Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Pořízení a přenos dat z IoT sítí

Anna Navrátilová

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Anna Dřevíkovská

Informatika

Název práce

Pořízení a přenos dat z IoT sítí

Název anglicky

Data acquisition and data transmission from IoT networks

Cíle práce

Bakalářská práce je tematicky zaměřena na problematiku pořízení a přenosu dat z IoT sítí. Hlavním cílem práce je analýza a komparace možností sběru, přenosu, integrace a zpracování dat z IoT zařízení. Dílčí cíle práce jsou:
- charakteristika možností pořízení, přenosu, integrace a zpracování dat z IoT zařízení a
- analýza efektivity sběru a přenosu dat z LPWAN senzorové sítě vytvořené v rámci univerzitního kampusu.

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Teoretická část bude věnována analýze odborných článků pro vypracování přehledu technologií pro přenos a možnosti zpracování pomocí metod vědeckého popisu, explanace a komparace. Vlastní práce navazuje na teoretickou za účelem experimentu pro využití standardu pro přenos dat. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry bakalářské práce.

Doporučený rozsah práce

40 – 50 stran textu

Klíčová slova

IoT, data, pořízení dat, přenos dat, zpracování dat, integrace dat, senzor, sítě, analýza

---

Doporučené zdroje informací

AL-FUQAHA, A., GUIZANI, M., MOHAMMDI, M., ALEDHARI, M. a AYYASH, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. In IEEE Communications Surveys & Tutorials, vol. 17, no. 4, Fourthquarter 2015, doi: 10.1109/COMST.2015.2444095. [online]. Dostupné z: https://ieeexplore.ieee.org/document/7123563

ALUR, R. et al. Systems computing challenges in the Internet of Things. In arXiv:1604.02980, 2016. [online]. Dostupné z: http://arxiv.org/abs/1604.02980

JAYAVARDHANA, G., RAJKUMAR B., SLAVEN a M. MARIMUTHU, P. Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, Volume 29, Issue 7, 2013, ISSN 0167-739X. https://doi.org/10.1016/j.future.2013.01.010. [online]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0167739X13000241

MARSH-HUNN, D., TRILLES, A., GONZÁLEZ-PÉREZ, J., TORRES-SOSPEDRA a RAMOS, F. A Comparative Study in the Standardization of IoT Devices Using Geospatial Web Standards. In IEEE Sensors Journal, vol. 21, no. 4,15 Feb.15, 2021, doi: 10.1109/JSEN.2020.3031315. [online]. Dostupné z: https://ieeexplore-ieee-org.infozdroje.czu.cz/document/9224992

TRILLES, S., LUJÁN, A., BELMONTE, Ó., MONTOLIU, R., TORRES-SOSPEDRA, J. HUERTA, J. SEnviro: A Sensorized Platform Proposal Using Open Hardware and Open Standards. Sensors 2015, 15. https://doi.org/10.3390/s150305555. [online]. Dostupné z: https://www.mdpi.com/1424-8220/15/3/5555

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

doc. Ing. Pavel Šimek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 27. 10. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 09. 03. 2023

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Pořízení a přenos dat z IoT sítí" jsem vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autorka uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15. 3. 2023 _____

Poděkování

Ráda bych touto cestou poděkovala panu doc. Ing. Pavlovi Šimkovi, Ph.D. za jeho trpělivost, cenné rady, podporu a vedení práce. Dále bych chtěla poděkovat panu Ing. Vojtěchovi Novákovi, Ph.D. za ochotnou spolupráci, pomoc a poskytnutí dat využitých v práci. V neposlední řadě bych ráda poděkovala své rodině, zejména Vojtěchovi Navrátilovi, za dlouholetou trpělivost a podporu.

Pořízení a přenos dat z IoT sítí

Abstrakt

Práce je tematicky zaměřena na problematiku pořízení a přenosu dat z IoT sítí. Hlavním cílem práce je analýza a komparace možností sběru, přenosu, integrace a zpracování dat z IoT zařízení. Analýza se nejprve zaměřuje na zpracování přehledu existujících technologií, které je možné využít k pořízení a přenosu dat z IoT zařízení a následujících možností zpracování takto získaných dat. Dále je teoretická práce věnována komparaci konkrétních technologií z vypracovaného přehledu. Výsledná analýza a komparace následně hodnotí výhody a nevýhody popsaných technologií. Vlastní práce se věnuje analýze efektivity standardizace sběru a přenosu prostorových dat z LPWAN senzorových sítí vytvořených v rámci univerzitního kampusu. V rámci praktické části práce je následovně využita jedna z popsaných technologií, která nejvíce splňuje standardizaci, určenou pro sběr a přenos dat prostorových dat, na implementaci prototypu. Prototyp primárně implementuje integraci a zpracování dat v rámci ucelené IoT platformy. Získaná data jsou v prototypu zpracovávána tak, aby byly využity možnosti persistence dat, které byly charakterizovány v přehledu. Dále prototyp rozšiřuje technologii o implementaci, která řeší některé z jejích nevýhod a zkoumá přínos tohoto rozšíření. Závěrem práce je pak syntéza efektivity vybraných technologií pro pořízení, přenos, zpracování a integraci dat v rámci daného prototypu, a to včetně analýzy využitelnosti standardizace přenosu a integrace prostorových dat v IoT technologiích.

Klíčová slova: IoT, data, pořízení dat, přenos dat, zpracování dat, integrace dat, standardizace, senzor, sítě, analýza

Data acquisition and data transmission from IoT networks

Abstract

This thesis is thematically focused on problematics of data acquisition and data transmission from IoT networks. Main goal of the thesis is analysis and comparative study of possibilities for collection, transmission, integration, and processing of data from IoT devices. Analysis first focuses on drawing up a review of existing technologies which can be used for acquisition and transmission of data from IoT devices and further possibilities of processing thus obtained data. Next the theoretical part of the thesis is dedicated to comparation of specific technological solutions which were described in the above-mentioned review. The resulting analysis and comparative study list advantages and disadvantages of the described technologies. Own work pursues an analysis of effectiveness of standardization for collecting and transmitting geospatial data from LPWAN sensor networks created on the university campus premises. As part of the practical part of the thesis one of the described technologies, where the standardization will apply the most, for collecting and transmitting the geospatial data, will be used for implementation of a prototype. The prototype primarily implements integration and processing of the data as part of comprehensive IoT platform. Data collected are processed in the prototype in a way, so the described persistence technologies are utilized. Furthermore, the prototype, is extended with an implementation, which will tackle one of the discovered disadvantages of the chosen technology. Analysis will be performed to determine the scale of the expected improvement to the technology. Finally, the thesis performs a synthesis of effectiveness of chosen technologies for collection, transmission, integration, and processing of data on the premises of the implemented prototype, including analysis of the usability of standardization for transmission and integration of geospatial data in the IoT technologies.

Keywords: IoT, data, data acquisition, data transmission, data processing, data integration, standardization, sensor, networks, analysis

Obsah

1 Úvod

Internet Věcí (Internet of Things) za poslední dekádu zaznamenal prudký nárůst využití, přestože koncept technologie IoT byl poprvé navržen Kevinem Ashtonem již v roce 1999. [2][3]

V roce 2013 reportoval McKinsey Global Institute nárůst připojených zařízení od roku 2008 o 300 %. [1][4] Předpokládá se, že do roku 2025 vzroste množství připojených IoTzařízení na 21 miliard s celosvětovým ekonomickým dopadem až 1,6 bilionů amerických dolarů. [3][5] Jiné předpovědi dokonce počítají s možným celosvětovým ekonomickým zastoupením v rozmezí 3,9 až 11,1 bilionů amerických dolarů. [7][8]

Lze tedy předpokládat, že význam IoT technologií a počet zapojených zařízení bude nadále rapidně růst. [9] Díky tomuto růstu lze tedy považovat, za nutné, vytvoření nových protokolů a standardů, příkladem může být IPv6. [6]

Jeden z takto vzniklých standardů je taktéž LPWAN (Low power wide area network). Oproti klasicky využívanému HTTP nebo IP umožňuje operovat na velkých vzdálenostech s nízkými náklady a s využitím bezdrátově propojených zařízení napájených bateriemi. [9][10] Mimo jiné taktéž omezuje možnou velikost rámců vysílaných ze senzorů, což přispívá k šetření napájecích baterií. [10][11]

Nezbytnost vytváření standardů a protokolů jako je LPWAN bude důležité i pro aspekt udržitelnosti. Přestože již teď má IoT zastoupení v řešení problémů, které přispívají k dlouhodobé udržitelnosti, díky nimž dochází ke zlepšení řízení a snížení plýtvání energií a zdrojů, [13] lze předpokládat, že samotný nárůst zapojených zařízení bude přispívat k ekologické i ekonomické zátěži. [12] Téma udržitelnosti v IoT bude zajisté, do budoucna, hrát čím dál větší roli, jelikož je ale mimo rámec této bakalářské práce nebude se práce nadále věnovat její analýze.

I když růst IoT přináší nové technologie a standardy, pouze jejich malé množství je schopné mezi sebou spolupracovat. [3] Pro širší využití, snadnější přenos a integraci dat, pořízených z IoT zařízení, bude zřejmě nutné zavést obecné standardy, které by podporovaly společný datový model a rozhraní. [3][8][4][14]

Zatímco teoretická část této práce se věnuje charakteristice, přehledu a komparaci existujících oblíbených technologií pro sběr, přenos a integraci senzorových dat, tak praktická část se věnuje analýze efektivity využití otevřeného standardu

pro implementaci IoT Platformy za účelem provedení experimentu a analýzy výsledků z experimentu pro určení efektivnosti dané standardizace.

Pro analýzu je využito jedno z řešení SensorThings API v1.1 standardu [15][17] vyvíjeného spravující organizací Open Geospatial Consortium (OGC). [122] Samozřejmě řešení, které implementují tento standard existuje více. U jednotlivých existujících implementací je hodnoceno, zdali disponují OGC certifikací [17], jejich rozšířenost a současné využití v komunitě a dostupnost a otevřenost kódu.

IoT Platforma je nasazena za pomoci Docker kontejnerů. K persistenci dat je využito více často používaných technologií (RDBMS, databáze využívající časových řad). Komunikace mezi jednotlivými službami probíhá za pomoci konceptu Publisher- Subscriber.

Závěr práce se pak věnuje analýze provedeného experimentu za cílem syntézy, zdali daný standard splňuje své dané předpoklady jako je například nutnost interoperability v IoT systémech. V teoretických i praktických částech práce jsou pro analýzu upřednostňovány technologie, které již existují, a jsou využívány, v rámci projektů na univerzitním kampusu. Primárně pak v projektu datové platformy Life 4.0 Sciences [18].

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je analýza a komparace možností sběru, přenosu, integrace a zpracování dat z IoT zařízení. Dílčím cílem práce je pak charakteristika možností pořízení, přenosu, integrace a zpracování dat z IoT zařízení ve formě přehledu.

Dalším dílčím cílem práce je analýza efektivity standardizace sběru a přenosu prostorových dat z LPWAN senzorové sítě vytvořené v rámci univerzitního kampusu.

2.2 Metodika

Metodika řešené problematiky je založena na studiu a analýze odborných informačních zdrojů. Teoretická část je věnována analýze odborných článků pro vypracování přehledu technologií pro přenos a možnosti zpracování pomocí metod vědeckého popisu, explanace a komparace.

Vlastní práce navazuje na teoretickou za účelem experimentu pro využití standardu pro přenos dat. Na základě syntézy teoretických poznatků a výsledků praktické části jsou formulovány závěry bakalářské práce.

Analýza zkoumá více IoT zařízení využívajících standardy LPWAN sítí. Mezi dílčí cíle práce lze taktéž zahrnout implementaci ucelené IoT platformy na základě provedené analýzy a komparace různých technologií implementujících standardizaci pro přenos a integraci prostorových dat.

Za pomoci platformy je nadále prováděn experiment za účelem analýzy efektivity daného standartu a vybrané implementace splňující tento standard.

Posledním dílčím cílem práce je analýza nevýhod dané implementace. Na základě analýzy je proveden experiment, který se snaží o eliminaci jedné z vybraných nevýhod, kde na základě syntézy je rozhodnuto o přínosu daného rozšíření.

Jednotlivé technologie jsou nasazeny v Docker kontejnerech za účelem dalších analýz, monitorování, rozšiřitelnosti budoucích řešení a zobrazení dat koncovým uživatelům.

3 Teoretická východiska

3.1 Internet věcí

Termín internet věcí byl zaveden v roce 1999 Kevinem Ashtonem. [2][3] Existuje mnoho definic internetu věcí (anglicky Internet of Things). [8] Lze použít jednu z nejčastěji citovaných definicí, která sémanticky popisuje internet věcí jako celosvětově vzájemně propojenou síť objektů, které lze unikátně adresovat na základě užití standartních komunikačních protokolů. [19]

Růst technologické vlny internetu věcí byl umožněn technologickými pokroky, díky kterým mohou objekty interagovat v prostředí internetu bez nutného lidského zásahu. Byl transformován ze začínající technologie, na technologii, která figuruje v mnoha odvětvích a mění současný internet, propojením a integrováním miliardy nezávislých a inteligentních zařízení. [3]

Je zde i nutnost řešení nových problematik, které vznikají s velkou heterogenitou dat pořízených ze sensorů, IoT zařízení a protokolů. Mnoho studií předpokládá, že jeden z největších současných problémů internetu věcí je nízká interoperabilita a absence obecných standardů, které by mimo jiné přinesly lepší škálovatelnost a komunikaci mezi rozdílnými zařízeními. Dalším současným problémem aplikačních řešení internetu věcí je proprietární uzamčení koncových uživatelů a firem, které je následně činí ekonomicky náročné při potřebě změny, nebo přechodu na jiné aktuálnější technologie. [3][6][20]

IoT Elements		Samples
Identification	Naming	EPC, uCode
	Addressing	IPv4, IPv6
Sensing		Smart Sensors, Wearable sensing devices, Embedded sensors, Actuators, RFID tag
Communication		RFID, NFC, UWB, Bluetooth, BLE, IEEE 802.15.4, Z-Wave, WiFi, WiFiDirect, , LTE-A
Computation	Hardware	SmartThings, Arduino, Phidgets, Intel Galileo, Raspberry Pi, Gadgeteer, BeagleBone, Cubieboard, Smart Phones
	Software	OS (Contiki, TinyOS, LiteOS, Riot OS, Android); Cloud (Nimbits, Hadoop, etc.)
Service		Identity-related (shipping), Information Aggregation (smart grid), Collaborative-Aware (smart home), Ubiquitous (smart city)
Semantic		RDF, OWL, EXI

obrázek č. 1 – Základní bloky a technologie IoT[1]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza.t2-2444095-small.gif

3.1.1 Senzory

Základním stavebním kamenem internetu věcí jsou chytré objekty neboli senzory, kterým je možné přidělit unikátní identifikátor. Jsou schopné komunikovat se vzájemně propojenými zařízeními v bezdrátových internetových sítích. Tyto sítě jsou častěji nazývány jako bezdrátové senzorové sítě. [14]

Technologické pokroky v hardwarové výrobě umožnily výrobcům, i na základě Moorových zákonů, zmenšovat velikosti mikroprocesorů a zároveň zvyšovat jejich výpočetní schopnosti. S hardwarovými pokroky přichází i trend otevřených hardwarových schémat, které zvyšují jejich využívání a základnu vývojářů, které dané mikroprocesory a mikro počítače využívají. [14]

Díky vzájemně propojeným a dostupným zařízením lze pořizovat velké množství automatizovaných strojových dat, které je možné ukládat a analyzovat. [8] Podrobný přehled možností sběru dat z IoT zařízení bude popsán v kapitole č. 3.2.

3.1.2 Komunikační protokoly

Přestože se předpokládá, že jedna z největších překážek pro IoT je nízká interoperabilita a absence obecných standardů, tak existují komunikační protokoly, které byly často navrhnuty pro usnadnění a zjednodušení práce pro programátory a poskytovatele služeb spojených s IoT. [6]

Jako hlavní kategorie lze vyjmenovat Aplikační protokoly, Service Discovery protokoly a Infrastrukturní protokoly. Je vhodné mimo základní standardy a protokoly zmínit, že v rámci komunikace mezi jednotlivými zařízeními v senzorových sítích je nutné řešit i bezpečnost a interoperabilitu. Kvůli rozmanitosti popsaných technologií jsou obě disciplíny obtížné. Jak v současné době, tak i do budoucna jim zajisté bude věnována čím dál větší pozornost. [6]

Dle dokumentu vydaného v roce 2015 organizací Internet Architecture Board [21] lze rozdělit komunikaci mezi IoT zařízeními na čtyři základní komunikační vzory: Zařízení – Zařízení, Zařízení – Cloud, Zařízení – Brána, Cloud – Cloud. [8] Komunikační protokoly a možnosti přenosu dat jsou podrobněji rozebrány v kapitole č. 3.3.

<table>
<tr><td colspan="2">Application Protocol</td><td>DDS</td><td>CoAP</td><td>AMQP</td><td>MQTT</td><td>MQTT-SN</td><td>XMPP</td><td>HTTP REST</td></tr>
<tr><td colspan="2">Service Discovery</td><td colspan="4">mDNS</td><td colspan="3">DNS-SD</td></tr>
<tr><td rowspan="4">Infrastructure Protocols</td><td>Routing Protocol</td><td colspan="7">RPL</td></tr>
<tr><td>Network Layer</td><td colspan="5">6LoWPAN</td><td colspan="2">IPv4/IPv6</td></tr>
<tr><td>Link Layer</td><td colspan="7">IEEE 802.15.4</td></tr>
<tr><td>Physical/ Device Layer</td><td colspan="2">LTE-A</td><td colspan="2">EPCglobal</td><td colspan="2">IEEE 802.15.4</td><td>Z-Wave</td></tr>
<tr><td colspan="2">Influential Protocols</td><td colspan="5">IEEE 1888.3, IPSec</td><td colspan="2">IEEE 1905.1</td></tr>
</table>

obrázek č. 2 – Přehled současných standardů v IoT[2]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza.t3-2444095-large.gif

3.1.3 Senzorové sítě

„Senzorová síť je infrastruktura, která zahrnuje pozorování či měření, zpracování a komunikaci, jež dává administrátorovi možnosti řízení, pozorování a reagování na události a jevy ve specifickém prostředí“. [8]

Existuje řada využití senzorových sítí. Jako příklady lze uvést sběr dat, měření, monitorování nebo lékařskou telemetrii. Senzorové sítě se skládají z lokálně propojených uzlů, které obstarávají sběr dat a jejich přenos do dalších uzlů. Dalším uzlem může být propojující nebo konečný uzel. Koncový uzel funguje jako centrální sběrný bod, který je taktéž obecně nazývaný jako komunikační brána (anglicky Gateway). Komunikační brána implementuje ověřování správnosti dat, dotazování, řízení událostí a data mining. [8][14][22][23]

Nejčastěji používanými topologiemi v senzorových sítích jsou hvězdicová, stromová a mesh. Taktéž jsou zmiňovány hybridní topologie, které vzniknou propojením dvou anebo více rozdílnými topologiemi do jedné. [14][23]

3.2 Sběr dat z IoT zařízení

Jak již bylo zmíněno, jeden z důvodů růstu IoT technologií je i růst popularity otevřeného hardwaru a snížení jeho ceny. Proto vzniká velké množství a variant mikrokontrolerů a minipočítačů neboli jednodeskových počítačů. [14] Právě mikrokontrolery a minipočítače s integrovanými senzory, vestavěnými bezpečnostními funkcionalitami a TCP/IP, jak znázorňuje diagram v obrázku č. 3, jsou v současné době nejvíce využívanými zařízeními pro sběr dat v rámci internetu věcí. [6]

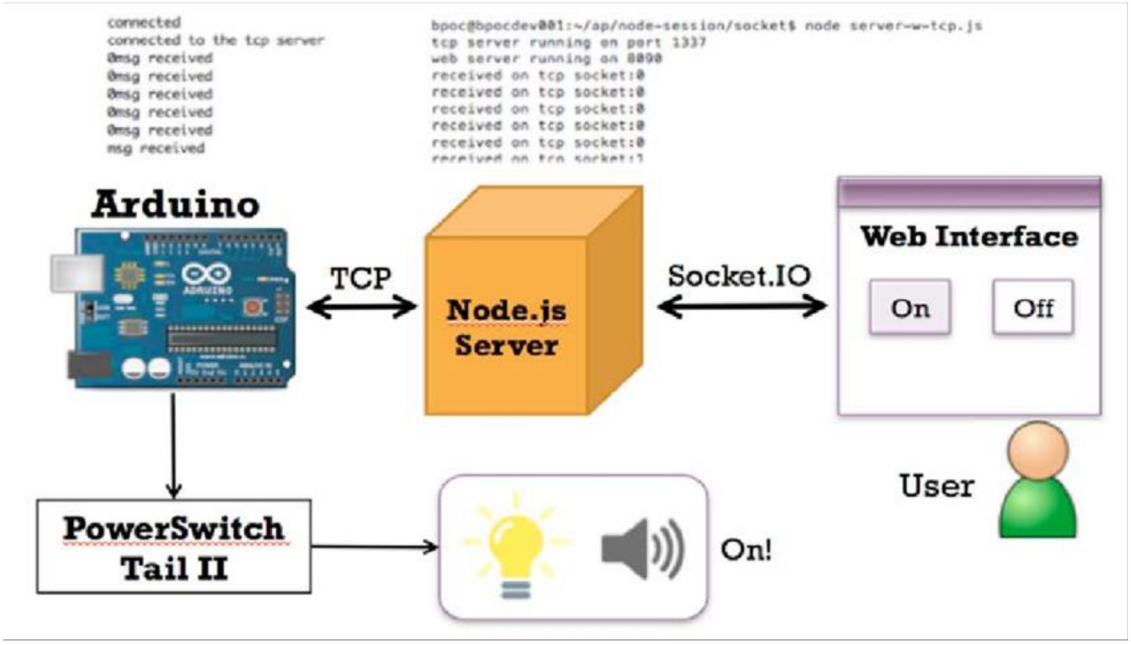
3.2.1 Mikrokontroler

„Mikrokontroler je kompaktní integrovaný obvod navržený k obsluze specifických operací ve vestavěném systému. Typický mikrokontroler zahrnuje procesor, paměť a vstupy a výstupy v jediném čipu. [24] *Mikrokontroler je vestavěný uvnitř systému tak aby obsluhoval samostatnou funkci v zařízení.“* [25]

3.2.2 Minipočítač

„Minipočítač neboli jednodeskový počítač, v angličtině single-board computer (SBC) je kompletní, funkční počítač, ve kterém jsou jeho mikroprocesor, vstupy a výstupy,

