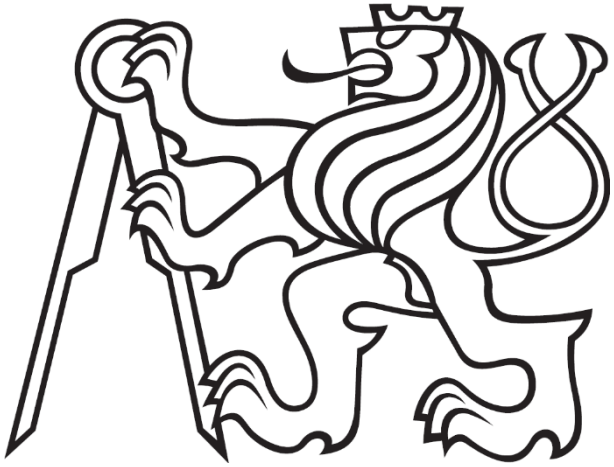


ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra elektrických pohonů a trakce

Rozbor komunikace a sběru dat ve zkušební lince

Analysis of the Communication and Data Collection in the Test Center

Diplomová práce

Studijní program: Elektrotechnika, energetika a management

Studijní obor: Elektrické pohony

Vedoucí práce: Ing. Pavel Kobrle, Ph.D.

Bc. Kryštof Kuzma

Praha 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kuzma** Jméno: **Kryštof** Osobní číslo: **457023**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektrických pohonů a trakce**

Studijní program: **Elektrotechnika, energetika a management**

Specializace: **Elektrické pohony**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Rozbor komunikace a sběru dat ve zkušební lince

Název diplomové práce anglicky:

Analysis of the Communication and Data Collection in the Test Center

Pokyny pro vypracování:

1. Vypracujte rešerši o problematice komunikace a sběru dat z pohledu IoT.
2. Zpracujte studii proveditelnosti obousměrného toku dat pomocí technologií National Instruments.
3. Zajistěte zpracování dat pomocí softwaru LabVIEW.
4. Zprovozněte management zpracovaných dat pomocí serverové aplikace SystemLink.

Seznam doporučené literatury:

[1] Vaculík, J:. Od telemetrie k internetu vecí I. Veci, siete, dáta a ich uloženie. EDIS, Žilina, 2019.
[2] Buyya, R., Dastjerdi, A.V.: Internet of Things. Principles and Paradigms. Burlington, Massachusetts: Morgan Kaufmann, 2016.
[3] Sinclair, B.: IoT Inc.. How Your Company Can Use the Internet of Things to Win in the Outcome Economy. New York: McGraw-Hill Education, 2017.
[4] National Instruments. SystemLink Architecture [online]. 15. 10. 2019. Dostupné z: http://www.ni.com/product-documentation/55045/en/

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Pavel Kobrle, Ph.D., katedra elektrických pohonů a trakce

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

Datum zadání diplomové práce: **09.02.2020** Termín odevzdání diplomové práce: **22.05.2020**

Platnost zadání diplomové práce: **30.09.2021**

Ing. Pavel Kobrle, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

Datum převzetí zadání	Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ....

podpis

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce Ing. Pavlu Kobrlemu, Ph.D. a vedoucímu Zkušebního centra AV R&D Ing. Jiřímu Jelínkovi za podnětné připomínky k obsahu práce v průběhu její tvorby. Dále bych chtěl poděkovat své rodině za morální a materiální podporu v průběhu studia.

Abstrakt

Cílem práce je seznámit čtenáře s problematikou internetu věcí a s jeho využitím realizovat konkrétní praktickou úlohu - monitoring zkušebního stavu. V úvodu je vysvětlen pojem internet věcí, je prezentován přehled topologií a druhů sítí, které pro účely internetu věcí vznikají, a jsou popsány přenosové technologie a komunikační protokoly společně s jejich zabezpečením. Jedna kapitola se věnuje studii proveditelnosti připojení zkušebního stavu do sítě internetu věcí, přičemž důraz klade zejména na přenos dat s využitím softwaru od společnosti National Instruments. Uvedeny jsou příklady typických způsobů přenosu dat mezi serverovou aplikací SystemLink a softwarem LabVIEW a zmíněny jsou také možnosti záznamu vizuální podoby průběhu zkoušky. Praktická část práce podrobně probírá způsob implementace přenosu dat do stávajícího softwaru zkušebního stavu. Rozbor je doplněn o části kódu a vývojové diagramy. V závěru jsou shrnuty výsledky práce.

Klíčová slova

Internet věcí, IoT, průmyslový internet věcí, IIoT, spotřebitelský internet věcí, CIoT, Bluetooth, WiFi, Celulární síť, Zigbee, HTTP, CoAP, MQTT, AMQP, DDS, SSL, TLS, Zwick 1873, LabVIEW, SystemLink, CompactRIO

Abstract

The focus of the thesis is to explain the concept of the Internet of Things and its implementation for the purpose of monitoring the test bench. In the beginning, the concept of Internet of Things is explained. Next the topologies and types of networks that are created for purpose of Internet of Things are mentioned. In addition, transmission technologies and communication protocols, together with their security, are introduced. Next part of the thesis examines possibilities of connecting the test bench to the Internet of Things. Main focus is put on data transfer using software from National Instruments. Typical methods of data transfer between SystemLink server application and LabVIEW software are mentioned. Part of the work is also dedicated to the possibilities of recording the image of the test bench. In the practical part, the implementation of data transfer to the existing test bench software is described. The description is completed with parts of code and flowcharts. In the conclusion, the results of the work are summarized.

Keywords

Internet of Things; IoT; Industrial Internet of Things; IIoT; Consumer Internet of Things; CIoT; Bluetooth; WiFi; Cellular network; Zigbee; HTTP; CoAP; MQTT; AMQP; DDS; SSL; TLS; Zwick 1873; LabVIEW; SystemLink; CompactRIO

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Průmysl moderní doby prochází velkým rozvojem. S tímto rozvojem je neodmyslitelně spjato i obrovské množství dat v elektronické podobě, které musí být zpracováno. Data jsou snímána v reálném čase, následně zpracovávána řídicí jednotkou a poté odesílána do úložiště. Aby bylo vyhověno nárokům na objem přenášených dat, je užíváno různých typů datových přenosů, konverzí, formátování, střádání a uchovávání.

Tato diplomová práce (dále již označení „práce“) se zabývá zejména monitorováním dat a jejich zpracováním. Jedním z trendů současnosti je management důležitých údajů pomocí budování sítí. Taková síť je obvykle nazývána internet věcí neboli *Internet of Things* (IoT). Pro průmyslové aplikace je tato síť označována jako *Industrial Internet of Things* (IIoT). Internet věcí využívaný ke zjednodušení každodenních úkonů běžného člověka je nazýván *Consumer Internet of Things* (CIoT).

Sítě IoT propojují uživatele s jednotlivými zařízeními, jejichž výstupem jsou inkriminovaná data určená k dalšímu zpracování. Každé z těchto zařízení je v takovéto síti jasně identifikovatelné. Zařízení je zpravidla schopné pracovat nezávisle na této síti a extrakce kýžených dat nikterak neomezuje jeho funkci. IoT, respektive IIoT sítě bývají v posledních letech zmiňovány zejména v souvislosti s průmyslem 4.0, lze je však využít také pro účely běžné veřejnosti. Za IoT lze tak například považovat správu domácích spotřebičů pomocí ethernetového spojení na lokální síti, což je v současné době věc poměrně běžná. Pojem IoT tak nelze jednoznačně omezit na konkrétní aplikace, jelikož zahrnuje obrovské množství aplikací automatizace. Definice internetu věcí bude podrobněji probrána v další části práce.

Jako příklad průmyslového nasazení IoT je možné uvést užití vysokozdvižných vozíků se sledováním pohybu. Tyto vozíky umožňují automatizovat například plnění zboží do velkých skladů. Vozíky v sobě mají zabudovány lokátory, díky nimž se dokáží pohybovat po skladových prostorách. Zvýšení bezpečnosti je dosaženo implementací lokátorů také do nositelných prvků pracovních oděvů. Automatizované vozíky tak vyhodnocují pohyb jednotlivých pracovníků a jsou schopny kromě automatizované práce také zamezit případné kolizi s pracovníky. Díky sledování pohybu vysokozdvižných vozíků je možné optimalizovat jejich pohyb a zvýšit úroveň bezpečnosti ve skladových prostorách. [1]

Tato práce je zaměřena zejména na průmyslovou aplikaci internetu věcí, tedy na IIoT. Cílem teoretické části je přiblížit tři klíčové části IoT, a sice hardware, middleware a software. Důraz je kladen také na přenosové protokoly, které jsou typicky užívány v IoT sítích.

Cílem studie proveditelnosti je navrhnout koncepční řešení realizace IoT pro účely Zkušebního centra AV R&D s.r.o. v Chrudimi. Podrobněji budou vymezeny a popsány nástroje, pomocí kterých bude vhodné IoT realizovat.

Praktická část vychází z poznatků studie proveditelnosti. Jejím cílem je zprovoznění komunikace mezi zkušebním stavem a serverovou aplikací SystemLink. K tomuto účelu bude využito softwaru od společnosti National Instruments, který je primárně určen právě pro průmyslové aplikace. Ke sběru dat je použita aplikace

vytvořená v softwaru LabVIEW. Tato aplikace je současně řídicím softwarem zkušebního stavu Zwick 1873. Data jsou shromažďována a upravena do požadované podoby a následně odeslána do serverové aplikace SystemLink. Komunikace mezi těmito softwarovými aplikacemi probíhá prostřednictvím datových paketů zvaných *Tagy* a pomocí souborů ve formátu TDMS. Pomocí serverové aplikace SystemLink jsou *Tagy* přijímány, střádány a vizualizovány na *dashboardu*. *Dashboard* je přístupný koncovým zařízením (iPad, notebook aj.), na kterých je prováděn vzdálený „on-line“ monitoring a ovládání funkce zkušebního stavu. Soubory TDMS slouží k podrobnější analýze probíhajících dějů.

V závěru práce je ověřena a zhodnocena kvalita IoT vytvořeného v praktické části. Ověření bude spočívat především v hodnocení kvality přenášených dat.

2 IoT

Pojem internet věcí, tedy IoT, je stěžejní téma mnoha webových stránek a odborných publikací. Po bližším seznámení-se s tímto tématem si však můžeme povšimnout, že neexistuje jednotná definice internetu věcí. Podle [2] je IoT rozšířením internetu za hranice počítačů a chytrých telefonů na celou řadu jiných věcí, které jsou určeny k shromažďování informací, odesílání informací, nebo k obojímu. S touto definicí se většina publikací ztotožňuje. Velice zjednodušeně lze tedy říci, že podstata internetu věcí spočívá v připojení oněch věcí k internetu. Zde je však patrný zásadní nedostatek jasné definice právě oněch věcí, které mají být připojeny k internetu. Vhodné vysvětlení nabízí například publikace [3] v následující podobě: *„Ačkoli by se mohlo zdát, že každý objekt, schopný připojení k internetové síti, by mohl spadat do kategorie „věcí“, toto označení je užíváno k zahrnutí obecnější skupiny entit, včetně „chytrých“ zařízení, senzorů, osob a dalších objektů, které si jsou vědomy svého kontextu a jsou schopny komunikace s ostatními entitami, což je tvoří dostupnými kdykoli a kdekoli“.* Obdobnou, avšak jednodušší, definici věcí nabízí také [4], v níž je „věc“ popsána následovně: „*Věc z pohledu IoT představuje neživý objekt (fyzický nebo virtuální) obsahující elektroniku, software a senzory, pomocí kterých snímá určitou veličinu nebo veličiny a poskytuje schopnost sloužit k danému účelu. Jedná se tedy o zařízení (systém), které autonomně poskytuje data (osobní počítač, který neposkytuje data, nepředstavuje věc z pohledu IoT), která jsou kabelově nebo bezdrátově sdílena s dalšími věcmi nebo systémy. Paradoxem však je, že v rámci Internetu věcí nejsou základem věci, ale data, která tyto věci poskytují.“*

Podstatou a podmínkou funkční IoT sítě je tedy připojení věcí k internetu. [5] Právě ono internetové spojení zajišťuje dostupnost nepřeberného množství dat. Tato data nemusí být zpracovávána, případně ukládána věcmi. Věcmi tak mohou být drobná zařízení s malým výkonem, například senzory. Cílem takového zařízení je poté pouze snímání požadovaných dat a předávání těchto dat do internetové sítě. Nutná je ovšem jasná identifikace takového zařízení na síti, aby bylo možné ke konkrétní věci přistupovat.

2.1 Identifikace zařízení

Identifikace je nezbytnou součástí každého prvku připojeného k síti. Identifikace je prováděna pomocí identifikátoru. Identifikátorem se rozumí stručná informace, která vymezuje konkrétní entitu. Podoba identifikátoru bývá zpravidla udávána ve strojovém formátu, který je pro člověka jen těžce čitelný. Výpočetní jednotka s takovým formátem však dokáže efektivně pracovat a jednoduše tak přesně určit umístění dané entity.

Pro účely IoT jsou identifikátory důležité zejména pro správné určení umístění zařízení v síti. Základním protokolem síťové vrstvy je internetový protokol (angl. *internet protocol*, zkráceně IP). IP slouží ke směrování paketů od odesílatele k příjemci. Skládá se z metadat a uživatelských dat. Zatímco uživatelská data přenáší kýženou informaci, metadata obsahují informace nutné pro správné doručení paketu. IP sítě přenáší data v blocích, nazývaných *datagramy*. Identifikátorem *datagramů* jsou IP adresy.

IP adresy jednoznačně identifikují síťové rozhraní v síti pracující s IP protokoly. Nejčastěji se můžeme setkat s IPv4 protokoly, které užívají 32bitovou IP adresu zapsanou dekadicky. IPv4 protokol je užit také v praktické části této práce. Na jeho základě probíhá komunikace mezi zařízeními na lokální síti. Vlivem nedostatku adres je postupně zaváděn IPv6 protokol, používající 128bitovou IP adresu zapsanou hexadecimálně. IPv4 adresa má podobu čtyř čísel oddělených tečkou. Jelikož se jedná o 32bitové číslo, může takovýchto adres existovat 2^{32}. IPv4 adresa přenáší tři základní informace, a to adresu sítě, adresu podsítě a adresu síťového rozhraní.

2.2 Hardware

Specifikace hardwaru určeného pro IoT aplikace vyplývá z požadavků kladených na tuto síť. V první řadě musí být umožněn spolehlivý sběr dat. Ideálně by toto zařízení mělo mít nízkou spotřebu a být co nejjednodušší. Požadavek na jednoduchost zařízení je však často vlivem složité aplikace nemožné splnit. Dalším mechanickým aspektem zařízení pro IoT aplikaci by měla být robustnost. Zařízení v IIoT síti by tak mělo být schopno vyhovět i požadavkům na bezproblémovou funkčnost v náročných podmínkách průmyslového nasazení. Sejmutá data jsou následně ukládána a případně zpracovávána. Můžeme tak provádět analýzu žádaných dat a jejich vyhodnocení přímo v příslušném hardwaru, který data sbírá. Z požadavku na zpracování dat přímo v zařízení, které je snímá, však plyne vyšší nárok na jeho výpočetní kapacitu. Právě z tohoto důvodu jsou data často zpracovávána až dodatečně. Data, ať už zpracovaná nebo surová, jsou následně sdílena pomocí internetu s dalšími hardwarovými prvky IoT. Data pak mohou být také publikována například na serveru, odkud jsou dostupná ostatním zařízením, které IoT sdružuje. V neposlední řadě je také kladen velký důraz na bezpečnost přenášených a střádaných dat. Často se totiž jedná o data obsahující důvěrné informace. Ideální IoT hardware si tak můžeme představit jako malou levnou komponentu připojenou k drahému zařízení, která je jakousi bránou do IoT sítě. Měla by komunikovat pomocí známých a bezpečných protokolů a měla by být vysoce spolehlivá a energeticky nenáročná.

Při snaze o tvorbu IIoT sítě můžeme narazit na problém, že zařízení nejsou tzv. IoT kompatibilní. Zařízení mohou být sestrojena tak, že vyžadují fyzickou přítomnost obsluhy a naměřená data zpracovávají výhradně analogově. V takovém případě je nutné stávající přístroj doplnit o hardware umožňující naměřená data převést do digitální podoby a v této podobě s nimi pak nadále pracovat. Prakticky se tak jedná o nejkomplikovanější připojení stávajícího hardwaru do IoT sítě. Často je totiž nutný mechanický zásah do stávajícího zařízení. Úpravu zařízení tak často musí provádět odborník a nestačí pouhá znalost práce s ovládacími prvky přístroje.

Průmyslová zařízení dnešní doby jsou však nejčastěji přístroje, které již jistý druh automatizace obsahují. Jedná se například o různé kontroléry a PLC. V takových případech je nutné starší technologie doplnit o zařízení, které dokáže se stávajícím zařízením komunikovat pomocí jednoho ze standardních průmyslových komunikačních protokolů. Většina výrobců průmyslových zařízení si potřebu aplikace IIoT dnes již uvědomuje. Snaží se tak o produkci právě oněch drobných doplňkových zařízení, která jsou kompatibilní s jejich původním hardwarem a umožňují takové zařízení zpětně připojit do IoT sítě. Pokud však výrobce takovouto úpravu nenabízí, je nutné

management dat zpětně doplnit. V takovém případě je nutné vhodně využít komunikačních protokolů, pomocí kterých je možná většinou obousměrná komunikace se zařízením. Tato varianta je však komplikovanější, neboť je nutné management dat nějakým způsobem realizovat. Nadneseně pak lze takovýto IoT hardware chápat jako jakousi redukci na průmyslový komunikační protokol.

Nastat však může také případ, že přístroj je tvořen volně programovatelnou řídicí jednotkou, umožňující udělat přístroj se sítí IoT zpětně kompatibilní pomocí přeprogramování softwaru. Jako příklad takového přístroje lze uvést CompactRIO od firmy National Instruments. Jedná se o řídicí jednotku koncipovanou pro průmyslové aplikace. Umožňuje mimo jiné připojení k síti pomocí ethernetového rozhraní. Zpětná kompatibilita s IoT sítí tak není nikterak komplikovaná a data lze poměrně jednoduše střádat i odesílat. Takové řešení kompatibility s IoT je optimální a není třeba do procesu managementu dat zapojovat další hardware.

Velkým problémem při snaze o zavedení IIoT je také vzájemná kompatibilita jednotlivých zařízení v takovéto síti. Hardware bývá zpravidla velmi nesourodý. Vedle sebe tak mohou fungovat staré a nové technologie. Jednotliví výrobci u svých zařízení také často používají různé komunikační protokoly, které nemusí být vzájemně kompatibilní. Často nastává také situace, kdy jsou komunikační protokoly proprietární či záměrně nekompatibilní s konkurenčním zařízením. Dalším problémem je fakt, že většina zařízení je z výroby připravena pouze pro komunikaci na lokální síti. Z hlediska bezpečnosti tak neumožňují přístup na internet.

Snaha o zprovoznění IIoT sítě tedy většinou spočívá v homogenizaci komunikačních protokolů a ve snaze o vzájemnou kompatibilitu zařízení, která chceme pod IIoT sítí sdružit. Jedině systém zařízení se vzájemně kompatibilními komunikačními protokoly lze sjednotit pod jednou sítí a tuto síť vhodně spravovat.

2.3 Typy IoT spojení

V první fázi života dat jsou daná data sejmuta IoT zařízením. Jelikož toto zařízení nebývá obvykle určeno k vyhodnocení a střádání sejmutých dat, je nutné data dále přesouvat. Jednotlivá zařízení takto mohou data odesílat různým příjemcům. Výčet možných příjemců a odesílatelů dat je blíže rozebrán v následující kapitole.

2.3.1 Přenos dat mezi dvěma IoT zařízeními

První možností je přenos dat mezi dvěma zařízeními. Příkladem může být přenos dat mezi dvěma IoT zařízeními, která jsou svázána právě datovými protokoly umožňujícími jejich vzájemnou komunikaci. Data bývají zpravidla přenášena nepřetržitě a je kladen důraz na možnost přenosu dat v co nejrychlejším čase. Nutno však podotknout, že tento druh komunikace není v současnosti příliš běžný. Zařízení totiž nejsou na vzájemnou komunikaci často připravena.

2.3.2 Přenos dat mezi IoT zařízením a úložištěm

Druhou možností je požadavek na přenos dat do datového centra, případně do *cloudového* úložiště. Tento přenos nebývá zpravidla realizován přímo, ale přes

tzv. datové brány. Datové brány si lze představit jako jakýsi portál, který daná data přesměruje od zařízení, které je sejmulo, do datového centra případně *cloudového* úložiště, kde jsou data střádána. Tok dat lze tedy rozdělit na komunikaci mezi zařízením a datovou bránou, následně na komunikaci mezi bránou a daným úložištěm. V datovém úložišti jsou data střádána a lze je následně v libovolném čase zpracovat.

2.3.2.1 Komunikace mezi IoT zařízením a datovou bránou

Komunikace mezi IoT zařízením a datovou bránou lze v podstatě chápat jako přenos dat od jednoduchého zařízení k jakémusi portálu, jehož účelem je přesměrování datového toku. Při komunikaci mezi IoT zařízením a datovou bránou dochází také k selekci dat. Pokud se v této fázi přenosu vyskytne problém při sběru dat, data jsou navrácena zpět do zařízení, které je vyslalo, případně jsou ignorována. Datové brány také mohou sloužit k zefektivnění bezpečnosti přenosu dat. Mohou být jakýmsi spojem lokální sítě s internetem. Zatímco na lokální síti mohou být data přenášena bez zabezpečení, na internetu je s určitými daty nutno nakládat opatrněji a je nezbytné jejich šifrování.

2.3.2.2 Komunikace mezi datovou bránou a úložištěm

Volba vhodného protokolu pro komunikaci mezi datovou bránou a datovým centrem případně *cloudovým* úložištěm je podmíněna především dostatečnou znalostí objemu dat a frekvence, s jakou je nutné data přesouvat.

2.3.3 Přenos dat mezi úložišti

Poslední možností je přenos dat přímo mezi datovými centry, resp. *cloudovými* úložišti. Tento datový přenos probíhá nejčastěji pomocí datových protokolů, které jsou součástí prostředí datového centra případně *cloudového* úložiště. Pokud je vyžadován tento typ přenosu dat, měl by být výběr úložiště podmíněn znalostí dostupných a z bezpečnostního hlediska přijatelných možností manipulace s daty.

2.4 Typy IoT sítí

V závislosti na maximální vzdálenosti, na jakou jsou schopna dvě zařízení pomocí určitého protokolu komunikovat, a na způsobu, jakým je komunikace využívána, lze IoT sítě dělit do několika následujících skupin. [6]

2.4.1 *Nanonetwrok*

Jedná se o síť tvořenou miniaturními zařízeními velikosti maximálně v řádu několika mikrometrů. Tato zařízení jsou schopna provádět pouze velmi jednoduché úkony, jako například snímat data. Uplatnění lze hledat například v nanotechnologiích.

2.4.2 NFC (*Near-Field Communication*)

NFC technologie umožňuje bezdrátovou komunikaci na vzdálenost do 4 cm. Využití nachází především v aplikaci různých čipových karet případně jako doplňková technologie tzv. „chytrých“ zařízení.

2.4.3 BAN (*Body Area Network*)

Síť BAN je určena pro nositelná zařízení. Zařízení mohou být odnímatelná nebo neodnímatelná. Aplikaci neodnímatelných zařízení lze nalézt například v medicínském průmyslu, kde hovoříme o MBAN (*Medical* BAN) síti. Příklad neodnímatelného zařízení pracujícího v MBAN síti může být kardiostimulátor. Jako příklad odnímatelného zařízení pracujícího v BAN či MBAN síti lze uvést různé chytré náramky, které střádají informace o jeho nositeli. Síť BAN spojuje takovéto zařízení se zařízením umožňujícím vyhodnocení střádaných dat. Příkladem může být spojení chytrého náramku s mobilním telefonem umístěným v kapse.

2.4.4 PAN (*Personal Area Network*)

PAN síť je určená k propojení zařízení na pracovišti uživatele. V závislosti na konkrétní technologii umožňuje spojit zařízení v několika málo místnostech jednoho objektu. Příkladem může být propojení mobilního telefonu s počítačem a televizí.

2.4.5 LAN (*Local Area Network*)

Jedná se o přenosovou síť určenou k přenosu dat v rámci jedné budovy. Typickým příkladem může být podniková síť LAN spojující počítače v jedné firmě.

2.4.6 CAN (*Campus/Corporate Area Network*)

CAN síť spojuje zařízení podobně jako LAN. Na rozdíl od LAN sítě však CAN spojuje větší prostory, ve kterých jsou podružné LAN sítě. Typickým příkladem jsou kampusy univerzit, kde jsou pomocí CAN sítě spojeny všechny univerzitní budovy do jednoho komunikačního celku.

