ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA STROJNÍ

Studijní program: N0715A270012 - Průmyslové inženýrství a management

DIPLOMOVÁ PRÁCE

Řízení modelu Fischertechnik a sběr dat s využitím RaspberryPi

Autor: Bc. Matouš HUCL
Vedoucí práce: Doc. Ing. Zdeněk Ulrych, Ph.D.

Akademický rok 2021/2022

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta strojní
Akademický rok: 2021/2022

Studijní program: Průmyslové inženýrství a management
Forma studia: Prezenční
Specializace/kombinace: Průmyslové inženýrství a management (PIMN)

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení:	**Bc. Matouš HUCL**
Osobní číslo:	**S20N0086P**
Adresa:	Šťáhlavice 63, Šťáhlavy – Šťáhlavice, 33204 Nezvěstice, Česká republika
Téma práce:	Řízení modelu Fischertechnik a sběr dat s využitím RaspberryPi
Téma práce anglicky:	Fischertechnik model control and data collection using RaspberryPi
Vedoucí práce:	Doc. Ing. Zdeněk Ulrych, Ph.D. Katedra průmyslového inženýrství a managementu

Zásady pro vypracování:

- Analýza problematiky
- Návrh, stavba a naprogramování ukázkového modelu
- Návrh a realizace propojení s RaspberryPi s možností sběru dat a řízení modelu Fischertechnik
- Závěr

Seznam doporučené literatury:

- MALAGA MIROSLAV a ULRYCH ZDENĚK, *Základy řízení robotů pro strojní inženýrstoí*. První vydání. 2020, Plzeň: Západočeská univerzita v Plzni. ISBN 978-80-261-0486-5.
- PECINOVSKÝ, RUDOLF, *Začínáme programovat v jazyku Python*. 2020. Praha: Grada. 270 s. ISBN 978-80-271-1237-1.
- UPTON, Eben a Gareth HALFACREE, *Raspberry Pi Uživatelská příručka*. 2013. Brno: COMPUTER PRESS. 232 s. ISBN 978-80-251-4116-8.
- MONK, Simon, *Programming the Raspberry Pi: getting started with Python*. 2013 New York: McGraw-Hill. ISBN 978-0-07-180783-8.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum:

Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů, uvedených v seznamu, který je součástí této diplomové práce.

V Plzni dne: ……. …….
podpis autora

Poděkování

Rád bych tímto poděkoval svému vedoucímu Doc. Ing. Zdeněk Ulrych, Ph.D. za odbornou pomoc, vedení a svému konzultantovi Ing. Bc. Miroslavu Malagovi za ochotu a cenné rady. Dále bych chtěl poděkovat svojí rodině za veškerou podporu a trpělivost při mém studiu.

ANOTAČNÍ LIST DIPLOMOVÉ PRÁCE

AUTOR	**Příjmení** Hucl	**Jméno** Matouš
STUDIJNÍ PROGRAM	N0715A270012 - Průmyslové inženýrství a management	
VEDOUCÍ PRÁCE	**Příjmení (včetně titulů)** Doc. Ing. Ulrych, Ph.D	**Jméno** Zdeněk
PRACOVIŠTĚ	ZČU - FST - KPV	

DRUH PRÁCE	**DIPLOMOVÁ**	~~**BAKALÁŘSKÁ**~~	**Nehodící se škrtněte**
NÁZEV PRÁCE	Řízení modelu Fischertechnik a sběr dat s využitím RaspberryPi		

FAKULTA	strojní		**KATEDRA**	KPV		**ROK ODEVZD.**	2022

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	77		**TEXTOVÁ ČÁST**	49		**GRAFICKÁ ČÁST**	9

STRUČNÝ POPIS (MAX 10 ŘÁDEK) **ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY**	Práce se zabývá možnostmi využití jednodeskového počítače RPi při řízení a sběru dat ze skušebního modelu Fishertechnik. Práce je rozdělena na dva specifické úseky, teoretický a praktický. Teoretická část se věnuje popsání použitého hardwaru, jeho konektory, sběrnice, možnosti připojení a komunikace. V této části jsou popsány možnosti komunikace mezi zařízeními pomocí prokolů zpráv a jejich využití v průmyslu 4.0 a internetu věcí. V praktické části je vysvětleno nastavení MQTT serveru a práce s ním. Jsou zde také představeny dva koncepty sběru dat z modelu Fishertechnik. První koncept využívá GPIO piny na RPi. Druhý koncept využívá řídící jednotku Fischertechnik Controller TXT4.0. U obou konceptů bylo vytvořeno vzdálené řízení a ukládání dat do centrální SQLite databáze pomocí protokolu MQTT.
KLÍČOVÁ SLOVA **ZPRAVIDLA JEDNOSLOVNÉ POJMY, KTERÉ VYSTIHUJÍ PODSTATU PRÁCE**	Raspberry Pi, Fishertechnik, Protokol zpráv, MQTT, IoT, ROBO Pro Coding, Automatizace, Robotizace, Řízení, Sběr dat, Databáze, Python, SQLite

SUMMARY OF DIPLOMA SHEET

AUTHOR	**Surname** Hucl	**Name** Matouš
STUDY PROGRAMME	N0715A270017 - Industrial Engineering and Management	
SUPERVISOR	**Surname (Inclusive of Degrees)** Doc. Ing. Ulrych, Ph.D	**Name** Zdeněk
INSTITUTION	ZČU - FST - KPV	
TYPE OF WORK	**DIPLOMA**	~~**BACHELOR**~~ **Delete when not applicable**
TITLE OF THE WORK	Fischertechnik model control and data collection using RaspberryPi	

FACULTY	Mechanical Engineering	**DEPARTMENT**	KPV	**SUBMITTED IN**	2022

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	77	**TEXT PART**	49	**GRAPHICAL PART**	9

BRIEF DESCRIPTION **TOPIC, GOAL, RESULTS AND CONTRIBUTIONS**	The thesis deals with the possibilities of using a single board computer RPi, in controlling and collecting data from the Fishertechnik test model. The thesis is divided into two specific sections, namely theoretical and practical. The theoretical part is devoted to the description of the hardware used, its connectors, bus, connection and communication options. In this section, the possibilities of communication between devices using message passing and their use in Industry 4.0 and IoT are described. The practical part explains the setup of the MQTT server and how to work with it. Two data acquisition concepts from the Fishertechnik model are also introduced. The first concept uses GPIO pins on the RPi. The second concept uses the Fishertechnik Controller TXT4.0. For both concepts, remote control and data storage in a central SQLite database using the MQTT protocol was created.
KEY WORDS	Raspberry Pi, Fishertechnik, Message protokols, MQTT, IoT, ROBO Pro Coding, Automation, Robotics, Control, Data collection, Database, Python, SQLite

Obsah

Přehled použitých zkratek a symbolů

RPi	Raspberry Pi	Raspberry Pi
OS		Operační systém
STEM	Science, Technology, Engineering, Mathematics	Věda, Technologie, Technika, Matematika
IoT	Internet of things	Internet věcí
MQTT	MQ Telemetry Transport	MQ Telemetrická přeprava
SBC	Single-Board Computer	Jednodeskový počítač
I/O	Inputs/Outputs	Vstupy výstupy
GPIO	General Purpose Input/Output	vstupy/výstupy pro obecné účely
I^2C	Inter-Integrated Circuit	Interintegrovaný obvod
PC	Personal computer	Osobní počítač
PLC	Programmable Logic Controllers	Programovatelné logické ovladače
PCB	Printed Circuit Boards	Desky s plošnými spoji
WLAN	Wireless Local Area Network	Bezdrátová lokální síť
KPV		Katedra průmyslového inženýrství a managementu
ZČU		Západočeská univerzita v Plzni
SCL	Clock singal	Hodinový signál
SDA	Data signal	Datový signál
UART	Univerzal asynchronous transmitter/receiver	Univerzální asynchronní přijímač/vysílač
SPI	Serial Peripheral Interface	Sériové periferní rozhraní
CS	Chip Select	Výběr čipu
SS	Slave Select	Výběr podřízeného zařízení
CLK	Clock signal	Hodinový signál
MOSI	Master out slave in	Master ven slave dovnitř
MISO	Master out slave out	Master dovnitř slave ven
SQL	Structured Query Language	Strukturovaný dotazovací jazyk
RFID	Radio Frequency Identification	Radiofrekvenční identifikace
LPWAN	Low Power Wide Area Network	Rozsáhlá síť s nízkou spotřebou energie
BLE	Bluetooth Low Energy	Bluetooth s nízkou spotřebou energie
AR	Augment reality	Rozšířená realita
CoAP	Constrained Application Protocol	Omezený aplikační protokol
HTTP	Hypertext Transfer Protocol	Hypertext Transfer Protokol
AMQP	Advanced Message Queuing Protocol	Pokročilý protokol front zpráv
TCP	Transmission Control Protocol	Protokol řízení přenosu
QoS	Quality of Service	Kvalita služeb
TLS	Transport Layer Security	Zabezpečení přenosové vrstvy
IP	Internet Protocol	Internetový protokol
PyPi	Python Package Index	Rejstřík balíčků jazyka Python
UI	User interface	Uživatelské rozhraní

Úvod

Malé jednodeskové počítače se v současné době stávají stále oblíbenější, především kvůli jejich nízkým pořizovacím cenám, potenciálu v oblasti výkonu a všestrannému použití. Jednodeskový počítač je mikroprocesor s jednou deskou plošných spojů. Většinou obsahují rozsáhlou sadu podpůrných obvodů. Jde typicky o bloky pro logické a analogové vstupy/výstupy, pro komunikační linky a další aplikační logiku. Jedním z takových zařízení je Raspberry Pi. Raspberry Pi se od svého vydání v roce 2012 používá k vytvoření všeho - od robotů po vzdálená monitorovací zařízení. Jednodeskové počítače našly uplatnění v technickém vzdělání, protože skvěle zapadají do konceptu STEM, kterým se označuje kombinace vzdělávání v oborech vědy, techniky, technologie a matematiky. Kromě vzdělávání se též začínají široce využívat v průmyslové praxi pro své projektové schopnosti, jak mezi inženýry, tak i výrobci. Tyto mikropočítače díky snadnému propojení a vzdálené komunikaci splňují požadavky konceptu Průmyslu 4.0 a jeho nástrojů, jako je automatizace a platforma IoT (Internet věcí).

Tato práce se zabývá možnostmi využití hardwaru pro sběr dat a jeho řízení. Jako model pro sběr dat slouží polytechnická stavebnice Fischertechnik. Stavebnice se skládá z mechanických součástí (převodovky, ozubená kola apod.) a z výkonových součástí (elektromotory, ventily, světelné moduly atd.). Z této stavebnice byl vytvořen zkušební model, který simuluje výrobní linku. Výrobní linka reprezentuje pracoviště rentgenu, selekci zmetků a balení materiálu. Stavebnice Fishertechnik nabízí i řídící jednotku Fishertechnik Controller TXT4.0 a vývojové grafické prostředí ROBO Pro Coding. V práci je uvedena i varianta s použitím tohoto hardwaru a softwaru a možnosti jeho využití. Cílem je vyzkoušet možnosti ROBO Pro Coding při tvorbě řídícího programu a komunikace s jinými zařízeními, nejenom s těmi, které nabízí společnost Fishertechnik.

Pro možnost síťového propojení a výměny dat byl použit protokol MQTT. V tomto případě slouží jako snadný způsob pro vzdálené získávání dat, ovládání Raspberry Pi a komunikace se softwarem ROBO Pro Coding od společnosti Fishertechnik. Díky tomu je možné vytvořit síť z více zařízení, řídit je a získávat z nich data s pomocí jednoho centrálního PC.

V této práci jsou uvedeny dva implementační koncepty propojení:

1. Zkušební model stavebnice Fishertechnik je napojen na Raspberry Pi pomocí GPIO pinů. Řídící program byl vytvořený pomocí programovacího jazyka Python.
2. Zkušební model stavebnice Fishertechnik je napojen na řídící jednotku Fischertechnik Controller TXT4.0. Řídící program byl vytvořen pomocí softwaru ROBO Pro Coding.

U obou konceptů bylo vytvořeno vzdálené řízení a výměny dat pomocí protokolu MQTT. Data získaná z modelu se ukládají do centrálního počítače do SQLite databáze. Vzdálené řízení je řešeno pomocí jednoduchého uživatelského rozhraní. Primárním programovacím jazykem je Python.

1 Jednodeskové počítače

V posledním desetiletí se ve strojírenství a v informatice stále častěji používají jednodeskové počítače, jak na technické tak i vzdělávací úrovni. Za tím stojí celá řada faktorů, jako je všestrannost, nízká cena, a možnost zlepšit proces učení informatiky a robotiky. Od uvedení prvního modelu Raspberry Pi v roce 2012 se zájem o jednodeskové počítače (Single-Board Computers, dále jako SBC) v technice a informatice v průběhu let zvýšil. Uvedení počítače Raspberry Pi (dále jako RPi) vedlo k výrazným změnám na trhu SBC. Podobné produkty jako je Gumstix byly k dispozici již od roku 2003, Raspberry Pi se však prodával v mnohem větších objemech. To způsobilo, že společnost, která za ním stojí, je jednou z nejrychleji rostoucích počítačových společností na světě [1]. Tento úspěch vedl k dramatickému nárůstu počtu výrobců SBC. Každý z těchto produktů je zaměřený na jiné využití. Vznikla tak velká škála funkcí, které jsou na SBC k dispozici. Tento trend byl podpořen faktory, jako je snižování výrobních nákladů, rozvoj elektronických technologií a jejich využití v praxi a naléhavá potřeba reformovat tradiční učební osnovy v inženýrství a informatice. Jednodeskové počítače jsou doménou průmyslového a automatizačního odvětví. V tomto odvětví se uchytily na pozici řídicích systémů jako vestavěné moduly určené k řízení výrobních systémů, robotů, dále pak ke zpracování signálů, kontrole datové komunikace apod. V oblasti průmyslu se používají specifické jednodeskové počítače. Jsou vybírány cíleně (na míru) pro daný účel a poté jsou nakonfigurovány. [2]

Jednodeskový počítač je kompletní počítač postavený na jediné obvodové desce. Jednodeskový počítač je označení vzniklé spíše historicky. Dnes je mnoho osobních počítačů, např. notebooky majících všechny komponenty na jedné desce, a tak původní vymezení, že jednodeskový počítač integruje všechny nutné komponenty na jeden plošný spoj, je sice stále platné, ale je platné i pro další typy počítačů. Na rozdíl od moderních desktopových počítačů SBC disponují RAM pamětí o podstatně nižší kapacitě a také levnějšími a méně výkonnými procesory s menší bitovou šíří. Jednodeskové počítače jsou specifické svojí architekturou. Zatímco osobní počítače jsou vybaveny vícejádrovými procesory s taktem v řádech GHz a architekturou x86, jednodeskové počítače používají mnohem jednodušší procesory převážně architektury AVR nebo ARM. Modernější modely obsahují také vícejádrové procesory, na mnoho projektů stačí velmi jednoduché čipy s pracovní frekvencí v řádech MHz. SBC jako např. mikrokontroléry mají programovatelné vstupy a výstupy. Největší rozdíl mezi SBC a osobními počítači je ten, že osobní počítače obvykle nemají rozšiřující sloty a vlastní programovatelné I/O (vstupy, výstupy). Na rozdíl od mikrokontrolérů mají SBC obvykle rozšiřitelnou paměť pro uložení operačního systému potřebného k provozu systému. S rostoucím výkonem však SBC již dokáží zastoupit plnohodnout roli osobního počítače (např. Raspberry Pi 4B).

Mezi rozšířené SBC patří např. Arduino (Arduino UNO, MEGA), Intel Galileo, ESP32 a již zmíněné Raspberry Pi. Každá z těchto značek nabízí celou řadu výrobků pro jiné využití. Po Raspberry Pi jednodeskových počítačích je mezi komunitou nejvíce rozšířeno Arduino. Tento SBC je založen na mikrokontrolerech ATmega od firmy Atmel. Arduino vzniklo kolem roku 2005 jako nástroj pro studenty Interaction Design Institute Ivrea v Itálii. Studenti ve svých projektech používali mikrokontrolér BASIC Stamp, ale aby ušetřili peníze a zvýšili flexibilitu, vzniklo Arduino a jazyk Wiring, který se používá k psaní kódu pro desky založené na ATmega. Od té doby se objevilo mnoho různých modelů Arduina, nejrozšířenější se stala deska Arduino Uno. [3] Největší rozdíly mezi deskami Arduino a RPi: Arduino jsou

mikrokontroléry (nikoli plnohodnotné počítače), zatímco desky Raspberry Pi jsou mikroprocesory. Mikrokontrolér na desce Arduino obsahuje CPU, RAM a ROM. Veškerý další hardware na desce Arduino je určen pro napájení, programování a IO konektivitu. Raspberry Pi SBC má všechny vlastnosti počítače s procesorem, pamětí, úložištěm, grafickým ovladačem a konektory na desce. Z tohoto důvodu je k RPi možné připojit periferie jako je obrazovka, klávesnice a myš. Raspberry Pi má vlastní operační systém, zatímco desky Arduino žádný nemají. Arduino nepotřebuje žádný operační systém. Vše, co potřebuje, je binární soubor zkompilovaného zdrojového kódu. Desky Arduino pracují na základě jednoduchých instrukcí, které jim poskytuje IDE (integrované vývojové prostředí). Díky těmto „ústupkům“ jsou obecně desky Arduino levnější, desky Raspberry Pi jsou o něco dražší. [4]

1.1 RaspberryPi

Raspberry Pi (dále již jen RPi) je řada malých jednodeskových počítačů vyvinutých společností Raspberry Pi Foundation pro výuku základů počítačové vědy. Díky jejich jednoduchosti a všestranné použitelnosti se staly populárním, ale i experimentálním nástrojem pro vývoj školních projektů, hardwarového programování, robotiky, základních automatizovaných strojů, obvodů atd. RPi je plně vybavený počítač velikosti kreditní karty, na kterém se nejčastěji používají operační systémy s linuxovým jádrem. Počítač má všechny potřebné propojovací porty, kam může uživatel připojit periferní zařízení. Monitor lze připojit přes zásuvku HDMI, myš a klávesnici do USB portů a pro reproduktory RPi poskytuje 3,5mm audio jack. U modelu B+ je navíc ethernetová zásuvka pro připojení k internetu. Zvládne tedy cokoli, co dokáže běžný počítač. Kromě toho poskytuje počítač RPi široké možnosti v oblasti multimédií a 3D grafiky. Další předností RPi je, že disponuje řadou portů GPIO (General Purpose Input/Output, vstupy/výstupy pro obecné účely), které lze použít k interakci s klávesnicemi, myší, monitorem, sensory, motory, světly a dalšími zařízeními. Díky těmto pinům, všestrannosti, velikosti a výkonu, RPi postupně nachází uplatnění i v celých řadách průmyslových aplikací. [5]

První část názvu RPi vychází z dlouhodobé tradice pojmenování počítačových systémů podle ovoce (Apple, Apricot atd.). Druhá polovina názvu odkazuje na programovací jazyk Python. Na počátku vývoje počítače byl Python plánován jako jediný použitelný programovací jazyk, ale vzhledem k rozmanitosti a výkonnosti této platformy se začaly používat i jazyky další. [5]

1.1.1 DOSTUPNÉ MODELY

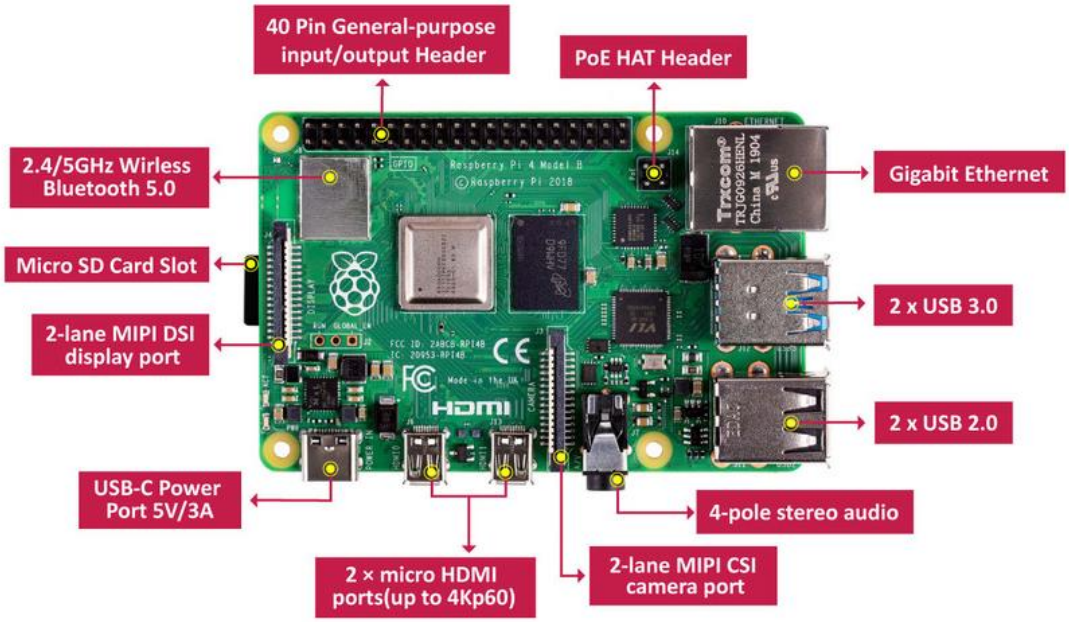
Raspberry Pi foundation nabízí celou řadu modelů pro různá využití. Nejmenší varianta, Raspberry Pi Pico, se dá pořídit na českém trhu již za 110 Kč. RPi Pico je levné zařízení s mikrokontrolérem a konkuruje/doplňuje tak např. mikrokontroléry z rodiny Arduino a podobné. Mikrokontroléry jsou velmi malé počítače, ale většinou postrádají velké úložiště a periferní zařízení, která k nim lze připojit (například klávesnice nebo monitory). RPi Pico však disponuje piny GPIO, stejně jako hlavní série počítače RPi, což znamená, že jej lze použít k ovládání a přijímání vstupů z různých elektronických zařízení. [6][7]

Dalším modelem je Raspberry Pi Zero. Tento model již disponuje slotem na microSD kartu, konektorem pro kameru CSI-2, čtyřicetikolíkovou GPIO hlavičku kompatibilní s HAT. Je napájen přes zásuvku micro USB a obrazový výstup je veden přes mini HDMI port. [7]

K této práci byl zvolen produkt RPi 4 Model B. RPi 4 Model B je nejnovější produkt v řadě počítačů RPi. Tento produkt byl zvolen proto, že má možnosti připojení periferií, velkého uložiště a výkonu.

1.1.2 RASPBERRY PI 4 MODEL B

RPi 4 model B nabízí zvýšení rychlosti procesoru, výkonu, paměti a konektivity ve srovnání s předchozí generací. Pro koncového uživatele poskytuje RPi 4 Model B desktop výkon srovnatelný se základními x86 PC systémy. Mezi klíčové vlastnosti tohoto produktu patří vysoce výkonný 64bitový čtyřjádrový procesor, Broadcom BCM2711, Quad core Cortex-A72 (ARM v8). Procesor obsahuje Videocore VI Graphics Processing Unit (GPU) obsluhující veškeré grafické vstupy/výstupy. Byl navržen tak, aby si poradil s rozlišením 4K a videem H.265, stejně jako se škálováním videa, vstupem pro kameru a všemi HDMI a kompozitními video výstupy. RPi podporuje připojení dvou displejů v rozlišení až 4K prostřednictvím dvojice micro-HDMI portů, hardwarové dekódování videa až 4Kp60. Na trhu je dostupná verze s až 8GB RAM (k této práci byla zvolena verze s 4GB pamětí). Pi navíc disponuje dvoupásmovou bezdrátovou LAN 2,4/5,0 GHz, Bluetooth 5.0, Gigabit Ethernet, 4x USB 3.0 a PoE HAT Header. PoE HAT umožňuje napájet RPi pomocí sítí s podporou napájení přes Ethernet. Aby bylo možné tento produkt používat, musí mít síť, ke které je připojen, nainstalováno zařízení pro zdroj napájení. [7] Přehled rozhraní lze vidět na obr. 1-1.