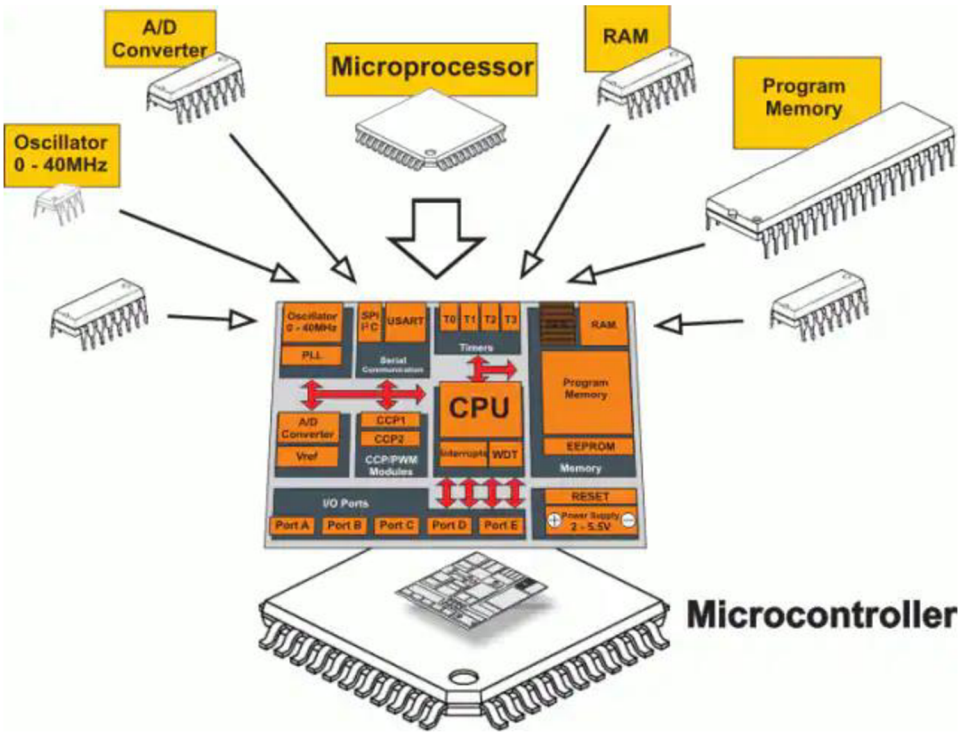
paměť a další funkce vestavěné do jediné obvodové desky s vestavěnou operační pamětí, která má předem pevně stanovenou velikost a není rozšiřitelná o žádné další periferie.“ [26]



obrázek č. 3 – Diagram popisující příklad obecné architektury hardwaru a softwaru využívající mikrokontroler Arduino Uno[3]

Zdroj: https://www.researchgate.net/figure/Diagram-outlining-the-general-architecture-of-the-hardware-and-software-for-the-Arduino_fig1_282270640

Hardwarovou architekturu mikrokontrolerů lze znázornit diagramem na obrázku č. 4.



obrázek č. 4 – Hardwarové části mikrokontroleru[4]

Zdroj: https://static4.arrow.com/-/media/arrow/images/miscellaneous/0/0218-mcu_image.gif

3.2.3 Identifikace zařízení v senzorové síti

Identifikace zařízení je stěžejní částí IoT. Bez identifikace zařízení není možné, aby ho mohly využívat odpovídající služby spojené s IoT. Existuje mnoho metod pro identifikaci IoT zařízení, jako nejznámější lze uvést elektronické produktové kódy (EPC) [27], všudypřítomné kódy – uCode [28] a identifikaci pomocí rádiových vln – RFID [29]. [6]

Je klíčové rozlišovat mezi identifikací a adresováním IoT zařízení. Identifikace zařízení odpovídá pojmenování zařízení v síti, zatímco jeho adresace odpovídá adrese, která je v rámci senzorových sítí, využívaná ke komunikaci mezi jednotlivými zařízeními. [6]

Rozlišování identifikátorů a adresací je nezbytné i z toho důvodu, že metody, které jsou využívané pro identifikaci zařízení, nevytváří globálně unikátní identifikátory. Adresace je tedy primárně určená k unikátní identifikaci zařízení. [6]

Zatímco identifikační metody jsou využívané pro srozumitelnější identifikaci zařízení v konkrétní senzorové síti, adresace může být využívána i pro veřejné IP adresy dostupné v internetu. Tedy i mimo senzorové sítě, které mají primárně lokální charakter. [6]

3.2.4 Typy pořízených dat z IoT zařízení

Existuje více způsobů, jak rozlišovat mezi typy pořízených dat z IoT zařízení. Mezi základními typy dat patří:

- Data o současném stavu zařízení
- Data pořízená z měřičů např. tepla, vodoměrů a průtokoměrů
- Data související se životním prostředím

Data, která popisují současný stav IoT zařízení se využívají k tzv. prediktivní údržbě, která napomáhá k prodloužení životnosti zařízení či předcházení poruch. Vzdálené odběry dat, naměřené pomocí měřičů tepla, vodoměrů atd. automatizují sběr dat a umožňují i větší přehled o měření tohoto typu dat např. ve společných prostorách bytových domů. Data pořízená v souvislosti s životním prostředím napomáhají např. ke zlepšení kvality vzduchu, detekci venkovních teplot a vlhkosti vzduchu a výskytu a množství pohybu v určitých oblastech. [30][8]

Dále lze typy pořízených dat z IoT zařízení rozdělovat podle typů senzorů, které daná data detekují:

- Akcelometry
- Senzory blízkých objektů
- Pasivní infračervené senzory pohybu
- Senzory detekující míru zaplnění prostor
- Senzory detekující kvalitu vody
- Převodníky

Akcelometry snímají změny v gravitační akceleraci a jsou často používány na detekci vibrací, náklonů a akcelerace. Senzory blízkých objektů jsou například využívány jako parkovací senzory. Pasivní infračervené senzory (PIR) detekují pohyb za pomoci senzoru snímajícího infračervené světlo. Ten je schopný určit presenci nebo absenci zvířat nebo lidí a jejich pohyb v daném prostoru. Senzory, které detekují míru zaplnění prostoru například snímají výšku hladiny vody. Převodníky měří parametry jako teplotu, vlhkost vzduchu, tlak, směr větru, rychlost, intenzitu světla a zvuku, koncentraci chemikálií, míru znečištění atd. [32]

3.2.5 Formáty pořízených dat z IoT zařízení

Podle předpovědí lze předpokládat, že množství vygenerovaných dat do roku 2025 dosáhne až 80 zeta bytů. [31] „*IoT data jsou výstupem zařízení nebo procesů asociovaných s takovými využitími, která produkují fyzikální veličiny spojené se životním prostředím. Senzory v prostředí IoT jsou využívány pro detekci událostí a k získání detekovaných hodnot.*“ [32]

IoT zařízení produkují velké množství heterogenních dat, která mohou mít strukturovanou, semistrukturovanou a nestrukturovanou podobu. Obecně v datech produkovaných počítačem můžeme nalézt například záznamy logů, bankovních transakcí atd. Nestrukturovaná data nemají žádné logické schéma ani předdefinovaný model nebo reprezentaci. Jejich příkladem jsou texty, mluvené slovo, obrázky a videa. Semi strukturovaná data jsou mixem strukturovaných a nestrukturovaných dat. Dobrým příkladem mohou být e-mailové zprávy, které mají předdefinovaná pole, ale jejich obsah a přílohy jsou nestrukturované. [32]

Mezi hlavní formáty dat pořízených z IoT zařízení patří:

- Text
- Binární data

- XML
- CSV
- JSON
- RFID

Data pořízená v IoT zařízeních v sobě obecně zahrnují informace o stavu zařízení, metadata o zařízení a detekované hodnoty. Takto generovaná data nemají jednotný formát. Je tedy složité vytvořit jednu společnou reprezentaci nebo model pro detekovaná data. [31]

V současné době je nejvíce využívaným datovým formátem XML. Příkladem vhodného datového formátu pro IoT je formát, který snadno reprezentuje daný zdroj dat zařízení, má nízké nároky na elektrickou spotřebu a je schopný interoperability.

XML bohužel ve své komunikaci přidává k detekovaným datům velkou hlavičku, což nadbytečně zvyšuje velikost přenášených dat. Dalším slibným formátem je JSON, který ale taktéž přenáší příliš velká data kvůli neefektivnímu kódování velkého množství tzv. bílých znaků (anglicky white-spaces) do ASCII. Lze tedy předpokládat, že do budoucna bude nutné zavést nový datový formát, který bude disponovat výše popsanými vlastnostmi. [32]

3.2.6 Nejpoužívanější zařízení pro sběr IoT dat

Následující přehled porovnává nejčastěji zmiňovaná zařízení s využitím v IoT sítích. Přehled byl vytvořen na základě zdrojů několika internetových článků:

Název	Mikroprocesor	Architektura	Taktování	RAM	Paměť	Ethernet / BLE / Wi-Fi (Cell)	Programovací jazyky
Raspberry Pi 4 Model B	Broadcom BCM2711	64-bit	1,5 GHz	1/4/8 GB	microSD	Ano / Ano / Ano	Python, C, Basic
BeagleBone Black	AM3358	32-bit	1 GHz	512 MB	4 GB + SD	Ano / Ne / Ne	C, C++, Python, Java, Perl, Ruby
Particle Boron	Nordic nRF52840 SoC	32-bit	64 MHz	256 KB	1 MB	Ne / Ano / Cell	C++
Particle Argon	Nordic nRF52840 SoC	32-bit	64 MHz	256 KB	1 MB	Ne / Ano / Ano	C++
Giant Board	Microchip SAMA5D2	32-bit	500 MHz	128 MB	microSD	Ne / Ne / Ne	CircuitPython
Arduino Uno R3	Microchip ATmega328P	8-bit	16 MHz	2 KB	32 KB Flash + 1 KB EEPROM	Ne / Ne / Ne	C++
NVIDIA Jetson Nano	ARM A57	64-bit	1,43 GHz	4 GB	microSD	Ano / Ne / Ne	Python, C++
Banana Pi BPI-M2 Pro	Amlogic S905X3	64-bit	1,5 GHz	2 GB	16 GB EEPROM	Ano / Ne / Ne	Java, C++
Tessel 2	Atmel SAMD21	32-bit	48 MHz	64 MB	32 MB	Ano / Ne / Ano	JavaScript
Arduino Nano 33 IoT	Atmel SAMD21	32-bit	48 MHz	2 MB	256 KB	Ne / Ano / Ano	C++

tabulka č. 1 – Přehled porovnávající nejčastěji používaná IoT zařízení[1]

Zdroje: [33][34][35][36][37][38][39]

3.3 Přenos dat z IoT zařízení

Stávající používané technologie pro přenos dat a komunikaci v IoT lze popsat za pomoci přehledu současně využívaných standardů.

3.3.1 Aplikační protokoly

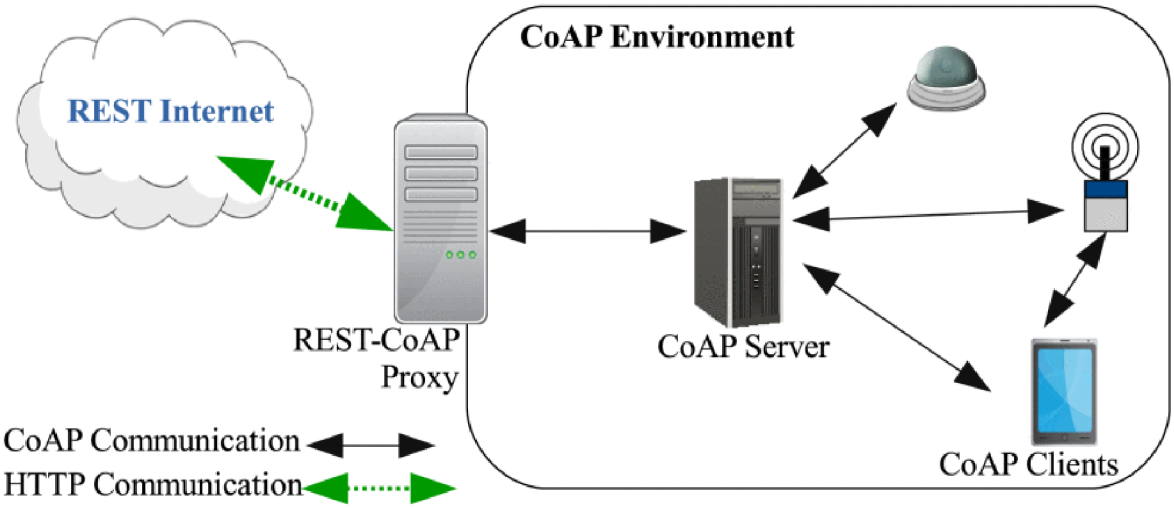
Mezi nejčastěji zmiňované aplikační protokoly lze zahrnout:

- Constrained Application Protocol (COAP)
- Message Queue Telemetry Transport (MQTT)
- Extensible Messaging and Presence Protocol (XMPP)
- Advanced Message Queuing Protocol (AMQP)
- Data Distribution Service (DDS)

CoAP je založen na REST architektuře společně s využitím funkcionalit HTTP. Architektura REST (REpresentational State Transfer) zjednodušuje klient-server komunikaci za pomoci vyrovnávání paměti a využívání HTTP pravidel. Je stateless, to znamená, že každý jednotlivý dotaz na server je kompletně izolovaný od ostatních dotazů a obsahuje všechny potřebné informace vyžadované od serveru. Nikdy se nespoléhá na výsledky z předchozích dotazů na server. [6][40][41]

REST architektura vyžaduje dodržování základních pravidel. Jedno z pravidel stanovuje, že každý zdroj, nebo koncový bod má jednotný unikátní identifikátor. Dále stanovuje, že pro označení zdrojů se výhradně využívají pouze podstatná jména a pro HTTP metody (GET, POST, PUT, DELETE) pouze slovesa. REST na rozdíl od SOAP (Simple Object Access Protocol) nevyžaduje, aby byl formát posílaných dat mezi klientem a serverem pouze ve formátu XML. CoAP se od RESTu liší tím, že místo TCP využívá UDP, tedy nespolehlivou komunikaci. [6][41][42]

Dále upravuje HTTP funkcionality, tak, aby splnily IoT požadavky na nízkou energetickou náročnost. Protože je CoAP navržený na základě REST architektury je komunikace mezi klasickým RESTem a CoAP aplikačním rozhraním snadno implementovatelná za pomoci tzv. proxy služby viz. obrázek č. 5. [6]



obrázek č. 5 – Přehled funkcionalit CoAP[5]

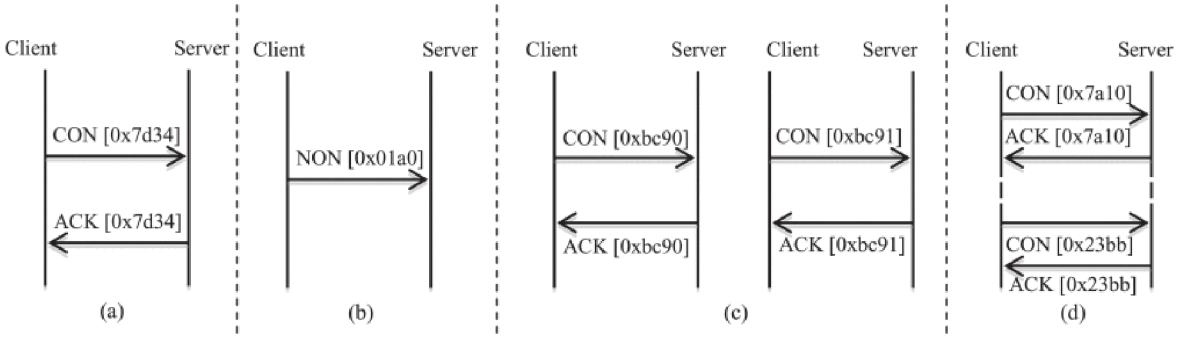
Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza5-2444095-large.gif

CoAP rozdělujeme do dvou vrstev:

- Vrstva určená pro zasílání zpráv
- Vrstva obstarávající dotazy a odpovědi

Vrstva pro zasílání zpráv řeší duplikace ve zprávách a za pomoci exponenciálního ustupování (backoff) chybovost, kterou běžně UDP neřeší. Vrstva obstarávající dotazy a odpovědi implementuje klasické REST dotazování a odpovědi. [6]

CoAP obsahuje čtyři typy zasílaných zpráv: ověřitelné, neověřitelné, obnovující (reset) a potvrzující. Na obrázku č. 6 lze vidět scénáře různých typů zpráv a jejich interakcí se serverem. [6]

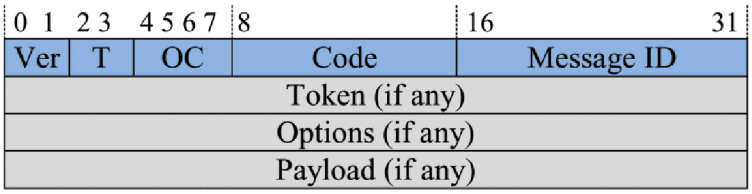


obrázek č. 6 – Typy CoAP zpráv[6]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza6abcd-2444095-large.gif

Jedna z nejdůležitějších funkcionalit CoAP je možnost sledování zdrojů, které na vyžádání monitoruje zdroje pomocí tzv. Publisher-Subscriber konceptu. Další podstatnou funkcionalitou je zasílání dat mezi klientem a serverem ve formě bloků, není tak nutné provádět aktualizace dat jako jejich celku, ale pouze po jejich částech. Je schopný využít odkazů pro snadnou orientaci v dostupných zdrojích.

Díky využití architektury REST a HTTP je CoAP velmi flexibilní v komunikaci s velkým množstvím různých IoT zařízení a umožňuje interagovat s klientskými aplikacemi za pomoci proxy služby. V neposlední řadě je výhoda CoAP i v jeho bezpečnosti, jelikož je postavený nad DTLS (Datagram Transport Layer Security) protokolem, který zajišťuje integritu a soukromí posílaných zpráv. [6]

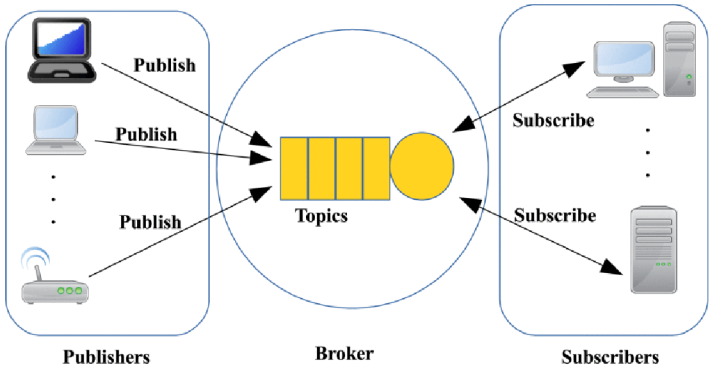


obrázek č. 7 – Formát CoAP zpráv[7]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza7-2444095-large.gif

MQTT je protokol založený na posílání zpráv. Jeho primárním cílem je propojení vestavěných zařízení a sítí s aplikacemi a jejich middlewarem, více o middlewaru v kapitole č. 3.4. Využívá více směrovacích mechanismů, 1:1, 1:N, a N:M, díky kterým je MQTT jedním z nejvhodnějších kandidátů pro přenos dat v IoT sítích. [6]

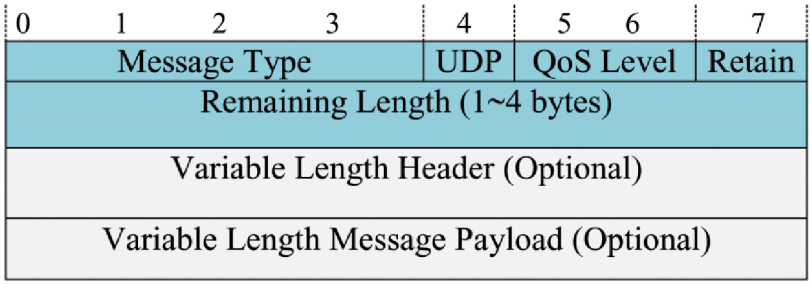
Podobně jako již zmiňovaný CoAP je založený na principu Publisher-Subscriber. Je taktéž vhodný pro zařízení, která vyžadují nízkou spotřebu energetických zdrojů. Oproti CoAP využívá MQTT ke komunikaci TCP a využívá tří úrovní QoS (Quality of Service). [6]



obrázek č. 8 – Architektura MQTT protokolu[8]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza8-2444095-large.gif

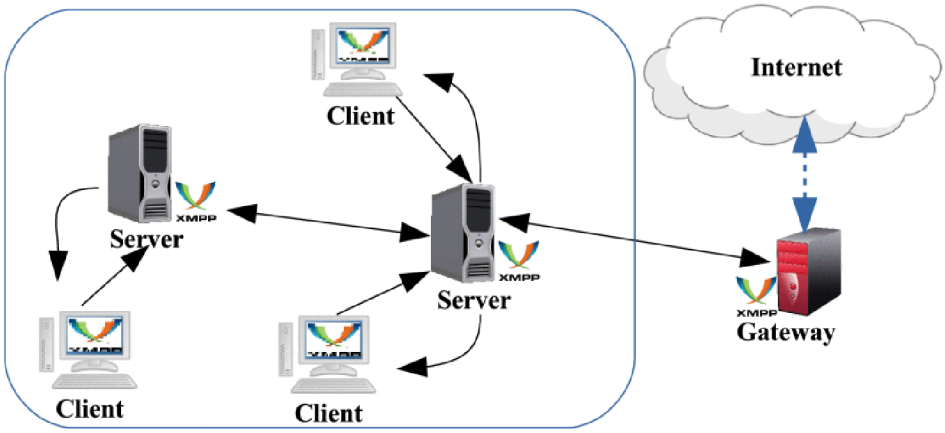
MQTT je sestaveno ze tří hlavních komponent. Odběratel (Subscriber), vydavatel (Publisher) a zprostředkovatel (Broker). Odběratel se přihlašuje k tzv. tématům (topics), ze kterých ho zajímají data. Data z témat jsou předávána od vydavatele ke zprostředkovateli a následně až k odběrateli. Vydavatel oproti odběrateli naopak vysílá potenciálně důležitá data ke zprostředkovateli. Tomu, jak funguje koncept Publisher-Subscriber, a jak se dá využít pro integraci dat z IoT zařízení se podrobněji věnuje kapitola č. 3.4. [6] Formát posílaných zpráv mezi jednotlivými zařízeními, v rámci MQTT, je vidět na obrázku č. 9.



obrázek č. 9 – Formát MQTT zpráv[9]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza10-2444095-large.gif

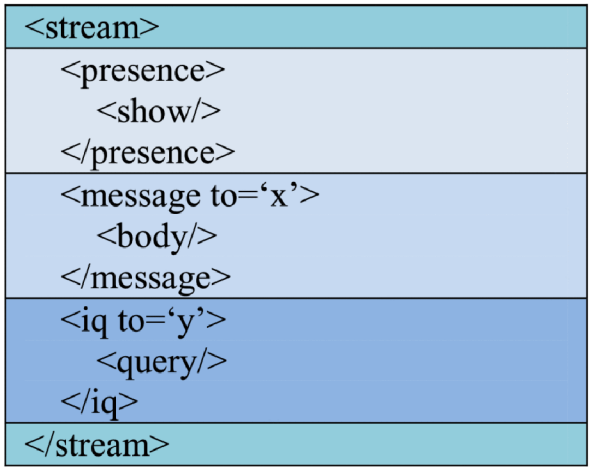
XMPP protokol využívá technologie tzv. instant messagingu (IM, v českém jazyce okamžité zasílání zpráv). Je primárně určený pro posílání textových, obrazových a zvukových zpráv mezi více stranami. XMPP byl původně vytvořen komunitou Jabber. Je nezávislý na operačním systému a umožňuje komunikaci více uživatelů a téměř okamžité posílání zpráv přes síť Internet. Podporuje autentizaci, spravování přístupových seznamů, spravování soukromí, šifrování zpráv a kompatibilitu s ostatními protokoly. Na obrázku č. 10 lze vidět funkcionalitu a architekturu protokolu XMPP. [6]



obrázek č. 10 – Architektura protokolu XMPP[10]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza11-2444095-large.gif

