri systémoch, kde je treba na základe rozličných stavov istého podsystému využiť rozdielne algoritmy riadenia spojitého podsystému (napr. PID algoritmy-regulátory s rozdielnymi parametrami). Výhoda tejto metódy spočíva v tom, že koncept a formalizmus Petriho sietí dokáže takéto riadiace pravidlá, ktoré súvisia so stavom systému, pokryť veľmi robustným, efektívnym a prehľadným (grafickým) spôsobom. Napríklad je možné na ilustráciu uviesť riadenie motora helikoptéry, pričom v závislosti od výšky, v ktorej sa helikoptéra nachádza, je treba prepínanie medzi rôznymi režimami riadenia.

5.1.1 Softvérový nástroj PN2ARDUINO

Existuje viac konceptov riadenia prostredníctvom Petriho sietí. Petriho sieť ako riadiacu logiku je potrebné prepojiť s riadeným systémom a to napríklad prostredníctvom nízkonákladového riadiaceho systému — mikrokontroléra. Podstatným bodom návrhu riadiaceho systému je položenie si otázky, či umiestniť logiku Petriho siete priamo do mikrokontroléra alebo do počítača, ktorý by s mikrokontrolérom komunikoval. Obidva spomínané prístupy majú svoje výhody a nevýhody.

Ak logika Petrihu sietí umiestnime do mikrokontroléra (obrázok č. 186), hlavnou výhodou tohto prístupu je nezávislosť riadiacej jednotky od softvérovej aplikácie, teda programu na počítači. Logiku Petriho sietí je síce potrebné namodelovať prostredníctvom počítača, ale následne je táto logika preložená do programového kódu, ktorý je nahraný priamo na mikrokontrolér. Potom je možné mikrokontrolér od počítača odpojiť. Výhodou je tiež schopnosť reakcie na podnety z vonkajšieho prostredia v reálnom čase. Medzi nevýhody možno zaradiť obmedzené pamäťové a výpočtové zdroje mikrokontroléra. Ďalšou značnou nevýhodou (hlavne počas ladenia návrhu riadiaceho systému) je nutnosť opakovaného kompilovania a nahrávania riadiaceho programu do mikrokontroléra.

Ak je logika Petriho sietí umiestnená do špecializovanej softvérovej aplikácie na počítači, takéto riešenie prináša možnosť riadiť systém priamo z tohto počítača (obrázok č. 187). Na mikrokontroléri je uložený iba program s komunikačným protokolom. Tento komunikačný protokol (v tomto prípade Firmata [39]) je využitý pre komunikáciu me-

Obrázok č. 186: Schéma navrhovaného riešenia — implementácia Petriho siete do mikrokontroléra [56]

dzi počítačom a mikrokontrolérom. Uvedené riešenie odstraňuje nutnosť prekompilovávania a nahrávania programu do mikrokontroléra počas ladiacej fázy návrhu riadiaceho systému. Ďalšou výhodou je odstránenie obmedzenia pamäťových a výpočtových zdrojov, nakoľko v porovnaní s mikrokontrolérom má bežný počítač tieto zdroje prakticky neobmedzené. Nevýhodou konceptu však je, že nedokážeme zaručiť reakcie na podnety v reálnom čase.

Obrázok č. 187: Schéma navrhovaného riešenia — implementácia Petriho siete do počítača [56]

Tabuľka č. 7: Porovnanie dvoch konceptov riadenia s využitím Petriho sietí [56]

Logika Petriho sietí v počítači	**Logika Petriho sietí v mikrokontroléri**
Limitované možnosti riadenia v reálnom čase	riadenie v reálnom čase
Dostupnosť väčšieho množstva výpočtových a pamäťových zdrojov	obmedzené výpočtové a pamäťové zdroje
Kód v mikrokontroléri nie je nutné počas vývoja rekompilovať	kód v mikrokontroléri je nutné počas vývoja opakovane rekompilovať
Počítač musí byť stále zapnutý	nezávislosť riadiacej jednotky

Originálna softvérová aplikácia PN2ARDUINO je navrhnutá na základe druhého konceptu. Nadradená riadiaca Petriho sieť beží na počítači. Ako základ tejto aplikácie bol použitý PNEditor [81]. Je to grafický editor určený pre modelovanie základných (P/T) Petriho sietí. PNEditor je napísaný v programovacom jazyku Java a ide o open-

source program. Výhodou je dobre štruktúrovaný kód a prehľadný návrh celej softvérovej aplikácie.

Na komunikáciu medzi vyvinutou softvérovou aplikáciou PN2ARDUINO a mikrokontrolérom bol využitý protokol Firmata. Jeho charakteristika bude uvedená v pokračovaní predloženej práce. Na strane mikrokontroléra Arduino ide o verziu 2.3.2 (Standard Firmata). Na strane Java aplikácie PN2ARDUINO bola použitá klientska knižnica Firmata4j (verzia 2.3.3). Výhodou predstaveného návrhu aplikácie je, že namiesto mikrokontroléra rodiny Arduino možno použiť akýkoľvek mikrokontrolér, ktorý podporuje protokol Firmata.

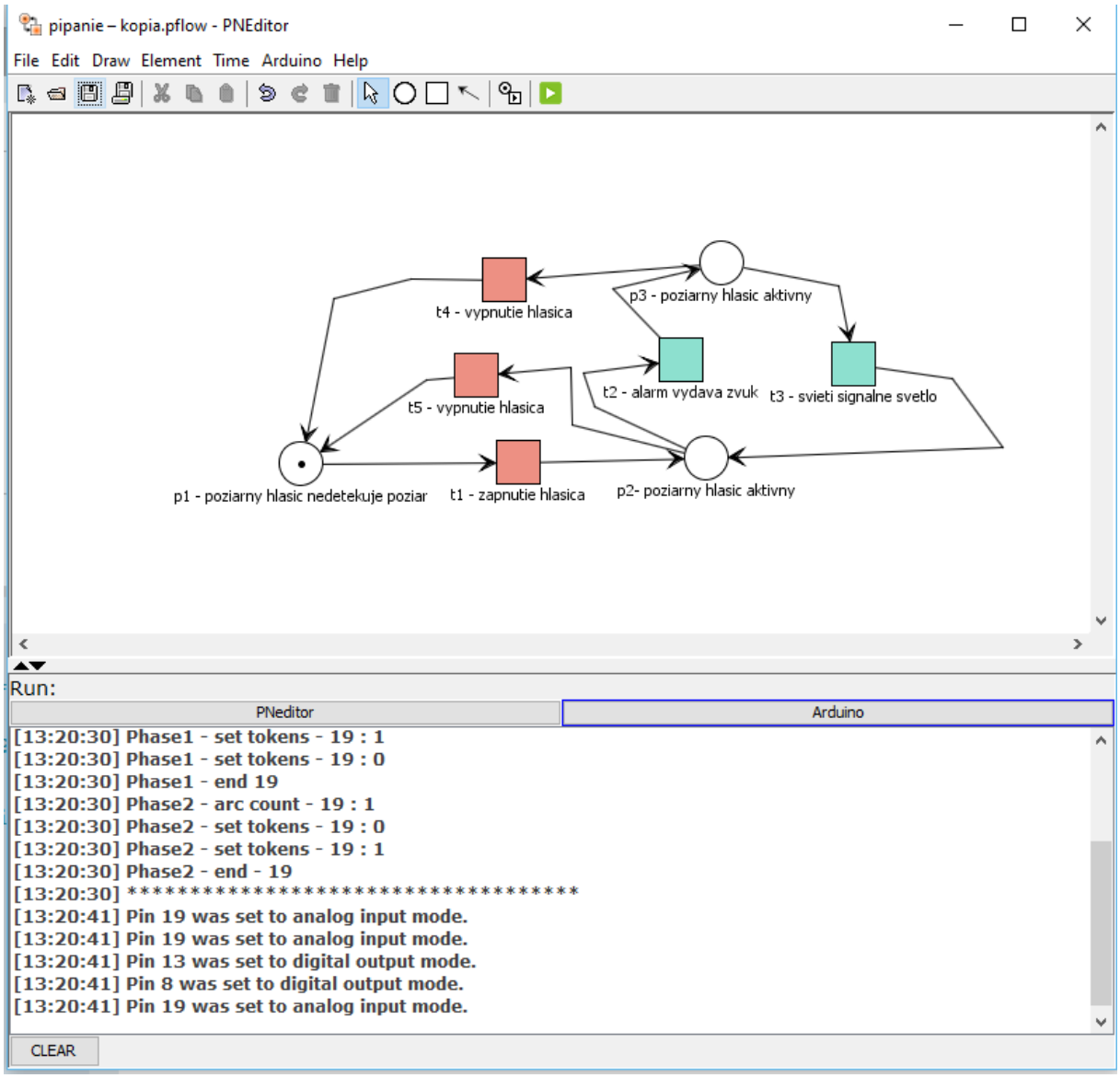
Nový softvérový modul PN2ARDUINO dopĺňa PNEditor o viacero funkcionalít. Na strane Petriho sietí ide hlavne o možnosť pridania oneskorenia na prechody a doplnenie vlastnosti kapacity pre miesta. Takisto bol implementovaný automatický režim spúšťania prechodov, a to práve pre potreby riadenia, nakoľko pôvodný PNEditor obsahoval len režim manuálny.

PN2ARDUINO do PNEditora ďalej prináša nový modul, ktorý zodpovedá za komunikáciu s kompatibilným mikrokontrolérom. Je zložený z dvoch základných časti. Prvá z nich je zodpovedná za vytvorenie spojenia s mikrokontrolérom. Toto zahŕňa nastavenie komunikačného (COM) portu, na ktorom sa mikrokontrolér nachádza. Druhá časť nám implementuje možnosť pridania Arduino komponentu na prechod alebo miesto Petriho siete. Podporované sú nasledovné typy komponentov:

- digitálny vstup a výstup;
- PWM výstup;
- analógový vstup;
- ovládanie servomotora;
- posielanie správ;
- posielanie vlastných správ typu SYSEX.

Overenie a demonštrácia metódy riadenia diskrétnych udalostných systémov pomocou originálneho softvérového modulu PN2ARDUINO bola realizovaná na navrhnutom laboratórnom systéme požiarneho hlásiča s mikrokontrolérom typu Arduino Uno. Pre demonštráciu navrhovanej metódy riadenia hybridných systémov bol vybraný jednosmerný motor s enkóderom.

Opis týchto prípadových štúdií a aj samotného nástroja je možné nájsť vo vedeckom časopise v zdroji [52].

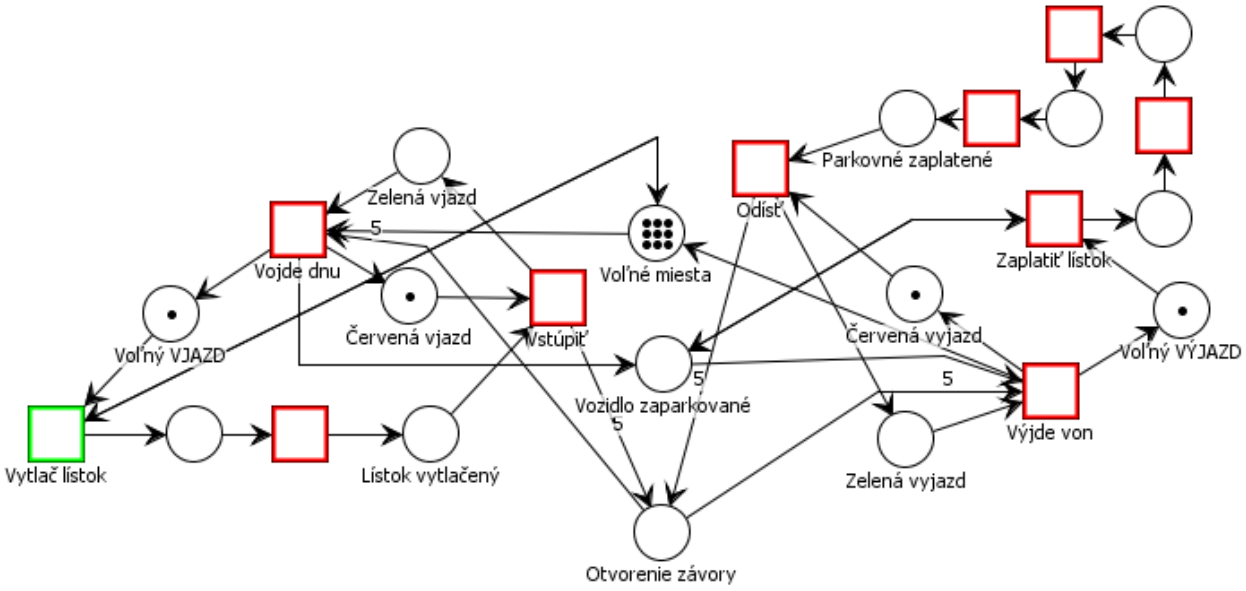


Obrázok č. 188: Prostredie softvérovej aplikácie PN2ARDUINO [55]

5.1.2 Softvérové nástroje PetriNet editor a PetriNet engine

PN2ARDUINO je softvérové riešenie na riadenie diskrétnych udalostných a hybridných systémov založené na koncepte, kedy je riadiaca logika umiestnená v počítači a komunikácia s mikrokontrolérom prebieha pomocou protokolu Firmata.

V rámci prác [9] a [96] bola navrhnutá softvérová aplikácia pre riadenie diskrétnych udalostných systémov pomocou Petriho sietí založená na odlišnom koncepte, kde je riadiaca logika nahraná priamo v mikrokontroléri.



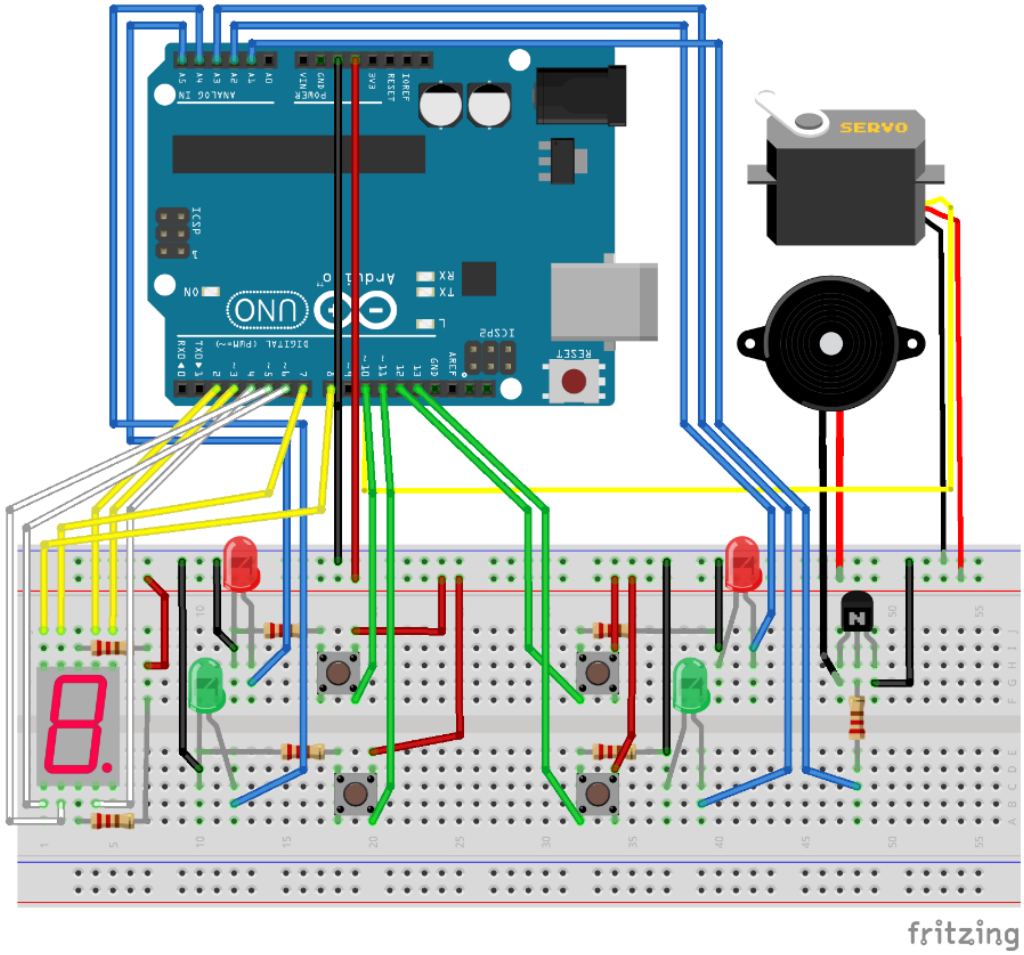
Obrázok č. 189: Model Petriho siete parkoviska [9]

Riešenie sa skladá z dvoch modulov: PetriNet editor a PetriNet engine. PetriNet editor je podobne ako PN2ARDUINO založený na editore PNEditor. Doňho bola implementovaná podpora prepojenia s mikrokontrolérmi rodiny Arduino, miest s kapacitami, prechody s prioritami a podpora pre časové Petriho siete. PetriNet engine vykonáva samotnú logiku siete a celý je implementovaný na spomínanom type mikrokontroléra. Po nahratí kódu teda umožňuje chod nezávislý od počítača. PetriNet engine má podporu viacerých stratégií výberu prechodu, ktorý sa má spustiť. Konkrétne podľa poradia, ako boli vytvorené, podľa priorít, round-robin a náhodne. Je takisto možné asociovať s každým prechodom oneskorenie. Toto oneskorenie je možné podľa požiadavky užívateľa a konkrétneho použitia aplikovať pred spustením, počas spustenia alebo po spustení prechodu.

Výhodou uvedeného prístupu je nezávislosť riadiaceho programu v mikrokontroléri od počítača. Ako už bolo uvedené, limitujúce sú jeho výpočtové a pamäťové kapacity.

V uvedených prácach [9] a [96] bolo opísané riešenie overené na laboratórnom modeli parkoviska (obrázok č. 189 a č. 190) a stieračov. V spomínaných prácach je možné

nájsť aj ďalšie detaily implementácie systému a používateľský manuál. Riešenie je takisto opísané vo vedeckom článku [61].



Obrázok č. 190: Schéma laboratórneho modelu parkoviska [9]

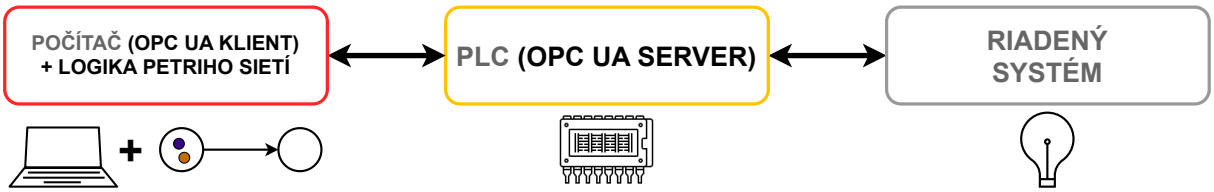
5.1.3 Modelovanie a riadenie diskrétnych udalostných a hybridných systémov s využitím Petriho sietí a OPC UA

Táto podkapitola predstavuje originálne softvérové riešenie prvýkrát predstavené v práci [68], ktoré využíva grafickú reprezentáciu formalizmu Petriho sietí v spojení s komunikačným štandardom OPC Unified Architecture. Táto kombinácia umožňuje používateľovi modelovať a riadiť diskrétne udalostné a hybridné systémy pomocou intuitívneho, užívateľsky prívetivého grafického rozhrania. Softvérová aplikácia tiež prináša nové možnosti vďaka implementácii rozšírenia formalizmu Petriho siete — **spojitých prvkov** (miest a prechodov), čím vznikajú **hybridné Petriho siete**, ktoré výrazne rozširujú oblasť systémov, ktoré možno pomocou tohto nástroja modelovať a riadiť. Vyvinutý softvérový nástroj bol úspešne overený pri riadení virtuálnych modelov systémov. Ponúka grafické prostredie pre návrh algoritmov riadenia diskrétnych udalostných a hybridných systémov, možno ho použiť na vzdelávanie, výskum a prax v oblasti

kyberneticko-fyzikálnych systémov (Industry 4.0).