2.4.7 MAN (*Metropolitan Area Network*)

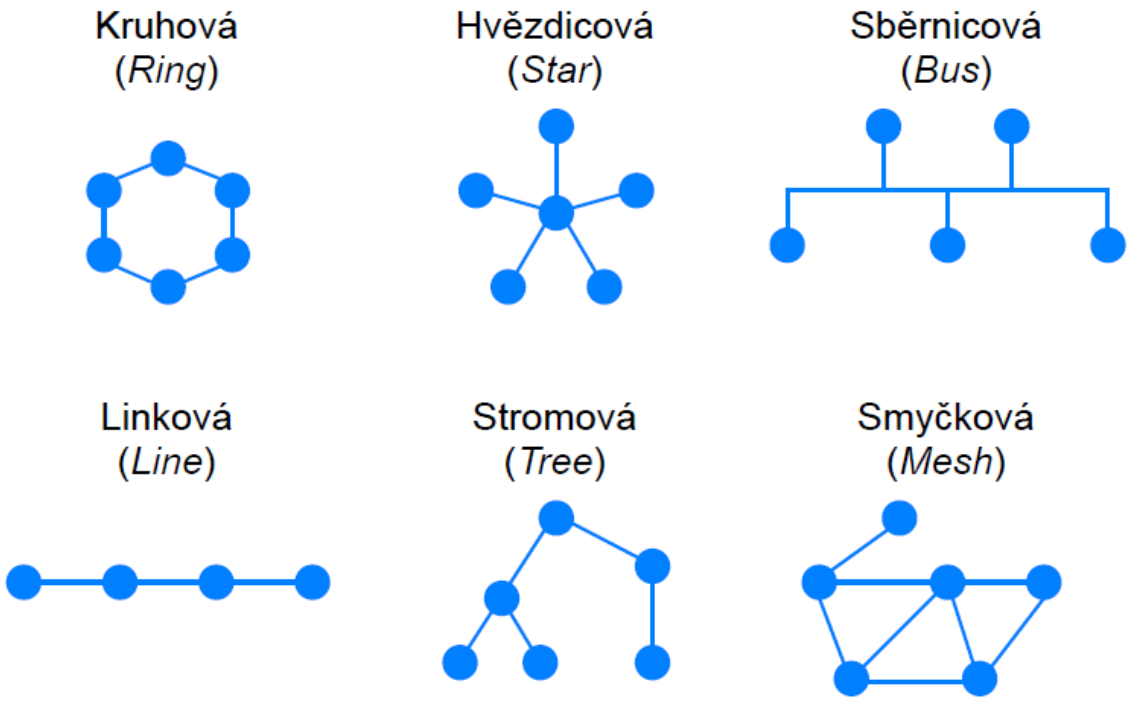
MAN síť spojuje výpočetní zařízení v geografické oblasti odpovídající svou velikostí například městu. Jedná se o rozsáhlejší síť než CAN.

2.4.8 WAN (*Wide Area Network*)

WAN představuje síť spojující velké geografické oblasti. V síti WAN jsou obsaženy podružné sítě LAN, MAN a případně CAN.

2.5 Fyzická topologie IoT sítí

Jednotlivá zařízení v síti je nutné vzájemně propojit. Způsob, jakým jsou zařízení propojena, tedy topologie sítě, může mít různou podobu a z toho pramenící vlastnosti. Různé topologie sítí jsou patrné z obrázku 1.



Obrázek 1: Topologie IoT sítí

2.5.1 Kruhová (*Ring*)

Při této topologii sítě je každé zařízení spojeno se dvěma dalšími. Poslední dvě zařízení jsou spolu propojena tak, že celý systém tvoří kruh. Přenos dat je jednoduchý a nevznikají kolize. Odesílaná data jsou postupně předávána mezi zařízeními v jednom směru. Každé zařízení v síti funguje jako opakovač a zesiluje signál. Náklady na tvorbu takovéto sítě bývají zpravidla poměrně nízké. Nevýhodou je fakt, že data musí projít přes každý uzel mezi odesílatelem a příjemcem, což přenos prodlužuje. V případě výpadku jednoho uzlu je ochromena celá síť. Spolehlivost přenosu a robustnost sítě je tak poměrně malá a v některých aplikacích se užívá záložní kruhové spojení. Síť s kruhovou topologií je realizována pomocí koaxiálního kabelu a optických vláken technologií *Token-ring*.

2.5.2 Hvězdicová (*Star*)

V síti s hvězdicovou topologií je každé zařízení připojeno k centrálnímu prvku, tedy rozbočovači nebo přepínači. Odeslaná data jsou k dispozici všem zařízením, ale přijmout je může pouze adresát. Spoje jednotlivých zařízení jsou na sobě nezávislé a při výpadku jednoho zařízení není ochromena celá síť, ale pouze spojení s daným zařízením. Lze tedy velice snadno nalézt místo závady. Selhání centrálního prvku však zapříčiní výpadek celé sítě, a proto tento prvek bývá před výpadkem chráněn záložním zdrojem. Síť s hvězdicovou topologií se velice jednoduše rozšiřuje o nová zařízení. Hvězdicová topologie je nejběžněji užívanou topologií. Tato topologie je nejčastěji užívána s technologií *Ethernet*.

V síti s touto topologií je realizována praktická část práce, jejímž stěžejním prvkem je přenos dat na lokální síti pomocí ethernetového spojení jednotlivých komponent.

2.5.3 Sběrnicová (*Bus*)

Síť se sběrnicovou topologií je realizována pomocí jednoho souvislého úseku kabelu, na který jsou jednotlivá zařízení připojována pomocí spojek. Síť je ukončena pomocí terminátorů, které zabraňují odražení signálu. Informace je vyslaná z jednoho zařízení a obdrží ji všechna zařízení současně, ale pouze adresát informace ji může nadále zpracovávat. Výhodou je jednoduchá realizace sítě a připojení dalších zařízení. Nevýhodou je fakt, že při porušení centrálního spoje dojde k výpadku celé sítě a také omezení délky centrálního spoje a počtu připojených zařízení. Sběrnicová síť je realizována pomocí technologie *Ethernet*.

2.5.4 Linková (*Line*)

Linková topologie má stejné vlastnosti jako sběrnicová topologie. Data z jednoho zařízení jsou odesílána do všech ostatních, přičemž přijmout data může pouze adresát.

2.5.5 Stromová (*Tree*)

Stromová topologie je složena z několika hvězdicových topologií, jejichž centrální prvky jsou vzájemně spojeny. Síť se stromovou topologií má stejné vlastnosti jako síť s hvězdicovou topologií. Sítě se stromovou topologií se obvykle užívá pro účely velkých firem.

2.5.6 Smyčková (*Mesh*)

Smyčková síť se vyskytuje v několika variacích. Je-li v síti propojeno každé zařízení s každým, jedná se o plnou smyčkovou topologii. Jsou-li některé spoje v síti vynechány, mluvíme o částečné smyčkové topologii. Výhodou smyčkové topologie je vysoká spolehlivost. Při výpadku jednoho spoje jsou data dodána jinou cestou, přes ostatní zařízení. Této topologie je užíváno zejména v bezdrátových sítích, jelikož spotřeba kabelů kabelové sítě s touto topologií by byla značná.

2.6 Komunikační technologie

Komunikační technologie v IoT zajišťují datový tok mezi zařízeními a umožňují tak samotný přenos dat. Ačkoli lze využívat technologie kabelové, trendem je přenos dat zejména pomocí bezdrátových komunikačních technologií. Komunikační technologie jasně vymezují limity dané IoT aplikace, kterými jsou zejména zabezpečení datového přenosu a objem přenesených dat.

Pro účely IoT aplikací je využíváno velké množství komunikačních technologií. Při bližším prozkoumání jejich výčtu bychom zjistili, že jsou často využívány technologie, které jsou již poměrně staré a nevznikly přímo pro potřeby IoT aplikací. IoT se jakožto poměrně mladé odvětví automatizace zprvu muselo spokojit pouze s využitím právě starších, již dobře zavedených, komunikačních technologií. Omezení IoT aplikací těmito technologiemi nespočívalo ani tak v nedostatečné přenosové kapacitě, bezpečnosti či vzdálenosti, jako spíše v energetické náročnosti. Právě z toho důvodu je kladen důraz na vývoj nových komunikačních technologií, které požadavky IoT aplikací

uspokojí lépe. Tyto nové technologie vznikají často pouze úpravou technologií stávajících. Příkladem může být třeba komunikační technologie *Bluetooth Low Energy* (BLE), která využívá dobře známé technologie *Bluetooth* a její optimalizací dociluje nižší spotřeby energie.

Následující podkapitoly popisují vybrané komunikační technologie, které jsou v dnešní době hojně využívány pro IoT aplikace.

2.6.1 *Bluetoot*h

Bluetooth je bezdrátová technologie definovaná otevřeným standardem IEEE 802.15.1, která je využívána pro bezdrátovou komunikaci. Stejně jako WiFi pracuje ve frekvenčním pásmu 2,4 GHz. Vytvořena byla v roce 1994 firmou Ericsson. *Bluetooth* technologie vznikala postupem času v několika verzích, které se liší především rychlostí přenosu. Rozdíl mezi jednotlivými verzemi je patrný z tabulky 1. [7]

Tabulka 1: *Bluetooth* verze

Verze	**Rok vydání**	**Rychlost přenosu (Mbit/s)**	**Standardní dosah (m)**
1.0	1999	0,721	~100
2.0	2004	2.1	~10
3.0	2009	24	~10
4.0	2010	1	~50
5.0	2016	0,125/0,5/1/2	~240
5.2	2020	0,125/0,5/1/2	~240

Z tabulky 1 je jasně patrné, že mezigenerační rozdíl nespočívá vždy v navýšení rychlosti. Poslední generační změny *Bluetooth* spočívají převážně ve značném snížení spotřeby technologie a také v navýšení přenosové vzdálenosti. Údaje standardního dosahu jsou uvedeny pro volné prostranství. Při použití zařízení v budově dochází ke značnému omezení přenosové vzdálenosti vlivem objektů umístěných v cestě signálu. Z přehledu je také patrné, že poslední generace *Bluetooth* má čtyři možné rychlosti přenosu. Přenosová rychlost je volitelná a vždy zde platí, že čím nižší rychlost přenosu, tím vyšší dosah technologie. V praxi je přenos pomocí *Bluetooth* technologie využíván na kratší vzdálenosti především v sítích typu PAN.

Bluetooth podporuje dva typy komunikace. Prvním typem je dvoubodová komunikace typu *point to point*, která přímo spojuje dvě zařízení. Druhým typem je vícebodová komunikace, kde je vždy jedno zařízení typu *master*. Ostatní zařízení typu *slave* jsou připojena na *master* zařízení. Takovéto seskupení se nazývá *Piconet*.

Zabezpečení *Bluetooth* přenosu spočívá v párování zařízení. Každé zařízení má unikátní 48 bitovou adresu, pod kterou k němu lze přistupovat. Při požadavku na přenos dat pomocí *Bluetooth* je nutné zařízení nejprve spárovat. Při procesu párování je potvrzována totožnost obou zařízení a zároveň je nutné zadat vygenerované heslo. Takto spárovaná zařízení spolu pak mohou komunikovat.

V odvětví IoT se uplatňuje zejména odnož Bluetooth s názvem *Bluetooth Low Energy*, nebo *Bluetooth Smart*. Poprvé bylo uvedeno v roce 2006 a v roce 2010 bylo sloučeno do hlavního standardu *Bluetooth 4.0*. V porovnání se standardním *Bluetooth* má BLE značně sníženou spotřebu. Díky tomu šetří baterii zařízení, ve kterém je užíváno. Přesto je zachován komunikační dosah. Díky svým vlastnostem nachází *Bluetooth* uplatnění v různých nositelných technologiích, jako jsou například chytré hodinky a ve zdravotnictví.

2.6.2 WiFi

WiFi je bezdrátová technologie, využívající standard IEEE 802.11. Ke svému provozu využívá bezlicenční nekoordinované frekvenční pásmo. Typicky se jedná o 2,4 GHz, avšak v poslední době se čím dál tím více pro běžné aplikace uplatňuje pásmo 5 GHz. Standardů IEEE 802.11 v průběhu let vzniklo několik. Rozdíl mezi nimi je zejména v maximální rychlosti přenosu a také ve frekvenčním pásmu, na kterém fungují. Rozdíly mezi jednotlivými verzemi standardu jsou patrné z tabulky 2. [8]

Tabulka 2: IEEE 802.11 standardy

Standard	**Označení**	**Rok vydání**	**Pásmo (GHz)**	**Maximální rychlost (Mbit/s)**
Původní IEEE 802.11	-	1997	2,4	2
IEEE 802.11a	WiFi 1	1999	5	54
IEEE 802.11b	WiFi 2	1999	2,4	11
IEEE 802.11g	WiFi 3	2003	2,4	54
IEEE 802.11n	WiFi 4	2009	2,4/ 5	600
IEEE 802.11y	-	2008	3,7	54
IEEE 802.11ac	WiFi 5	2013	2,4/5	3466,8
IEEE 802.11ad	-	2012	60	6757
IEEE 802.11ax	WiFi 6	2019	2,4/5/6	10530

Dosah WiFi sítí se různí pro jednotlivé generace. Velkou roli na dosah sítě však mají objekty, které stojí v cestě signálu. Lze říci, že WiFi signál bude mít větší dosah na volném prostranství než uvnitř budovy. Maximální dosah standardní WiFi sítě pracující na frekvenci 2,4 GHz je uvnitř budovy okolo 50 m a ve volném prostranství zhruba 100 m. [9] Přístroje jsou také díky poměrně nízkému výkonu velmi citlivé na rušení. S rostoucí vzdáleností od zdroje signálu, případně s rostoucím množstvím překážek v cestě signálu, klesá síla signálu. Poklesu síly signálu odpovídá také pokles přenosové rychlosti.

Sítě IEEE 802.11 mohou pracovat v několika režimech, respektive mohou mít několik struktur. Rozlišení struktury sítě je dáno SSID identifikátorem, který je jednotlivými zařízeními vysílán. Prvním je režim *ad hoc*. *Ad hoc* je vhodný zejména pro běžné malé sítě, do kterých je připojeno zhruba 2 až 5 zařízení, kdy jednotlivá zařízení mohou komunikovat mezi sebou. Klienti jsou si tedy rovni. Druhým režimem je

infrastrukturní režim. Je zde využito klasického vztahu klient-server. Serverem je zde zařízení, umožňující komunikovat se všemi ostatními zařízeními tzv. AP (*Access Point*). Komunikace poté probíhá pouze s AP zařízením, které řeší předávání dat mezi jednotlivými klienty. Dalším režimem je režim *bridge*. V tomto režimu jsou bezdrátově spojeny dvě drátové sítě LAN. Je nutné využít dva shodné AP, které danou funkci podporují. Posledním, trochu atypickým režimem, je režim, ve kterém se AP chová jako klient k jinému AP. Přístupové body lze v jednotlivých režimech využít pro připojení lokální sítě k různým službám, typicky k internetu.

Hlavní výhoda WiFi technologie tkví v její příznivé ceně, která je daná především velkým množstvím vyrobených zařízení s požadovanou certifikací.

2.6.2.1 *WiFi HaLow*

Ve spojitosti s IoT lze zmínit druh WiFi technologie s názvem *WiFi HaLow*. Tento druh WiFi pracuje se standardem IEEE 802.11ah. *WiFi HaLow* pracuje na frekvenci 900 MHz, na rozdíl od WiFi pracujících s běžnými standardy, které typicky využívají frekvence 2,4–5 GHz. Pracovní frekvence 900 MHz zajišťuje snadnější průnik signálu překážkami. Je dosaženo prodlouženého rozsahu, a to až na hodnotu 1 km. Významně je také zredukována energetická náročnost komunikace pomocí *WiFi HaLow*, což usnadňuje bateriový provoz. Přenos dat probíhá v dávkách nikoliv kontinuálně. *WiFi HaLow* je přímým konkurentem *Bluetooth*, avšak oproti *Bluetooth* má větší dosah. [10]

2.6.3 Celulární síť

Celulární síť je rádiová telekomunikační síť. Tuto síť však lze vhodně využít také pro účely IoT aplikací.

Komunikace je v rozlehlé oblasti zajišťována množstvím základních stanic, které vytvářejí soustavu vzájemně se překrývajících buněk. Poloměr buněk se může značně lišit, a to v rozsahu přibližně 1 až 30 km. Spojením jednotlivých buněk je zajištěno pokrytí velké oblasti. Celulární sítě pracují s frekvencemi od 300 MHz do 3 GHz. Jednotlivé buňky mezi sebou navzájem komunikují a umožňují tak přenos signálu napříč celým celulárním systémem. Každá buňka je tvořena přijímačem a vysílačem. V rámci jedné sítě je nutné používat různé frekvence. Aby nedošlo k rušení, musí sousední buňky používat vždy různé frekvence. Dvě vzdálenější buňky však mohou používat frekvence stejné a často je využito opakování frekvence. V praxi se také frekvence volí v závislosti na charakteru pokrývané oblasti. V oblasti s velkým osídlením je vhodné použít vyšší frekvence, které mají velkou kapacitu. Jejich nevýhodou je však pokrytí menší oblasti. V řidčeji osídlených oblastech je vhodné využít nižší frekvence, které mají sice menší kapacitu, avšak pokrývají větší oblast. V průběhu let se technologie celulárních sítí měnila. Rozdíly mezi jednotlivými generacemi jsou patrné z tabulky 3. [11]

Tabulka 3: Technologie celulární sítě

Označení	Rok vydání	Maximální rychlost (Mb/s)
1G	1980	0,0024
2G	1990	0,064
3G	2003	2
4G	2009	100
5G	2020	1000

Předností technologie je především přenos dat na velké vzdálenosti. Umožněn je poměrně velký datový tok, avšak komunikace je výkonově poměrně náročná. Není tedy vhodná pro drobná, bateriově napájená zařízení, u kterých nelze baterii v krátkých časových intervalech vyměňovat.

2.6.4 *ZigBee*

ZigBee je bezdrátová komunikační technologie platná od roku 2004. Jejím základem je standard IEEE 802.15.4.

ZigBee pracuje na frekvenci 868 MHz při přenosové rychlosti 20, 40 a 250 kbit/s. Díky pracovní frekvenci netrpí na rušení od WiFi případně *Bluetooth*. Umožňuje bezdrátové spojení nízko-výkonových zařízení na malé vzdálenosti, nejčastěji v rozsahu sítě PAN. Tato technologie je schopna přenášet data na vzdálenosti do 100 metrů v otevřeném prostoru a přibližně do 50 metrů uvnitř budov. *ZigBee* je schopno komunikace i bez přímé radiové viditelnosti, a to díky multiskokovému ad-hoc směrování. Většinou jsou *ZigBee* zařízení spojena do sítě s topologií *mesh*. Jednotlivá zařízení v takové síti slouží jako opakovače a umožňují propojení i rozsáhlejších objektů. [12]

Zabezpečení *ZigBee* spočívá v použití AES (*Advanced Encryption Standard*) s klíčem o délce 128 bitů. Šifrována jsou data rozdělená do bloků a k šifrování i dešifrování se používá stejný klíč.

ZigBee standard se primárně uplatňuje ve spotřební elektronice. Technologie *ZigBee* je hojně podporována např. termostaty, alarmy, klimatizacemi a nejrůznějšími čidly. Výjimkou však nejsou ani průmyslové aplikace. Její hlavní přednosti jsou spolehlivost, nízká spotřeba, snadná implementace a příznivá cena.

2.7 Komunikační protokoly

Funkční síť IoT musí být kromě vhodně zvoleného hardwaru, kterému je věnována kapitola 2.2, podložena také vhodně zvolenými komunikačními protokoly. Jelikož v síti IoT jde především o práci s daty, jsou komunikační protokoly alfou a omegou tvorby těchto sítí. Uplatňují se jak léta známé a zavedené komunikační protokoly, které vyhovují IoT požadavkům, tak zcela nové komunikační protokoly. Tyto protokoly vznikly právě pro naplnění požadavků na přenos dat v IoT sítích.

Požadavky na přenosové protokoly v IoT sítích lze částečně zobecnit. Přenosové protokoly by měly být především dobře zabezpečené, aby nedocházelo k nechtěnému úniku dat. V síti IoT jsou často přesouvána citlivá data, která jsou klíčová pro daného uživatele. Dalším důležitým kritériem je energetická nenáročnost. Jak již bylo zmíněno v kapitole 2.2, hardware, který obstarává připojení zařízení do IoT sítě, by měl být zpravidla energeticky nenáročný a poměrně kompaktní. Ve snaze docílit těchto parametrů by měl být hardware podpořen také vhodným komunikačním protokolem, který daný hardware nebude příliš zatěžovat a obstará potřebnou manipulaci s daty. Tento požadavek se samozřejmě různí s různým výkonem zařízení, mezi kterými mají být data přesouvána. Dále jsou jednotlivé komunikační protokoly vybírány především podle požadavků konkrétní aplikace potažmo zvolené komunikační technologie, která bude s protokoly pracovat. Důležitými kritérii při výběru komunikačních protokolů jsou tak například způsob jejich šifrování, nebo přenesený objem dat za jednotku času.

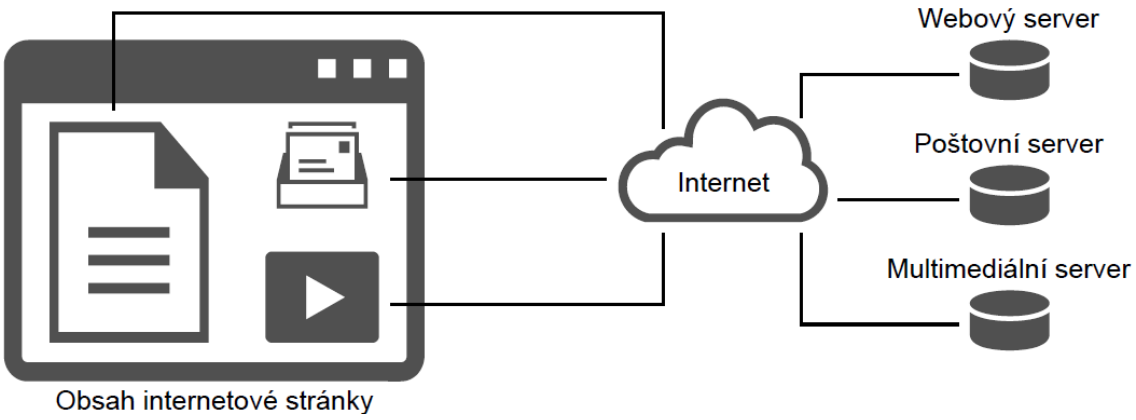
V následujících podkapitolách je popsáno několik protokolů typických pro IoT aplikace.

2.7.1 HTTP (*HyperText Transfer Protocol*)

Nejstarší verze http pochází z roku 1991 a nese označení 0.9. HTTP 0.9. Tato verze byla rychle nahrazena verzí HTPP 1.0, jejíž popisy byly publikovány již v roce 1992 a definitivně byla vydána v roce 1996. Další verzí je HTTP 1.1 vydaná v roce 1999. Následující verze (HTTP 1.2) byla pouze experimentální a v praxi se nevyužívala. V roce 2015 vznikla verze HTTP 2.0, která je v současnosti nejrozšířenější. Její devizou je zejména zrychlování přenosu dat. V současnosti je vyvíjena verze HTTP 3, která byla uveřejněna roku 2018. [13]

HTTP je internetový protokol. Užívá se v případě, kdy je potřeba publikovat velké množství dat a je určen pro komunikaci s WWW servery. HTTP protokol funguje na základě komunikace mezi klientem a serverem. Klientem je nejčastěji internetový prohlížeč. HTTP server je pak program spuštěný na počítači v serverovně, běžně označovaném jako server. Jako příklad http serveru může být program Apache. HTTP protokol lze popsat jako jazyk, ve kterém komunikují klient a server. Tato komunikace probíhá nejčastěji po internetové síti. Příklad komunikace lze demonstrovat na snaze fyzické osoby o zobrazení obsahu internetové stránky. Osoba zadá do prohlížeče URL adresu stránky. Prohlížeč je v tomto případě klientem, který danou URL adresu zpracuje a přes DNS zjistí, jaké IP adresy se má ptát. Přes TCP protokol naváže spojení se serverem na zjištěné IP adrese. V této fázi začíná samotná HTTP komunikace. Prohlížeč pošle na server volání s metodou GET, která definuje požadavek na získání obsahu webové stránky. Server následně může vrátit požadovaný dokument/ oznámit, že daný dokument je jinde/ oznámit problém. V ideálním případě jsou data klientovi poskytnuta a v našem případě (klientem je internetový prohlížeč) klientem zobrazena. Zjednodušený model http komunikace je patrný z obrázku 2. [14]

Zabezpečení HTTP komunikace je realizováno pomocí HTTPS URI nebo pomocí nadstavby HTTP 1.1. HTTPS URI má stejný syntax jako HTTP. Přidává pouze signalizaci klientovi, aby použil šifrovací metodu SSL/TLS. V případě zabezpečení pomocí HTTP 1.1 začíná klient komunikovat prostým textem, který je později nahrazen TLS. Zabezpečení komunikace může vyžádat klient nebo server.



Obrázek 2: Komunikační model http

HTTP je velice rozšířený protokol, pro účely IoT však není vždy tou nejlepší volbou. Komunikace pomocí http protokolu je určena pouze pro komunikaci mezi dvěma systémy najednou. Ačkoli tento fakt není překážkou pro účely webových stránek, pro IoT aplikace to není zcela vhodné. V IoT síti je totiž třeba často zpracovávat data z více zdrojů najednou, jinak by mohlo dojít k jejich ztrátě. Komunikace s více zařízeními také zapříčiňuje velké zatěžování serveru, který musí s každým zařízením navázat zvlášť spojení. Je tak kladen velký nárok na výpočetní výkon serveru. Další nevýhodou je jednosměrnost HTTP komunikace. V daném čase může žádost posílat pouze jedno ze zařízení, která mezi sebou komunikují. Nevýhodou je také prodleva po zaslání dané žádosti, jelikož tato žádost musí být serverem zpracována, což vede ke zpomalování přenosu dat. HTTP protokol také není navržen pro komunikaci na základě události (*event-base*). Pro vyvolání komunikace je třeba zaslat požadavek (*request*). Jedná se tedy o tzv. *request-base* komunikaci. Jelikož většina IoT aplikací je *event-based*, stává se realizace komunikace pomocí HTTP protokolu poměrně náročnou. Poslední velká nevýhoda je poměrně velká energetická náročnost HTTP komunikace. Nízkovýkonová zařízení tak s touto komunikací mohou mít problém a např. bateriový provoz je nevhodný. [15]

2.7.2 CoAP (*Constrained Application Protocol*)

CoAP je stejně jako http protokol určený k přenosu dokumentů. Komunikace je typu klient-server. Klient zasílá požadavek na server, který následně odpovídá. Komunikace probíhá mezi dvěma zařízeními. Princip komunikace je tedy obdobný, jako v případě HTTP protokolu.