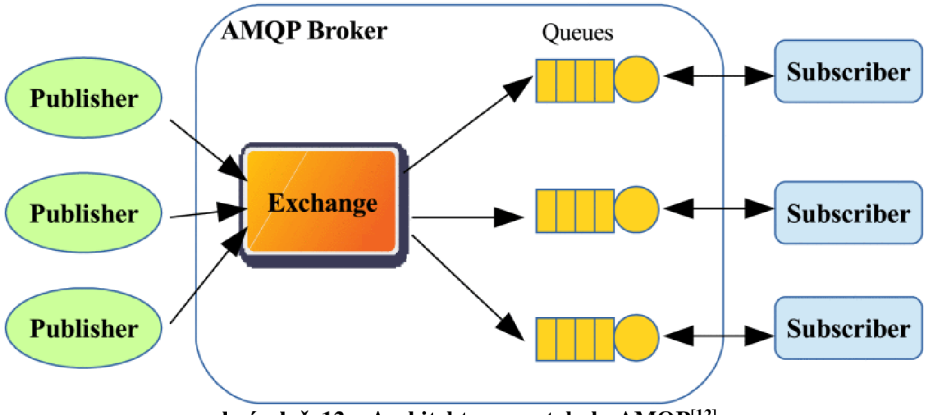
Díky jeho funkcionalitám je XMPP dalším vhodným kandidátem pro přenos dat v IoT sítích. Je využíván v mnoha IoT Platformách za účelem decentralizace. XMPP používá tzv. XML sloky jako formát pro zasílání jednotlivých zpráv mezi klientem a serverem. Na obrázku č. 11 je vidět formát jednotlivých zpráv. Nevýhody XMPP spočívají právě ve využívání XML slok jako formátu pro posílání zpráv. XML má poměrně vysoké nároky na síťovou režii, tento problém lze dále optimalizovat například za pomoci komprese proudových XML dat (EXI). [6]



obrázek č. 11 – Formát XMPP zpráv[11]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza12-2444095-large.gif

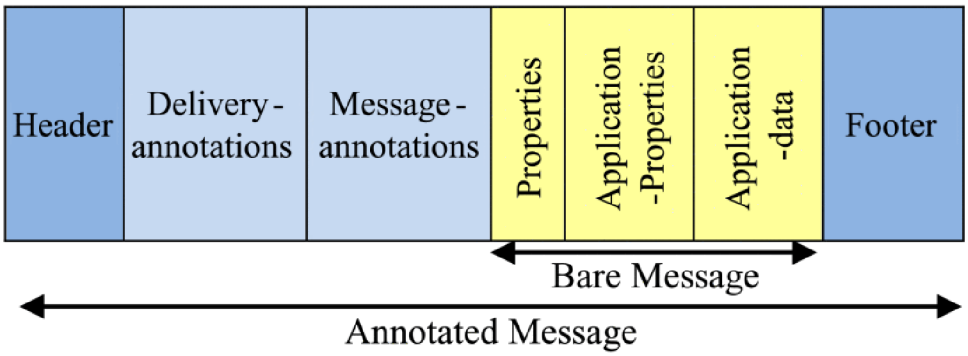
AMQP je otevřený aplikační protokol specificky implementovaný pro využití v IoT a jiná prostředí zaměřená na přijímání a odesílání zpráv. Podobně jako předchozí protokoly je znovu stavěn na Publisher-Subscriber konceptu. Je schopný spolehlivě předávat zprávy za pomoci vlastní implementace QoS, díky kterému umožňuje určit minimální množství pokusů o doručení zprávy. Pro komunikaci používá TCP. [6]



obrázek č. 12 – Architektura protokolu AMQP[12]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza13-2444095-large.gif

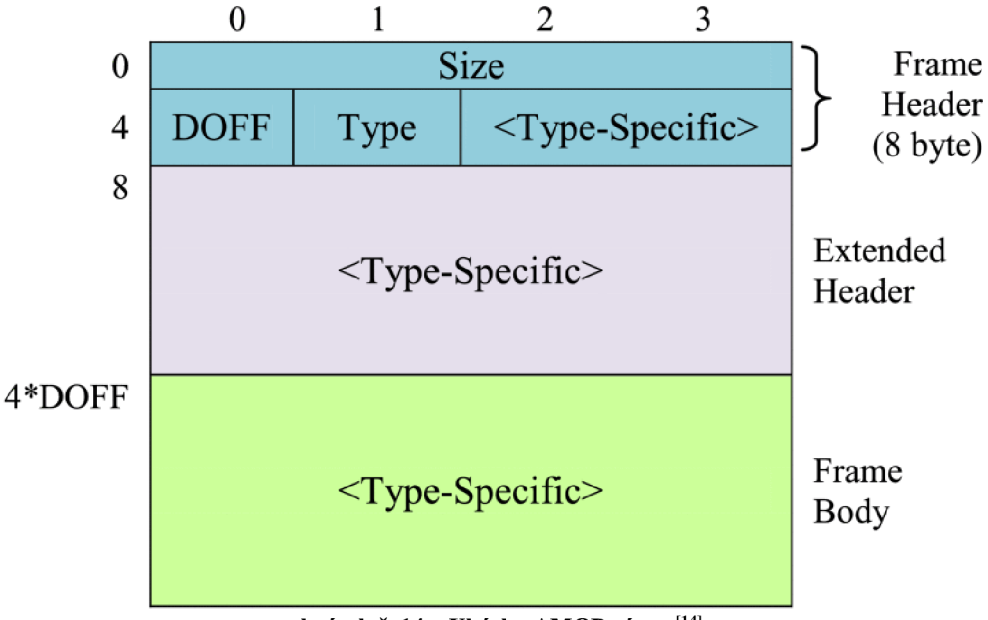
AMQP si definuje vlastní vrstvu pro zasílání zpráv nad transportní vrstvou TCP/IP modelu. Tím získává velkou interoperabilitu mezi jednotlivými aplikačními řešeními. Jsou definovány dva typy zpráv: základní zprávy a anotované zprávy. Základní zprávy jsou dodávané odesílatelem, zatímco anotované zprávy jsou určené pro adresáta. AMQP protokol využívá rámců, obrázek č. 13 ukazuje formát jednotlivých zpráv z obecného pohledu. [6]



obrázek č. 13 – Formát AMQP zpráv[13]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza14-2444095-large.gif

Zatímco na obrázku č. 14 lze vidět jednotlivé položky AMQP rámce.

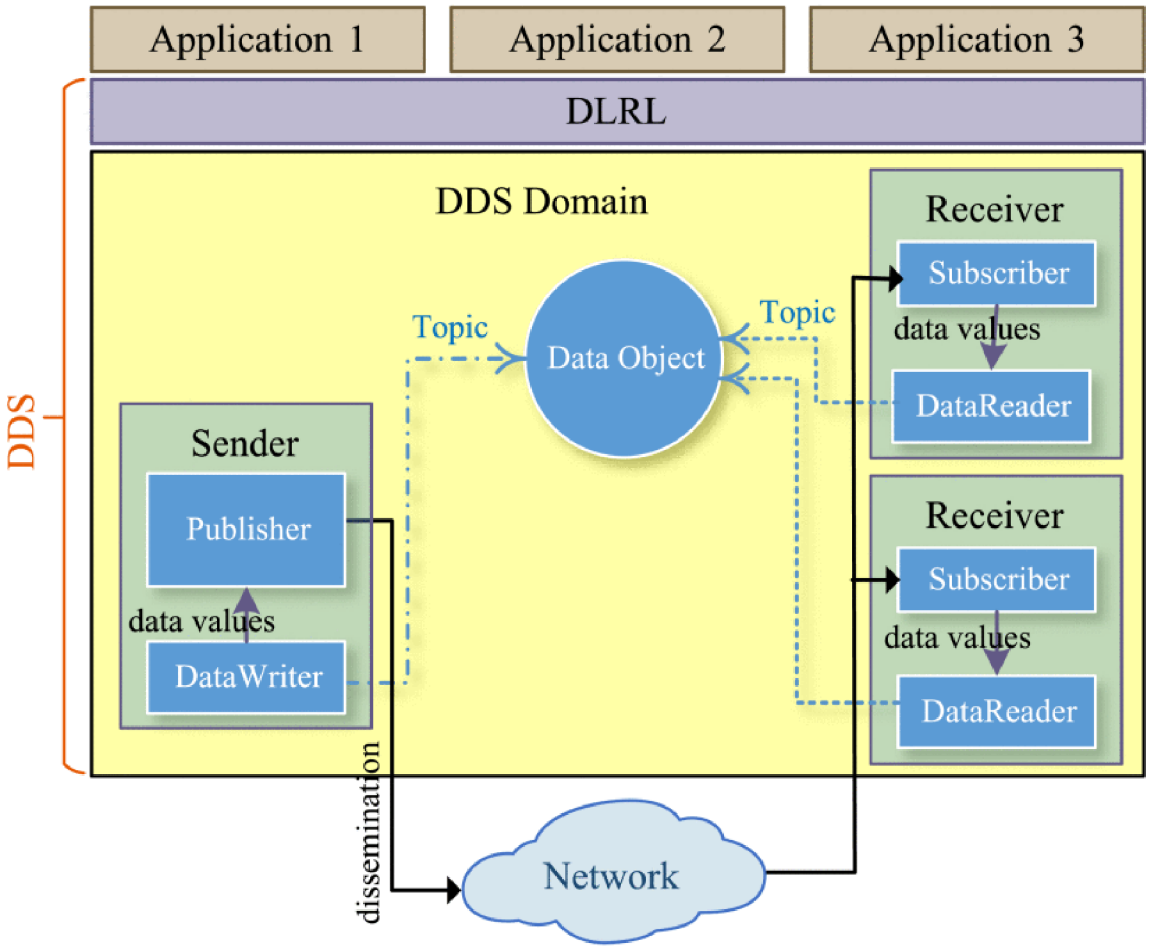


obrázek č. 14 – Ukázka AMQP rámce[14]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza15-2444095-large.gif

DDS protokol taktéž využívá konceptu Publisher-Subscriber. Vyvinula ho společnost Object Management Group (OMG). [43] Na rozdíl od již zmíněných protokolů nevyužívá ve své architektuře tzv. zprostředkovatele (broker). Namísto zprostředkovatele využívá multicastingu pro rozesílání zpráv a díky této vlastnosti disponuje vysokou úrovní QoS (Quality of Service), podporuje až 23 úrovní QoS. Mezi funkcionality DDS taktéž patří bezpečnost, rychlost, prioritizace, odolnost a spolehlivost. Všechny vlastnosti lze libovolně kombinovat a využívat. Programátoři se tedy mohou rozhodnout, zdali je v jejich řešení nutné využít všechny dostupné funkcionality, nebo jen některé z nich. [6]

DDS architektura obsahuje dvě vrstvy: Data-Centric Publish-Subscribe (DCPS) a Data-Local Reconstruction Layer (DLRL). DPCS vrstva obstarává poslání zpráv k odběratelům. DLRL vrstva je nepovinnou součástí protokolu a abstrahuje DPCS vrstvu skrz implementační rozhraní. Specifika architektury DDS protokolu lze vidět na obrázku č. 15. [6]



obrázek č. 15 – Architektura protokolu DDS[15]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza16-2444095-large.gif

Co se týče porovnání jednotlivých protokolů mezi sebou bylo mimo jiné zjištěno, že MQTT protokol je schopný dodávat zprávy, ze zařízení, s menším zpožděním, než jak je to u CoAP protokolu, pokud je nízká ztrátovost paketů v dané síti. Naopak pokud je ztrátovost paketů v síti vysoká CoAP má lepší výsledky než MQTT. Pokud je ztrátovost paketů pod 25 procenty a zprávy mají malou velikost, produkuje protokol CoAP méně nadbytečného provozu v síti. Další studie porovnala tyto dva protokoly v prostředí implementující mobilní aplikace. Zde se ukázalo, že u CoAP je využití šířky pásma menší a cestování paketů mezi zařízeními trvá méně času než u MQTT. [6]

U porovnání CoAP a HTTP je CoAP energeticky úspornější a server odpovídá na dotazy rychleji. Díky zmenšené hlavičce dotazu a malé velikosti paketů je CoAP energeticky i časově úspornější, než je HTTP. [6]

Studie taktéž porovnává XMPP a HTTP protokoly pro využití potřeby téměř okamžitých zpráv, kde výhodněji se pro potřeby okamžitých zpráv jeví XMPP. Pokud mají předávané zprávy velký objem má AMQP lepší výsledky než klasická REST architektura. Měření ve studiích bylo prováděno na základě průměrného množství zpráv vyměněných mezi klientem a serverem ve specifickém časovém intervalu. [6]

Další studie ukazuje, že DDS implementace prokazuje potenciál, pokud je u ní možné škálovat množství uzlů v senzorové síti. Obrázek č. 16 ukazuje komparaci jednotlivých aplikačních protokolů a jejich dostupných funkcionalit. [6]

Application Protocol	RESTful	Transport	Publish/ Subscribe	Request/ Response	Security	QoS	Header Size (Byte)
COAP	✓	UDP	✓	✓	DTLS	✓	4
MQTT	×	TCP	✓	×	SSL	✓	2
MQTT-SN	×	TCP	✓	×	SSL	✓	2
XMPP	×	TCP	✓	✓	SSL	×	-
AMQP	×	TCP	✓	×	SSL	✓	8
DDS	×	TCP UDP	✓	×	SSL DTLS	✓	-
HTTP	✓	TCP	×	✓	SSL	×	-

obrázek č. 16 – Komparace aplikačních protokolů a jejich funkcionalit[16]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza.t4-2444095-large.gif

3.3.2 Service Discovery protokoly

Mezi nejčastěji používané protokoly pro řízení zdrojů, registraci a objevení zdrojů a služeb patří:

- Multicast DNS (mDNS)
- DNS Service Discovery (DNS-DS)

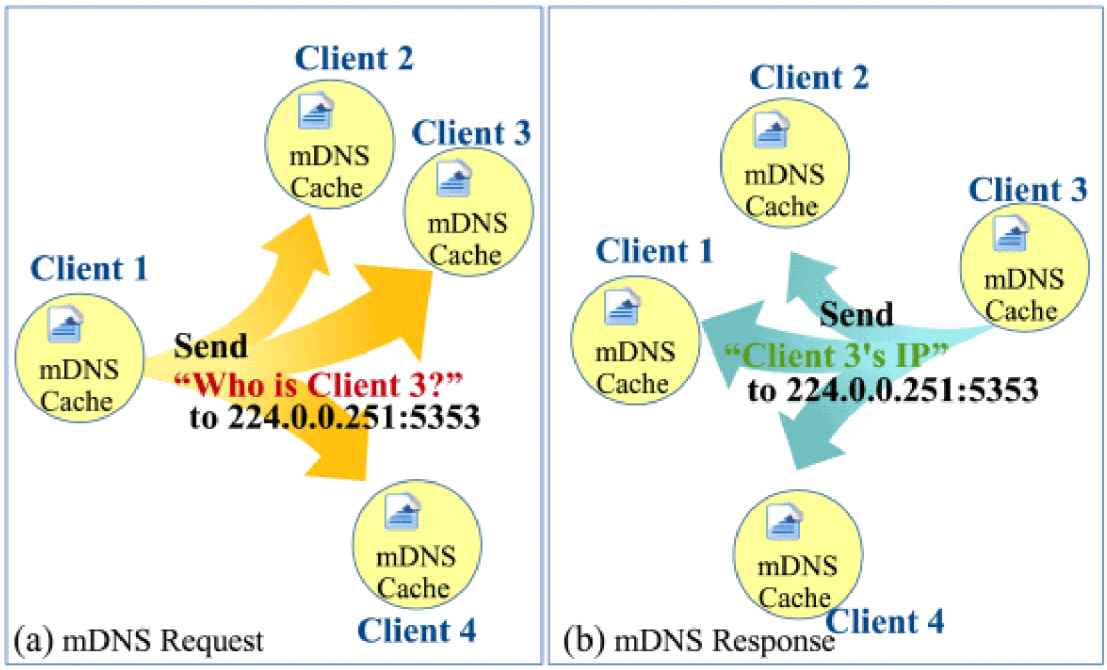
Oba protokoly byly původně vytvořeny s myšlenkou zařízení, které mají k dispozici velké množství zdrojů, například co se týče energetického vytížení. Prozatím pouze existují studie a snahy o adaptaci těchto protokolů pro prostředí IoT. [6][44][45]

Přesto jsou využívané k zjišťování dostupných zdrojů a služeb v senzorových sítích. Service Discovery protokoly jsou v IoT sítích žádoucí kvůli jejich požadavkům na vysokou škálovatelnost, která vyžaduje takové řízení zdrojů, které je efektivní, s minimální obslužností a je dynamické. [6]

Domain Name System (v českém jazyce Systém doménových jmen), který je označovaný anglickou zkratkou DNS, je protokol, který je součástí aplikační vrstvy modelu TCP/IP. Dle RFC 8499 neexistuje jedna konzistentní definice pro protokol DNS. [46] RFC je formální dokument vydávaný organizací IETF. [47] Zmíněné RFC popisuje DNS takto: *„Obecně využívané jmenné schéma pro objekty, které jsou součástí internetu. Je to distribuovaná databáze reprezentující názvy a určité vlastnosti těchto objektů. Je to architektura poskytující distribuovanou údržbu, odolnost a volné vazby pro tuto databázi a jednoduchý protokol na bázi dotaz-odpověď.“* [46]

Multicast DNS (mDNS) je služba, která využívá tzv. unicastového vysílání, které umožňuje jedna ku jedné komunikaci mezi jednotlivými uzli sítě. Mezi výhody mDNS patří následující vlastnosti. Její jmenný prostor je lokální bez nutnosti manuálních rekonfigurací nebo dodatečné administrace zařízení. Je schopná běžet bez podpory infrastruktury a její běh neselže v případě selhání infrastruktury. [6]

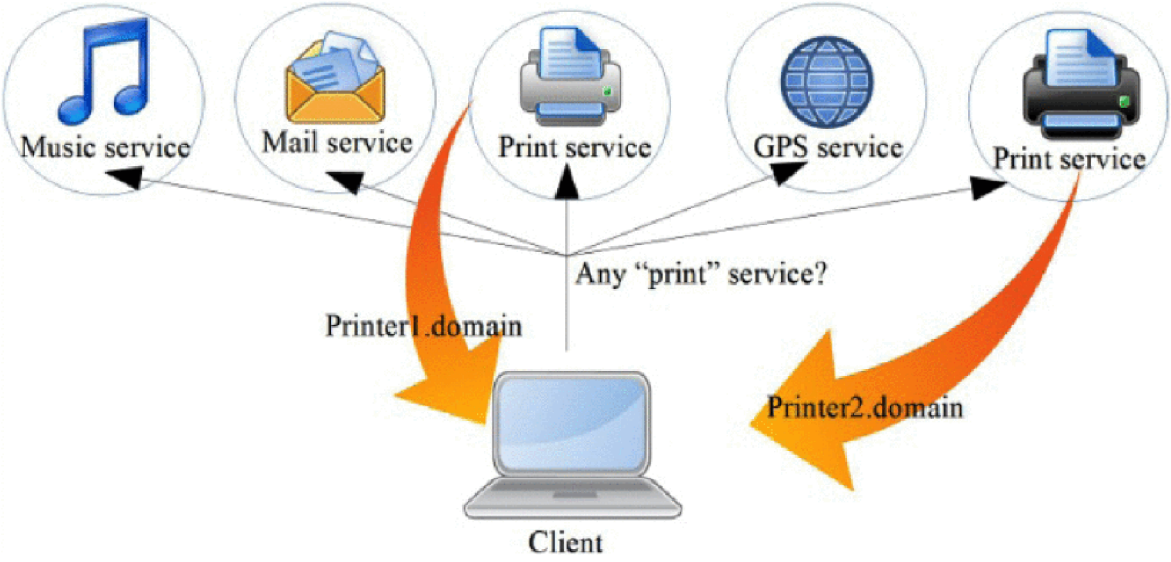
Příklad komunikace služby mDNS je možné vidět na obrázku č. 17. Na základě multicastového vysílání, v rámci lokální domény, si služba vyžádá jména od všech uzlů. Ve formátu dotaz-odpověď. Po přijetí jména, služba znovu za pomoci multicastu, odesílá danému uzlu jeho IP adresu. Zařízení si následně přijaté informace o své IP adrese a jménu uchovávají v lokální paměti. [6]



obrázek č. 17 – Příklad komunikace služby mDNS[17]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza17ab-2444095-small.gif

DNS Service Discovery (DNS-SD) je spárovaná služba, která je nutnou podmínkou pro klienty využívající mDNS službu. Funkcionalita služby spočívá ve schopnosti zjistit množinu požadovaných služeb ve specifické síti za pomoci standartní DNS komunikace. Podobně jako mDNS, DNS-SD nevyžaduje konfiguraci ani externí administraci pro spravování a napojení zařízení. [6]



obrázek č. 18 – Příklad komunikace služby DNS-SD[18]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza18-2444095-large.gif

I přes výhody obou služeb, které nevyžadují konfiguraci, což je potřebná charakteristika pro IoT architekturu, mají tyto služby jednu hlavní nevýhodu. Nevýhoda spočívá ve faktu, že zařízení jsou nucena ukládat informace do své lokální paměti, což je u IoT zařízení, která jsou omezená primárně svými dostupnými zdroji, ať je to energetická náročnost nebo paměťová náročnost, podstatný problém. [6]

3.3.3 Infrastrukturní protokoly

Následující přehled popisuje protokoly využívané v IoT sítích pro fyzickou komunikaci mezi jednotlivými zařízeními:

- Routing Protocol for Low Power and Lossy Networks (RPL)
- 6LowPan
- IEEE 802.15.4
- Bluetooth Low Energy (BLE)
- EPCglobal
- Long Term Evolution (LTE-A)
- Z-Wawe

RPL (v českém jazyce Směrovací protokol pro nízko energetické a ztrátové sítě) je směrovací protokol založený na IPv6 standardizovaný IETF organizací. [47] Je určený pro zařízení, která jsou omezená svými zdroji. Jeho hlavní funkcionalitou je robustní topologie pro ztrátové spoje, v síti, která má požadavky na co nejmenší nutné směrování jednotlivých zpráv. [6]

Technologie protokolu je postavena na orientovaných acyklických grafech, které mají pouze jeden kořen. Konkrétně se pro RPL využívá tzv. směrově orientovaný acyklický graf s anglickou zkratkou DODAG (Direction Oriented Directed Acyclic Graph). Každý uzel v grafu ví o svém rodiči, ale nemá žádné informace o svých sourozencích. [6]

RPL má čtyři různé typy zpráv pro svoji směrovací komunikaci:

- DODAG Information Object (DIO)
- Destination Advertisment Object (DAO)
- DODAG Information Solicitation (DIS)
- DAO Acknowledgement (DAO-Ack)