Obrázek 1-1 RPi model 4B rozhraní [8]

RPi neobsahuje jako klasický stolní počítač pouze USB a síťovou kartu s konektorem RJ45, ale i specializované sběrnice pro připojení hardwaru. To dělá z Raspberry nejen hračku, ale též nástroj určený pro řízení a monitorování.

1.1.3 SBĚRNICE RASPBERRY PI

Sběrnice dělají z RPi zajímavá zařízení pro mnohé praktické účely, například pro protokolování dat. Záznam dat znamená shromažďování informací ze senzorů a jejich ukládání pro pozdější analýzu. RPi může sbírat data pomocí těchto sběrnic:

GPIO – poskytuje vstup/výstup piny, speciálně pak rozhraní UART, sběrnice I²C / SPI. Zároveň slouží i k připojení rozšiřujících modulů. Moduly (shieldy) jsou hotové desky, které plní řadu dalších funkcí, jimiž RPi nedisponuje. Díky GPIO můžeme připojit k Raspberry velkou řadu čidel, expandérů sběrnic a převodníků. Podrobnější popis GPIO pinů je v kapitole 3.1. [7]

CSI camera interface – slouží pro připojení specializované kamery přes rozhraní CSI. Existují dva různé druhy konektorů fotoaparátu RPi CSI: 15pinový a 22pinový. 15pinový konektor je ve standardních modelech RPi (řada A&B) a modulech fotoaparátu Pi. 22-pin je na Raspberry Pi Zero-W a Compute Module IO Board. [9]

DSI display interface – slouží k připojení externího LCD displeje. DSI je zkratka pro Display Serial Interface a definuje vysokorychlostní sériové rozhraní mezi hostitelským procesorem a zobrazovacím modulem. Často se nazývá MIPI DSI (mobile industry processor interface display serial interface). Displeje DSI jsou však na rozdíl od HDMI určeny pro konkrétní zařízení. Rozhraní DSI je široce používáno v mobilních telefonech, noteboocích, nositelných zařízeních a různých dalších zařízeních. Všechny desky Raspberry Pi mají na desce 15kolíkový konektor DSI a k propojení mezi RPi a displejem DSI je potřeba 15kolíkový plochý kabel. [10]

1.1.4 OS RASPBERRY PI

V této práci bylo pracováno s operačním systémem Raspberry Pi OS (dříve nazývaný Raspbian). Tento OS je oficiálně, optimalizovaný a podporovaný operační systém pro Raspberry Pi společností Raspberry Pi Foundation. Je to systém odvozený z Debianu, jedna z nejrozšířenějších linuxových distribucí na světě. [3] Na počítač RPi existuje mnoho operačních systémů. Každý se liší zejména v míře optimalizace pro danou architekturu procesoru. Jako příklad je možno uvést OpenELEC, Risc OS, Arch Linux, OSMC. [11]

1.1.5 PROGRAMOVACÍ JAZYKY

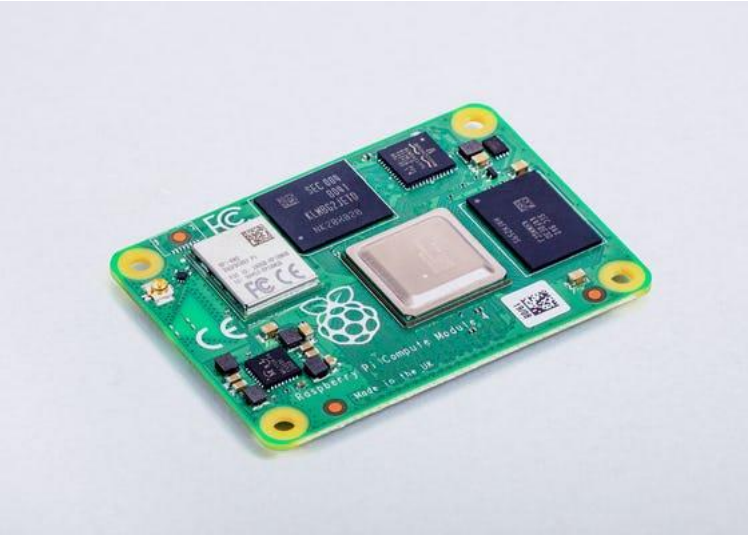
Existuje značné množství programovacích jazyků, které byly upraveny pro RPi. Nejpoužívanějším jazykem na RPi je Python, ale na zařízení se dá použít v podstatě jakýkoli programovací jazyk, jehož lze zkompilovat pro ARMv6. Uživatelé tedy nejsou omezeni na používání pouze Pythonu. Na RPi je předinstalovaných několik jazyků, například C, C++, Java, Scratch a Ruby. [7]

1.2 Využití Raspberry Pi

Využití Raspberry Pi je takřka neomezené a záleží pouze na kreativitě vývojáře. Pro běžného uživatele může posloužit jako stolní PC, herní server, rádio FM, či webový server. Dalším zajímavým použitím je chytrá domácnost, kde RPi může měřit a regulovat teplotu, vlhkost a sloužit jako kamerový bezpečnostní systém [12]. Jeho využití najdeme i v průmyslu. RPi je široce využíván pro své projektové schopnosti mezi inženýry i výrobci. Od svého vydání v

roce 2012 se používá od robotiky po vzdálená monitorovací zařízení. Nedlouho poté vzali průmysloví manažeři na vědomí schopnosti malé desky a upravili ji pro použití ve výrobě a v automatizačních operacích. Někteří vytvořili pro RPi specializované vybavení, aby tyto aplikace byly efektivnější a levnější. RPi samo o sobě poskytuje atraktivní alternativu k PLC (Programmable Logic Controllers), které jsou upřednostňovány v mnoha průmyslových odvětvích,~~,~~ včetně výroby, automatizace a platforem IoT (Internet of things). Být nákladově výhodnější i efektivní při provádění naprogramovaných úkolů prostřednictvím jednoduchého rozhraní GPIO, umožnilo Raspberry proniknout do těchto odvětví a použít je pro běžné operace. Použití desky v průmyslovém prostředí bylo tak rozšířené, že nadace navrhla speciální desku založenou na Pi pouze za účelem implementace do průmyslových provozů. Tak se zrodil Raspberry Pi Compute Module (obr. 1-2). [13]

Raspberry Pi Compute Module (dnes již k dostání Module s pořadovým číslem 4). je vyzbrojen výpočetním výkonem populárního RPi, který je zabalen do kompaktního formátu. První modul byl navržen tak, aby se vešel do standardního konektoru DDR2 SODIMM. Výpočetní modul se nápadně podobal paměti RAM a byl určen k integraci do desek plošných spojů neboli PCB (Printed Circuit Boards). [14] Poté, co se Raspberry Compute Module dostal na trh, se výrobci snažili začlenit kartu do různých modulárních designových skříní, aby se zaměřili na specifické průmyslové systémy, jako jsou systémy vyžadující tenké platformy na DIN lištu, interní napájecí zdroje pro redundanci a relé pro řízení automatizačních systémů. Příklad začlenění Compute Module pro průmyslové, IoT a automatizační aplikace pochází od společnosti TechBase s jejím průmyslovým počítačem ModBerry 500/9500. Vše o ModBerry lze přizpůsobit pro cílené platformy, včetně počtu sériových portů, analogových vstupů/výstupů, ethernetových portů, bezdrátových/GSM modemů, softwarových možností a dokonce i způsobu montáže krytu. [13]



Obrázek 1-2 RPi Compute Module [7]

Příkladem je produkt Monarco HAT (obr. 1-3) od společnosti Monarco. Tento nástavec je rozšíření pro standartní Raspberry Pi a umožňuje snadnější monitorování, sběr dat a automatizaci. K Monarco HAT lze připojit standardní průmyslové senzory nebo komunikační rozhraní a je tak možné snadno monitorovat stroj nebo výrobní proces. Monarco k produktu dodává i jednoduchý systém REXYGEN, což je sada softwarových nástrojů pro automatizaci, měření a regulaci, sběr dat, robotiku a řízení technologických procesů. REXYGEN nabízí grafické vývojové prostředí pro tvorbu algoritmů. Pracuje na standardním PC nebo notebooku

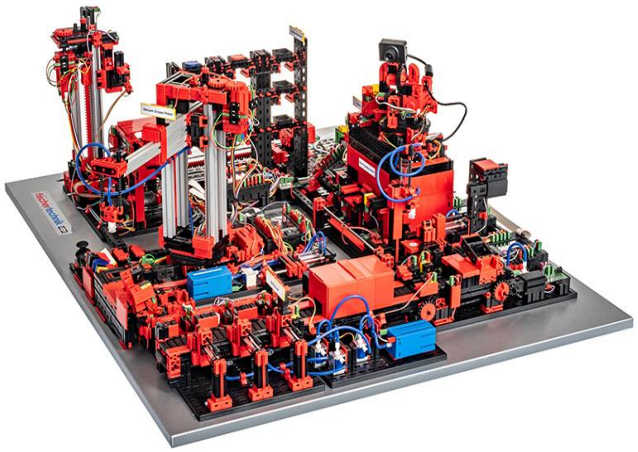
a programuje se pomocí tzv. funkčních bloků. Těch je k dispozici celá řada, od jednoduchých časovačů a komparátorů, až po stavový automat nebo PID regulátor. [8]

Obrázek 1-3 Monarco HAT [8]

2 Fischertechnik

Fischertechnik je německá společnost založená roku 1965 Arthurem Fisherem. Společnost se zaměřuje na vývoj a výrobu polytechnické stavebnice, umožňující stavbu kostry modelů ze stavebních bloků a mechanických součástí (včetně převodovek, ozubených kol apod.), výkonových součástí (elektromotory, ventily, kompresory, USB kamery, optické barevné senzory, světelné moduly) a součástí pro sběr/vyhodnocení údajů (např. NTC rezistor pro měření teploty). Stavebnici lze v případě potřeby rozšířit o další mechanické i výkonové součástky. Fischertechnik byl dříve určen jako hračka pro děti zajímající se o konstrukci. [15] Dnes se však využití stavebnice rozrůstá, například pro výuku a porozumění technologiím na středních odborných školách, výzkum a vývoj na univerzitách, v průmyslových firmách a IT odděleních. [16]

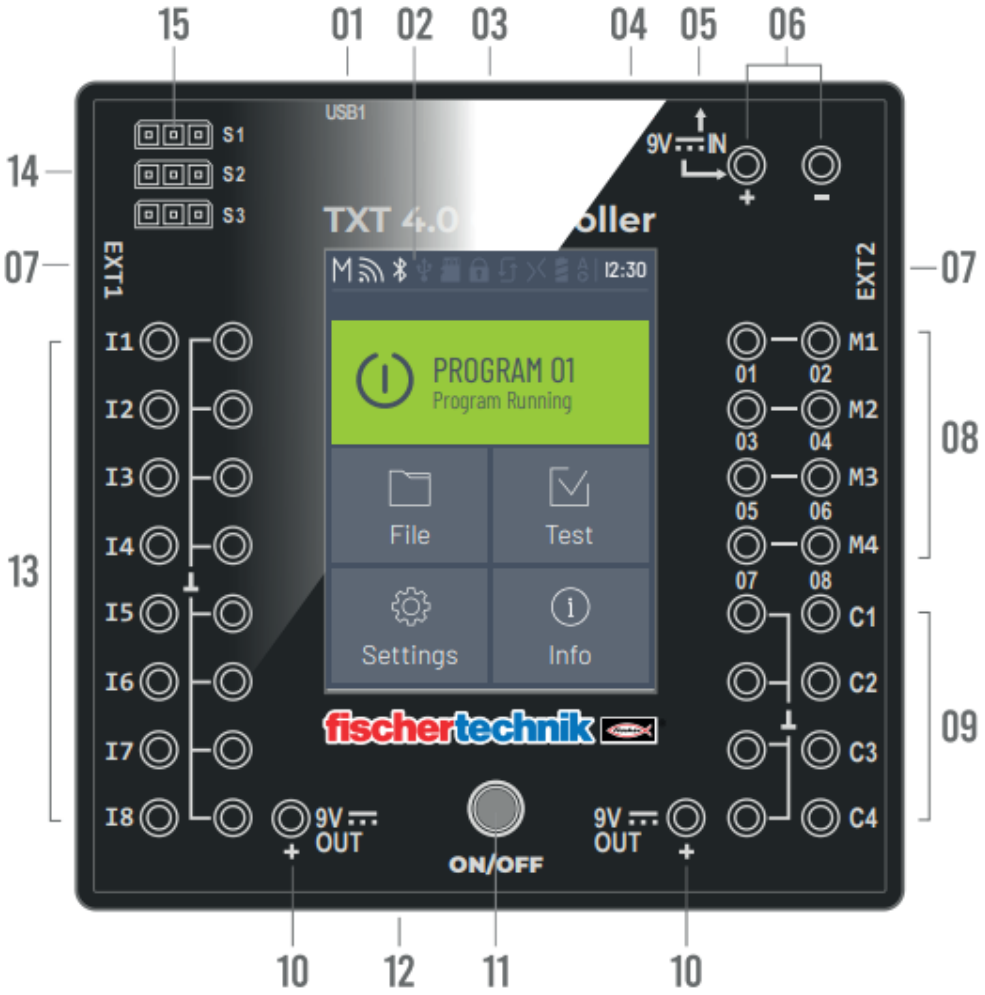
V roce 2014 byla představena nová generace mobilních robotů s inovativním ovládáním, TXT ovladačem a kamerou se zpracováním obrazu. Tato řada nese název ROBOTICS set. Všechny díly, s kterými se pracuje v praktické části jsou z této série. Roku 2020 byla také představena nová řada Fischertechnik Training Factory 4.0 (obr. 2-1). Ta se zaměřuje na vzdělávání v oblasti nových technologií a průmyslu 4.0 a cílí na edukaci do škol a průmyslových firem. [15] Fischertechnik Training Factory 4.0 znázorňuje jednoduchou automatizovanou továrnu. Tato továrna je tvořena pěti moduly, které reprezentují jednotlivé výrobní úseky. V setu se nachází automatizovaný paletový sklad, multifunkční obráběcí stanice, tříosý manipulační jeřáb, třídící linka a stanice pro import a export a kamera mapující prostředí. Dohromady tvoří celou linku a simulují moderní výrobní proces. Model je ovládán pomocí webového ovládacího rozhraní, které monitoruje a ukládá aktuální stav modelu a nabízí řízení ze tří různých pohledů (zákaznický pohled, dodavatelský pohled a výrobní pohled). Training Factory 4.0 disponuje zároveň integrovaným senzorem prostředí, který hlásí hodnoty teploty, vlhkosti, tlaku vzduchu a kvality vzduchu. [16]



Obrázek 2-1 Fischertechnik Training Factory 4.0 [16]

2.1 Robotics set

Stavebnice Fischertechnik Robotics je soubor modulárních komponent, které jsou určeny pro stavbu robotů. I vzhledem k jednoduchosti stavebnice se dají vytvářet komplexní systémy. K oživení a řízení modelů Fischertechnik slouží programovatelná řídící jednotka Robotics TXT Controller. Set nabízí dvě verze řídící jednotky Robotics TXT controller a Robotics Controller TXT4.0 (viz. obr. 2-2), který byl použit pro tuto práci. Tvorba řídících programů pro řídící jednotku TXT4.0 primárně probíhá ve vývojovém prostředí ROBO Pro Coding (více viz. kapitola 7.1). [15][17]



Obrázek 2-2 Řádící jednotka ROBOTICS TXT4.0 Controller [17]

Přehled rozhraní (vstupů/výstupů) Robotics TXT Controlleru:

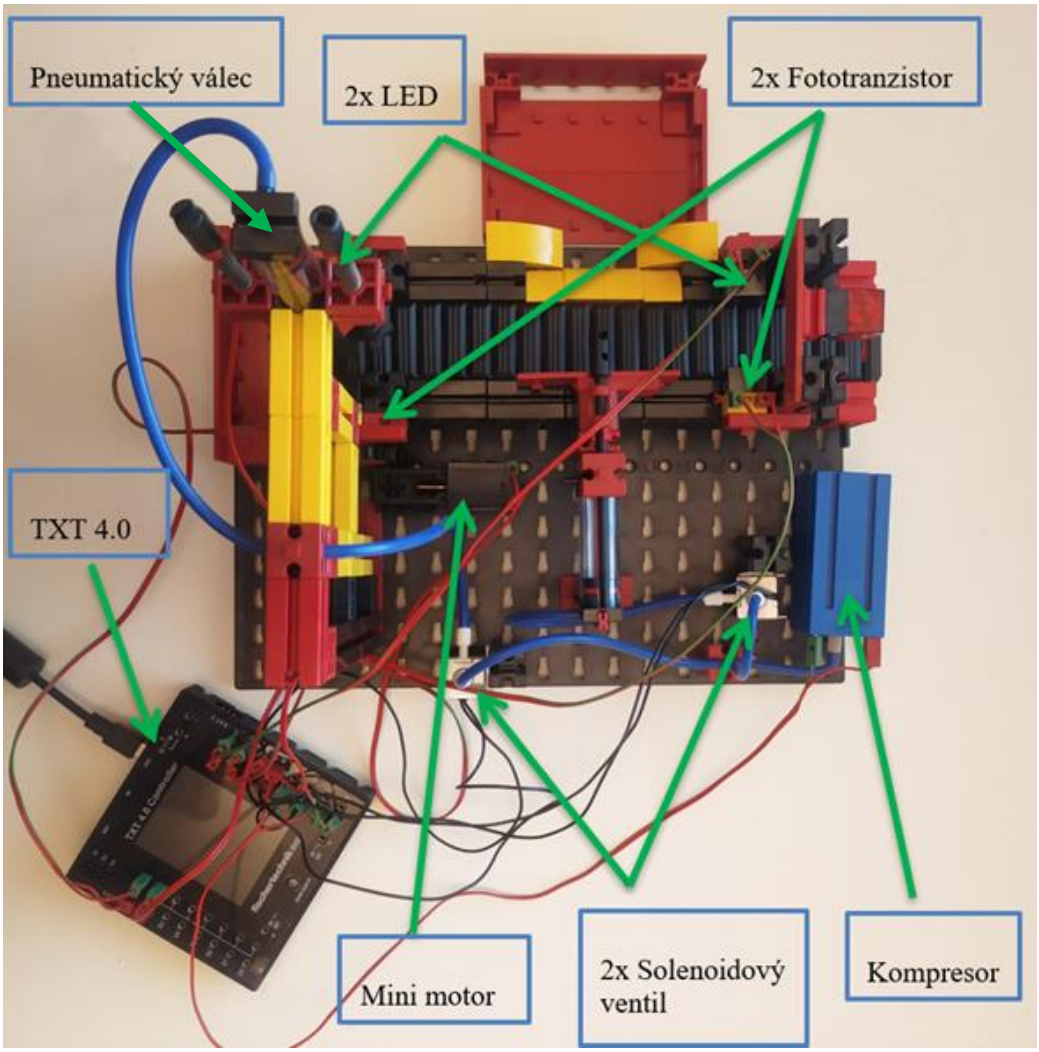
1. USB-A port (USB1)
2. Dotykový display
3. Slot pro micro SD kartu
4. Mini USB port (USB2)
5. 9V IN, DC port (zdroj)
6. 9V IN, Battery Pack connection
7. EXT připojení pro další rozšíření
8. Outputy M1–M4 nebo O1–O8
9. Inputy C1–C4

10. 9V Out
11. On / off tlačítko
12. Reproduktor
13. Univerzální inputy I1– I8
14. Outputy S1-S3 for servo motory [17]

TXT 4.0 disponuje 512 MB RAM a 4 GB paměti eMMC, na kterou lze ukládat programy pro řízení jednotky. Dále disponuje třemi servo výstupy a kapacitním dotykovým displejem. Vylepšený modul WLAN a Bluetooth nabízí vhodné bezdrátové rozhraní pro řadu aplikací. Dalším rozhraním je hostitelský port USB, ke kterému lze připojit například USB kameru fischertechnik. Jednotka může fungovat dle nastavení buď jako Master nebo Extension (rozšíření). Řídicí jednotka nastavená jako Master přijímá řídící příkazy přímo z počítače a předává je přídavným zařízením. Řídicí jednotka nastavená jako extension přijímá řídicí příkazy pouze od řídicí jednotky master. K jednomu ovladači lze připojit až 8 dalších TXT ovladačů jako rozšíření. ROBO Pro Coding lze používat i s mobilními zařízeními (Android / iOS). Pomocí aplikace (Android / iOS) lze ovladač TXT 4.0 ovládat také pomocí rozpoznávání hlasu. [17]

2.2 Použité komponenty modelu Fischertechnik

Model Fischertechnik (obr. 2-3) použitý v této práci je model výrobní linky postavené na KPV/ZČU s využitím setu ROBOTICS. Tato výrobní linka má selektovat materiál pomocí rentgenu a následně určit, jestli se materiál vyřadí (zmetky) nebo se dopraví na balící stanici a do přepravky na hotové výrobky. Model se skládá ze stavebních bloků, z elektronických a mechanických komponent.



Obrázek 2-3 Testovací model Fischertechnik a použité komponenty [Autor]

Použití těchto komponent vyžaduje správné zapojení do pinů. Seznam použitých komponent a jejich zapojení:

- **Mini motor**

Jedná se o jednoduché elektromotory, u kterých lze bez dalších přidaných prvků pouze řídit rychlost otáček a jejich směr. Motory mini a XS se liší pouze velikostí a možnostmi mechanického přichycení ke konstrukci. Zapojujeme je na řídící jednotce (TXT controller) do pinů M(1 - 4). Princip elektromotoru je založený na využití silových účinků magnetického pole. Zjednodušeně lze říci, že je využito vzájemné přitahování a odpuzování dvou

elektromagnetů. Síla a polarita těchto elektromagnetů se řídí velikostí protékajícího elektrického proudu (regulace rychlosti otáček). [18]

Obrázek 2-4 Mini motor Fischertechnik [12]

- **LED**

V tomto případě se jedná o obyčejnou žárovku, tedy nejjednodušší zařízení k přeměně elektrické energie na světlo. Funguje na principu zahřívání tenkého vodiče elektrickým proudem. Žárovka vždy vydává teplé světlo. Na řídící jednotku ji napojujeme na piny M(1 - 4), nebo O(1-8) a zem. [18]

Obrázek 2-5 LED Fischertechnik [12]

- **Fototranzistor**

Fototranzistor je v elektrotechnice typ polovodičové součástky využívající vnitřní fotoelektrický jev. Světelné záření dopadající do kolektorového PN přechodu otevře průchod proudu mezi bází a emitorem. Tato součástka vyhodnocuje, jestli na ni dopadá světlo a podle toho nabývá stavu 1, nebo 0. Lze ji například využít pro automatické spouštění různých událostí programu. Např. svítí-li na fototranzistor světlo, bude automaticky spuštěn motor. Při zapojení je potřeba dbát na to, aby PLUS bylo přivedeno na červeně označený vstup fototranzistoru. Protože slouží jako vstupní zařízení, dává informaci, jestli světlo dopadá/nedopadá (tato informace je následně vyhodnocována). Je potřeba jej zapojovat na řídící jednotce na piny I(1 - 8). [18]

Obrázek 2-6 Fototranzistor Fischertechnik [12]

- **Kompresor**

Vzduchový kompresor je jednoduše řečeno motorová pumpa vzduchu. Větší, popř. složitější pneumatické systémy doplňují vzduchový kompresor vzduchovou nádrží, která slouží jako rezervoár pro udržení tlaku vzduchu např. i při vypnutí kompresoru. Kompresor je možné zapojit jak na 9V+ pin a zem z pinu I8 (pak poběží kompresor neustále), tak i na piny M(1 - 4), kdy lze řídit jeho zapnutí/vypnutí na potřebnou dobu. [18]

Obrázek 2-7 Kompresor Fischertechnik [12]

- **Solenoidový ventil**

Solenoidový ventil je elektromagneticky ovládaný ventil. V tomto případě se jedná o 3/2 cestný ventil. Tzn. že v jedné poloze dochází k průtoku tekutiny a ve druhé poloze k vyčerpání tekutiny. Trojka v označení znamená, že jsou 3 vstupy/výstupy. První dva jsou na obrázku na první pohled patrné, třetí je na zadní straně ventilu překrytý molitanem, který slouží jako tlumič. Ventil se zapojuje stejně jako např. motor do pinů M(1 - 4) na řídící jednotce. [18]

Obrázek 2-8 Solenoidový ventil Fischertechnik [12]

- **Pneumatický válec**

Pneumatický válec je mechanické zařízení sloužící k převodu síly stlačeného vzduchu na mechanický pohyb. V tomto případě se jedná o jednočinný válec, tedy při přivedení vzduchu se pístnice vysune a její vratný pohyb obstará pružina uvnitř válce. [18]

Obrázek 2-9 Pneumatický válec Fischertechnik [12]

3 Řízení a sběr dat s Raspberry Pi

RPi zastává v systému sběru dat unikátní pozici prostředníka mezi vnějším světem a senzory. Ke komunikaci s vnějším světem je vybaveno mnoha periferiemi.:

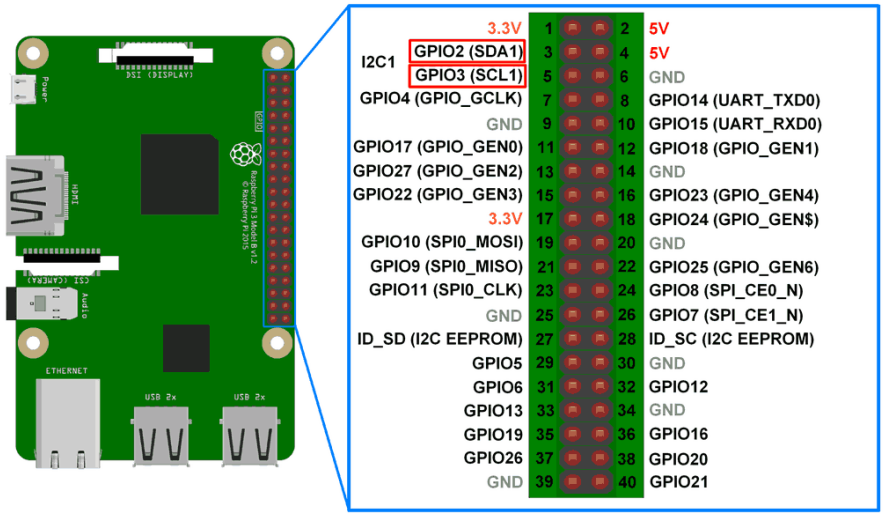
- Ethernet, kterým je možné provádět vzdálený monitoring a řízení systému prostřednictvím sítě Internet z libovolného zařízení.
- HDMI umožňuje připojení monitoru a zobrazení právě měřených dat.
- USB zaručuje možnost lokálního řízení systému při zapojení vstupních zařízení - klávesnice nebo myši.