Rovnako ako v predošlých podkapitolách, bol ako základ využitý editor Petriho sietí PNEditor. Na začiatku treba spomenúť hlavné aplikačné výhody riešenia vyvinutého ako rozšírenie PNEditora:

- Možnosť implementácie riadiaceho algoritmu pomocou Petriho sietí a súvisiacich výhod, ktoré tento matematický formalizmus so sebou prináša.
- Použitie hybridných Petriho sietí, ktoré prináša možnosť riadenia nielen diskrétnych udalostných systémov, ale aj systémov so spojitými zložkami.
- Použitie Petriho sietí a ich grafické znázornenie umožňuje jednoduchú a rýchlu zmenu algoritmu riadenia pomocou grafického editora.
- Vyvinuté riešenie komunikuje pomocou univerzálneho OPC UA komunikačného štandardu.

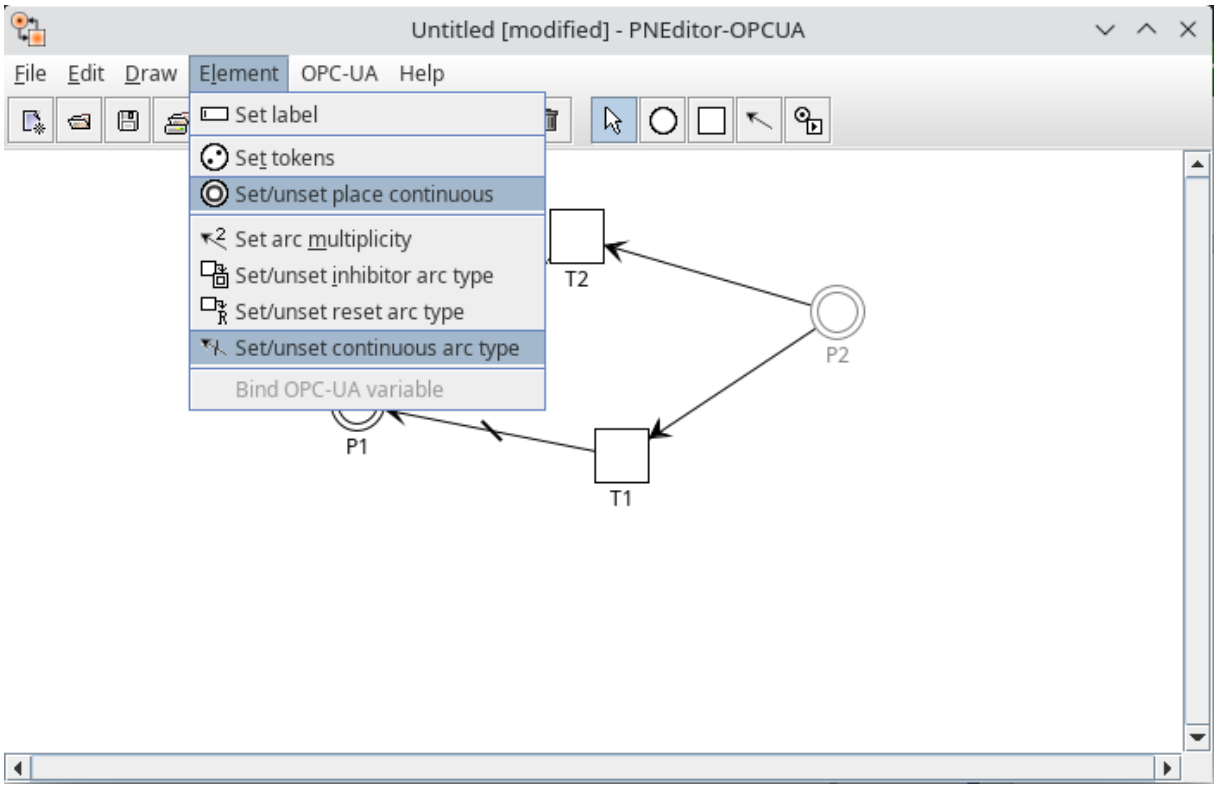


Obrázok č. 191: Schéma navrhovaného riešenia — Petriho siete v kombinácií s komunikáciou cez OPC UA [68]

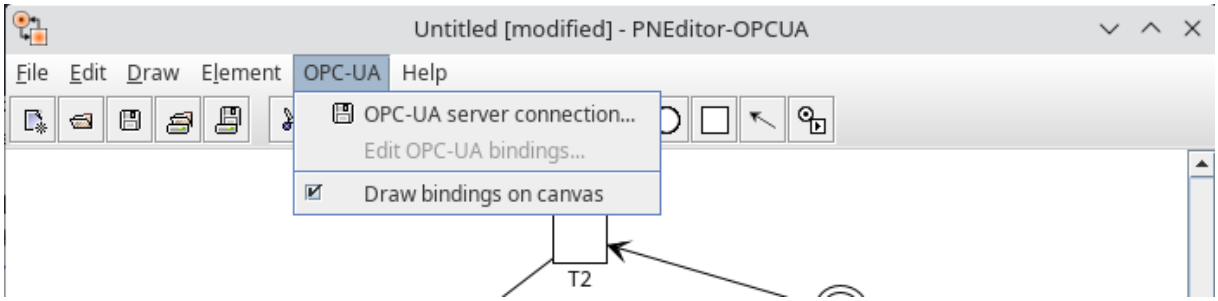
Schému navrhovaného riešenia je možné vidieť na obrázku č. 191. Riešenie je teda z hľadiska architektúry podobné ako v prípade PN2ARDUINO. Nové riešenie však namiesto veľmi úzko zameraného komunikačného protokolu Firmata využíva široko akceptovaný štandard **OPC Unified Architecture**. Takto je možné pomocou logiky Petriho sietí riadiť praticky akýkoľvek diskrétny udalostný alebo hybridný systém, ktorý je pripojený k OPC UA serveru. Riadenie hybridných systémov (diskrétnych udalostných systémov so spojitou zložkou) je zabezpečené tým, že do editora je tiež pridaná podpora **spojitých miest, prechodov a hrán Petriho siete**. Táto nadstavba PNEditora teda podporuje **hybridné Petriho siete**.

Na obrázku č. 192 je možné vidieť grafické rozhranie editora, kde je možné nastaviť vybrané prvky Petriho siete ako spojité. Spojité miesta a prechody sú označené dvojitým kruhom, resp. obdĺžnikom.

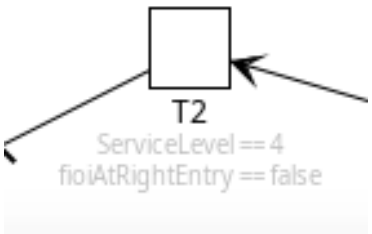
Pre zabezpečenie komunikácie Petriho siete s OPC UA serverom je možné vybrané mieste a prechody zviazať s OPC UA premennými adresného priestoru OPC UA servera.



Obrázok č. 192: Grafické rozhranie nadstavby PNEditora — nastavovanie hrany/miesta Petriho siete na spojitý typ [68]



Obrázok č. 193: Grafické rozhranie nadstavby PNEditora — pripojenie na OPC UA server [68]



Obrázok č. 194: Grafické rozhranie nadstavby PNEditora — prechod s podmienkou súvisiacou s premennou OPC UA adresného priestoru [68]

Pripojenie na OPC UA server je možné vidieť na obrázku č. 193 a zviazanie konkrétneho prechodu s premennou OPC UA je možné vidieť na obrázku č. 194.

Nástroj bol overený v rámci prípadových štúdií riadenia virtuálneho diskrétneho udalostného systému a hybridného systému, ktoré boli realizované v simulačnom softvéri Factory I/O. Ich opis je možné nájsť v práci [68] a vedeckom článku [60].

Videoukážky je možné nájsť na adresách:

- `https://www.youtube.com/watch?v=SHjZ7VDZ8E0`
- `https://www.youtube.com/watch?v=U33glvqCctw`

5.2 Ovládanie a monitorovanie mechatronických systémov s využitím IoT a rozšírenej reality

Ešte v nedávnej minulosti sa mechatronické systémy v sieťach IoT ovládali a diagnostikoval ich stav prakticky výhradne prostredníctvom priemyselných HMI panelov, konzolových, webových alebo mobilných aplikácií. V prípade použitia týchto konvenčných metód ovládania a diagnostiky mechatronických systémov v sieťach IoT môže byť tento spôsob v menšej miestnosti plne vyhovujúci. Nakoľko zoznam zariadení sa zmestí na jednu obrazovku, tak vieme monitorovať stav a ovládať tieto zariadenia prakticky okamžite. Ak však ide o viacero miestností alebo budov, čo v prípade digitálnych tovární nemožno vylúčiť, tak takýto spôsob interakcie s mechatronickými systémami sa stáva ťažkopádnym. V tomto prípade sa naskytuje možnosť využitia moderných trendov a pokročilých digitálnych technológií z oblasti *počítačom generovanej reality*. Pomocou týchto technológií dokážeme vkladať digitálne (počítačom generované) objekty do reálneho sveta.

Pracovisko ÚAMT FEI STU sa významnou mierou podieľalo na vývoji originálnych softvérových modulov pre počítačom generovanú realitu, preto obsahom tejto podkapitoly je charakteristika návrhu a implementácie novej metodiky ovládania a monitorovania mechatronických systémov pripojených na sieť IoT s využitím vybraného segmentu počítačom generovanej reality pre tvorbu inovatívnej formy HMI (rozhranie človek-stroj). Podkapitola vychádza najmä z práce [83].

Rozvoj metód ovládania a monitorovania mechatronických systémov s využitím nových informačno-komunikačných technológií patrí k moderným smerom v oblasti kybernetiky, automatizácie a mechatroniky. Na základe analýzy dostupných literárnych zdrojov a najnovších výskumných projektov na Slovensku a aj vo svete sa počas riešenia opisovaného projektu zistilo, že metódy ovládania a monitorovania mechatronických systémov pripojených do sietí Internet of Things využívajúce počítačom generovanú realitu boli realizované vo forme rozličných prototypových riešení pre jeden typ zaria-

dení, resp. nanajvýš vo forme uzavretých jednoúčelových aplikačných systémov. Tieto systémy nemali ucelenú formu, neumožňovali jednoduché pridávanie možností ovládať a monitorovať ďalšie mechatronické zariadenia bez úpravy klientskej softvérovej aplikácie. Takéto systémy nebolo možné považovať za zovšeobecnené a modulárne riešenia. V rámci exkurzií a diskusií s priemyselnými partnermi realizovanými pracovníkmi ÚAMT FEI STU sa ukázalo, že o takéto komplexné riešenia je záujem. V kontexte prebiehajúcej priemyselnej revolúcie Industry 4.0 sa už aj malé a stredné podniky zaujímajú o možnosti implementácie moderných digitálnych technológií, ako sú Internet of Things, cloud a počítačom generovaná realita, do svojich procesov [56].

Ovládanie a monitorovanie mechatronických systémov pripojených v sieťach IoT s využitím vybraného segmentu počítačom generovanej reality prináša so sebou nové výzvy, nakoľko tento koncept spája prácu s hardvérom, jeho mechanickými časťami, mikrokontrolérmi a elektronickými systémami, 3D enginom pre počítačom generovanú realitu, mobilnými zariadeniami a komunikačnými protokolmi v rámci Internet of Things a cloudom. Správnym návrhom metodiky ovládania a monitorovania mechatronických IoT systémov a podporného softvérového modulu je možné tieto prístupy a digitálne technológie synergicky kombinovať. Týmto sa získal funkčný, originálny a tiež modulárny systém využiteľný pre vybranú triedu mechatronických systémov.

Pred vytvorením systému bolo treba rozhodnúť, akým spôsobom budú rozpoznávané a identifikované jednotlivé mechatronické zariadenia, ktoré budú v rozšírenej realite slúžiť na ukotvenie počítačom generovaných prvkov. Existuje dnes viacero alternatív [10]:

- **Pomocou QR kódu** — Termín QR kód pochádza z anglického výrazu *Quick Response* (rýchla reakcia), nakoľko QR kód bol vyvinutý pre rýchle dekódovanie údajov. Ide o dvojrozmerný kód, ktorý môže byť v digitálnej forme alebo vytlačený na papieri. Po nasnímaní je potrebné tento kód potrebné dekódovať napríklad pomocou mobilného zariadenia. Samotný kód predstavuje čiernobielu štvorcovú maticu skladajúcu sa zo štvorcových modulov. Medzi výhody QR kódu patrí jeho rýchle vygenerovanie pri rozširovaní systému. Ďalšou z výhod je, že každý snímač, aktuátor alebo zariadenie môže mať unikátny QR kód. Takto možno jednoducho rozlíšiť aj tvarovo rovnaké objekty. Nevýhodou však je, že pri snímaní je nutné držať mobilné zariadenie paralelne s QR kódom a takisto dostatočne blízko neho.

- **Pomocou obrázka** — Počítačom generované prvky je možné v rozšírenej alebo zmiešanej realite vygenerovať a ukotviť aj na základe 2D obrázka. Výhodou tohto spôsobu je, že vytváranie obrázkov je jednoduché a nie je potrebné žiadne špeciálne zariadenie. Existuje však viacero nevýhod. Podobne, ako v prípade QR kódu, pri

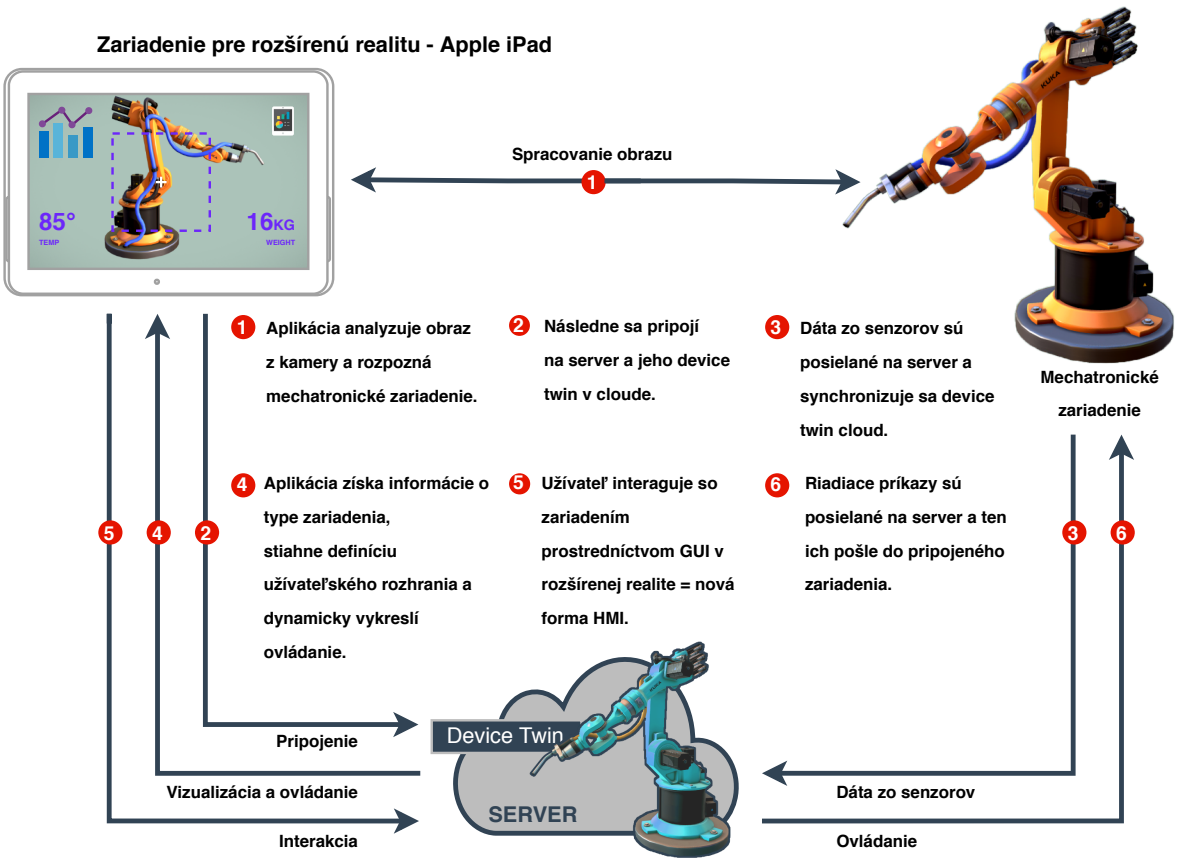
rozpoznávaní obrázka musíme držať mobilné zariadenie dostatočne blízko k objektu a toto zariadenie musí byť držané paralelne s obrázkom.
Po naštudovaní metodík vytvárania vhodných obrázkov sa zistilo, že obrázok musí spĺňať určité parametre. Jeho veľkosť, teda výška a šírka, musí byť od 500 do 1000 pixlov. Nemôže obsahovať opakujúce sa vzory, nízky kontrast a nevýraznú textúru. Farebnosť obrázka môže byť klamlivá, nakoľko farby, ktoré rozlíši ľudské oko veľmi dobre, môžu byť pre danú technológiu zameniteľné. Rozloženie textúrovanej časti by malo byť rovnomerné, malo by obsahovať málo textu a čo najmenej bielych častí [10].

- **Pomocou 3D modelu** — Zaujímavou možnosťou je vytváranie trojdimenzionálnej mapy na základe reálnych trojdimenzionálnych objektov (napr. mechatronických zariadení). Mobilná aplikácia na základe streamu z integrovanej kamery hľadá zhodu v reálne svete s vytvoreným modelom mechatronického zariadenia uloženým v databáze. Nespornou výhodou tejto metódy je možnosť snímania cieľového 3D objektu z ľubovoľného uhla a aj z väčšej vzdialenosti. Takisto mobilná aplikácia nestratí nájdený 3D objekt tak jednoducho, ako by to bolo v prípade predošlých metód. Nevýhodou je, že vytváranie cieľovej 3D mapy je zdĺhavá a náročná úloha.

Po dôkladnom zvážení výhod a nevýhod vyššie uvedených variánt, sa riešitelia rozhodli využiť rozpoznávanie pomocou 3D modelu. Týmto sa síce vytváranie a rozširovanie systému skomplikuje, no plynulý chod a intuitívne ovládanie mobilnej aplikácie a celého systému je v tomto prípade prioritou. Dnešné nástroje pre mapovanie a rozpoznávanie 3D objektov pre aplikácie počítačom generovanej reality sú však pomerne spoľahlivé a pokračuje trend ich zdokonaľovania. Využitie trojdimenzionálneho modelu a trojdimenzionálnej mapy bolo v problematike monitorovania a ovládania mechatronických systémov s využitím rozšírenej reality v tom čase originálne a bolo jedným z prínosov riešenia.

Na základe analýzy bol na pracovisku ÚAMT FEI STU navrhnutý koncept, ktorý je uvedený na obrázku č. 195.