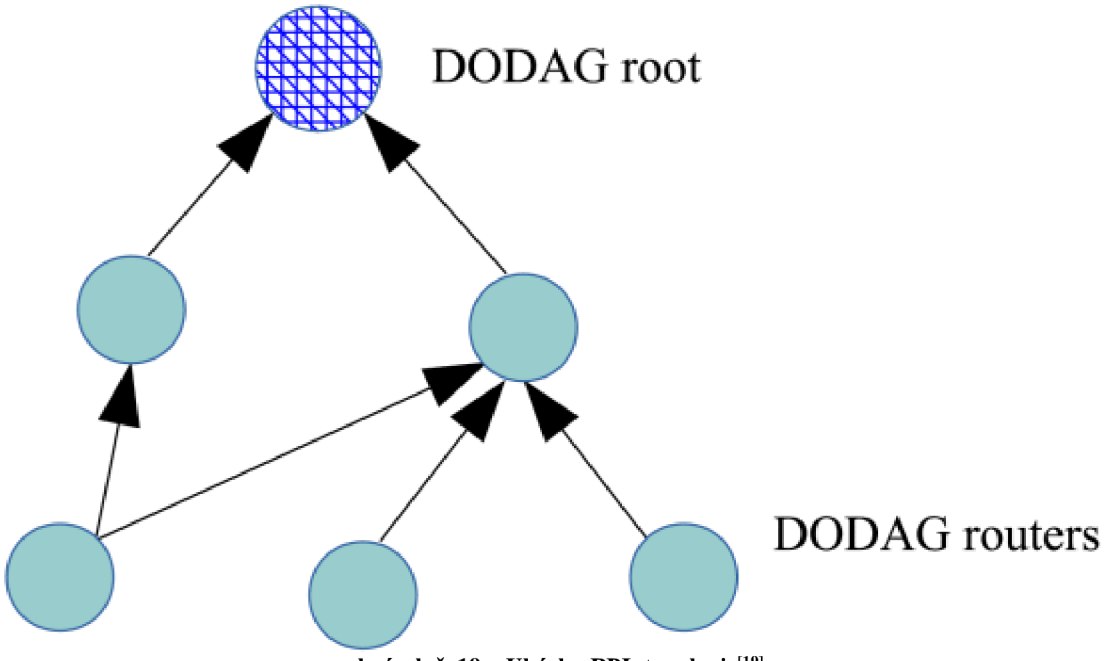
DIO zprávy obstarávají nejdůležitější část funkcionality protokolu, drží si informaci o tom, jakou má daný uzel hloubku v grafu. Tedy jaká je jeho vzdálenost, množství hran (spojů), od kořenu grafu a preferovanou cestu ke kořenovému uzlu. [6]

DAO zprávy jednotlivě rozesílají, libovolným směrem, informaci o destinaci vybraným uzlům. Jako odpověď k DAO zprávám se posílají DAO-Ack zprávy, které daný odpovídající uzel rozliší, buď jako kořenový, nebo jako uzel, ze kterého vzešla DAO zpráva. DIS zprávy jsou využívány k získání DIO zpráv z dostupného sousedícího uzlu. [6]

Existují dva druhy konfigurací tohoto protokolu:

- Ukládající data
- Neukládající data

Na obrázku č. 19 lze vidět ukázku jedné z možných topologií tohoto směrovacího protokolu. RPL protokol je schopný podporovat různé druhy vícebodových i dvoubodových topologií. [6]



obrázek č. 19 – Ukázka RPL topologie[19]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza19-2444095-small.gif

Nízko energeticky náročné bezdrátové osobní sítě (v anglickém jazyce Low power Wireless Personal Area Networks), jsou často využívané v IoT sítích. [48] Od klasických technologií, nacházejících se na spojové vrstvě, se liší svou omezenou velikostí paketů, variabilní adresní délkou, malou šířkou pásma a zmenšenou hlavičkou. [6] Stejně jako u RPL, je 6LowPAN standard, který je spravovaný organizací IETF. [47][6]

Organizace IETF vydala 6LowPAN standard v roce 2007, standard je založený na protokolu IEEE 802.15.4, oproti IPv6 byla odstraněna nadbytečná data, tak, aby vznikl zmenšený paket, který má velikost hlavičky pouze dva byty. [49][6]

Protokol IEEE 802.15.4 je využíván v IoT sítích pro své následující vlastnosti:

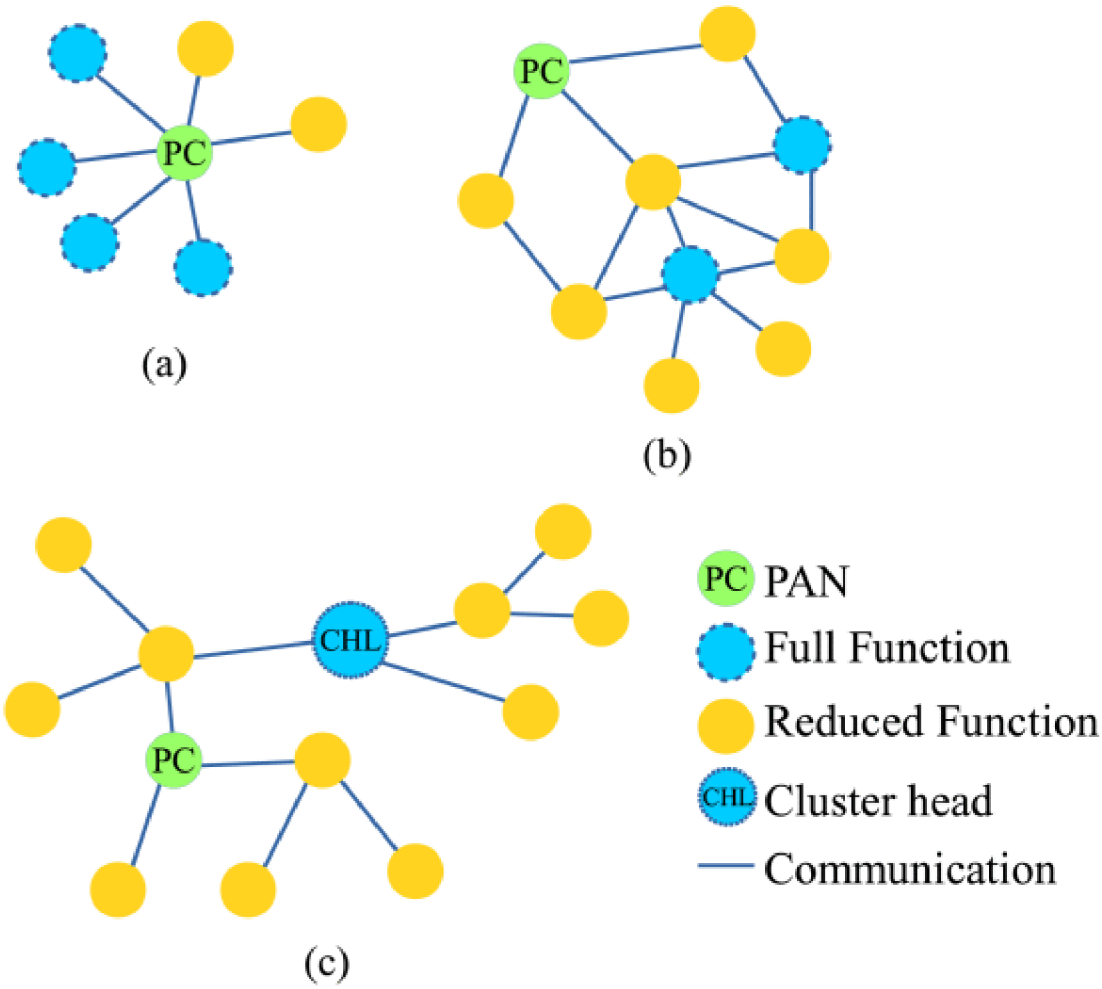
- Nízká energetická spotřeba
- Nízké přenosové rychlosti dat
- Nízké cenové náklady
- Vysokou datovou propustnost
- Spolehlivou komunikaci
- Provozuschopnost na různých platformách
- Schopnost zvládat velké množství uzlů (až 65 tisíc)
- Vysokou míru bezpečnosti, šifrování a autentizaci

Přesto, že má protokol zmíněné výhody postrádá garanci kvality služeb (QoS). IEEE 802.15.4 je základem pro ZigBee nebo např. LoRaWAN protokol. Oba protokoly (RPL a IEEE 802.15.4) se soustředí na nízkou přenosovou rychlost dat, pro zařízení, která jsou závislá na co nejnižší energetické spotřebě. Spolu tvoří kompletní technickou základnu pro bezdrátové senzorové sítě. [6]

IEEE 802.15.4 podporuje tři frekvenční kanály a tři rychlosti přenosu dat:

- 250 kb za sekundu při frekvenci 2.4 GHz
- 40 kb za sekundu při frekvenci 915 MHz
- 20 kb za sekundu při frekvenci 868 MHz

Vyšší frekvence a širší pásma umožňují vyšší propustnost společně s nízkou latencí, naopak nízké frekvence umožňují lepší citlivost a jsou schopné pokrýt větší vzdálenosti. Na obrázku č. 20 je možné vidět diagramy standartních topologií využívaných protokolem IEEE 802.15.4. [6]



obrázek č. 20 – IEEE 802.15.4 topologie, a) Hvězda, b) Peer-to-Peer, c) Shluková stromová (Cluster-tree) topologie[20]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza20abc-2444095-small.gif

Bluetooth Low Energy technologie ve zkratce BLE se oproti jiným verzím Bluetooth technologie liší svou minimální energetickou náročností. Je schopná pokrýt vzdálenosti až do sto metrů, což zhruba odpovídá desetinásobku standartního Bluetooth. Zároveň s patnácti násobně nižší latencí. [50] Využívá krátkých rádiových vln a nízké energetické náročnosti, která se pohybuje mezi 0,01 mW až 10 mW. [51][6]

V porovnání se ZigBee je BLE efektivnější ve spotřebě energie a poměru spotřebované energie za přenesený bit. [52] BLE využívá hvězdicové topologie a povoluje zařízením v její síti operovat jako tzv. Master nebo Slave zařízení. [6]

Electronic Product Code ve zkratce EPC (v českém jazyce Elektronický Produktový Kód) je unikátní identifikační číslo, které je uloženo na RFID tagu. Je nejčastěji používaný v logistice za účelem označení přepravovaného zboží. EPCglobal je organizace původně zodpovědná za vývoj EPC, spravuje EPC a RFID a standardy spojené s RFID a EPC. [6]

Podle současných zjištění lze EPC charakterizovat takto: *„Tato architektura je uznávána jako slibná technologie pro budoucnost IoT, díky její otevřenosti, škálovatelnosti, interoperabilitě a spolehlivosti. I mimo její primární podporu IoT požadavků jako jsou identifikace a schopnost nalezení zařízení v síti.“* [53][6]

EPC je klasifikováno do čtyř typů:

- 96bitové
- 64bitové (I)
- 64bitové (II)
- 64bitové (III)

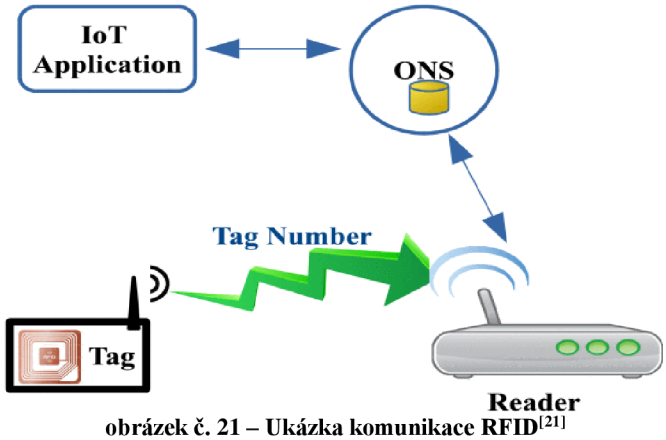
RFID systém může být rozdělen do dvou hlavních částí:

- Transpondér rádiového signálu (Tag)
- Čtečka tagů

Tag má dvě části svých komponentů:

- Čip pro ukládání unikátní identity objektu
- Anténu pro umožnění komunikace čipu se čtečkou tagů za pomoci rádiových vln

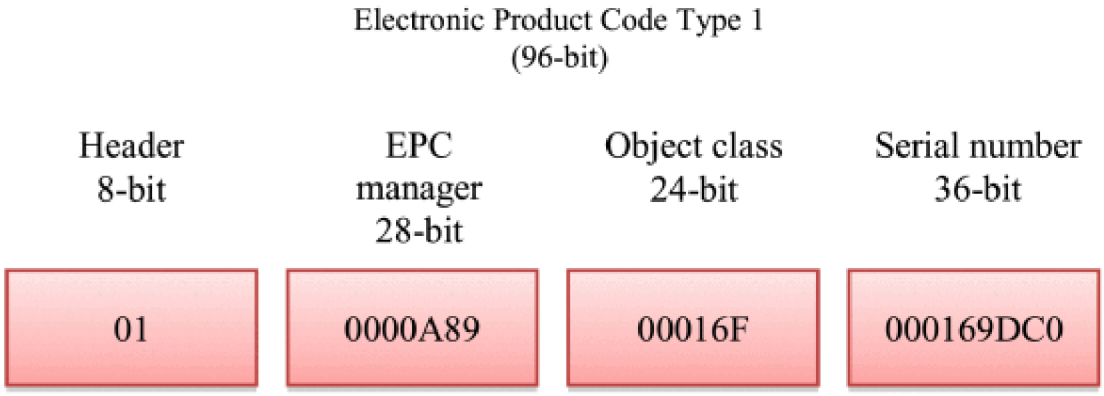
Na obrázku č. 21 lze vidět, jak RFID posílá číslo tagu čtečce tagů za pomoci rádiových vln. Poté je číslo tagu předáno počítačové aplikaci nazvané Object-Naming Services ve zkratce ONS. ONS vyhledává detaily k jednotlivým tagům v databázi, např. kdy a kde byl tag vyroben. [6]



obrázek č. 21 – Ukázka komunikace RFID[21]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza21-2444095-small.gif

Na obrázku č. 22 lze vidět formát unikátního čísla EPC, které se skládá ze čtyř částí.



obrázek č. 22 – Ukázka EPC[22]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza22-2444095-small.gif

Z-Wave je nízko energeticky náročný bezdrátový protokol využívaný především v chytrých domácnostech. [54] Protokol je schopný pokrýt komunikaci z bodu do bodu (point-to-point v anglickém jazyce) ve vzdálenosti až do třiceti metrů. Specializuje se na přenos dat ve velmi malých datových objemech vhodných pro chytrá domácí zařízení. Jako příklady využití lze uvést ovládání světel, termostatických hlavic a dalších. [6]

Protokol operuje v ISM frekvencích, které se pohybují kolem hodnot frekvence zhruba 900 MHz s přenosovou rychlostí dat až 40 kb za sekundu. Novější verze protokolu podporují přenosové rychlosti až 200 kb za sekundu. [6]

V rámci porovnání jednotlivých protokolů a technologií bylo na základě různých studií zjištěno následující. Jedna studie výkonnosti RPL protokolu [55] poukázala na několik problémů spojených s touto technologií:

- Nedostačující specifikace
- Nekompatibilita dvou konfiguračních režimů
- Smyčky

Naopak jiná studie poukazuje na efektivitu protokolu díky jeho rychlosti nastavení senzorové sítě a ohraničenou délku zpoždění v komunikaci. Naopak za jeho nevýhody považuje vysoké režijní nároky. [56]

Další studie zkoumající RPL protokol dokonce poukazovala na problémy s nespolehlivostí z důvodu chybějících informací o kvalitě jednotlivých spojů v síti. [57] Směrování je klíčovou součástí IoT infrastruktury. Mnoho dalších vlastností IoT sítí

jako je spolehlivost, škálovatelnost a výkon výrazně spoléhají na RPL, proto bude nutné ji nadále optimalizovat a zlepšovat. [6]

Analýza výkonnosti 6LoWPAN standardu v bezdrátových senzorových sítích využívajících komunikace z bodu do bodu (point-to-point v anglickém jazyce) poukázala na zvýšené zpoždění potvrzení přijetí zprávy při vyšších velikostech ICMP (Internet Control Message Protocol) zpráv. [58][6]

Při komparaci mezi BLE a IEEE 802.15.4, byla v jedné z analýz zjištěna, nižší energetická náročnost technologie BLE [52]. V jiné výkonností komparaci mezi protokolem IEEE 802.15.4 a protokolem IEEE 802.11ah bylo zjištěno, že IEEE 802.11ah má lepší propustnost. Na druhou stranu protokol IEEE 802.15.4 má nižší energetickou náročnost než protokol IEEE 802.11ah. [59][6]

Při komparaci dvou technologií běžně používaných v chytrých domácnostech, přesněji ZigBee a Z-Wave, bylo zjištěno, že Z-Wave má přijatelnou výkonnost. I přesto, že je Z-Wave dražší než jeho protějšek, je poměrně často využíván v domácích řešeních. Zároveň Z-Wave disponuje výhodou větší flexibility a bezpečnosti. Celková výkonnost Z-Wave protokolu je dle jiných studií vyšší než protokolu ZigBee. I přesto, že má ZigBee výrazně vyšší přenosovou rychlost dat než Z-Wave. [60][6]

Obrázek č. 23 ukazuje komparaci jednotlivých infrastrukturních protokolů a jejich dostupných funkcionalit.

PHY Protocol	Spreading Technique	Radio Band (MHz)	MAC Access	Data Rate (bps)	Scalability
IEEE 802.15.4	DSSS	868/ 915/ 2400	TDMA, CSMA/CA	20/ 40/ 250 K	65K nodes
BLE	FHSS	2400	TDMA	1024 K	5917 slaves
EPCglobal	DS-CDMA	860~960	ALOHA	Varies 5~640 K	-
LTE-A	Multiple CC	varies	OFDMA	1G (up), 500M (down)	-
Z-Wave	-	868/908/ 2400	CSMA/CA	40K	232 nodes

obrázek č. 23 – Komparace infrastrukturních protokolů a jejich funkcionalit[23]

Zdroj: https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/9739/7331734/7123563/guiza.t6-2444095-small.gif

3.4 Integrace dat z IoT zařízení

Definice integrace zní takto: *„Obecný pojem „integrace“ můžeme chápat jako propojování počítačových systémů, společností nebo lidí.“* [8]

3.4.1 Middleware a IoT Platformy

Kvůli velké rozlišnosti IoT zařízení vznikla nutnost pro integraci dat ze zařízení do ucelených systémů. V posledních letech je běžnou praxí zavádění tzv. middleware řešení, anebo taktéž IoT Platforem. [61]

IoT Platformy mají za cíl řešit hned několik dílčích cílů:

- Správu heterogenních zařízení
- Zpracování dat z různých zařízení a zdrojů, jako komunikačních bran a brokerů (vydavatelů)
- Homogenizaci zpráv z rozlišných zařízení do jednotného formátu
- Zpracování a persistenci homogenizovaných dat
- Integraci s dalšími službami v decentralizované architektuře

Všechny zmíněné dílčí cíle by měly napomáhat k větší interoperabilitě mezi jednotlivými zařízeními a standardy. Platforma by měla zamezovat, nebo alespoň snižovat, proprietární uzamčení a zlepšovat nároky IoT systémů na škálovatelnost a komunikaci mezi jednotlivými zařízeními. [3][6][20][61]

Existují jak placené formy, tak i pokusy o vlastní implementace na míru. Lze vyjmenovat některá placená řešení ve formě PaaS (Platform as a Service) či SaaS (Software as a Service) řešení:

- Amazon AWS IoT [62]
- Azure Event Hub [63]
- IBM Watson IoT [64]
- Oracle Internet of Things Cloud Service [65]
- Google Cloud IoT Core [66]
- ThingsBoard [67]
- SensorUp [68]
- DeviceHive [69]

Mnohá z řešení nabízí distribuované, škálovatelné a často i modulární a multiplatformní a multiprotokolové systémy. [8][61] Jejich nevýhodou je ale fakt, že často, již dopředu, rozhodují a omezují, jakou budou mít data konečnou formu. Tedy jak budou data zpracovávána a zobrazována. Na druhou stranu jsou zmíněná řešení připravena k okamžitému používání, kdy často není potřeba věnovat čas implementaci vlastní platformy. Vlastní implementace může být často složitá a časově a finančně náročná. I tak je nutné zmínit, že IoT Platformy se často musí platit, ve formě licence, která je časově omezená. [8] Placené platformy taktéž mohou mít uzavřené API, mohou mít problémy s přijímáním nových standardů a zařízení a nemusí dávat přístup ke zpracování a monitoringu všech dostupných dat. [70]

Dalším zajímavým řešením problematiky jsou middleware systémy. Různá IoT zařízení mohou používat různé formy komunikačních protokolů i datových formátů. Middleware systém může být využit k sjednocení komunikace a datových formátů s napojením na IoT Platformu. Dále může validovat zprávy, ošetřovat autorizaci a autentizaci uživatele. [70][8]

Problematikou middleware systémů zůstává jejich různorodý přístup k abstrakci programatického řešení a různých architektur. Ať už pro přístup k IoT datům nebo napojení na jednotlivá zařízení. Studie *„Middleware for the Internet of Things: A survey on requirements, enabling technologies, and solutions“*, z roku 2021, představuje velmi propracovaný přehled popisující middleware systémy z pohledu jejich požadavků, potřebných technologií, taxonomie existujících middleware systémů a současných otázek a budoucích problémů k řešení. [71]

Dle zjištěných poznatků lze požadavky pro middleware systémy rozdělit na povinné a nepovinné. Povinné požadavky musí splňovat všechny middleware systémy, zatímco nepovinné jsou nutné pouze pro určité specifické aplikace. Např. ve zdravotnictví existuje požadavek na schopnost systému téměř okamžitě a korektně zpracovat data. Jiné systémy už takové nároky na okamžité zpracování dat mít nemusí. [71]

Mezi povinné parametry middleware systémů patří interoperabilita a bezpečnost. Jak již bylo několikrát zmíněno interoperabilita je nutná z důvodu heterogennosti IoT zařízení a jejich produkovaných dat. S IoT zařízeními je velmi často spojený bezdrátový přenos dat, zároveň jsou zařízení omezená svými zdroji, díky kterým nejsou schopná podporovat složitá bezpečnostní řešení. Díky jejich bez obslužnému stavu a již zmíněné bezdrátové komunikaci jsou díky těmto podmínkám náchylná na bezpečnostní útoky. [71]

Jako další nepovinné vlastnosti middleware systémů lze zmínit:

- Schopnost samostatné změny chování systému
- Okamžité zpracování dat
- Škálovatelnost
- Spolehlivost
- Jednoduchou strukturu řešení

Všechny vyjmenované vlastnosti mohou a nemusí být povinné pro konkrétní řešení. Záleží na zaměření daného systému. Například řešení spojená se zdravotnictvím budou mít komplexní řešení velmi často nasazená v cloudu. Naopak middleware systémy řešící zařízení v senzorových sítí nesmí být komplexní. [71]

Obrázek č. 24 zobrazuje přehled a komparaci existujících middleware systémů a jejich schopnosti splňovat popsané vlastnosti:

Type	Middleware	Platform	Interoperability[a]	Real-timeness	Scalability	Security	Light-weightness	Context awareness	Knowledge discovery	self-adaptability
Service based	Hydra	Sensor node	SE	√	√	√	√	√	×	×
	CoCaMAAL	Sensor node, cloud	SY	√	√	×	×	√	×	×
	BDCaM	Sensor node, cloud	SE	√	√	×	×	√	√	×
	Atlas	Sensor node	SE	√	√	√	×	√	×	√
Event based	IoT-MP	Sensor node, cloud	SY	√	×	√	×	×	×	×
	PRISMA	Sensor node	SY	√	×	×	×	√	×	√
	SeCoMan	Sensor node, cloud	SE	√	×	√	×	√	√	×
	SenSocial	Mobile phone	SY	√	×	√	√	×	×	×
VM based	Mate	TinyOS	SY	√	√	×	√	√	×	√
	SensorWare	Linux, ARM platform	SY	√	×	×	√	√	×	×
	Actinium	JVM	SY	×	√	√	×	×	×	√
	MODE	JVM	SY	√	√	×	×	√	×	√
Database based	TinyDB	TinyOS	×	√	×	×	√	×	×	×
	SINA	Sensor node	×	√	×	×	√	×	×	×
	Cougar	Berkeley MICA, Unix	×	√	×	√	√	×	×	×
	DSWare	JVM	SY	√	×	×	√	×	×	×
Agent based	Agilla	MICA2 Mote, TinyOS	SE	√	×	×	√	√	×	√
	Eagilla	Sensor node	SE	√	√	√	√	√	×	√
	ACOSO	Smart object	SY	√	√	√	√	√	×	√
	Sensomax	JVM, ARM platform	SY	√	√	√	√	×	×	√

*red: mandatory features; green: optional features depend on the application scenario.
[a]SEmantic (SE); SYntactic (SY).

obrázek č. 24 – Přehled a komparace Middleware systémů[24]

Zdroj: https://ars-els-cdn-com.infozdroje.czu.cz/content/image/1-s2.0-S1383762121000795-fx1001.jpg

Menší pozornost ve studiích je věnována řešením IoT platforem, které jsou otevřené nebo jsou zadarmo. [70] Jedna z mála studií, která zpracovala přehled volně přístupných platforem zmiňuje následující systémy:

- Eclipse Kapua [72]
- Home Assistant [73]
- Mainflux [74]
- openEMS [75]
- openHAB [76]
- OpenRemote [77]
- SiteWhere [78]
- ThingsBoard [67]

Studie porovnává jejich licencování, zaměření, dostupné programovací jazyky, podporované integrace, podporované databáze, automatizace, možnosti datových analýz a vizualizací. [70]

Dalším zajímavým kandidátem volně přístupných IoT platforem je otevřený standard, který podporuje přenos prostorových dat. Je zde snaha o zavedení otevřeného standardu se standardizovaným aplikačním rozhraním, nad kterým lze stavět vlastní IoT Plaftormu, buď za využití RESTful API, nebo MQTT brokera. Jde o SensorThings API od OGC (Open Geospatial Consortium) [122], které staví na myšlence sémantického webu, který dle zmíněné studie middleware systémů bude mít stále větší a větší význam s nárůstem rozsahu IoT dat. [71]

Sémantický web je jeden ze tří pohledů na IoT. Dalšími směry jsou směry se zaměřením na Internet a na samotná zařízení. Zatímco směr zaměřený na zařízení se primárně soustředí na hardwarové technologie, sběr dat a komunikační technologie, tak směr se zaměřením na Internet se především zabývá obecnými koncepty jako je Web of Things (Web věcí) a IP for Smart Objects (IPSO). [71]

Oproti zmíněným směrům se naopak sémantický směr soustředí na schopnost získávání informací z dat různými metodami, jako je např. Context-aware procesování a získávání informací. Je zde důraz na to, aby měl systém povědomí o současné lokaci zařízení. Podstatou sémantického směru je myšlenka, že ze zpracovaných dat musí být možné udělat informované, a především správné rozhodnutí. [71]

Přesnější definice zmíněného standardu zní: „*SensorThings API od OGC je otevřený standard, který podporuje přenos prostorových informací a je navržen tak, aby usnadnil naplňování konceptu IoT zavedením společného datového modelu a rozhraní pro komunikaci mezi zařízeními navzájem a zařízeními a aplikacemi přes síť Internet.*“ [8][3]

Standard se skládá ze dvou částí, Snímání (Sensing) a Úkolování (Tasking). Existují zatím pouze dvě plně certifikované implementace [17] SensorThings API: FROST Server 2.X [16] a implementace od organizace ICCS, která již není na uvedeném odkazu od OGC dostupná [80].

3.4.2 Integrační topologie

Existují čtyři základní integrační topologie: *„Jedná se především o integraci Point to Point, Broker Based, Publisher-Subscriber, Message Bus.“* [8]

Z pohledu sémantického webu se pro IoT komunikaci jeví jako nejvhodnější forma integrační topologie Publisher-Subscriber. A to díky své schopnosti posílat zprávy na základě jejich obsahu. K odběratelům (Subscriber) přicházejí pouze zprávy, o které mají reálně zájem. Topologie dále umožňuje, aby middleware systémy nebyly tzv. bottle neckem IoT řešení, jelikož umožňují horizontální škálování. Přijaté zprávy jsou tříděny a rozesílány odběratelům v momentě jejich přijetí. [8][70]

Nadstavbou Publisher-Subscriber řešení je pak Message Bus, která ke zdroji a příjemci zpráv přidává prostředníka tzv. zprostředkovatele, který přijímá zprávy a stará se o jejich přesun ke konkrétním odběratelům. [8][71]

3.4.3 Databázové systémy

V současné době se nejčastěji používají tři typy databází pro persistenci IoT dat. Klasické relační databáze, NoSQL (Not Only SQL) databáze a databáze určené pro časové řady. [71]

Relační databáze jsou založené na relačních modelech. [70] *„Jsou to kolekce tabulek, které mezi sebou mají relace, kde jednotlivé tabulky mají své atributy a omezení. Relační databáze používají dotazovací jazyk SQL (Structured Query Language) za jehož pomoci manipulují s daty v databázi. SQL jazyk je založený na tzv. ACID principu.“* [81]

Databáze NoSQL nevyužívají dotazovacího jazyka SQL jako v případě relačních databází a neexistují v nich relace. Umožňují pružně škálovat a jsou navrhnuté pro používání

na nízko nákladovém hardwaru. Jsou schopné persistovat obrovská množství dat, nemají schéma, umožňují distribuci dat. [81]

Oba databázové systémy čelí více bezpečnostním rizikům jako jsou například rootkity a SQL injekce. NoSQL databáze jsou náchylnější na bezpečnostní útoky díky nestrukturovanosti a velké distribuci uložených dat. Z důvodu velké distribuce dat je nutné pro NoSQL databáze implementovat komplexnější bezpečností procedury. [81]

Databáze určené pro časové řady umožňují efektivní ukládání dat popisující časové řady spojené s dalšími užitečnými hodnotami. Díky této sémantické vlastnosti jsou často využívaným formátem pro persistenci dat v IoT Platformách. [70]

Následující tabulka č. 2 zobrazuje komparaci běžně využívaných databázových systémů v IoT:

Typ databáze	Softwarový projekt	Zodpovědný vývojář	Datum prvního vydání	Licence	Komerční podpora
Relační	PostgreSQL [82]	PostgreSQL Global Development Group	8. 7. 1996	PostgreSQL [92]	Externí
	MySQL [83]	Oracle	23. 5. 1995	GPLv2 [93]	Ano
	MariaDB [84]	MariaDB Foundation	29. 10. 2009	GPLv2 [93]	Ano
	SQLite [85]	D. Richard Hipp	17. 8. 2000	Public domain	Ano
NoSQL	Apache Cassandra [86]	Apache Software Foundation	Červenec 2008	Apache 2.0 [94]	Ne
	Apache Couch DB [87]	Apache Software Foundation	2005	Apache 2.0 [94]	Ne
	MongoDB [88]	MongoDB Inc.	11. 2. 2009	Server Side Public License [95]	Ano
Časové řady	InfluxDB [89]	InfluxData	24. 9. 2013	MIT [96]	Ano
	TimescaleDB [90]	Timescale Inc.	1. 11. 2018	Apache 2.0 [94]	Ano
	OpenTSDB [91]	B. Sigoure et al.	2010	GPLv3 [97]	Ne

tabulka č. 2 – Komparace databázových systémů využívaných v IoT[2]

Zdroj: https://www.sciencedirect.com/science/article/pii/S254266052200107X [70]

3.5 Zpracování dat z IoT zařízení

U zpracování IoT dat jsou primárně důležité co nejrychlejší doba zpracování, náprava ruchů v získaných datech a bezpečnost přenosu dat. [98][99]

Mezi nejčastější rušení v IoT datech patří následující: šum, chybějící naměřené hodnoty, chybějící detekce hodnotových odchylek. Šumy jsou nejčastěji způsobené utilizací nelicencovaných frekvenčních pásem pro komunikaci. [99]

Pro nápravu zmíněných chyb existují různé algoritmy od matematických funkcí až po strojové a hluboké učení. Konkrétní metody jsou mimo rozsah této práce a lze je podrobněji zkoumat z matematického hlediska v této studii [99] z roku 2020 a dále také z programatického a experimentálního hlediska v této studii [100] z roku 2022.

Lze vyjmenovat následující architektury využívané pro zpracování IoT dat:

- Grid
- Cloud
- Fog
- Mobilní Edge
- Cloudlety
- Na místě
- V paměti

Zpracování dat v Gridu využívá distribuovaných systémů, které využívají společnou rozvodnou výpočetní síť vzájemně propojených zdrojů. Tento typ zpracování dat je využívaný primárně ve výzkumu a v průmyslové sféře díky svým velkým výpočetním kapacitám. Jeho velká výhoda spočívá v téměř okamžitých výsledcích koncových dat, a taktéž větší bezpečnosti celého systému a uložených dat. Nevýhodou Gridu je nutnost vlastní správy a údržby daných výpočetních systémů. [98]

Cloud computing má narozdíl od Gridu tu výhodu, že není nutné spravovat ani udržovat výpočetní systémy. Nabízí přístup k vysokým výpočetním výkonům za poměrně nízkou cenu a vysokou možnost škálovatelnosti. Cloud computing je momentálně jedním z nejvíce využívaných řešení. Jeho poměrně velkou nevýhodou je zpoždění výsledků zprocesování dat oproti Gridu. Proto Cloud computing není příliš vhodný pro systémy, které potřebují téměř okamžitou odezvu jako jsou např. nemocniční, letecké nebo systémy monitorující potenciální katastrofy. Dále jsou IoT sítě nadměrně

zatěžovány kvůli potřebě posílání dat k zpracování, na server, běžící v cloudovém prostředí. [98]

Potencionálním řešením nevýhod je Fog nebo taktéž nazývaný Edge computing, kdy data, u kterých je potřeba mít odezvu na jejich zpracování téměř okamžitou jsou zpracována tzv. „na kraji“ dané senzorové sítě. Veškerá data, u kterých není nutná okamžitá odezva a mohou být analyzována později jsou odeslána do cloudu. Řešení má i tu výhodu, že data, které není nutné odesílat do cloudu, zbytečně nezatěžují senzorovou síť. Proto je nastavené zprocesování dat spolehlivější než Cloud computing, jelikož je méně zatěžovaná síť i její zařízení. Nevýhodou obou řešení je chybějící kontrola nad výpočetními systémy v cloudu, které mohou mít občasné výpadky. Zároveň je zde menší kontrola nad uloženými daty, jelikož jsou uložená na cizím serveru. Je zde i potenciální bezpečnostní nevýhoda, jelikož jsou cloudové servery veřejně dostupné a tím pádem jsou i více náchylné na potencionální útoky. [98]

Mobilní Edge computing je velice podobný zmiňovanému Fog computingu, liší se především formou sběru dat v senzorové síti. Data jsou sbírána za pomoci mobilních a chytrých zařízení a jako síť je využívaná existující mobilní síť. Kdy mobilní zařízení jsou považovaná jako zařízení „na okraji“ sítě. Nepotřebují primárně využívat cloudu. Díky rychlému zpracování se pak signifikantně zvyšuje použitelnost systému pro koncové uživatele. Nevýhodou těchto řešení je, že se nemohou porovnávat co se týče spolehlivosti, výpočetní síly ani škálovatelností s cloudovými řešeními. Zároveň je zde ještě větší potencionální bezpečnostní riziko než v cloudovém prostředí, kdy chybí kontrola nad jednotlivými zařízeními a je snadné odposlouchávat mobilní sítě. [98]

Cloudlety jsou kombinace Mobilního Egde a Cloud computingu. Vyvíjené na univerzitě Carneige Mellon. [101] Cloudlet je využívaný jako střední vrstva mezi Cloudem a mobilními zařízeními. Cílí na snížení odezvy přiblížením Cloudových služeb k „okraji“ senzorové sítě. Pokud Cloudlet selže nebo není v blízkosti senzorové sítě nalezen je zprocesování dat provedeno na zařízení, nebo je směrováno na klasický Cloud. V případě zprocesování dat na Cloudu je potřeba počítat se stejnými nevýhodami jako u klasického Cloudového řešení. [98]

Zprocesování dat tzv. „Na Místě“ spočívá v zprocesování dat přímo v IoT zařízeních. Zde je velký rozdíl oproti předchozím zmíněným architekturám. Typicky IoT zařízení nemají dostatek zdrojů na zprocesování dat, existují ale případy, kdy zařízení nemají anebo mají velmi špatný přístup k externí infrastruktuře a zároveň je nutná okamžitá odezva

zařízení s výsledky z dat. Toto řešení je pouze vhodné pro aplikace, kde není nutné archivovat, centrálně zpracovat nebo spojovat výsledky dat z více zdrojů. I přesto, že je řešení poměrně omezené, co se týče ukládání dat, tak pořád je ze všech zmíněných řešení nejbezpečnější, jelikož nevyžaduje komunikaci mimo svojí interní síť. [98]

Zprocesování dat „V paměti“ se potencionálně jeví jako jedno z nejvýkonnějších řešení. V tradičních výpočetních systémech se data přenáší pro zpracování do výpočetních jednotek, které využívají především ukládání na sekundární a záložní paměti. Následně jsou takto uložená data přenášena za pomoci primární paměti, nejčastěji RAM, do procesoru, kde za pomoci vyrovnávací paměti jsou data postupně zpracovávána. Po dokončení zpracování dat jsou data znovu přes primární paměť zapsána na sekundární paměť a tento cyklus lze opakovat do nekonečna. Oproti tomuto způsobu zpracování dat je zpracovávání dat přímo na primární paměti signifikantně výkonnější. Je zde ale nevýhoda samotné ceny dynamických pamětí i fakt toho, že DRAM jsou volatilní paměti, které po ztrátě napájení ztrácí svá data. [98]

4 Vlastní práce

Implementaci IoT systému a jeho celého životního cyklu není možné pokrýt v rámci jedné bakalářské práce. Proto je ve formě analýzy, pro prototyp, vybrána pouze omezená část životního cyklu pro sběr, přenos, integraci a zpracování IoT dat.

Vlastní práce se věnuje analýze a komparaci komunikačních protokolů, dostupných implementací standardu SensorThings API a dostupných databázových systémů využívajících časové řady. Na základě výsledné analýzy je z každé kategorie vybrán jeden nástroj za účelem implementace prototypu.

Cílem SensorThings API jako standardu je z různorodých prostorových dat vytvořit využitelné informace, které mohou pomáhat k lepšímu a více informovanému rozhodování na jakékoliv úrovni, ať už v podnikání nebo například ve státní sféře.

Prototyp byl využit v experimentu, který zkoumal efektivnost vybrané implementace standardu, její nevýhody a prostory pro zlepšení. Následně na základě analýzy dokončeného prototypu byla jedna z nevýhod vybrána k potencionálnímu vylepšení dané implementace. Prototyp by měl umožňovat následující:

- Zaregistrovat více senzorů s rozdílnými datovými formáty
- Ukládat prostorová data
- Efektivně vizualizovat a poskytovat uložená data
- Snadnou integraci s ostatními službami v síti Internet

Pro sběr dat byla využita existující senzorová síť vybudována na univerzitním kampusu, proto se analýza nevěnovala možnostem sběru ani výběru senzorů, jelikož je vlastní práce omezena zařízeními dostupnými v univerzitním kampusu.

4.1 Analýza komunikačních protokolů

Na základě analýzy a charakteristiky provedené v teoretické části práce byly ke komparaci vybrány následující vlastnosti:

- Velikost hlavičky přenášených zpráv
- Podporovaná architektura
- Spolehlivost doručení zpráv
- Bezpečnost přenášení zpráv
- Podpora QoS (Quality of Service)

Jak již bylo zmíněno v kapitole č. 3.2.5 nejvíce vhodným formátem pro přeposílání zpráv z IoT zařízení je takový formát, který má nízké nároky na spotřebu energie. Čím větší objem dat se přeposílá z jednotlivých zařízení tím větší je spotřeba energie, proto je nutné se zamýšlet nad možnostmi, jak co nejvíce a efektivně šetřit objem dat v zasílaných zprávách. Například zmenšením síťové režie pro posílání zpráv na co největší možné minimum, tedy přenášením jen skutečně nutných dat pro komunikaci.

Očekává se, že množství zařízení a objem dat v rámci IoT sítí se bude neustále zvyšovat společně i s nároky na zobrazování výsledků, nebo notifikací pro koncové uživatele v reálném čase. Přesto, že je REST běžně zavedená a standardizovaná architektura je to architektura využívající systému žádost-odpověď, který je méně vhodný k získání informací v reálném čase. Zároveň je REST oproti Publisher-Subscriber návrhovému vzoru více omezený v možnosti obsluhovat velké množství dat z více zdrojů najednou. Publisher-Subscriber předává data distribuovaným způsobem a v reálném čase, v okamžiku, kdy je zpráva přijata od zařízení nebo integrovaného systému.

Proto lze očekávat, že s nárustem nároků na reálný čas předávaných zpráv, množství obsluhovatelných zařízení a objemu dat se stane preferovanější architekturou Publisher-Subscriber. Naopak pokud se u systému neočekává velké množství dat a zařízení a je zde důraz na větší bezpečnost může být preferovaná REST architektura.

V implementovaném prototypu byl preferován koncept Publisher-Subscriber i pro jeho lepší možnost horizontální škálovatelnosti a pro lepší vhodnost testování co největšího objemu dat a heterogenních zařízení.

Dle sémantického přístupu k IoT systémům, který byl popsán v kapitole č. 3.4.1, je jedním z hlavních aspektů IoT systému jeho schopnost předávat koncovému uživateli přesná a pravdivá data. Proto lze říct, že čím větší bude v IoT systému spolehlivě přijatých zpráv tím je větší pravděpodobnost korektnosti zpracovaných informací z těchto dat. Je nutné podoktnout, že musí být v systému zaručena funkční validace, očištění a de-duplikace dat, bez kterých není možné produkovat korektní závěry o datech.

Při komparaci dvou nejčastěji využívaných transportních protokolů pro přenos dat v IoT sítích, TCP a UDP, je zřejmé, že spolehlivě doručit zprávy i s možností opakovaného doručení je možné pouze s TCP protokolem, proto byl TCP protokol upřednostněn oproti UDP protokolu.

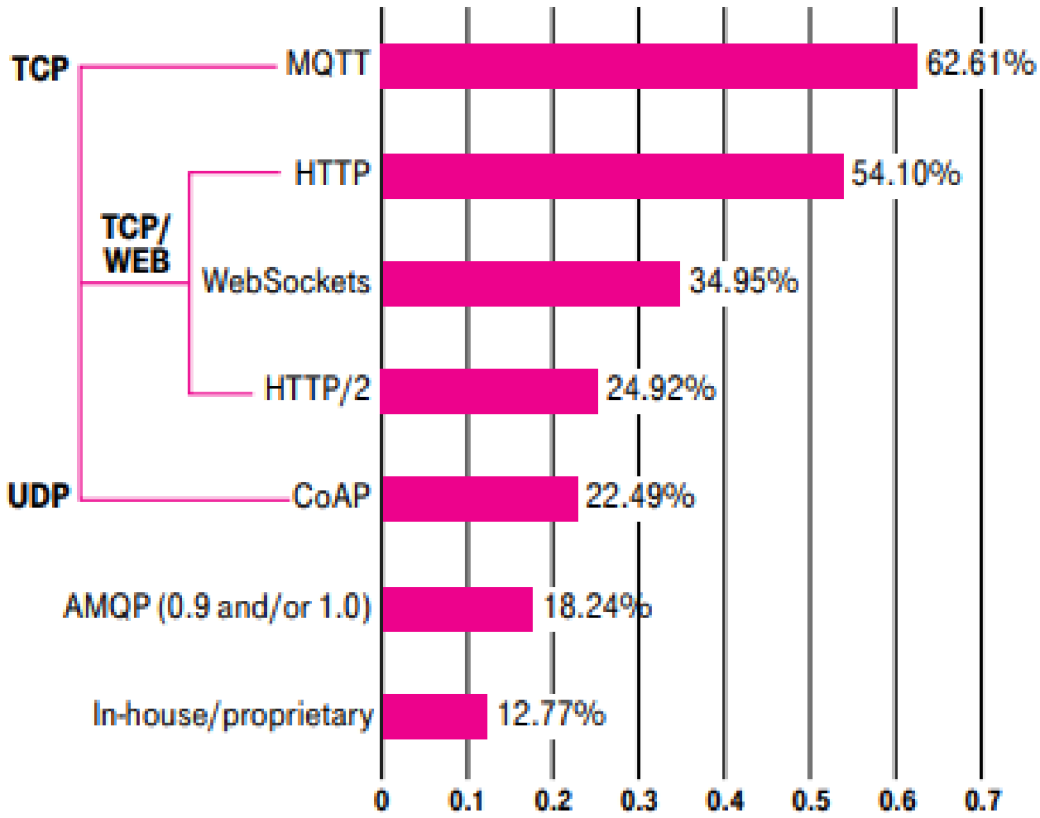
IoT zařízení díky své jednoduchosti a omezeným energetickým zdrojům nemají obecně k dispozici bezpečnostní mechanismy, které by zajišťovaly bezpečný přenos dat

a komunikaci v síti. Proto je nutné mít vysoké nároky na bezpečnost i pro přenosové protokoly, je nutné, aby přenosové protokoly disponovaly možností šifrování přeposílaných zpráv např. za pomoci TLS/SSL protokolů, ideálně v nejnovější a nejspolehlivější formě.

Zařízení v IoT sítích posílají různé typy zpráv, jak již bylo popsáno v kapitole č. 3.2.4. Je důležité, aby mohl správce zařízení, IoT systémů, nebo celé IoT sítě určit jaké typy zpráv, nebo které konkrétní zprávy mají v systému přednost doručení před ostatními, tak, aby koncoví uživatelé měli možnost správného a včasného rozhodnutí. Tento mechanismus se v počítačových sítích nazývá Quality of Service, je tedy podstatné, aby dané přenosové technologie touto funkcionalitou disponovali.

4.1.1 Komparace analyzovaných protokolů

Následující komparace porovnává protokoly: MQTT, COAP, AMQP a HTTP. Všechny protokoly a jejich vlastnosti již byly popsány v teoretické části práce v kapitole č. 3.3.1. Zmíněné protokoly jsou jedny z nejpoužívanějších komunikačních protokolů v rámci IoT viz. graf na obrázku č. 25.



obrázek č. 25 – Popularita komunikačních protokolů v roce 2018 podle T-Mobilu[25]

Zdroj: https://www.daviteq.com/blog/wp-content/uploads/2020/09/WP_2006_H2_data_transport_protocol.png

Na základě teoretických částí bakalářské práce byl vytvořen výsledek analýzy, který je graficky zobrazený v následující tabulce č. 3.