RPi disponuje i dalšími rozhraními, na která je možné připojit senzory a měřit data. Mezi tato rozhraní patří:

- GPIO (I^2C, SPI, UART) rozhraní, na něž je možné připojit senzor komunikující digitálně.
- CSI rozhraní - usnadňuje využití kamery. [7]

3.1 GPIO piny

Jednou z výkonných funkcí RPi je řada kolíků/pinů GPIO podél horního okraje desky. Tyto piny jsou fyzickým rozhraním mezi RPi a vnějším světem. Na nejjednodušší úrovni si je lze představit jako spínače, které je možné zapnout nebo vypnout. Piny GPIO umožňují RPi ovládat a monitorovat vnější svět připojením k elektronickým obvodům. RPi je schopno ovládat např. LED diody, zapínat a vypínat je, spouštět motory, měřit teplotu, světlo nebo zda byl stisknut spínač. Na RPi model 4B je 40 pinů (26 pinů na prvních modelech) a poskytují různé funkce. Jednotlivé vývody je možné adresovat podle umístnění na desce, např. pin s číslem 7 se označuje jako GPI04. Jednotlivé adresování a k čemu se jednotlivé piny používají je zobrazeno na obr. 3-1 (čísla vedle pinů). K usnadnění ovládání pinů vytvořila komunita kolem RPi několik knihoven v různých programovacích jazycích. V této práci byla použita knihovna RPi.GPIO, která je předinstalována spolu s operačním systémem RPi OS.



Obrázek 3-1 Schéma RPi GPIO pinů [19]

3.2 I²C sběrnice

I²C (inter-integrated circuit) je synchronní sériový protokol původně vyvinutý společností Philips Semiconductors (nyní NXP). Jedná se o multi-master, multi-slave sériovou sběrnici pro nízkorychlostní zařízení, která vyžaduje pouze dva vodiče a používá se pro sériovou datovou komunikaci mezi více zařízeními (Clock signál – SCL, Datový signál – SDA). Master/slave je model asymetrické komunikace nebo řízení, kde jedno zařízení nebo proces ("master") řídí jedno nebo více dalších zařízení nebo procesů ("slave") a slouží jako jejich komunikační hub (rozbočovač). I²C lze snadno implementovat pomocí dvou digitálních vstupních/výstupních kanálů na zařízení. I²C sběrnice má pouze dva vodiče, přes které mohou stovky zařízení posílat sériová data. Jako komunikační standard typu master-slave by alespoň jedno zařízení připojené ke sběrnici mělo být master. [20] Sběrnici I²C spravuje master zařízení a ostatní zařízení jsou slave:

- Master nastavuje clock signál a vybírá slave zařízení, s nímž bude komunikovat.
- Slave jednotky odpovídají master jednotce.
- Každé slave zařízení má jedinečnou adresu zakódovanou v 7 bitech od 0 do 127.

Master je hlavní zařízení, které generuje hodinový signál (clock signal - elektrický signál, jehož změna způsobuje změnu stavu sekvenčního digitálního elektronického obvodu [21]) pro synchronní sériovou komunikaci. Slave zařízení mohou přenášet data do a z hlavního zařízení. Master zařízení přistupují k slave zařízením pomocí svých I²C adres. Adresa každého podřízeného zařízení na sběrnici I²C musí být jedinečná. [20]

Na Raspberry Pi jsou I²C sběrnice přístupné na pinech GPIO2 (Board pin č. 3) a GPIO3 (Board pin č. 5) – viz. obr. 3-1. GPIO2 je linka Serial Data (SDA) a GPIO3 je linka Serial Clock (SCL) I2C1. [19]

3.3 UART

Univerzální asynchronní přijímač/vysílač (transmitter/receiver) neboli UART se v počítačové terminologii obvykle nazývá „sériový port“. RPi není výjimkou. Broadcom SoC na modelech RPi má dva typy vestavěných UART:

1. PL011 UART (sekundární typ)
2. Mini UART (primární typ)

RPi 4 má šest portů UART, z nichž pět je typu PL011 (UART0, UART2 až UART5) a jeden je typu mini-UART (UART1). Z pěti typů PL011 jsou čtyři (UART2 až UART5) ve výchozím nastavení zakázány. Jeden ze dvou UART na GPIO 14 (pin 8 desky) je vysílač a druhý, GPIO15 (pin 10 desky), je přijímač. Obecně je tento primární UART standardně používán konzolou Linux. To znamená, že pokud má být sériový port TTL (GPIO14 a GPIO15) na GPIO hlavičce RPi použit pro sériovou komunikaci se zařízením používajícím protokol UART, je třeba jej nakonfigurovat pomocí raspi-config (možnosti konfigurace RPi).

GPIO14 a GPIO15 jsou označovány jako sériový hardwarový port nebo sériový TTL port RPi. Sekundární UART není na hlavičce GPIO. Ve výchozím nastavení je standartně připojen ke straně Bluetooth kombinovaného ovladače WLAN/Bluetooth. [22]

3.4 SPI

Jedná se o synchronní datovou sběrnici, což znamená, že pro data používá samostatné linky, které udržují obě strany v dokonalé synchronizaci pomocí hodinového signálu. Adresace se provádí pomocí vodičů, které při logické nule aktivují příjem a vysílání zvoleného zařízení (piny SS - Slave Select, nebo CS - Chip Select). Hodiny jsou oscilující signál, který přijímači přesně říká, jak seřadit bity na datové lince. Komunikace SPI je založena na posuvném registru, kde hlavní a více podřízených zařízení získávají data vzájemným posunem registrů.

SPI používá ke komunikaci s cílovým zařízením 4 samostatná připojení. Těmito připojeními jsou sériové hodiny (CLK), Master Input Slave Output (MISO), Master Output Slave Input (MOSI) a Chip Select (CS):

- Master out slave in (MOSI): Master odesílá data do slave
- Master in slave out (MISO): Slave odesílá data do masteru
- Hodiny (CLK): Master a slave musí pracovat se stejnou rychlostí hodin
- Chip select (CS): K jednomu masteru lze připojit více zařízení. Pro přístup ke každému zařízení však musí být CS nejprve povoleno, což zajistí, že všechna ostatní zařízení na sběrnici zůstanou deaktivována.

RPi header SPI (MOSI, MISO a CLK) je přítomen na GPIO 9, 10, 11. SPI CS je přítomen na GPIO 7 a 8. Počet pinů CE (CE0 a CE1) znamená kolik slave zařízení může RPi ovládat. [23]

4 Softwarové řízení a zpracovávání dat

Jako programovací jazyk k programování modelu Fischertechnik a Raspberry Pi byl zvolen program Python. Pro tvorbu a správu databáze byl zvolen jazyk SQL. Pro implementaci konceptů uvedené v kapitole 6 do IoT a předávání zpráv mezi zařízeními byl zvolen protokol MQTT.

4.1 Python

Python je vysokoúrovňový programovací jazyk, který koncem 80. let vznikl v institutu National Research Institute for Mathematics and Computer Science. Autor jazyka Guido van Rossum vycházel z jazyka ABC. Od svého zveřejnění jazyk Python neustále získává na oblibě díky jasné a expresivní syntaxi, při jejímž vývoji byl kladen důraz na čitelnost kódu. [5] Vysokoúrovňový jazyk má vyšší úroveň abstrakce než počítač a zaměřuje se více na logiku programování než na základní hardwarové komponenty, jako je adresování paměti a využití registrů. Python umožňuje velmi jednoduše navrhnout jednoduché programy, ale na druhou stranu nabízí dostatečně mocné prostředky k návrhům rozsáhlých programů. Pro tento jazyk je vyvinuto obrovské množství knihoven a frameworků, které uživatelům umožňují soustředit se na řešený úkol a nerozptylovat se vývojem nejrůznějších pomocných podprogramů. Popularita jazyka Python nepřetržitě roste. Postupně se stává klíčovým jazykem v řadě oblastí, především v těch, které souvisejí s výukou a výzkumem. Je to nejčastěji vyučovaný první jazyk na univerzitách i středních školách. Je nejpoužívanějším jazykem ve statistice, programování umělé inteligence, v aplikacích využívajících strojové učení, v oblasti analýzy dat a postupně proniká do dalších oblastí tvorby softwaru. [24] Python lze rozšířit v C a C++. Python tak může poskytnout rychlost potřebnou i pro výpočetně náročné úlohy. Díky svým silným strukturujícím konstrukcím (vnořené kódové bloky, funkce, třídy, moduly a balíčky) a konzistentnímu používání objektů a objektově orientovaného programování Python umožňuje psát jasné, logické aplikace pro malé i velké úkoly. [25] Python nepředstavuje ideální volbu v každé situaci. Pro jednoduché obvody funguje dobře, ale nemá vlastnost fungování v deterministickém reálném čase. Pro naše účely však tuto vlastnost nepotřebujeme. Pro komplexnější robotické platformy se doporučuje použít jazyk nižší úrovně jako C++ a programy v těchto jazycích spouštět v mikrokontoléru fungujícím v reálném čase. [5]

Jazyk Python je publikován pod licencí open source a je volně dostupný pro operační systémy Linux, iOS i Windows. Díky této multiplatformní podpoře lze software napsaný v jazyku Python využívat i na počítačích s jiným operačním systémem. Python lze také volně upravovat a redistribuovat. [5] Důležité vlastnosti Pythonu:

- Používá elegantní syntaxi, která usnadňuje čtení programů.
- Python je ideální pro vývoj prototypů a další ad-hoc programovací úlohy, aniž by byla ohrožena udržovatelnost.
- Lze ho zabudovat do aplikace a poskytnout programovatelné rozhraní.
- Běží na mnoha různých počítačích a operačních systémech: Windows, MacOS, Linux atd. [26]
- Disonuje vestavěnými datovými typy na vysoké úrovni: řetězce, seznamy, slovníky atd.

- Lze použít obvyklé řídicí struktury: if, if-else, if-elif-else, while plus výkonný iterátor kolekce (for).
- Organizační struktury jsou víceúrovňové: funkce, třídy, moduly a balíčky. Ty pomáhají při organizování kódu. Vynikajícím a velkým příkladem je standardní knihovna Python, která podporuje mnoho běžných programovacích úloh, jako je připojení k webovým serverům, vyhledávání textu pomocí regulárních výrazů, čtení a úpravy souborů, či propojení GPIO pinů.
- Je objektově orientovaný. Python poskytuje konzistentní způsob použití objektů: vše je objekt. A v Pythonu je snadné implementovat nové typy objektů (nazývané třídy v objektově orientovaném programování).
- Snadno se rozšíří přidáním nových modulů implementovaných v kompilovaném jazyce, jako je C nebo C++. Rozšiřující moduly a typy rozšíření lze psát ručně. Existují také nástroje, které s tím pomáhají, například SWIG, sip, Pyrex. [25]

Jazyk Python byl zvolen z důvodů snadného přístupu k portům GPIO, pro snadnou tvorbu uživatelského rozhraní a ukládání dat do SQLite databáze. Od uvedení RPi na trh, bylo vytvořeno mnoho modulů, označovaných jako knihovny, které umožňují plně využít různých funkcí tohoto počítače. Uživatelé tak mohou přistupovat k portu GPIO, aniž by se museli zabývat nízkoúrovňovým programováním. Po instalaci těchto knihoven získá jazyk Python schopnost snadno adresovat port GPIO počítače RPi. [5]

4.2 SQL

SQL (Structured Query Language) je počítačový jazyk pro ukládání, manipulaci a získávání dat uložených v relační databázi. SQL zahrnuje vytváření databáze, mazání, načítání řádků, úpravu řádků atd. Je to standardní jazyk ANSI (American National Standards Institute), ale existuje mnoho různých verzí tohoto jazyka. SQL je standardní jazyk pro relační databázový systém. Všechny systémy správy relačních databází jako MySQL, SQLite, MS Access, Oracle, Sybase, Informix, Postgres a SQL Server používají SQL jako svůj standardní databázový jazyk.

Pro vzdálený přístup k databázi je možné využít LAMP server, který kombinuje Linux (OS), Apache (analytický engine pro zpracování dat), MySQL a PHP. Tento přístup je na bázi komunikace server-klient. [27] K práci s databází a modelem Fischertechnik prostřednictvím RPi však postačí využít lokální databázi SQLite. SQLite je knihovna C, která poskytuje lehkou diskovou databázi a nevyžaduje samostatný serverový proces. Umožňuje přístup k databázi pomocí nestandardní varianty dotazovacího jazyka SQL. Některé aplikace mohou používat SQLite pro interní ukládání dat. [28]

4.3 Protokoly zpráv pro IoT systémy

Pokrok v síťových technologiích výrazně přispěl k tomu, jak zařízení internetu věcí vytvářejí, vyměňují a získavají data. Objem a rychlost generování dat zařízeními internetu věcí rychle roste. To také přispělo k nasazení široké škály protokolů pro zasílání zpráv, které umožnily zařízením IoT efektivnější výměnu zpráv. Protokoly aplikační vrstvy jsou považovány za základní vrstvy používané aplikacemi při definování struktury výměny zpráv a způsobu jejich přenosu. Vzhledem k tomu, že zařízení internetu věcí mají obvykle omezené výpočetní zdroje a výpočetní výkon. Výběr lehkého, spolehlivého, škálovatelného, rozšiřitelného a bezpečného protokolu pro zasílání zpráv se stává velmi náročným úkolem. [29] V důsledku toho není neobvyklé, že systémy IoT mohou využívat více protokolů pro zasílání zpráv pro podporu heterogenity zařízení a různých vzorců výměny zpráv. Připojení zařízení IoT k síti, internetu nebo dokonce k sobě navzájem může probíhat několika způsoby. Zařízení IoT mohou být připojena přes Wifi, mobilní síť, Bluetooth, ZigBee, LoRaWAN nebo jinou metodou připojení. Jakmile jsou součásti řešení IoT vzájemně propojeny, k přenosu telemetrie zařízení (tzv. zpráv) do zařízení a ze zařízení je použit protokol pro zasílání zpráv. [30]

4.3.1 INTERNET VĚCÍ

Internetu věcí je kombinace fyzických a digitálních prvků za účelem vytvoření nových produktů a umožnění nových aplikačních modelů. Je to síť fyzických zařízení, vozidel, domácích spotřebičů a dalších zařízení, která jsou vybavena elektronikou, softwarem, senzory/čidly a hlavně síťovou konektivitou. Ta umožňuje těmto zařízením se navzájem propojit a vyměňovat si data. Díky stále efektivnějšímu řízení spotřeby energie, širokopásmové komunikaci, spolehlivé paměti a pokroku v mikroprocesorových technologiích je možné digitalizovat funkce a klíčové schopnosti výrobků průmyslového charakteru. [31] IoT se charakterizuje jako síťový model, který zaplní propast mezi kybernetickým a fyzickým světem. Základním konceptem internetu věcí je připojení všudypřítomných objektů kolem nás, jako jsou tagy RFID (Radio Frequency Identification), mobilní zařízení, senzory a akční členy, k internetu prostřednictvím kabelové sítě nebo bezdrátové sítě. Umožňuje tedy interakci mezi objekty mezi sebou a se svými sousedy, aby se zvýšila efektivita systému. Internet věcí vytváří nové možnosti pro vzájemnou komunikaci zařízení a to i na globální úrovni. [32] V roce 2022 se odhaduje že je celkově připojeno 46 miliard zařízení do internetu věcí. [33] Pro propojení zařízení se nejčastěji používají tyto bezdrátové technologie:

- Sítě LPWAN - poskytují komunikaci na velké vzdálenosti, při nízké přenosové rychlosti a s nízkou spotřebou energie. Jejich cílem je podpora rozsáhlých sítí internetu věcí na velkých lokalitách.
- Mobilní (3G/4G/5G) - Zatímco pro většinu aplikací internetu věcí, které jsou napájeny z bateriových sítí senzorů, nejsou mobilní sítě použitelné, hodí se pro specifické případy použití, jako jsou propojená auta nebo správa vozového parku v dopravě a logistice. Například infotainment v automobilech, směrování dopravy, pokročilé asistenční systémy řidiče (ADAS) spolu s telematickými službami a službami sledování vozového parku se mohou opírat o všudypřítomné mobilní připojení s vysokou šířkou pásma.
- Zigbee a další mesh protokoly - bezdrátový standard s krátkým dosahem a nízkou spotřebou energie využívající data ze senzorů a uzlů. Může se pochlubit vysokou rychlostí přenosu dat, ale není energeticky úsporný jako LPWAN.

- Bluetooth a BLE - komunikační technologie s krátkým dosahem, která umožňuje výměnu dat mezi více uzly. Pro malé spotřebitelské aplikace IoT byla vynalezena verze s nízkou spotřebou energie.
- Wi-Fi - v oblasti IoT není tak rozšířená kvůli problémům se škálovatelností, pokrytím a spotřebou energie.
- RFID - známá jako radiofrekvenční identifikace, rádiové vlny přenášejí data ze štítku RFID do čtečky na krátkou vzdálenost. Běžně se používá v logistice a maloobchodě.[33] [34]

4.3.2 IoT V PRŮMYSLU

Řešení internetu věcí lze použít v mnoha oblastech a prostředích, proto je lze rozdělit do těchto základních odvětví:

- Oblast životního prostředí: Zahrnuje aplikace, které chrání, monitorují a rozvíjejí všechny přírodní zdroje. Služby v oblasti životního prostředí, hospodaření s energií, recyklace, zemědělství atd.
- Průmyslová oblast: V této doméně se jedná o aplikace, které zahrnují finanční nebo obchodní transakce, mezi podniky, organizacemi a dalšími subjekty. Kromě toho se týkají výroby, logistiky, bankovnictví, zdravotnictví, finanční státní orgány atd.
- Sociální oblast: Tato oblast zahrnuje aplikace týkající se rozvoje a začlenění společnosti, měst a lidí, jakož i vládních služeb vůči občanům a dalším společenským strukturám. [35]

Specifická kategorie internetu věcí se zaměřuje na jeho aplikace a případy použití v moderních průmyslových odvětvích a inteligentních technologiích. Považuje se za komplexní systém spojením různých systémů. Kromě toho zahrnuje klíčovou složku průmyslové oblasti a je úzce spjat se čtvrtou průmyslovou revolucí (Industry 4.0). Spojuje v sobě několik inovativních klíčových technologií tak, aby vznikl systém, který funguje efektivněji než součet jeho částí. Tato specifická oblast se vyznačuje různorodými inovativními aplikacemi a službami, různorodými propojenými zařízeními i novými výrobními operacemi. [35] Cílem Průmyslu 4.0 je vylepšit a modernizovat stávající výrobní zařízení, systémy a technologie řízení a údržby na inteligentní úroveň s využitím klíčových technologií jako jsou internet věcí, internet služeb (IoS), CPS, autonomní, flexibilní a kooperativní robotika, simulace, které využívají data v reálném čase a zrcadlí reálný svět do virtuálního modelu, analýza velkých dat, rozšířená realita (AR), aditivní výroba, informační a komunikační technologie a pokročilé síťové technologie (např. cloud computing atd.). V kontextu Průmyslu 4.0 se sběr, analýza a pochopení dat z mnoha různých zdrojů (data z výrobních systémů a zařízení, zdroje týkající se zákazníků a podnikového řízení) stává normou, která podporuje rozhodování v reálném čase. [35][36] Hlavní součástí těchto systémů je výměna dat mezi jednotlivými zařízeními. Proto vznikla řada protokolů které tuto výměnu spravují a řídí.

4.3.3 PROTOKOLY ZPRÁV

Zjednodušeně řečeno, protokoly pro zasílání zpráv definují strukturovaný způsob, jak si aplikace mezi sebou vyměňují užitečné datové soubory. Popisují také, jak má implementace zpracovávat zprávy, určovat jejich prioritu a směrovat data mezi producenty a konzumenty. Přestože existuje značný počet iniciativ nebo standardů, které se pokoušejí řešit problémy internetu věcí v různých doménách, značně se liší z hlediska následujících faktorů:

- architektura,

- komunikace,
- bezpečnost
- interoperabilita,
- integrace,
- typy zařízení a technologie senzorů,
- modely nasazení,
- poskytování služeb
- správa aplikací a zařízení. [37]

Dle [37] lze identifikovat čtyři klasické typy komunikace pro prostředí IoT. Tyto typy komunikace jsou:

- Device-to-Device (D2D), kde je komunikace poskytována mezi dvěma uzly nebo zařízeními přímo.
- Device-to-Application (D2A), kde probíhá komunikace mezi zařízeními a aplikací IoT.
- Device-to-Gateway (D2G), kde je komunikace poskytována prostřednictvím brány (gateway), která je umístěna v síti při interakci se zařízeními IoT.
- Device-to-Cloud (D2C), kde komunikace probíhá přímo mezi zařízeními IoT a poskytovateli cloudových služeb. [37]

Ne všechny protokoly podporují všechny identifikované typy komunikace. Například MQTT funguje v komunikačním typu D2C, zatímco CoAP (Constrained Application Protocol) podporuje pouze D2D. Podpora rozsahu komunikace navíc závisí na řadě faktorů, jako jsou možnosti zařízení. Aby zařízení IoT posílalo datové toky přímo do cloudové platformy IoT, musí být zařízení vybaveno síťovou technologií (např. WiFi, Ethernet, mobilní data) pro odesílání dat do cloudu. Pokud však zařízení nepodporuje přímou konektivitu ke cloudu (např. RFID tag), není možné provádět D2C komunikaci. Bránu lze tedy použít ke shromažďování datových toků ze zařízení IoT a následnému předávání toku do cloudu. V případě této práce je touto bránou Rpi.