1. **Softvérová aplikácia analyzuje obraz z kamery mobilného zariadenia a rozpozná mechatronický systém**
   Mobilná aplikácia pre zobrazenie rozšírenej reality rozpozná reálne mechatronické zariadenie s pomocou uloženej 3D mapy, ktorá sa vytvorila v nástroji Wikitude Studio. 3D mapa snímaného mechatronického zariadenia bola vytvorená prostredníctvom fotografií zachycujúcich zariadenia z viacerých uhlov. Následne knižnica pre rozšírenú a zmiešanú realitu Wikitude SDK dokáže interpretovať túto 3D mapu



Obrázok č. 195: Názorná schéma systému pre monitorovanie a ovládanie mechatronických systémov s využitím rozšírenej reality [56]

z databázy, ktorá je uložená v softvérovej aplikácií pre tablet Apple iPad. Výhoda tejto metódy je možnosť rozpoznávania objektu z akéhokoľvek uhla. Následne aj pri menšej viditeľnosti sa sledovanie nemusí prerušiť, keďže Wikitude si vie ukladať aj blízke okolie snímaného objektu. Realizácia navrhovaného riešenia je teda bez konvenčných metód rozpoznávania objektov spoliehajúcich sa na QR kódy, kedy je možné zariadenie rozpoznať len pri kolmom namierení na tento kód.

2. **Mobilné zariadenie sa pripojí na server a device twin mechatronického zariadenia v cloude**
   Mobilné zariadenie sa pripojí na cloud, kde má rozpoznaný mechatronický systém svoju digitálnu kópiu tzv. *device twin*[1].

3. **Dáta zo senzorov mechatronického zariadenia sú posielané na server a synchronizuje sa device twin v cloude**
   Mechatronické zariadenie automaticky odosiela údaje zo senzorov pod svojím identifikátorom na server, pričom tu sa tieto údaje aj ukladajú. Pre tento účel slúži databáza InfluxDB, určená pre časovo závislé dáta, ktoré je následne možné vizualizovať v prostredí Grafana. Zároveň sa digitálna kópia mechatronického zariadenia synchronizuje na úrovni Microsoft Azure Device Twin, čo zabezpečí viditeľnosť aktuálnych údajov aj v prostredí cloudu.

4. **Aplikácia získa informácie o type mechatronického zariadenia, stiahne definíciu užívateľského rozhrania a vykreslí grafického rozhranie pre ovládanie a monitorovanie systému**
   Navrhnutý systém funguje takým spôsobom, že mobilná aplikácia rozpozná mechatronické zariadenie a podľa jeho identifikátora získa unikátnu definičnú schému užívateľského rozhrania pre potreby jeho monitorovania a ovládania. Koncept definičných schém pre dynamické generovanie grafického užívateľského rozhrania v rozšírenej realite je jedným z pilierov modularity realizovaného riešenia a zároveň jedným z vedeckých a aplikačných prínosov. Mobilná aplikácia má prístup k týmto definičným schémam vďaka prepojeniu komunikácie na databázu. Prepojenie je realizované pomocou vizuálneho tokovo-orientovaného programovania v prostredí Node-RED, kde sa získa vhodná schéma na základe parametra.

5. **Užívateľ interaguje s mechatronickým zariadením prostredníctvom grafického rozhrania v rozšírenej realite — novej formy HMI**

[1]Tento pojem nemožno zamieňať s pojmom *digital twin*. Digital twin resp. digitálne dvojča musí poskytovať ďalšie pokročilé funkcie, ktoré sú využiteľné napríklad na prediktívnu údržbu alebo optimalizáciu kybernetico-fyzikálneho systému.

Na základe unikátnej definičnej schémy sa v mobilnej aplikácii zobrazí grafické užívateľské rozhranie v rozšírenej realite, ktoré sa skladá z dvoch častí. Prvá časť je diagnostická a zobrazuje aktuálne údaje z dostupných senzorov. Druhá časť je ovládacia a zobrazuje ovládacie prvky priamo určené pre dané mechatronické zariadenie. Následne je užívateľovi umožnené interagovať s mechatronickým zariadením prostredníctvom grafického rozhrania v rozšírenej realite, ktoré predstavuje jednu z nových moderných foriem rozhrania človek-stroj (HMI).

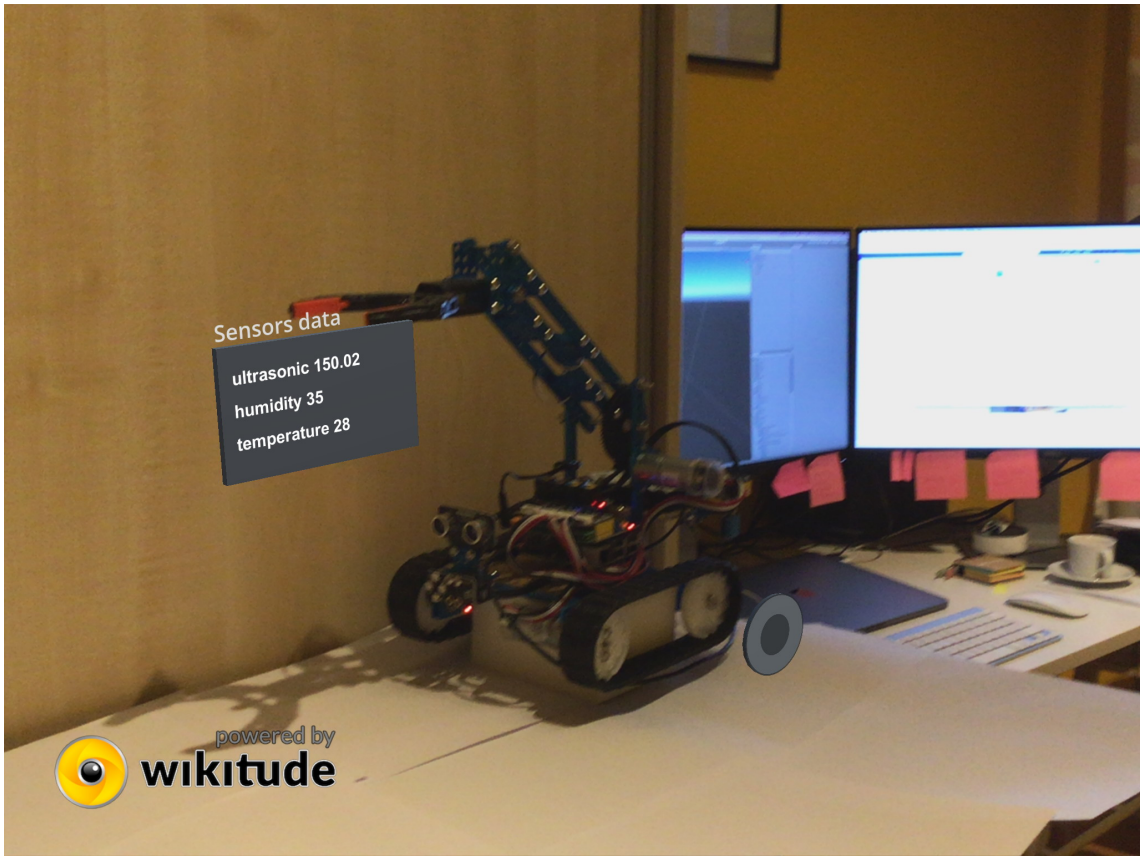
6. **Riadiace príkazy sú posielané na server a ten ich odošle do pripojeného mechatronického zariadenia**
   Riadiace príkazy sú posielané z mobilného zariadenia na server pomocou komunikačného protokolu MQTT, kde sa spracúvajú a vykonávajú. Softvérová aplikácia na mechatronickom zariadení počúva na MQTT tému (angl. topic) a následne tieto požiadavky posiela prostredníctvom sériovej komunikácie k senzorom a aktuátorom.

Z predošlého textu vyplýva, že pre dosiahnutie vytýčených cieľov bolo potrebné navrhnúť a realizovať komplexný hardvérovo-softvérový systém pre ovládanie a monitorovanie mechatronických IoT systémov s využitím počítačom generovanej reality a **konceptu definičných schém pre dynamické generovanie grafického používateľského rozhrania**. Systém bol otestovaný na laboratórnom mechatronickom systéme pripojenom na Internet of Things. Vývoj takéhoto komplexného systému bolo nutné realizovať vo viacerých paralelných líniách a bolo nutné pokryť všetky štyri oblasti mechatroniky (mechanika, elektronika, automatizácia a informačné-komunikačné technológie).

Rozpoznané mechatronické zariadenie s dynamicky generovaným GUI v rozšírenej realite je možné vidieť na obrázku č. 196 alebo na nasledovnom videu: `https://www.youtube.com/watch?v=moe1Y_EBex0` .

Opis celého riešenia je možné nájsť v práci [83] alebo v časopise vo vedeckom článku [84]. Podobnú prípadovú štúdiu z prostredia inteligentnej domácnosti je tiež možné nájsť v práci [10] alebo v článku [11].

Modifikácia tohto konceptu bola neskôr predstavená v publikácii [100], pričom toto riešenie bolo modifikované pre priemyselné prostredie. Namiesto mikrokontrolérov rodiny Arduino boli využité PLC značky Siemens. Namiesto tabletu s podporou rozšírenej reality bol využitý headset pre **zmiešanú realitu Microsoft HoloLens 2**, čo prinieslo **lepšiu ergonómiu** riešenia, nakoľko ruky pracovníka v tomto prípade zostávajú voľné. Týmto sa riešenie svojou filozofiou posúva smerom k **Industry 5.0**. Tento nadchádzajúci koncept bude stručne opísaný v závere učebnice.



Obrázok č. 196: Rozpoznané mechatronické zariadenie s dynamicky generovaným GUI v rozšírenej realite [84]

5.3 Nízkonákladová edukačná súprava pre výučbu hlbokého učenia pre aplikácie kontroly kvality

Tento projekt sa zaoberal návrhom cenovo dostupnej vzdelávacej súpravy zameranej na vybrané digitálne a inteligentné technológie pre Industry 4.0, ako sú **konvolučné neurónové siete**, **hlboké učenie** a **strojové videnie** pre aplikácie **kontroly kvality** [73].

Koncept Industry 4.0 so sebou prináša mnohé výzvy a príležitosti pre mladých nadšených inžinierov a inžinierky. Aby však mohli tieto výzvy zvládnuť, potrebujú mať v tejto oblasti pevné základy, a práve tie im môže poskytnúť škola, kde majú možnosť oboznámiť sa s problematikou a vyskúšať si riešenie rôznych úloh v tejto oblasti. Riešenie problémov si však vyžaduje rôzne hardvérové a softvérové nástroje, ktoré sú často drahé, nie vždy ľahko dostupné a je potrebné ich neustále zdokonaľovať s rozvojom nových technológií.

Univerzity zdôrazňujú svoju úlohu ako testovacie pracoviská inovácií a vychovávatelia budúcich generácií pri vývoji nových technológií. Tradičné vzdelávanie významne prispelo k súčasnej úrovni hospodárskeho rozvoja a technologického pokroku. V dôsledku toho je začlenenie koncepcie Industry 4.0 do technických študijných programov jednou z hlavných priorít akademických inštitúcií. Na základe prehľadu dostupnej literatúry a aktuálnych výskumných projektov možno konštatovať, že používanie lacných zariadení a navrhovanie lacných zostáv je typickým javom v akademickej a univerzitnej praxi. Zo zistení prieskumu vyplýva, že v kontexte Industry 4.0 možno na vzdelávanie v oblasti high-tech technológií využívať nízkonákladové zariadenia. Na akademickej pôde sa ukázalo, že obľúbený je hardvér ako napríklad Arduino alebo stavebnica LEGO. Výsledok tohto prehľadu povzbudil k návrhu vlastnej nízkonákladovej stavebnice. Okrem toho bolo zistené, že témy, ktorými sa chceli autori zaoberať (hlboké učenie, strojové videnie a kontrola kvality), nie sú v týchto prácach zahrnuté, čo naznačuje, že zostavená stavebnica by mohla byť v týchto oblastiach užitočná. Navyše potreba vzdelávania v oblasti nových digitálnych a inteligentných technológií sa netýka len vysokých škôl. Je potrebné pokryť celú škálu úrovní vzdelávania od základného, stredoškolského a vysokoškolského vzdelávania až po postgraduálne celoživotné vzdelávanie odborníkov s technickým vzdelaním.

V tomto projekte bol predstavený vlastný návrh a implementácia **nízkonákladovej edukačnej súpravy**. Na základe prehľadu literatúry bolo zistené, že súčasné práce a nízkonákladové riešenia sa nezaoberajú témami, ako sú hlboké učenie, konvolučné neurónové siete, strojové videnie a kontrola kvality. To bolo podnetom na vytvorenie nového nízkonákladového vzdelávacieho riešenia, ktoré by mohlo byť originálnym riešením a obohatiť súčasný stav v tejto oblasti. Stavebnica je jednoduchá, názorná, ľahko modi-

fikovateľná a pre študentov zaujímavá a príťažlivá na prácu, pretože kombinuje prvky elektroniky (Arduino), mechaniky (výrobná linka), riadenia (senzory a akčné členy), informatiky (konvolučné neurónové siete, grafické rozhranie) a komunikácie. Ide teda o **celé spektrum mechatroniky**. Vzdelávacia súprava využíva lacné a ľahko dostupné komponenty, ako napríklad Arduino, stavebnicu LEGO a kameru smartfónu, aby sa zabezpečila jej modifikovateľnosť a dostupnosť pre školy. Pomocou navrhovanej súpravy možno riešiť rôzne úlohy kontroly kvality výrobkov pomocou konvolučných neurónových sietí s podporou strojového videnia, ako sú binárna klasifikácia, klasifikácia viacerých tried, aplikácie YOLO v reálnom čase alebo úlohy klasifikácie jednej triedy s cieľom rozlíšiť požadovaný stav od akéhokoľvek iného, neželaného stavu.

Ako už bolo uvedené, cieľom bolo navrhnúť výučbovú súpravu na účely výučby strojového videnia s podporou hlbokého učenia, ktorá by bola jednoduchá, lacná, názorná, ľahko modifikovateľná, a čo je najdôležitejšie, aby sa s ňou študentom pracovalo dobre a aby podnecovala ich túžbu po skúmaní. Úmerne tomuto cieľu boli zvolené aj rôzne úrovne náročnosti úloh:

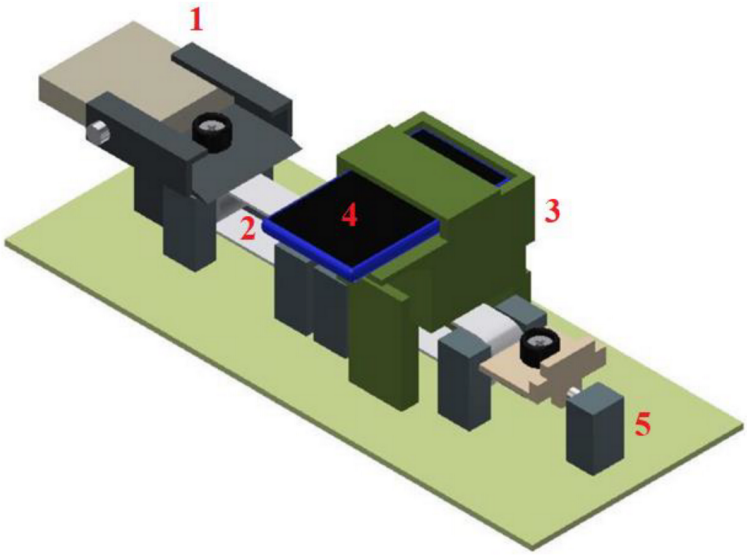
- jednoduchá binárna klasifikačná úloha (kontrola kvality stavu OK/NOK výrobku, pričom natrénovaná konvolučná neurónová sieť pozná oba stavy);
- zložitejšia úloha klasifikácie jednej triedy (kontrola kvality stavu OK/NOK výrobku, pričom vyškolená konvolučná neurónová sieť pozná len stav OK a musí správne rozlíšiť všetky ostatné výrobky ako NOK).

Keďže bolo cieľom, aby súprava bola v každom ohľade jednoduchá, návrh výrobnej linky vytvorený v prostredí Inventor (obrázok č. 197) bol jednoduchý a zrozumiteľný. Konštrukcia pozostáva z nakladacieho posuvníka (1), ktorý sa používa na vpúšťanie výrobku na dopravný pás (2). Potom, keď sa výrobok na páse dostane do kamerového tunela (3), do smartfónu (4) sa vyšle signál na vytvorenie obrazu, ktorý sa ďalej vyhodnocuje konvolučnou neurónovou sieťou v počítači. Na základe predpovede konvolučnej neurónovej siete triediaci mechanizmus (5) na konci linky výrobok zatriedi.

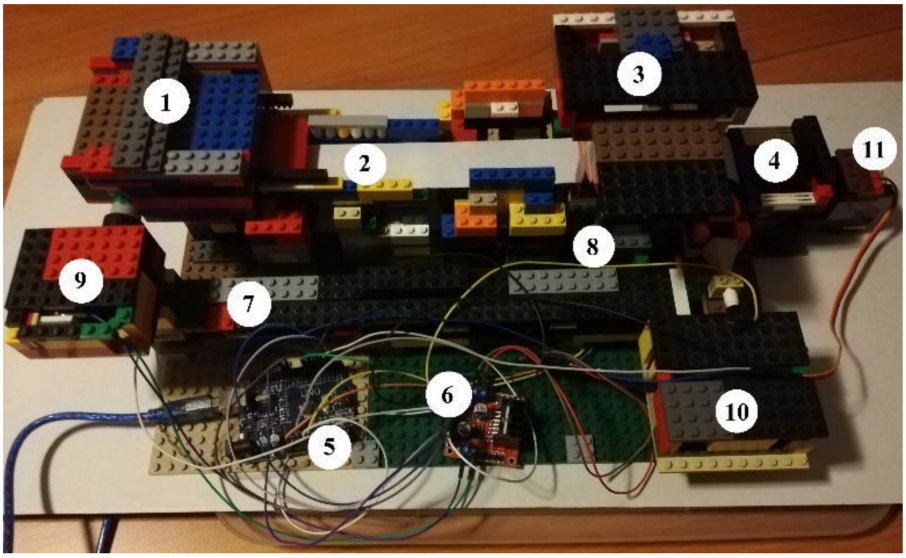
Aby bolo zabezpečené, že komponenty vzdelávacej súpravy budú jednoduché a ľahko dostupné, boli vybrané nasledovné komponenty:

- stavebnica LEGO na stavbu linky;
- mikrokontrolér Arduino Uno na ovládanie senzorov a aktuátorov;
- fotoaparát smartfónu na snímanie obrázkov výrobkov na dopravníku.