Komunikační protokol	Velikost hlavičky	Architektura	Transportní protokol	Bezpečnost	QoS
MQTT	2 B	Publisher-Subscriber	TCP	SSL	Ano
COAP	4 B	REST	UDP	DTLS	Ano
AMQP	8 B	Publisher-Subscriber	TCP	SSL	Ano
HTTP	-*	REST	TCP	SSL	Ne

tabulka č. 3 – Komparace základních vlastností komunikačních protokolů[3]

***Velikost hlavičky je závislá na použitém serveru, např. Apache server může mít velikost hlavičky až 8 tisíc bytů**

Zdroj: Vlastní práce

Při komparaci jednotlivých vlastností vychází jako nejvýhodnější kandidáti MQTT nebo COAP protokoly. MQTT na rozdíl od COAP protokolu navíc disponuje výhodou nejmenší velikosti přenosové hlavičky, pouze 2 bytů, a využívá TCP protokolu, který má na rozdíl od UDP k dispozici některé funkce navíc, jako je například fragmentace a možnost spolehlivého a zaručeného doručení zprávy. MQTT taktéž disponuje, oproti COAP, přenosovou technologií na základě konceptu Publisher-Subscriber, která je více vhodná pro sémantický web než REST. COAP sice částečně disponuje Publisher-Subscriber funkcionalitou, ale pouze pro zasílání zpráv monitorující zdroje zařízení.

4.2 Analýza implementací SensorThings API

Obecná charakteristika standardu SensorThings API již byla popsána v teoretické části práce v kapitole č. 3.4.1. Je nutné zmínit, že existují i jiné implementace pro zpracování prostorových dat. V IoT platformě vyvíjené univerzitou se primárně, pro prostorová data, využívá platforma ArcGIS [102], která je implementována společností Esri [103]. ArcGIS nabízí celou řadu funkcionalit co se týče prostorových dat. Od strojového učení, po správu zařízení, vizualizace, bezpečnost a schopnost integrace s jinými službami. V rámci vlastní práce byla vybrána možnost implementace SensorThings API ze dvou důvodů. Prvním hlediskem je, zdali daný systém dodržuje obecnou standardizaci, která je obecně v IoT technologiích opomíjena. Standardizací se zde myslí obecný přístup

z architektonického pohledu na řešení IoT systémů. Pokud v IoT nadále bude chybět společný základní přístup k implementaci systémů bude nadále docházet k proprietárnímu uzavírání koncových uživatelů a zákazníků.

Druhým hlediskem je fakt, že ArcGIS je komerční platformou, která nenabízí verzi, která by mohla být provozována zdarma. Pro individuální a studentské potřeby stojí licence ArcGIS 100 amerických dolarů ročně. Pro komerční sféru již nelze cenu ArcGIS na webových stránkách společnosti Esri nalézt. Lze minimálně předpokládat, že pro komerční účely bude cena vyšší než zmíněných 100 amerických dolarů. Dále je cílem práce ověřit, zdali je daná standardizace pro prostorová data efektivní. Jelikož ArcGIS primárně neimplementuje standard pro implementaci IoT systémů byla za účelem splnění cílů práce vybrána technologie od organizace, která spravuje hned několik standardů pro prostorová data. [122]

Následující tabulka č. 4 graficky zobrazuje komparaci existujících implementací standardu SensorThings API.

Implementace	Správa zařízení	Bezpečnost	Protokoly	Notifikace	Otevřený kód / SDK	Vizualizace	Cena
FROST Server	Ano	X.509, TLS	MQTT, HTTP	Ano	Ano / Ano	Částečně	Zdarma
SensorUp	Ano	X.509, TLS	MQTT, HTTP	Ano	Ne / Ano	Ano	N/A*
GOST	Ano	X.509, TLS	MQTT, HTTP	Ano	Ano / Ano	Ne	Zdarma

tabulka č. 4 – Komparace základních vlastností SensorThings API implementací[4]

***SensorUp cena není známá, kód pro SensorUp není veřejně dostupný, pouze dokumentace k API, pro využití platfomy je nutné kontaktovat zákaznické centrum**

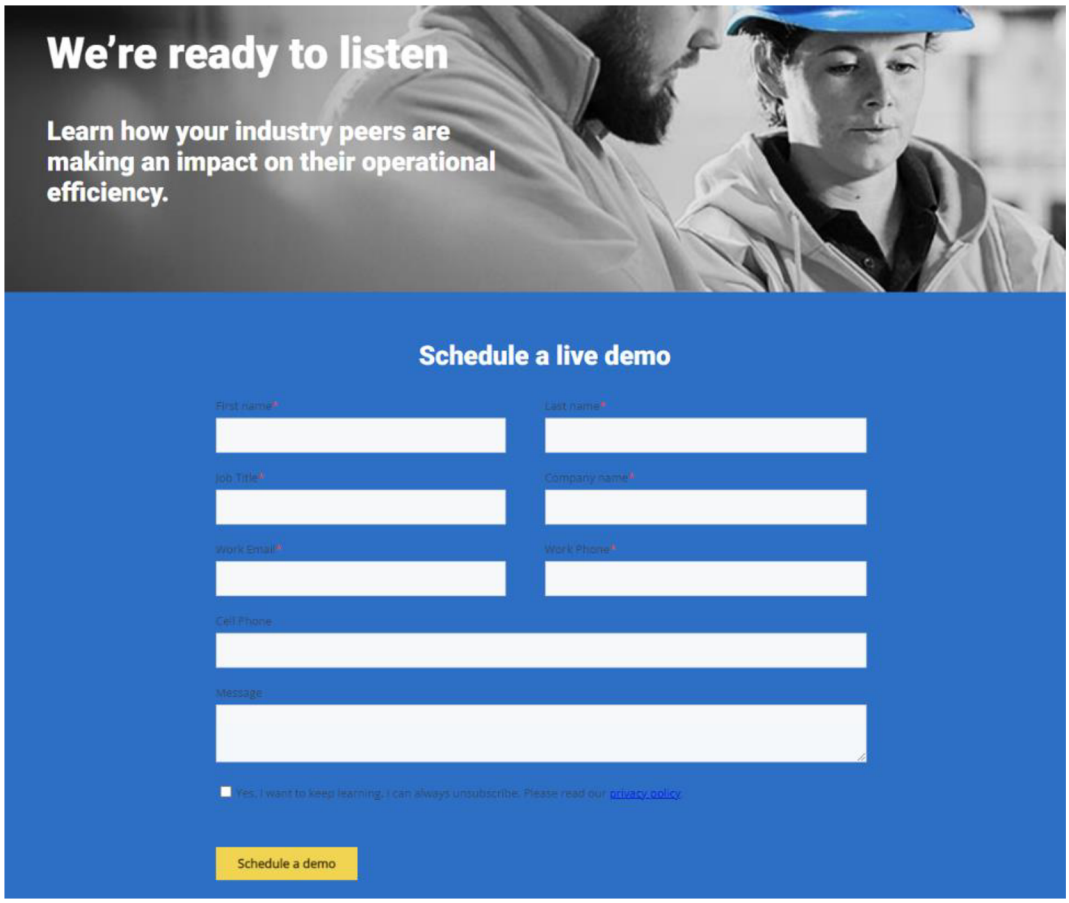
Zdroj: Vlastní práce

4.2.1 SensorUp

SensorUp [68] je ze všech implementací jediným řešením, které se soustředí primárně na komerční sféru a zároveň poskytuje nejvíce dokumentované řešení. Na rozdíl od zbylých dvou řešení se ve své dokumentaci zmiňuje o možnosti vytvoření vizualizací pro koncové uživatele.

Z dostupných zdrojů o SensorUp není jasné, zdali je nabízená platforma placená nebo zdarma. Dalo by se předpokládat, že je toto řešení placené. A to z důvodu

neposkytnutého zdrojové kódu a nutnosti kontaktovat zákaznické centrum společnosti pro přístup k instanci jejich platformy, viz. obrázek č. 26.



obrázek č. 26 – Formulář zobrazený při žádosti o přístup k SensorUp[26]

Zdroj: https://sensorup.com/contact-us/

SensorUp firma byla založena v roce 2011 a je z vybraných implementací nejvyspělejší. Získala několik ocenění [104][105], podílí se i na projektech ve státní sféře v Kanadě [106][107] a spolupracuje už i se zmíněnou firmou Esri [108]. Nabízí ve svém řešení implementaci všech momentálně dostupných a vydaných částí SensorThings API standardu a zároveň ve svém řešení poskytuje nadstavbu ve formě strojového učení, vizualizací a řešení na míru pro komerční využití.

I přesto, že je nejvyspělejší není bohužel vhodná pro implementaci experimentu, a to z důvodu chybějícího snadného přístupu k běžící instanci a zároveň chybějícího zdrojového kódu, ze kterého by bylo možné nasadit řešení.

Ze současných dostupných informací z webových stránek SensorUp je nutné spíše stavět na domněnce, že primární řešení se soustředí na získání co nejvíce platících zákazníků a konkrétně firem, které jsou ochotné zaplatit za řešení tvořené na míru. Tedy nelze příliš předpokládat, že by bylo možné si vyžádat přístup ke zkušební instanci pro studijní práci.

4.2.2 GOST Server

GOST Server neboli v plném znění Go-SensorThings server je IoT platforma implementující standard SensorThings API. GOST není spravován žádnou konkrétní firmou a nemá k dispozici oficiální webové stránky, pouze repositář se svým otevřeným zdrojovým kódem na GitHub platformě. [79] Je tedy k dispozici zdarma pod MIT licencí, což je ze všech implementací nejvíce benevolentní licence. SensorUp svůj kód nemá veřejně k dispozici a FROST Server má své řešení pod GPLv3 licencí.

Oproti ostatním vybraným implementacím není GOST dokončený a není tedy ani vhodným kandidátem pro experiment. Bohužel poslední vydaná verze aplikace je z roku 2017, kdy další verze 0.6 měla být vydána v roce 2018 a očividně se tak již nestalo. Z těchto důvodů se práce nevěnovala dalším analýzám této implementace.

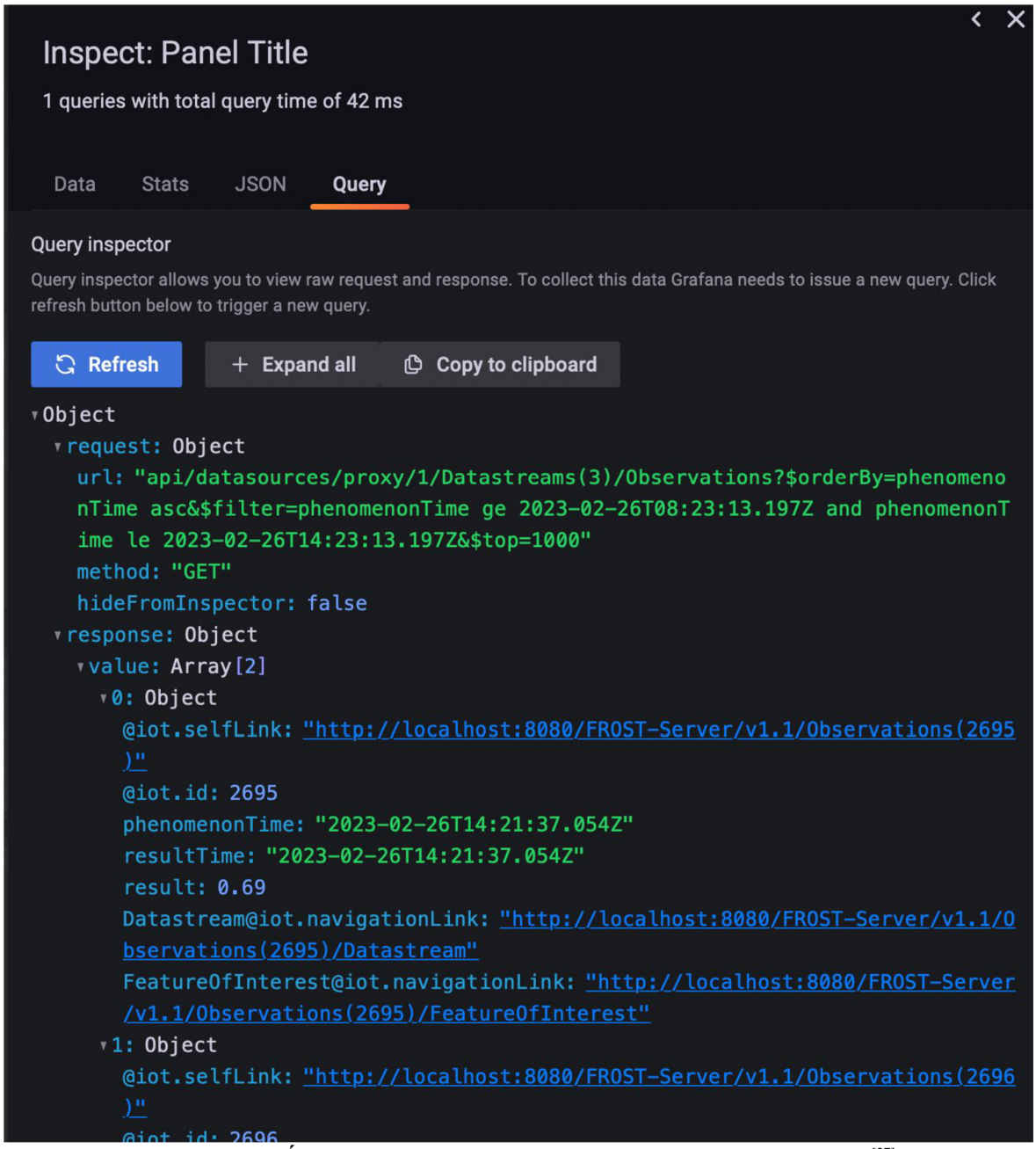
4.2.3 FROST Server

FROST Server je implementován firmou Fraunhofer Institute of Optronics, System Technologies and Image Explotation [109], která byla založena v roce 2010. Oproti SensorUp nabízí implementaci SensorThings API zdarma a jako otevřený kód na GitHub platformě. [16]

FROST Server implementace podobně jako SensorUp nabízí hotovou implementaci všech vydaných API od OGC, tedy části standardu pojmenované jako Sensing a Tasking. Jedním z nedostatků implementace je chybějící funkční základní front end, ve kterém by bylo možné vytvářet a kontrolovat základní vizualizace z integrovaných dat. Taktéž na rozdíl od GOST serveru chybí na jejich GitHub stránkách zmínka o tom, že by byly vizualizace nebo základní front end plánované v příštích verzích.

FROST Server k vizualizacím částečně nabízí vlastní implementaci pluginu do platformy známé jako Grafana [110]. Plugin bohužel nabízí jen velice základní vizualizace v podobě základního spojnicového grafu s naměřenými hodnotami v čase. Nenabízí žádné vizualizace pro prostorová data. Zároveň v současné době není možné dokončit konfiguraci pluginu viz. obrázek č. 27, 28 a 29. Díky této chybě není možné vizualizaci využívat ani v její základní formě.

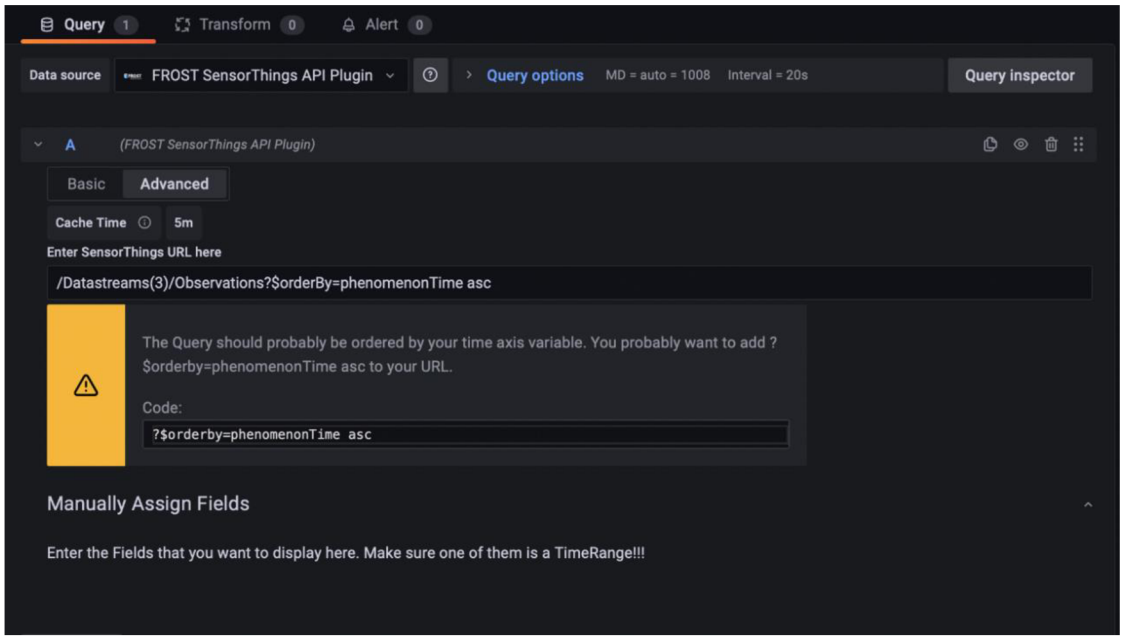
Obrázek č. 27 zobrazuje úspěšné provolání REST API FROST Serveru pro dotaz měřených zkušebních dat.



obrázek č. 27 – Úspěšný dotaz Grafana pluginu proti REST API FROST Serveru[27]

Zdroj: Vlastní práce

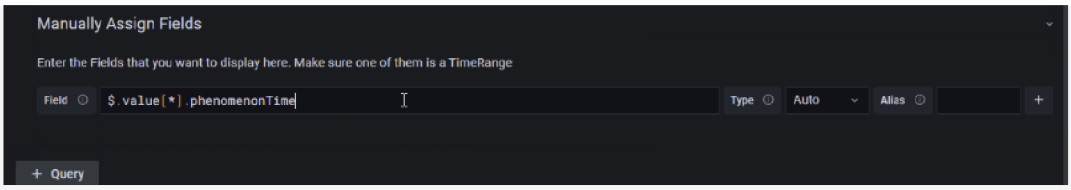
I přesto, že je možné provést úspěšný dotaz pro zaznamená měření proti REST API FROST Serveru není možné dokončit konfiguraci Grafana „Dashboardu“ pro úspěšnou validaci dle návodu ke konfiguraci pluginu viz. obrázek č. 28 a obrázek č. 29.



obrázek č. 28 – Nefunkční sekce pro zadávání popisujících proměnných pro graf v Grafana pluginu[28]

Zdroj: Vlastní práce

Na obrázku č. 29 lze vidět, jak má proběhnout korektní nastavení pluginu podle návodu.



Using the Plugin

1. Click the *Gear* icon in Grafana to enter the configuration menu, select Plugins, and add *FROST SensorThings Plugin*
2. Go back and select *Data Sources*. Click *Add data source*, choose *FROST SensorThings API Plugin*
3. Enter a name for the datasource, and enter the base URL to a SensorThings API server, e.g. `https://airquality-frost.k8s.ilt-dmz.iosb.fraunhofer.de/v1.1/`
4. Add authentication settings if needed and click "Save & Test".
5. Create a new dashboard, under *Dashboards*, *Browse*, *New Dashboard*.
6. Add a panel (bar icon with plus on the top right)
7. Make sure the correct datasource is chosen.
8. Choose Advanced mode and select the desired cache time. Add your path to the Observations you want to visualize (without the baseURL, but keeping the leading */*) e.g. `/Datastreams(1)/Observations` . It is recommended to order the Observations by phenomenonTime (`?$orderby=phenomenonTime`).
9. The fields must be mapped manually to the axes. Select *Manually assign fields*. For the time axis select `Type:time` and assign it to `$.value[*].phenomenonTime` . For the value select `Type:number` and assign it to `$.value[*].result` .

obrázek č. 29 – Návod pro konfiguraci Grafana pluginu pro vizualizace FROST Serveru[29]

Zdroj: https://grafana.com/grafana/plugins/iosb-sensorthings-datasource/

I přes své nevýhody se ze tří certifikovaných řešení jeví FROST Server jako nejvhodnější právě díky své otevřenosti a dostupnosti. GOST server je taktéž zdarma, ale bohužel je ve fázi začínající implementace, zatímco FROST Server již disponuje funkční a dokončenou implementací SensorThings API standardu.

4.3 Analýza databázových systémů využívající časové řady

Jelikož byla na základě předchozí analýzy vybrána implementace FROST Server byla k persistenci dat v relačních databázích vybrána jeho základní databáze a to PostgreSQL. Dále je na základě analýzy a komparace vybrán databázový systém využívající časové řady.

Databáze využívající časové řady jsou vhodné pro data z IoT sítí z důvodu přirozeného formátu těchto dat, který primárně obsahuje časová razítka. Databáze využívající časové řady jsou nejen schopné pojmout velký objem dat, ale primárně ukládají data na základě právě časových razítek, zároveň jsou i přes velký objem dat schopné rychle vyhledávat a dotazovat se nad takto formátovanými daty.

Podle serveru Medium [111] jsou první čtyři nejpopulárnější databázové systémy využívající časové řady: InfluxDB, Kdb+, Prometheus a Graphite, podrobnější přehled lze nalézt na obrázku č. 30.

RANK	DBMS	SCORE		
DEC 2020		DEC 2020	24 MOS ▲	12 MOS ▲
1	**InfluxDB**	**26.15**	**+10.79**	**+5.01**
2	Kdb+	7.66	+2.46	+2.16
3	Prometheus	5.75	+3.41	+1.67
4	Graphite	4.68	+1.82	+1.33
5	RRDtool	3.30	+0.69	+0.52
6	TimescaleDB	2.98	+2.20	+1.06
7	ApacheDruid	2.59	+1.18	+0.72
8	OpenTSDB	2.44	+0.31	+0.47
9	FaunaDB	2.01	+1.71	+1.21
10	GridDB	0.82	+0.60	+0.29

obrázek č. 30 – Přehled nejpopulárnějších databází využívajících časové řady v roce 2020[30]

Zdroj: https://miro.medium.com/max/640/1*eCBDX1Lc6Bo6GI-9fzHb1w.webp

4.3.1 InfluxDB

InfluxDB [89] je databázový systém využívající časové řady s otevřeným kódem. Je spravován firmou InfluxData [112], která byla založena v roce 2012. Databáze se velmi jednoduše používá a má více možností použití ať už jako plug-in bez nutnosti dalšího programování, nebo je možné provolávat přímo její API za pomoci klientských knihoven

pro různé programovací jazyky. Pro dotazování nad databází se pak používá jazyk Flux, což je skriptovací jazyk podobný ve své jednoduchosti používání např. Pythonu.

InfluxDB má k dispozici kvalitní dokumentaci jak pro instalaci, tak i pro podporu systému. Je možné ji nainstalovat lokálně i do cloudu. Databázi lze snadno nasadit ve formě Docker kontejneru nebo Kubernetes podu za pomoci jednoduchých příkazů.

Databázi lze provozovat zadarmo i v placené verzi InfluxDB Cloud, kde se jedná o hostující službu v cloudu. U cloudové služby je možné si vybrat mezi různými placenými plány. Buď lze platit podle využívaného množství bytů dat nebo podle množství zaslaných dotazů.