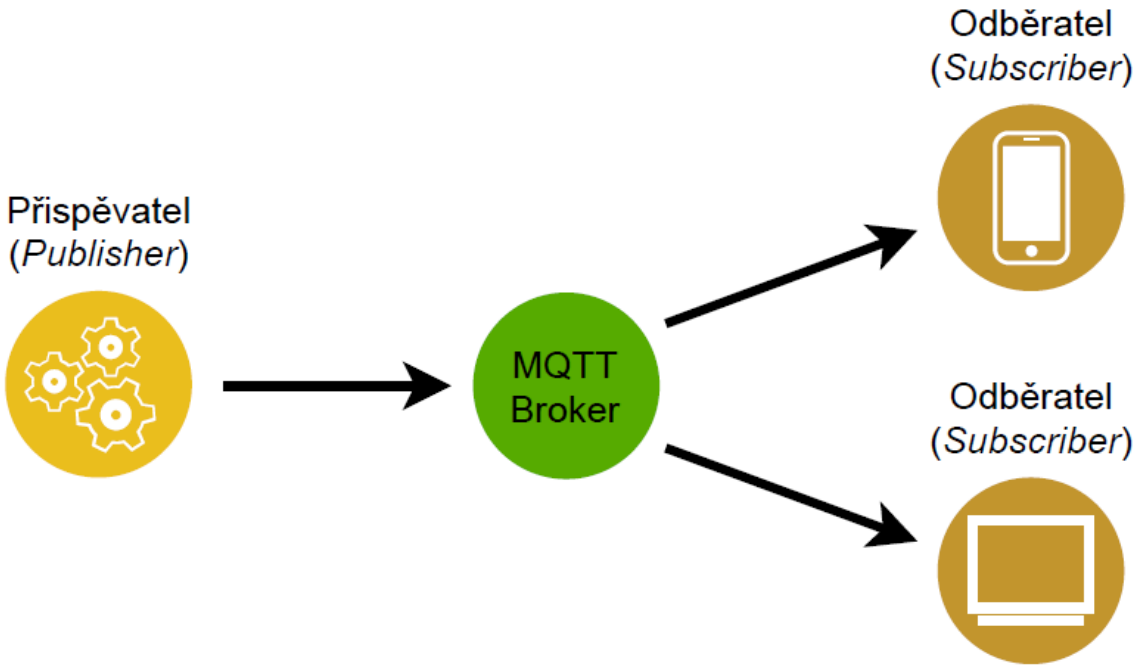
Základem CoAP protokolu je UDP nikoliv TCP komunikace. Není tedy umožněno SSL/TLS šifrování. Šifrování je zajišťováno pomocí DTLS, který je podporována UDP přenosem. Bezpečnost přenosu je tak obdobná jako v případě TLS šifrování.

CoAP protokol je podobný http protokolu. Na rozdíl od HTTP je však CoAP designován pro využití se zařízeními s nízkou výpočetní kapacitou. Přenášené pakety CoAP protokolu jsou výrazně menší než pakety HTTP protokolu. Pro úsporu objemu paketů je využívána konverze datového typu *string* na *integer*. S pakety je díky jejich menší velikosti možno snadněji manipulovat a práce s nimi nezatěžuje tolik paměť RAM daného zařízení. Protokol je tedy určen zejména pro komunikaci s malými přístroji, disponujícími malou výpočetní kapacitou.

2.7.3 MQTT (*Message Queue Telemetry Transfort*)

Protokol byl vyvinut v roce 1999 Andym Stanford-Clarkem, který v té době pracoval v IBM. Za vznikem MQTT protokolu stojí snaha o vytvoření alternativy k protokolu HTTP a CoAP, jež mají pro účely IoT aplikací výrazné nedostatky. [16]

MQTT je jednoduchý protokol jehož základem je TCP/IP komunikace. Základem MQTT komunikace je systém zveřejnění (*publish*) a odebírání (*subscribe*). Zprávy jsou publikovány s určitým názvem. Tyto zprávy jsou rozeslány všem klientům, kteří jsou přihlášení k jejich odběru. Za předávání zpráv je zodpovědný centrální server. Rozesílané zprávy jsou tříděny do témat (*topics*). Jednotlivá zařízení mohou do těchto témat přispívat (*publish*), nebo mohou být přihlášeny k jejich odběru (*subscribe*). Právě podle toho, jestli do tématu přispívají nebo ho odebírají, se zařízení dělí na zařízení typu *publisher* a *subscriber* (přispěvatel či autor a odběratel). Teoreticky může být každé zařízení zároveň *publisher* i *subscriber*, v praxi však tyto funkce bývají často rozděleny. Mezi autora a odběratele dat vstupuje *broker*, který má za úkol zajistit bezpečnost přenosu pomocí neustálého ověřování autorizace odběratele a autora. Za publikaci zprávy může být odpovědný například snímač a klientem může být například displej, který zprávu zobrazí. Zabezpečení zprávy lze posílit přidáním jména a hesla k odeslanému paketu. Schématický model MQTT komunikace je patrný z obrázku 3.



Obrázek 3: Komunikační model MQTT

Obsah zprávy není nijak definovaný. Jedná se pouze o binární data, která mohou mít libovolný charakter, protože *broker* tyto data nijak nezpracovává, pouze je přeposílá. Zpracování dat je tak čistě v režii příjemce. V současné verzi protokolu je velikost zpráv omezena na 256 MB, ale pro účely IoT bývá velikost zprávy běžně menší. [17]

Pro MQTT komunikaci není vyžadován žádný způsob šifrování a využití šifrování je na každém klientovi, resp. správci. V uzavřených sítích je bezpečnost ošetřena. Pro

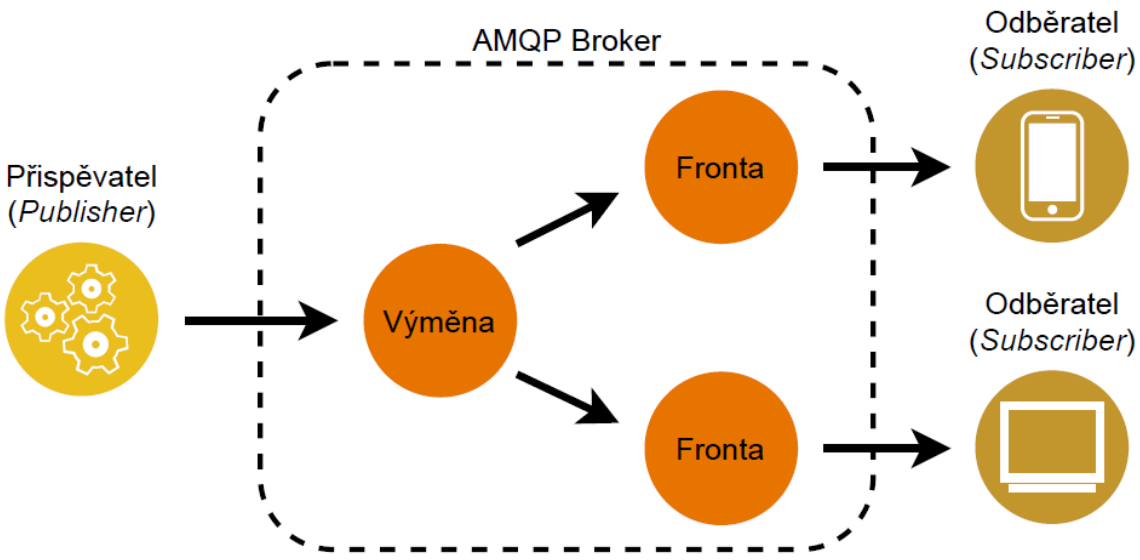
přihlášení klienta je třeba jeho ID a volitelně také uživatelské jméno a heslo. Díky podpoře SSL/TLS je podporováno přihlášení klienta pomocí SSL certifikátu. Pokud jsou data přenášena pomocí veřejného *brokera*, neexistuje žádná možnost kontroly dalších současně připojených *brokerů*.

Díky svým parametrům je MQTT protokol určen pro nespolehlivé sítě, které jím nejsou moc zatěžovány. Je schopen zajistit spolehlivé připojení i ve špatných podmínkách. Problémem není ani vysoká latence připojení s úzkou šířkou datového pásma. MQTT také klade velmi malé nároky na zařízení, na kterém je užíván. Zařízení tak nejsou příliš zatěžována a dochází k úspoře energie. Výborně se tedy hodí pro bezdrátovou komunikaci v síti IoT. V praxi je využíván zejména pro senzorové sítě. Také bývá často nabízen jako součást *cloudových* služeb, což umožňuje snadnou integraci kompatibilních zařízení.

2.7.4 AMQP (Advanced Message Queuing Protocol)

Vývoj AMQP byl započat v roce 2003 Johnem O'Harem na JP Morgan Chase v Londýně. Verze 1.0 byla vydána roku 2011. Jedná se o protokol aplikační vrstvy. Protokol je zaměřený na komunikaci pomocí zpráv. Původně byl protokol využíván ve finančnictví, později však našel uplatnění také v odvětví IoT. [18]

Komunikace pomocí AMQP protokolu je tvořena třemi základními komponentami. Jedná se o: výměnu, frontu a zprávu. Proces komunikace probíhá tak, že výměna umístí zprávu do fronty, kde zpráva čeká na zpracování. Fronta má tedy za úkol uchovávat zprávy, dokud není zpracována. Mezi výměnu a frontu vstupuje ještě vazba, která zajišťuje jejich správnou provázanost. Komunikační model AMQP protokolu je zobrazen na obrázku 4.



Obrázek 4: Komunikační model AMQP

Pro zabezpečení AMQP komunikace je možné využít TLS šifrování.

Hlavní výhodou AMQP protokolu je robustnost komunikace. AMQP garantuje doručení zprávy. Tato vlastnost často není v IoT aplikacích vyžadována. Pro přenos

klíčových dat, kde není přípustná jejich ztráta, je však tato vlastnost klíčová. Garance doručení zprávy s sebou přináší také nevýhodu v podobě poměrně velké náročnosti na paměť zařízení a stejně tak i na kvalitu internetového připojení a výpočetní výkon. AMQP komunikace tedy není vhodnou volbou pro aplikaci v IoT sítích, ve kterých se pracuje se zařízeními s malým výkonem.

2.7.5 DDS (*Data Distribution Service*)

Vývoj DDS protokolu započala společnost Real-Time Innovation a Thales Group v roce 2001. V roce 2003 byla společností Object Management Group (OMG) publikována verze 1.0. DDS je přenosový standard pro *real-time* aplikace, tedy aplikace v reálném čase. Je určen pro komunikaci mezi dvěma zařízeními. Je tedy kladen důraz na vysokou rychlost přenosu. S pomocí DDS lze data přenášet jak na menších zařízeních, tak na *cloudových* platformách. [19]

DDS protokol využívá komunikaci typu *publish-subscribe*, kdy jsou data publikována zdrojovým zařízením, avšak zpracovat je může pouze zařízení přihlášené k jejich odběru. Možné je však také implementovat komunikaci typu *request-reply*. Na rozdíl od MQTT a CoAP podporuje DDS protokol komunikaci bez *brokera*.

Komunikace pomocí DDS protokolu využívá RTPS protokol pro transport dat. Pro zabezpečení přenosu je pak možné využít TLS případně DTLS šifrování.

DDS protokol nachází díky vysoké přenosové rychlosti uplatnění všude tam, kde je to vyžadováno. Příkladem může být letecký průmysl. Nevýhodou DDS protokolu může být poměrně velký nárok na paměť RAM daného zařízení. Protokol není tedy vhodný pro velmi jednoduchá zařízení se slabším hardwarem, pro něž je aplikace buď nemožná, nebo je poměrně komplikovaná.

2.8 Šifrování

V předchozí kapitole bylo ve spojení s komunikačními protokoly hojně zmiňováno jejich šifrování. Šifrování má sloužit k zabezpečení přenášených informací. Jedná se o bezpečnostní prvek, který zajišťuje, že obsah zprávy bude schopen přečíst pouze její adresát. Ve spojitosti s IoT se jedná o velice významný aspekt komunikace. Přeposílaná data jsou totiž často důvěrného charakteru a například v průmyslových aplikacích obsahují mnohé informace o interních procesech společnosti, které podléhají firemnímu tajemství.

Zmíněny byly hlavně dva druhy šifrování, a to SSL a TLS, přičemž velké množství komunikačních protokolů podporuje oba dva. V následujícím textu je tedy podrobněji popsán způsob, jakým tyto způsoby šifrování chrání přenášená data.

2.8.1 SSL (*Secure Socket Layer*)

Secure Socket Layer neboli vrstva bezpečných vstupů je vrstva vložená mezi transportní a aplikační část komunikace. Nejčastěji se užívá ve spojení s komunikací s webovými servery pomocí HTTPS čili se zabezpečenou verzí protokolu HTTP. Jak již bylo zmíněno, komunikace pomocí HTTP protokolu probíhá mezi klientem a serverem. Příklad komunikace byl vysvětlen na požadavku klienta zobrazit internetovou stránku

pomocí prohlížeče. Při jeho opětovném prozkoumání si nyní můžeme uvědomit, že obsahem komunikace mezi klientem a serverem není pouze obsah dané internetové stránky, ale také informace o činnosti klienta na dané internetové stránce. Činností lze v tomto případě chápat mimo jiné také vyplňování hesel, kdy obsahem komunikace jsou samozřejmě také hesla samotná. Jedná se tedy o důvěrné informace, které veřejné být nesmí. Bezpečnost takovéto komunikace zajišťuje právě SSL šifrování.

Samotné šifrované komunikace předchází navázání spojení klienta se serverem, tzv. *handshake*. Navazování spojení zpravidla probíhá dle následujících bodů: [20]

1. Klient pošle serveru požadavek na SSL spojení. (Součástí této zprávy jsou i detailní informace o verzi SSL, stanovení kryptografického algoritmu atd.)
2. Server pošle klientovi odpověď, která obsahuje stejný typ informací a certifikát serveru.
3. Na základě přijatého certifikátu si klient ověří autentičnost serveru. Certifikát obsahuje také veřejný klíč serveru.
4. Klient vygeneruje základ šifrovacího klíče na základě přijatých informací. Tímto klíčem se bude šifrovat následná komunikace. Klíč je zašifrován veřejným klíčem serveru a odeslán do serveru.
5. Server použije svůj klíč k rozšifrování zprávy, obsahující základ šifrovacího klíče. Ze základu šifrovacího klíče vygeneruje server i klient hlavní šifrovací klíč.
6. Klient a server si potvrdí šifrování komunikace vygenerovaným klíčem.
7. Je ustanoveno zabezpečené spojení šifrované vygenerovaným šifrovacím klíčem.
8. Klient a server nyní komunikují pomocí šifrovaného spojení.

Kromě šifrování komunikace slouží SSL také k jasné identifikaci klienta a serveru v síti. Identifikace je dána podobou šifrovacího klíče, která je unikátní pro každé spojení klienta se serverem.

2.8.2 TLS (*Transport Layer Security*)

Transport Layer Security je nástupce SSL šifrování. SSL verze 3.0 se tak od TSL verze 1.0 prakticky neliší. Opět se jedná o šifrování komunikace především pomocí protokolů HTTP. Navazování komunikace mezi serverem a klientem probíhá obdobně jako v případě SSL.

Mezi oběma způsoby šifrování jsou však drobné rozdíly. TSL umožňuje začínat komunikaci v nešifrovaném formátu, a tak nasadit šifrování na servery, které hostují více domén na jedné IP adrese. Lze pak zabezpečit komunikaci pro více domén bez nutnosti vyhrazení veřejné IP adresy pro každou z nich. V případě TLS šifrování roste samozřejmě také míra zabezpečení, díky nově používaným kryptografickým algoritmům.

Kvůli vysoké míře podobnosti obou šifrovacích certifikátů bývají často dané certifikáty nepřesně označovány. Jelikož SSL je certifikát starší, je zakořeněn v lidském povědomí daleko hlouběji. Často tedy uživatel hovoří o TSL šifrování jako o SSL. Principiální podstata obou certifikátů je však prakticky totožná. Z uživatelského hlediska je proces šifrované komunikace totožný jak v případě SSL, tak TLS certifikátů. Odlišnost je až na drobné výjimky daná především verzí certifikátu, resp. datem jeho

uveřejnění. S tím souvisí i podpora určité verze šifrování na jednotlivých zařízeních. Šifrování pomocí SSL a TLS certifikátů je v praxi označováno jako SSL/TLS šifrování. [21]

Komunikační kanál v praktické části této práce využívá právě TLS šifrování.

2.9 Aplikace IoT sítí

IoT sítě jsou v současné době stále poměrně novou záležitostí na poli automatizace. Jejich zdařilou aplikaci lze pozorovat jen málokde. Většinou bývá doménou velkých společností, které plně přijaly podmínky průmyslu čtvrté generace a snaží se o vysokou míru technologické vyspělosti. Běžný člověk se spíše setká s aplikací IoT, která není plně integrována do daného systému, nebo je kvalita jejího provedení diskutabilní. V následujících podkapitolách jsou uvedeny příklady uplatnění internetu věcí. Nutno podotknout, že se povětšinou jedná o rozsáhlé projekty dotované velkým kapitálem.

2.9.1 Chytrá města

Chytrá města jsou příkladem nasazení IoT pro veřejné užití. Jedná se o zavádění moderních technologií do řízení chodu měst. Cílem je zvýšit energetickou hospodárnost měst, zvýšit životní úroveň obyvatel a šetřit životní prostředí. V praxi jsou implementovány především technologie zajišťující úsporu energie a efektivní řízení dopravy. Moderní technologie se v řízení měst začínají uplatňovat v celé Evropě. V České republice se směrem chytrých technologií ubírá například Praha, Brno, Písek či Pardubice. Koncept chytrých měst sdružuje jednotlivá zařízení v IoT síti velikosti přibližně MAN.

Jako konkrétní příklady aplikace chytrých technologií ve městech lze uvést adaptivní řízení semaforů v závislosti na provozu, rozsvěcení a zhasínání lamp v závislosti na okolním jasu, efektivní využívání a regulace energií, regulace spotřeby vody a mnoho dalších.

2.9.2 Průmysl 4.0

Průmysl 4.0 je jinak pojmenovaný trend digitalizace a automatizace procesů průmyslové výroby. Číslice 4 v názvu značí čtvrtou průmyslovou revoluci, kterou podnítil zejména rozvoj internetu. Průmysl 4.0 by měl znamenat zkrácení času potřebného k uvedení produktu na trh, zkvalitnění výroby a zvýšení flexibility firem.

Předpokladem nasazení průmyslu 4.0 je do jisté míry nahrazení lidské práce stroji. To povede k úbytku zejména fyzicky náročných a monotónních pracovních pozic. Pracovní pozice, které přetrvají, budou vyžadovat vyšší odbornost zaměstnance. Lze tedy říci, že nasazení průmyslu 4.0 bude největší měrou spočívat v automatizaci daného pracovního procesu. Nutno však podotknout, že automatizace výrobního procesu není žádnou novinkou a mnoho firem je již částečně, případně plně automatizováno. Označení Průmysl 4.0 se tak do jisté míry jeví pouze jako jiné jméno pro něco, co je již dlouho zavedeno. Velkou nevýhodou digitalizace je závislost na výpočetních systémech a z toho plynoucí ohrožení hackerskými útoky, které by v případě úspěšnosti ochromily chod celé společnosti.

Internet věcí bude stěžejním předpokladem průmyslu 4.0, jelikož právě do něho budou jednotlivé stroje integrovány. Jednat se bude o vytvoření sítě přibližně velikosti CAN. Díky němu bude možné jednotlivá zařízení sdružovat v síti. Velkou předností bude možnost prediktivního chování jednotlivých zařízení na základě vyhodnocování dlouhodobých dat. Tato data bude také možné využít ke strojovému učení a optimalizaci výrobních procesů. Prakticky se tak odstraní negativní lidský faktor.

2.9.3 Chytré zemědělství

Internet věcí použitý pro účely chytrého zemědělství napomáhá zjednodušení a zkvalitnění rostlinné a živočišné produkce. Zpravidla se jedná o nasazení senzorů monitorujících povětrnostní a podnebné podmínky, stav a připravenost techniky. Mohou se také uplatnit kamerové systémy a detektory pohybu. Celek poté vede k optimalizaci zemědělských procesů. Lze například optimalizovat závlahu rostlinné produkce či dávkování hnojiv založené na geografických predispozicích daného pozemku. Nastřádaná data lze také použít k přípravě na nadcházející sezónu a predikci budoucích podmínek. Lze se tak do jisté míry chránit proti ztrátám vlivem klimatických změn. Plně integrovaný internet věcí v zemědělství také podává informace vypovídající o aktuálním stavu techniky, případně varuje před nutnými opravami. Vrcholem pak může být například plná automatizace techniky.

Chytré zemědělství není žádná utopie a v praxi se začíná vyskytovat čím dál tím častěji. Implementací širokého spektra technologií do odvětví živočišné a rostlinné produkce se zabývá například rakouská společnost Pessl Instruments. [22]

2.9.4 Doprava a logistika

V rámci dopravy a logistiky lze do IoT sítě sdružit mnoho zařízení. Dobrým příkladem může být tzv. *carsharing* neboli sdílení aut. V poslední době také vznikají koncepty autonomních dopravních vozidel, která by byla schopna samostatných výjezdů a doručování nákladu. Nasazením autonomních dopravních vozidel by došlo k eliminaci nutných časových intervalů pro odpočinek řidiče nákladního vozu.

Kromě samotných dopravních prostředků jsou inovovány také technologie na jejich řízení. Příkladem může být vzdálené řízení semaforů, které své uplatnění našlo v Pardubicích. [23] Dalším příkladem může být monitoring stavu komunikací s vyhodnocováním optimální jízdní trasy. Toto se týká zejména optimalizace průjezdu hromadné dopravy. Cílem je tak předejít zpožděním. Nutno podotknout, že zmíněný monitoring je již do jisté míry funkční ve velkém množství navigačních zařízení, která jsou schopná řidiče upozornit na možné komplikace na trase a navrhnout alternativní trasu. Celkovým cílem řízení provozu je samozřejmě zrychlení a zefektivnění přepravy. Mimo to je také kladen velký důraz na zvýšení bezpečnosti například pomocí vyhodnocování opakovaných nehod a snaze o jejich předcházení na základě strojové predikce.

3 Studie proveditelnosti

Cílem praktické části této práce je realizace IoT pro účely Zkušebního centra AV R&D s.r.o. v Chrudimi. AV R&D s.r.o. je česká společnost orientovaná na vývoj výrobků ve strojírenství. Poskytuje vývoj a dodávku zkušebních stavů a prototypů, vývojové služby ve strojírenství a služby v oblasti zkušebnictví. Téma této práce je zaměřeno především na odvětví vývoje a dodávky zkušebních stavů a prototypů. Práce se zabývá integrací řídicích systémů, jejichž základem je platforma CompactRIO, do systému IoT. Tyto řídicí systémy se implementují do vyvíjených a dodávaných zkušebních stavů společnosti AV R&D. Pro účely této práce byl uvolněn zkušební stav Zwick 1873 (trhací stroj), kterému byla v roce 2019 provedena modernizace spočívající v rekonstrukci řídicího systému. Základem zcela nového řídicího systému (HW i SW) je platforma CompactRIO a zkušební stav Zwick 1873 je tedy ideálním objektem pro účely této diplomové práce.

AV R&D je aliančním partnerem společnosti National Instruments v oblasti vývoje a dodávek zkušebních stavů a prototypů. Jako takový musí plnit podmínky aliančního partnerství, kterými jsou například certifikovaní programátoři různých úrovní (Certified LabVIEW Associated Developer, Certified LabVIEW Developer, Certified LabVIEW Architect).

Podrobnější podmínky realizace praktické části této práce jsou vymezeny produkty, které má společnost, jakožto alianční partner NI k dispozici.

Hlavním požadavkem realizace je možnost spolehlivého monitoringu a kontroly probíhajících zkoušek na zkušebních stavech s automatizovanými procesy (CompactRIO, CompactDAQ, kontroléry od jiných firem). Vizualizace stavu jednotlivých veličin bude využívána především k ověření správného průběhu zkoušky a provozu zkušebního stavu, případně k vyhodnocení poruchového stavu, na jehož základě lze zkoušku následně zastavit. U prvků vizualizovaných na *dashboardu* bude postačovat nižší vzorkovací frekvence. Detailnější monitoring kýžených dat pak lze realizovat odlišně, například pomocí nezávislého ukládání souborových paketů. Dalším požadavkem je kamerový monitoring průběhu zkoušky. Opět bude postačovat zobrazování zkušebního stavu a upnutého vzorku s nižší vzorkovací frekvencí řádově v jednotkách Hz.