Vychádzajúc z navrhnutého modelu výrobnej linky a vybraných komponentov, ktoré by mali linku tvoriť, bol vytvorený fyzický model výrobnej linky (obrázok č. 198). Vý-

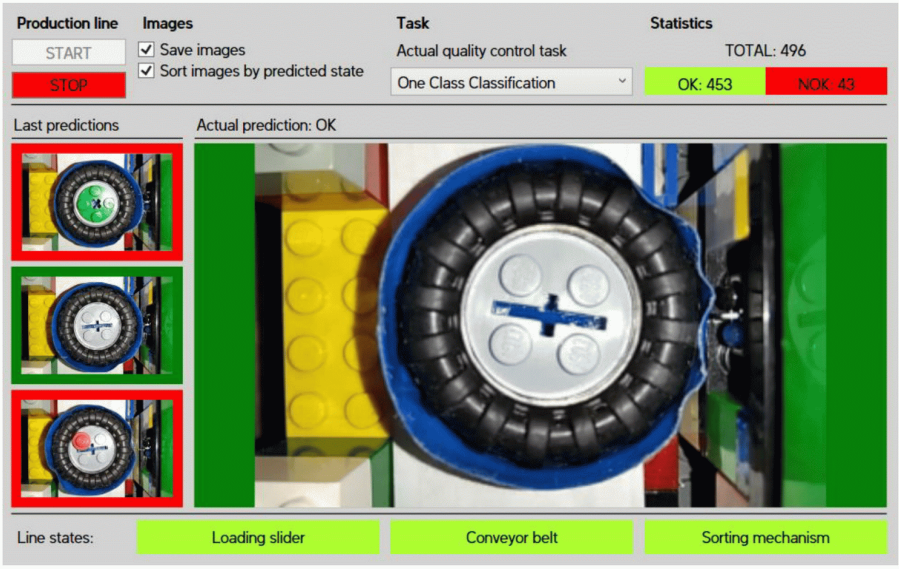


Obrázok č. 197: Návrh výrobnej linky v softvérovej aplikácii Inventor: 1 — nakladací posuvník, 2 — prepravný pás, 3 — stojan na kamery v spojení s tunelom na kamery, 4 — smartfón s kamerou, 5 — triediaci mechanizmus [73]



Obrázok č. 198: Výrobná linka zostavená zo stavebnice LEGO — pohľad spredu: 1 — nakladací posuvník, 2 — prepravný pás, 3 — stojan na kameru v spojení s tunelom na kamery, 4 — triediaci mechanizmus, 5 — Arduino UNO, 6 — dvojitý H-most L298N, 7 — batériový blok, 8 — infračervený senzor prekážok TCRT5000, 9, 10 — servomotory DS04 360, 11 — servomotor FS90R [73]

robná linka zostavená zo stavebnice LEGO pozostáva z nakladacieho posuvníka (1), dopravníkového pásu (2), stojana na kameru v kombinácii s kamerovým tunelom (3), triediaceho mechanizmu (4), mikrokontroléra Arduino Uno (5) na ovládanie snímačov a aktuátorov, ale aj na komunikáciu s počítačom, H-mostíka L298N (6) na ovládanie motorov linky, batériový blok (7) na napájanie motorov linky, infračervený (IR) snímač prekážok TCRT5000 (8) zabudovaný do nohy kamerového tunela na zaznamenávanie prítomnosti výrobku v kamerovom tuneli a servomotory ovládajúce pohyby linky (9 — 11) — dvakrát typ DS04 360 pre nakladací posuvník a dopravníkový pás a jeden typ FS90R pre triediaci mechanizmus.



Obrázok č. 199: Obslužná aplikácia s GUI vytvorená pomocou knižnice tried Windows Presentation Foundation [73]

Sada pozostáva aj zo softvérových aplikácií vyvinutých v programovacích jazykoch Python (hlboké neurónové siete) a C# (obslužná aplikácia s GUI — obrázok č. 199).

Kompletný opis sady aj zo softvérovej stránky je možné nájsť v časopise vo vedeckom článku [73].

Videoukážky je možné nájsť na adresách:

- `https://www.youtube.com/watch?v=7oi4GxHWsEQ`
- `https://www.youtube.com/watch?v=1dwt6GVVInI`

5.4 Edukačno-vývojové platformy pre digitálne technológie Industry 4.0

Opisovaný projekt sa zaoberal vytvorením modernej vedomostnej a experimentálnej základne pre výučbu digitálnych technológií, ktorá je plne v súlade s konceptom Industry 4.0 a zároveň je ekonomicky dostupná. V rámci projektu boli vybudované **štyri edukačno-vývojové platformy** (**EVP**) pre Industry 4.0 na báze reálnych riadiacich a komunikačných systémov prepojených s virtuálnymi technológiami. Bola zverejnená vytvorená metodika a poznatky pre prácu s digitálnymi technológiami, ktorá umožňuje na týchto platformách vzdelávať študentov a tvorivých odborníkov z praxe pre priemyselné aplikácie, ako sú digitálne dvojča, virtuálna a zmiešaná realita, jednotná interoperabilná komunikácia od senzorov až po cloud, edge a cloud computing (v spojení s kontajnerizáciou pre aplikáciu umelej inteligencie pre ovládanie zariadení ľudským hlasom) a moderné HMI realizované pomocou smart tabletov alebo headsetov zmiešanej reality série HoloLens.

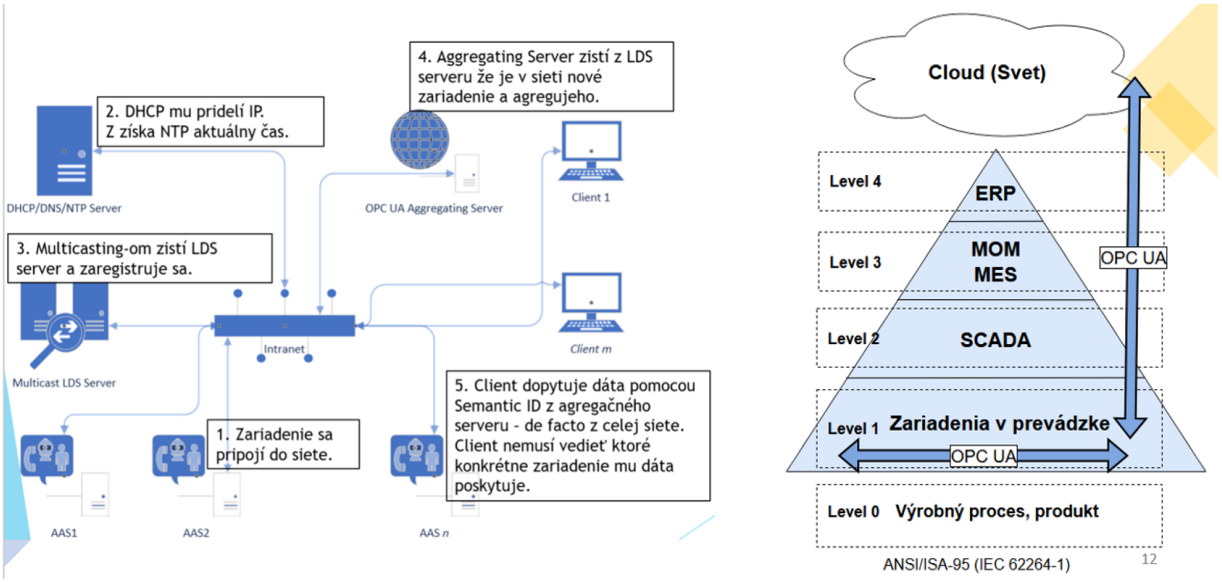
Projekt prakticky umožnil overiť nové metódy výučby a organizáciu práce koncipovanú za účelom posilniť kompetencie a syntetické schopnosti študentov nielen chápať, ale aj navrhovať a vyvíjať komponenty pre komplexné potreby Industry 4.0. Cieľom bolo vytvoriť experimentálnu bázu vo forme konkrétnych laboratórnych pracovísk tzv. EVP. Výsledky pedagogickej a výskumnej činnosti boli publikované vo vedeckých prácach a v zborníkoch z vedeckých konferencií.

Nasleduje zoznam jednotlivých EVP:

1. **EVP pre interoperabilitu**: interoperabilná OPC UA komunikačná sieť založená na agregačnom OPC UA serveri;
2. **EVP pre Industry 4.0 komponenty**: využitie fyzických modelov výrobných liniek pre demonštráciu tvorby digitálnych dvojčiat aj s využitím Asset Administrative Shell (AAS);
3. **EVP pre virtuálnu a zmiešanú realitu**: platforma pre ovládanie a monitorovanie zariadení pomocou zmiešanej reality — moderné HMI (HMI má dvojitú rolu — edukačnú a operátorskú);
4. **EVP pre orientáciu na služby v reálnom čase**: platforma založená na cloud, edge computingu a kontajnerizácii pre ovládanie zariadení hlasovými povelmi.

5.4.1 Edukačno-vývojová platforma pre interoperabilitu

Interoperabilita založená na OPC UA štandarde je kľúčová pre Industry 4.0. Postup pri agregovaní OPC UA servera na sieti do adresného priestoru agregačného serveru OPC UA Industrial Communication Platform (OPC UA ICP) je uvedený na obrázku č. 200 [79].



Obrázok č. 200: Sieť OPC UA a metodický postup pri agregovaní nového zariadenia AAS do OPC UA servera

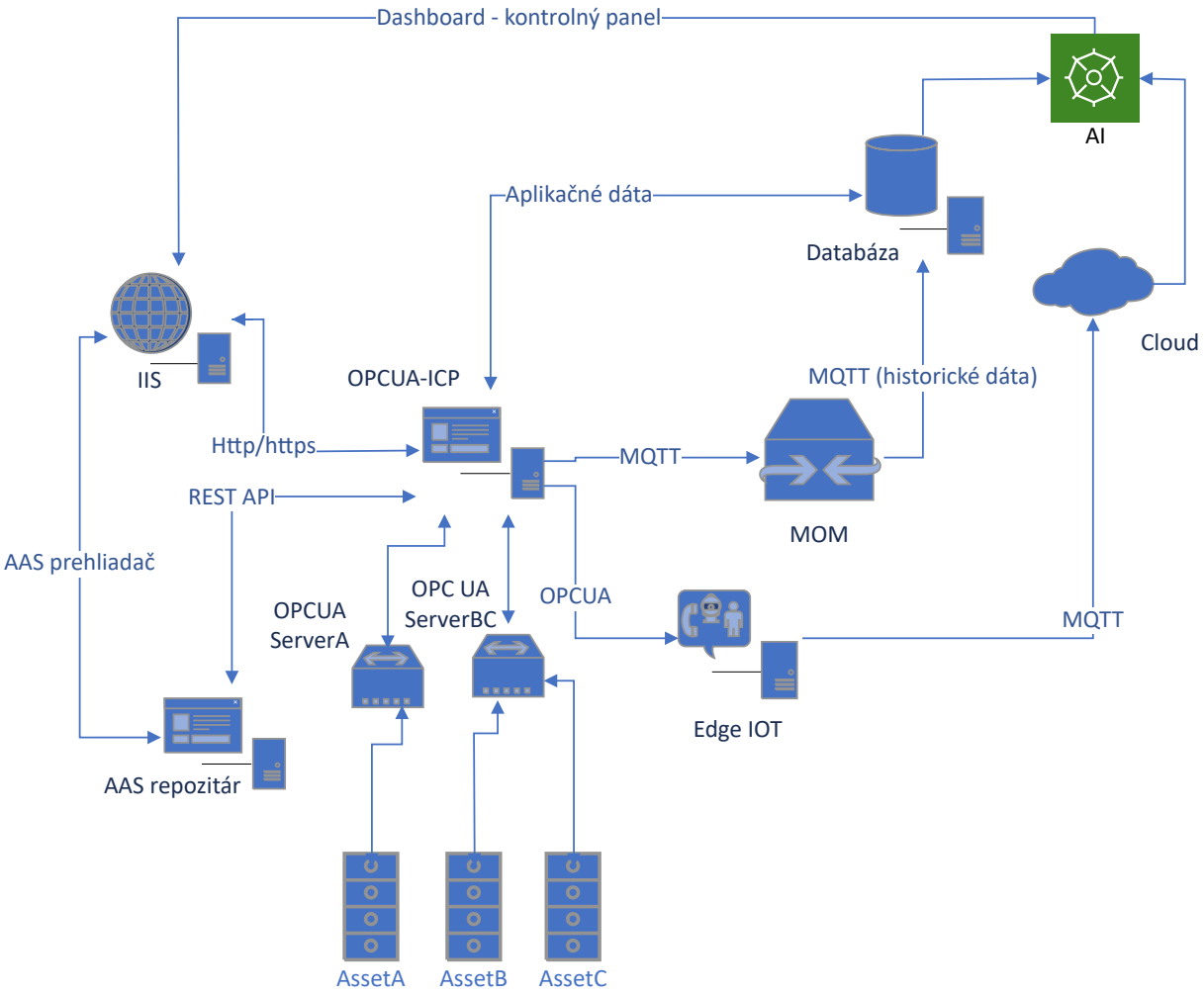
Funkcionalitu, ktorú možno označiť ako *Plug and Produce* tvorí niekoľko častí a krokov:

1. *Local Discovery Server with Multicast Extensions* (LDS ME) objaví OPC UA server v sieti.
2. OPC UA ICP, ktorý agreguje objavený OPC UA server.
3. *Asset Administration Shell* (AAS — administratívna schránka aktív) repozitár, ktorý obsahuje digitálne dvojčatá pridaných zariadení a ich rozhranie k OPC UA serverom. Na účely overovania bolo vytvorených niekoľko administratívnych schránok *AAS-Temperature*, *AAS-Motor* atď. To umožňuje „biznis“ logike pracovať s pripojeným (*plugged*) zariadením a dať mu príkazy (*produce*).

Priemyselná komunikačná platforma OPC UA ICP obsahuje aj webové rozhranie na správu serverov založené na *ASP .NET CORE MVC*. Samotný server je implementovaný pomocou SDK od OPC Foundation. Local Discovery Server LDS-ME je implementovaný ako *SDK — open62541*.

Komunikačná platforma agregačného servera OPC UA pre opisovaný projekt je ilustrovaná na obrázku č. 201 a je postavená na výkonnom hardvérovom serveri ProLiant DL380.

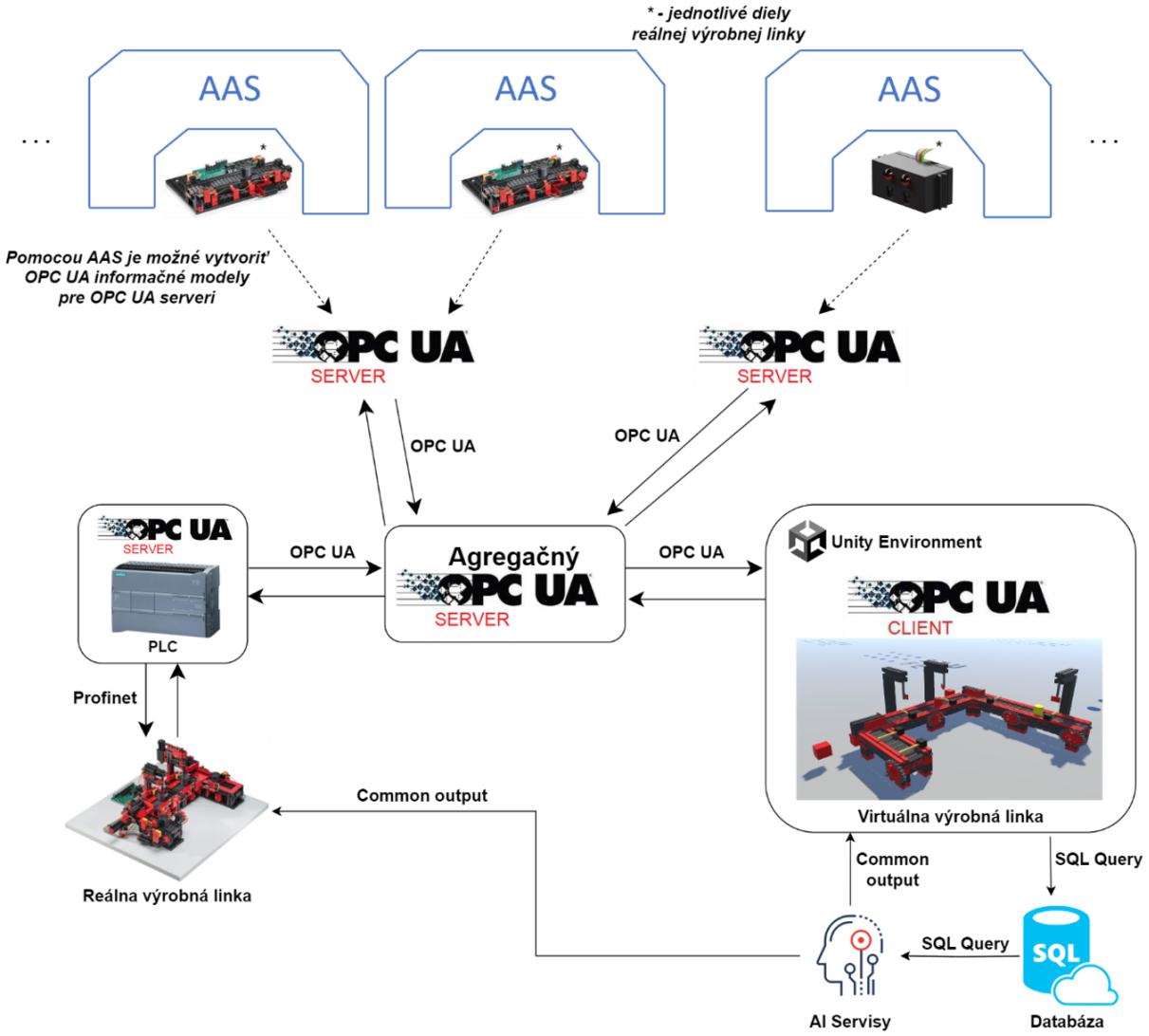
Táto EVP rozvíja štúdium OPC UA špecifikácií *Discovery*, *Aggregates* a *PubSub* s cieľom vytvoriť OPC UA sieť ako EVP platformu (s ambíciou Plug and Produce).



Obrázok č. 201: Komunikačná platforma agregačného servera OPC UA-ICP umožňuje širokú konektivitu

5.4.2 Edukačno-vývojová platforma pre Industry 4.0 komponenty

Druhá EVP predstavuje prvý vlastný **„Industry 4.0 komponent“** realizovaný ako interoperabilné digitálne dvojča pre fyzický model výrobnej linky Fischertechnik. Toto interoperabilné digitálne dvojča je „zapúzdrené“ v Asset Administrative Shell a využíva Unity engine [74]. Schému je možné vidieť na obrázku č. 202.



Obrázok č. 202: Interoperabilné digitálne dvojča a reálna linka tvoria „Industry 4.0 komponent"

Uvedené interoperabilné digitálne dvojča je vytvorené pomocou štandardu OPC UA a nástroja pre tvorbu AAS. Jednotlivé časti výrobnej linky boli vytvorené ako administratívne schránky (AAS), ktoré sú nosičom kompletného digitálneho opisu. Z týchto AAS boli následne vytvorené OPC UA informačné modely pre OPC UA servery.

Výhodou navrhnutej architektúry je implementácia agregačného OPC UA servera, ktorý zabezpečí jednoduchú dostupnosť ku všetkým OPC UA serverom z jedného prís-

tupového bodu. Hodnoty zo všetkých serverov sú navzájom zrkadlené, t. j. v prípade, ak dôjde k zmene hodnoty na príslušnom OPC UA serveri je hodnota automaticky prenesená na druhý.