Co se týče podpory je k dispozici velmi dobře sepsaná dokumentace a aktivní komunita uživatelů. Pro placenou verzi je pak k dispozici i oficiální podpora přímo od společnosti InfluxData.

4.3.2 Kdb+

Kdb+ [113] je spravována firmou KX [114], která byla založena v roce 1993. Kdb+ je možné využívat skrz mnoho rozhraní, jedním z nich může být tzv. ODBC (Open Database Connectivity) rozhraní. K dispozici jsou taktéž klientské knihovny v různých programovacích jazycích. Dotazovací jazyk využívaný v Kdb+ se nazývá q langunage, který se automaticky instaluje společně s databázovým systémem. Q language je velmi podobný SQL. Pokud se Kdb+ nainstaluje a spustí lokálně tak pracuje s tímto dotazovacím jazykem.

Co se týče dokumentace Kdb+ a jejího využití v aplikačním kódu tak zde bohužel není tak dobře sepsaná dokumentace jako např. pro InfluxDB a je nutné více hledat na internetu a v komunitě. K dispozici jsou pouze klientské knihovny pro C# a Python jinak je nutné využít ODBC připojení.

K instalaci samotné databáze neexistuje dostatečná dokumentace, i když sama o sobě není náročná. Kdb+ lze využívat v cloudových službách, ale pouze jako virtualizaci, a ne jako kontejner.

Databáze je zdarma pouze pro osobní využití, pro komerční využití je nutné platit za licenci, kde k této variantě není nikde veřejná oficiální informace o tom, kolik daná licence stojí. Jednou z mála výhod u placené verze Kdb+ je přístup k velkému týmu s podporou.

4.3.3 Prometheus

Prometheus [115] je jedním z nejpopulárnějších databázových systémů využívajících časové řady v prostředí Kubernetes. Je spravován organizací Cloud Native Computing Foundation [116] založenou v roce 2015. Je to de facto standard pro monitorování v prostředí Kubernetes. Promethus lze využívat za pomoci dotazovacího jazyka PromQ, anebo lze taktéž využít jednu z jejich klientských knihoven v programovacích jazycích jako je Go, Python, Rust, ale taktéž knihovny vyvíjené třetími stranami pro např. C#, Node.js nebo PHP.

Prometheus nemá placenou verzi, neexistuje oficiální hostovaná služba od vývojářů zodpovídajících za tuto službu, ale je obecně dostupný v mnoha existujících cloudových službách jako je Google Cloud Platform, Microsoft Azure nebo Amazon Web Services.

Prometheus není tak snadné nainstalovat jako např. InfluxDB, v Kubernetes sice je k dispozici automaticky, ale jinak je nutné stáhnout binární verzi a z té nainstalovat databázi. Ta se musí konfigurovat skrz konfigurační soubor s názvem prometheus.yml, pro který je nutné mít znalosti skriptovacího jazyka YAML, než je možné systém spustit.

Oficiální placená podpora pro Prometheus neexistuje. Na základě populárnosti tohoto databázového systému alespoň existuje velká uživatelská komunita. Díky které by nemělo být náročné najít odpověď na položenou otázku.

4.3.4 Graphite

Graphite [117] je databázový systém využívaný velkými firmami jako je GitHub, Reddit nebo Lyft. Byl vyvinutý firmou Orbtiz [118], která byla založena v roce 2001. Graphite je silný databázový nástroj soustředěný na spolehlivý běh na jakémkoliv dostupném hardwaru.

Bohužel pro tento databázový systém chybí dostatečně dobře strukturovaná dokumentace. Samotná dokumentace poskytuje velké množství informací, ale brzy začíná být matoucí, jelikož v ní chybí dostatečná struktura. Dále pro tento systém chybí oficiální dotazovací jazyk a lze ovládat skrz plaintext, pickle nebo AMQP protokol.

Jedna z výhod Graphite je jeho snadná instalace skrz kontejnerizaci, pokud ale není zvolen tento přístup pak je Graphite možné nainstalovat skrz příkazový řádek např. za pomoci pip, virtualenv, Synthesize nebo REsynthesize.

Graphite existuje podobně jako Prometheus pouze v neplacené formě a neexistuje zde žádná oficiální placená podpora. V případě problémů nebo dotazů je nutné se obrátit na komunitu využívající tento systém, která ale bohužel není příliš aktivní.

4.3.5 Komparace analyzovaných databázových systémů

V následující tabulce č. 5 je graficky zobrazena komparace jednotlivých popsaných systémů.

Jméno DB	Kvalitní dokumentace	Snadnost používání	Snadnost instalace	Placená podpora
InfluxDB	Ano	Ano	Ano	Ano
Kdb+	Ne	Ne	Ne	Ano
Prometheus	Ne	Ano	Ne	Ne
Graphite	Ne	Ne	Ano	Ne

tabulka č. 5 – Komparace analyzovaných databázových systémů využívajících časové řady[5]

Zdroj: Vlastní práce

Co se týče výsledků dané komparace je téměř okamžitě jisté, která z vybraných technologií má nejlepší výsledky. U Kdb+ a Prometheus je nutné zkonstatovat, že k nim existuje dokumentace, ale v porovnání s InfluxDB není tak kvalitní. Co se týče snadnosti používání tak zde vítězí i Prometheus, ale absentuje zde snadnost instalace mimo Kubernetes prostředí a placená podpora oproti InfluxDB.

4.4 Implementace prototypu

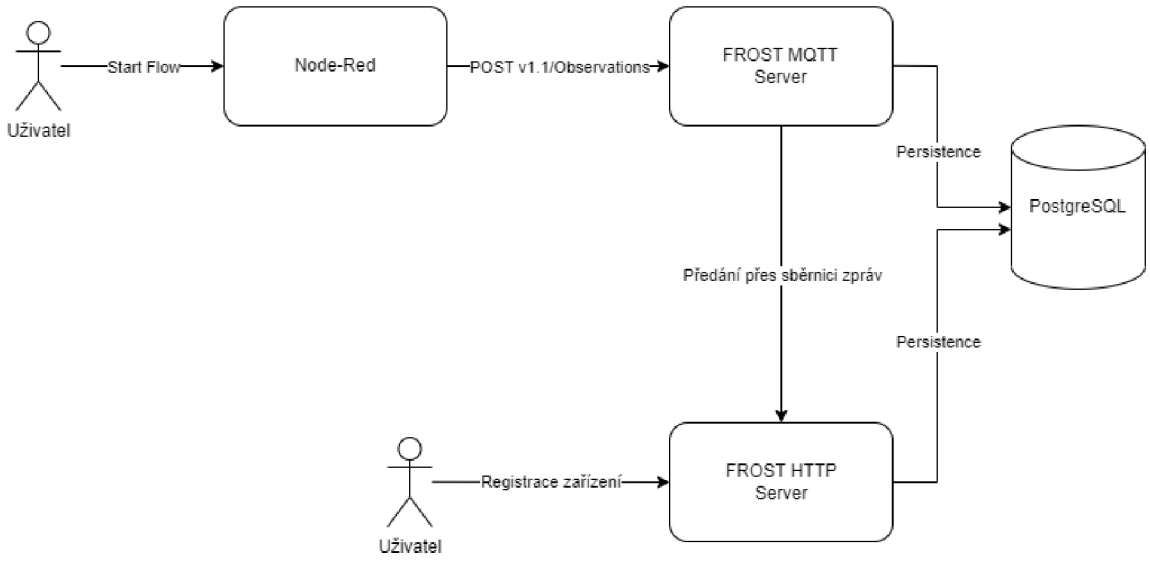
Základní forma prototypu se skládá z těchto částí:

- FROST HTTP Server
- FROST MQTT Server
- PostgreSQL databáze
- Node-Red Server

Základem zprovoznění jakékoliv funkční platformy je konfigurace jednotlivých služeb, konfigurace pro nasazení a spuštění služeb. Jednotlivé sekce těchto kroků byly popsány v následujících kapitolách.

4.4.1 Architektura prototypu

Na obrázku č. 31 lze vidět základní diagram architektury implementovaného prototypu.



obrázek č. 31 – Diagram implementovaného prototypu[31]

Zdroj: Vlastní práce

Základní komunikace v protokolu je řízená přes REST API a MQTT brokera. Obě služby v podobě MQTT brokera i HTTP serveru jsou schopné persistovat data do společné RDBMS databáze PostgreSQL.

Přijaté požadavky nebo zprávy jsou mezi službami předávány za pomoci sběrnice zpráv. Node-Red server je v prototypu využívaný pro abstrakci schopnosti rozlišovat zprávy z různých zařízení a rozlišovat i různé typy zpráv z těchto zařízení. Které jsou následovně předávány do platformy za účelem persistence a dalšího zpracování.

4.4.2 Poskytnutá senzorová data

Data ke zpracování byla poskytnuta na základě spolupráce univerzity a pana doktora Vojtěcha Nováka, který byl ochotný poskytnout data, které zpracovává v komerční i univerzitní sféře.

Zdrojem jednotlivých zpráv byla IoT zařízení využívající technologie LoRaWAN běžící v LPWAN síti. Byla využita zařízení Abeeway Industrial Tracker [119]. Z těchto zařízení, pak byla exportována data za časové období prosinec 2022 – březen 2023 ve formě JSON souborů, viz. obrázek č. 32.


```
{
  "results": [
    {
      "statement_id": 0,
      "series": [
        {
          "name": "abeewayIndustryTracker",
          "columns": [
            "time",
            "appState",
            "batteryLevel",
            "batteryStatus",
            "bleFirmwareVersion",
            "dynamicMotionState",
            "eui",
            "f_accuracy",
            "f_ackToken",
            "f_appState",
            "f_batteryLevel",
            "f_batteryStatus",
            "f_bleFirmwareVersion",
            "f_data",
            "f_dynamicMotionState",
            "f_eui",
            "f_eventType",
            "f_fcnt",
            "f_firmwareVersion",
            "f_geoType",
            "f_gpsOnRuntime",
            "f_gpsStandbyRuntime",
            "f_lat",
            "f_lon",
            "f_messageType",
            "f_periodicPosition",
            "f_resetCause",
            "f_sosFlag",
            "f_source",
            "f_temperatureMeasure",
            "f_trackingMode",
            "f_xOrientation",
            "f_yOrientation",
            "f_zOrientation",
            "fcnt",
            "firmwareVersion",
            "geoType",
            "messageType",
            "periodicPosition",
            "sosFlag",
            "source",
            "trackingMode"
          ],
          "values": [
            [
              "2022-12-29T17:38:20.355Z",
              "1",
              "71",
              "OPERATING",
              null,
              "STATIC",
              "20635F0134001777",
              null,
              10,
              1,
              71,
              "OPERATING",
              null,
              "032a4766a9191071b351b961c150e03937c528b91459c03a2b78b360a4b7c250d6aa",
              "STATIC",
              "20635F0134001777",
              null,
              127350,
              null,
              null,
              null,
              null,
              null,
              null,
              "POSITION_MESSAGE",
              true,
              null,
              null,
              null,
              7.6,
              "MOTION_TRACKING",
              null,
              null,
              null,
              null,
              null,
              null,
              "POSITION_MESSAGE",
              "true",
```


obrázek č. 32 – Ukázka poskytnutých senzorových dat ve formátu JSON[32]

Zdroj: Vlastní práce

Za účelem snadnějšího zpracování dat ve službě Node-Red [120] byl vytvořen jednoduchý Python program, který převedl poskytnuté JSON soubory do snadněji konzumovatelných CSV souborů. Ukázka Python implementace lze vidět na obrázku č. 33.

```python
import json
import csv


def process_csv_json(file_name):
    json_file = open(file_name)

    exported_csv_json = json.load(json_file)

    results = exported_csv_json['results'][0]
    series = results['series'][0]

    columns = series['columns']
    list_of_rows = []

    for i, values in enumerate(series['values']):
        row = {}

        for j, value in enumerate(values):
            row['name'] = series['name']
            row[columns[j]] = value

        row['index'] = i + 1
        list_of_rows.append(row)

    location_index = 1
    for i in list_of_rows:
        if i['f_geoType'] == 'WIFI_BSSIDS_WITH_NO_CYPHER' or i['f_geoType'] == 'GPS':
            i['location_index'] = location_index
            location_index = location_index + 1

    csv_file = open('export_A.csv', 'w', newline='')
    headers = ['name']

    for i in series['columns']:
        headers.append(i)

    headers.append('index')
    headers.append('location_index')

    with csv_file:
        writer = csv.DictWriter(csv_file, fieldnames=headers)

        writer.writeheader()

        for i in list_of_rows:
            writer.writerow(i)

    json_file.close()


if __name__ == '__main__':
    process_csv_json('/Users/annadrevikovska/Downloads/export_A.json')
```

obrázek č. 33 – Ukázka implementace Python programu pro převedení JSON dat do CSV[33]

Zdroj: Vlastní práce

Konkrétní zpracování jednotlivých formátů zpráv je pak více popsáno v kapitole č. 4.4.4.

4.4.3 Konfigurace prototypu

Konfigurace prototypu spočívala především v přípravě tzv. Docker compose souboru. Docker compose soubor obstarává stažení jednotlivých Docker images, nastavení vystavení

portů služeb a v neposlední řadě proměnných pro běžící prostředí. Docker images jsou vytvořené z Dockerfile souborů, které instalují a konfigurují knihovny a programy nutné pro běh jednotlivých Docker kontejnerů.

Pro poskytnutá data bylo nutné zvýšit velikost fronty a množství použitých vláken určených pro předávání a přijímání zpráv na sběrnici zpráv komunikující mezi FROST HTTP a MQTT servery. Ukázku Docker compose konfigurace lze vidět znázorněné na obrázku č. 34.

```
version: '3'

services:
  web:
    hostname: frost-server
    image: fraunhoferiosb/frost-server-http:2.0
    ports:
      - 8080:8080
    depends_on:
      - database
      - mosquitto
    environment:
      - serviceRootUrl=http://localhost:8080/FROST-Server
      - queueLoggingInterval=1000
      - plugins.multiDatastream.enable=true
      - http_cors_enable=true
      - http_cors_allowed_origins=*
      - bus_mqttBroker=tcp://mqtt-server:1883
      - bus_sendWorkerPoolSize=100
      - bus_sendQueueSize=100000
      - bus_recvWorkerPoolSize=200
      - bus_recvQueueSize=100000
      - bus_maxInFlight=100000
      - persistence_db_driver=org.postgresql.Driver
      - persistence_db_url=jdbc:postgresql://database:5432/sensorthings
      - persistence_db_username=sensorthings
      - persistence_db_password=qX69cyhtza3MZUF6WrFrte!NcQC88cnq
      - persistence_autoUpdateDatabase=true
```

obrázek č. 34 – Ukázka Docker compose konfigurace[34]

Zdroj: Vlastní práce

Dále bylo na základě doporučeného nastavení změněno heslo k PostgreSQL databázi. Pro možnost lokálního napojení se na databázi přes nástroj DataGrip byl veřejně vystaven port 5432. V konfiguračním souboru pro MQTT brokera Mosquitto, pak byl ponechán základní port 1883 a byl zakázán náhodný přístup, na zakládě nastavení konfiguračních hodnot *listener* a *allow_anonymous*.

4.4.4 Konfigurace RDBMS databáze

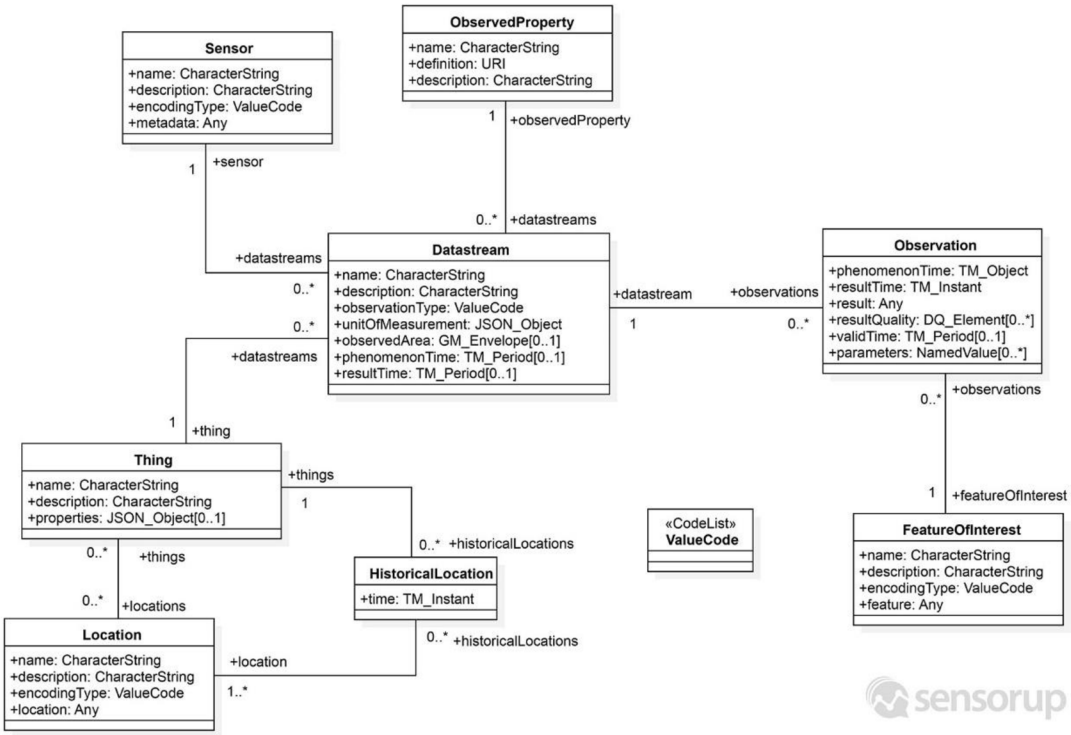
Konfigurace RDBMS databáze proběhla ve formě poskytných Liquibase [121] migrací, které je možné, po nasazení FROST HTTP serveru, spustit přes HTTP POST

požadavek. Požadavek má následující MIME formát: *application/x-www-form-urlencoded*, jedná se o odeslání formuláře, který je předvyplněný a poskytnutý serverem za pomoci servletu. Na obrázku č. 35 lze vidět webovou stránku poskytující možnost nastavení databáze.

obrázek č. 35 – Ukázka servletu poskytujícího spuštění migrací v databázi[35]

Zdroj: Vlastní práce

Zmíněné Liquibase migrace provedly základní nastavení tabulek v databázi. Seznam vytvořených tabulek ve formě UML diagramu lze vidět na obrázku č. 36.



obrázek č. 36 – SensorThings API UML diagram[36]

Zdroj: https://developers.sensorup.com/assets/images/STA-1.0-UML.jpg

Po úspěšném dokončení konfigurace databáze bylo nutné naplnit databázi základními daty. Vložením základních dat je v podstatě simulována část životního cyklu IoT, který není pokrytý touto prací, a to je automatizace registrace jednotlivých zařízení. Na základě schématu entit v SensorThings API standardu je nutné v následujícím pořadí vytvořit tyto entity:

1. *POST v1.1/Things* – jednotlivá zařízení v IoT síti
2. *POST v1.1/Sensors* – definice senzorů reprezentujících IoT zařízení
3. *POST v1.1/FeaturesOfInterest* – poskytnutí prostorové informace o dané IoT síti
4. *POST v1.1/ObservedProperties* – definice měřených hodnot pro jednotlivé proudy dat
5. *POST v1.1/Datastreams* a *POST v1.1/MultiDatastreams* – definice jednotlivých proudů dat pro zařízení, proudy dat mohou ve svých pozorováních obsahovat jeden nebo více důležitých ukazatelů

Následující JSON soubory představují ukázku dat potřebných k vytvoření jednotlivých zařízení, senzorů, prostorových informací, definicí měřených hodnot a definicí proudových dat nutných pro možné zpracování poskytnutých dat v základním prototypu.

Ukázka JSON dat pro požadavek na vytvoření zařízení:

```
{
      "name": "Industrial Tracker",
      "description": "Multi-mode tracker with support for geolocation
and embedded sensors",
      "properties": {
            "sensorManufacturerId": "20635F0181000B81"
      }
}
```

Ukázka JSON dat pro požadavek na vytvoření senzoru:

```
{
      "name": "Abeeway Industrial Tracker",
      "description": "Multi-mode tracker with support for geolocation
and embedded sensors",
      "encodingType": "application/pdf",
      "metadata": "https://www.abeeway.com/wp-
content/uploads/2021/04/Abeeway_Products_Industrial_10.pdf"
}
```

Ukázka JSON dat pro požadavek na vytvoření prostorových dat sítě:

```
{
      "name": "PEF KIT CZU",
      "description": "Czech Univerzity of Life Sciences Prague,
Department of Information Technologies",
      "encodingType": "application/vnd.geo+json",
      "feature": {
            "type": "Point",
            "coordinates": [14.3733,50.1309]
      }
}
```

Ukázka JSON dat pro požadavek na vytvoření měřených hodnot:

```
{
      "name": "Temperature",
      "description": "Detected temperature in degress celcius.",
      "definition": "https://qudt.org/vocab/unit/DEG_C"
}
```

Ukázka JSON dat pro požadavek na vytvoření proudových dat:

```
{
      "name": "Temperature measurement",
      "description": "Detected temperature in degress celcius.",
      "observationType":
"http://www.opengis.net/def/observationType/OGC-
OM/2.0/OM_Measurement",
      "unitOfMeasurement": {
            "name": "Temperature",
            "symbol": "degC",
            "definition": "https://qudt.org/vocab/unit/DEG_C"
      },
      "Thing": {
            "@iot.id": 1
      },
      "ObservedProperty": {
            "@iot.id": 7
      },
      "Sensor": {
            "@iot.id": 1
      }
}
```

Pro korektní zpracování dat bylo nutné definovat měřené hodnoty a proudová data pro následující formáty zpráv:

- Frame Pending – vyžádání si následující zprávy od Gateway v IoT síti
- Energy Status – informace určené pro výpočet spotřeby energie zařízení na základě energetické aktivity GPS modulu
- Heartbeat – poskytnutí informace o posledním nutném resetu zařízení
- Temperature – naměřená teplota
- Position – GPS koordinace
- Shock Detection – notifikace v případně překročení nastavených hodnot pro přetížení
- Event – notifikace o změně u zařízení, podle nastavení buď geolokace nebo zpráva o začátku a ukončení pohybu zařízení
- Configuration – vyžádání nastavených konfiguračních parametrů zařízení
- Battery Status – současná hodnota a stav baterie zařízení
- Activity Status – měření aktivity zařízení v určitém časovém úseku

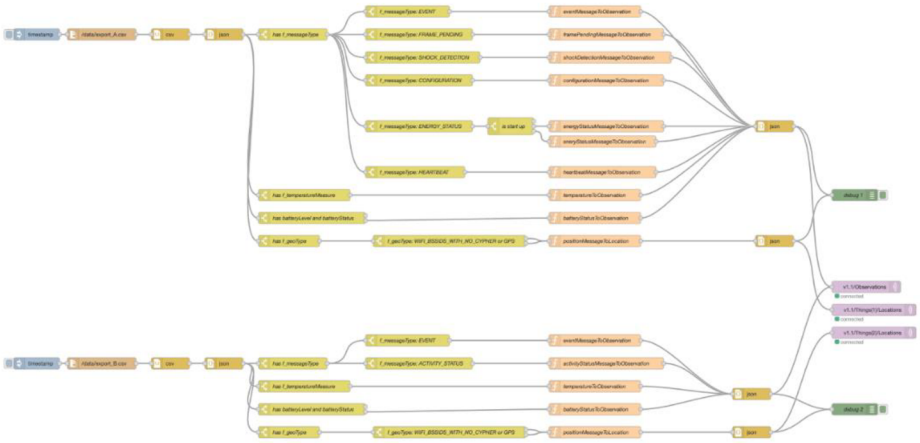
Formát zpráv se pro obě zařízení z části lišil. Zařízení byla rozdělena na zařízení s ID: *20635F0181000B81* a s ID: *20635F0134001777*. Následující formáty zpráv byly zpracovávány pouze zařízením s ID *20635F0181000B81*:

- Frame Pending
- Energy Status
- Heartbeat
- Shock Detection
- Configuration

Zařízezení s ID *20635F0134001777* pak na rozdíl od zařízení s ID *20635F0181000B81* zpracovávalo výhradně formát zpráv Activity Status, které zařízení s ID *20635F0181000B81* neposílalo. Zbylé formáty zpráv byly pro obě zařízení společné.