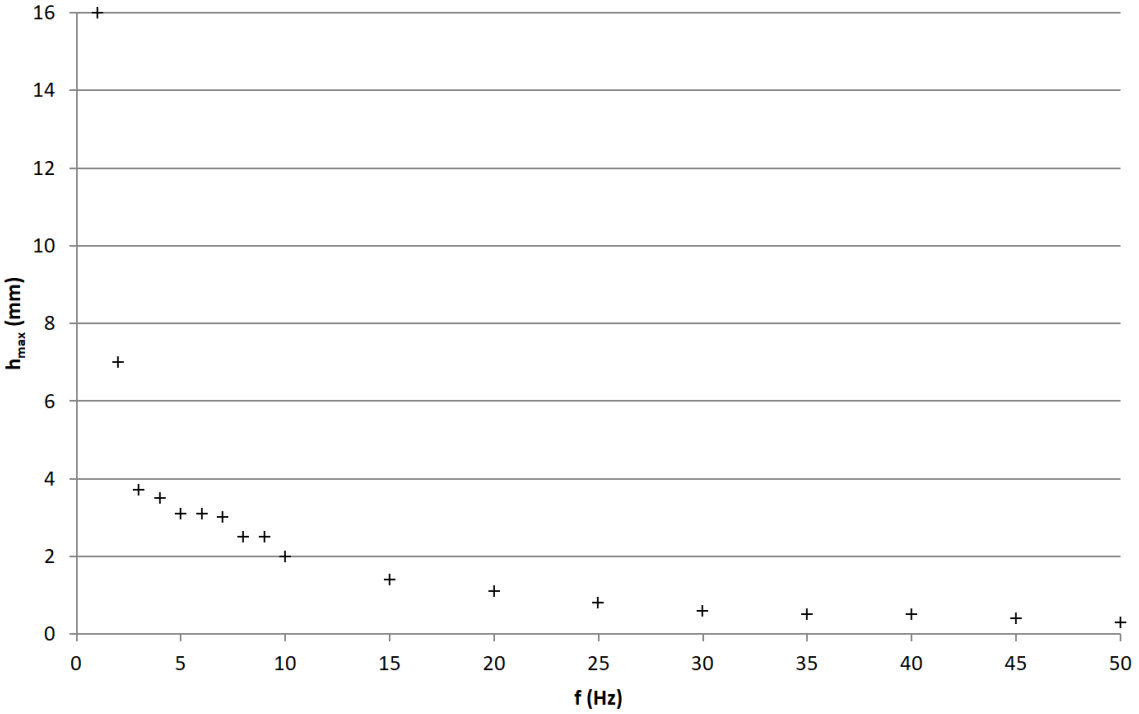
3.1 Podmínky aplikace

Monitorovaným zkušebním stavem bude hydraulická rychlotrhačka Zwick 1873. Pro extrakci a zpracování dat bude využito softwaru společnosti National Instruments. V tomto případě bude využit software LabVIEW k extrakci a odeslání dat. Data budou následně přijata a zpracována serverovou aplikací SystemLink. V této aplikaci také proběhne vizualizace požadovaných dat.

3.1.1 Rychlotrhačka Zwick 1873

Hydraulická rychlotrhačka Zwick byla vyrobena v roce 1982. V roce 2019 byla rychlotrhačka modernizována. Proces modernizace spočíval v nahrazení skříňového

řídicího počítače programovatelným průmyslovým kontrolérem CompactRIO a vytvoření řídicího softwaru v programovacím prostředí LabVIEW. Řídicím softwarem je samostatně funkční aplikace, která vznikne kompilací zdrojového kódu v LabVIEW. Současný řídicí systém umožňuje různé způsoby řízení rychlotrhačky. Rychlotrhačku lze regulovat na polohu pístu, případně na sílu, která na hydraulický válec působí. Požadovaná poloha, resp. síla může být zadávána manuálně, nebo pomocí funkčního generátoru. Funkční generátor umožňuje řízení pístu podle signálu se sinusovým, pilovým, trojúhelníkovým nebo obdélníkovým průběhem. Mezní parametry řízení rychlotrhačky jsou dány především jejími mechanickými limity. Mezní parametry byly ověřovány při řízení nezatíženého pístu podle sinusového signálu. Maximální dosažitelné amplitudě (h_{max}) signálu při dané frekvenci (f) odpovídají body, vynesené do grafu v obrázku 5.



Obrázek 5: Graf závislosti maximální dosažitelné amplitudy polohy pístu trhačky na frekvenci při buzení harmonickým signálem

Během modernizace došlo také ke kompletní přestavbě bezpečnostních prvků trhačky za použití bezpečnostních relé a relé s nuceně vedenými kontakty. Součástí rychlotrhačky je hydroagregát s dvojicí motorů zajišťující dostatečný tlak v hydraulickém válci trhačky. Dvojice motorů byla v procesu přestavby zachována. Pracovní tlak v hydraulickém okruhu trhačky je 200 barů. Maximální síla, kterou je hydraulický válec trhačky schopen vyvinout, je ±100 kN. Trhačka umožňuje monitorovat polohu pístu a sílu působící na píst.

Zmíněná přestavba na řídicí systém CompactRIO je pro účely realizace IoT velmi příhodná. Výhodou je možnost kontrolér CompactRIO programovat v prostředí LabVIEW. Takto je možné řídicí software kontroléru doplnit o část, která bude obstarávat extrakci požadovaných dat a jejich odesílání do serverové aplikace SystemLink.

Tlak v hydraulickém okruhu trhačky je zajišťován hydroagregátem. Podoba hydroagregátu před modernizací v roce 2019 je patrná z obrázku 6. Podoba hydraulické rychlotrhačky Zwick 1873 společně s původním řídicím skříňovým rozvaděčem je patrná z obrázku 7. Vlevo na obrázku je hydraulická rychlotrhačka a vpravo je řídicí skříňový rozvaděč. Do původního hydraulického agregátu byl zabudován nový řídicí systém. Bylo tak možné zcela odstranit starý řídicí skříňový rozvaděč. Ovládací panel hydroagregátu byl doplněn o nové řídicí prvky společně s monitorem, na kterém je vizualizován ovládací panel řídicího softwaru kontroléru CompactRIO 9035. Zapojení nové řídicí logiky, která je umístěna v těle hydroagregátu rychlotrhačky, včetně nových řídicích prvků, je patrné z obrázku 8

Obrázek 6: Hydroagregát před modernizací v roce 2019

Obrázek 7: Hydraulická rychlotrhačka Zwick 1873 před modernizací v roce 2019

Obrázek 8: Řídicí systém hydraulické rychlotrhačky po modernizaci v roce 2019

3.1.2 LabVIEW

Stěžejním softwarem užitým v tomto projektu je LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*). Jedná se o programovací prostředí zaměřené na testování, měření a ovládání zařízení. Programování v tomto prostředí je objektově zaměřeno.

Vývoj softwaru LabVIEW byl započat v roce 1983 pod záštitou společnosti National Instruments. Společnost National Instruments byla založena v roce 1976 trojicí podnikatelů Jamesem Truchardem, Jeffem Kodovskym a Billem Nowlinem. Za otce LabVIEW je považován Jeffrey Kodovsky, který započal vývoj grafického programovacího prostředí. Prvotním impulzem vývoje LabVIEW byl fakt, že společnost National Instruments uvedla první zařízení pro použití s PC. S vývojem LabVIEW je také úzce spjata Texaská univerzita v Austinu. V roce 1986 bylo prostředí LabVIEW uvedeno na trh. [24]

Výchozí myšlenka ovládání softwaru byla možnost zápisu programu podobným způsobem, jakým lze zapsat poznámku do poznámkového bloku. Velkou roli tedy hraje intuitivní ovládání. Jednotlivé softwarové funkce jsou reprezentovány funkčními bloky, tedy jakýmisi kostičkami, na kterých jsou vstupy a výstupy. Výstupy jednotlivých funkčních bloků jsou poté propojovány s jednotlivými vstupy pomocí linek, které vizuálně evokují vodiče. Podle svého vzezření se také označují virtuální vodiče. V současnosti se vývoj LabVIEW ubírá směrem využití vícejádrových procesorů za účelem optimalizace chodu softwaru a zvýšení efektivnosti práce s ním. Je tak umožněno například paralelní programování. Vedle sebe tak může běžet několik smyček typu while, přičemž jednotlivé smyčky na sobě mohou a nemusí být závislé.

Softwarové prostředí LabVIEW je pro účely této práce více než vhodné. Jelikož hydraulická rychlotrhačka, která je objektem implementace do sítě IoT, je řízena kontrolérem CompactRIO. Software pro CompactRIO byl vytvořen právě pomocí LabVIEW. Při úpravě současné verze řídicího software trhačky tedy nebude nutné řešit problémy se zpětnou kompatibilitou. LabVIEW samotné podporuje kompatibilitu s kontroléry CompactRIO. Do LabVIEW je možné implementovat doplněk s názvem Real-time, který přímo obsahuje funkční bloky určené pro základní programování kontroléru CompactRIO v systému reálného času.

Alternativou k programovacímu prostředí LabVIEW je LabVIEW NXG. Jedná se o produkt společnosti National Instruments, který koexistuje s klasickým LabVIEW. Verze NXG má novou grafickou podobu a umožňuje odlišnou manipulaci s funkčními bloky, jako třeba zoomování na programovací ploše, což byl dlouho vytýkaný nedostatek klasické verze LabVIEW. LabVIEW NXG má lépe zpracovanou podporu komunikace se serverovou aplikací SystemLink, které je plánováno využít pro IoT. K dispozici je například nápověda k funkčním blokům pro SystemLink komunikaci, která u klasické verze LabVIEW chybí. LabVIEW NXG také nabízí podporu snadné tvorby webových aplikací, které pak lze umístit na server. Pro účely této práce má však LabVIEW NXG jeden zásadní nedostatek. V současné verzi nepodporuje *real-time* aplikace. Mezi *real-time* aplikace se mimo jiné řadí také software pro platformu CompactRIO, na které v současné době funguje hydraulická rychlotrhačka. LabVIEW NXG nelze tedy použít pro úpravu řídicího softwaru a extrakci dat. Pro tyto činnosti je nutné použít LabVIEW s modulem real-time.

3.1.3 SystemLink

SystemLink je serverová aplikace, která slouží ke střádání a managementu dat. Zaměřuje se především na data ve zkušebnictví a klade důraz na kompatibilitu s LabVIEW. Ve zkušebnictví se velmi často vyskytuje potřeba rychlých měření v řádech desítek kHz. Tato data tedy představují velké soubory v řádech GB.

Cílem SystemLinku je eliminace manuálních úkonů při udržování monitorovaných systémů. Poskytuje možnost monitoringu a správy dat po dobu životnosti sledovaného zařízení. Nastřádaná data je možné vizualizovat ve webovém prohlížeči pomocí *dashboardů*. *Dashboardem* se rozumí panel, na kterém jsou vizualizovány kontrolní, případně řídicí prvky systému. Jedná se tak o jakousi obdobu ovládacích panelů, které jsou součástí aplikace vytvořené v prostředí LabVIEW. Objekty vizualizované pomocí *dashboardu* mohou mít různé datové typy a jejich vizuální podobu lze konfigurovat. Kromě monitoringu dat umožňuje SystemLink také jednoduché ovládání monitorovaného zařízení pomocí zpětné komunikace.

Serverová aplikace SystemLink je kompatibilní s vývojovým prostředím LabVIEW. Do LabVIEW je možné přidat doplněk obsahující funkční bloky podporující komunikaci právě s aplikací SystemLink. Zprovoznění vzájemné komunikace je tak poměrně jednoduché. Zmíněný doplněk nabízí několik možností vzájemné komunikace.

První možností komunikace mezi softwarovým prostředím LabVIEW a serverovou aplikací SystemLink je předávání informací pomocí *tagů*. *Tag* je paket obsahující data určitého typu. Data mohou být typu *Boolean, Integer, Double*, případně *String*. Datový typ se volí v procesu konfigurace daného *tagu*. *Tagy* jsou uloženy na serveru, kde k nim lze pod jejich názvem přistupovat. Libovolně lze také ukládat hodnoty *tagů* v čase. Jednotlivé *tagy* lze v závislosti na jejich datovém typu vizualizovat na *dashboardu*. Do *tagů* na serveru je možné informace zapisovat, případně z *tagů* informace číst. Za tímto účelem je v LabVIEW funkční blok *Tag Read* a *Tag Write*. Speciálním případem je zápis, případně čtení několika *tagů* najednou. Za tímto účelem byly vytvořeny funkční bloky *Multiple Tag Read* a *Multiple Tag Write*. *Tagy* jsou primárně koncipovány pro monitorování a relativně pomalé rychlosti přenosu. Operace s *tagy* jsou běžně prováděny s frekvencí kolem 1 Hz. Zápis do *tagů* je možné provádět i rychleji, ale rychlosti streamovacích služeb dosáhnout nelze.

Další možností je komunikace pomocí zpráv tzv. *message*. Jedná se o komunikaci typu *publish-subscribe*. Data jsou odesílána pomocí zpráv serverovou aplikací všem adresátům, kteří jsou přihlášeni k odběru. Pomocí zpráv lze posílat data typu *Boolean, Integer, Double*, případně *String*.

Poslední možností komunikace se serverovou aplikací SystemLink je ukládání datových souborů do úložiště na serveru. Soubor lze na serveru uložit, lze ho také přečíst a smazat. Soubory uložené na serveru je možné v závislosti na jejich typu pomocí aplikace SystemLink vizualizovat. Společně se zápisem *tagů*, které mají v případě rychlých dějů pouze hrubý informativní charakter, lze tedy paralelně ukládat soubory nesoucí informaci o rychlých dějích, například průběhy požadovaných veličin ve formátu TDMS. Pokud nastane nenadálá situace, kterou nelze dostatečně přesně analyzovat pomocí *tagů*, lze k detailnímu posouzení stavu využít právě uložené soubory, například ve formátu TDMS.

3.1.4 CompactRIO

Současný řídicí systém hydraulické rychlotrhačky je založen na průmyslovém kontroléru CompactRIO 9035. Jedná se o *real-time* průmyslový kontrolér s modulárním systémem. Vyrábí se v několika velikostech, které ovlivňují počet možných připojitelných modulů, které lze vsunout do slotů kontroléru. Podstatné funkce kontroléru jsou dány především využitím modulárních karet (modulů), které k němu lze

připojit. Jednotlivé karty umožňují například čtení analogových hodnot, případně generování digitálních výstupů a mnoho dalších funkcí. CompactRIO je možné programovat pomocí software LabVIEW a pomocí programovacích jazyků C, C++ a Java. Operační systém CompactRIO kontroléru ve verzi 9035 je založen na linuxové distribuci. Výpočetní hardware je umístěn v robustním šasi, které zajišťuje dobrou odolnost kontroléru a umožňuje průmyslové nasazení i v náročných podmínkách (vibrace, teplota). Podoba CompactRIO včetně připojených karet je patrná z obrázku 9.

Obrázek 9: CompactRIO 9035 s připojenými modulárními kartami

V praktické části této práce je použito CompactRIO ve verzi 9035. V této verzi je CompactRIO vybaveno Dual-Core procesorem pracujícím na frekvenci 1,33 GHz, 1 GB paměti RAM a 4 GB vnitřního úložiště, které je možné rozšířit pomocí paměťových karet až o 32 GB. Verzi 9035 odpovídá obrázek 9, na kterém stojí za povšimnutí, že CompactRIO 9035 disponuje osmi volnými sloty pro připojení rozšiřujících karet (modulů). [25]

3.1.5 Modulární karty

Společně s kontrolérem CompactRIO jsou v této práci využity rozšiřující karty. Tyto karty umožňují zpracovávání případně generaci signálu s různými parametry. Použité rozšiřující karty jsou dále v tomto blíže popsány.

3.1.5.1 NI 9265

Karta NI 9265 umožňuje generovat analogový signál proudového typu. Signál může nabývat hodnot od 0 do 20 mA. Karta má čtyři na sobě nezávislé analogové výstupy. Signál je generovaný s rozlišením 16 bitů.

Pomocí karty NI 9265 je řízen řídicí ventil hydraulické trhačky. Ventil však není možné řídit přímo, protože rozsah řídicího signálu ventilu je ±10 mA. Výstup karty je tedy nutné upravit. Úprava signálu je provedena pomocí převodníku Knick BL 570. Převodník umí zpracovávat vstupní signál v rozmezí 4 až 20 mA, na základě kterého generuje signál v rozmezí ±20 mA. Maximální frekvence řídicího signálu je omezena mezní frekvencí převodníku Knick, která je 100 Hz. [26]

3.1.5.2 NI 9375

Karta NI 9375 umožňuje generovat a zpracovávat digitální signál. Pracuje s napěťovou úrovní 0 V pro logickou hodnotu *false* a 24 V pro logickou hodnotu *true*. Na kartě se nachází šestnáct digitálních vstupů a šestnáct digitálních výstupů. Maximální frekvence karty NI 9375 je 2 MHz, přičemž limitním faktorem je především rychlost zápisu hodnot na digitální výstupy.

Digitálními výstupy jsou ovládána jednotlivá řídicí relé zajišťující spínání a vypínání hydroagregátu. Dále je zde ovládáno bezpečnostní relé, resp. stvrzení plného naběhnutí CompactRIO. Je zde také spínáno počítadlo strojohodin a zvuková signalizace. Digitální vstupy zobrazují stav chybových hlášek týkajících se filtrů hydraulické kapaliny, překročení maximální teploty hydraulické kapaliny, dosažení minimálního pracovního tlaku, hladinu oleje, stav bezpečnostního relé, stav dvou NO kontaktů mechanického stop tlačítka a tři stopy rotačního enkodéru sloužícího pro manuální nastavení žádané polohy pístu. [27]

3.1.5.3 NI 9201

Karta NI 9201 umožňuje podle *datasheetu* měřit napěťový signál v rozsahu ±10 V. Udávaný rozsah však není zcela přesný a skutečný rozsah měření je ±10,53 V. Rozlišení karty je 12 bitů a karta je schopna měřit s maximální frekvencí 500 kHz.

V řídicím systému hydraulické rychlotrhačky je karta využita ke zpracovávání signálu snímače dráhy, pomocí kterého je měřena aktuální poloha pístu. Rozlišením karty NI 9201 je dána maximální přesnost, se kterou lze polohu pístu měřit. Uvážíme-li, že karta má rozlišení 12 bitů a maximální poloha pístu je 294 mm, zjistíme, že karta je schopná měřit dráhu s rozlišením 0,072 mm. [28]

3.1.5.4 KI-5171A1

Výrobcem karty KI-5171A1 na rozdíl od předchozích karet není společnost National Instruments, ale firma Kistler. Jedná se o kartu, která je ve své podstatě nábojovým zesilovačem a umožňuje měřit náboj v rozsahu ±1 μC. Měřicí rozsahy karty lze pro zvýšení přesnosti přepínat. Karta má rozlišení 24 bitů a umožňuje měření na frekvenci maximálně 50,8 kHz.

Karta KI-5171A1 je využita k měření výstupu snímače síly hydraulické rychlotrhačky. Jestliže na snímač síly působí síla, generuje piezoelektrikum ve snímači náboj odpovídající působící síle. Náboj je ze snímače pomocí elektrody odváděn a nízkokapacitním koaxiálním kabelem přiveden na měřicí kartu. Pomocí měřicí karty je náboj zesilován. Použitý snímač síly umožňuje měření v rozsahu ±120 kN. Převodní konstanta snímače síly má hodnotu přibližně −4 pC/N. [29]

3.1.6 Snímání obrazu průběhu zkoušky

Obraz průběhu zkoušky lze snímat několika způsoby. K dispozici je USB kamera Basler ACE, kterou lze připojit k libovolnému počítači s USB portem ve verzi 3.0 a vyšším.

Jedním z možných řešení jak vizualizovat obraz z kamery pomocí aplikace SystemLink je vytvoření WebVI na lokálním počítači. Jedná se o funkci, kterou umožňuje LabVIEW. Principem je vytvoření internetové stránky nad řídicí logikou

LabVIEW, která zobrazuje určité prvky z řídicí logiky. Tuto stránku pak lze publikovat na lokální síti a lze k ní z aplikace SystemLink pomocí objektu *iframe* přistupovat. Nevýhodou je však neustálé vytížení jednoho zařízení s připojenou kamerou.

Další možností je extrakce obrazu pomocí software LabVIEW. Následuje datová konverze jednotlivých snímků na datový typ *String*. Řetězec je poté odeslán pomocí zpráv do serverové aplikace SystemLink. Řetězec obsahující obraz kamery lze poté přijmout libovolným zařízením s přístupem na SystemLink a pomocí dekódovacího programu vytvořeném v LabVIEW řetězec konvertovat zpět na snímek. Dekódovací program lze spustit na počítači bez nutnosti instalace LabVIEW. Tento způsob přenosu obrazu je poměrně rychlý, avšak nedovoluje vizualizaci v aplikaci SystemLink. Vhodný je pro přenos obrazu například mezi dvěma počítači.

Obraz z kamery lze také po sejmutí ukládat do serverové aplikace SystemLink jako soubor. Každý nový snímek z kamery může přepsat stávající a lze tak zobrazovat průběh zkoušky. Nutno však podotknout, že zobrazovací frekvence je poměrně malá (okolo 1 Hz). K uloženým snímkům lze v paměti serveru přistupovat z aplikace SystemLink. Vyskytuje se zde však několik problémů. Prvním je fakt, že program na snímání obrazu pomocí USB kamery a následná konverze na požadovaný typ obrázku je výpočetně poměrně náročný úkon. Pro implementaci do současného softwaru je tedy nevhodný. Mimoto soubory uložené v paměti serveru lze vizualizovat pouze pomocí správce souborů. Jednotlivé snímky tak nelze vizualizovat na *dashboardu*, který bude pro danou aplikaci vytvořen. Snímání a následná vizualizace průběhu zkoušky by tak byla uživatelsky nepřívětivá.

Poslední přijatelnou možností je pořizování záznamu zkoušky pomocí IP kamery. Kamera by pracovala na lokální síti, stejně tak jako server s aplikací SystemLink. Obraz z kamery by pak byl přístupný přes IP adresu kamery. Byla by tedy možná vizualizace v prostředí aplikace SystemLink v objektu typu *image*. Způsob implementace je poměrně jednoduchý. Zobrazovací frekvence kamerového záznamu v rozlišení 1280 x 1024 pixelů bude přibližně 15 snímků za sekundu.

3.2 Rozbor současného softwaru hydraulické rychlotrhačky

Software, který řídí hydraulickou rychlotrhačku, byl vytvořen v LabVIEW ve verzi 2018. Skládá se z několika procesních vrstev, které je nutné doplnit o extrakci a odesílání dat do aplikace SystemLink. Jednotlivé části stávajícího softwaru a jejich funkce budou probrány níže. Software LabVIEW ve spojení s kontrolérem CompactRIO umožňuje programování ve dvou základních vrstvách. Jedná se o vrstvu FPGA, tedy *Field Programming Gate Array,* neboli vrstvu programovatelných hradlových polí, která jsou modulárními kartami podporována a *real-time* vrstvu neboli aplikační vrstvu reálného času. Jednotlivé vrstvy obsahují struktury typu *while*, které budou nadále označovány jako smyčky. Tyto smyčky paralelně a s různým časováním vykonávají požadované příkazy. Časování jednotlivých smyček odpovídá nastavenému zpoždění smyček. Zpoždění daných smyček lze nastavit pomocí funkčních bloků, které jsou k dispozici v základní paletě LabVIEW.

3.2.1 FPGA

Návrh hradlových polí je umožněn hardwarovou podporou této funkce. Jednotlivé karty použité v jednotce CompactRIO obsahují hradlová pole, jejichž funkci je možné navrhnout tak, že vykonávají část programu. Tato část programu může být vykonávána mnohem rychleji než program prováděný na úrovni r*eal-time*. FPGA vrstva je tak vhodná pro prvotní snímání a filtraci signálů, bezpečnostní funkce a ostatní úkony, které je třeba vykonávat s vysokou rychlostí a na které je třeba reagovat co nejrychleji. Maximální frekvence vykonávání části programu pomocí hradlových polí je standardně nastavena na 40 MHz. Tuto frekvenci je možné v závislosti na konkrétní aplikaci zvýšit či snížit. Změna maximální frekvence provádění FPGA vrstvy je podmíněna hardwarovou vybaveností jednotlivých modulárních karet a složitostí vykonávaných logických operací.

V případě projektu hydraulické rychlotrhačky jsou v FPGA vrstvě vykonávány čtyři smyčky. Každá má jinou funkci a každá je vykonávána s jinou frekvencí.

3.2.1.1 FPGA smyčka 1

V první smyčce jsou čteny digitální vstupy a zapisovány informace na digitální výstupy karty NI 9375. Dochází zde také k ovládání proporcionálního ventilu hydraulického válce pomocí karty NI 9265. Funkce obou karet je blíže popsána v odstavcích 3.1.5.2 a 3.1.5.1. Vhodné je však blíže popsat funkci enkodéru, jehož digitální výstupy jsou snímány pomocí karty NI 9375.

Enkodér je zařízení umožňující měření rychlosti otáčení a úhlu natočení. Mechanický pohyb hřídele je enkodérem přeměněn na elektrické signály. Použitý enkodér, stejně jako většina běžných enkodérů, má tři stopy. Jedná se o stopu X, Y a Z. Stopy X a Y v průběhu otáčení nabývají hodnot logické 1 a 0 v závislosti na úhlu natočení. Stopy Jsou vůči sobě vzájemně posunuty, což umožňuje detekci směru otáčení hřídele. Stopa Z nabývá hodnoty logické 1, pouze pokud této hodnoty nabývá stopa X a Y současně. Stopy enkodéru jsou v FPGA smyčce 1 zpracovány pomocí příslušného funkčního bloku, jehož výstupem je hodnota, která je enkodérem inkrementována, resp. dekrementována v závislosti na směru otáčení. Této hodnoty je dále v programu využito k manuálnímu zadávání žádané polohy pístu, případně síly, který na něj působí.

První FPGA smyčka nemá limitní frekvenci. Maximální frekvence FPGA vrstvy je však nastavena na 40 MHz. Rychlost vykonávání je tak prakticky omezena pouze maximální možnou rychlostí čtení, resp. zápisu na karty využívané v této smyčce.

3.2.1.2 FPGA smyčka 2

Druhá v pořadí je smyčka, ve které je zpracováván údaj o síle působící na pístnici pomocí karty KI-5171A1. Dochází zde ke čtení hodnot siloměru v pikocoulombech a následnému přepočtu na newtony pomocí konstanty siloměru. Limitní frekvence smyčky je dána parametry karty KI-5171A1 uvedenými v odstavci 3.1.5.4. Proces přepočtu je provázen datovou konverzí výsledku na data typu *fixed point*, která je nutná z důvodu změny bitové délky vlivem konverze. Sejmuté hodnoty jsou dále filtrovány dolnopropustným filtrem. Filtr je nastaven tak, aby odfiltroval nežádoucí ruchy v měřeném signálu a zároveň neomezoval chod rychlotrhačky. Po filtrování hodnot následuje jejich průměrování. Průměrováno je vždy 50 vzorků. Při frekvenci smyčky 50 kHz a průměrování z 50 vzorků můžeme odvodit, že nový vzorek je k dispozici

s frekvencí 1 kHz. Tato frekvence odpovídá frekvenci hlavní *real-time* smyčky, v níž se s daty bude dále pracovat a zajišťuje tak včasné předání požadované informace.