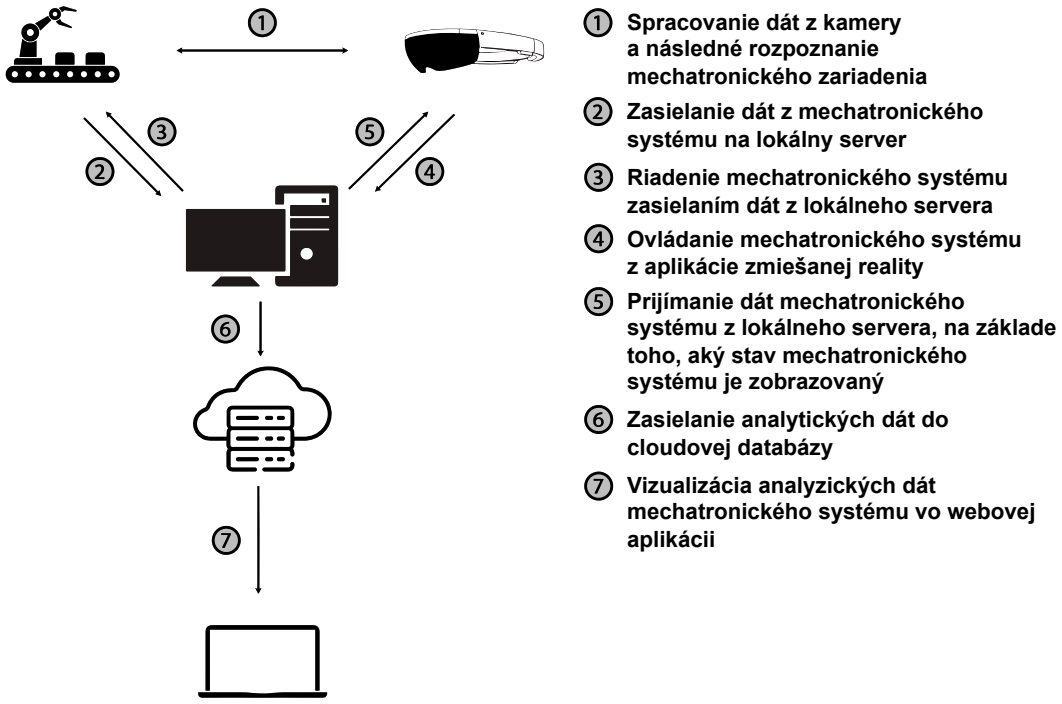
Industry 4.0 komponent je tvorený reálnou a virtuálnou entitou, ktorá je interoperabilná na úrovni AAS. V rámci reálnej entity je využitá linka Fischertechnik, ktorá je riadená PLC Siemens S7-1200. Na PLC je vytvorený OPC UA server, ktorý možno agregovať.

Virtuálna entita je vytvorená v Unity engine. Pri zostavení virtuálnej linky vieme využiť súbory nahraté v AAS, ako sú CAD modely pre jednotlivé časti linky či skripty, ktoré ich majú obsluhovať. Knižnica OPC UA klienta následne zabezpečí komunikáciu medzi agregačným OPC UA serverom a virtuálnou výrobnou linkou. Virtuálna entita je ďalej rozšírená o databázu zabezpečujúcu uchovanie dát z reálneho aj virtuálneho výrobného procesu a AI servisy, ktoré môžu tieto dáta ďalej analyzovať a ich výstup môže slúžiť na optimalizáciu výrobného procesu, prediktívnu údržbu, simuláciu *„what-if“* scenárov a pod.

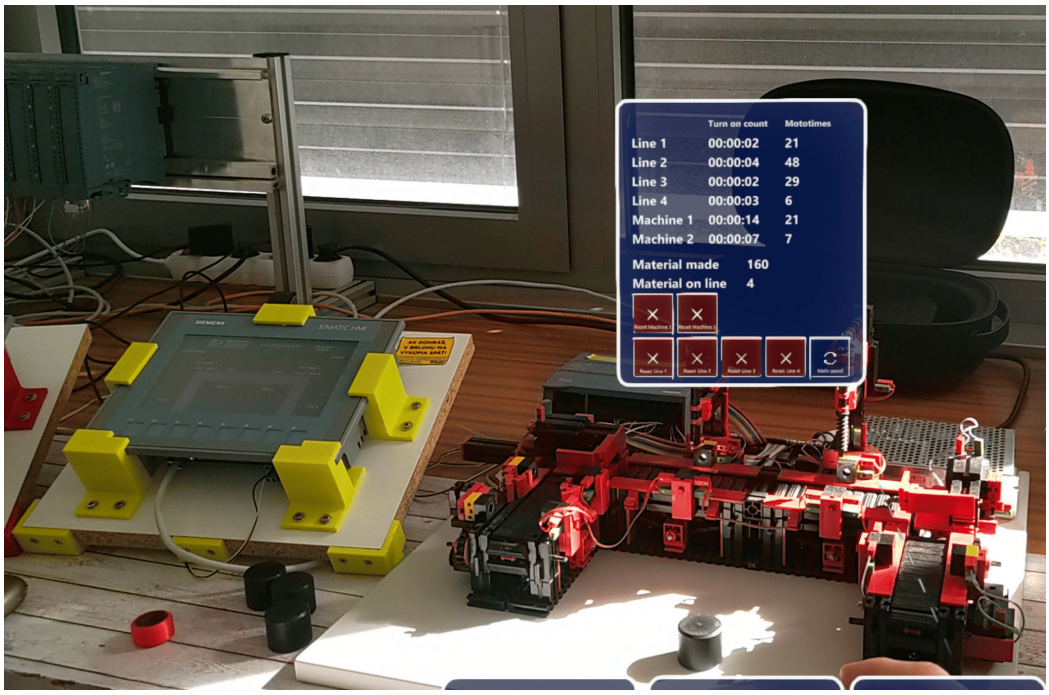
5.4.3 Edukačno-vývojová platforma pre virtuálnu a zmiešanú realitu

V rámci tejto časti bolo nadviazané na výsledky v oblasti moderných metód interakcie človek-stroj (moderné HMI) pomocou rozšírenej a zmiešanej reality v rámci kapitoly 5.2. Takisto v edukačnej prípadovej štúdii v kapitole 4.5 bolo prezentované riešenie pre rozšírenú realitu, ktorá bola realizovaná na tablete s OS Android. Monitorovaný a ovládaný bol model výrobnej linky Fischertechnik.

V rámci opisovaného projektu bolo toto riešenie ďalej modifikované a namiesto tabletu bol využitý moderný headset pre zmiešanú realitu Microsoft HoloLens 2. Po namierení headsetu na vybraný model linky (triediaca linka alebo výrobná linka s dvomi stanoviskami) sa tento model rozpozná metódami strojového videnia a výpočtovej inteligencie v prostredí zmiešanej reality podľa jeho vlastného fyzického tvaru. Po rozpoznaní sa aplikácia na headsete pripojí k danému modelu linky a je možné získať z nej dáta a riadiť ju. Na riadenie liniek sú využité PLC značky Siemens. Pre ukladanie dát je využitá cloudová databáza nasadená v Microsoft Azure. Schéma funkcionality riešenia je uvedená na obrázku č. 203. Na obrázku č. 204 je možné vidieť pohľad priamo z headsetu Microsoft HoloLens 2 na model výrobnej linky s informáciami v prostredí zmiešanej reality.



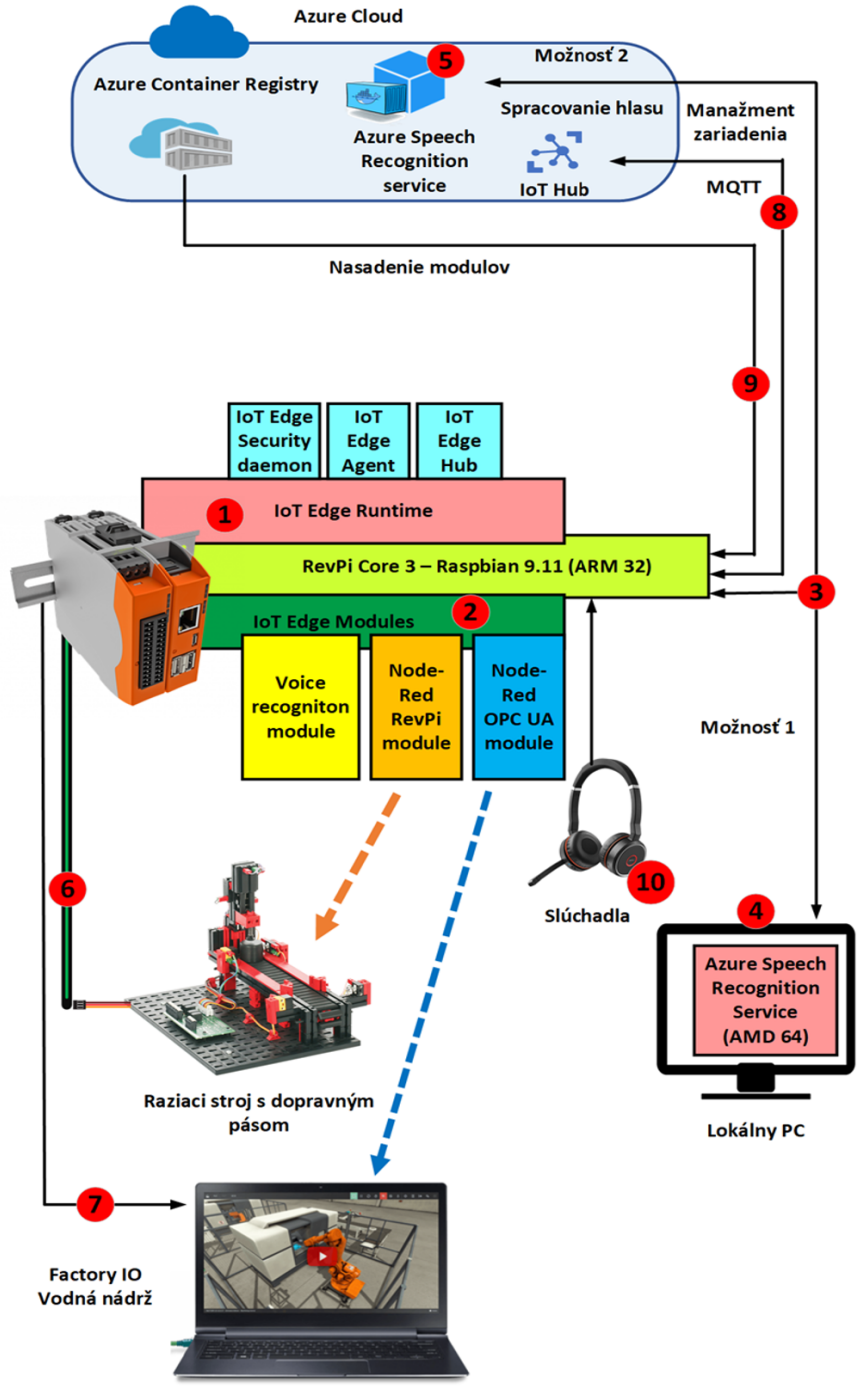
Obrázok č. 203: Schéma funkcionality edukačno-vývojovej platformy pre virtuálnu a zmiešanú realitu [100]



Obrázok č. 204: Pohľad na výrobnú linku v prostredí zmiešanej reality [100]

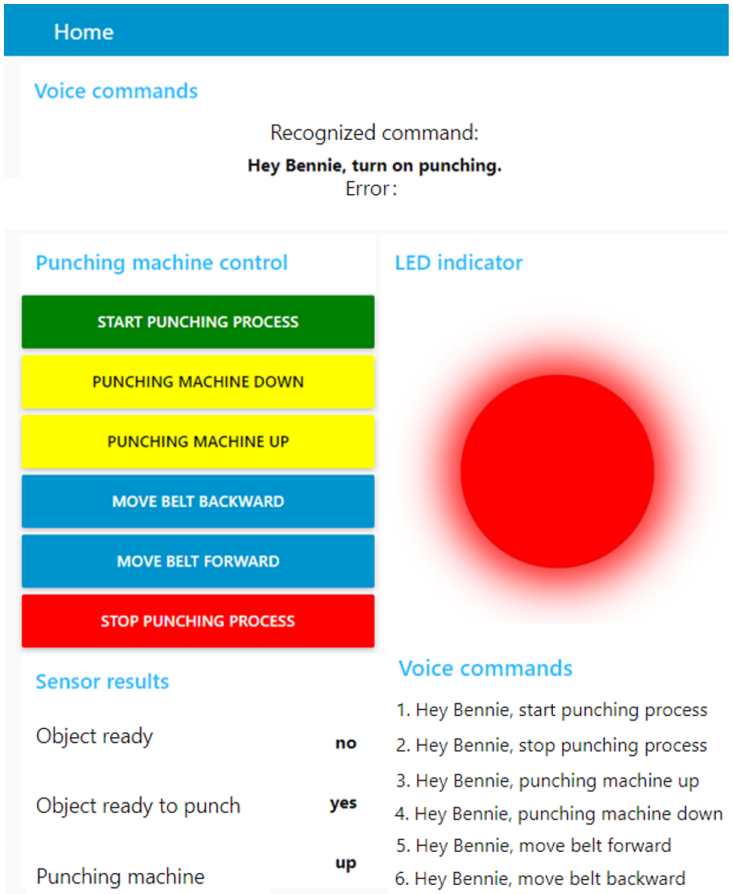
5.4.4 Edukačno-vývojová platforma pre orientáciu na služby v reálnom čase

Táto EVP je založená na ovládaní hlasom s využitím technológie edge computingu, cloud computingu a kontajnerizácie. Navrhnutá IoT architektúra hlasového ovládania (obrázok č. 205) bola implementovaná na reálnom priemyselnom PLC zariadení série Revolution Pi [5].



Obrázok č. 205: Architektúra originálneho návrhu ovládania akčného člena hlasom [4]

Súčasťou riešenia bol návrh vizualizácie formou aplikačného modulárneho systému s modernou formou rozhrania HMI využívajúceho hlasové povely operátora (obrázok č. 206), ktoré dopĺňa a zlepšuje konvenčné spôsoby ovládania mechatronických zariadení. Nový aplikačný modulárny systém prináša rýchlosť, jednoduchosť v správe a nasadzovaní vyvinutých aplikácií a je súčasťou konceptu moderného HMI.



Obrázok č. 206: Ovládací panel (HMI) s vizualizáciou hlasových povelov pre raziaci stroj s dopravným pásom [5]

Overenie implementovaného programového systému prebehlo na laboratórnom mechatronickom zariadení — modelovej výrobnej linke od spoločnosti Fischertechnik a tiež na virtuálnom modeli vodnej nádrži realizovanej v prostredí Factory I/O. Navrhnutý systém využíva umelú inteligenciu ako službu umiestnenú na lokálnom PC (edge) na obrázku č. 205 — objekt 4. Pre porovnanie bolo overené aj alternatívne rozpoznávanie hlasu v cloude (obrázok č. 205 — objekt 5), ktoré sa ukázalo ako menej vhodné pre väčšie latencie. Realizovaný systém kombinuje výhody hlasového ovládania ako používateľského rozhrania a edge computingu zameraného na bezpečnosť osobných dát, nízku latenciu, jednoduchú správu a nasadenie aplikácie pomocou kontajnerizácie (Docker).

6 Výhľad do budúcnosti — príchod konceptu Industry 5.0

Koncepty na princípe Industry 4.0 sa v poslednom desaťročí stali globálne akceptovanými, pričom mnohé krajiny zaviedli svoju vlastnú stratégiu a jej pomenovanie. Medzi takéto krajiny je možné zaradiť napríklad [94]:

- Austrália – Industry 4.0 Testlabs;
- Belgicko – Made Different;
- Francúzsko – Industrie de Futur;
- Japonsko – Society 5.0;
- Holandsko – Smart Industry;
- Čína – Made in China 2025;
- Spojené kráľovstvo Veľkej Británie a Severného Írska — The Future of Manufacturing;
- USA — Advanced Manufacturing Partnership;
- Južná Kórea — Manufacturing Industry Innovation 3.0;
- Slovensko — Smart Industry for Slovakia.

Výskum a vzdelávanie ovplyvnili vývoj a implementáciu riešení digitálnych a inteligentných technológií v rôznych odvetviach priemyslu a služieb. Vyvstávajú otázky, prečo už po približne 10 rokoch, v rámci ktorých sa hovorí o štvrtej priemyselnej revolúcii Industry 4.0, sa začína objavovať ďalší koncept — **Industry 5.0**. Európska komisia vydala oficiálny dokument o Industry 5.0 [22], v ktorom sa uvádza, že základný rozdiel vyplýva z toho, čím sú tieto revolúcie/koncepty *hnané* a čo je ich *hlavný cieľ* a *myšlienka*.

Koncept Industry 4.0 je hnaný **technológiami** (angl. *technology-driven*), avšak koncept Industry 5.0 je hnaný **hodnotami** (angl. *value-driven*) [94].

Revolúciu Industry 5.0 je teda možné chápať aj ako evolúciu Industry 4.0, pri ktorej by bolo možné povedať, že sa ako keby zabudlo na človeka a myslelo sa len na modernizáciu výrob, vyššiu efektivitu a ekonomický profit. Industry 5.0 tento pohľad otáča **smerom na človeka**, jeho komfort, zdravie a prostredie, v ktorom pracuje a žije. Nie je náhoda, že koncepty Industry 4.0 a 5.0 aktuálne kooexistujú a nejakú dobu ešte koexistovať pravdepodobne budú, nakoľko sa jeden druhý veľmi dobre dopĺňajú a vzájomne

sa vyvažujú. Vedú sa debaty o tom, či ide v prípade Industry 5.0 o revolúciu alebo len evolúciu. Odpoveď stále nie je jasná a záleží na uhle pohľadu.

6.1 Pozadie vzniku konceptu Industry 5.0

Celý svet už dlhodobo čelí zmene klímy a z toho vyplývajúcej hrozbe kolapsu biodiverzity. Tieto zmeny môžu spustiť reťazovú reakciu s veľkými následkami v oblasti politiky, demografie, priemyslu a služieb. Globalizácia s cieľom maximalizácie ziskov ukázala svoju nízku odolnosť voči takýmto otrasom, čo sa ukázalo aj počas pandémie ochorenia COVID-19, kedy najmä v úvodných fázach došlo k narušeniu dodávateľsko-odberateľských reťazcov a stability ekonomického systému. Svet čelí celému radu výziev prevažne ekologického charakter — jedna z nich je napríklad **prechod od lineárneho ekonomického modelu** (princíp vziať — vyrobiť — zlikvidovať) k **obehovému (cyklickému) modelu**. Tento model sa snaží čo najviac maximalizovať hodnotu výrobkov a materiálov tým spôsobom, že ich udržiava v obehu. V tomto smere je možné spomenúť napríklad posledné snahy Európskej únie o jednoduchšiu opraviteľnosť spotrebných výrobkov a podobne.

Koncept Industry 4.0 je typickým predstaviteľom lineárneho modelu, nakoľko vychádza z paradigmy orientovanej na ekonomický rast bez ohľadu na jeho energetickú náročnosť a bez ohľadu na využívanie zdrojov. Industry 4.0 je najmä technologická paradigma založená na vzniku kyberneticko-fyzikálnych systémov, konvergencii informačných a operačných technológií. Táto paradigma však nerieši sociálne napätie a dosah klimatickej krízy.