Mezi nejpoužívanější přijímané a nově vznikající protokoly zasílání zpráv pro systémy IoT jsou MQTT (viz. Kapitola 4.4) , CoAP, AMQP a HTTP. [38]

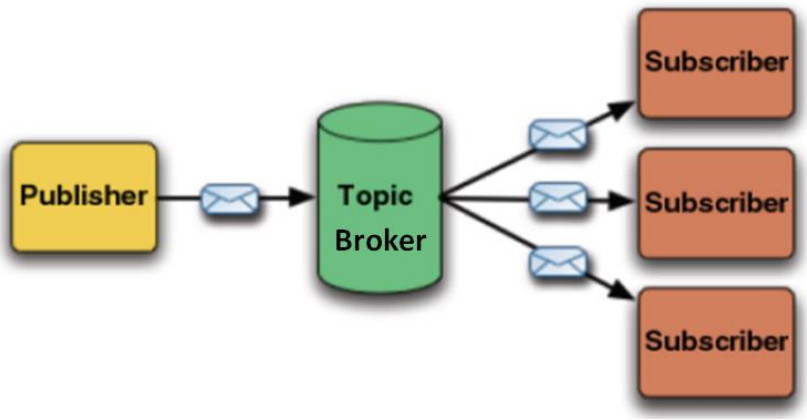
- CoAP (Constrained Application Protocol) je odlehčený protokol M2M od skupiny IETF CoRE (Constrained RESTful Environments). CoAP podporuje architekturu request/response i resource/observe (varianta publish/subscribe). CoAP je vyvinut především pro spolupráci s HTTP a webem RESTful prostřednictvím jednoduchých proxy serverů. Na rozdíl od MQTT používá CoAP místo témat univerzální identifikátor zdrojů (URI). Vydavatel publikuje data na URI a odběratel se přihlásí k odběru konkrétního zdroje označeného URI. Když vydavatel publikuje nová data do URI, jsou všichni odběratelé informováni o nové hodnotě uvedené v URI. CoAP je binární protokol a obvykle vyžaduje pevnou hlavičku o velikosti 4 bajtů s malým nákladem zprávy až do maximální velikosti závislé na webovém serveru nebo programovací technologii. CoAP používá UDP jako transportní protokol a DTLS pro zabezpečení. Klienti a servery tedy komunikují prostřednictvím datagramů bez spojení s menší spolehlivostí. Používá však "potvrzující" nebo "nepotvrzující" zprávy, které poskytují dvě různé úrovně QoS (Quality of service). Potvrzující zprávy musí příjemce potvrdit paketem ACK a nepotvrzující zprávy nikoliv. [38][39]

- HTTP je převážně webový protokol pro zasílání zpráv. Standartní protokol společně vyvinuly IETF a W3C v roce 1997. Protokol HTTP podporuje webovou architekturu RESTful (request/response). HTTP používá stejně jako CoAP univerzální indetifikátory (URI). Server odesílá data prostřednictvím URI a klient přijímá data prostřednictvím konkrétního URI. HTTP je textový protokol a nedefinuje velikost záhlaví a užitečného zatížení zpráv, spíše závisí na webovém serveru nebo programovací technologii. Protokol HTTP používá jako výchozí transportní protokol TCP a pro zabezpečení TLS/SSL. Komunikace mezi klientem a serverem je tedy orientovaná na spojení. HTTP je celosvětově uznávaný standard pro webové zprávy, který nabízí několik funkcí, jako jsou trvalá spojení, pipelining požadavků, či kódování přenosu po částech. [38]

- AMQP (Advanced Message Queuing Protocol) je protokol, který využívá vysokoúrovňové fronty zpráv a který začal v roce 2003 vyvíjet John O'Hara ve společnosti JPMorgan Chase v Londýně. AMQP vykazuje vynikající vlastnosti díky frontám zpráv, distribuci multifunkčních zpráv do front v systému a dobrému zabezpečení. Oblíbeným zprostředkovatelem AMQP je RabbitMQ. RabbitMQ má výkonný cloudový systém, pohodlné a podrobné rozhraní a server RabbitMQ lze snadno nainstalovat do systémů Windows, Macos nebo Linux. Protokol AMQP lze použít ke spolehlivému přenosu dat mezi zařízeními a softwarovými aplikacemi. [40]

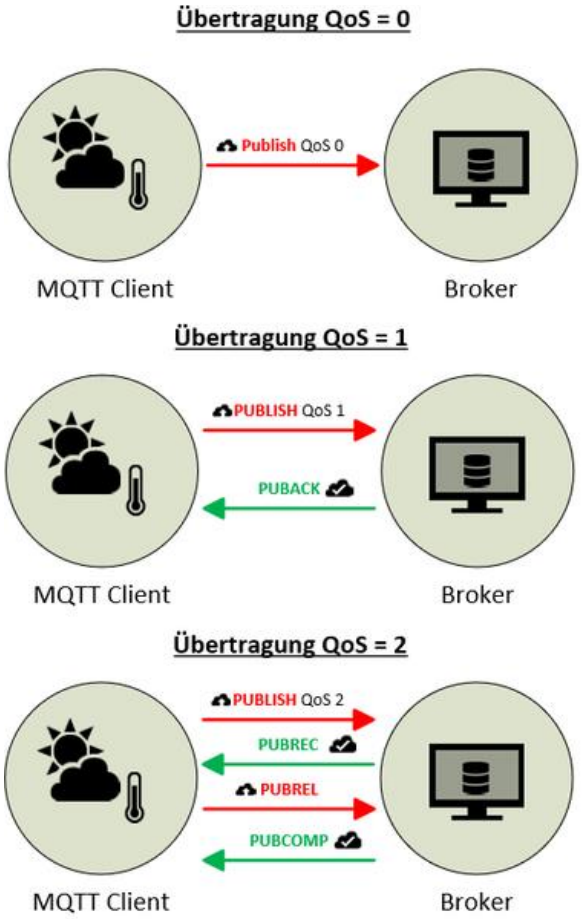
4.4 Protokol MQTT

MQTT (dříve: Message Queuing Telemetry Transport, dnes MQ Telemetry Transport) je jednoduchý a nenáročný protokol pro předávání zpráv mezi klienty prostřednictvím centrálního bodu – brokeru. Díky této nenáročnosti a jednoduchosti je snadno implementovatelný i do zařízení s „malými“ procesory a poměrně rychle se rozšířil. Navržen byl v IBM, dnes za ním stojí Eclipse foundation a před nedávnem proběhla standardizace OASIS. U protokolu MQTT probíhá přenos pomocí TCP a používá návrhový vzor publisher – subscriber. Tedy existuje jeden centrální bod (MQTT broker), který se stará o výměnu zpráv. Zprávy jsou tříděny do tzv. témat (topic) a zařízení buď publikuje v daném tématu (publish), to znamená, že posílá data brokeru a ten je ukládá a distribuuje dalším zařízením (subscribes) nebo je přihlášeno k odběru tématu či témat a broker pak všechny zprávy s daným tématem posílá do zařízení (viz. obr. 4-1). Jedno zařízení může být najednou v některých tématech publisher, v jiných subscriber. [41]



Obrázek 4-1 Spravování dat prostřednictvím MQTT Brokera [42]

MQTT minimalizuje množství balastních dat a předává jen minimum servisních dat. MQTT byl určen pro nestabilní připojení, a proto existují tři různé kvality služeb, Quality of Service nebo QoS. [41] Na obr. 4-2 je schematicky znázorněno, jak vypadá tok zpráv se všemi třemi variantami QoS.



Obrázek 4-2 MQTT schéma toku zpráv QoS [41]

Při úrovni 1 nebo QoS = 0 je zpráva odeslána právě jednou. Potvrzení, jako je tomu u ostatních QoS, je zcela ignorováno. Důležité je pouze publikování, tzv. publish. [41]

Při QoS = 1 se také mluví o „doručeno alespoň jednou“, což znamená, že odesílatel čeká na potvrzení od příjemce. Toto potvrzení se nazývá PUBACK a je povinné pro vzdálenou stanici, jinak odesílatel vysílá, dokud není přijato potvrzení. To vede obráceně k tomu, že zpráva je příjemci přenášena několikrát. [41]

QoS = 2 je kombinací QoS = 0 a QoS = 1, přičemž také představuje nejpomalejší přenos. Zde je zpráva odeslána pouze jednou a je nutné dvoufázové potvrzení o přijetí. [41]

Odesílatel nejprve odešle (PUBLISH) svou zprávu příjemci, který zprávu přijme a odešle potvrzovací zprávu (PUBREC) s přiloženým obsahem od odesílatele jako kopii. Pokud odesílatel obdrží tuto potvrzovací zprávu (PUBREC), uloží informace a také odpoví potvrzovací zprávou (PUBREL). Nakonec, když příjemce obdrží PUBREL, odešle poslední potvrzovací zprávu (PUBCOMP). Na konci takového přenosu dat je zajištěno, že zpráva skutečně dorazila. [41]

V IoT (Internet of things) systémech by snímače neměly určovat, co kdo má udělat. Snímače snímají a posílají zprávy do centrálního místa. O víc se nestarají. Ti, co mají zájem o jejich zprávy, se přihlásí k jejich odběru a nějak na ně reagují. V ideální podobě se přidává i další vrstva abstrakce, kde např. žárovka reaguje na pokyn „vypnout / zapnout“, vypínač informuje o tom, že byl stisknut, a někdo třetí čte zprávy od vypínače a nějak na ně reaguje. Tímto způsobem lze vytvářet interaktivní komplexní systémy. Pro tyto aplikace je protokol MQTT ideální. [43] MQTT v praktické části slouží jako snadný způsob pro získávání dat a ovládání RPi prostřednictvím WiFi sítě. Na RPi byl vytvořen server (Broker), který spravuje veškeré zprávy publikované ostatními zařízeními a dále je odesílá určeným odběratelům. Existuje několik webových stránek, které nabízejí konfigurované Broker servery, které je možné využívat pro správu odeslaných dat. Tyto služby poskytují i nejrůznější aplikace pro snadné připojení, odesílání zpráv a vizualizaci dat. Některé služby jsou poskytovány zdarma, jako například Eclipse Mosquitt open source MQTT broker nebo HiveMQ Public MQTT Broker.

Výhody MQTT:

- efektivní přenos dat a rychlá implementace – jedná se o odlehčený protokol.
- nízká spotřeba sítě díky minimalizaci datových paketů.
- efektivní distribuce dat.
- rychlé a účinné doručování zpráv.
- spotřebovává malé množství energie, což je výhodné pro přenosná zařízení.
- optimalizuje šířku pásma sítě. [44][45]

Nevýhody MQTT:

- MQTT má ve srovnání s protokolem CoAP (Constrained Application Protocol) pomalejší přenosové cykly.
- Zjišťování prostředků MQTT funguje na základě flexibilního odběru témat, zatímco CoAP používá stabilní systém zjišťování prostředků.
- Protokol MQTT není šifrovaný. Místo toho používá k bezpečnostnímu šifrování TLS/SSL (Transport Layer Security/Secure Sockets Layer).
- Je obtížné vytvořit globálně škálovatelnou síť MQTT.
- Další problémy MQTT se týkají zabezpečení, interoperability a ověřování. [45]

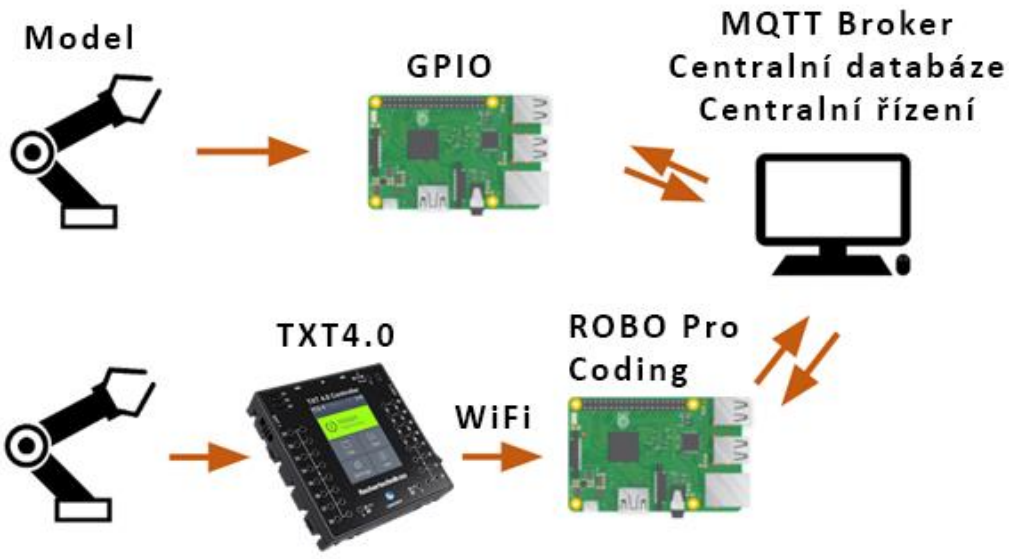
Protože protokol MQTT nebyl navržen s ohledem na bezpečnost, tradičně se používá v zabezpečených koncových sítích pro specifické aplikační účely. Struktura témat protokolu MQTT může snadno vytvořit obrovskou nepřehlednou strukturu a neexistuje jasný způsob, jak jí rozdělit na menší logické domény, které by bylo možné sdružit. To ztěžuje vytvoření globálně škálovatelné sítě MQTT, protože s rostoucí velikostí struktury témat roste i jeho složitost. Dalším negativním aspektem MQTT je jeho nedostatečná interoperabilita. Vzhledem k tomu, že zprávy jsou binární data, bez informací o tom, jak jsou zakódována, mohou vznikat problémy - zejména v otevřených architekturách, kde mají různé aplikace od různých výrobců vzájemně bezproblémově spolupracovat. [45]

Pro správu dat a řízení pomocí protokolu MQTT existuje i mnoho mobilních aplikací. MQTT pracuje na TCP/IP stacku, to znamená, že jediným požadavkem na mobilní zařízení je možnost připojení k internetu. [42] Nejrozšířenější aplikace pro mobilní zařízení je MQTT Dash (IoT, Smart Home). Aplikace umožňuje tvorbu prostředí pro řízení zařízení v IoT propojené pomocí MQTT, jako je například chytrá domácnost, další mobilní aplikace, či průmyslové zařízení. [47]

5 Praktická část

Cíl této práce je naprogramovat, řídit a sbírat data z modelu Fischertechnik prostřednictvím RPi. Dalším požadavkem je, aby koncept umožňoval ukládání dat do centrální databáze a aby bylo umožněno vzdálené centrální řízení modelu. Pro výměnu informací byl zvolen protokol MQTT (viz. kapitola 4.4). Tento síťový protokol umožňuje řídit a přenášet data z více modelů najednou, je tak možné snadno vytvořit IoT a je skvělým nástrojem pro průmysl 4.0. MQTT server, který se stará o distribuci dat byl nainstalován na RPi. Veškerá data se ukládají do lokální SQLite databáze umístěné na centrálním zařízení. Jako centrální zařízení byl použit počítač s operačním systémem Windows. Na tomto zařízení byla vytvořena lokální SQLite databáze a program pro koncového uživatele pro ovládání modelu. Veškerý kód je psaný v programovacím jazyce Python. Tyto programy lze spustit po nainstalování požadovaných knihoven na kterémkoli zařízení s operačním systémem Windows nebo Linux. Je tak možné využít další RPi pro ukládání dat a řízení modelu. Byly vytvořeny dva koncepty možnosti sběru dat z Fishertechnik modelu:

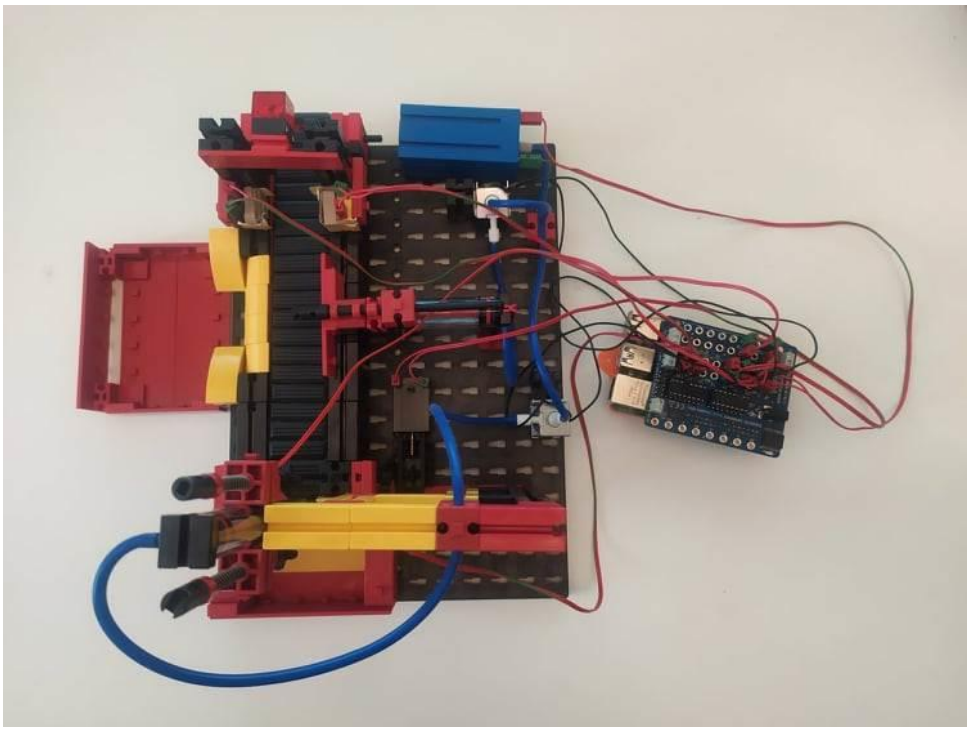
- Koncept 1 využívá propojení mezi Fishertechnik a RPi GPIO piny a napájecí Raspberry Pi F5 adaptér. Jednotlivé piny jsou řízeny pomocí programu Python. Pomocí protoklu MQTT se data posílají z tohoto propojení do centrálního počítače, kde se ukládají do SQLite databáze. Z centrálního počítače se řídí model a jeho jednotlivé komponenty.
- Koncept 2 využívá Fishertechnik Controller TXT4.0 a vývojové prostředí ROBO Pro Coding. Sběr dat a řízení modelu je řešeno stejně jako v konceptu 1 pomocí protokolu MQTT.



Obrázek 5-1 Schéma propojení konceptů [Autor]

5.1 Testovací model

Model Fischertechnik (obr. 5-2) je část modelu výrobní linky postavené na KPV/ZČU s využitím setu ROBOTICS. Tato výrobní linka má selektovat materiál pomocí rentgenu a následně určit, jestli se materiál vyřadí (zmetek) nebo se dopraví na balící stanici a do přepravky na hotové výrobky.

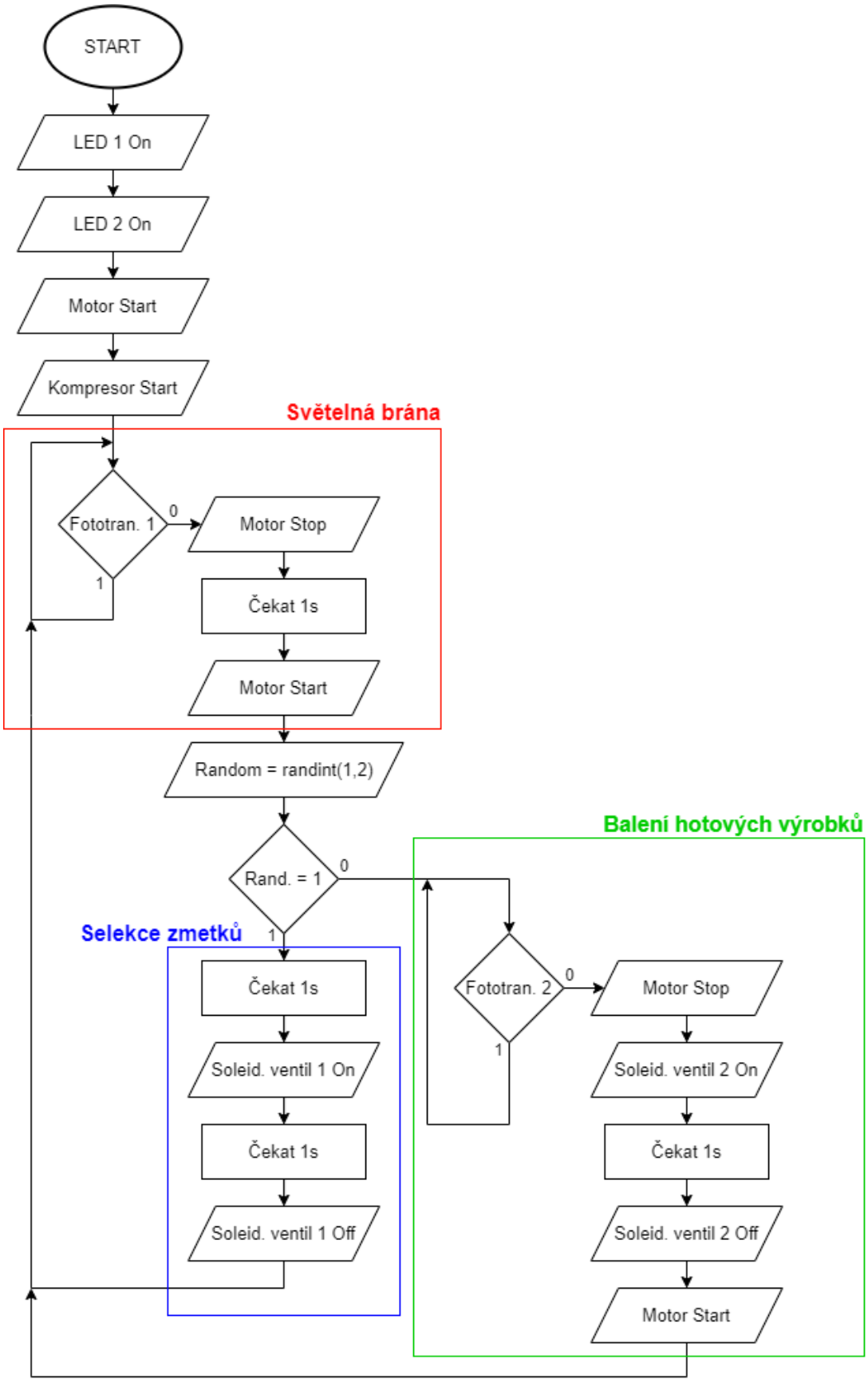


Obrázek 5-2 Testovací model Fischertechnik [Autor]

Materiál se pohybuje po dopravním pásu, který je poháněn minimotorem. Rentgen je simulován světelnou bránou (fototranzistorem a LED žárovkou). Koncept funguje tak, že zastínění LED žárovky materiálem před fototranzistorem změní jeho hodnotu z 1 na 0. Simulace rentgenování probíhá pouze zastavením pásu na určitou dobu. Vyřazení zmetků je vyhodnocováno náhodně pomocí generování náhodného čísla (1 nebo 2) a je provedeno pomocí pneumatického válce, který je poháněn kompresorem a solenoidovým ventilem. Válec vytlačí zmetky do postranní přihrádky. Poslední část modelu je stanice, která simuluje balení materiálu. Tento úsek disponuje světelnou bránou, jež zaznamená přítomnost výrobku a zastavý pás. Simulace balení probíhá dalším pneumatickým válcem, který stlačí hlavu nástroje směrem dolů. Následně se materiál pomocí pásu dopraví do přihrádky na hotové výrobky. Přehled částí a komponent, z kterých se skládají:

- Dopravní pás (mini motor)
- Světelná brána - Rentgen (fototranzistor, LED žárovka)
- Odsun zmetků (kompresor, solenoidový ventil, pneumatický válec)
- Světelná brána (fototranzistor, LED žárovka)
- Balení materiálu (solenoidový ventil, pneumatický válec)

Použití těchto komponent vyžaduje správné zapojení do pinů. Použité komponenty a jejich zapojení je popsáno v kapitole 2.1. Vývojový diagram celého programu pro řízení modelu je zobrazen na obrázku 5-3.



Obrázek 5-3 Vývojový diagram programu pro řízení modelu

5.2 Implementace MQTT

Pro implementaci obou konceptů do IoT byl použit pro vzdálené propojení centrálního PC a ostatních zařízení protokol MQTT. Z tohoto PC je možné získávat data a komunikovat s RPi. Na PC byl použit operační systém Windows. Pro podporu protokolu MQTT, byl použit serverový software s názvem Mosquitto [48]. Mosquitto je zprostředkovatel zpráv, který implementuje několik verzí protokolu MQTT, včetně nejnovější revize 5.0. Je to také relativně výpočetně nenáročný software, díky čemuž je Mosquitto perfektní volbou pro práci s protokolem MQTT na RPi.