3.2.1.3 FPGA smyčka 3

Třetí smyčka FPGA vrstvy zajišťuje zpracování údajů o poloze pístu, které poskytuje karta NI 9201. Smyčka je prováděna s frekvencí 100 kHz, což odpovídá mezní frekvenci čtení z dané karty. Obdobně jako v případě měření síly je zde použit dolnopropustný filtr s nastavením zajišťujícím odfiltrování nežádoucích ruchů v signálu. Po filtraci následuje průměrování. Jelikož se smyčka provádí s frekvencí 100 kHz, je možné průměrovat vždy 100 vzorků. Nový údaj o poloze je pak opět k dispozici s frekvencí 1 kHz, což odpovídá frekvenci hlavní *real-time* smyčky, kde budou data dále zpracovávána.

3.2.1.4 FPGA smyčka 4

Poslední smyčka FPGA vrstvy programu zajišťuje nastavování karty KI-5171A1, která měří sílu působící na pístnici. Smyčka je prováděna s frekvencí 20 Hz, je tedy ze všech smyček FPGA vrstvy nejpomalejší. Ve smyčce je možné měnit měřicí rozsah karty a spouštět, resp. vypínat měření síly.

3.2.2 *Real-time*

Zatímco FPGA vrstva programu sloužila k prvotnímu zpracování dat a k ovládání jednotlivých výstupů, *real-time* vrstva obsahuje hlavní programovou logiku. Obě programové vrstvy jsou navzájem propojeny. Hodnoty FPGA výstupů jsou dodávány *real-time* vrstvou a hodnoty FPGA vstupů jsou *real-time* vrstvou vyhodnocovány, případně vizualizovány. Je tedy nutné synchronizovat komunikaci smyček, které se provádí v různých programových vrstvách.

V *real-time* programové vrstvě je hojně využíváno *clusterů*. *Clusterem* se rozumí seskupení proměnných s různými názvy, které mohou mít různý datový typ. V případě objektového programování je vhodné *clustery* využívat k předávání velkého množství dat mezi jednotlivými částmi programu, případně k definici nového datového typu. Využití *clusterů* výrazně zpřehledňuje výsledný program.

Využívána je také metoda FIFO (*First In First Out*). Podstatou této metody je předávat hodnoty určité proměnné. Pořadí předávání hodnot je dáno stářím jejich pořízení. Hodnoty jsou předávány od nejstarší po nejnovější. Předávání hodnot probíhá nepřetržitě a pokud nejsou k dispozici nové hodnoty, jsou předávány hodnoty nulové. Pro využití v *real-time* vrstvě je metoda FIFO často doplněna o posuvný registr, který eliminuje předávání nulové hodnoty. Pomocí posuvného registru jsou tak dokola předávány hodnoty staré, a to do doby, dokud nejsou nahrazeny hodnotami novými. Posuvný registr je vhodné využít pro přenos dat mezi smyčkami s různým časováním. Datový typ, přenášený metodou FIFO je často tvořen *clusterem*. Tímto způsobem lze předávat řetězec dat různého typu.

Speciálním typem metody FIFO je fronta *queue*. Fronta *queue* stejně jako metoda FIFO předává hodnoty od nejstarší po nejnovější, avšak předávání hodnot probíhá pouze pokud jsou k dispozici hodnoty nové. Pokud tedy frontu *queue* přivedeme do smyčky typu *while*, bude tato smyčka provedena pouze tehdy, obdrží-li fronta nová data.

Setkat se můžeme také s pojmem SubVI. SubVI je podprogram umístěný do samostatného souboru. Tomuto podprogramu lze přiřadit vstupy a výstupy a poté ho lze umístit do jiného programu, kde se SubVI chová podobně jako funkční blok. SubVI slouží především ke zvýšení přehlednosti celého programu.

3.2.2.1 *Real-time* smyčka 1

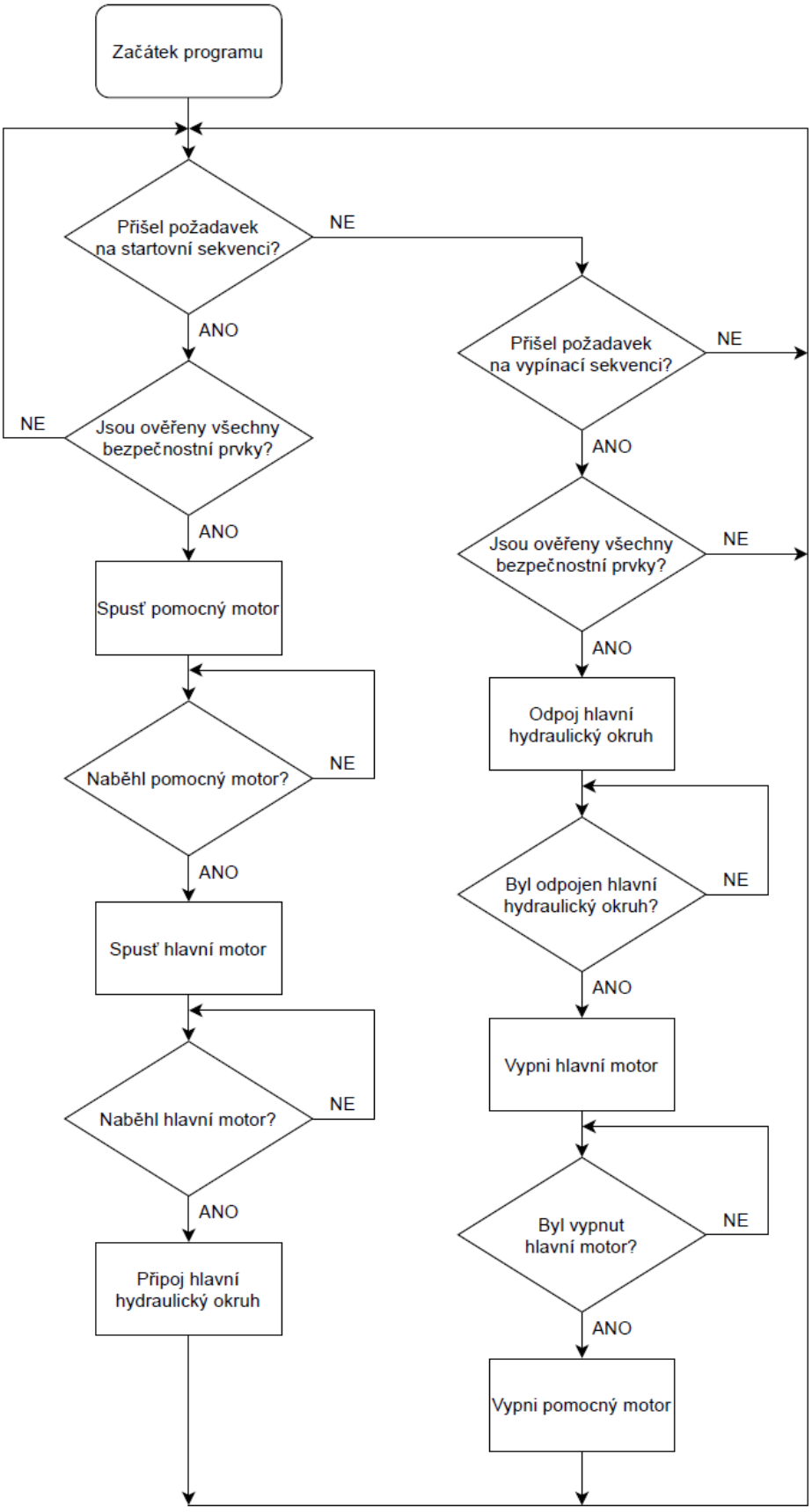
První a nejdůležitější smyčka *real-time* vrstvy se provádí s frekvencí 1 kHz. Jedná se o atypickou smyčku typu *while* s názvem *timed loop*. U této smyčky je možné podrobnější nastavení časování a priority jejího vykonávání. Výchozí funkcí smyčky je také monitoring jejího včasného provedení. Sledování včasného provedení je klíčové pro ladění programu a jeho kontrolou lze vyhodnocovat vliv přidaného kódu. *Timed loop* smyčku lze použít v celém programu pouze jednou. První *real-time* smyčka obstarává základní funkce rychlotrhačky.

První funkcí hlavní *real-time* smyčky je spínání a vypínání hydraulické rychlotrhačky. Spínání i vypínání jsou procesy, při kterých je nutné řídit několik silových prvků hydraulického systému se správným načasováním. Proces spínání a vypínání rychlotrhačky je patrný z vývojového diagramu na obrázku 10.

V době, kdy je trhačka spuštěná, je nutné ji řídit. K řízení je využit PID regulátor. Řídicí systém rychlotrhačky umožňuje volit zpětnou vazbu PID regulátoru. Trhačku je tak možné regulovat na požadovanou polohu nebo sílu. Žádanou vstupní veličinu regulátoru lze měnit také. Vstup regulátoru lze nastavovat manuálně pomocí enkodéru, případně lze na vstup přivést výstup funkčního bloku generátoru funkcí. Generátor funkcí pak pomocí hodnot nastavených v jiné části programu generuje funkci s odpovídajícím průběhem. Tento způsob řízení trhačky je možné použít pro zkoušky životnosti, kde je vyžadováno cyklické zatěžování zkušebního vzorku.

V hlavní *real-time* smyčce je zajištěna komunikace s FPGA vrstvou. Jsou zde přijímána data čtená v FPGA vrstvě ze vstupů jednotlivých karet. Naopak data, která jsou určena pro zápis na výstupy jednotlivých karet, jsou zde do FPGA vrstvy předávána. Jelikož ne všechna data čtená z FPGA vrstvy a ne všechna data určená pro zápis do FPGA vrstvy jsou zpracovávána, resp. generována v této smyčce, je nutné tato data v *real-time* vrstvě distribuovat. K distribuci dat v *real-time* vrstvě je využito metod FIFO a fronty queue. V programu je vytvořena zvlášť metoda pro data čtená z FPGA vrstvy a pro data určená k zápisu do FPGA vrstvy. Metodu pro data čtená z FPGA vrstvy lze v ostatních smyčkách *real-time* vrstvy pouze číst a do metody pro data, která mají být do FPGA vrstvy zapsána, lze v ostatních smyčkách *real-time* vrstvy pouze zapisovat. Tímto způsobem se předejde nechtěným přepisům dat vlivem nestejného časování smyček. Daným metodám je pomocí nově vytvořených *clusterů* nadefinován výchozí datový typ, který obsahuje všechny proměnné, které budou pomocí metod předávány. V hlavní *real-time* smyčce jsou data do odpovídajících metod zapisována, případně jsou z nich čtena.

Další funkcí hlavní *real-time* smyčky je střádání dat, určených k ukládání. Hlavní *real-time* smyčka je nejrychlejší periodicky prováděná smyčka *real-time* vrstvy. Je tedy vhodné v této smyčce navzorkovat data, která pak lze v jiné části programu uložit. Data je možné dočasně uložit do polí, která budou předávána frontou *queue* do části programu, ve které dojde k jejich uložení.



Obrázek 10: Vývojový diagram SubVI *Motor_start_stop*

3.2.2.2 *Real-time* smyčka 2

Další strukturou v *real-time* programové vrstvě je smyčka bez časového omezení. Tato smyčka obsahuje strukturu *event*, která generuje libovolnou akci na základě změny stavu určité proměnné. V programu je *event* struktura nejčastěji využívána k detekci změny stavu ovládacího prvku. Na základě této změny je pak upravena odpovídající programová proměnná, případně vykonána určitá programová logika. Hlavní výhodou *event* struktury je sdružování velkého množství ovládacích prvků a její flexibilita. Při vyvolání dané události tak lze například změnit nastavení, či zablokovat jisté funkce programu. Do *event* struktury lze také přidat další spouštěcí podněty (události), program tak lze jednoduše modifikovat. Umístění *event* struktury ve smyčce, jejíž provádění není časově omezeno, není náhodné. Smyčku je třeba provádět co nejrychleji, aby došlo k co možná nejpohotovější reakci na změny parametrů programu. Z toho plyne také fakt, že do této části programu nelze umísťovat výpočetně a časově náročné funkce. Dané příkazy je třeba provést velice rychle, aby tak nedocházelo ke zpožďování reakcí programu na ovládání. Ovládací prvky programu nemusí být díky *event* struktuře propojeny s proměnou, kterou ovládají. Změna dané proměnné je vyvolána na základě vygenerování příslušné události *event* struktury. V rámci této události je do dané proměnné zapsána požadovaná hodnota a ta je pak zapsána do metody *FIFO*, odkud je vyčítána v jiných částech *real-time* vrstvy. Kromě zápisu je v *event* struktuře využíváno také čtení z metody FIFO, na základě kterého mohou být prováděny některé logické operace.

3.2.2.3 *Real-time* smyčka 3

V *real-time* programové vrstvě se nachází také smyčka řídící prvky grafického zobrazení na kontrolním panelu. Tato smyčka je vykonávána s frekvencí 30 Hz, která by měla dostačovat pro plynulé zobrazování informačních dat na kontrolním panelu. Zobrazovány jsou zde chybové hlášky, stav bezpečnostního relé a údaje o aktuální poloze a síle. Síla a poloha jsou zde zobrazovány číselně i graficky, přičemž graficky zobrazovaná veličina je měněna v závislosti na zvoleném zdroji zpětné vazby do PID regulátoru. V této smyčce jsou také zaznamenávány změny hodnot jednotlivých kontrolních prvků, aby na jejich základě bylo možné vyvolat událost v *event* struktuře.

3.2.2.4 *Real-time* smyčka 4

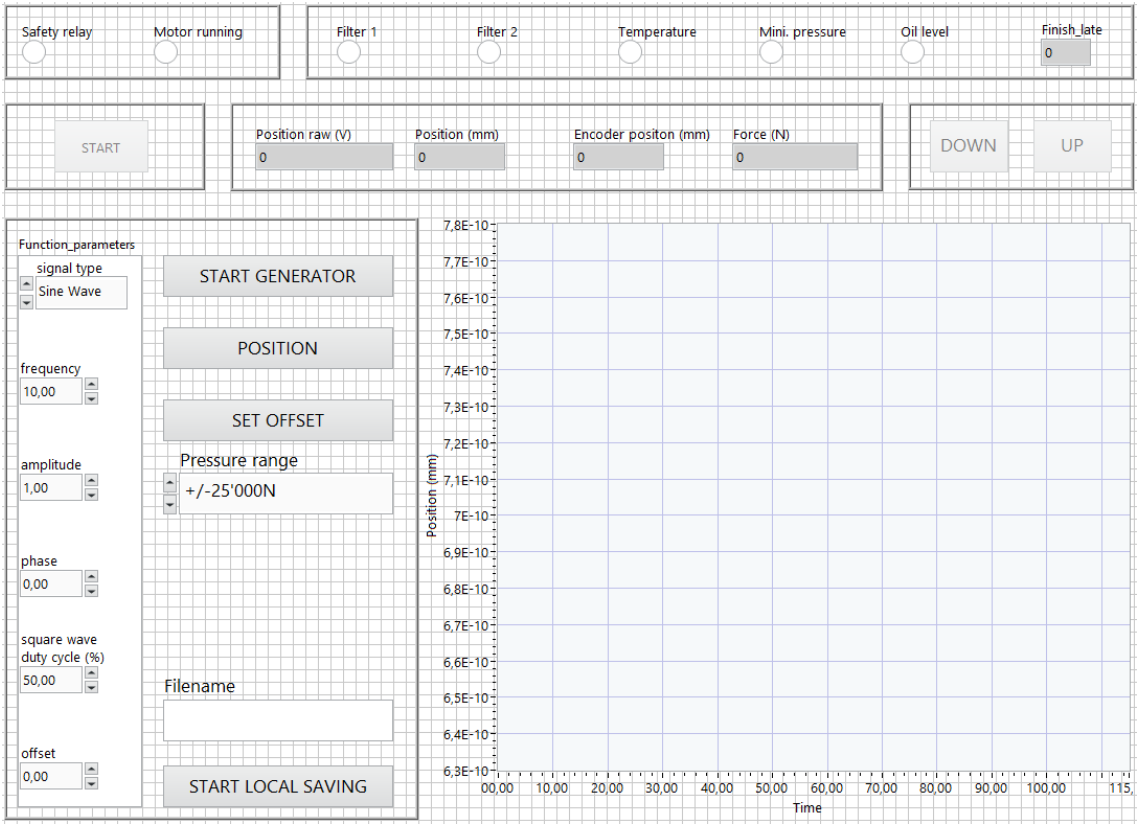
Poslední částí *real-time* programové vrstvy je smyčka, jejímž účelem je ukládání dat na disk kontroléru. Pomocí *queue* fronty jsou do této smyčky předávány data z hlavní *real-time* smyčky, kde byla požadovaná data navzorkována s frekvencí 1 kHz a nastřádána do pole o sto prvcích. Snadno lze dopočítat, že pole o sto prvcích budou v hlavní smyčce naplňována s frekvencí 10 Hz. Poslední, čtvrtá smyčka *real-time* struktury, tedy musí být vykonávána také s frekvencí 10 Hz, aby nedocházelo ke ztrátě dat. Data jsou ukládána s frekvencí pomalejší, než s jakou jsou vzorkována cíleně. Ukládání dat trvá poměrně dlouho a ukládání s frekvencí stejnou, jako je frekvence hlavní *real-time* smyčky, tak není možná.

V této smyčce je možné datům přiřadit jméno. Data jsou pak pod daným jménem ukládána na disku kontroléru CompactRIO 9035 ve formátu TDMS. TDMS je výchozím formátem pro ukládání dat softwaru od společnosti National Instruments. Uložená data pak lze snadno analyzovat například pomocí programu DIAdem.

3.2.3 Ovládací panel

Nedílnou součástí řídicího softwaru trhačky je ovládací panel. Ten pracuje v *real-time* softwarové vrstvě. Obsahuje všechny kontrolní prvky, které jsou nezbytné k ovládání trhačky. Jeho podoba je patrná z obrázku 11.

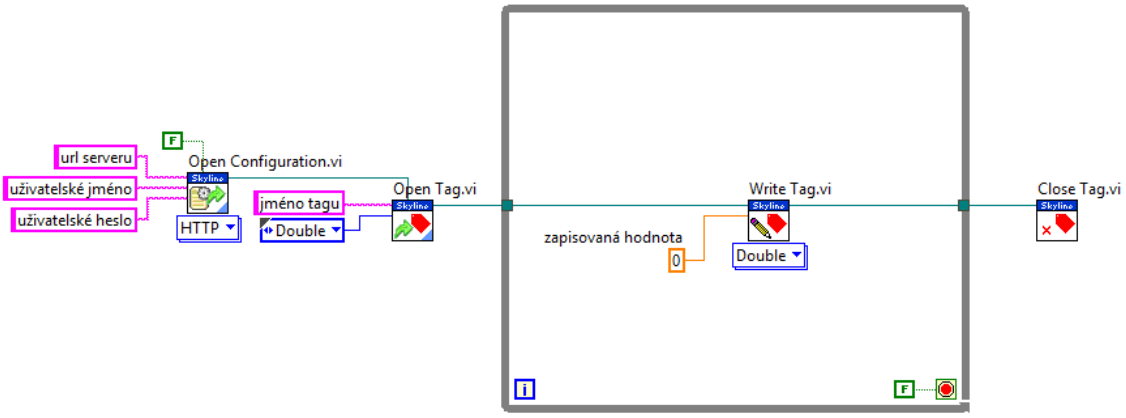
Prvky ovládacího panelu lze pomyslně rozdělit na prvky zobrazovací a prvky ovládací. Zobrazovací prvky jsou umístěny uvnitř *real-time* smyčky 3 a ovládací prvky jsou pak umístěny uvnitř *event* struktury v *real-time* smyčce 2, kde vyvolávají příslušné události. Ovládací panel slouží pouze k vizualizaci těchto prvků a tvoří tak grafické rozhraní řídicího softwaru.



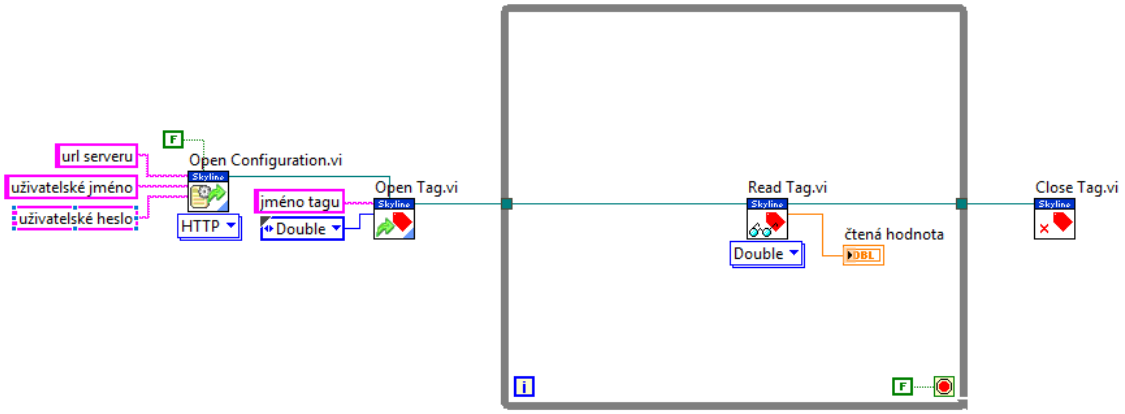
Obrázek 11: Ovládací panel

3.3 Teorie přenosu dat pomocí *tagů*

Hlavním komunikačním prostředkem pro přenos dat v tomto projektu jsou *tagy*. V prostředí LabVIEW jsou k dispozici funkční bloky, které tuto komunikaci umožňují, je však nutné je správně nakonfigurovat. Typický příklad odesílání informací pomocí *tagů* je zobrazen na obrázku 12. Přijímaní *tagů* je patrné na obrázku 13.



Obrázek 12: Zápis do *tagů*



Obrázek 13: Čtení z *tagů*

Všechny funkční bloky uvedené v tomto příkladu lze nalézt v paletě LabVIEW s názvem *Data Communication* v kategorii SystemLink.

Samotnému odesílání či přijímání *tagů* předchází konfigurace komunikace mezi zařízením se softwarem LabVIEW a serverovou aplikací SystemLink. K tomu slouží funkční blok s názvem *Open Configuration*. Funkční blok konfigurace spojení nabízí několik možností připojení k serverové aplikaci. Autorizaci lze provést obdobně jako v ukázce pomocí HTTP protokolu. Lze však také zvolit autorizaci pomocí AMQP případně API klíče. V našem případě je nutné nastavit url serveru, uživatelské jméno a heslo. Je také možné zakázat ověřování serveru konstantou *false* jako v našem případě. Výstupem bloku *Open Configuration* je informace o HTTP komunikaci, která má být navázána.

Informace o HTTP komunikaci dále vstupuje do funkčního bloku s názvem *Open Tag*. Tento funkční blok slouží k adresaci *tag*u, se kterým budeme chtít manipulovat. Funkční blok *Open Tag* identifikuje na serveru *tag* podle jeho jména, které je nutné vyplnit. Nutné je také bloku *Open Tag* říci, jakého datového typu *tag* je a s jakým datovým typem tedy bude pracovat. V našem případě se jedná o typ *Double*. Výstupem bloku je informace o *tag*u, s kterým bude manipulováno. Tímto je ukončena konfigurace navazovaného spojení. V obrázku 12 a obrázku 13 vidíme, že navazování komunikace se serverem je shodné pro zápis i čtení *tag*u. Další část programu je

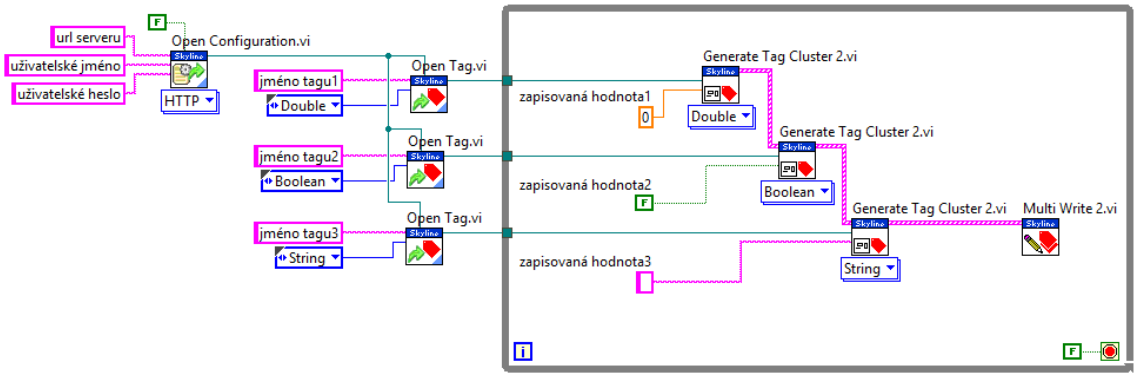
prováděna periodicky ve smyčce typu *while*, jak můžeme vidět v obrázku 12 a obrázku 13. Odesílání a přijímaní *tagů* ze serverové aplikace SystemLink je totiž zpravidla nutné provádět opakovaně a smyčka typu while je k tomuto úkonu vhodná.