Obrázok č. 207: 17 cieľov udržateľného rozvoja podľa Organizácie spojených národov [72]

Industry 4.0 na zvládnutie nových výziev chýbajú najmä tieto kľúčové aspekty [22]:

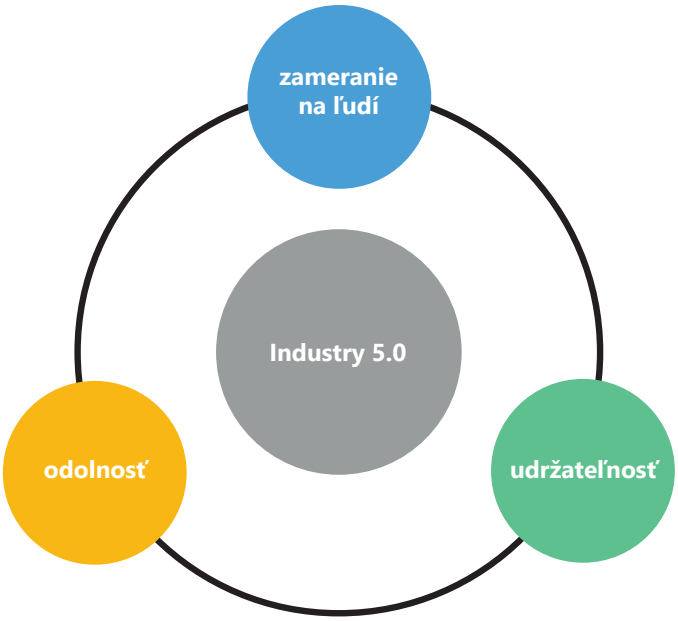
- **regeneračný rozmer** — ide o to, aby sa obehové hospodárstvo a pozitívne regeneračné spätné väzby nepovažovali za vedľajšie aspekty, ale za kľúčové piliere návrhu celých priemyselných reťazcov;
- **sociálny rozmer** — vyžadovanie pozornosti na psychickú pohodu pracovníkov, ich sociálne začlenenie, prijatie technológií a technologických riešení, ktoré nenahrádzajú, ale skôr dopĺňajú ľudské schopnosti, ak je zo možné (v istom zmysle je možné tento rozmer chápať aj ako príklon k videniu Tomáša Baťu);
- **environmentálny rozmer** — podpora transformácie eliminujúcej používanie fosílnych palív a energetickej účinnosti, zachytávanie uhlíka, využívanie riešení založených na prírode a obnova biodiverzity, návrhy opatrení na systematické zhodnocovanie odpadov a minimalizáciu znečistenia, maximalizácia udržiavania výrobkov a materiálov v produktívnom používaní a obehu.

Z vyššie uvedeného vyplýva, že koncept Industry 5.0 ani tak nepredstavuje technologický skok vpred, ale z istého uhla pohľadu vkladá koncept Industry 4.0 do širšieho kontextu, čím stáča smerovanie technologickej transformácie priemyselnej výroby k prvkom obnovy a v prospech **ľudí** a životného prostredia. Zámerom je prepojenie digitálnej transformácie s **udržateľnosťou** životného prostredia a jeho zdrojmi. Industry 5.0 prináša odklon od modelov zameraných na výrobu pre zisk smerom k vyváženejšiemu pohľadu na hodnotu v čase a k viachodnotovému chápaniu kapitálu (ľudského, prírodného a finančného). V prospech tohto konceptu hrá aj fakt jeho zvýšenej **odolnosti** vzhľadom na zabezpečenie dodávok produktov a služieb počas neočakávaných kríz (ako sa ukázala napríklad pandémia ochorenia COVID-19). Spoločnosti uplatňujúce kritériá sociálneho a environmentálneho riadenia ukázali väčšiu odolnosť vo vzťahu k dodávkam.

6.2 Paradigmy konceptu Industry 5.0

Ako už bolo uvedené, koncept Industry 5.0 má svoje korene v Industry 4.0, avšak pôvodný koncept Industry 4.0 sa menej zameriaval na princípy sociálnej spravodlivosti a udržateľnosti, ale viac na digitalizáciu s cieľom zvýšenia efektívnosti a flexibility výroby. Koncept Industry 5.0 poskytuje odlišné zameranie, zdôrazňuje dôležitosť výskumu a inovácií na podporu priemyslu v jeho dlhodobej službe ľudstvu a rešpektuje hranice našej planéty [22].

Vzhľadom na vyššie uvedené východiská Európska komisia uviedla **tri kľúčové elementy (paradigmy)** konceptu **Industry 5.0** (obrázok č. 208) [22]:



Obrázok č. 208: Paradigmy Industry 5.0

- **Udržateľnosť** (angl. *sustainability*) — Koncept dbá na využívanie nových technológií na poskytnutie prosperity nielen s cieľom zefektívniť pracovnú silu a rastu spoločnosti, ale dbá aj na limity planéty. V tomto kontexte sa priemyselné procesy musia vyvíjať tak, aby minimalizovali negatívny dopad na životné prostredie. Toto zahŕňa využívanie obnoviteľných zdrojov energie, znižovanie emisií, recykláciu a opätovné používanie materiálov.

- **Odolnosť** (angl. *resilience*) — Revízia existujúcich hodnotových reťazcov môže zvýšiť odolnosť priemyslu voči vonkajším vplyvom a neočakávaným krízam. Cieľom je vybudovať flexibilný a prispôsobivý priemyselný systém, ktorý dokáže udržať kontinuitu výroby a služieb aj v nestabilných alebo nepredvídateľných situáciách. Odolné podniky tiež často zdôrazňujú dôležitosť rozmanitosti zdrojov, decentralizácie výroby a vytvárania lokálnych dodávateľských reťazcov, čím sa znižuje závislosť od jednotlivých trhov alebo zdrojov. To zahŕňa digitalizáciu a automatizáciu procesov, ako aj vytváranie decentralizovaných výrobných sietí, ktoré dokážu rýchlo reagovať na zmeny a minimalizovať riziká spojené s centralizovanými výrobnými modelmi.

- **Zameranie na človeka** (angl. *human-centricity*) — Paradigma spolupráce človeka a stroja v Industry 5.0 sa zameriava na vytvorenie synergického pracovného prostredia, kde ľudia a inteligentné stroje (ako roboty, systémy umelej inteligencie, automatizované systémy atď.) spolupracujú. To zahŕňa vytváranie pracovných miest,

ktoré lepšie využívajú ľudské schopnosti a kreativitu a zároveň delegujú robotom a automatizovaným systémom úlohy, ktoré sú repetitívne alebo nebezpečné. Takýto prístup umožňuje firmám využívať výhody technológie, zatiaľ čo udržiavajú ľudskú tvorivosť a inovácie ako kľúčové prvky svojich procesov. Zameranie na človeka v Industry 5.0 kladie veľký dôraz na vývoj pokročilých rozhraní človek-stroj (HMI), ktoré sú kľúčové pre efektívnu a intuitívnu spoluprácu medzi ľuďmi a strojmi. Tu má úlohu napríklad technológia headsetov (náhlavných súprav) pre zmiešanú realitu, kedy dokáže pracovníkov monitorovať a diagnostikovať strojné zariadenia, pričom ruky pracovníka popritom ostávajú voľné, čím sa zvyšuje *ergonómia* pracovných úloh. Ďalšími možnosťami sú napríklad hlasové ovládanie, ovládania gestami alebo v budúcnosti rozhranie mozog-počítač. Koncept berie na vedomie fyzickú a psychickú pohodu pracovníka a využíva nové technológie aj pre *inklúziu* znevýhodnených skupín ľudského spoločenstva do pracovného procesu.

6.3 Technológie a aplikácie v rámci konceptu Industry 5.0

Technológie a aplikácie v rámci konceptu Industry 5.0 sa zameriavajú na naplnenie požiadaviek uvedených v predošlom texte.

Zameranie na *potreby ľudí a vhodné interakcie medzi ľudmi a strojmi* by sa malo dosahovať technológiami, ktoré prepájajú a kombinujú silné stránky strojov a ľudí. Medzi takéto technológie aplikácie môžeme zaradiť **rozpoznávanie hlasu** (aj viacjazyčné), **rozpoznávanie gest**, **zmiešanú realitu**, **kolaboratívne roboty** alebo **exoskeletony**.

Komunikácia človeka s počítačom a inými zariadeniami či strojmi si prešla od vzniku prvých digitálnych systémov rozsiahlym vývojom. Od myši a klávesnice si spoločnosť postupne zvykla na ovládanie pomocou dotykových displejov. V dnešnej dobe sa črtá jasný trend, ktorý zabezpečí odbremenenie používateľa od zbytočnej námahy a umožní mu, aby sa rozhraním stal on sám (human-centric prístup). Znamená to, že interakcia medzi človekom, počítačom alebo strojom bude prebiehať len pomocou hlasu, pohybmi, gestami alebo dokonca s využitím snímania mozgových signálov.

Ovládanie zariadení hlasom sa ukazuje ako jeden z prvkov v rozvoji human-centric prístupu k HMI, pretože umožňuje prirodzenú a intuitívnu interakciu a znižuje fyzickú námahu. Hlasové ovládanie umožňuje pracovníkom komunikovať so strojmi a systémami prirodzeným spôsobom, podobne ako v každodennej komunikácii medzi ľuďmi [8]. Tento prístup minimalizuje potrebu učenia sa zložitých príkazov alebo navigácie komplexnými menu, čím sa znižuje čas potrebný na tréning a zvyšuje sa prirodzená interakcia s technológiou. Napriek mnohým výhodám je implementácia hlasového ovládania zariadení v priemyselnom prostredí spojená aj s určitými výzvami. Patria medzi ne zabezpečenie presnosti a spoľahlivosti rozpoznávania hlasu, zvládnutie rôznych

dialektov a jazykov a ochrana proti hluku a rôznych ruchom vo výrobných závodoch. Riešenie týchto výziev si vyžaduje pokročilé technológie v oblasti umelej inteligencie a strojového učenia, ktoré sú schopné učiť sa a prispôsobovať sa špecifickým potrebám a prostrediu každého výrobného závodu.

Ďalším prostriedkom pre interakciu medzi človekom a strojom môžu byť **gestá**. Gestá sú expresívne, zmysluplné pohyby tela zahŕňajúce fyzické pohyby prstov, rúk, paží, hlavy, tváre alebo tela so zámerom sprostredkovať informácie alebo interakciu s prostredím [8]. Rozpoznávanie ľudských gest prevzalo dôležitú úlohu v priemyselných aplikáciách. Na detekciu a rozpoznávanie gest je možné využívať metódy strojového videnia (v prípade využitia kamerových systémov) a/alebo metódy výpočtovej inteligencie v prípade využitie **oblekov na snímanie pohybu** (angl. motion-capture oblek). Obleky na snímanie pohybu prinášajú výhodu v tom smere, že ich je možné integrovať pod pracovné oblečenie človeka a nie je nutné neustále snímať tohto pracovníka kamerou. Ich využitie je tiež možné v prípadoch, kde by snímanie prostredia kamerou bolo ťažké alebo nemožné (zadymené prostredie, vesmír a podobne).

Obrázok č. 209: Ilustrácia využitia zmiešanej reality v priemysle (*generované umelou inteligenciou s pomocou modelu DALL-E*)

Inovatívnym prostriedkom HMI môžu tiež byť **zariadenia zmiešanej reality**. Zmiešaná realita na rozdiel od virtuálnej reality neprekrýva reálny svet čisto virtuálnym. Zmiešaná realita prekrýva (dopĺňa) reálny svet digitálnymi (počítačom generovanými) prvkami, ako sú virtuálne objekty, animácie, texty, dáta alebo zvuky. V prípade výrob-

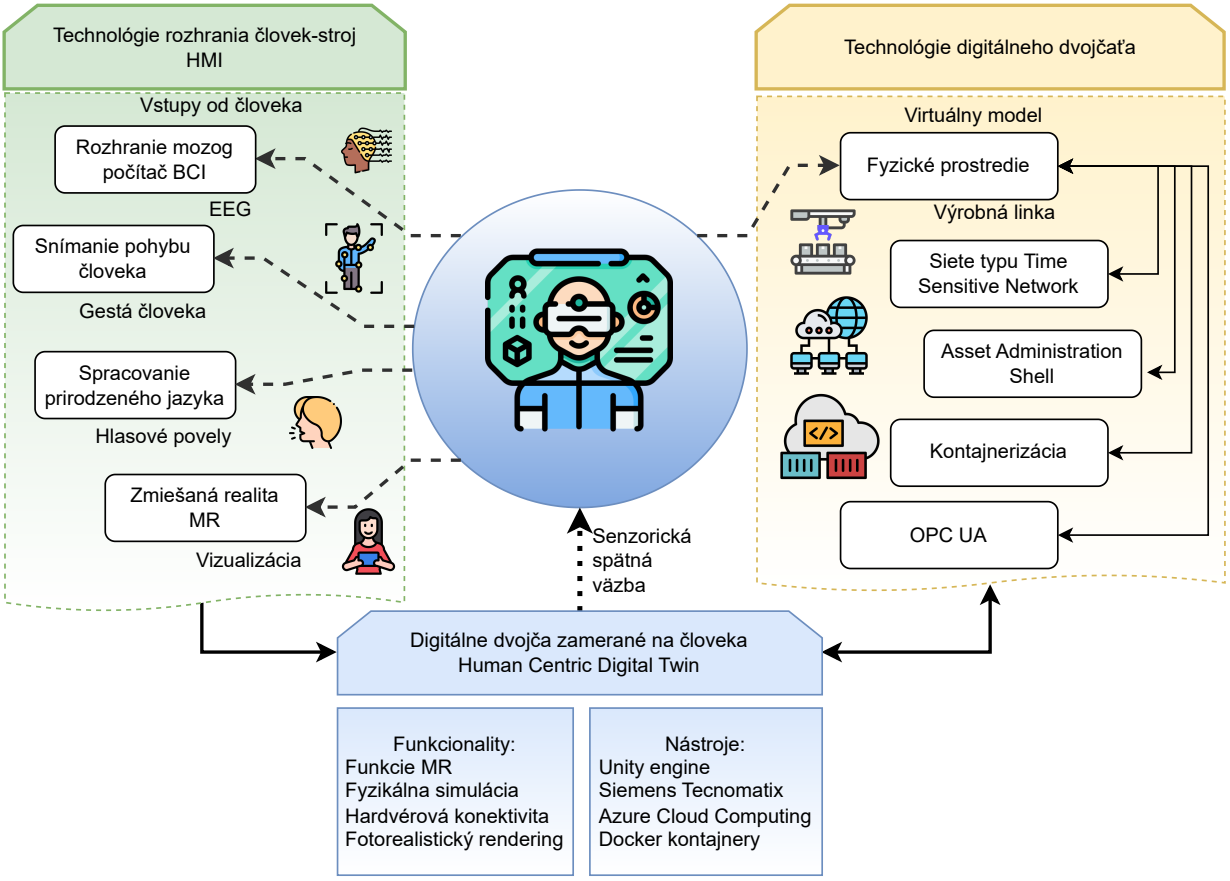
ných podnikov môže ísť o diagnostické dáta z jednotlivých strojov alebo aj ovládacie prvky týchto strojov. Reálny svet a digitálne informácie sú synchronizované vďaka geolokalizácii alebo zabudovaným senzorom (akcelerometer, gyroskop), ktoré lokalizujú používateľa vo vzťahu k jeho prostrediu a prispôsobujú displej jeho pohybom [8]. Už v súčasnosti existujú headsety pre zmiešanú realitu, ktoré majú oproti tabletom pre podporu zmiešanej reality v úlohách HMI zásadnú výhodu. Pri využití headsetu (na rozdiel od tabletu) ostávajú ruky operátora voľné, čo prináša väčšiu ergonómiu pri využívaní a tiež bezpečnosť.

Obrázok č. 210: Open-source BCI zariadenie Ultracortex "Mark IV" EEG [71]

Je možné očakávať, že v priebehu ďalších rokov bude hrať úlohu v aplikáciách Industry 5.0 aj **rozhranie mozog-počítač** (*angl. brain-computer interface — BCI*). BCI predstavuje vysoko pokročilú technológiu, ktorá umožňuje priamu komunikáciu medzi ľudským mozgom a externými zariadeniami. V kontexte Industry 5.0 môže toto rozhranie zohrať kľúčovú úlohu nielen v zvyšovaní efektivity a produktivity, ale tiež v inklúzii a podpore pracovníkov s rôznymi schopnosťami. BCI prináša novú úroveň interakcie, kde pracovníci môžu ovládať stroje, roboty a iné mechatronické systémy priamo svojimi myšlienkami. Jednou z najvýznamnejších výhod BCI je potenciál pre inklúziu ľudí so špeciálnymi potrebami. Technológia BCI otvára dvere pre pracovníkov, ktorí by inak boli obmedzení svojimi fyzickými schopnosťami. Napriek veľkému budúcemu potenciálu sa používanie BCI v priemysle stretáva aj s technickými a etickými výzvami. Pres-

nosť, spoľahlivosť a odozva systémov BCI sú kľúčové pre bezpečnú a efektívnu prácu, čo vyžaduje pokročilé algoritmy a ďalšie technologické inovácie v tomto odvetví. Z etického hľadiska je potrebné zabezpečiť ochranu súkromia a integrity údajov, zvlášť pokiaľ ide o zaznamenávanie a spracovanie signálov z ľudského mozgu.

V publikácii [82] z roku 2016 bol zadefinovaný pojem **Operátor 4.0**, ktorý zahŕňa nové typy interakcie medzi človekom a strojmi meniace pracovnú silu a majúce výrazný dosah na charakter práce. Spomína sa tiež pojem **ľudské kyberneticko-fyzikálne systémy** (angl. *Human-Cyber-Physical Systems — H-CPS*). V súlade s konceptom Industry 5.0 je uvedené, že by mali byť navrhnuté tak, aby nenahrádzali zručnosti a schopnosti ľudí, ale skôr aby s ľuďmi koexistovali a pomáhali byť efektívnejšími. Koncept Operátor 4.0 teda obohacuje Industry 4.0 o princípy a technológie, ktoré majú pomôcť so začlenením človeka do meniaceho a modernizujúceho sa prostredia. Z tohto pohľadu je možné povedať, že tento technologický cieľ a jeden z troch elementov Industry 5.0 nie je revolúciou, ale skôr evolúciou a začlenením už existujúcich konceptov priamo do konceptu Industry 5.0 Európskej komisie [35].



Obrázok č. 211: Príklad funkcionalít digitálneho dvojčaťa zameraného na človeka

Existuje aj pojem **digitálneho dvojčaťa zameraného na človeka** (angl. human-centric digital twin — HCDT) [1]. Digitálne dvojčatá poskytujú kontrolu a dohľad nad celým

systémom a prinášajú nám možnosti plánovanej údržby, simulácie a takisto možnosti merania vplyvu na životné prostredie (zložka *udržateľnosti* konceptu Industry 5.0). Digitálne dvojčatá tiež vieme doplniť *human-centric* zložkou vo forme prístupu k fyzickým zariadeniam a ich digitálnym náprotivkom (dvojčatám) pomocou moderných HMI, ako je napríklad zmiešaná realita. Synergiou týchto prístupov a technológií vzniká spomínané HCDT.

Obrázok č. 212: Ilustrácia spolupráce ľudského operátora s kolaboratívnym robotom (*generované umelou inteligenciou s pomocou modelu DALL-E*)

V aplikáciách Industry 5.0 by mala byť snaha upraviť prostredie tak, aby roboty pomáhali ľudským operátorom. Takéto roboty sú známe aj ako **kolaboratívne roboty** alebo ako **koboty**. Ako už bolo uvádzané, stroje teda plne nenahrádzajú ľudí, ale dopĺňajú ich schopnosti a odbremeňujú ich od namáhavých alebo repetetívnych úloh. Je možné si napríklad predstaviť sklad, kde operátori prikazujú gestom ruky robotom, aby podali nástroj alebo zdvihli škatuľu. Synergia ľudskej inteligencie a sily robotov uľahčí život [8].