Protokol MQTT funguje tak, že klienti vystupují jako publishers (vydavatelé) a subscribers (předplatitelé). Publishers zasílají zprávy subscribers, který slouží jako prostředník. Subscribers se připojují k zprostředkovateli MQTT a čtou zprávy, které jsou vysílány v rámci určitého tématu. [49]

Aby bylo možné používat MQTT, je potřeba server MQTT, takzvaný broker, a také odpovídající koncová zařízení, která odesílají nebo přijímají data, nazývaná také klienty. Jako broker bylo zvoleno RPi.

Jak bylo zmíněno v kapitole 4.3, výměna zpráv může probíhat ve třech variantách QoS = 0, QoS = 1, QoS = 2. K našim účelům byla zvolena kvality služeb QoS = 0.

5.2.1 MQTT SERVER PRO RPI

Broker MQTT je zodpovědný za příjem všech zpráv, filtrování zpráv, rozhodování o tom, kdo o ně má zájem, a následné zveřejnění zprávy všem přihlášeným klientům k určitému tématu. Broker může být ktérékoli zařízení v sítí, musí však být neustále zapnuté pro udržení MQTT spojení mezi zařízeními.

Broker Mosquitto MQTT je k dispozici pro RPi OS jako součást úložiště APT (Advanced Packaging Tool), takže instalace softwaru je jednoduchá. Mosquitto spolu s klientským softwarem byl nainstalován následujícíim příkazem:

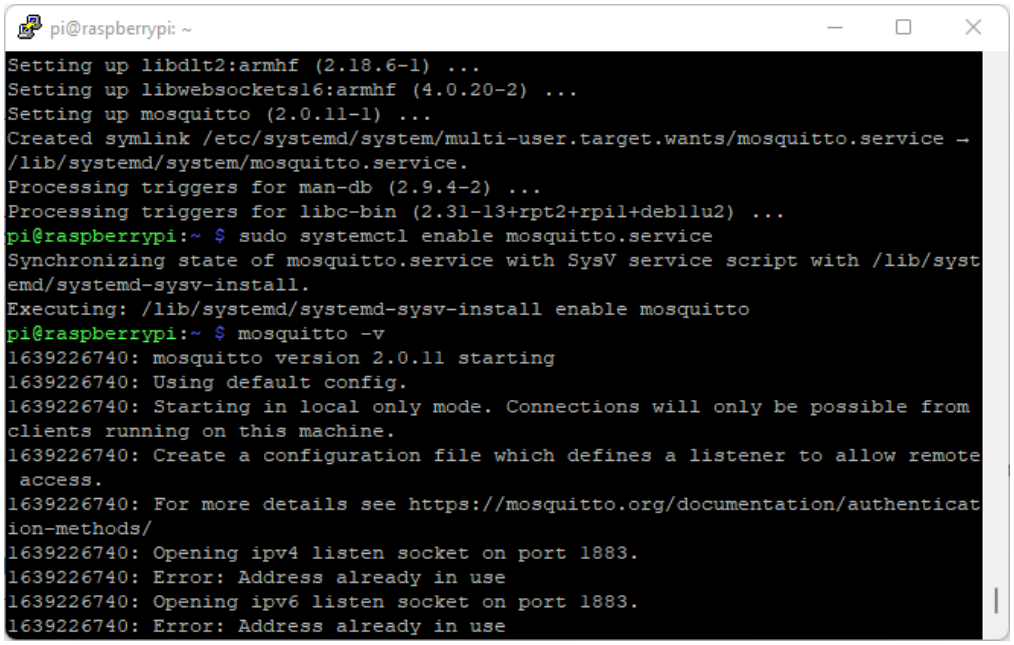
```
pi@raspberry:~ $ sudo apt install -y mosquitto mosquitto-
clients
```

Aby se Mosquitto server automaticky spustil při spuštění Raspberry Pi, je nutné spustit následující příkaz (to znamená, že zprostředkovatel Mosquitto se automaticky spustí při spuštění Raspberry Pi):

```
pi@raspberry:~ $ sudo systemctl enable mosquitto.service
```

Instalaci je možné otestovat spuštěním následujícího příkazu:

```
pi@raspberry:~ $ mosquitto –v
```



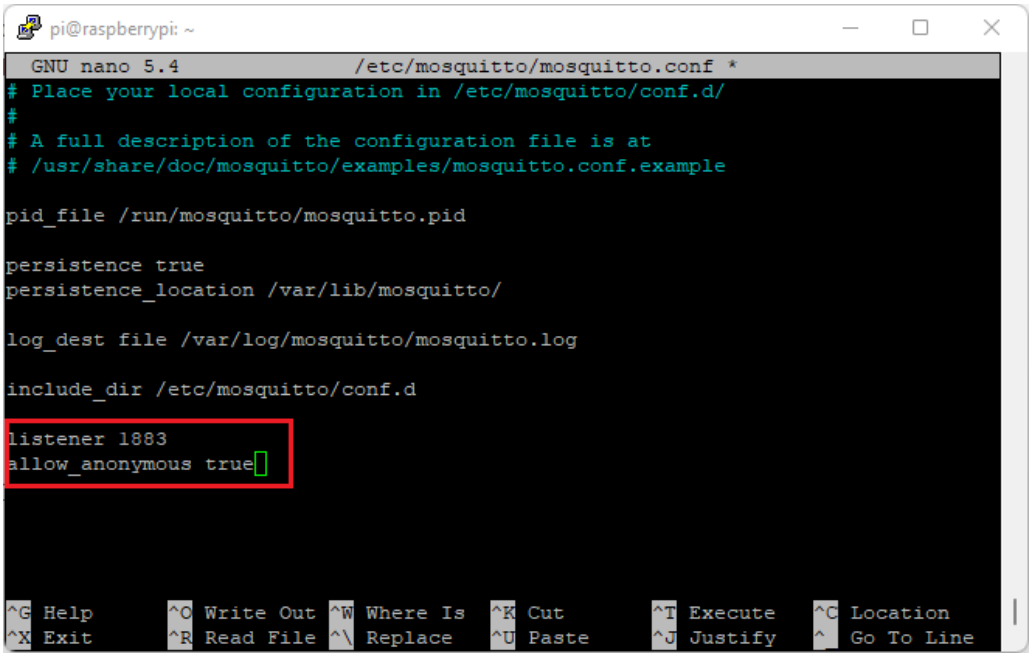
Obrázek 5-4 Test správného naistalování MQTT brokera [Autor]

Zobrazí se následující zpráva: “Starting in local only mode. Connections will only be possible from clients running on this machine. Create a configuration file which defines a listener to allow remote access.” – “Spuštění pouze v místním režimu. Připojení bude možné pouze z klientů běžících na tomto počítači. Vytvořte konfigurační soubor, který definuje posluchače umožňující vzdálený přístup.“

Aby byl povolený vzdálený přístup, a aby bylo možné komunikovat s ostatními zařízeními v síti, byl upraven/vytvořen konfigurační soubor. Spuštěním následujícího příkazu byl otevřen soubor mosquitto.conf:

```
pi@raspberry:~$ sudo nano /etc/mosquitto/mosquitto.conf
```

Do konfiguračního souboru byly přidány dva řádky – listener 1883 a allow_anonymous true.



Obrázek 5-5 Nastavení konfiguračního souboru [Autor]

Příkazem listener se stanovuje kdo se může na daný server připojit. 1883 znamená že k připojení k brokeru není potřeba ověření a šifrování. Toto je výchozí port MQTT. Pro šifrované připojení je možné použít port 8883: Toto je výchozí port MQTT pro MQTT přes TLS (Transport Layer Security). Pokud jsou známy adresy IP klientů MQTT, lze povolit přístup pouze pro potřebné rozsahy IP. Toto opatření zablokuje všechny klienty, kteří nejsou v definovaných rozsazích IP adres. Jelikož tato práce je ukázková demonstrace použití MQTT, šifrováním a zabezpečením se nezabývá. Příkaz allow_anonymous true umožňuje připojení neznámých zařízení.

Pro připojení k broker serveru stačí IP adresa zařízení, na kterém je broker nainstalován. Konkrétně k zjištění IP adressy na RPi, lze použít příkaz:

```
pi@raspberry:~$ hostname –I
```

5.2.2 MQTT TOPICS

Témata MQTT jsou formou adresování, která umožňuje klientům MQTT sdílet informace. Témata MQTT jsou strukturována v hierarchii podobné složkám a souborům v systému souborů pomocí lomítka (/) jako oddělovače (např. device/lamp/status). Pomocí tohoto systému je možné vytvořit uživatelsky přívětivé a samopopisné pojmenovací struktury dle vlastního výběru. Pro správné adresování musíme dodržet následující

- Dodržovat velká a malá písmena
- Používat řetězce UTF-8.
- Aby adresa byla platná, musí obsahovat alespoň jeden znak. [50]

V topics existují speciální znaky, které se umožňují přihlásit k odběru více témat současně. Symbol # představuje víceúrovňový zástupný znak v tématu. Aby broker určil, která témata se shodují, musí být víceúrovňový zástupný znak umístěn jako poslední znak v tématu a předcházet mu lomítko (např. device/lamp/#). Jednoúrovňový zástupný znak nahrazuje jednu úroveň tématu. Symbol + představuje jednoúrovňový zástupný znak (např device/+/status). [51]

5.2.3 MQTT PYTHON

Tato kapitola popisuje zdrojový kód klientské knihovny Eclipse Paho MQTT Python. Jsou zde vysvětleny základní funkce pro posílání, obdržení zpráv prostřednictvím MQTT a vytvoření spojení pomocí programovacího jazyka Python. Všechny poznatky uvedené v této kapitole jsou převzaty z [59]. Nejdříve bylo nutné nainstalovat potřebné knihovny. Knihovny pro OS Windows je možné nainstalovat pomocí PyPi, pro RPi OS pomocí APT. Python Package Index (PyPI) je úložiště softwaru pro programovací jazyk Python. PyPi umožňuje najít a nainstalovat software vyvinutý a sdílený komunitou Pythonu. Tento balíček je součástí instalačního programu jazyka Python. Pomocí PyPi byl nainstalován Eclipse Paho MQTT. Tento kód poskytuje třídu klienta, která umožňuje aplikacím připojit se k zprostředkovateli MQTT, publikovat zprávy a přihlásit se k odběru témat a přijímat publikované zprávy. Poskytuje také některé pomocné funkce, díky nimž je publikování jednorázových zpráv na server MQTT velmi jednoduché. [52] Pro instalaci byl použit příkaz:

```
pip3 install paho-mqtt
```

MQTT rozděluje zařízení na publisher a client. Zařízení může být zároveň client a publisher. Pro přístup ke všem funkcím potřebným pro připojení k brokerovi je potřeba naimportovat třídu klienta.

```
import paho.mqtt.client as mqtt
```

Pro publikování zpráv (publish) slouží modul paho.mqtt.publish:

```
import paho.mqtt.publish as publish
```

Pro připojení k serveru stačí využít následující funkci, zadat IP adresu a číslo portu.

```
client.connect("192.168.0.127", 1883, 60)
```

Dále byla vytvořena funkce zpětného volání (callback function) *on_connect*, která se volá, když broker odpoví na žádost o připojení.

```
client.on_connect = on_connect
```

Obdobně funkce on_message se volá, když program dostane zprávu od brokera.

```
client.on_message = on_message
```

funkce *loop*()* slouží pro udržení toku síťového provozu s brokerem. Pokud nejsou volány, příchozí síťová data nebudou zpracována a odchozí síťová data nemusí být odeslána včas. Existují čtyři možnosti správy síťového cyklu. *Forever* je síťový cyklus a nevrátí se, dokud klient nezavolá *disconnect()*. Automaticky se stará o opětovné připojení.

```
client.loop_forever()
```

Aby bylo možné kontrolovat stav připojení k brokerovi, byla deklarována funkce *on_connect*, která určuje jaké téma program odebírá. Pro odběr určitého tématu byla použita funkce *client.subscribe("téma")*:

```
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("Backmsg")
```

Funkce *on_message* určuje co se stane při obdržení zprávy. V tomto případě vypíše obdrženou zprávu. Msg je objekt a vlastnost payload obsahuje data zprávy, což jsou binární data. Pomocí této funkce je možné spustit příkazem jednotlivé komponenty fishertechnik, popřípadě uložit obdrženou zprávu do databáze.

```
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
```

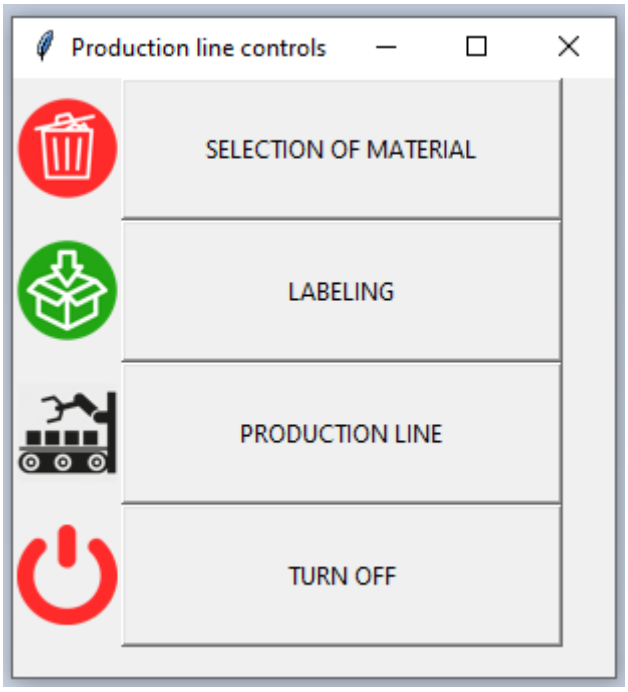
Pro odeslání zprávy brokerovi a následně od brokera všem klientům, kteří se přihlásí k odběru odpovídajících témat je nutné založit funkci publish. V této funkci můžeme určit o jakou kvalitu služeb se jedná (QoS). QoS je defaultně nastavené na 0. Tento modul poskytuje několik pomocných funkcí, které umožňují přímočaré jednorázové publikování zpráv. Jsou užitečné v situaci, kdy je žádoucí publikovat zprostředkovateli zprávu, a pak se od něj odpojit, aniž by bylo potřeba cokoli dalšího. K dispozici jsou dvě funkce: *single()* - publikuje jedinou zprávu zprostředkovateli, *multiple()* – publikuje více zpráv najednou. Jako příklad je uvedena funkce *publish.single*, která je připojena na téma *"Control"* a odešle zprávu *"4"*.

```
publish.single("Control", payload="4",hostname=
"192.168.0.127")
```

5.3 Vzdálené řízení modelu

Vzdálené řízení je řešeno pomocí jednoduchého uživatelského rozhraní (UI – user interface). UI se skládá ze 4 tlačítek (viz. obr. 5-6):

1. SELECTION OF MATERIAL – spustí pracoviště pro selekci zmetků (pouze jednou)
2. LABELING – spustí pracoviště na balení materiálu (pouze jednou)
3. PRODUCTION LINE – spustí se cyklus řízení celé výrobní linky (viz. vývojový diagram na obr. 5-3)
4. TURN OFF – vypne cyklus a jednotlivé komponenty



Obrázek 5-6 UI programu pro vzdálené řízení modelu [Autor]

Program funguje tak, že po stisku zvoleného tlačítka se odešle zpráva na RPi prostřednictvím MQTT pomocí funkce *publish.single*. Na RPi se po obdržení zprávy provede požadovaná funkce. Funkce *publish* je připojena na téma *Control* a k příslušnému brokeru pomocí IP

adresy (blíže vysvětleno v kapitole 5.2.3). Příklad publikování zprávy pomocí tlačítka *Selection*:

```
Selection.config(command=publishSelection)
def publishSelection():
    publish.single("Control",payload="1",hostname="192.168
    .0.127")
    print("Selection material")
```

UI bylo vytvořeno pomocí knihovny Tkinter. Tkinter je knihovna grafického uživatelského rozhraní pro Python. Python ve spojení s Tkinterem poskytuje rychlý a snadný způsob vytváření aplikací s grafickým uživatelským rozhraním. Tkinter poskytuje výkonné objektově orientované rozhraní pro sadu nástrojů Tk GUI. Je multiplatformní, takže stejný kód funguje v systémech Windows, MacOS i Linux. Vizuální prvky jsou vykreslovány pomocí nativních prvků operačního systému, takže aplikace vytvořené pomocí Tkinteru vypadají, jako by patřily na platformu, na které jsou spuštěny. Celý program *ControlsUI.py* je možné nalézt v příloze č.1.

5.4 Propojení databáze s použitím Python a SQLite

Pro ukládání dat z modelu Fischertechnik do lokální databáze byl využit SQLite, který je zmíněný v kapitole 4.2. Tento přístup byl zvolen z důvodu, že SQLite má malou kódovou stopu, efektivně využívá paměť, místo na disku a šířku pásma disku, je vysoce spolehlivý a nevyžaduje žádnou údržbu od správce databáze. Na rozdíl od databází založených na principu klient-server, kde je databázový server spuštěn jako samostatný proces, je SQLite pouze nevelká knihovna. Po přilinkování k aplikaci je k dispozici pomocí jednoduchého rozhraní. Každá databáze je uložena v samostatném souboru *.dbm* (Database Manager), kde se data ukládají za použití primárního klíče. Nevýhodou SQLite je, že funguje pouze jako lokální vestavěná databáze. Pro přístup k databázím v síti online se musí použít např. některé rozhraní API webové služby používané přes HTTP(S) nebo zmíněný protokol MQTT. [53]

Python poskytuje dvě oblíbená rozhraní pro práci s databázovou knihovnou SQLite: PySQLite a APSW. Každé rozhraní se zaměřuje na soubor různých potřeb.

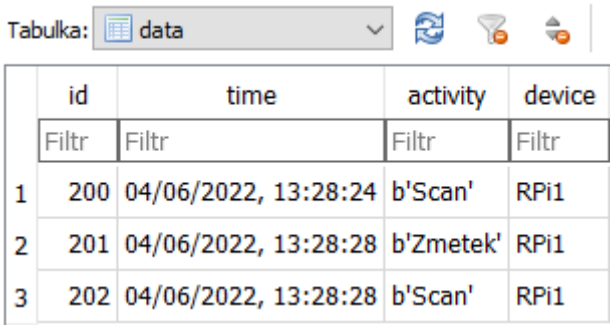
- PySQL se doporučuje pokud aplikace potřebuje podporovat nejen databázi SQLite, ale také další databáze, jako je MySQL, PostgreSQL a Oracle. PySQLite je součástí standardní knihovny Python od verze Pythonu 2.5

- Pokud aplikace potřebuje podporovat pouze databázi SQLite, doporučuje se použít modul APSW, který je známý jako Another Python SQLite Wrapper. APSW je minimální vrstvou nad SQLite a snaží se pouze přeložit kompletní SQLite API (Application Programming Interface) do Pythonu. [54]

Pro vytvoření a správu databáze byl použit software DB browser. Tento program nevyžaduje znalost příkazů SQL. Slouží jako jednoduchý a rychlí nástroj pro tvorbu jednoduchých databází. Pro účely této práce je tento nástroj dostačující. Software umožňuje vytvářet, modifikovat, mazat, vyhledávat záznamy či tabulky.

Byla vytvořena databáze *DeviceDatabase.db* (viz. příloha 6). Tato databáze slouží jako přehled zpráv z jednotlivých zařízení připojených k RPi. Jedná se o jednoduché zprávy, které

nesou informaci kdy a jaké zařízení bylo spuštěno. K tomu poslouží jedna tabulka (s názvem *data*) s položkami – id, time, activity, device (viz. obr. 5-7). Id slouží jako primární klíč tabulky.


Tabulka: data

	id	time	activity	device
	Filtr	Filtr	Filtr	Filtr
1	200	04/06/2022, 13:28:24	b'Scan'	RPi1
2	201	04/06/2022, 13:28:28	b'Zmetek'	RPi1
3	202	04/06/2022, 13:28:28	b'Scan'	RPi1

Obrázek 5-7 SQLite databáze data [Autor]

Pro práci s SQLite databází a příjmání dat prostřednictvím MQTT bylo nutné naimportovat potřebné knihovny. Knihovny se naimportovala pomocí příkazu:

```
import sqlite3
from sqlite3 import Error
```

Následně byla vytvořena funkci pro navázání databázového připojení k databázi SQLite specifickým databázovým souborem:

```
def create_connection(devices):
    conn = None
    try:
        conn = sqlite3.connect(devices)
    except Error as e:
        print(e)
  return conn
```

Dále byla vytvořena funkce pro vložení nového záznamu do tabulky data. Do tabulky se vkládají 3 hodnoty (time, device, status - Id záznamu se vyplňuje automaticky):

```
def create_record(conn, data):
    sql = ''' INSERT INTO projects(time,device,status)
              VALUES(?,?,?) '''
    cur = conn.cursor()
    cur.execute(sql, data)
    conn.commit()
    return cur.lastrowid
```

Poté byla vytvořena funkce main(), která vytvoří propojení k určité databázi (v tomto případě *DeviceDatabase.db*), nový řádek v tabulce data a vloží do databáze potřebná data (čas vložení, zprávu obdrženou od brokera a od jakého zařízení byla zpráva obdržena):

```
def main(msg, date_time):

    database = r"C:\Users\mhucl\OneDrive\Plocha\Diplomka\
    DeviceDatabase.db"
    conn = create_connection(database)
    with conn:
       data_1 = (str(date_time), str(msg.payload),"RPi1")
       create_data(conn, data_1)
```

Vkládání dat do databáze prostřednictvím MQTT se provádí funkcí *on_message*, která se volá po obdržení zprávy od brokera. Funkce vytvoří proměnou *date_time*, která využívá python knihovnu *datetime* pro určení času vložení dat a následně zavolá funkci *main* s potřebnými proměnnými:

```
def on_message(client, userdata, msg):
   today = datetime.datetime.now()
   date_time = today.strftime("%m/%d/%Y, %H:%M:%S")
   print("message received ",
   str(date_time),str(msg.payload.decode("utf-8")))
   main(msg, date_time)
```

Python program pro příjem dat z modelu a k jejich následnému ukládání lze nalézt v příloze č.2. Programy pro vzdálené řízení modelu a pro příjem dat byly použity pro oba koncepty stejně.

6 Koncept 1

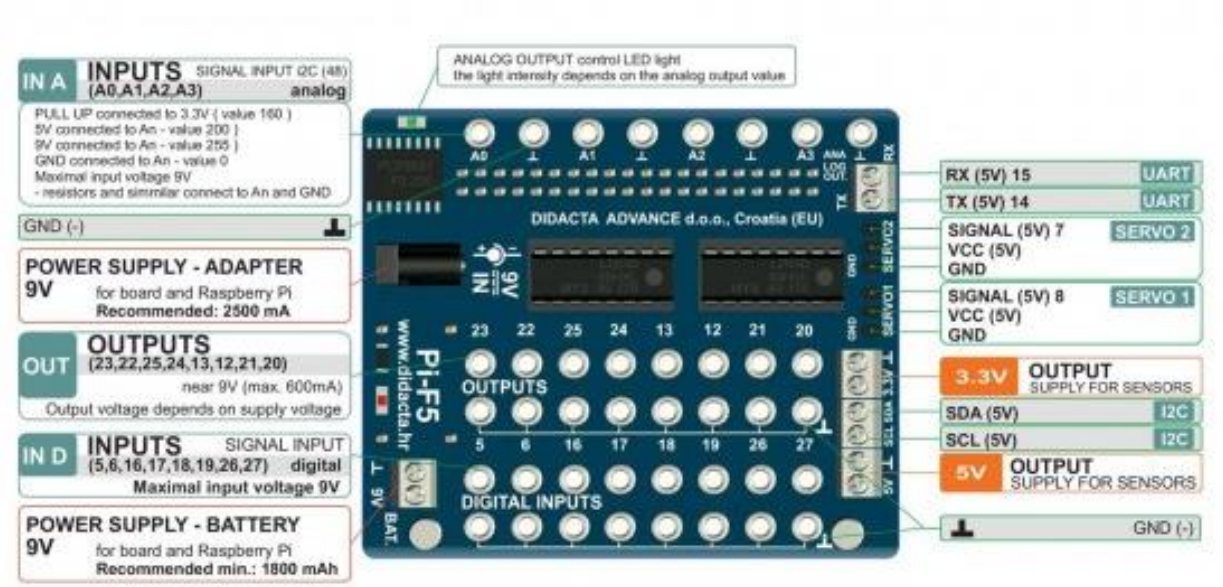
První koncept je navržen tak, že RPi přímo komunikuje s modelem Fischertechnik prostřednictvím GPIO pinů. Fischertechnik a GPIO RPi však pracují s jinými hodnotami napětí. GPIO RPi pracuje s 3,3 volty, k neopravitelnému poškození může dojít i při napětí 5 voltů. Komponenty Fischertechnik naproti tomu využívají provozní napětí 9 voltů, což rozhodně převyšuje možnosti RPi. Motory Fischertechnik také vyžadují proud kolem 250 miliampérů. Porty GPIO RPi nemohou poskytnout takové množství energie. [55] V praxi však scénář nepředstavuje zásadní problém, protože úpravy úrovní byly v elektronice vždy nutné a existují běžné komponenty, které tento problém řeší.