Pro základní operace s *tag*em jsou v prostředí LabVIEW k dispozici dva funkční bloky, a to *Write Tag* a *Read Tag*. Funkční blok *Write Tag* umožňuje zápis hodnoty do *tag*u a funkční blok *Read Tag* umožňuje čtení hodnoty z *tag*u. Oběma těmto funkčním blokům je nutné přiřadit správný datový typ, který *tag* má, obdobně jako při nastavování funkčního bloku *Open Tag*. Zapisovaná, případně čtená hodnota *tag*u tomuto datovému typu poté odpovídá. Funkčnímu bloku *Write Tag* lze také přiřadit časovou konstantu, která udává čas zápisu daného *tag*u, z funkčního bloku *Read Tag* pak tuto konstantu lze vyčítat. Zapisovanou, resp. čtenou hodnotu lze zakomponovat do řídicí logiky softwaru tak, aby v serverové aplikaci SystemLink byly k dispozici informace o chodu softwaru, případně aby bylo možné jeho řízení.

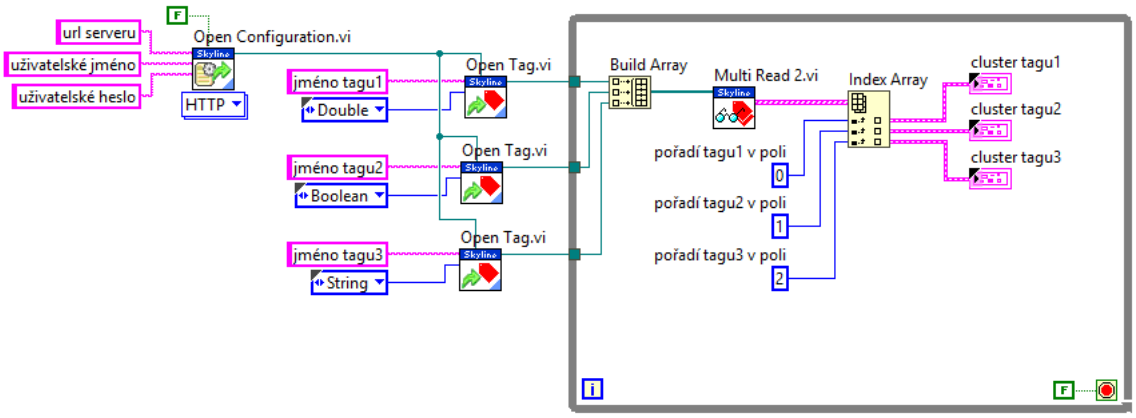
Po ukončení smyčky *while*, ve které se provádí čtení, resp. zápis do *tag*u, je možné zavřít komunikační kanál. To lze provést pomocí funkčního bloku s názvem *Close Tag*. Vstupem tohoto bloku je pouze informace o daném *tag*u. V případě že nedojde k použití funkčního bloku *Close Tag* však manipulace s *tagy* není nijak narušena.

Kromě základních funkcí *Read* a *Write* je s *tagy* možné provádět další operace. K dispozici jsou funkční bloky, které umožňují vymazat *tag*, resetovat nastřádané hodnoty v *tag*u, vybudovat novou cestu k *tag*u vyžádat si vlastnosti *tag*u a dotázat se na *tag*, resp. zjistit, jestli je *tag* dostupný. Tyto funkční bloky však nebudou nadále použity, a tudíž nejsou blíže popsány.

Speciální funkcí, která je v této práci využita, je možnost zápisu, případně čtení z více *tagů* v jednom cyklu smyčky *while*, ve které je operace umístěna. K tomu lze využít funkčních bloků *Multi Read 2* a *Multi Write 2*, kde číslovka za názvem značí pouze verzi funkčního bloku. Program je však nutné výrazně upravit. Ukázka vícenásobného čtení a zápisu je patrná z obrázku 14 a obrázku 15.



Obrázek 14: Vícenásobný zápis do *tagů*

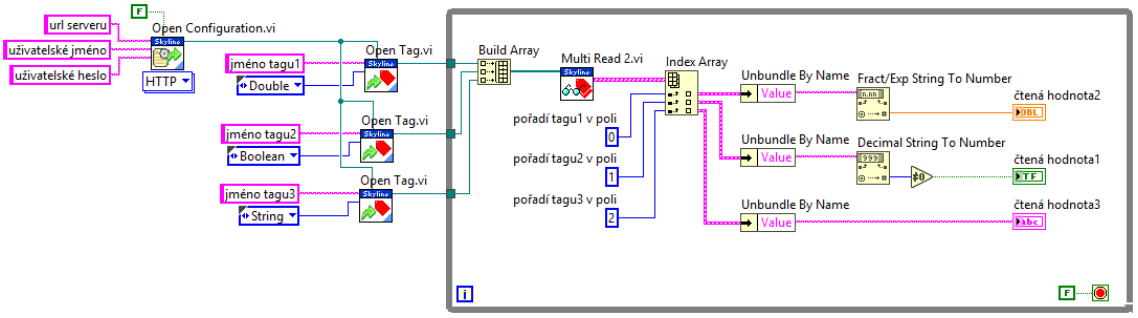


Obrázek 15: Vícenásobné čtení *tagů*

Jak je v přiložených ukázkách vícenásobného čtení a zápisu patrné, je třeba provést navázání spojení se serverem jinak než v případě základního čtení a zápisu. Prvotní autorizace je shodná, avšak adresovat je nutné každý *tag* zvlášť. V případě vícenásobného zápisu je pak pomocí funkčního bloku *Generate Tag Cluster 2* generován *cluster tagů*, který bude hromadně odeslán pomocí bloku *Multi Write 2*. Vstupem jednotlivých funkčních bloků *Generate Tag Cluster 2* je informace o každém *tag*u společně s jeho datovým typem a hodnotou, kterou je do něho potřeba zapsat. Dané funkční bloky *Generate Tag Cluster 2* jsou pak pomocí svých vstupů a výstupů propojeny tak, že výstupem posledního funkčního bloku je finální cluster. Tento cluster vstupuje do funkčního bloku *Multi Write 2*, pomocí kterého jsou hromadně zapsány údaje do vybraných *tagů*.

V případě hromadného čtení *tagů* je situace odlišná. Adresaci je nutné opět provést pro každý čtený *tag* zvlášť. Výstupy funkčních bloků *Open Tag* jsou poté sestaveny do pole pomocí bloku *Build Array*. Toto pole je předáno funkčnímu bloku *Multi Read 2*, ve kterém jsou dané *tagy* hromadně přečteny. Výstupem bloku *Multi Read 2* je opět pole, které obsahuje údaje jednotlivých *tagů*. Toto pole lze rozdělit pomocí bloku *Index Array*, přičemž indexace jednotlivých *tagů* odpovídá pořadí jejich zápisu do pole určeného ke zpracování blokem *Multi Read 2*. Po rozdělení pole na jednotlivé části získáme informace o každém *tagu*. Získané informace jsou reprezentovány *clusterem*, který nese údaje o přístupové cestě k *tag*u, datovém typu *tag*u, hodnotě *tag*u ve formátu *string* a časové známce odpovídající času pořízení *tag*u. Pokud chceme přečtené *clustery* zpracovat tak, aby bylo možné čtení hodnoty *tag*u v jejím původním formátu, je tak možné učinit například jako na obrázku 16. *Cluster* je zde rozložen pomocí funkčního bloku *Unbundle By Name* a na základě hodnoty *tag*u ve formátu *string* lze hodnotu na daný datový typ konvertovat. Způsob konverze se pro různé datové typy liší. Datový typ *tag*u je nutné uvádět již ve funkčním bloku *Open Tag*. Lze tedy předpokládat, že programátor daný datový typ zná. Informaci o datovém typu lze získat také z bloku *Unbundle By Name*, který ji poskytuje. V ukázce na obrázku 16 je konverze z datového typu *Double* prováděna pomocí funkčního bloku *Fract/Exp To Number*. Konverze datového typu *boolean* je prováděna pomocí funkčního bloku *Decimal String To Number*, jehož výstupem je však číslo ve formátu Integer, které nabývá hodnoty 0 nebo 1. Toto číslo lze konvertovat na datový typ *Boolean* pomocí ověření nerovnosti nule. Pokud se číslo nule nerovná, musí nabývat hodnoty 1 a výstupem komparačního bloku je *boolean* hodnota *true*. Když se číslo nule rovná,

výstupem komparačního bloku je hodnota *false*. Poslední čtená hodnota *tag*u je ve formátu *string*. Získanou hodnotu není tedy třeba konvertovat a pro její získání stačí použít funkční blok *Unbundle By Name.*



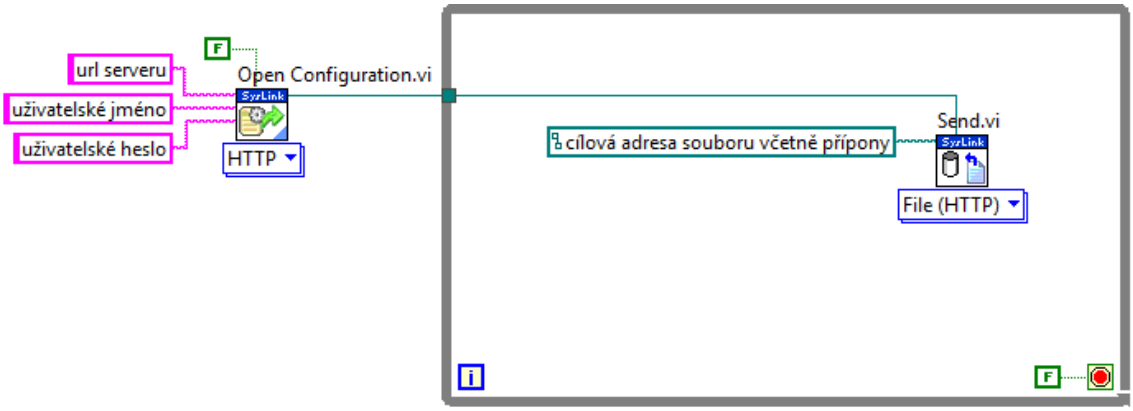
Obrázek 16: Konverze dat vícenásobného čtení

3.4 Teorie odesílání souborů do serverové aplikace SystemLink

Základní komunikace pomocí *tagů* je pro monitoring rychlých dějů nedostatečná. Pomocí tagů lze přenášet data s frekvencí do 1 Hz. K monitoringu rychlejších dějů lze vhodně využít souborů. V této práci budou přenášeny soubory ve formátu TDMS. Tento formát je výchozím formátem softwaru společnosti National Instruments a lze ho jednoduše vizualizovat v serverové aplikaci SystemLink. Soubory ve formátu TDMS obsahují množství naměřených dat, která lze například v programu DIAdem vizualizovat ve formě grafů.

Soubory lze odesílat z úložiště odesilatele. Pro účely monitoringu rychlých dějů není třeba soubory odesílat příliš často. Odesílané soubory však musí obsahovat všechny hodnoty uložené v mezičase mezi jednotlivými odesíláními. Samotné ukládání souborů do paměti kontroléru je poměrně rychlé. Naproti tomu odesílání souborů do serverové aplikace SystemLink je poměrně pomalé. Je tedy vhodné nastřádat v souboru velké množství dat a až poté tento soubor odeslat.

Samotné odesílání souborů probíhá obdobně jako práce s *tagy*. Nejprve je nutné navázat spojení odesilatele se serverem. Obdobně jako v případě *tagů* lze odesilatele autorizovat pomocí HTTP, AMQP či API klíče. Všechny funkční bloky potřebné k odesílání souborů lze nalézt v paletě LabVIEW s názvem *Data Communication* v kategorii SystemLink. K samotnému odesílání souborů lze využít funkční blok *Send File*. Vstupem do tohoto funkčního bloku je informace o navázané autorizované komunikaci a úplná adresa souboru na disku. Adresa souboru musí být zadána včetně jména a přípony daného souboru. Jednoduchý zápis programu do úložiště serverové aplikace Systemlink je patrný z obrázku 17.



Obrázek 17: Odesílání souborů

3.5 Koncepční návrh realizace IoT pro Zkušební centrum AV R&D a zkušební stav Zwich 1873

Současný řídicí software trhačky bude doplněn o extrakci dat a jejich následné odesílání do serverové aplikace SystemLink. Pro odesílání stavových proměnných bude možné použít *tagy* bez agregace hodnot. Zápis do těchto *tagů* bude vhodné provést přímo v *event* struktuře *real-time* aplikační vrstvy. Odesílání stavových proměnných do aplikace SystemLink tak bude probíhat pouze při jejich změně a zbytek programu nebude odesíláním příliš zatížen. Údaje o aktuální poloze pístu a o síle působícím na píst bude třeba odesílat do aplikace SystemLink periodicky. Za tímto účelem bude vhodné odesílání těchto *tagů* umístit do *real-time* smyčky, ve které se ukládají data ve formátu TDMS. Tato smyčka se provádí s frekvencí 10 Hz a je v ní k dispozici pole o sto prvcích požadovaných veličin vzorkovaných s frekvencí 1 kHz. Jelikož 10 Hz smyčka je pro odesílání *tagů* příliš rychlá, bude nutné smyčku doplnit o *posuvný* registr, který bude odesílání *tagů* časovat. Do *tagů* lze zapisovat pouze jednu hodnotu a ne pole. Odesílána tak bude vždy pouze první hodnota pole o sto prvcích. Odesílání bude podmíněno softwarovým tlačítkem, o které je nutné doplnit ovládací panel hlavního programu. Důležitou roli také bude hrát optimalizace programu. Je nutné, aby přidaná softwarová část nijak neomezovala chod stávajícího softwaru, a to zejména rychlost vykonávání *event* struktury. Zpoždění programu vlivem přidané softwarové části bude nutné prakticky ověřit například kontrolou včasného provedení hlavní *real-time* smyčky.

Pro monitorování rychlých dějů bude využito ukládání souborů do serverového úložiště. Konkrétně se bude jednat o soubory typu TDMS, které ponesou informaci o poloze pístu a síle, která na něj působí. V ukládaném souboru budou data vzorkovaná s frekvencí 1 kHz. Soubory nelze vizualizovat na *dashboardu* aplikace SystemLink. Přistupovat k nim lze pouze pomocí správce souborů. Soubory tak budou sloužit k analýze rychlých dějů na základě výskytu nenadálé události, jejíž hrubá podoba bude patrná z *tagů*, které na *dashboardu* vizualizovat lze.

Zpětná komunikace a řízení softwaru LabVIEW pomocí serverové aplikace SystemLink bude muset být umístěna v samostatné smyčce. Aby nedocházelo k nechtěnému přepisu *tagů*, bude nutné program doplnit o logiku vyhodnocující, zda

řízení realizovat na základě ovládacích prvků softwaru LabVIEW, nebo na základě hodnot přijatých z aplikace SystemLink.

Vizualizace průběhu zkoušky bude provedena pomocí IP kamery Edimax IC-3110W. Serverová aplikace SystemLink umožňuje na *dashboard* umístit obrázek (objekt typu *image*). V tomto objektu bude poté zobrazen obraz z IP kamery, která bude umístěna na lokální síti. Obnovovací frekvence objektu *image* bude dána nastavenou obnovovací frekvencí kamery. Jelikož kamera bude umístěna na lokální síti, nebude třeba přístup ke kameře zabezpečovat.

4 Praktická část

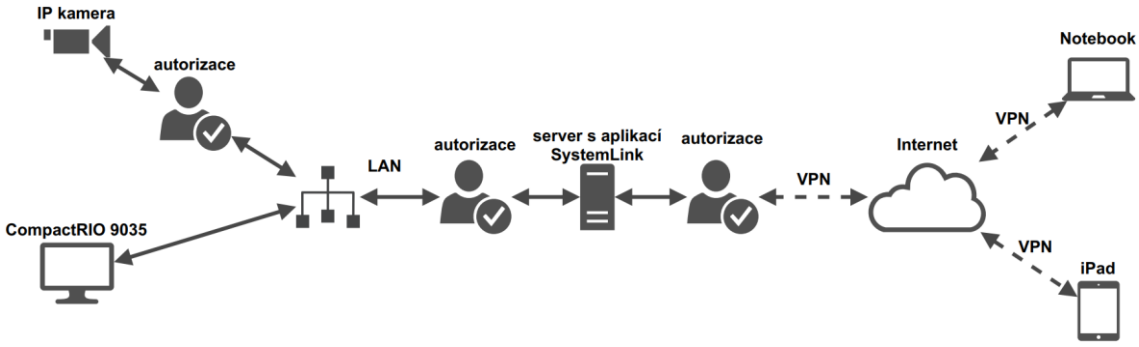
Realizace praktické části této práce vychází z konceptu a z poznatků v kapitole 3. Součástí praktické části musí být i dokumentace změn provedených v řídicím softwaru včetně popisu jejich funkce.

Praktická část se zabývá zejména předáváním dat mezi řídicím softwarem na kontroléru CompactRIO 9035 a serverovou aplikací SystemLink. Data jsou předávána pomocí *tagů* a v podobě souborů ve formátu TDMS. Tato komunikace probíhá na lokální síti. Zabezpečení komunikace spočívá ve dvoufaktorové autorizaci kontroléru serverové aplikaci SystemLink pomocí uživatelského jména a hesla. Přístup do sítě LAN mají pouze povolané osoby. Další zabezpečení komunikace není požadováno.

V praktické části je také uveden způsob realizace snímání obrazu zkoušky. Záznam obrazu je prováděn pomocí IP kamery. Kamera je opět připojena na lokální síť a přístup k obrazu z kamery je podmíněn znalostí uživatelského jména a hesla. Dodatečné zabezpečení komunikace pro lokální síť tak opět není třeba.

Data jsou střádána na serveru s aplikací SystemLink, který je umístěn v lokální síti. Pro zobrazení *dashboardu* je nutné se do lokální sítě připojit pomocí VPN. Po připojení k lokální síti lze do serverové aplikace SystemLink přistupovat na webovém prohlížeči pod uživatelským jménem a heslem. Samotná datová komunikace podléhá TLS šifrování.

Diagram předávání dat, popisující realizaci praktické části diplomové práce je znázorněn na obrázku 18.



Obrázek 18: Diagram předávání dat

Zprovoznění IoT na zkušebním stavu Zwick 1873 v sobě zahrnuje činnosti programování v LabVIEW, konfiguraci serverové aplikace SystemLink a konfiguraci parametrů jednotlivých zařízení pracujících na lokální síti.

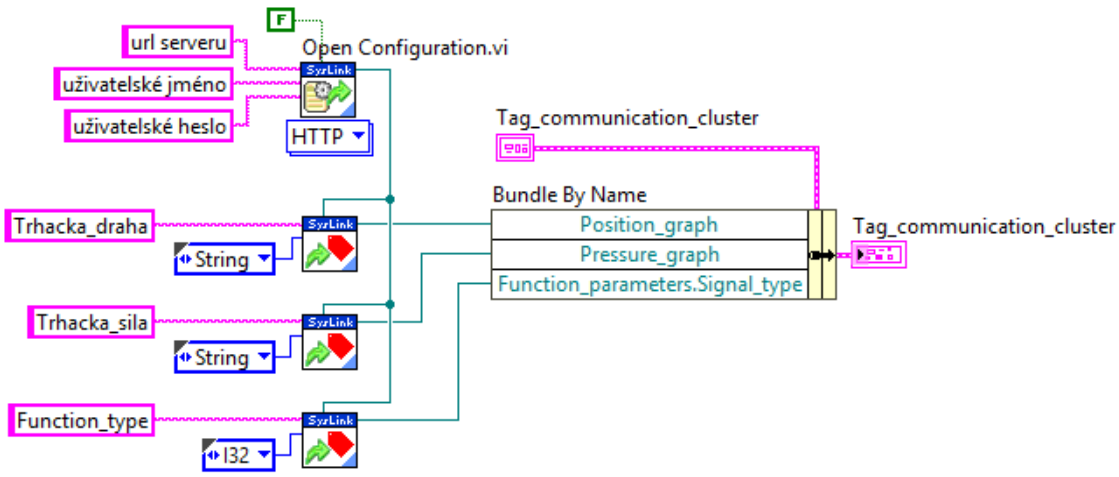
V následujících podkapitolách jsou uvedeny jednotlivé činnosti a jejich základní popis. Vlastní realizace je velmi obsáhlá a podléhá utajení požadovaném firmou AV R&D. Proto jsou zde uvedeny pouze typové příklady z vlastní realizace.

4.1 Implementace zápisu do *tagů* do softwaru trhačky

V této podkapitole jsou popsány kroky, které byly podniknuty k úspěšné implementaci zápisu informací do *tagů* do řídicího programu trhačky.

4.1.1 Konfigurace komunikačních kanálů

Před samotným odesíláním *tagů* je třeba nakonfigurovat komunikační cestu mezi jednotlivými *tagy*, umístěnými na serveru a zařízením, které s nimi bude pracovat. Konfigurace komunikačních kanálů proběhla v SubVI s názvem *Tag_definition*. V tomto SubVI je provedeno přihlášení na server pomocí přihlašovacích údajů a následně je každému *tag*u přiřazen název a datový typ. Nadefinované komunikační kanály jsou zapsány do předem předdefinovaného clusteru, se jménem *Tag_communication_cluster* pomocí funkčního bloku *Bundle By Name*. Tento cluster bude sloužit k předávání informace o požadovaném *tag*u, případně seskupení *tagů* jednotlivým smyčkám typu *while*. Kromě informací týkajících se *tagů* je do clusteru také zapsána informace o prvotní konfiguraci komunikace se serverovou aplikací SystemLink, která bude později v programu využita pro přenos TDMS souborů. Nově vzniklý *cluster* lze rozkládat pomocí funkčního bloku *Unbundle By Name* a po rozložení lze pomocí jména přistupovat k jednotlivým komunikačním kanálům *tagů*. Podoba konfigurace komunikačních kanálů a jejich zápisu do *clusteru* je patrná z obrázku 19. Na něm je uvedena konfigurace komunikačních kanálů pouze pro tři tagy. Konfigurace pro všechny tagy užité v práci je provedena obdobným způsobem, avšak její snímek by byl pro účely práce zbytečně podrobný a příliš rozměrný.



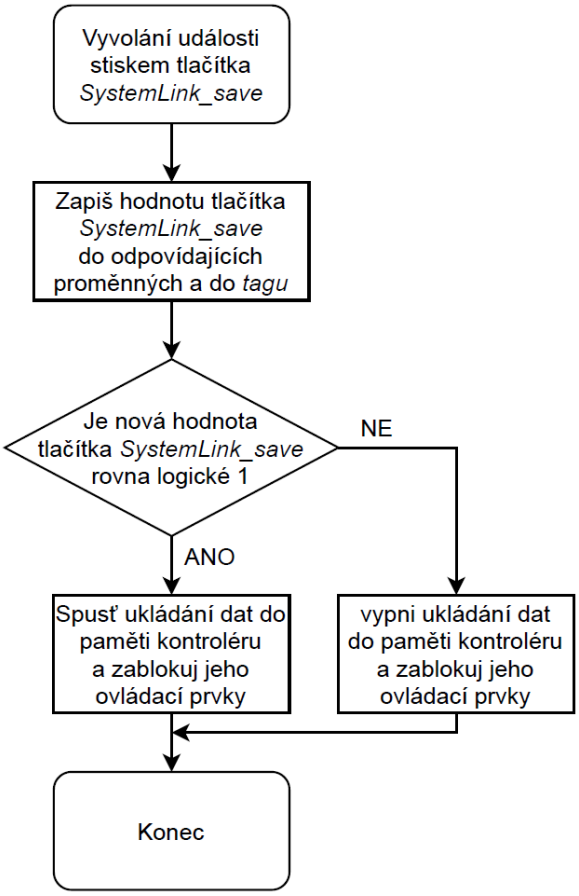
Obrázek 19: Konfigurace komunikačních kanálů

4.1.2 Spouštěč odesílání *tagů*

Komunikace mezi kontrolérem CompactRIO a serverovou aplikací SystemLink nemusí probíhat nepřetržitě. Monitoring bude vyžadován převážně v případě, že ve zkušebně, kde je trhačka umístěna, nebude přítomen personál obsluhy stroje. Za tímto účelem byl do programu implementován softwarový spouštěč ukládání dat na server. Do oblasti ovládacího panelu bylo umístěno nové tlačítko s názvem *SystemLink_save*.

Změně stavu tohoto tlačítka byla přiřazena nová událost ve struktuře *event*, která je vyvolána po stisku tlačítka. V události dojde k zápisu nové hodnoty tlačítka do stejnojmenné proměnné datového typu metody *FIFO* a do globální proměnné, která taktéž nese jméno *SystemLink_save*.

Kromě zápisu do dvou proměnných dojde také k zápisu do *tagu* a k provedení *case* struktury upravující ovládací prvky *Local_save* a *Filename*. Zápis do *tag*u je realizován pro *tag*, reprezentující stav tlačítka *SystemLink_save* a od ukládání ostatních *tagů* se liší tím, že zápis do něho není podmíněn stavem proměnné *SystemLink_save*. Samotná realizace zápisu do *tagů* je popsána v kapitole 4.1.3. Úprava ovládacích prvků *Local_save* a *Filename*, které slouží ke spouštění ukládání a pojmenování TDMS souborů na diskovou jednotku zařízení, spočívá ve změně hodnoty tlačítka Local_save a zakázání další manipulace s oběma prvky po spuštění zápisu dat do aplikace SystemLink. Data ve formátu TDMS odesílaná do aplikace SystemLink jsou totiž podmíněna jejich přítomností na disku. Je tedy nutné spustit ukládání na disk, které musí probíhat po celou dobu komunikace s aplikací SystemLink, a během něho nelze měnit název TDMS souboru. Logika v události vyvolané stiskem tlačítka *SystemLink_save* je patrná z vývojového diagramu na obrázku 20.