S human-centric prístupom súvisia aj **exoskeletony**, ktoré už v súčasnej dobe nachádzajú uplatnenie najmä v Ázii z dôvodu starnutia obyvateľstva. Ich vytvorenie umožnil technologický pokrok oblasti v akčných členov, senzorov, materiáloch a procesorov. Bežným cieľom exoskeletonu je poskytnúť nadľudskú silu alebo vytrvalosť. Exoskeletony však môžu pomôcť aj pacientom s neurologickým postihnutím zlepšiť ich motorický výkon [8].

Ďalším zaujímavým odvetvím sú **bio-inšpirované technológie**. **Biomimetika**, nová veda, sa inšpiruje prírodou na vytváranie materiálov a technológií, ktoré napodobňujú biologické procesy od makro do nano úrovne. Táto oblasť využíva komplexné vzťahy medzi štruktúrou a vlastnosťami prirodzených materiálov na vývoj inovatívnych nanomateriálov a zariadení. Príklady z prírody ako superhydrofóbnosť, samočistenie, energetická efektivita, zníženie odporu pri prúdení tekutín, premena a šetrenie energie, vysoká pevnosť a senzorické schopnosti, poskytujú modely pre tvorbu pokročilých komerčne využiteľných technológií. Príkladom bio-inšpirovaných technológií môžu byť pneumatické svaly, ktoré sa často používajú v robotike [7].

Biosenzor je zariadenie kombinujúce biologický komponent a elektroniku na premenu biochemických signálov na elektrické, ktoré sú potom ľahko merateľné. Funguje tak, že biologická časť reaguje so špecifickou látkou, vyvoláva zmenu, ktorá sa elektronicky zosilní a kvantifikuje. S ich širokým využitím v zdravotníctve, monitoringu prostredia a priemysle, biosenzory predstavujú kľúčový pokrok v biotechnológii. Hlavné výzvy spočívajú v presnom zachytávaní a premenení biosignálov a v miniaturizácii technológií. Ich prednosti zahŕňajú rýchlosť, nákladovú efektivitu, špecifickosť a spoľahlivosť [7].

Inteligentné materiály neoznačujú materiály vytvorené s podporou strojového učenia, ale skôr ide o také materiály, ktoré majú schopnosti ako **samooprava** a **regenerácia**. Tieto pokročilé materiály reagujú na rozličné podnety, vrátane elektrických, magnetických, teplotných a ďalších fyzikálnych faktorov, a dokážu sa vrátiť do svojho pôvodného stavu po odstránení týchto podnetov. Vďaka vývoju vo materiálových vedách, inteligentné materiály ponúkajú špecifické funkcie pre pokročilé aplikácie, čo predstavuje významný pokrok oproti tradičným materiálom ako polyméry, kovy či sklo. Tieto materiály fungujú na nízkej úrovni, ale môžu byť integrované do komplexných technických systémov, pridávajúc im nové vlastnosti a funkcie. Inteligentné materiály nachádzajú uplatnenie v aplikáciách ako prevodníky, snímače a konštrukčné prvky, čím stimulujú rast ich trhu. Široké využitie v rôznych priemyselných sektoroch a vplyv rozvoja IoT zvyšujú dopyt po týchto materiáloch, najmä v rozvíjajúcich sa ekonomikách. Hoci vývoj inteligentných materiálov vyžaduje značné investície, ich výroba je často ekonomickejšia vďaka úspore materiálov a možnosti nahradenia viacerých tradičných technológií jedným materiálom. Tento prístup smeruje k cieľom Industry 5.0, akými sú *udržateľnosť* a efektívnejší rozvoj, a to všetko pri využití dostupných surovín [7].

V súčasnosti, keď je **recyklácia** nevyhnutná, sa Industry 5.0 sústredí na inovácie v tejto oblasti. S rastúcim nedostatkom surovín, európska recyklačná miera rastie a recyklácia a opätovné použitie materiálov sa stáva kľúčové. Nové technológie umožňujú efektívnejšie zhodnocovanie surovín, vrátane recyklácie kovov z komplexných vyrade-

ných produktov. Rozvoj produktov navrhnutých pre ľahkú separáciu, opätovné použitie a recykláciu znižuje využívanie skládok, spotrebu energie a emisie skleníkových plynov a zároveň znižuje dopyt po nových surovinách, nakoľko recyklovaný materiál je často cenovo výhodnejšou alternatívou. Inteligentné materiály, ktoré sú jednoducho recyklovateľné alebo môžu byť využité v iných procesoch, prispievajú k tomuto udržateľnému trendu [7].

Obrázok č. 213: Proces tvorby domu pomocou aditívnej výroby — špecializovanej 3D tlače [19]

Udržateľnosť a ekologická zodpovednosť sú kľúčové aspekty Industry 5.0, ktoré sa zameriavajú na vytvorenie viac ekologicky udržateľných a efektívnych výrobných procesov. Tento prístup sa snaží nielen minimalizovať negatívny dopad na životné prostredie, ale aj optimalizovať využitie zdrojov a energie. V tejto súvislosti má významnú úlohu aj **aditívna výroba** vo forme **3D tlače**, ktorá prispieva k udržateľnosti v niekoľkých kľúčových oblastiach. 3D tlač umožňuje vytvárať objekty vrstva po vrstve, čo znamená, že materiál sa pridáva iba tam, kde je to potrebné. Tento proces je v porovnaní s tradičnými výrobnými metódami, ktoré často vyžadujú odstránenie materiálu a tým produkujú odpad, oveľa efektívnejší. Využitím aditívnej výroby môžu firmy výrazne *zredukovať množstvo odpadu* a zvýšiť efektivitu využívania materiálov. 3D tlač umožňuje výrobu dielov priamo na mieste, čo znižuje potrebu prepravy. Lokalizovaná výroba nielenže znižuje emisie spojené s prepravou, ale tiež zvyšuje rýchlosť a flexibilitu v procese výroby. Firmy môžu rýchlo reagovať na zmeny v dopyte a potrebách zákazníkov bez dlhých dodacích lehôt a komplexných logistických reťazcov. V kombinácii s inými strategickými inováciami, ako sú napríklad recyklácia materiálov využívaných 3D tlačiarňami, využívanie obnoviteľných zdrojov energie a inteligentné riade-

nie výrobných procesov, 3D tlač tak predstavuje jeden z nástrojov pre dosiahnutie cieľov v oblasti *udržateľnosti* a *odolnosti*.

V rámci Industry 5.0 sa **umelej inteligencii** priznáva stále väčší význam, keďže sa stáva kľúčovým prvkom v podpore human-centric prístupu, udržateľnosti a odolnosti priemyselných systémov. Umelá inteligencia nie je len nástrojom na automatizáciu a zefektívnenie výrobných procesov, ale predstavuje aj platformu pre inovácie, ktoré sú v súlade s potrebami a hodnotami ľudí, ako aj s environmentálnymi a ekonomickými výzvami súčasnosti. Ako už bolo uvádzané v predošlom texte, umelá inteligencia prináša revolúciu v interakcii medzi ľuďmi a strojmi. Umelá inteligencia môže zlepšiť pochopenie a adaptabilitu technológií voči *ľudským potrebám*. Tým umožňuje vytvorenie pracovného prostredia, kde sa ľudia môžu sústrediť na kreatívne a strategické úlohy, zatiaľ čo repetitívne a náročné úlohy sú ponechané na stroje. Tento prístup nielenže zvyšuje produktivitu, ale aj zlepšuje spokojnosť a pohodu pracovníkov. Umelá inteligencia je tiež kľúčovým nástrojom v boji proti klimatickým zmenám a pri presadzovaní *udržateľnosti* v priemysle. Systémy založené na umelej inteligencii môžu optimalizovať spotrebu zdrojov, minimalizovať odpad a maximalizovať efektivitu výrobných procesov. Napríklad, algoritmy pre prediktívnu údržbu môžu predvídať poruchy strojov a zariadení ešte pred ich výskytom, čím sa znižuje množstvo odpadu a zvyšuje sa životnosť strojov. Umelá inteligencia tiež umožňuje inteligentné riadenie energetických zdrojov a optimalizáciu logistických a distribučných reťazcov, čím znižuje emisie a zvyšuje energetickú efektivitu. Umelá inteligencia nachádza svoje miesto aj pre koncept odolnosti. V dnešnom rýchlo sa meniacom svete sa *odolnosť* stala kľúčovým faktorom pre úspech podnikov. Umelá inteligencia umožňuje podnikom rýchlejšie reagovať na zmeny v trhových podmienkach, predvídať budúce výzvy a adekvátne sa na ne pripraviť. Systémy založené na umelej inteligencii dokážu analyzovať veľké množstvá dát, identifikovať trendy a vzory, ktoré by mohli ľudia prehliadnuť. Táto schopnosť umožňuje podnikom adaptovať svoje stratégie, procesy a produkty tak, aby boli odolnejšie voči vonkajším vplyvom, ako sú ekonomické krízy, prírodné katastrofy alebo pandémie.

Tak ako bola pred desaťročiami hybnou silou ekonomiky ropa, dnes sú touto hybnou silou dáta — aplikácie súvisiace s paradigmou veľkých dát — angl. **Big Data**. Preto sú Big Data dôležitým konceptom aj pre Industry 5.0. Spracovanie dát pri koncepte Industry 4.0 bolo zamerané na technológie (*technology-driven*) a zvyšovanie zisku, pri Industry 5.0 sú to hodnoty (*value-driven*) a snaha ho udržateľný rozvoj spoločnosti, odolnosť, zamerania na človeka/zamestnanca. Big Data otvárajú nové príležitosti pre odvetvia zamerané na udržateľnosť a ziskovosť. Zber a analýza dát o ekonomických ukazovateľoch, výrobných objemoch a emisiách umožňujú efektívnejšiu produkciu a poskytujú hĺbkové poznatky pre zvýšenie výnosov. Interaktívne a intuitívne dátové rozhrania výrazne

rozširujú množstvo informácií dostupných pre riadenie operácií, nákladov a dodávateľských reťazcov. Kľúčom je správne navrhnutie algoritmov a programovanie na odhalenie dôležitých vzťahov medzi dátami. Napríklad v papierenskom priemysle pomáhajú veľké dáta optimalizovať emisie a náklady výroby, umožňujú monitorovanie emisií v reálnom čase a zvyšujú transparentnosť a dôveru zákazníkov v udržateľnosť výroby [75].

Vyššie uvedené technológie a aplikácie poskytujú pohľad na širokú paletu inovatívnych riešení, ktoré sú kľúčom k adaptívnej, efektívnej budúcnosti, kde hlavným cieľom je spokojnosť jednotlivca v prosperujúcej spoločnosti. Veríme, že osveta v oblasti moderných digitálnych a inteligentných technológií prináša základ pre ďalší rast a inšpiráciu na ceste za vytváraním inteligentnejších, inkluzívnejších a udržateľnejších priemyselných systémov.

Záver

Priemysel je považovaný za významnú hospodársku aktivitu človeka. Tak, ako sa vyvíjala spoločnosť v priebehu histórie, rozvíjala sa aj priemyselná výroba. Dynamika priemyselného rozvoja, cieľavedomé uplatňovanie vedeckých poznatkov a technologického pokroku vždy určovala aj dynamiku a stupeň spoločenského rozvoja. Dôležitým prahom technologického rozvoja v priemysle je štvrtá priemyselná revolúcia — koncept Industry 4.0. Nositeľom zmien je v tomto prípade najmä digitalizácia, automatizácia, mechatronizácia a integrácia moderných informačno-komunikačných technológií na všetkých úrovniach riadenia podnikových procesov a služieb.

V tradičnom ponímaní sa informačné alebo digitálne technológie vyvíjali a využívali oddelene od priemyselných operačných technológií. Tieto technologické smery neboli v minulosti účelovo vyvíjané na vzájomnú spoluprácu a ani na spoluprácu s inými systémami. Štvrtá priemyselná revolúcia Industry 4.0 je preto dominantne založená na rozvoji a integrácii nových progresívnych digitálnych technológií (výpočtová inteligencia, Big Data, kontajnerizácia, cloud a edge computing, virtuálna, rozšírená a zmiešaná realita, koncept Internet of Things atď.), ktoré boli ešte nedávno súčasťou výhradne sektora služieb a entertainmentu, do priemyselných aplikácií. Trend prepájania uvedených dvoch svetov je v súčasnosti nesporný a v praxi žiadaný. Vzdelávací systém začal len postupne zohľadňovať požiadavku konvergencie informačno-komunikačných technológií, automatizácie a mechatroniky. Apel na väčšiu dynamiku v oblasti modernizácie študijných osnov v súlade s modernými trendmi predstavuje aj predložená učebnica.

Prienik digitálnych alebo inteligentných technológií do procesov v Industry 4.0 je v predloženej učebnici zameraný na aplikácie monitorovania a riadenia diskrétnych udalostných systémov. Využívané sú technológie a koncepty ako Internet of Things, počítačom generovaná realita, cloud computing a webové a mobilné aplikácie. Monitorovania a riadenie je podporované inovatívnymi prostriedkami, ako sú komunikačný štandard OPC UA, MQTT alebo middleware Node-RED. Predložené edukačné prípadové štúdie využívajú otvorené a ľahko dostupné softvérové nástroje, aby mohli byť realizované širokým spektrom záujemcov o problematiku.

Záverečné kapitoly učebnice sa venujú aktuálnym trendom v oblasti Industry 4.0, kde sú opísané vybrané originálne aplikácie tohto konceptu vytvorené na univerzitnom pracovisku, a tiež výhľadu do blízkej budúcnosti, kde čoraz väčšiu rolu začne hrať prichádzajúci koncept Industry 5.0.

V súvislosti s konvergenciou informačných a operačných technológií je nevyhnutné pohľad na vzdelávanie prijímať holisticky a interdisciplinárne. Je kľúčové vytvárať medziodborové vzdelávacie programy, ktoré integrujú technické zručnosti z oblasti auto-

matizácie či elektrotechniky s najnovšími poznatkami z oblasti informačných a komunikačných technológií. Dôležitá je aj znalosť základov fungovania ekonomiky a spoločnosti ako takej. Týmto spôsobom pripravíme jednotlivcov nielen na porozumenie komplexným systémom, ale aj na tvorivé riešenie problémov a inovácie, ktoré sú nevyhnutné pre prosperujúcu a odolnú spoločnosť v ére digitálnej transformácie.

Opisované technológie a aplikácie odkrývajú pestrosť a rozmanitosť inovačných prístupov, ktoré stoja v popredí adaptívnej a efektívnej budúcnosti. Je nevyhnutné, aby sme našu súčasnú a budúce generácie vybavili potrebnými znalosťami a zručnosťami v oblasti digitálnych a inteligentných technológií. Rozvoj vzdelanostnej ekonomiky je kľúčovým kameňom pre prechod od tradičnej pracovnej sily k inovačnej a vedomostne orientovanej spoločnosti. Ak sa chce Slovensko posunúť vpred a nezostávať závislé len od lacnej pracovnej sily, je potrebné investovať do komplexného vzdelávania a podporovať kontinuálny rozvoj zručností. Veríme, že osveta v moderných technológiách položí pevný základ pre rast a poskytne inšpiráciu na ceste k tvorbe inteligentnejších, inkluzívnejších a udržateľnejších priemyselných systémov. Zdôrazňovanie významu vedy, technológie, inžinierstva a matematiky (STEM) v rámci vzdelávacích programov je esenciálne pre udržanie konkurencieschopnosti a zaistenie prosperujúcej budúcnosti pre všetkých.

Zoznam použitej literatúry

[1] Asad, U., Khan, M., Khalid, A., a Lughmani, W. A. Human-Centric Digital Twins in Industry: A Comprehensive Review of Enabling Technologies and Implementation Strategies. *Sensors 23*, 8 (2023).

[2] Ashton, K., et al. That 'Internet of Things' thing. *RFID journal 22*, 7 (2009), 97–114.

[3] Benešová, A., a Tupa, J. Requirements for education and qualification of people in industry 4.0. *Procedia Manufacturing 11* (2017), 2195 – 2202. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. https://doi.org/10.1016/j.promfg.2017.07.366.

[4] Beňo, L. *Moderné metódy ovládania mechatronických zariadení s využitím hlasových povelov založené na technológii Edge computingy*. Dizertačná práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2022. 106 s.

[5] Beňo, L., Pribiš, R., a Drahoš, P. Edge Container for Speech Recognition. *Electronics 10*, 19 (2021). DOI: 10.3390/electronics10192420.

[6] Bourke, K. *What is CODESYS and Why is it Important?* Real Pars. [online]. [cit. 2023-04-26]. Dostupné z: `https://www.motioncontroltips.com/instruction-lists-ils-plc-programming/`.

[7] Brecko, A., Kajáti, E., a Papcun, P. Industry 5.0 – technológie: bio-inšpirované technológie a inteligentné materiály. *ATP Journal 02/2022* (2022).

[8] Brecko, A., Kajáti, E., a Zolotová, I. Industry 5.0 – technológie: interakcie medzi človekom a strojom. *ATP Journal 01/2022* (2022), 40–41.

[9] Briš, L. *Využitie formalizmov Petriho sietí v riadení laboratórnych systémov*. Diplomová práca (*Vedúci práce: Alena Kozáková; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2016. 58 s.

[10] Bucsai, S. *Ovládanie a monitorovanie IoT zariadení s využitím zmiešanej reality*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2019.

[11] Bucsai, S., Kučera, E., Haffner, O., a Drahoš, P. Control and Monitoring of IoT Devices Using Mixed Reality Developed by Unity Engine. In *2020 Cybernetics & Informatics (K&I)* (2020), pp. 1–8.

[12] Burger, A., Koziolek, H., Rückert, J., Platenius-Mohr, M., a Stomberg, G. Bottleneck Identification and Performance Modeling of OPC UA Communication Models. In *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering* (Apr. 2019), ICPE '19, ACM. DOI: 10.1145/3297663.3309670.

[13] Cartwright, A. *OPC-UA: the Flow of Data*. [online]. [cit. 2024-01-20]. Dostupné z: `https://medium.com/ai-build-techblog/opc-ua-the-flow-of-data-7c3e5c870a4c`.

[14] Cavalieri, S., Salafia, M. G., a Scroppo, M. S. Integrating OPC UA with web technologies to enhance interoperability. *Computer Standards & Interfaces 61* (Jan. 2019), 45–64. DOI: 10.1016/j.csi.2018.04.004.

[15] Chaudhry, T., Juneja, A., a Rastogi, S. AR Foundation for Augmented Reality in Unity. *International Journal of Advances in Engineering and Management 3*, 1 (2021), 1–7.

[16] Ciolacu, M. I., Svasta, P., Berg, W., a Popp, H. Education 4.0 for Tall Thin Engineer in a Data Driven Society. *2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)* (2017), 432–437.

[17] Coito, T., Martins, M. S., Viegas, J. L., Firme, B., Figueiredo, J., Vieira, S. M., a Sousa, J. M. A Middleware Platform for Intelligent Automation: An Industrial Prototype Implementation. *Computers in Industry 123* (Dec. 2020), 103329. DOI: 10.1016/j.compind.2020.103329.

[18] Croatti, A., a Ricci, A. Towards the Web of Augmented Things. In *Software Architecture Workshops (ICSAW), 2017 IEEE International Conference on* (2017), IEEE, pp. 80–87.

[19] Dean, James. *First-of-its-kind 3D-printed home blends concrete, wood*. [online]. [cit. 2024-01-27]. Dostupné z: `https://news.cornell.edu/stories/2022/09/first-its-kind-3d-printed-home-blends-concrete-wood`.