Zvoleným programovacím jazykem pro tuto práci je Python. Python nabízí celou řadu knihoven a programátor se nemusí zabývat nízkoúrovňovým programováním. V tomto konceptu byly vytvořeny celkem 3 python programy:

1. *DP_Hucl_UIcontrols.py* - Program pro uživatele pro vzdálené řízení modelu (centrální PC/RPi – viz. kapitola 5.3)
2. *DP_Hucl_MQTTdatabase.py* - Program pro příjmání dat z jednotlivých komponent a jejich ukládání je do SQLite databáze (centrální PC/RPi – viz. kapitola 5.4)
3. *DP_Hucl_RPicontrol.py* - Program pro příjmání příkazů od uživatele, řízení jednotlivých komponent a zasílání dat do databáze (program spuštěn na RPi připojeném ke zkušebnímu modelu prostřednictvím GPIO – viz. kapitola 6.2)

6.1 Raspberry Pi F5 adaptér

K připojení Fischertechnik k počítači Raspberry Pi nám poslouží speciální Raspberry Pi F5 Adapter viz obr. 6-1 . Fischertechnik 179448 Raspberry Pi F5 Adapter je speciálně navržený elektronický adaptér pro ovládání DC motorů a servomotorů, stejně tak digitálních a analogových senzorů Fischertechnik. [56]



Obrázek 6-1 Fischertechnik 179448 Raspberry Pi F5 Adapter [57]

Je možné pracovat současně se 4 DC motory obousměrně a se 2 servomotory (microservo 4,8V/6V). Počet motorů, které mohou pracovat současně, závisí na napájecím zdroji připojeném k adaptéru (např. Napájecí zdroj Fischertechnik 9V/2,5A). Adaptér a microcontroller Raspberry PI mohou být společně napájeny 9V napájecí jednotkou nebo baterií Fischertechnik Accu Set (1800 mAh). Výstupní napětí, které napájí DC motory, odpovídá přibližně vstupnímu napájecímu napětí adaptéru (9V). Adaptér samozřejmě také podporuje všechny DC motory od Fischertechnik (S motor, XS, XM). Adaptér má 8 digitálních vstupů, 4 analogové vstupy a jeden I^2C - lze použít pro všechny senzory Fischertechnik: fototranzistor, fotorezistor, magnetický senzor, spínač, NTC, barevný senzor a IR senzor sledování dráhy. Senzory od Fischertechnik, které fungují prostřednictvím komunikace I^2C s 3,3V, jako jsou senzory prostředí, kombinované senzory a ultrazvukové senzory TXT, nejsou podporovány. Lze však použít senzory jiných výrobců (napájení přes 3,3V nebo 5V výstupní připojení). Adaptér obsahuje také sériový konektor UART (TX/RX). [56]

6.2 Ovládání Fishertechnik modelu pomocí Python

Pro ovládání a adresování GPIO pinů byl použit modul RPi.GPIO. RPi.GPIO je modul Pythonu pro ovládání rozhraní GPIO na Raspberry Pi. Byl vyvinut Benem Crostonem a uvolněn pod licencí softwaru MIT.

Existují dva způsoby číslování IO pinů na Raspberry Pi v rámci RPi.GPIO. Prvním je použití systému číslování BOARD. Ten odkazuje na čísla pinů na hlavičce P1 desky Raspberry Pi. Druhým systémem číslování jsou čísla BCM. Jedná se o způsob práce na nižší úrovni - odkazuje na čísla kanálů v systému Broadcom SOC. Je nutné vždy pracovat se schématem viz. kapitola 3.1, a určit které číslo kanálu patří ke kterému pinu na desce RPi. [58] Čísla pinů pro Raspberry Pi F5 Adapter jsou označena na vrchní straně desky (viz. obr. 6-1). Modul RPi.GPIO se importoval následujícím kódem:

```
import RPi.GPIO as GPIO
```

Systém číslování byl zvolen BCM:

```
GPIO.setmode(GPIO.BCM)
```

Pro nastavení kanálu je nutné určit zda se jedná o vstup nebo výstup. Nastavení kanálu jako input(vstup):

```
GPIO.setup(pinnumber, GPIO.IN)
```

Pro čtení hodnot input pinu GPIO se používá kód:

```
GPIO.input(pinnumber)
```

Nastavení kanálu jako output(výstup):

```
GPIO.setup(pinnumber, GPIO.OUT)
```

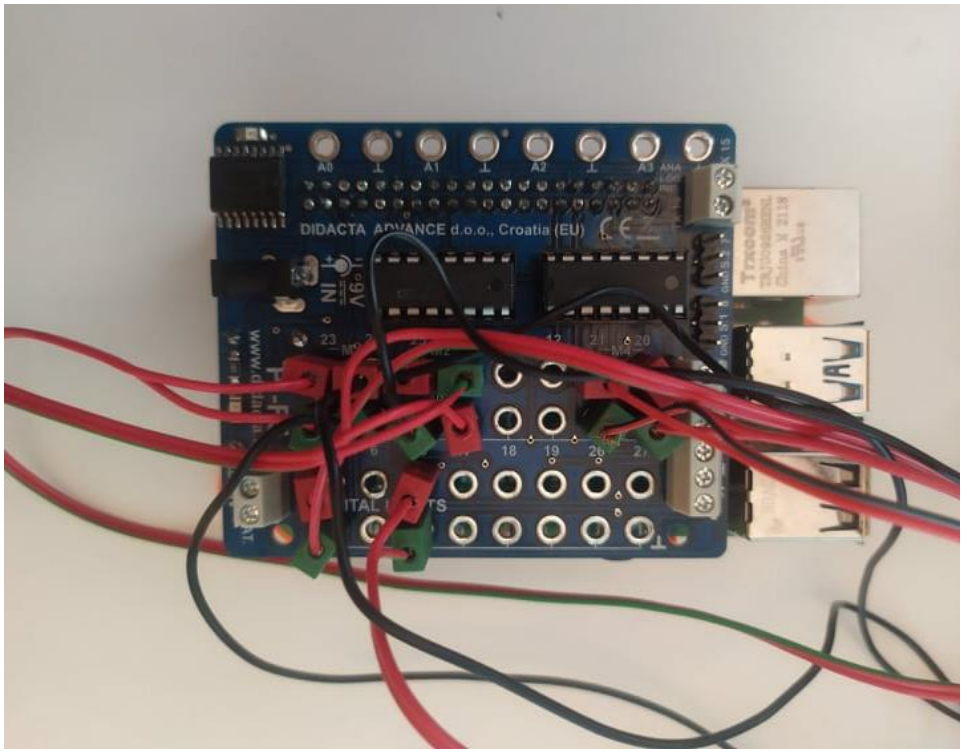
Lze zadat počáteční hodnotu výstupního kanálu (output):

```
GPIO.setup(pinnumber, GPIO.OUT, initial=GPIO.HIGH)
```

Pomocí změny výstupů tak tyto komponenty lze vypínat a zapínat, popřípadě získavat požadovaný input. Přehled komponent a jejich zapojení do příslušných pinů je znázorněn v tabulce 6-1. Jejich zapojení do příslušných pinů je znázorněn na obrázku č. 6-2.

Použité komponenty	**Outout/input**	**Pin**
LED žárovka 1	output	20
LED žárovka 2	output	21
Píst 1	output	22
Píst 2	output	25
Kompresor	output	23
Fotorezistor 1	input	5
Fotorezistor 2	input	16
XS motor	motor	24

Tabulka 6-1 Přehled použitých elektronických komponent a jejich zapojení [Autor]



Obrázek 6-2 Zapojení elektronických komponent do RPi [Autor]

Pro snadnější ovládání jednotlivých komponent a částí byly vytvořeny následující funkce. Kompresor, který stlačuje a žene vzduch do pístů může být zapnutý pořád, proto funkce *kompresor1* pouze nastavý output pinu 23 na hodnotu High.

```
def kompresor1():
```

```
        GPIO.output(kompresor,GPIO.HIGH)
```

Pro ovládání dopravního pásu byly vytvořeny dvě funkce: *motorstart* a *motorstop*. Funkce nastavují output pinu na High nebo Low.

```
def motorstart():
    GPIO.output(motor,GPIO.HIGH)
def motorstop():
    GPIO.output(motor,GPIO.LOW)
```

Funkce pro LED žárovky (*led1* a *led2*) nastavý hodnotu output na High – žárovky svítí neustále. Popřípadě na Low pro jejich vypnutí.

```
def led1():
    GPIO.output(ledone,GPIO.HIGH)
```

Funkce *píst1* otevře ventil na dobu jedné vteřiny a pošle zprávu pomocí MQTT protokolu centrálnímu PC. Píst 1 žene stlačený vzduch do pneumatického válce, který selektuje zmetky. V prvním kroku určí hodnotu outputu na High, pomocí funkce time.sleep zastaví program na jednu vteřinu a následně pomocí funkce *publishZmetek* pošle zprávu Brokerovi. Píst se po zaslání zprávy opět uzavře. Obdobně je řešena funkce *pist2*.

```
def pist1():
    GPIO.output(pistone,GPIO.HIGH)
    time.sleep(1.0)
    publishZmetek()
    GPIO.output(pistone,GPIO.LOW)
```

Pro detekci materiálu na dopravním pásu slouží podmínka *if* a funkce *photo1*. Obdobně se je řešena detekce materiálu na pracovišti balení.

```
def photo1():
    publishSkener()
    motorstop()
    time.sleep(2)
    motorstart()
if GPIO.input(photoone)==1:
    photo1()
```

6.3 Publikování zpráv a řízení RPi

RPi figuruje zároveň jako publisher i jako client. Cílem této práce je zaznamenávat aktivitu jednotlivých úseků modelu fishertechnik a tyto úseky i řídit. RPi publikuje 3 druhy zpráv – zmetek, balení a scan. Zpráva *zmetek* se publikuje, když se spustí funkce *pist1* (provede se odsun materiálu). Zpráva *balení,* když se spustí funkce *pist2* a *scan,* spustí-li se funkce *photoone* (input fototranzistoru se rovná 0). Tyto zprávy jsou napojeny na téma *Backmsg/RPi1*. Jedná se o funkce:

```
    def publishZmetek():

    def publishBaleni():

    def publishSkener():
```

Pro řízení RPi byla vytvořena funkce *on_message*, která se volá po obdržení zprávy od brokera. Na základě obdržené zprávy se spustí požadované funkce. Pomocí funkce *client.subscribe* je klient napojený na téma *Control*. Klient může obdržet od uživatele celkem 4 zprávy (*1, 2, 3, 4*). Po obdržení zprávy *1* se zapne funkce *kompresor()* a *pist1()* a následně se kompresor vypne. Obdobně funguje zpráva *2*. Zpráva *3* zapne cyklus celé funkcionality modelu, konkrétně funkci *mainprogram*. Problém je v tom, že v programu už jeden cyklus běží - *client.loop_forever*. Aby se udrželo MQTT spojení a uživatel mohl program i nadále ovládat a cyklus pomocí příkazu libovolně zastavit, byla použita metoda multithreadingu. Multithreading neboli vícevláknové zpracování v jazyce Python umožňuje procesorům spouštět různé části (vlákna) procesu současně, aby se maximalizovalo využití procesoru. Vícevláknové zpracování tak umožňuje procesoru zpracovávat více cyklů/programů najednou, aby se neovlivňovaly. Knihovna pro multithreading se importuje pomocí příkazu *import threading*. Tento problém by se dal řešit i pomocí zvolení *loop_start* funkce. Tato funkce spustí na pozadí vlákno, které automaticky zavolá funkci *loop()*. Tím se uvolní hlavní vlákno pro jinou práci a ta může být blokována. Voláním *loop_stop()* se vlákno na pozadí zastaví. [59] Objevil se však problém s duplicitou odeslaných zpráv po opětovném připojením k brokerovi, a proto byla zvolena metoda spuštění funkce *mainprogram* na jiném vláknu. Funkce *on_message*:

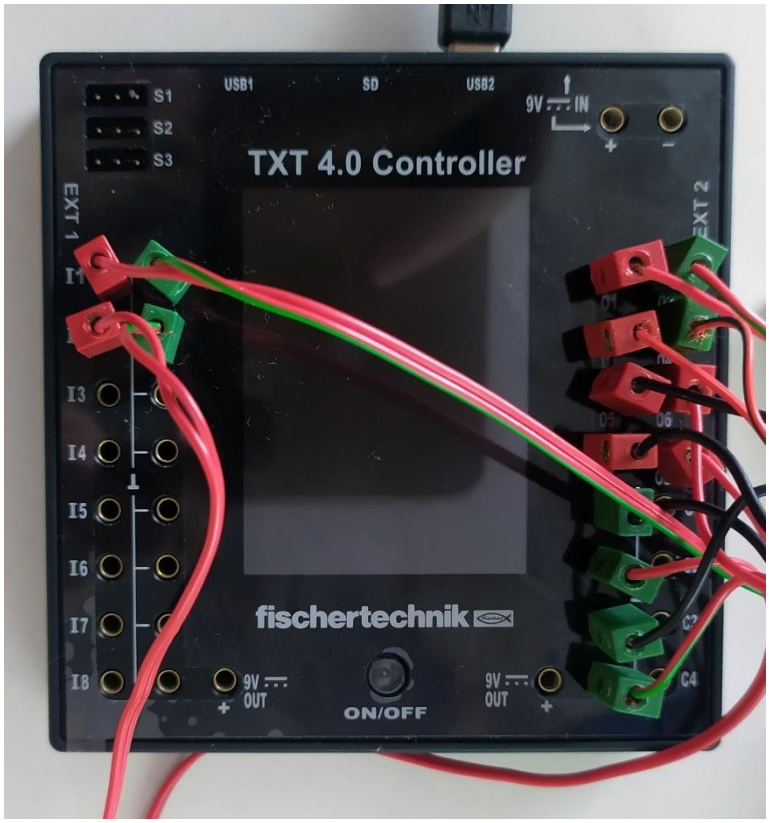
```
def on_message(client, userdata, msg):

    print(msg.payload)
    if msg.payload == b'1':
        kompresor1()
        pist1()
        GPIO.output(kompresor,GPIO.LOW)

    if msg.payload == b'2':
        kompresor1()
        pist2()
        GPIO.output(kompresor,GPIO.LOW)

    if msg.payload == b'3':
        exit_event.clear()
        global main_thread
        main_thread = threading.Thread(target=mainprogram,
        name="mainprogram")
        main_thread.start()

    elif msg.payload == b'4':

        exit_event.set()
        GPIO.output(ledone,GPIO.LOW)
```

```
GPIO.output(ledtwo,GPIO.LOW)
GPIO.output(pistone,GPIO.LOW)
GPIO.output(pisttwo,GPIO.LOW)
GPIO.output(motor,GPIO.LOW)
GPIO.output(kompresor,GPIO.LOW)
```

Po obdržení zprávy *4* se vlákno uvolní a vypnou se všechny komponenty (všechny hodnoty GPIO outputů se nastaví na Low). Pro uvolnění vlákna a zastavení cyklu *mainprogram* se použila metoda zpracování události (Event handling). Zpracování událostí vytváří responzivní aplikaci, která dokáže detekovat a generovat akce na základě odezvy. Byla vytvořena událost *exit_event = threading.Event().* V programu je pak možné sledovat stav této události. Cyklus *while* funkce *mainprogram* sleduje, jestli není tato událost aktivována (*while not exit_event.is_set():*). Po obdržení zprávy *4* se událost aktivuje příkazem *exit_event.set()* a tím se cyklus přeruší. Aby se cyklus mohl opětovně spustit, musí se událost opět deaktivovat příkazem *exit_event.clear()*. Tento příkaz se spustí po obdržení zprávy *3*. Vypnutí všech komponent a přerušení cyklu nefunguje bez problémů. Někdy je potřeba příkaz pro zastavení modelu odeslat dvakrát. Když se cyklus přeruší uprostřed vykonávané funkce, nemusí se zcela ukončit některé činnosti, např. běh motoru.

7 Koncept 2

V další variantě je model Fischertechnik ovládán pomocí řídící jednotky TXT Controller 4.0. Ten je naprogramován pomocí programu ROBO Pro Coding (viz. kapitola 7.2). TXT4.0 komunikuje s PC pomocí WiFi sítě. Vzdálené řízení a sběr dat ze zkušebního modelu je řešeno stejně jako v konceptu 1 pomocí MQTT protokolu (viz. kapitola 5.3 a kapitola 5.4). Na obr. 7-1 je zobrazena řídící jednotka a zapojení jednotlivých komponent do pinů.



Obrázek 7-1 Zapojení komponentů Fischertechnik do TXT4.0 [Autor]

Komunikovat s řídící jednotkou lze pomocí WiFi, Bluetooth, USB, či lze program uložit přímo na paměť zařízení. Další možným komunikačním portálem je propojení pomocí I^2C sběrnice přes rozšiřujícího rozhraní EXT přes piny SCL – hodinový signál a SDA – datový signál (viz. obr. 7-2).

Pin	
1	+3,3V
2	GND
3	CANL
4	CANH
5	SCL
6	SDA

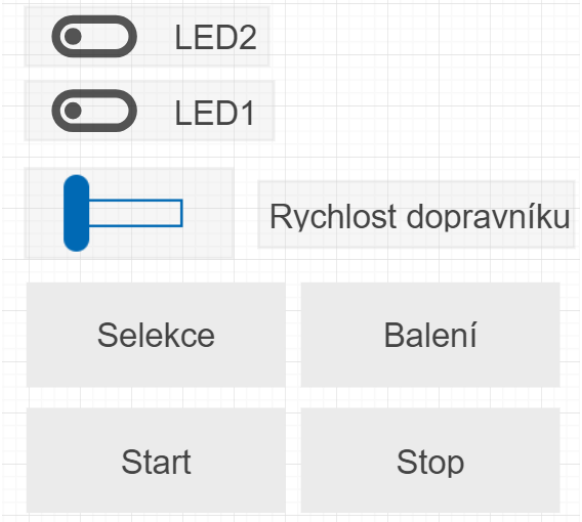
Obrázek 7-2 EXT konektor Fishertechnik Controller 4.0 [61]

Toto propojení je blíže popsáno ve studii [60] a nese s sebou jistá omezení, jako nutnost použití převodníku kvůli jiným napěťovým úrovním, či nemožnost řídící jednotky TXT figurovat v tomto propojení jako slave. Studie byla provedena na starší jednotce Robotics TXT Controller a je možné, že popsaná omezení nejsou u nové jednotky aktuální (napěťová hladina zůstala stejná). Kvůli možnosti komunikace jednotky prostřednictvím WiFi sítě, USB, Bluetooth a možnosti rozšíření ROBO Pro Coding programu o python moduly a funkce nebyla tato varianta dál zkoumána.

7.1 ROBO Pro Coding

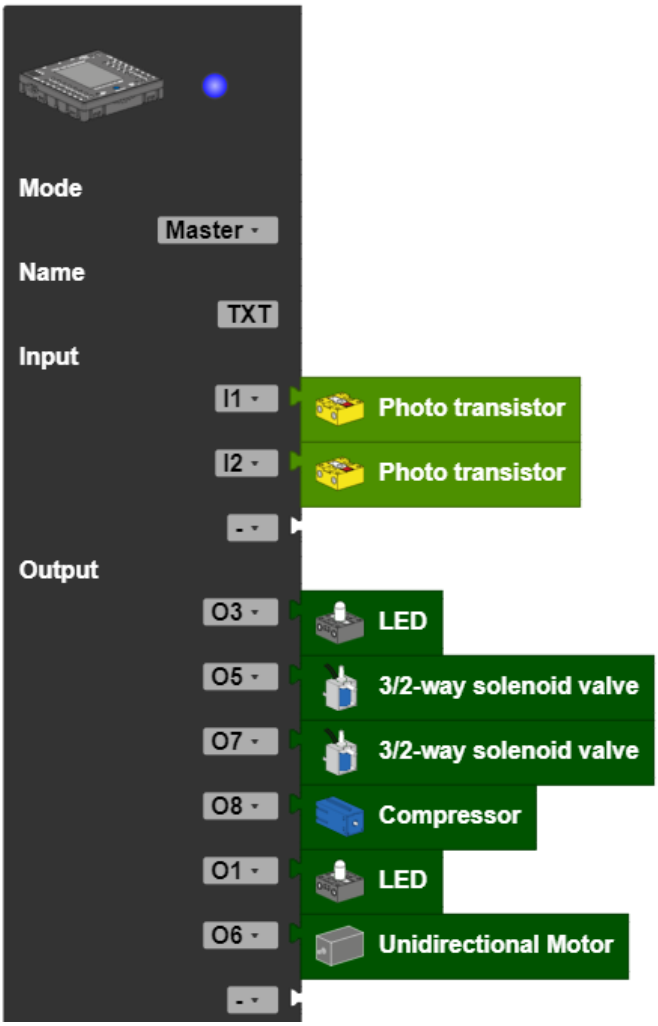
Pomocí softwaru ROBO Pro Coding byl pro model vytvořen řídící program. ROBO Pro Coding je grafický programovací software, kde se programuje formou skládání programovacích bloků, nicméně umožňuje i textové programování přes Python. To umožňuje zobrazit funkce programu srozumitelným způsobem. V případě potřeby lze programovat řídící kód čistě i v jazyku C/C++ nebo v Pythonu. Software je velice přehledný, jednoduchý a přístupný pro začátečníky. Programovací prostředí Fischertechnik ROBO Pro Coding je dostupné na platformách Windows10 / Mac OS / Linux / iOS / Android a je k dispozici zdarma v příslušných obchodech s aplikacemi. Nicméně se nepovedlo software nainstalovat na RPi OS, proto byl pro programování v softwaru a řízení řídící jednotky použit počítač s operačním systémem Windows. K zařízení se počítač připojil prostřednictvím WiFi sítě. Zařízení musí být připojená ke stejné síti, pro vzdálené propojení např. z domova by se propojení mohlo řešit přes routování. Fishertechnik nabízí také možnost komunikovat s řídící jednotkou prostřednictvím Fishertechnik Cloud. Tato varianta neni geograficky omezená a model se může ovládat pomocí smartphonu, tabletu nebo jakéhokoli počítače s přístupem k internetu. Stačí se zaregistrovat na oficiálních stránkách Fishertechnik Cloud [62] a pomocí programovacího bloku *fishertechnik cloud connect* daný program ke cloudu připojit.

V ROBO Pro Coding v záložce Display Configuration lze i vytvářet aplikace s uživatelským rozhraním a pomocí tlačítek, sliderů, přepínačů a textových inpůtu model ovládat. Tyto ovladače lze připojit k jednotlivým komponentům, popřípadě je propojit k celým funkcím programu. Na obr. 7-3 je zobrazeno ovládání LED diod, rychlost otáček motoru a spouštění funkcí selekce zmetků (*selekceZmetku*), balení (*baleni*) a zapnutí, vypnutí hlavního progamu linky (*mainprogram*). Tento přístup byl vyzkoušen, ale ve výsledném programu není použitý.