Obrázek 20: Vývojový diagram spouštění ukládání

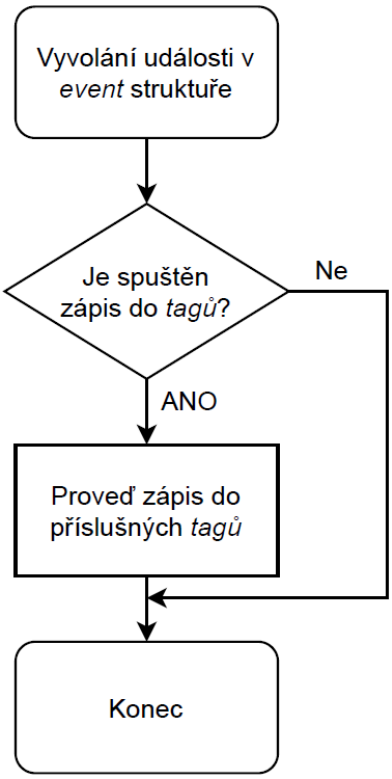
Jak již bylo zmíněno, ve vyvolané události dojde ke změně stavu dvou proměnných. Jedna proměnná je umístěna v datovém typu metody FIFO a druhá je globální. Proměnná v datovém typu metody FIFO slouží ke spouštění zápisu dat

o poloze pístu a síle, která na něj působí do fronty *queue*. Kromě spouštění zápisu dat ovládá tato proměnná také struktury *case*, které jsou umístěny v jednotlivých událostech *event* struktury a podmiňují zápis do jednotlivých *tagů*.

Globální proměnná *SystemLink_save* je využívána v místech, kde není užita metoda FIFO případně *event* struktura. Aplikace globální proměnné bude blíže rozebrána v následujícím textu.

4.1.3 Zápis do *tagů*

Pro zápis do *tagů*, který není nutné provádět s vysokou frekvencí, je využita *event* struktura ve stávajícím programu. Jedná se o *tagy* obsahující informace o chybových hláškách a aktuálních stavech jednotlivých tlačítek ovládacího panelu. Event struktura reaguje na změnu stavu daných proměnných vyvoláním události, ve které je možné hodnoty daných proměnných zapsat do *tagů*. Zápis do těchto tagů je tzv. *event-based*. Řídicí logika odesílání *tagů* je patrná z vývojového diagramu na obrázku 21. Zápis hodnoty do jednoho či více tagů probíhá způsobem popsaným v kapitole 3.3.



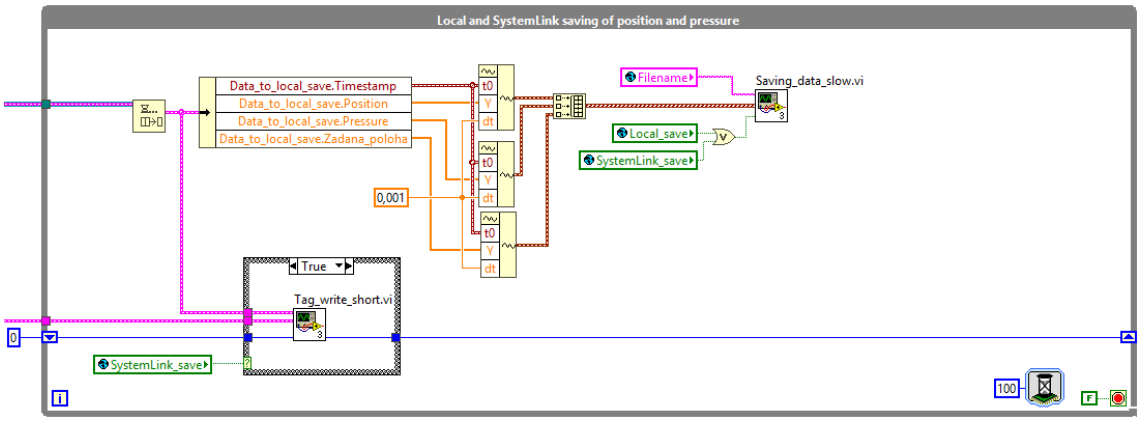
Obrázek 21: Vývojový diagram zápisu do *tagů* v *event* struktuře

Informace o komunikačním kanálu *tagů* je získávána z *clusteru* popsaného v odstavci 4.1.1, funkčním blokem *Unbundle By Name*.

Hlavní výhoda umístění zápisu do *tagů* do *event* struktury spočívá v tom, že k odeslání *tagů* dojde pouze v případě, že nastane změna hodnoty proměnných, které daným *tagů*m přísluší. Zápis *tagů* tak nemusí být periodický a ani nemusí být ošetřen další přídavnou logikou, která by chod programu zbytečně zatěžovala.

Zápis do *tag*u na základě vyvolání události *event* struktury je vhodný pro použití s proměnnými, které svou hodnotu nemění příliš často. V této práci je však nutné monitorovat také polohu pístu a sílu, která na něho působí. To jsou proměnné, jejichž hodnotu je nutné do *tagů* zapisovat periodicky a co nejrychleji. Údaj o aktuální poloze pístu a síle, která na něho působí je přenášen pomocí *queue* fronty. Obsah fronty je čten v *real-time* smyčce 4, jejíž funkce byla probrána v odstavci 3.2.2.4.

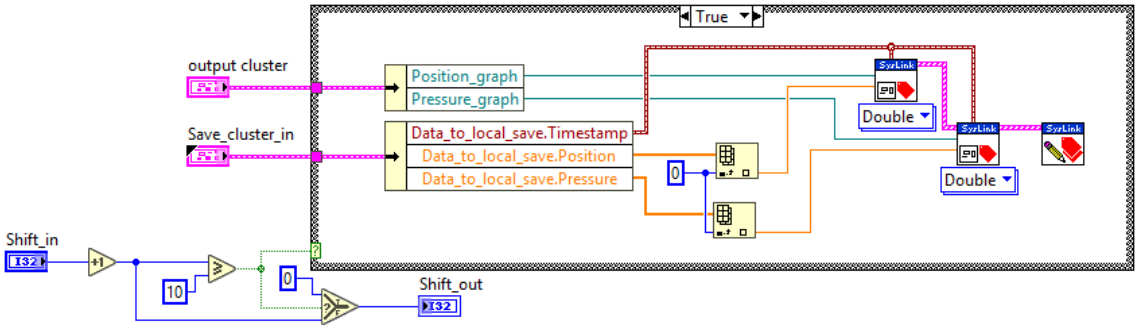
Pro zjednodušení přenosu dat je vhodné zápis příslušných hodnot do *tagů* provést také v této smyčce. Samotný zápis do *tagů* probíhá v SubVI *Tag_write_short*, které je umístěno v *case* struktuře. Struktura *case* je ovládána pomocí globální proměnné *SystemLink_save*, která je popsána v odstavci 4.1.2. Implementace SubVI do *real-time* smyčky 4 je patrná z obrázku 22.



Obrázek 22: Zapisování polohy a síly

Na obrázku 22 si můžeme všimnout také nově definovaného posuvného registru. Tento registr slouží k časování zápisu do *tagů*. Zapisovat do *tagů* lze s frekvencí maximálně 1 Hz. Je tedy nutné upravit časování zápisu do *tagů*, aby nedocházelo ke zbytečným pokusům o jejich zápis a tím ke zbytečnému zatěžování programu. Víme, že celá *real-time* smyčka 4 se provádí každých 100 ms, pokud budeme posuvný registr každý cyklus inkrementovat o jednotku, můžeme z něho v SubVI zjistit údaj o uplynulém čase s rozlišením 100 ms. Reálně je tedy posuvný registr inkrementován po jedné a pokud dosáhne hodnoty 10, je proveden program v *case* struktuře, který zapíše aktuální hodnoty do *tagů* a vynuluje posuvný registr. Uvážíme-li, že posuvný registr je před nulováním inkrementován desetkrát a doba mezi inkrementacemi je 100 ms, pak zjistíme, že program ve struktuře *case* se provádí každou jednu sekundu. Podoba této logiky je patrná v SubVI *Tag_write_short* zobrazeném na obrázku 23.

Obrázek 23 dále ukazuje, že při provádění programu ve struktuře *case* je vždy čtena pouze nultá hodnota z polí obsahujících údaje o poloze pístu a síle, která na něj působí, a také čas pořízení hodnot. Tyto hodnoty jsou následně zapsány do příslušných *tagů* pomocí mnohonásobného zápisu. Čtení pouze první hodnoty je dáno faktem, že pomocí *tagů* lze přenášet najednou pouze jednu hodnotu. Při pokusu o zapsání pole bychom narazili na nekompatibilitu datových typů. Řešením by byla datová konverze pole na datový typ *string*, avšak serverová aplikace SystemLink v tomto formátu vyhodnocuje a zobrazuje pouze první proměnnou. Zbylé proměnné v řetězci jsou pouze načteny a zahozeny, dochází tedy ke zbytečnému zatěžování komunikačního kanálu.



Obrázek 23: *Tag_write_short* SubVI

Poslední podstatný prvek nové části programu je vidět na obrázku 22. Jedná se o zakomponování globální proměnné *SystemLink_save* do ovládání ukládání na disk kontroléru. Ukládání lze nyní spouštět buď proměnnou *Local_save*, nebo proměnnou *SystemLink_save*, přičemž ovládání samotné je řízeno logickým hradlem *or*. Spouštění zápisu dat na disk pomocí proměnné *SystemLink_save* bude využito k odesílání souborů do aplikace SystemLink, které bude popsáno dále.

4.2 Odesílání souborů do serverové aplikace SystemLink

Přenášení informací pomocí *tagů*, které bylo probráno v kapitole 4.1, je vhodné k monitorování a vizualizaci pomalých dějů. Při rychlých změnách zjistíme, že přenesené informace mají malou výpovědní hodnotu. Tento nedostatek můžeme zaznamenat v případě přenášení údajů o poloze pístu a o síle, která na něj působí. Odpovídající *tagy* vizualizované na dashboardu slouží tak pouze k hrubé informaci o stavu pístu. Pro bližší posouzení probíhajících dějů je nutné analyzovat detailnější data. Za tímto účelem lze vhodně využít souborů TDMS, do kterých lze požadované průběhy ukládat. Uložené soubory je možné odeslat do serverového úložiště, které spravuje aplikace SystemLink.

4.2.1 Implementace ukládání souborů do softwaru trhačky

Původní program zvládá ukládat data navzorkovaná s frekvencí 100 Hz, do souborů ve formátu TDMS. Cílová adresa TDMS souborů je známá a jednotlivé soubory jsou programem ukládány do paměti kontroléru. Z paměti kontroléru je možné tyto soubory odeslat do serverové aplikace SystemLink. Pro úspěšné odeslání souboru je nutné znát jeho úplnou adresu, která musí obsahovat i název a příponu souboru a informaci o komunikaci s aplikací SystemLink. Název souboru lze získat z globální proměnné *Filename*, která se vyplňuje před spuštěním ukládání TDMS souborů a přípona TDMS je pro všechny zde užité soubory neměnná. Finální podoba adresy souboru je sestavena pomocí funkčního bloku *Build Path*. Informaci o komunikaci s aplikací SystemLink lze získat z clusteru blíže popsaném v odstavci 4.1.1. Samotné odesílání souborů probíhá způsobem popsaným v kapitole 3.4.

Stejně jako periodický zápis do *tagů* je i odesílání souborů časováno. Soubory není nutné odesílat příliš často, a tak je jejich odesílání prováděno každých 10 s. Časování odesílání souborů je provedeno stejným způsobem jako časování periodického zápisu

do *tagů*, který je popsaný v kapitole 4.1.3. Jelikož je k časování použit stejný posuvný registr jako v případě časování zápisu do *tagů*, je program řídící odesílání souborů umístěn ve stejném SubVI *real-time* smyčky 4 jako periodický zápis do *tagů*.

Pokud bychom soubory na server pouze zapisovali, hromadilo by se zde velké množství neužitečných dat. Každý nový uložený soubor totiž obsahuje jak data předchozího souboru, tak data nová. Při úspěšném uložení nového souboru není již starý soubor potřeba a dojde k jeho smazání.

Odesílání souborů doplňuje zápis do *tagů*. Nezvyklé chování nebo nenadálé události na monitorovaném stroji lze pozorovat i pomocí pomalých *tagů* s informacemi hrubého informativního charakteru, které jsou vizualizovány na *dashboardu*. Bližší prozkoumání dané události pak lze provést pomocí prohlížeče souborů v úložišti serverové aplikace SystemLink. Prohlížeč umožňuje otevřít a vizualizovat soubory TDMS, které požadované informace obsahují.

4.3 Implementace čtení z tagů do softwaru trhačky

Pomocí zápisu do *tagů*, v softwaru trhačky lze střádat a vizualizovat data vypovídající o aktuálním stavu trhačky v serverové aplikaci SystemLink. Aplikace Systemlink však také umožňuje zápis do *tagů*, a to pomocí ovládacích prvků umístěných na *dashboardu*. Ovládací prvky lze spojit s jednotlivými *tagy* tak, že je-li provedena změna hodnoty *tag*u v softwaru trhačky, dojde ke změně vizuální podoby daného ovládacího prvku na *dashboardu* aplikace SystemLink. Pokud je měněna hodnota ovládacího prvku na *dashboardu*, dojde také ke změně hodnoty v *tag*u, kterou je možné vyhodnocovat v softwaru trhačky a na jejím základě trhačku řídit.

4.3.1 Spouštění čtení z *tagů*

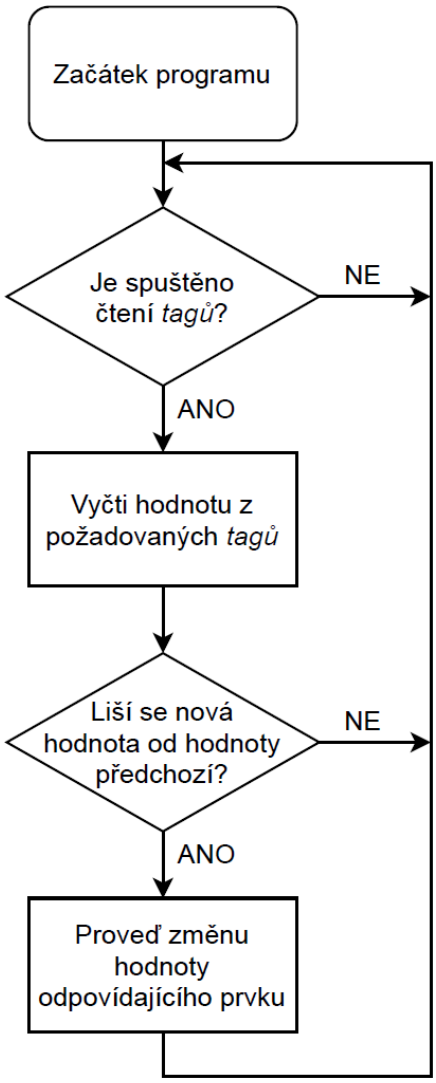
Pokud chceme umožnit řízení trhačky pomocí ovládacích prvků na *dashboardu* serverové aplikace SystemLink, je nutné software trhačky doplnit o logiku, která vyhodnotí, zdali byla hodnota *tag*u změněna řídicím softwarem trhačky, nebo ovládacím prvkem *dashboardu*. Pokud bychom neprovedli rozlišení zdroje změny, docházelo by k nechtěným přepisům *tagů* na hodnotu zdroje poslední změny. Program by se tak stal neřiditelným.

Za účelem spouštění čtení z *tagů* byl vytvořen ovládací prvek na hlavním panelu softwaru trhačky. Jedná se o tlačítko s názvem *SystemLink_control*. Změna stavu tohoto tlačítka vyvolá událost v *event* struktuře. V této události dojde ke změně hodnoty globální proměnné *SystemLink_control* a proměnné datového typu metody FIFO. Hodnota obou proměnných tak bude vždy odpovídat stavu tlačítka *SystemLink_control*. Ve vyvolané události také dojde k zablokování těch ovládacích prvků hlavního panelu, které bude možné ovládat vzdáleně pomocí aplikace SystemLink. Blokování prvků je řízeno strukturou *case*, která je ovládána aktuální hodnotou tlačítka *SystemLink_control*. Blokace tlačítek je nutná, aby nedocházelo k nechtěné změně hodnoty v *tag*u. Pokud by nedošlo k blokaci ovládacích prvků, do *tagů* by mohly být zapisovány hodnoty ze dvou různých ovládacích prvků s různou hodnotou. Hodnoty jednotlivých *tagů* by se nastavovaly podle zdroje s nejnovější hodnotou a *tagy* by tak nebyly pro řízení trhačky použitelné.

4.3.2 Čtení z *tagů*

Čtení z *tagů* je realizováno v samostatné smyčce typu *while*, umístěné v *real-time* části programu způsobem popsaným v kapitole 3.3. Tato smyčka je prováděna každou jednu sekundu. Provedení programu uvnitř smyčky je podmíněno sepnutím tlačítka *SystemLink_control*, resp. změnou stavu globální proměnné *SystemLink_control*.

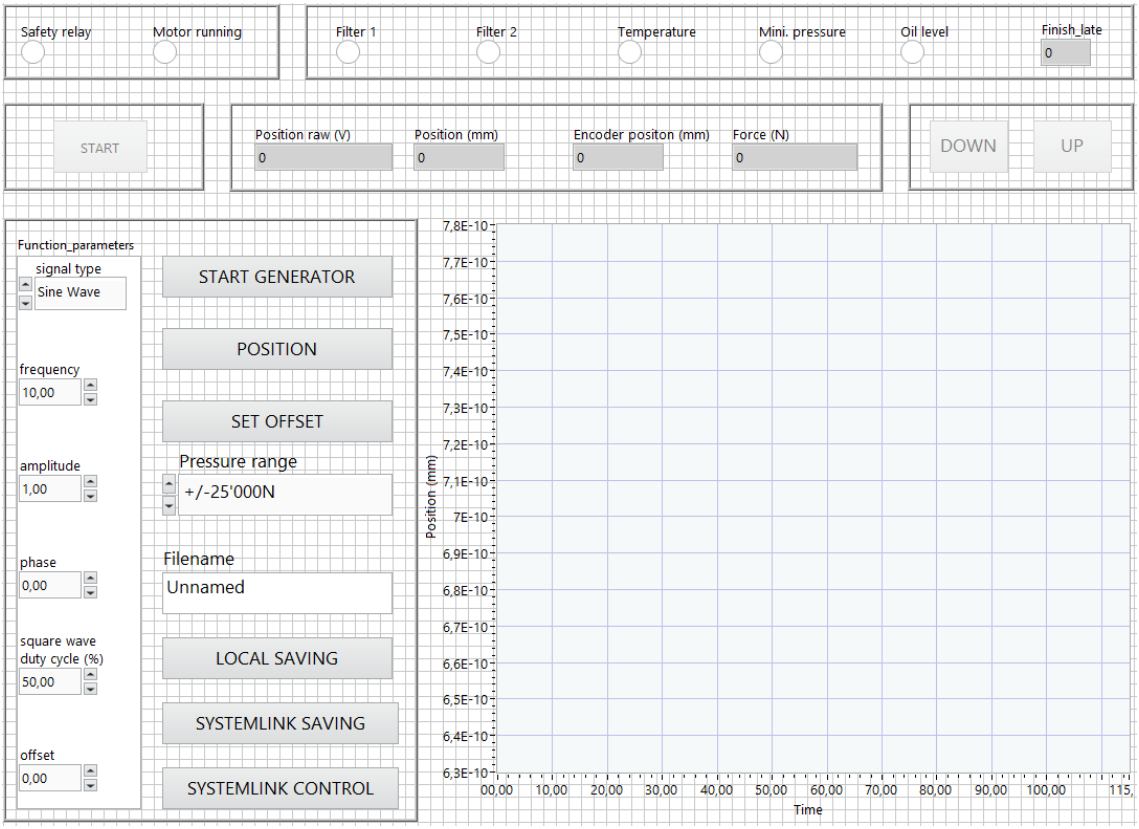
Pro každý z ovládaných prvků je vytvořen uzel vlastností (*property node*), který umožňuje měnit stav ovládacího prvku na hlavním panelu. Ovládány jsou pouze prvky, které byly v události *event* struktury zablokovány. Po zablokování ovládacích prvků na ovládacím panelu může dojít ke změně jejich hodnot pouze na základě zápisu hodnoty do jejich uzlu vlastností. Do uzlů vlastností umístěných ve smyčce jsou zapisovány jednotlivé hodnoty čtených *tagů*. Tímto způsobem dochází k ovládání kontrolních prvky ovládacího panelu řídicího programu trhačky. Podoba samotné smyčky není pro vysvětlení funkce programu vhodná. Logika programu čtení *tagů* a změny ovládacích prvků je tak demonstrována pomocí vývojového diagramu na obrázku 24.



Obrázek 24: Vývojový diagram čtení z *tagů*

4.4 Úpravy ovládacího panelu programu hydraulické rychlotrhačky

Ovládací panel hydraulické rychlotrhačky doznal v průběhu úprav softwaru také několika změn. Změny odpovídají úkonům popsaným v kapitolách 4.1.2 a 4.3.1. Bylo přidáno tlačítko určené ke spouštění zápisu do *tagů* s názvem *SystemLink_saving* a také tlačítko *SystemLink_control*, které slouží ke spouštění čtení z *tagů* a ovládání hlavního programu pomocí serverové aplikace SystemLink. Podoba ovládacího panelu s uvedenými změnami je patrná z obrázku 25.



Obrázek 25: Upravený ovládací panel

4.5 Realizace snímání obrazu průběhu zkoušky

Monitorování funkce zkušebního stavu bylo realizováno pomocí IP kamery Edimax IC-3110W, která umožňuje snímat zkušební stav s rozlišením 1280 x 1024 pixelů a frekvencí 15 snímků za sekundu. Bylo však nutné provést prvotní konfiguraci kamery.

Kameře byla přiřazena volná IP adresa v lokální síti. Dále byla nastavena příslušná brána a maska podsítě. Takto nakonfigurovanou kameru lze připojit pomocí *ethernetového* kabelu kdekoliv v prostorách zkušebního centra AV R&D, jehož *ethernetové* porty jsou sdruženy v lokální síti. Ke kameře lze přistupovat pomocí IP adresy například skrz webový prohlížeč. Zobrazení kamery je podmíněno autorizací.

Obraz kamery je vizualizován v aplikaci SystemLink pomocí objektu image. Vstupním parametrem objektu image je pouze adresa obrazu kamery. Po prvotní autorizaci se ve webovém prohlížeči spustí přenášení obrazu do serverové aplikace SystemLink a průběh zkoušky tak lze monitorovat na *dashboardu*.

4.6 Práce s daty pomocí serverové aplikace SystemLink

V předchozích kapitolách byl probrán přenos dat na úrovni aplikace LabVIEW. Nyní bude vysvětleno, jakým způsobem lze nakládat s přenesenými daty pomocí serverové aplikace SystemLink.

Po přihlášení do prostředí serverové aplikace SystemLink se zobrazí menu, ze kterého lze přistupovat k jednotlivým funkcím aplikace. Je zde možnost spravovat připojená zařízení, editovat *tagy*, editovat soubory a spravovat, případně vytvářet *dashboardy*.

Správa připojených zařízení nabízí přehled a kontrolu nad zařízeními, která mohou s aplikací SystemLink navazovat komunikaci. Lze spravovat stolní počítače a různá jiná zařízení z produkce společnosti National Instruments. Spravovaná zařízení nemusí být na lokální síti. Možnost přístupu k spravovanému zařízení je dána především jeho umístěním a viditelností v internetové síti vůči umístění serveru. V této sekci aplikace lze přidávat nová zařízení a spravovat ta, která jsou již připojena.

Sekce aplikace SystemLink věnovaná *tagům* umožňuje tvorbu nových *tagů* a editaci *tagů* stávajících. Při tvorbě *tagu* dostaneme na výběr, jaký datový typ má být *tagu* přiřazen a jestli mají být hodnoty *tagu* ukládány. Editace stávajících *tagů* nabízí možnost změny parametrů *tagů*, případně jejich smazání. *Tagům* lze také přiřadit klíčová slova a předepsané vlastnosti. Tyto parametry slouží k usnadnění třídění *tagů*. Za předpokladu, že je nastaveno ukládání hodnot *tagu*, vidíme po otevření konkrétního *tagu* také podobu záznamu jeho hodnot v číselné tabulce a v grafu.

Správa souborů spočívá v monitoringu stávajících souborů, případně v přidávání souborů nových. Soubory je možné na server jednoduše nahrát z libovolného adresáře v lokální síti. Soubory uložené na serveru lze prohlížet. Prohlížení je umožněno pouze u souborů s podporovaným formátem. Funkce prohlížení souborů je využita k monitorování rychlých dějů, které funguje na principu popsaném v kapitole 4.2. Soubory v serverové aplikaci SystemLink lze samozřejmě také mazat.

Významnou funkcí aplikace SystemLink je tvorba *dashboardů*. Pomocí *dashboardu* jsou vizualizovány jednotlivé *tagy*. *Dashboard*, který byl vytvořen pro hydraulickou rychlotrhačku, je jakousi zjednodušenou kopií ovládacího panelu hlavního programu. Nachází se zde většina ovládacích a kontrolních prvků ovládacího panelu rychlotrhačky, která je rozmístěna podobně jako na ovládacím panelu rychlotrhačky. Ovládání a monitoring rychlotrhačky je tak intuitivní a pro uživatele původního systému snadno zapamatovatelné. Podoba *dashboardu* hydraulické rychlotrhačky je patrná z obrázku 26.