[20] Drath, R., Mosch, C., Hoppe, S., Faath, A., Barnstedt, E., Fiebiger, B., a Schlögl, W. Diskussionspapier–interoperabilität mit der verwaltungsschale, opc ua und automationml. *Technical Report. AutomationML eV and Industrial Digital Twin Association (IDTA) and OPC Foundation and VDMA* (2023).

[21] Dvořák, J. *Integrace měřičů spotřeby energií do SCADA systému, zpracování a vyhodnocení dat*. Diplomová práca (*Vedúci práce: Jan Holub*). Praha: ČVUT Fakulta elektrotechnická, 2017. 68 s.

[22] European Commission - Directorate General for Research and Innovation. *Industry 5.0: towards a sustainable, human centric and resilient European industry.* Publications Office, 2021. DOI: 10.2777/308407.

[23] Fischertechnik. *Indexed Line with two Machining Stations 24V - Simulation*. [online]. [cit. 2021-04-07]. Dostupné z: https://www.fischertechnik.de/en/products/simulating/training-models/96790-sim-indexed-line-with-two-machining-stations-24v-simulation.

[24] Fischertechnik. *Punching Machine with Conveyor Belt 24v*. [online]. [cit. 2023-04-29]. Dostupné z: https://www.fischertechnik.biz/punching-machine-with-conveyor-belt-24v.

[25] Fortunato, D., a Bernardino, J. Progressive Web Apps: An Alternative to the Native Mobile Apps. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)* (2018), IEEE, pp. 1–6.

[26] Gallasch, A. ThingWorx–Plattform zur Integration herausfordernder Anforderungen auf dem Shopfloor. *Produktions-und Verfügbarkeits-optimierung mit Smart Data Ansätzen* (2018), 83–92.

[27] Grenčíková, A., Kordoš, M., a Navickas, V. The Impact of Industry 4.0 on Education Contents. *Business: Theory and Practice 22*, 1 (Jan. 2021), 29–38. DOI: 10.3846/btp.2021.13166.

[28] Guinard, D., Trifa, V., Pham, T., a Liechti, O. Towards Physical Mashups in the Web of Things. In *Networked Sensing Systems (INSS), 2009 Sixth International Conference on* (2009), IEEE, pp. 1–4.

[29] Habib, K., Saad, M. H. M., Hussain, A., Sarker, M. R., a Alaghbari, K. A. An Aggregated Data Integration Approach to the Web and Cloud Platforms through a Modular REST-Based OPC UA Middleware. *Sensors 22*, 5 (2022). DOI: 10.3390/s22051952.

[30] Haffner, O., a Kučera, E. Multiplatform Mobile Application for Identification and Localization of Objects in Space. In *2020 Cybernetics & Informatics (K&I)* (2020), pp. 1–9. DOI: 10.1109/KI48306.2020.9039849.

[31] Halliwell, R. *Programmable Logic Controllers*. op-tec. [online]. [cit. 2023-05-17]. Dostupné z: https://op-tec.co.uk/knowledge/using-programmable-logic-controllers/.

[32] Hrúz, B., a Mrafko, L. *Modelovanie a riadenie diskrétnych udalostných systémov: s využitím Petriho sietí a iných nástrojov*. Bratislava: Vydavateľstvo STU, 2003. ISBN: 80-227-1883-1.

[33] Huba, M., a Kozák, Š. From e-Learning to Industry 4.0. In *2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)* (Nov 2016), pp. 103–108. DOI: DOI: 10.1109/ICETA.2016.7802083.

[34] Imtiaz, J., a Jasperneite, J. Scalability of OPC-UA down to the chip level enables "Internet of Things". In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)* (July 2013), IEEE. DOI: 10.1109/indin.2013.6622935.

[35] Kajáti, E., a Zolotová, I. Industry 5.0 – revolúcia alebo evolúcia? *ATP Journal 12/2021* (2021), 36–37.

[36] Kolektív Advantech. *Quick Guide to OPC UA and Advantech IT/OT Convergent Solutions*. [online]. [cit. 2023-05-04]. Dostupné z: `https://page.advantech.com/en/global/industrial-automation/industrial-io/opc-ua`.

[37] Kolektív Automatizácia 365. *Ktoré PLC programovacie jazyky najviac používame?* [online]. [cit. 2022-05-17]. Dostupné z: `https://www.automatizacia365.sk/2021/04/13/ktore-plc-programovacie-jazyky-najcastejsie-pouzivame`.

[38] Kolektív autorov www.opto22.com. Ako preklenúť medzeru medzi IT a OT. *ATP Journal 1/2017* (2017), 32–33.

[39] Kolektív Firmata. *Firmata firmware for Arduino. GitHub*. 2016. [online]. [cit. 2020-08-20]. Dostupné z: `https://github.com/firmata/arduino`.

[40] Kolektív HiveMQ. *MQTT Essentials*. [online]. [cit. 2020-06-05]. Dostupné z: `https://www.hivemq.com/tags/mqtt-essentials/`.

[41] Kolektív Node-RED. *About Node-RED*. [online]. [cit. 2020-06-10]. Dostupné z: `https://nodered.org/about/`.

[42] Kolektív Node-RED. *Node-RED User Guide*. [online]. [cit. 2020-06-10]. Dostupné z: `https://nodered.org/docs/user-guide/`.

[43] Kolektív OpenPLC Project. *Modbus Address Mapping*. [online]. [cit. 2021-02-01]. Dostupné z: `https://www.openplcproject.com/reference/modbus/`.

[44] Kolektív Produktion 2030. *Ingenjör4.0*. [online]. [cit. 2021-02-07]. Dostupné z: `https://produktion2030.se/en/ingenjor-4-0`.

[45] Kolektív randed.com. *Information Technologies (IT) Vs Operational Technologies (OT). Webranded*. [online]. [cit. 2020-08-06]. Dostupné z: https://randed.com/information-technologies-it-vs-operational-technologies-ot/?lang=en.

[46] Kolektív Reality Reflection. *Virtual, Augmented, Mixed-Reality; What is these all about?. Medium.com*. [online]. [cit. 2020-06-02]. Dostupné z: https://medium.com/@realityreflection/vr-virtual-augmented-mixed-reality-what-is-these-all-about-25762bfda62a.

[47] Kolektív The Foundry. *VR? AR? MR? Sorry, I'm confused. The Foundry*. [online]. [cit. 2020-06-02]. Dostupné z: https://www.foundry.com/insights/vr-ar-mr/vr-mr-ar-confused.

[48] Kolektív Úradu podpredsedu vlády Slovenskej republiky pre investície a informatizáciu. *Cloud computing. Informatizácia - Úrad podpredsedu vlády Slovenskej republiky pre investície a informatizáciu*. [online]. [cit. 2020-06-01]. Dostupné z: http://www.informatizacia.sk/cloud-computing/22841s.

[49] Kostoláni, M., Murín, J., a Kozák, Š. Intelligent Predictive Maintenance Control Using Augmented Reality. In *2019 22nd International Conference on Process Control (PC19)* (2019), pp. 131–135. DOI: 10.1109/PC.2019.8815042.

[50] Kozák, Š., a Hypiusová, M. *Moderné metódy a algoritmy riadenia*. Bratislava: Slovenská chemická knižnica, 2016. ISBN: 978-80-89597-44-4.

[51] Kozák, Š., Ružický, E., Štefanovič, J., a Schindler, F. Research and Education for Industry 4.0: Present Development. In *2018 Cybernetics & Informatics (K&I)* (Feb 2018), pp. 1–7.

[52] Kučera, E., Haffner, O., Drahoš, P., Cigánek, J., Leskovský, R., a Štefanovič, J. New Software Tool for Modeling and Control of Discrete-Event and Hybrid Systems Using Timed Interpreted Petri Nets. *Applied Sciences 10*, 15 (2020), 5027. DOI: 10.3390/app10155027.

[53] KUNBUS GmbH. *Open Source IPC based on Raspberry Pi. Revolution Pi*. [online]. [cit. 2020-08-06]. Dostupné z: https://revolution.kunbus.com/.

[54] Kučera, E. *Využitie Petriho sietí na modelovanie a riadenie skladových systémov*. Diplomová práca (*Vedúci práce: Leo Mrafko; Konzultant: Branislav Hrúz*). Bratislava: FEI STU, 2013. 81 s.

[55] Kučera, E. *Modelovanie a riadenie hybridných systémov s využitím Petriho sietí vyšších úrovní*. Dizertačná práca (*Vedúci práce: Štefan Kozák*). Bratislava: FEI STU, 2016. 121 s.

[56] Kučera, E. *Digitálne technológie a systémy pre Industry 4.0*. Habilitačná práca. Bratislava: FEI STU, 2020. 224 s.

[57] Kučera, E. *Internet of Things - slajdy z prednášok Virtuálna a zmiešaná realita pre Industry 4.0*. [online]. [cit. 2020-06-01]. Dostupné z: `http://vzri.mechatronika.cool/sites/default/files/Prednaska%2010%20IoT.pdf`.

[58] Kučera, E. *Prehľad cloudových služieb Microsoft Azure - slajdy z prednášok Virtuálna a zmiešaná realita pre Industry 4.0*. [online]. [cit. 2020-06-01]. Dostupné z: `http://vzri.mechatronika.cool/sites/default/files/Prednaska%208%20Azure.pdf`.

[59] Kučera, E. *Virtuálna, rozšírená a zmiešaná realita - slajdy z prednášok Virtuálna a zmiešaná realita pre Industry 4.0*. [online]. [cit. 2020-06-01]. Dostupné z: `http://vzri.mechatronika.cool/sites/default/files/Prednaska%207%20VR.pdf`.

[60] Kučera, E., Haffner, O., Drahoš, P., a Kozáková, A. Modeling and Control of Discrete Event and Hybrid Systems Using Petri Nets and OPC Unified Architecture. *IEEE Access 10* (2022), 120735–120751. DOI: 10.1109/ACCESS.2022.3222828.

[61] Kučera, E., Haffner, O., Drahoš, P., Leskovský, R., a Cigánek, J. PetriNet Editor + PetriNet Engine: New Software Tool For Modelling and Control of Discrete Event Systems Using Petri Nets and Code Generation. *Applied Sciences 10*, 21 (2020). DOI: 10.3390/app10217662.

[62] Képešiová, Z. *Monitorovanie a riadenie mechatronických systémov s využitím digitálneho dvojčaťa a zmiešanej reality*. Písomná práca k dizertačnej skúške (*Vedúci práce: Danica Rosinová; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2019.

[63] Leskovský, R. *Moderné metódy ovládania a diagnostiky mechatronických zariadení s využitím IoT a zmiešanej reality*. Písomná práca k dizertačnej skúške (*Vedúci práce: Danica Rosinová; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2020.

[64] Lewicki, P. *Controlling lights with the Hololens and Internet of Things*. htmlfusion. [online]. [cit. 2019-04-08]. Dostupné z: `https://tinyurl.com/jgfbso2`.

[65] Liang, Q., a Li, L. The Study of Soft PLC Running System. *Procedia Engineering 15* (2011), 1234–1238. CEIS 2011. DOI: https://doi.org/10.1016/j.proeng.2011.08.228.

[66] Lopatniková, S. *Modelovanie a riadenie udalostného systému s využitím PLC a cloudového riešenia*. Diplomová práca (*Vedúci práce: Oto Haffner; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2022. 81 s.

[67] Mahnke, W., Leitner, S.-H., a Damm, M. *OPC Unified Architecture*. Springer Science & Business Media, 2009.

[68] Marton, M. *Riadenie udalostných systémov s využitím Petriho sietí a OPC UA*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2022. 44 s.

[69] Miksad, M. *Prepojenie open-source PLC s počítačom generovanou realitou*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2023.

[70] Milgram, P., a Kishino, F. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems 77*, 12 (1994), 1321–1329.

[71] OpenBCI. *Ultracortex "Mark IV" EEG Headset*. [online]. [cit. 2024-01-27]. Dostupné z: `https://shop.openbci.com/collections/frontpage/products/ultracortex-mark-iv`.

[72] Organizácia spojených národov (OSN). *17 cieľov udržateľného rozvoja*. [online]. [cit. 2024-01-27]. Dostupné z: `https://unis.unvienna.org/unis/sk/topics/sustainable_development_goals.html`.

[73] Pajpach, M., Haffner, O., Kučera, E., a Drahoš, P. Low-Cost Education Kit for Teaching Basic Skills for Industry 4.0 Using Deep-Learning in Quality Control Tasks. *Electronics 11*, 2 (2022). DOI: 10.3390/electronics11020230.

[74] Pajpach, M., Pribiš, R., Drahoš, P., Kučera, E., a Haffner, O. Design of an Educational-development Platform for Digital Twins using the Interoperability of the OPC UA Standard and Industry 4.0 Components. In *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)* (2023), pp. 1–6. DOI: 10.1109/ICECCME57830.2023.10252941.

[75] Papcun, P., Mičko, K., a Kajáti, E. Industry 5.0 – technológie: bezpečný prenos, ukladanie a analýza údajov. *ATP Journal 04/2022* (2022), 52–53.

[76] Pavlovič, D. *Riadenie udalostného systému s využitím PLC a mobilnej aplikácie*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2023.

[77] Pohanka, P. *Internet věcí*. [online]. [cit. 2020-06-01]. Dostupné z: `http://i2ot.eu/internet-of-things/`.

[78] Popjáková, D., a Mintálová, T. Priemysel 4.0, čo mu predchádzalo a čo ho charakterizuje - geografické súvislosti. *Acta Geographica Universitatis Comenianae 63*, 2 (2019), 173–192.

[79] Pribiš, R., Beňo, L., a Drahoš, P. Asset Administration Shell Design Methodology Using Embedded OPC Unified Architecture Server. *Electronics 10*, 20 (2021). DOI: 10.3390/electronics10202520.

[80] Rambach, J., Pagani, A., Stricker, D., Aleksy, M., Schmitt, J., Langfinger, M., Schneider, M., Schotten, H., Malignaggi, A., Ko, M., et al. Augmented Things: Enhancing AR Applications leveraging the Internet of Things and Universal 3D Object Tracking. In *IEEE International Conference on Industrial Technology (ICIT)* (2017), vol. 22, p. 25.

[81] Riesz, M. *PNEditor - a Petri Net editor. PNEditor*. 2014. [online]. [cit. 2016-04-24]. Dostupné z: `http://www.pneditor.org/`.

[82] Romero, D., Bernus, P., Noran, O., Stahre, J., a Fast-Berglund, Å. The Operator 4.0: Human Cyber-Physical Systems & Adaptive Automation Towards Human-Automation Symbiosis Work Systems. In *Advances in Production Management Systems. Initiatives for a Sustainable World* (Cham, 2016), I. Nääs, O. Vendrametto, J. Mendes Reis, R. F. Gonçalves, M. T. Silva, G. von Cieminski, a D. Kiritsis, Eds., Springer International Publishing, pp. 677–686.

[83] Stark, E. *Moderné metódy ovládania a monitorovania mechatronických systémov s využitím počítačom generovanej reality*. Dizertačná práca (*Vedúci práce: Peter Drahoš; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2019. 120 s.

[84] Stark, E., Kučera, E., Haffner, O., Drahoš, P., a Leskovský, R. Using Augmented Reality and Internet of Things for Control and Monitoring of Mechatronic Devices. *Electronics 9*, 8 (2020), 1272.

[85] Sterling, I., a Swaroop, P. *Control with your smart devices by staring and gesturing*. Arduino. [online]. [cit. 2019-04-08]. Dostupné z: `https://blog.arduino.cc/2016/07/26/control-with-your-smart-devices-by-staring-and-gesturing/`.

[86] Tiegelkamp, M., a John, K.-H. *IEC 61131-3: Programming industrial automation systems*, vol. 166. Springer, 2010.

[87] Trifa, V., Guinard, D., a Carrera, D. *Web Thing Model*. W3C. [online]. [cit. 2020-04-08]. Dostupné z: `http://model.webofthings.io/`.

[88] UniPi.technology. *Automatizace Jednoduše. Unipi*. [online]. [cit. 2020-08-06]. Dostupné z: https://www.unipi.technology/cs/.

[89] Unity Technologies. *Create a marker-based AR app*. [online]. [cit. 2023-05-06]. Dostupné z: https://learn.unity.com/project/create-a-marker-based-ar-app.

[90] Viganò, G. P. *M2MQTT for Unity*. GitHub. [online]. [cit. 2023-05-06]. Dostupné z: https://github.com/gpvigano/M2MqttUnity.

[91] Voráčová, v., Pěnička, M., a Veselý, J. *Úvod do modelování procesů Petriho sítěmi*. Vyd. 1. Praha: Česká technika - nakladatelství ČVUT, 2008, 126 s. ISBN 978-80-01-03979-3.

[92] Wipro. *Edge Computing - Understanding the User Experience*. [online]. [cit. 2024-01-11]. Dostupné z: https://www.wipro.com/infrastructure/edge-computing-understanding-the-user-experience/.

[93] Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L., a Zhou, W. A Comparative Study of Containers and Virtual Machines in Big Data Environment. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)* (2018), pp. 178–185. DOI: 10.1109/CLOUD.2018.00030.

[94] Zolotová, I., Kajáti, E., a Pomšár, L. Industry 5.0 – koncept, technológie, ciele. *ATP Journal 11/2021* (2021), 42–43.

[95] Český institut informatiky, robotiky a kybernetiky ČVUT. *Testbed pro Průmysl 4.0*. České vysoké učení technické v Praze. [online]. [cit. 2020-06-03]. Dostupné z: https://www.ciirc.cvut.cz/cs/teams-labs/testbed/.

[96] Češek, P. *Riadenie DEDS pomocou mikrokontrolérov s využitím Petriho sietí*. Diplomová práca (*Vedúci práce: Alena Kozáková; Konzultant: Erik Kučera*). Bratislava: FEI STU, 2016.

[97] Čížek, J. *Počítačový master/slave je společensky nekorektní. Připomíná otroctví, a proto zmizí*. [online]. [cit. 2024-01-20]. Dostupné z: https://www.zive.cz/clanky/pocitacovy-master/slave-je-spolecensky-nekorektni-pripomina-otroctvi-a-proto-zmizi/sc-3-a-204378/default.aspx.

[98] Štetáková, M. *Riadenie udalostného systému s využitím PLC a webovej aplikácie*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2023.

[99] Štěpánek, M. *Multiplatformová aplikácie pre získavanie dát o lokalitách s bezbariérovým prístupom*. Diplomová práca (*Vedúci práce: Erik Kučera*). Bratislava: FEI STU, 2018. 40 s.

[100] Žemla, F., Cigánek, J., Rosinová, D., Kučera, E., a Haffner, O. Smart Platform for Monitoring and Control of Discrete Event System in Industry 4.0 Concept. *Applied Sciences 13*, 19 (2023). DOI: 10.3390/app131910697.

doc. Ing. Erik Kučera, PhD.

DIGITÁLNE A INTELIGENTNÉ TECHNOLÓGIE PRE INDUSTRY 4.0

Vydala Slovenská technická univerzita v Bratislave vo Vydavateľstve SPEKTRUM STU, Bratislava, Vazovova 5, v roku 2024.

Edícia vysokoškolských učebníc

Rozsah 283 strán, 213 obrázkov, 7 tabuliek, 22,654 AH, 22,942 VH,
1. vydanie, edičné číslo 6199.

85 – 217 – 2024

ISBN 978-80-227-5430-9
DOI: 10.61544/GYSJ2849