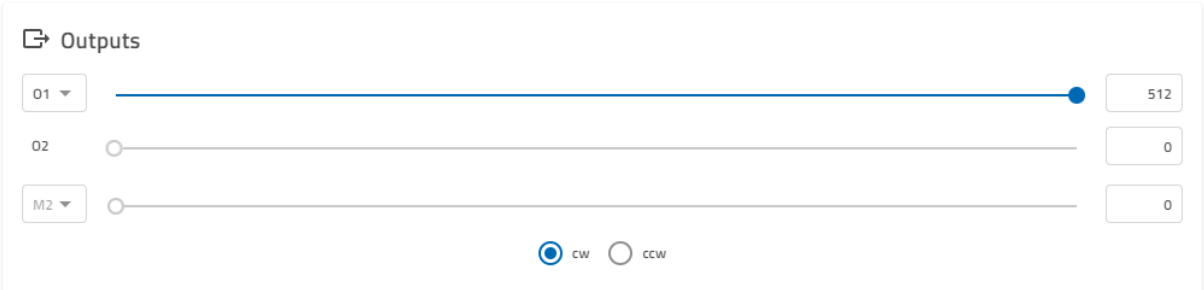
Obrázek 7-3 ROBO Pro Coding Display Configuration UI [Autor]

Pro použití jednotlivých akčních komponent stavebnice Fishertechnik je nutné je připojit k řídicí jednotce nejen fyzicky, ale i softwarově. Pro softwarové připojení v programu ROBO Pro Coding slouží záložka *Controller Configuration*, kde se nachází všechny bloky komponentů Fishertechnik pro připojení k řídicí jednotce. Použité komponenty a jejich softwarové zapojení lze vidět na obr. 7-4.



Obrázek 7-4 Konfigurace řídicí jednotky v programu ROBO Pro Coding [Autor]

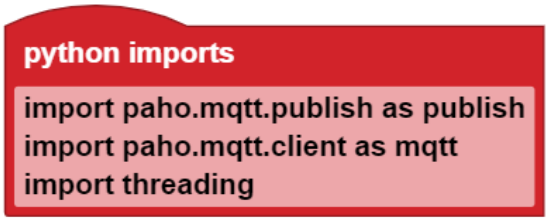
Nejsnazší způsob, jak ovládat zvolené komponenty, je prostřednictvím záložky Interface test viz. obr. 7-5. V tomto prostředí je možné nastavovat hodnoty jednotlivých výstupu, popřípadě číst hodnoty jednotlivých vstupů.



Obrázek 7-5 Robo Pro Coding - Interface test outputs [Autor]

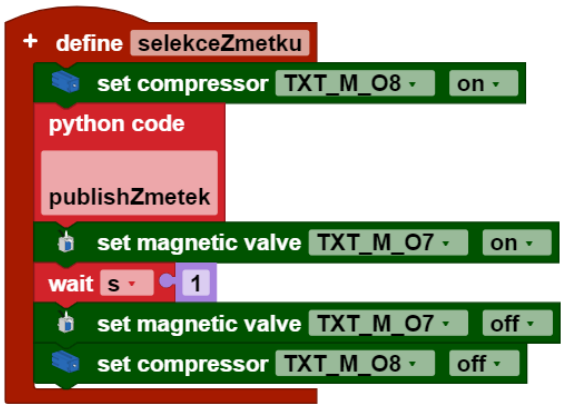
7.2 Robo Pro Coding program pro řízení zkušebního modelu

Program vytvořený ve vývojovým prostředí Robo Pro Coding má stejnou funkcionalitu jako python program uvedený v konceptu 1 (viz. kapitola 6). Program funguje jako klient a publisher zároveň. Respektive po obdržení určité zprávy MQTT se spustí požadovaná funkce, s tím rozdílem, že program byl vytvořen pomocí grafických programových bloků. Jak už bylo zmíněno Robo Pro Coding umožňuje kombinovat programovací bloky s textovým kódem python. Je tak možné vkládat i python knihovny. Knihovny se importovaly pomocí bloku *python imports* viz. obr. 7-6.



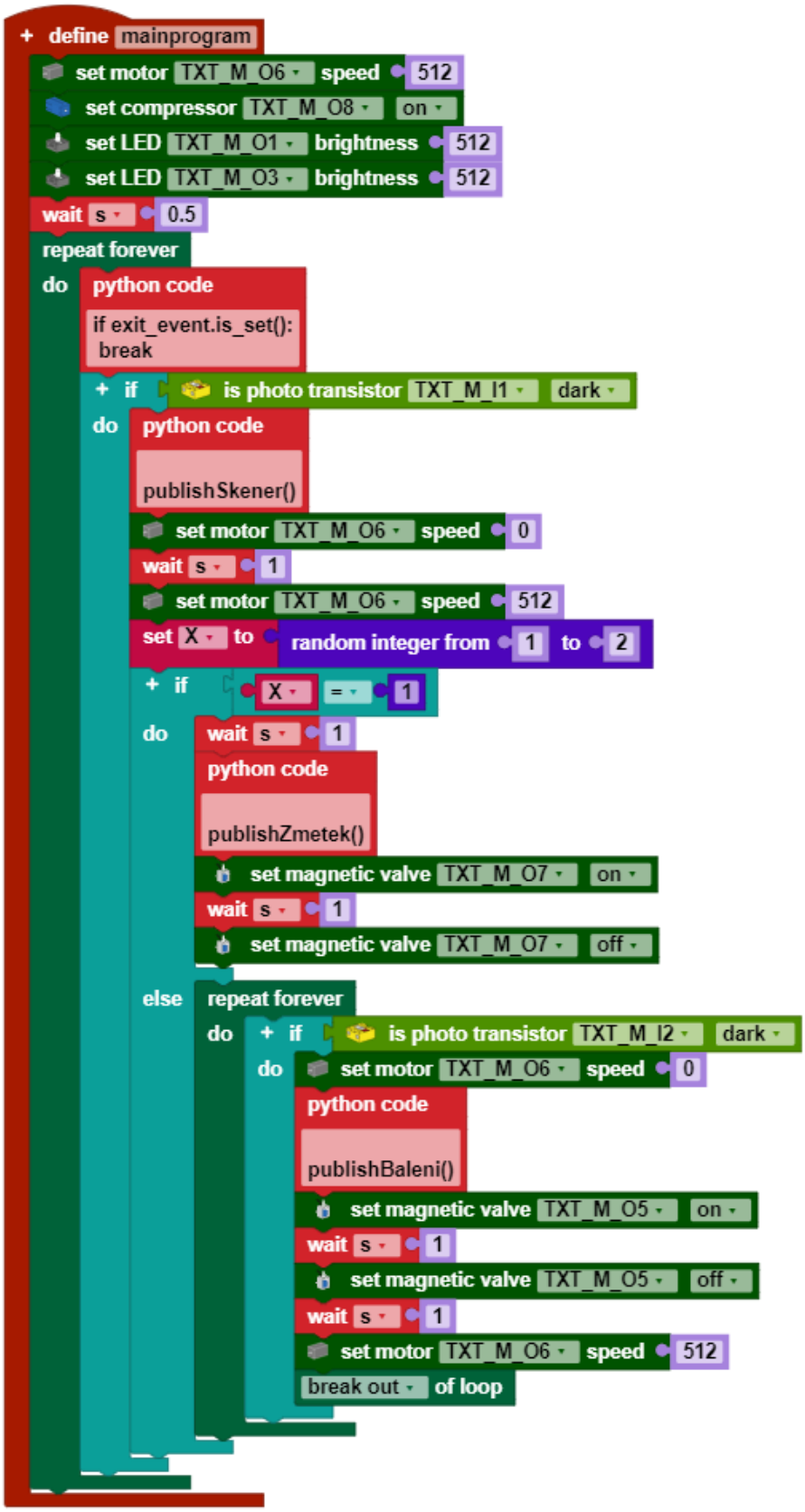
Obrázek 7-6 Python imports v programu ROBO Pro Coding [Autor]

Je možné zakládat jednotlivé funkce, které pak můžeme pomocí *python code* volat. Jako příklad je uvedená funkce *selekceZmetku* (viz. obr. 7-7), která zapne pracoviště pro selekci zmetků. Funkce zapne kompresor zavolá další funkci *publishZmetek*, otevře zvolený píst na 1 vteřinu a vypne kompresor.



Obrázek 7-7 Příklad vkládání funkce v programu ROBO Pro Coding [Autor]

Jednotlivé funkce se volají, stejně jako v konceptu 1, pomocí obdržené zprávy přes protokol MQTT. Zpracování těchto zpráv a volání jednotlivých funkcí zajišťuje funkce *on_message*. Logika této funkce je stejná jako v konceptu 1. Respektive po obdržení zprávy od uživatele spustí určité funkce. Funkce *mainprogram* je cyklus, který zajišťuje chod pro celou linku, a je uvedena na obrázku 7-8. Logika tohoto programu vychází z vývojového diagramu (viz. obr. 5-3). Celý program pro řídící jednotku TXT4.0 je uveden v příloze č. 4.



Obrázek 7-8 Funkce mainprogram – cyklus pro ovládání modelu

7.3 Zhodnocení konceptů

Oba dva koncepty jsou přístupem takřka indentické. Požadovanou funkcionalitu řízení a sběru dat splňují oba. Rozdíl je v přístupu tvorby řídícího programu. ROBO Pro Coding nabízí zajímavý přístup tvoření pomocí programovatelných grafických bloků na rozdíl od RPi, kde probíhala tvorba programu v klasickém python prostředí pomocí GPIO knihovny. Tento grafický přístup se hodí pro výuku programovací logiky, základů robotiky a průmyslového inženýrství. Dalším rozdílem je způsob připojení k řídící jednotce. ROBO Pro Coding nabízí snadný přístup ke Controlleru TXT4.0 prostřednictvím WiFi sítě a umožňuje číst a ukládat data přímo ze softwaru do databáze pomocí python nadstavby (obdobně jako v kapitole 5.4). Není tak nutný vzdálený přístup pomocí protokolu MQTT a vše se může ovládat v rámci programu ROBO Pro Coding, kde lze vytvořit i optimalizované uživatelské prostředí pomocí záložky Display Configuration. Software však nebylo možné nainstalovat na operační systém RPi OS i přes dotázání a ujištění na Fishertechnik support, že by software měl být s OS kompatibilní. Jedná se nejspíše o chybu nejnovější verze, která bude dozajista v nejbližší době opravena. Komunikace s programem ROBO Pro Coding prostřednictvím MQTT byl zvolen z důvodů vyzkoušení možnosti využití softwaru od společnosti Fishertechnik v „klasickém“ programování a možnosti komunikace s jinými zařízeními, nejenom s těmi, které nabízí společnost Fishertechnik.

Vzdálené řízení a ukládání dat do databáze prostřednictvím protokolu MQTT umožňuje ovládat a sbírat data z vícero zařízení najednou. Další zařízení (klienty) stačí přihlásit k odběru k požadovaném tématu zprávy odeslané brokerem, popřípadě data publikovat. Zvoleným přístupem se dá snadno vytvořit škálovatelná síť zařízení která spolu mohou komunikovat a reagovat na sebe. Princip může posloužit při tvorbě komplexních robotických modelů s použitím vícero řídících jednotek. Zařízení nejsou omezená operačním systémem díky multiplatformní podpoře programovacího jazyka Python a MQTT.

Závěr

V této práci byla řešena problematika řízení modelu Fischertechnik a sběr dat s využitím RaspberryPi. Cílem této práce bylo vyzkoušení principů IoT. K simulaci výrobní linky byl použit model Fischertechnik. Výrobní linka reprezentuje pracoviště rentgenu, selekci zmetků a balení. V práci byl popsán veškerý použitý hardware a jeho možnosti komunikace a propojení prostřednictvím sběrnic či bezdrátové sítě. Práce se také zabývala možnostmi multidevice komunikace pomocí protokolů zpráv. Byly vytvořeny dva koncepty propojení:

Koncept 1 využívá GPIO piny Raspberry Pi k přímé komunikaci se stavebnicí Fischertechnik. K řízení modelu byl zvolen programovací jazyk Python. Python byl zvolen, protože existuje velké množství knihoven od uživatelů pro ovládání Raspberry Pi, a proto nebylo nutné se zabývat nízkoúrovňovým programováním.

Koncept 2 pracuje s řídící jednotkou Controller TXT4.0 od výrobce Fischertechnik, který je propojený s počítačem prostřednictvím WiFi sítě pomocí programu ROBO Pro Coding.

U obou konceptů pro výměnu dat mezi jednotlivými zařízeními byl zvolen serverový přístup pomocí protokolu MQTT. Pro obě varianty byl zpracován koncept pro ukládání dat do SQLite databáze a řízení modelu pomocí jednoduchého UI. Python programy pro obě funkcionality byly spuštěny na centrálním PC s operačním systémem Windows. Koncept je navržen tak, aby programy mohly být spuštěny na kterémkoliv zařízení, které podporuje python a má nainstalované požadované knihovny. Jako centrální PC může figurovat například další RPi. RPi slouží kromě toho jako řídící jednotka v konceptu 1 i jako MQTT server (broker).

Model je možné ovládat a získávat data i z domova prostřednictvím internetové sítě a IP adresy. Pomocí tohoto přístupu je možné ukládat data do databáze z více zařízení najednou a vytvořit síť modelů, které na sebe mohou reagovat a spolu komunikovat (IoT). Stačí se přihlásit k odběru určitého tématu. Je tak možné vytvořit funkční, automatizovanou a inteligentní továrnu (smart factory), jež by mohla flexibilně reagovat na tok materiálu a zaznamenávat veškerou aktivitu. Výměna dat pomocí protoklu MQTT byla zvolena pro vyzkoušení principů IoT a jeho rostoucí popularity v průmyslové praxi. Principy demonstrované v této práci se dají aplikovat na řízení a sběr dat takřka jakýchkoliv robotických komponent.

Seznam použitých zdrojů

[1] JOHNSTON, Steven, BASFORD, Philip, PERKINS, Colin, HERRY, Herry, TSO PO, Fung, PEZAROS, Dimitrios, MULLINS, Robert, YONEKI, Eiko, COX, Simon, SINGER, Jeremy, *Future Generation Computer Systems*, 2018, Doi: https://doi.org/10.1016/j.future.2018.06.048, [online]. Dostupné z: https://www.cl.cam.ac.uk/~ey204/pubs/2018_SingleBoard.pdf
[2] ARIZA, Álvarez, HEYSON, Jonathan & Baez, *Understanding the role of single-board computers in engineering and computer science education: A systematic literature review. Computer Applications in Engineering Education*, 2021, Doi: 10.1002/cae.22439, [online]. Dostupné z: https://www.researchgate.net/publication/352132667_Understanding_the_role_of_single-board_computers_in_engineering_and_computer_science_education_A_systematic_literature_review
[3] BARRAGÁN, Hernando, *The Untold History of Arduino*, [online] [cit. 16.4.2022]. Dostupné z: https://arduinohistory.github.io/
[4] TEJA, Ravi, *What are the Differences Between Raspberry Pi and Arduino?*, 2021 [online] [cit. 16.4.2022]. _https://www.electronicshub.org/raspberry-pi-vs-arduino/
[5] UPTON, Eben a HALFACREE, *Gareth, Raspberry Pi Uživatelská příručka*. 2013. Brno: COMPUTER PRESS. 232 s. ISBN 978-80-251-4116-8.
[6] Raspberry Pi Foundation. *Getting started with Raspberry Pi Pico*. [online] [cit. 3.12.2021]. Dostupné z: https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico
[7] Raspberry Pi Foundation. *Raspberry Pi*. [online] [cit. 3.12.2021]. Dostupné z: https://www.raspberrypi.org/
[8] ŠEBEK, Cyril. *Úvod do Raspberry Pi*. [online] [cit.3.12.2021]. Dostupné z: https://www.itnetwork.cz/hardware-pc/raspberry-pi/uvod-do-raspberry-pi
[9] ARDUCAM. *Raspberry Pi camera pinout*. [online] [cit. 3.12.2021]. Dostupné z: https://www.arducam.com/raspberry-pi-camera-pinout/
[10] LAKSHAN. *HDMI micro vs DSI Raspberry Pi 4*. [online] [cit. 3.12.2021]. Dostupné z: https://www.seeedstudio.com/blog/2020/06/03/hdmi-micro-hdmi-vs-dsi-raspberry-pi-4-display-connectors-m/
[11] DIVINE, Okoi. *20 Best Operating Systems You Can Run on Raspberry Pi in 2021*. [online] [cit. 3.12.2021]. Dostupné z: https://www.fossmint.com/operating-systems-for-raspberry-pi/
[12] VESA-MATTI Yli-Heikkilä. *Home surveillancewith Raspberry Pi*. 2015. School of Technology, Automation Engineering. Vedoucí: Alpo Anttonen. [online] [cit. 3.12.2021]. Dostupné z: https://manualzz.com/doc/18662392/home-surveillance-with-raspberry-pi-vesa-matti-yli-heikki...
[13] CATWELL, *The Raspberry Pi Continues to Thrive for Industrial and Automation Applications.* [online] [cit. 3.12.2021]. Dostupné z: https://community.element14.com/products/raspberry-pi/b/blog/posts/the-raspberry-pi-continues-to-thrive-for-industrial-and-automation-applications
[14] Raspberry Pi Foundation. *Raspberry Pi 4B specifications*. [online] [cit. 3.12.2021]. Dostupné z: http://raspberrypi.cz/ops4/
[15] FISHERTECHNIK, *Fischertechnik.* [online] [cit. 3.12.2021]. Dostupné z: https://www.fischertechnik.de/en/

[16] MALAGA, Miroslav a ULRYCH, Zdeněk. (2019). *Koncept STEM se zaměřením na problematiku Industry 4.0*. DOi: 10.24132/PI.2019.08948.094-100. [online] dostupné z: https://www.researchgate.net/publication/336948497_Koncept_STEM_se_zamerenim_na_problematiku_Industry_40
[17] FISHERTECHNIK, *The fischertechnik TXT 4.0 Controller*. [online] [cit. 20.4.2022]. Dostupné z: https://www.fischertechnik.de/en/service/elearning/teaching/txt-40-controller
[18] MALAGA, Miroslav, ULRYCH, Zdeněk. *Základy řízení robotů pro strojní inženýrství*. 1. vyd. Plzeň: Západočeská univerzita v Plzni, 2020. 145 s. ISBN 978-80-261-0486-5. [online]. Dostupné z: https://dspace5.zcu.cz/handle/11025/36601
[19] NIKHIL, Agnihotri. *RPi Python Programming 25 – Synchronous serial communication in Raspberry Pi using I2C protocol*. 20.8.2020 [online] [cit. 3.12.2021]. Dostupné z: https://www.engineersgarage.com/articles-raspberry-pi-i2c-bus-pins-smbus-smbus2-python/
[20] NIKHIL, Agnihotri, *RPi Python Programming 24: I2C explained*. 20.7.2020 [online] [cit. 3.12.2021]. Dostupné z: https://www.engineersgarage.com/articles-raspberry-pi-i2c-protocol/
[21] WOJCIECHOWICZ Tyler, *What Are Clock Signals in Digital Circuits, and How Are They Produced?* 19.8.2020 [online] [cit. 3.12.2021]. Dostupné z: https://www.symmetryelectronics.com/blog/what-are-clock-signals-in-digital-circuits-and-how-are-they-produced-symmetry-blog/
[22] NIKHIL, Agnihotri, *RPi Python Programming 17: Serial communication using UART protocol*. 2020 [online] [cit. 3.12.2021]. Dostupné z: https://www.engineersgarage.com/articles-raspberry-pi-serial-communication-uart-protocol-serial-linux-devices/
[23] BUTT Ali Usman. *Interfacing Raspberry Pi with Arduino* [online] [cit. 3.12.2021]. Dostupné z: https://www.engineersgarage.com/interfacing-raspberry-pi-with-arduino-2/
[24] PECINOVSKÝ, Rudolf. Začínáme programovat v jazyku Python. 2020. Praha: Grada. 270 s. ISBN 978-80-271-1237-1.
[25] KULHAM, Dave. *A Python Book: Beginning Python, Advanced Python, and Python Exercises.* Vyd. 1.1. 2012. [online] [cit. 3.12.2021]. Dostupné z: https://web.archive.org/web/20120623165941/http://cutter.rexx.com/~dkuhlman/python_book_01.html#part-1-beginning-python
[26] PRITCHARD, David. *Beginners Guide to Python.* [online] [cit. 3.12.2021]. Dostupné z: https://web.archive.org/web/20120622063020/http://wiki.python.org/moin/BeginnersGuide
[27] Random nerd tutorials. *Raspberry Pi: Install Apache + MySQL + PHP (LAMP Server)* [online] [cit. 3.12.2021]. Dostupné z: https://randomnerdtutorials.com/raspberry-pi-apache-mysql-php-lamp-server/
[28] Docs Python. *DB-API 2.0 interface for SQLite databases.* [online] [cit. 3.12.2021]. Dostupné z: https://docs.python.org/3/library/sqlite3.html
[29] AL-MASRI, Eyhab. *Investigating Messaging Protocols for the Internet of Things (IoT),* 2020, DOi: 10.1109/ACCESS.2020.2993363. [online] [cit. 20.4.2022]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/9090208
[30] MCCLELLAND, Calum. *Messaging Protocols - How Do Machines Talk to Each Other?*, 2017, [online] [cit. 20.4.2022]. Dostupné z: https://www.iotforall.com/messaging-protocols-explained
[31] WORTMANN, Felix, FLUCHTER, Kristina. *Internet of Things. Business & Information Systems Engineering*, 2015. DOi:10.1007/s12599-015-0383-3. [online] [cit. 20.4.2022]. Dostupné z: https://sci-hub.ee/10.1007/s12599-015-0383-3
[32] NAUMAN Ali, QADRI, Yazdan Ahmad. *Multimedia Internet of Things: A Comprehensive Survey*, 2019, [online] [cit. 20.4.2022]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8950450

[33] G. Nick, *How Many IoT Devices Are There in 2022?*, 2022, [online] [cit. 20.4.2022]. Dostupné z: https://techjury.net/blog/how-many-iot-devices-are-there/#gref
[34] BEHRTECH, *6 Leading Types of IoT Wireless Tech and Their Best Use Cases*, [online] [cit. 20.4.2022]. Dostupné z: https://behrtech.com/blog/6-leading-types-of-iot-wireless-tech-and-their-best-use-cases/
[35] LAMPROPOULOS, Georgios, SIAKAS, Kerstin, ANASTASIADIS, *Theofylaktos, Internet of things in the context of industry 4.0: an overview,* 2019. DOi: 10.2478/IJEK-2019-0001, [online] [cit. 20.4.2022]. Dostupné z: http://dspace.vsp.cz/bitstream/handle/ijek/103/IJEK-1-2019%2cv.7_lampropoulos%2cg.siakas%2ck.anastasiadis%2ct...pdf?sequence=1&isAllowed=y
[36] BARTODZIEJ, C. J. *The concept Industry 4.0*. 2016. DOi:10.1007/978-3-658-16502-4_3. [online] [cit. 20.4.2022]. Dostupné z: https://www.researchgate.net/publication/310485455_The_Concept_Industry_40
[37] AL-MASRI, Eyhab. *Investigating Messaging Protocols for the Internet of Things (IoT),* 2020, DOi: 10.1109/ACCESS.2020.2993363. [online] [cit. 20.4.2022]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/9090208
[38] NAIK, N. *Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP*, 2017, DOi: 10.1109/SysEng.2017.8088251. [online] [cit. 20.4.2022]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/8088251
[39] SILVA, D., CARVALHO, L.I., SOARES, J., SOFIA, R.C. *A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA.* 2021, DOi: https://doi.org/10.3390/app11114879. [online] [cit. 20.4.2022]. Dostupné z: https://www.mdpi.com/2076-3417/11/11/4879
[40] UY, N. Q., NAM, V. H., *A comparison of AMQP and MQTT protocols for Internet of Things*, 2019, DOi: 10.1109/NICS48868.2019.9023812. [online] [cit. 20.4.2022]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/9023812
[41] WEISE, Jorn. *Controlling a robot with MQTT - [Part 1]*. 2021. [online] [cit. 3.12.2021]. Dostupné z: https://www.az-delivery.de/en/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/mit-mqtt-einen-roboter-steuern-teil-1
[42]CAVAGNIS, Leonardo, *Android and MQTT: A Simple Guide*, 2020, [online] [cit. 20.4.2022]. Dostupné z: https://medium.com/swlh/android-and-mqtt-a-simple-guide-cb0cbba1931c
[43]MALÝ, Martin. *Protokol MQTT: komunikační standard pro IoT*. 2016. [online] [cit.3.12.2021]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/?fbclid=IwAR2Il8GLD-k2-kPZ94YdLzAFtpetV4_zaqnKJ70KpioXIjzljWK3tGe6kvU
[44] HIVEMQ, *Modernizing the Smart Manufacturing Industry with MQTT*, [online] [cit. 2.3.2022]. Dostupné z: https://www.hivemq.com/solutions/manufacturing/modernizing-the-manufacturing-industry/
[45] BERNSTEIN, Corinne, *MQTT (MQ Telemetry Transport)*, 2021, [online] [cit. 2.3.2022]. Dostupné z: https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport
[47] GOOGLE Play, *MQTT Dash (IoT, Smart Home),* 2017, [online] [cit. 20.4.2022]. Dostupné z: https://play.google.com/store/apps/details?id=net.routix.mqttdash&hl=en_US&gl=US
[48] MOSQUITTO. *Eclipse Mosquitto*. [online] [cit. 2.3.2022]. Dostupné z: https://mosquitto.org/
[49] EMMET. *Installing the Mosquitto MQTT Server to the Raspberry Pi*. [online] [cit. 3.12.2021]. Dostupné z: https://pimylifeup.com/raspberry-pi-mosquitto-mqtt-server/