Obrázek 26: *Dashboard* hydraulické rychlotrhačky

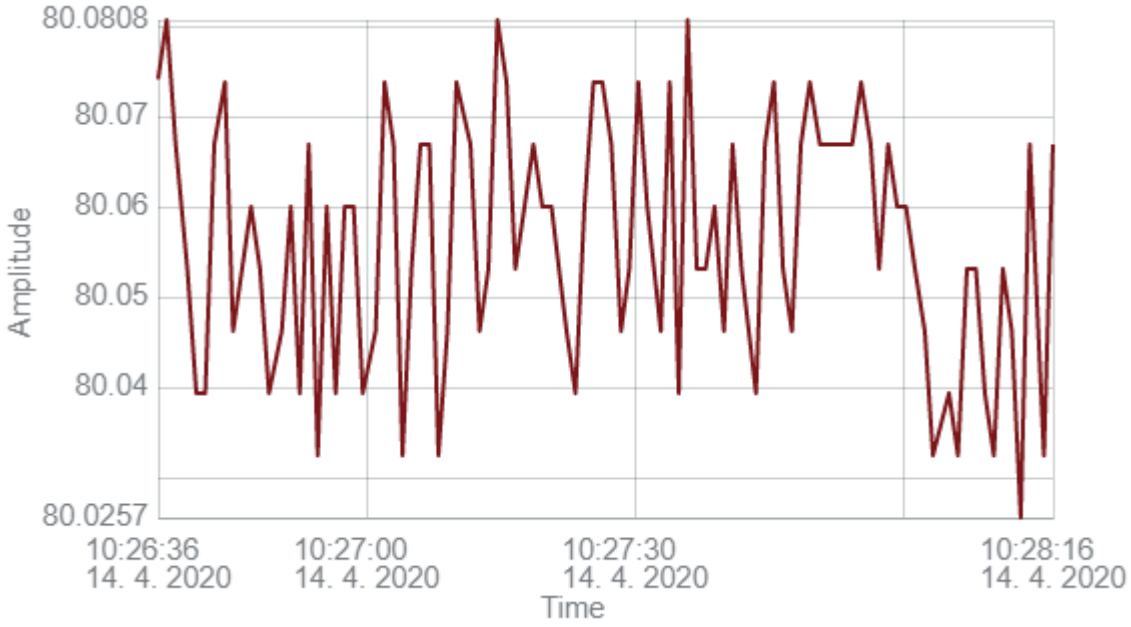
4.7 Verifikace IoT zkušebního stavu Zwick 1873

Cílem verifikace IoT je zjistit kvalitu přenášených dat, stanovit rizika špatného užívání či interpretace zobrazovaných hodnot a případný vliv uživatele.

4.7.1 Zhodnocení kvality přenesených dat

Pro reálné využití monitoringu zkušebního stavu je nutné ověřit výpovědní hodnotu přenášených dat. Pro ověření funkčnosti přenosu dat byla trhačka řízena podle harmonického signálu s offsetem 80 mm, amplitudou 2 mm a frekvencí 3 Hz. Zpětnou vazbou PID regulátoru byla v tomto případě poloha pístu. Zaznamenávaný signál bude dále označován jako „testovací průběh“. V tomto odstavci jsou srovnávány hodnoty odpovídající poloze pístu, které byly přeneseny pomocí *tagů*, a hodnoty, které byly přeneseny pomocí souborů ve formátu TDMS.

Jak již bylo uvedeno v odstavci 4.1.3, periodický zápis do *tagů* se provádí s frekvencí 1 Hz. Není tedy dodržen Nyquistův-Shannonův-Kotělnikovův vzorkovací teorém. Předpokladem tak je, že průběh polohy pístu o daných parametrech, monitorovaný pomocí *tagů*, nebude mít žádnou výpovědní hodnotu. Podoba tohoto průběhu je patrná z obrázku 27, na kterém je objekt typu *chart* umístěný na *dashboardu*.



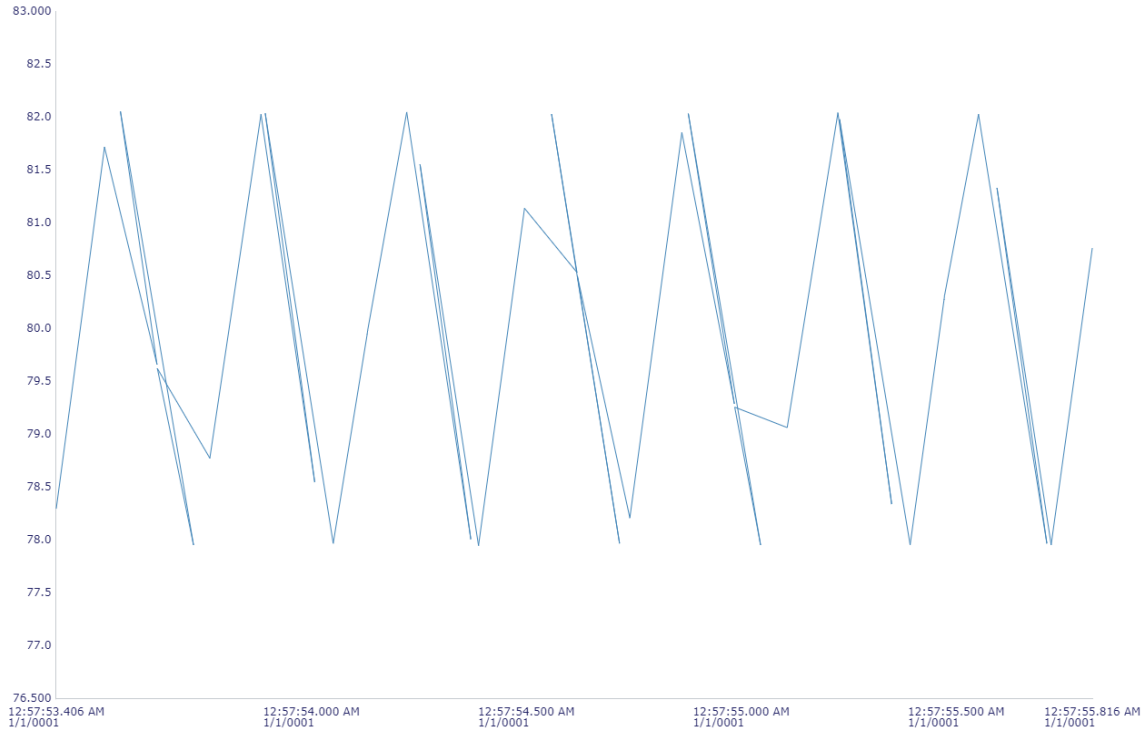
Obrázek 27: Testovací průběh dráhy monitorovaný pomocí *tagů*

Jak je z obrázku 27 patrné, parametry testovacího signálu určit nelze, a to ani přibližně. Průběh může sloužit pouze k detekci překročení mezních parametrů řízení např. při přetržení zkušebního vzorku, který by mohl být v trhačce upnut. Překročení mezních parametrů pak může být podnětem k detailnímu zkoumání průběhu jiným způsobem než pomocí hodnot uvnitř *tagů*.

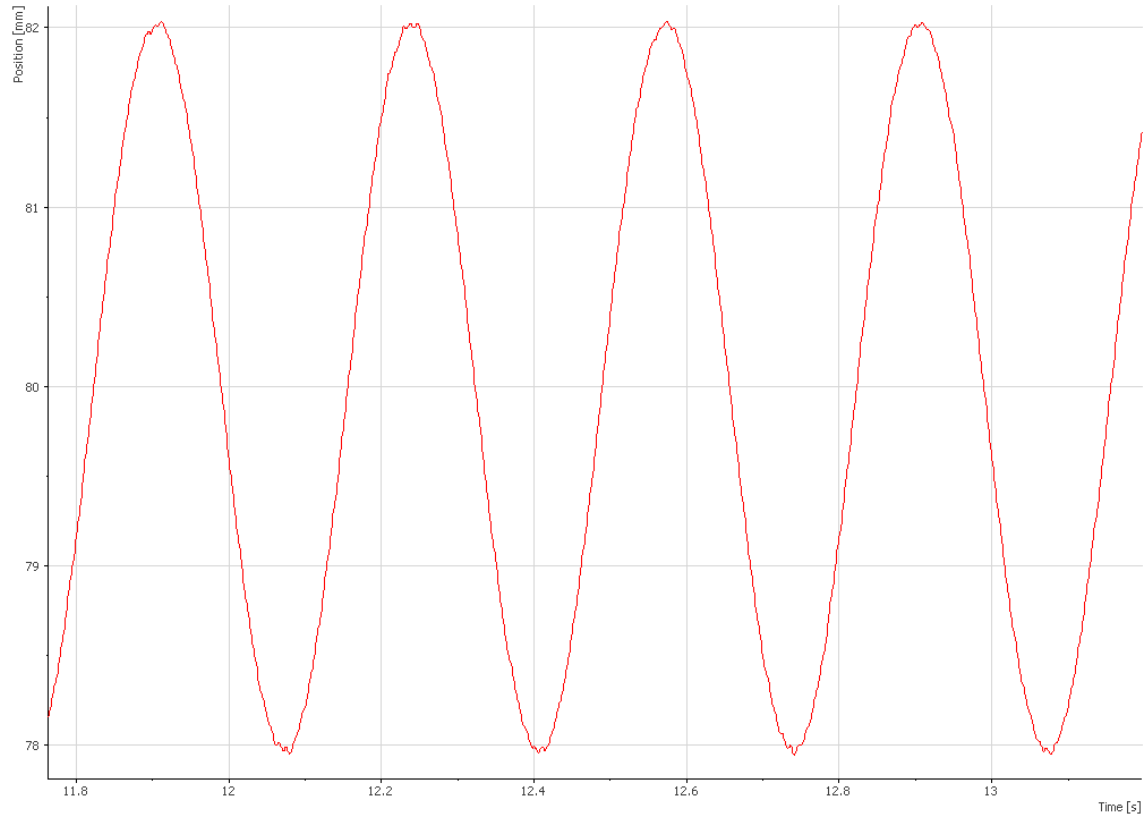
Za účelem monitorování dějů, které jsou prováděny s vyšší frekvencí, než je vzorkovací frekvence *tagů*, jsou do serverové aplikace SystemLink odesílány soubory ve formátu TDMS. V tomto případě je Nyquistův-Shannonův-Kotělnikovův vzorkovací teorém splněn, protože soubory jsou vzorkovány s frekvencí 1 kHz. Předpokladem tak je, že průběhy v TDMS souborech budou mít vysokou výpovědní hodnotu. Podoba takového souboru vizualizovaného pomocí prohlížeče souborů v aplikaci SystemLink je patrná na obrázku 28.

Jak si můžeme z obrázku 28 všimnout, časová osa neodpovídá realitě a jednotlivé body v TDMS souborech nejsou prokládány zcela optimálně. Prohlížeč souborů aplikace SystemLink nedokáže pracovat s časovým formátem, který používá kontrolér CompactRIO 9035 a současná verze aplikace LabVIEW neumožňuje formát časové stopy. Časová stopa je tak vyhodnocována nesprávně a jako počáteční čas je brán výchozí čas aplikace SystemLink. Časový interval mezi jednotlivými body je zobrazován správně. Chybné proložení bodů je způsobeno špatnou optimalizací prohlížeče souborů aplikace SystemLink. Prohlížeč souborů nezvládá vykreslovat všechny zaznamenané body a body, které vykreslí, nejsou vždy spojovány se správnou časovou posloupností. Průběhy tak působí kostrbatě a nejsou vhodné k bližší analýze. K analýze rychlých dějů v prohlížeči souborů serverové aplikace SystemLink lze použít tabulku dat, ve které jsou jednotlivé naměřené body zobrazeny korektně.

Pro kvalitnější zobrazení průběhů v TDMS souborech je možné tyto soubory stáhnout do počítače a zobrazit pomocí aplikace DIAdem. Podoba totožného souboru zobrazeného pomocí aplikace DIAdem je patrná z obrázku 29.



Obrázek 28: Testovací průběh dráhy monitorovaný pomocí TDMS souborů



Obrázek 29: Testovací průběh zobrazený pomocí aplikace DIAdem

V obrázku 29 vidíme, že jednotlivé body jsou propojeny odpovídající křivkou a průběh dráhy při daném řízení je vizualizován mnohem lépe než pomocí správce souborů aplikace SystemLink. Korektní je také čas na ose x. V obrázku 29 jsou zobrazeny sekundy přiblíženého časového intervalu, které odpovídají skutečnému času pořízení.

4.7.2 Rizika

V případě přenosu dat do serverové aplikace SystemLink je největším rizikem nedostatečně rychlé vyhodnocení kritického stavu. Toto vyhodnocení může být značně ovlivněno malou přenosovou rychlostí *tagů*, které jsou jediným prvkem vizualizovatelným na *dashboardu*. Obnovovací frekvence *tagů* je 1 Hz. V případě zkoušek, které jsou rychlejší než tato frekvence, se může stát, že obsluha vyhodnotí kritický stav (například přetržení zkušebního vzorku) s jistým zpožděním.

Dalším rizikem používání vzdáleného monitoringu je chybné vyhodnocení průběhů v souborech TDMS pouze na základě jejich vizuální podoby. V prostředí serverové aplikace SystemLink je soubory nutné vyhodnocovat na základě tabulky hodnot jednotlivých bodů. Vykreslování průběhů není zcela optimalizováno, a tak má pouze malou výpovědní hodnotu.

Je nutné, aby byla obsluha *dashboardu* serverové aplikace SystemLink s výše uvedenými riziky dobře obeznámena a při monitoringu zkušebního stavu brala tyto rizika v potaz.

4.7.3 Vliv uživatele

Primární funkce uživatele je monitoring průběhu zkoušky. Po přepnutí zkušebního stavu do módu vzdáleného ovládání z *dashboardu* může uživatel ovládat řídicí software zkušebního stavu. Je tedy možné nastavovat jednotlivé parametry řídicího softwaru. V případě užívání této funkce musí uživatel počítat s prodlevou propisování hodnot jednotlivých *tagů* do řídicího softwaru rychlotrhačky. Tato prodleva činí 3 sekundy.

Vzdálené řízení zkušebního stavu je stále v testovací fázi. Před uvolněním této funkce je nutné podrobit řídicí software dlouhodobému testu. Vzdálené ovládání zkušebního stavu vyžaduje naprostou spolehlivost datového přenosu a dostatečnou kompetenci obsluhy. Jen díky tomu lze zamezit případným poruchovým stavům.

5 Závěr

Teoretická část této práce přibližuje pojem IoT včetně jeho možné definice. Nastíněny jsou aspekty IoT hardwaru s typy jeho spojení a způsobem jeho identifikace. Probrány jsou také možné typy IoT sítí a jejich topologie, běžně užívané technologie pro přenos dat včetně používaných přenosových protokolů a jejich šifrování. V závěru teoretické části jsou uvedeny příklady realizace IoT.

Studie proveditelnosti obsahuje rozbor hardwaru a softwaru, pomocí kterého byla realizována praktická část práce. Je zde také uveden rozbor původního řídicího softwaru a typové příklady odesílání dat pomocí softwaru LabVIEW do serverové aplikace SystemLink. Závěrem studie proveditelnosti je navržen způsob tvorby IoT pro trhačku Zwick 1873.

V praktické části byl řídicí program hydraulické rychlotrhačky úspěšně doplněn o část určenou k extrakci dat. Extrahovaná data byla zapisována do *tagů*. Data, která nebylo třeba monitorovat periodicky, byla zapisována do *tagů* na základě jejich změny. Jejich monitoring byl tedy tzv. „*event-based*“. Údaje o poloze pístu a síle, která na něho působí, byly zapisovány do *tagů* periodicky s frekvencí 1 Hz.

Za účelem detailnějšího monitoringu zkušebního stavu byly do serverového úložiště odesílány soubory ve formátu TDMS. Tyto soubory obsahovaly data navzorkovaná s frekvencí 1 kHz. Data bylo možné vizualizovat pomocí prohlížeče souborů serverové aplikace SystemLink. Vizualizace souborů nebyla optimální. Prohlížeč serverové aplikace SystemLink neumožňoval kvalitní zobrazení jednotlivých bodů s takto vysokou vzorkovací frekvencí a zobrazené body nebyly vždy aproximovány správně. Prohlížeč souborů také nepodporoval časový formát kontroléru CompactRIO 9035. Časová osa tak nezobrazovala správné údaje. Analýzu dat v aplikaci SystemLink tak bylo vhodnější provádět na základě hodnoty jednotlivých bodů a ne podle vizuální podoby jejich vykreslení v grafu. Pro zkvalitnění vizualizace bylo vhodné soubory stáhnout a otevřít např. v aplikaci DIAdem.

Do původního softwaru hydraulické rychlotrhačky byla také implementována část programu umožňující vzdálené ovládání rychlotrhačky pomocí ovládacích prvků umístěných na *dashboardu* serverové aplikace SystemLink. Vzdálené řízení bylo nutné spustit pomocí odpovídající proměnné v softwaru trhačky. Funkce vzdáleného řízení trhačky je stále ve fázi dlouhodobého testování.

Úspěšně byla nakonfigurována IP kamera Edimax IC-3110W, pomocí které bylo možné monitorovat zkušební stav během zkoušky. Obraz kamery byl vizualizován na *dashboardu* serverové aplikace SystemLink v objektu typu *image*.

Odesílaná data byla zpracovávána pomocí aplikace SystemLink. V práci jsou detailněji probrány jednotlivé funkce této aplikace. Vzdálený přístup k *dashboardu* serverové aplikace SystemLink byl zajištěn skrze VPN. Tato VPN byla primárně zkonstruována pro připojení pomocí iPadu, ale je možné ji využít také pro připojení ostatních zařízení.

V závěru práce byla provedena verifikace IoT zkušebního stavu Zwick 1873.

Výsledkem praktické části práce je funkční IIoT. Byly splněny teoretické předpoklady pro označení vytvořené sítě jako internet věcí, které byly definovány

v teoretické části této práce. Použitá zařízení byla připojena k lokální síti, na které byla jasně identifikovatelná. Daná zařízení si byla vědoma svého kontextu v této síti a umožňovala sběr požadovaných dat a vzájemnou komunikaci. Možnou odchylkou vytvořené sítě od definice internetu věcí je fakt, že daná zařízení nejsou připojena k internetu. Z důvodu bezpečnosti je manipulace s daty prováděna na lokální síti. Data byla střádána, vyhodnocována a vizualizována pomocí serverové aplikace SystemLink. Přístup k této aplikaci je již umožněn i z internetu, a to pomocí VPN. Rozpor s teoretickou definicí internetu věcí tak lze považovat za odstraněný. Základní vyhodnocení dat mohla provádět i nezaškolená obsluha. Vizualizace rychlých dějů v prostředí aplikace SystemLink nebyla optimální, a tak jejich analýza vyžadovala obsluhu s vyšší odborností. Bylo nutné, aby obsluha chápala a rozuměla procesům, které jsou prováděny na pozadí programu. Potvrzen tak byl předpoklad růstu nároku na odbornost obsluhy, který automatizace zapříčiňuje.

Literatura

[1] SPURNÁ Ivona. *Příklady aplikací IoT v průmyslu a logistice*. SystemOnLine [online]. 2017. [cit. 2020-4-6]. Dostupné z: https://www.systemonline.cz/rizeni-vyroby/priklady-aplikaci-iot-v-prumyslu-a-logistice.htm?mobilelayout=false

[2] MCCLELLAND, Calum. *What Is IoT? – A Simple Explanation of the Internet of Things.* Iot for all [online]. 2020. [cit. 7.2.2020]. Dostupné z: https://www.iotforall.com/what-is-iot-simple-explanation/

[3] BUYYA, Rajkumar, DASTJERDI, Amir Vahid. *Internet of Things*. Principles and Paradigms. Burlington, Massachusetts: Morgan Kaufmann, 2016. ISBN 978-0-12-805395-9

[4] POHANKA, Pavel. *Internet věcí*. pavelpohanka [online]. 2017. [cit. 25.11.2019]. Dostupné z: http://www.pavelpohanka.cz/internet-of-things/

[5] VOJÁČEK, Antonín. *Základní úvod do oblasti internetu věcí (IoT)*. Automatizace.hw.cz [online]. 2016. [cit. 25.11.2019]. Dostupné z: https://automatizace.hw.cz/zakladni-uvod-do-oblasti-internetu-veci-iot.html

[6] SAKOVICH, Natalia. *Internet of Things (IoT) Protocols and Connectivity Options: An Overview*. sam solution [online]. 2018. [cit. 25.11.2019]. Dostupné z: https://www.sam-solutions.com/blog/internet-of-things-iot-protocols-and-connectivity-options-an-overview/

[7] Bluetooth. *The history of bluetooth SIG*. bluetooth.com [online]. 2020. [cit. 20.04.2020]. Dostupné z: https://www.bluetooth.com/about-us/our-history/

[8] THOMAS, Jessica. *The History of WiFi*. purple [online]. 2014. [cit. 20.04.2020]. Dostupné z: https://purple.ai/blogs/history-wifi/

[9] HARWOOD, Trevor. *IoT Standards and Protocols. Postscapes* [online]. 2020. [cit. 25.11.2019]. Dostupné z: https://www.postscapes.com/internet-of-things-protocols/#protocols

[10] Redakce IoT portál. Wi-Fi HaLow. *IoT portál brána do světa internetu věcí* [online]. 2016. [cit. 25.11.2019]. Dostupné z: https://www.iot-portal.cz/2016/02/29/wi-fi-halow/

[11] BRAINBRIDGE. *FROM 1G TO 5G: A BRIEF HISTORY OF THE EVOLUTION OF MOBILE STANDARDS*. Brainbridge [online]. 2020. [cit. 20.04.2020]. Dostupné z: https://www.brainbridge.be/news/from-1g-to-5g-a-brief-history-of-the-evolution-of-mobile-standards

[12] Redakce IoT portál. ZigBee. *IoT portál brána do světa internetu věcí* [online]. 2016. [cit. 25.11.2019]. Dostupné z: https://www.iot-portal.cz/2016/02/24/zigbee/

[13] GRIGORIK, Ilya. *Brief History of HTTP*. hpbn.co [online]. 2013. [cit. 6.4.2020]. Dostupné z: https://hpbn.co/brief-history-of-http/

[14] JANOVSKÝ, Dušan. *HTTP protokol*. Concurrency [online]. 2019. [cit. 24.2.2020]. Dostupné z: https://www.jakpsatweb.cz/server/http-protokol.html

[15] BHOLA, Siddharth. *Why HTTP is not suitable for IOT applications*. Concurrency [online]. 2019. [cit. 24.2.2020]. Dostupné z: https://www.concurrency.com/blog/june-2019/why-http-is-not-suitable-for-iot-applications

[16] MSV, Janakiram. *Get to Know MQTT: The Messaging Protocol for the Internet of Things*. Thenewstack.io [online]. 2016. [cit. 6.4.2020]. Dostupné z: https://thenewstack.io/mqtt-protocol-iot/

[17] MICHALEC, Libor. *MQTT: univerzální protokol nejen pro cloudové aplikace*. Automatizace.hw.cz [online]. 2019. [cit. 24.2.2020]. Dostupné z: https://automatizace.hw.cz/mqtt-univerzalni-protokol-nejen-pro-cloudove-aplikace.html

[18] KRAMER, Joshua. *Advanced Message Queuing Protocol (AMQP)*. Linuxjournal.com [online]. 2009. [cit. 6.4.2020]. Dostupné z: https://www.linuxjournal.com/article/10379

[19] Object Management Group. What is DDS?. DDS-FOUNDATION [online]. 2020. [cit. 20.4.2020]. Dostupné z: https://www.dds-foundation.org/what-is-dds-3/

[20] SSL-cerfifikaty.cz. *SSL protokol*. SSL-cerfifikaty.cz [online]. 2019. [cit. 3.2.2020]. Dostupné z: https://www.ssl-certifikaty.cz/o-certifikatech/ssl-protokol/

[21] Mgr. SALAŠOVÁ, Petra. *SSL & TLS – rozdíly, o kterých jste možná nevěděli!*. SSL market [online]. 2018. [cit. 3.2.2020]. Dostupné z: https://blog.sslmarket.cz/inpage/ssl-tls-rozdily-o-kterych-jste-mozna-nevedeli/

[22] MICHALEC, Libor. *Prevence je lepší než chemický postřik.* iot-portal.cz. [online]. 2018. [cit. 3.3.2020]. Dostupné z: https://www.iot-portal.cz/2018/01/03/prevence-je-lepsi-nez-chemicky-postrik/

[23] BAČKOROVÁ, Drahomíra, PTÁČEK, Honza. *Pardubice začaly montovat inteligentní semafory. První budou na rosickém mostě*. pardubice.rozhlas.cz [online]. 2019. [cit. 3.2.2020]. Dostupné z: https://pardubice.rozhlas.cz/pardubice-zacaly-montovat-inteligentni-semafory-prvni-budou-na-rosickem-moste-8092112

[24] MICHALEC, Libor. *30 let historie Labview*. vyvoj.hw.cz [online]. 2017. [cit. 25.11.2019]. Dostupné z: https://vyvoj.hw.cz/30-let-historie-labview.html

[25] National Instruments. *NI cRIO-9035*. ni.com [online]. 2018. [cit. 6.4.2020]. Dostupné z: http://www.ni.com/pdf/manuals/376935d_02.pdf

[26] National Instruments. *NI 9265 Datasheet*. ni.com [online]. 2015. [cit. 6.4.2020]. Dostupné z: https://www.ni.com/pdf/manuals/374067a_02.pdf

[27] National Instruments. *NI 9375*. ni.com [online]. 2020. [cit. 6.4.2020]. Dostupné z: https://www.ni.com/pdf/manuals/378026b_02.pdf

[28] National Instruments. *NI 9201*. ni.com [online]. 2016. [cit. 6.4.2020]. Dostupné z: https://www.ni.com/pdf/manuals/373783a_02.pdf

[29] Kistler. *Charge Amplifier Module Type 5171A…*. kistler.com [online]. 2016. [cit. 6.4.2020]. Dostupné z: https://www.kistler.com/?type=669&fid=107336&model=document

Obrázky

Obrázek 7 CGS. *cRIO Module Series*. www.cgs-gruppe.de [online]. 2020. [cit. 20.3.2020]. Dostupné z: https://www.cgs-gruppe.de/wp-content/uploads/2018/02/csm_k-cRIO-9035_CMYK_6640af8a85.jpg

Zbylé obrázky byly vytvořeny autorem této práce.