[50] COPE, Steve. *Understanding MQTT Topics*. [online] [cit. 1.3.2020]. Dostupné z: http://www.steves-internet-guide.com/understanding-mqtt-topics/
[51] The HiveMQ Team. *MQTT Topics & Best Practices - MQTT Essentials: Part 5*. [online][cit.1.4.2022]. Dostupné z: https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/
[52] Eclipse Paho, *paho-mqtt 1.6.1.*, [online] [cit. 6.3.2022]. Dostupné z: https://pypi.org/project/paho-mqtt/
[53] CHAOYU. *Data Collection on Raspberry Pi*. [online] [cit. 3.12.2021]. Dostupné z: https://medium.com/@parano/data-collection-with-raspberry-pi-8b7573943a1f
[54] SQLite tutorial. *SQLite Python.* [online] [cit. 3.12.2021]. Dostupné z: https://www.sqlitetutorial.net/sqlite-python/
[55] Raspberry Pi Geek. *Raspberry Pi steuert Fischertechnik-Roboter*. [online] [cit. 3.12.2021]. Dostupné z: https://www.raspberry-pi-geek.de/ausgaben/rpg/2018/08/raspberry-pi-steuert-fischertechnik-roboter/
[56] Stavebnice pro chytré děti. *Fischertechnik 179448 Raspberry Pi F5 Adapter*. [online] [cit. 3.12.2021]. Dostupné z: https://www.stavebniceprochytredeti.cz/cs/fischertechnik-179448-raspberry-pi-f5-adapter-769.html
[57] FISHERTECHNIK, *Raspberry PI-F5*, [online] [cit. 3.12.2021]. Dostupné z: https://www.fischertechnik.de/en/teaching/arduino-microbit-u-co/raspberry-pi/raspberry-pi-f5
[58] ANONYMOUS, *raspberry-gpio-python.* [online] [cit. 6.3.2022]. Dostupné z: https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/
[59] RALIGHT, *eclipse/paho.mqtt.python* , 2021, [online] [cit. 6.3.2022]. Dostupné z: https://github.com/eclipse/paho.mqtt.python
[60] ŠVRCULA, Petr, MALAGA, Miroslav, ULRYCH, Zdeněk, *Napojení na STEM model pro potřeby průmyslových Inženýrů*, 2021. ISBN: 978-80-261-0792-7, [online] [cit. 20.4.2022]. Dostupné z: https://dspace5.zcu.cz/handle/11025/46453
[61] FISHERTECHNIK, *Fihertechnik Controller TXT 4.0*, [online] [cit. 18.4.2022]. Dostupné z: https://www.fischertechnik.de/en/service/elearning/teaching/txt-40-controller
[62] FISHERTECHNIK, *Fihertechnik Cloud*, [online] [cit. 18.4.2022]. Dostupné z: https://www.fischertechnik-cloud.com/

Seznam obrázků

Seznam tabulek

Seznam příloh vložených na STAG

Příloha č. 1 – DP_Hucl_UIcontrols.py
Příloha č. 2 – DP_Hucl_MQTTdatabase.py
Příloha č. 3 – DP_Hucl_RPicontrol.py
Příloha č. 4 – DP_Hucl_TXT4_0_control.ft
Příloha č. 5 – DP_Hucl_PNGproUI.rar
Příloha č. 6 – DP_Hucl_DeviceDatabase.db

PŘÍLOHA č.1 - DP_Hucl_UI.controls.py

Vzdálené řízení modelu pomocí prokolu MQTT

```
import paho.mqtt.publish as publish

import PySimpleGUI as sg

from tkinter import *

from PIL import ImageTk, Image

def publishSelection():
    publish.single("Control",
    payload="1",hostname="192.168.0.127")
    print("Selection material")

def publishLabeling():
    publish.single("Control",
    payload="2",hostname="192.168.0.127")
    print("Labeling")

def publishProductionLine():
    publish.single("Control",
    payload="3",hostname="192.168.0.127")
    print("Production line on")

def publishOff():
    publish.single("Control",
payload="4",hostname="192.168.0.127")
    print("Off")

window = Tk()
window.title("Production line controls")
window.geometry("300x300")

Selection = Button(window,text="SELECTION OF MATERIAL")
Labeling = Button(window,text="LABELING")
ProductionLine = Button(window,text="PRODUCTION LINE")
GPIOclear = Button(window, text="TURN OFF")

Selection.config(command=publishSelection)
Selection.config(width=30, height=4)
Selection.config(activebackground="#57de49")

Labeling.config(command=publishLabeling)
Labeling.config(width=30, height=4)
Labeling.config(activebackground="#57de49")

ProductionLine.config(command=publishProductionLine)
ProductionLine.config(width=30, height=4)
ProductionLine.config(activebackground="#57de49")
```

```
GPIOclear.config(command=publishOff)
GPIOclear.config(width=30, height=4)
GPIOclear.config(activebackground="#57de49")

selection_img =
ImageTk.PhotoImage(Image.open("C:/Users/mhucl/OneDrive/Plocha/
Diplomka_Final/bin.png").resize((50, 50)))
selection_label = Label(image=selection_img)
selection_label.grid(row=0, column=0)
Selection.grid(row=0, column=1)

labeling_img =
ImageTk.PhotoImage(Image.open("C:/Users/mhucl/OneDrive/Plocha/
Diplomka_Final/packing.png").resize((50, 50)))
labeling_label = Label(image=labeling_img)
labeling_label.grid(row=1, column=0)
Labeling.grid(row=1, column=1)

Prodcution_line_img =
ImageTk.PhotoImage(Image.open("C:/Users/mhucl/OneDrive/Plocha/
Diplomka_Final/production line.png").resize((50, 50)))
Prodcution_line_label = Label(image=Prodcution_line_img)
Prodcution_line_label.grid(row=2, column=0)
ProductionLine.grid(row=2, column=1)

off_img =
ImageTk.PhotoImage(Image.open("C:/Users/mhucl/OneDrive/Plocha/
Diplomka_Final/off.png").resize((50, 50)))
off_label = Label(image=off_img)
off_label.grid(row=3, column=0)
GPIOclear.grid(row=3, column=1)

window.mainloop()
```

PŘÍLOHA č.2 - DP_Hucl_MQTTdatabase.py

Centrální databázové ukládání pomocí prokolu MQTT

```
import sqlite3

from sqlite3 import Error

import paho.mqtt.client as mqtt

import datetime


def create_connection(db_file):

    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return

def create_data(conn, data): #vloží novej řádek

    sql = ''' INSERT INTO data(time, activity, device)
              VALUES(?,?,?) '''

    cur = conn.cursor()
    cur.execute(sql, data)
    conn.commit()

    return cur.lastrowid

def main(msg, date_time):

    database =
r"C:\Users\mhucl\OneDrive\Plocha\Diplomka\DeviceDatabase.db"
    conn = create_connection(database)
    with conn:
        data_1 = (str(date_time), str(msg.payload),"RPi1")
        create_data(conn, data_1)

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("Backmsg/RPi1")

def on_message(client, userdata, msg):
    today = datetime.datetime.now()
    date_time = today.strftime("%m/%d/%Y, %H:%M:%S")
    print("message received ",
str(date_time),str(msg.payload.decode("utf-8")))
    main(msg, date_time)
```

```
def MQTTclientPC():
    client = mqtt.Client()
    client.connect("192.168.0.127", 1883, 60)
    client.on_connect = on_connect
    client.on_message = on_message
    client.loop_forever()

MQTTclientPC()
```

PŘÍLOHA č.3 - DP_Hucl_RPicontrol.py

Řízení modelu pomocí řídící jednotky RPi

```
import paho.mqtt.client as mqtt

import paho.mqtt.publish as publish

import threading

import random

import RPi.GPIO as GPIO
import time

ledone = 20
ledtwo = 21
pistone = 22
kompresor = 23
pisttwo = 25
photoone = 5
phototwo = 16
motor = 24

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledone,GPIO.OUT)
GPIO.setup(ledtwo,GPIO.OUT)

GPIO.setup(kompresor,GPIO.OUT)
GPIO.setup(motor,GPIO.OUT)

GPIO.setup(pistone,GPIO.OUT)
GPIO.setup(pisttwo,GPIO.OUT)

GPIO.setup(photoone,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(phototwo,GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.output(ledone,GPIO.LOW)
GPIO.output(ledtwo,GPIO.LOW)
GPIO.output(pistone,GPIO.LOW)
GPIO.output(pisttwo,GPIO.LOW)

GPIO.output(motor,GPIO.LOW)
GPIO.output(kompresor,GPIO.LOW)

def kompresor1():
    GPIO.output(kompresor,GPIO.HIGH)

def motorstart():
    GPIO.output(motor,GPIO.HIGH)

def motorstop():
    GPIO.output(motor,GPIO.LOW)
```

```
def led1():
    GPIO.output(ledone,GPIO.HIGH)

def led2():
    GPIO.output(ledtwo,GPIO.HIGH)

def pist1():
    GPIO.output(pistone,GPIO.HIGH)
    time.sleep(1.0)
    publishZmetek()
    GPIO.output(pistone,GPIO.LOW)

def pist2():
    GPIO.output(pisttwo,GPIO.HIGH)
    time.sleep(1.0)
    publishBaleni()
    GPIO.output(pisttwo,GPIO.LOW)

def photo1():
            publishSkener()
            motorstop()
            time.sleep(2)
            motorstart()

def photo2():
            motorstop()
            pist2()
            motorstart()

def publishZmetek():
    publish.single("Backmsg/RPi1", "Zmetek",
hostname="192.168.0.127")
    print("Zmetek")

def publishBaleni():
    publish.single("Backmsg/RPi1", "Baleni",
hostname="192.168.0.127")
    print("Baleni")

def publishSkener():
    publish.single("Backmsg/RPi1", "Scan",
hostname="192.168.0.127")
    print("scan")

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("Control")

def on_message(client, userdata, msg):

    print(msg.payload)
```

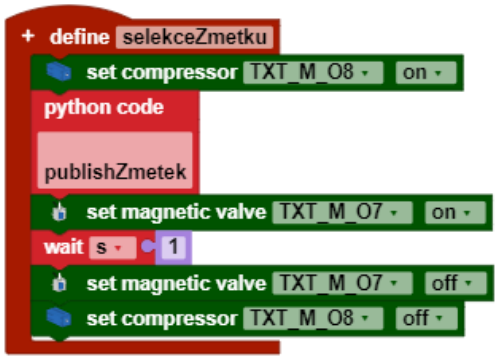
```
    if msg.payload == b'1':
        kompresor1()
        pist1()
        GPIO.output(kompresor,GPIO.LOW)

    if msg.payload == b'2':
        kompresor1()
        pist2()
        GPIO.output(kompresor,GPIO.LOW)

    if msg.payload == b'3':
        exit_event.clear()
        global main_thread
        main_thread = threading.Thread(target=mainprogram,
name="mainprogram")
        main_thread.start()

    elif msg.payload == b'4':

        exit_event.set()
        GPIO.output(ledone,GPIO.LOW)
        GPIO.output(ledtwo,GPIO.LOW)
        GPIO.output(pistone,GPIO.LOW)
        GPIO.output(pisttwo,GPIO.LOW)
        GPIO.output(motor,GPIO.LOW)
        GPIO.output(kompresor,GPIO.LOW)

def MQTTconection():
    client = mqtt.Client()
    client.on_connect = on_connect
    client.connect("192.168.0.127", 1883, 60)
    client.on_message = on_message
    client.loop_forever()

def mainprogram():

    kompresor1()
    motorstart()
    led1()
    led2()
    time.sleep(0.5)

    while not exit_event.is_set():

        if GPIO.input(photoone)==1:

            photo1()

            randomnumber = random.randint(1,2)

            if randomnumber == 1:
```

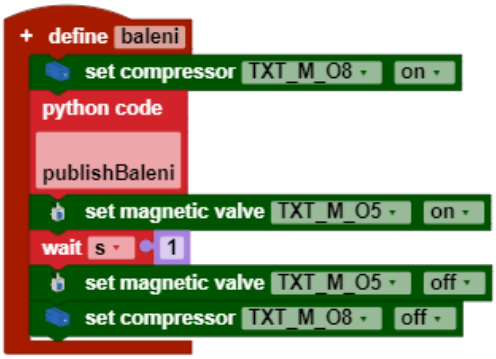
```
                time.sleep(1)
                pist1()
            else:
                while True:

                    if GPIO.input(phototwo)==1:
                        photo2()
                        break
                    else:
                        time.sleep(0.1)

        else:
            time.sleep(0.1)
exit_event = threading.Event()

client = mqtt.Client()
client.on_connect = on_connect
client.connect("192.168.0.127", 1883, 60)
client.on_message = on_message
client.loop_forever()
```

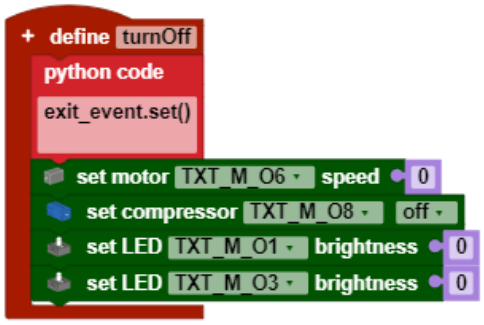
PŘÍLOHA č.4 - DP_Hucl_TXT4_0_control

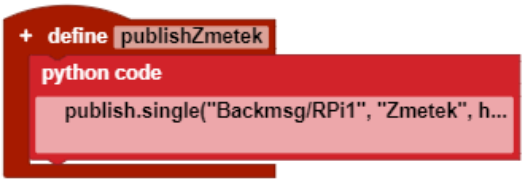
Řízení modelu pomocí řídící jednotky TXT4.0

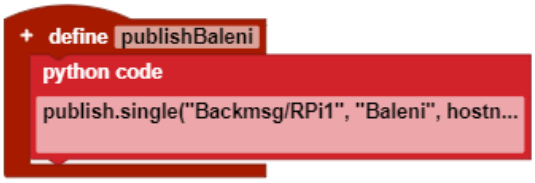

python imports
import paho.mqtt.publish as publish
import paho.mqtt.client as mqtt
import threading

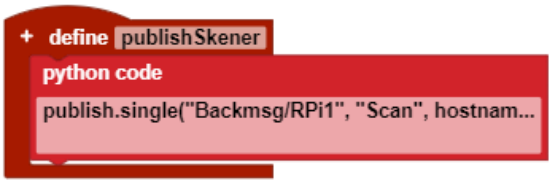

define selekceZmetku
set compressor TXT_M_O8 on
python code
publishZmetek
set magnetic valve TXT_M_O7 on
wait s 1
set magnetic valve TXT_M_O7 off
set compressor TXT_M_O8 off

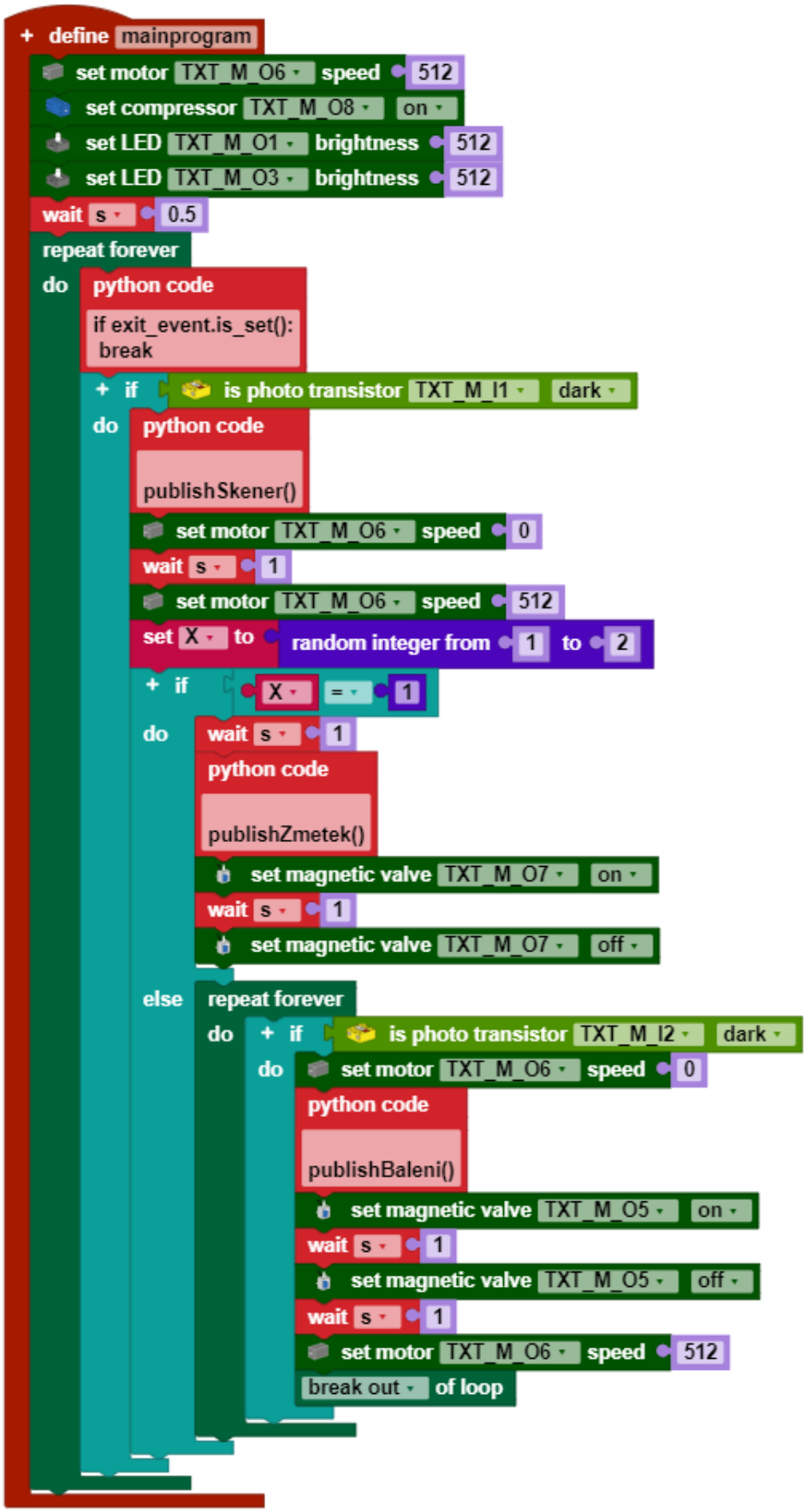

define baleni
set compressor TXT_M_O8 on
python code
publishBaleni
set magnetic valve TXT_M_O5 on
wait s 1
set magnetic valve TXT_M_O5 off
set compressor TXT_M_O8 off

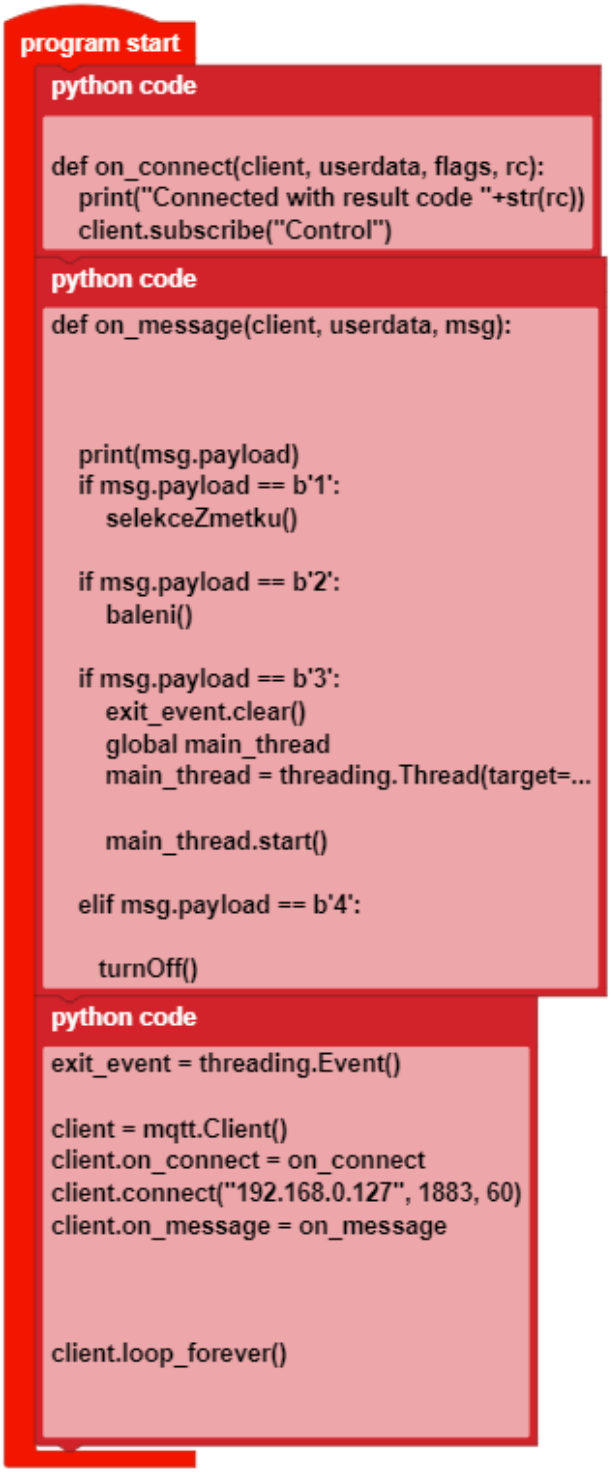

define turnOff
python code
exit_event.set()
set motor TXT_M_O6 speed 0
set compressor TXT_M_O8 off
set LED TXT_M_O1 brightness 0
set LED TXT_M_O3 brightness 0


define publishZmetek
python code
publish.single("Backmsg/RPi1", "Zmetek", h...


define publishBaleni
python code
publish.single("Backmsg/RPi1", "Baleni", hostn...


define publishSkener
python code
publish.single("Backmsg/RPi1", "Scan", hostnam...


define mainprogram
set motor TXT_M_O6 speed 512
set compressor TXT_M_O8 on
set LED TXT_M_O1 brightness 512
set LED TXT_M_O3 brightness 512
wait s 0.5
repeat forever
do
python code
if exit_event.is_set():
break
if is photo transistor TXT_M_I1 dark
do
python code
publishSkener()
set motor TXT_M_O6 speed 0
wait s 1
set motor TXT_M_O6 speed 512
set X to random integer from 1 to 2
if X = 1
do
wait s 1
python code
publishZmetek()
set magnetic valve TXT_M_O7 on
wait s 1
set magnetic valve TXT_M_O7 off
else
repeat forever
do
if is photo transistor TXT_M_I2 dark
do
set motor TXT_M_O6 speed 0
python code
publishBaleni()
set magnetic valve TXT_M_O5 on
wait s 1
set magnetic valve TXT_M_O5 off
wait s 1
set motor TXT_M_O6 speed 512
break out of loop


program start
python code
def on_connect(client, userdata, flags, rc):
print("Connected with result code "+str(rc))
client.subscribe("Control")
python code
def on_message(client, userdata, msg):
print(msg.payload)
if msg.payload == b'1':
selekceZmetku()
if msg.payload == b'2':
baleni()
if msg.payload == b'3':
exit_event.clear()
global main_thread
main_thread = threading.Thread(target=...
main_thread.start()
elif msg.payload == b'4':
turnOff()
python code
exit_event = threading.Event()
client = mqtt.Client()
client.on_connect = on_connect
client.connect("192.168.0.127", 1883, 60)
client.on_message = on_message
client.loop_forever()