4.4.5 Integrace s Node-Red

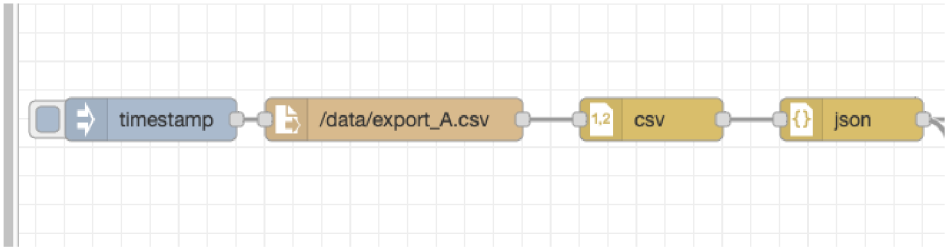
Na obrázku č. 37 lze vidět konfiguraci pro Node-Red flow, který řeší integraci a zpracování dat z CSV generovaného souboru do prototypu.



obrázek č. 37 – Diagram Node-Red flow[37]

Zdroj: Vlastní práce

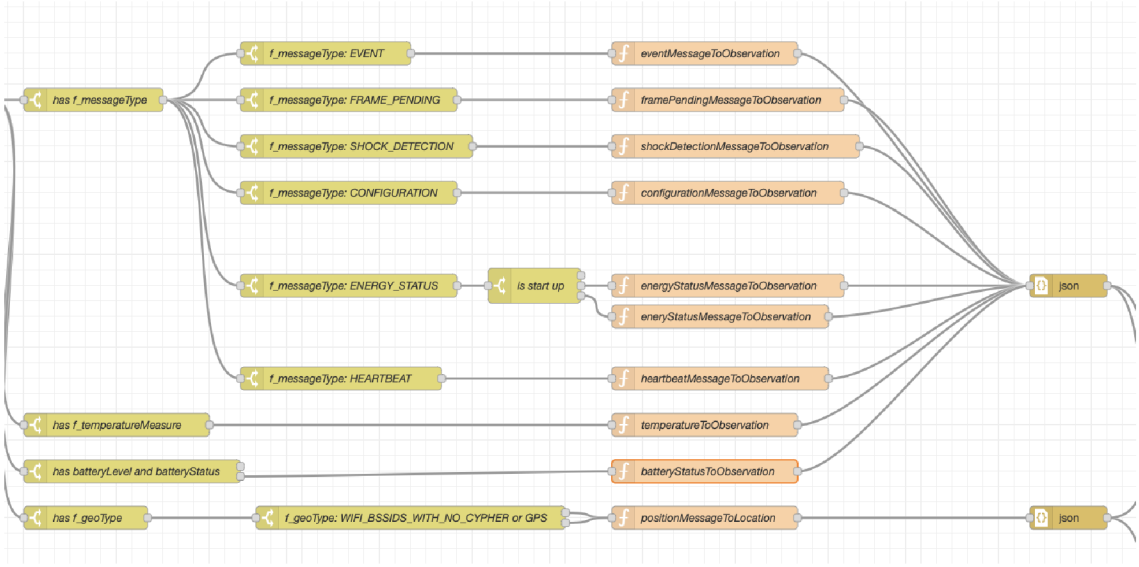
Node-Red umožňuje přečíst celý CSV soubor řádku po řádce a převést jednotlivé zprávy do JSON formátu viz. obrázek č. 38



obrázek č. 38 – Node-Red načte CSV soubor po řádkách a převede je do JSON objektů[38]

Zdroj: Vlastní práce

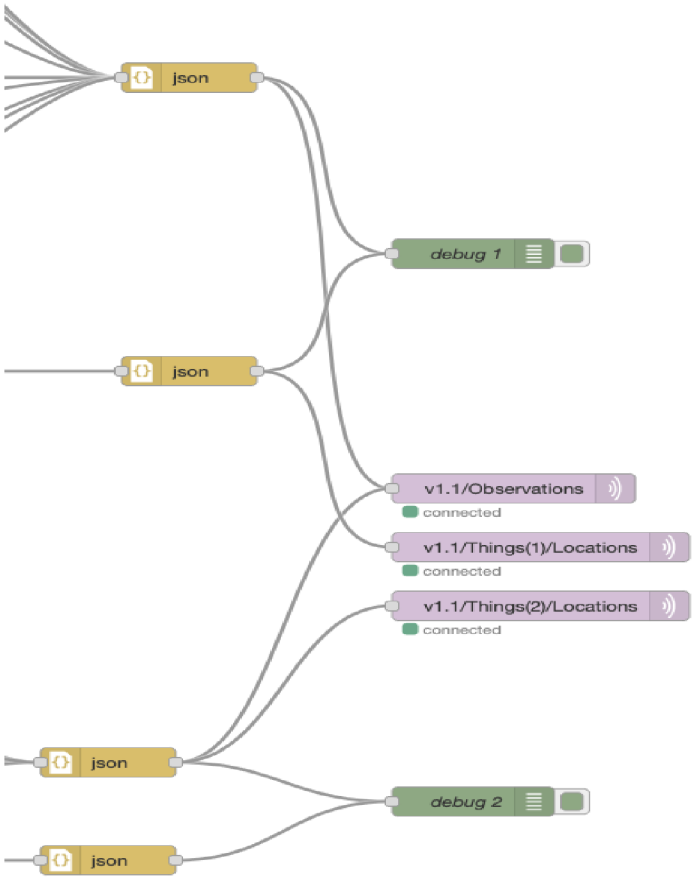
a následně filtrovat jednotlivé typy zpráv, tak, aby byly správně publikovány skrz MQTT brokera, viz. obrázek č. 39.



obrázek č. 39 – Filtrování podle typu zpráv v Node-Red[39]

Zdroj: Vlastní práce

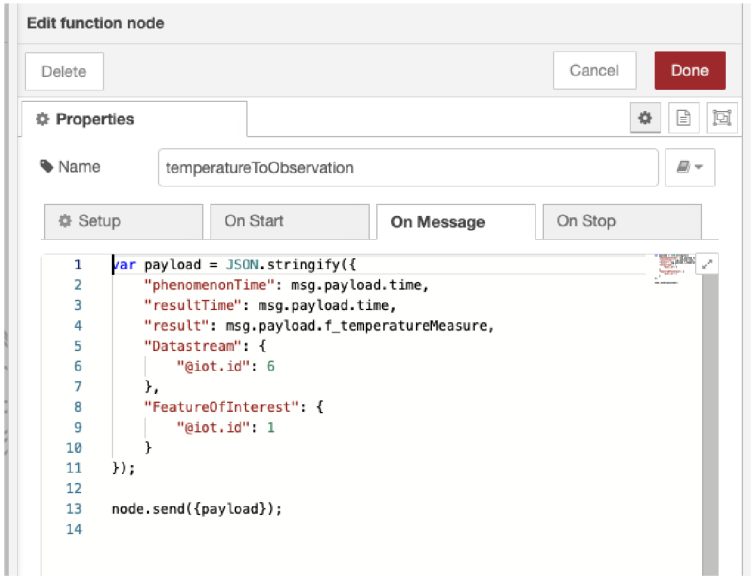
Na obrázku č. 40 lze vidět část flow, která publikuje zprávy na MQTT brokera.



obrázek č. 40 – Publikování zpráv v Node-Red na FROST MQTT brokera[40]

Zdroj: Vlastní práce

Obrázek č. 41 zobrazuje JavaScript funkci, která řeší vybrání správných dat a jejich předání ve správném formátu do prototypu.



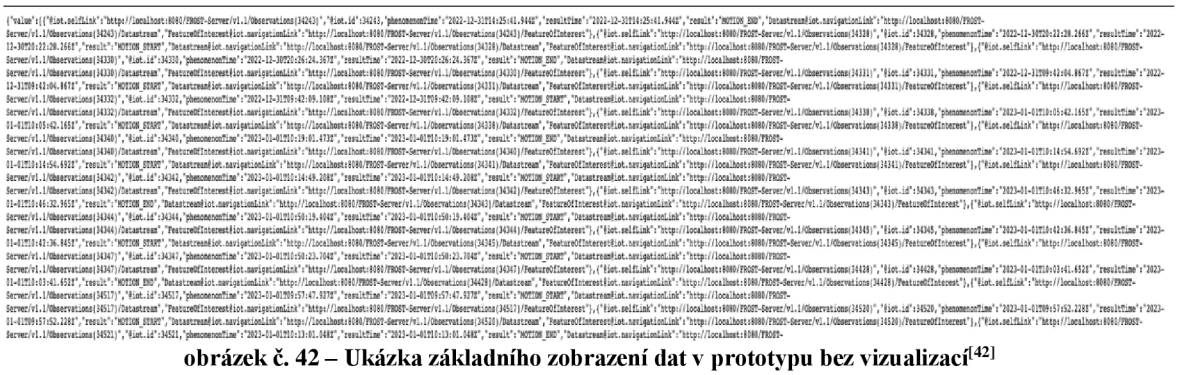
obrázek č. 41 – Node-Red JavaScript funkce, která formátuje zprávu do formátu FROST serveru[41]

Inject funkce (timestamp) v Node-Red umožňuje odstartovat celou flow pro zpracování poskytnutých dat do platformy. Přečte CSV soubor, převede ho na JSON objekty pomocí Switch funkcí, pak roztřídí jednotlivé zprávy podle jejich formátu a předá nově zformátované zprávy MQTT brokerovi k persistenci.

4.4.6 Ukázka běhu základního prototypu

Jak již bylo zmíněno v kapitole č. 4.2.3 FROST Server nemá k dispozici vizualizace persistovaných dat a ani funkční integraci, která by vizualizaci umožňovala.

Následující obrázek č. 42 poukazuje na velmi omezený a nepřehledný způsob jakým lze zobrazit persistovaná data ve webovém prohlížeči, a to ve formě neformátovaných JSON odpovědí od serveru.



obrázek č. 42 – Ukázka základního zobrazení dat v prototypu bez vizualizací[42]

Zdroj: Vlastní práce

4.4.7 Rozšíření implementace prototypu

Na základě požadavků z kapitoly č. 4 a analýzy FROST Serveru lze vyhodnotit, že největším nedostatkem této implementace SensorThings API standardu je absence vizualizací dat a nízká možnost integrace s jinými službami v síti Internet.

Dalším podstatným nedostatkem FROST Server implementace je poměrně těžkopádný způsob, jakým jsou ukládána prostorová data. FROST Server umožňuje ukládat lokace zařízení spolu s časem.

Podstatným problémem řešení FROST Serveru je ale fakt, že čas, který se ukládá s danou lokací je vždy současný čas. SensorThings API standard neumožňuje doplnit skutečný čas pozorování přímo při vytváření dané lokace. Pro doplnění skutečného času je nutné vytvořit požadavek proti zdroji *PATCH v1.1/HistoricalLocations*, který umožňuje zpětnou změnu času pozorované lokace. Toto může být potencionálně problematické v případě velkého množství dat a lokací dostupných v krátkém časovém období na zařízení. Jelikož by bylo nutné u velkého množství dat ručně upravovat hodnoty.

Proto byly vybrány následující dvě rozšíření základní implementace:

- Přidání podpory pro persistenci dat lokací do databáze využívající časové řady InfluxDB
- Doplnění prototypu o integraci se službou Grafana za účelem vizualizací

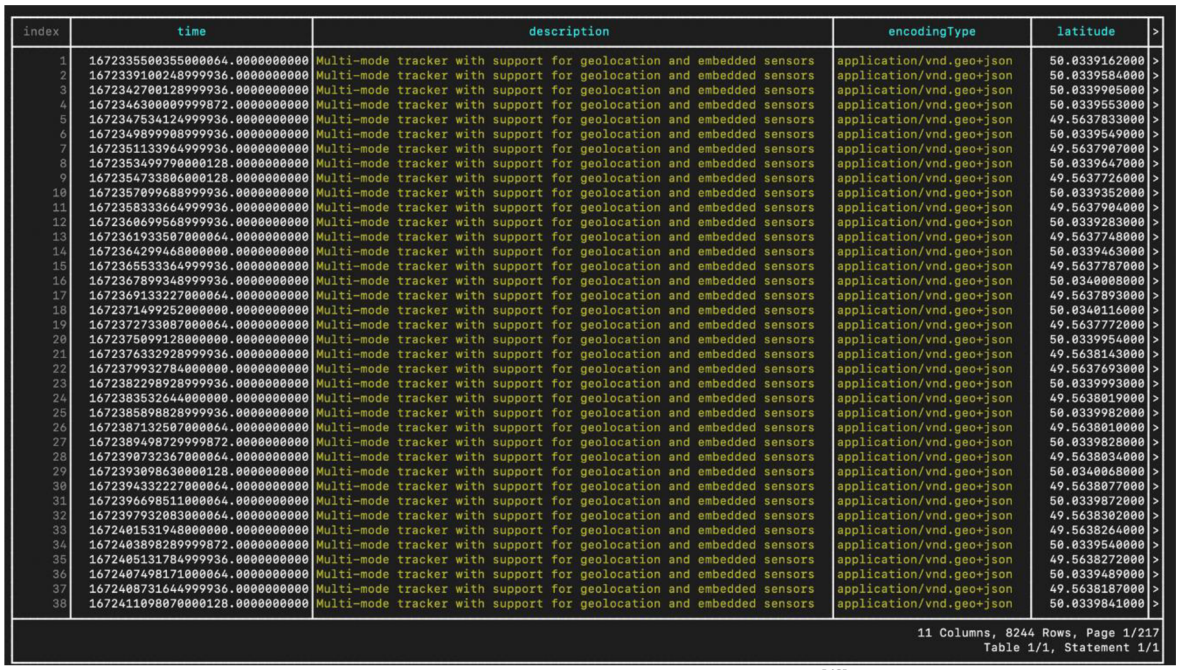
Integrace s InfluxDB spočívala především v implementaci jednoduché Spring Boot Java aplikace, která využila MQTT klienta pro asynchronní zpracovávání zpráv z MQTT brokera. Byl použit MQTT broker, který je poskytován FROST serverem.

Naimplementovaná integrace při přijetí zprávy od brokera persistovala data do InfluxDB v následujícím formátu:

- sensorId: Tag Key
- locationOgcId: Tag Key
- thingId: Tag Key
- name: Tag Key
- description: Tag Key
- longitude: Field Key
- latitude: Field Key
- type: Field Key

- encodingType: Field Key

Obrázek č. 43 demonstruje persistovaná data v InfluxDB databázi i se skutečnými časy měření poskytnutých v exportovaných datech.



index	time	description	encodingType	latitude
1	1672335500355000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339162000
2	1672339100248999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339584000
3	1672342700128999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339905000
4	1672346300009999872.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339553000
5	1672347534124999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5637833000
6	1672349899908999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339549000
7	1672351133964999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5637907000
8	1672353499790000128.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339647000
9	1672354733806000128.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5637726000
10	1672357099688999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339352000
11	1672358333664999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5637904000
12	1672360699568999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339283000
13	1672361933507000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5637748000
14	1672364299468000000.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339463000
15	1672365533364999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5637787000
16	1672367899348999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0340008000
17	1672369133227000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5637893000
18	1672371499252000000.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0340116000
19	1672372733087000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5637772000
20	1672375099128000000.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339954000
21	1672376332928999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5638143000
22	1672379932784000000.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5637693000
23	1672382298928999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339993000
24	1672383532644000000.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5638019000
25	1672385898828999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339982000
26	1672387132507000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5638010000
27	1672389498729999872.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339828000
28	1672390732367000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5638034000
29	1672393098630000128.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0340068000
30	1672394332227000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5638077000
31	1672396698511000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339872000
32	1672397932083000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5638302000
33	1672401531948000000.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5638264000
34	1672403898289999872.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339540000
35	1672405131784999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5638272000
36	1672407498171000064.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339489000
37	1672408731644999936.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	49.5638187000
38	1672411098070000128.0000000000	Multi-mode tracker with support for geolocation and embedded sensors	application/vnd.geo+json	50.0339841000

11 Columns, 8244 Rows, Page 1/217
Table 1/1, Statement 1/1

obrázek č. 43 – Lokace persistované v InfluxDB[43]

Zdroj: Vlastní práce

Po persistenci dat do InfluxDB následovalo předání zprávy skrz MQTT brokera do tématu *v1.1/Things(1-2)/Locations*, pro persistenci lokace i do PostgreSQL databáze.

Na obrázcích č. 44 a č. 45 lze vidět ukázky Java kódu, který obstarává implementaci přihlášení se k tématu a následnou logiku pro publikování zprávy o vytvořené lokaci do FROST MQTT brokera.

```java
@Async
@EventListener(ApplicationReadyEvent.class)
public void subscribe() {
    final CountDownLatch countDownLatch = new CountDownLatch(10);
    log.info("Starting to accept locations measurement messages.");

    Optional<FrostServerThingsResponse> thingsResponse = frostServerConnector.findAllThings();

    if (thingsResponse.isEmpty()) {
        log.warn("No valid response was fetched for things. Retrying.");

        for (int i = 0; i < 5; i++) {
            thingsResponse = frostServerConnector.findAllThings();

            if (thingsResponse.isPresent()) {
                log.info("Fetched things. Continuing with processing location measurements.");
                break;
            }
        }
    }

    if (thingsResponse.isEmpty()) {
        log.error("Could not fetch valid things response after several retries. Skipping.");
        return;
    }

    final FrostServerThingsResponse things = thingsResponse.get();

    try {
        mqttClient.subscribe(
                LOCATIONS_TOPIC,
                (topic, message) -> processReceivedMeasurementMessage(things.value(), message, countDownLatch)
        );

        countDownLatch.await( timeout: 2, TimeUnit.MINUTES);
    } catch (MqttException | InterruptedException e) {
        log.error("There was an unexpected exception during subscribing to locations measurement topic.", e);
    }
}
```

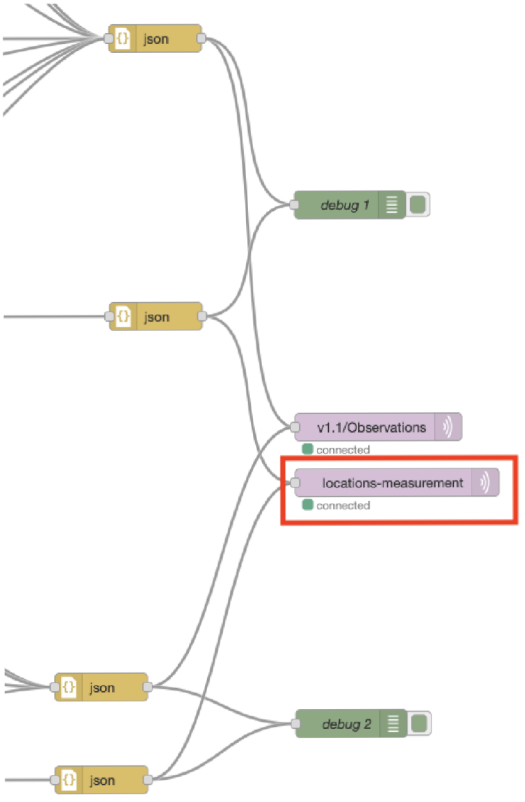
obrázek č. 44 – Implementace pro odebírání zpráv z locations-measurement tématu[44]
Zdroj: Vlastní práce

```java
@Validated
@Slf4j
public class MqttLocationPublisherService {

    private static final String LOCATION_PUBLISH_TOPIC = "v1.1/Things(%s)/Locations";

    private final IMqttClient mqttClient;
    private final ObjectMapper objectMapper;

    public void publish(@Valid PublishLocationCommand command) {
        if (!mqttClient.isConnected()) {
            log.error("MQTT client is currently not connected. Not publishing message for location = {}", command.location().locationId());
            return;
        }

        MqttMessage mqttMessage;

        try {
            mqttMessage = new MqttMessage(
                    objectMapper.writeValueAsString(command.location()).getBytes(StandardCharsets.UTF_8)
            );
        } catch (JsonProcessingException e) {
            log.error("Unexpected error during processing location POJO to JSON string for location = {}.", command.location().locationId(), e);
            return;
        }

        mqttMessage.setQos(1);
        mqttMessage.setRetained(true);

        try {
            mqttClient.publish(String.format(LOCATION_PUBLISH_TOPIC, command.thingId()), mqttMessage);
        } catch (MqttException e) {
            log.error("There was an error during publishing an MQTT message for location with id = {}.", command.location().locationId(), e);
        }
    }
}
```

obrázek č. 45 – Implementace publikování zpráv do v1.1/Things(1-2)/Locations tématu[45]

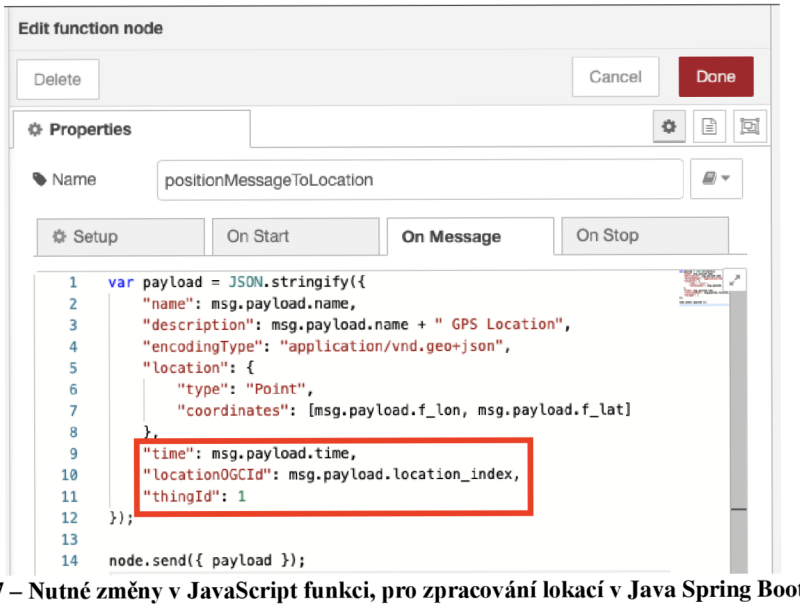
Zdroj: Vlastní práce

Pro rozšíření implementace bylo nutné upravit i flow v Node-Red serveru tak, aby byly zprávy o lokaci předávány implementované instanci Java aplikace namísto přímo FROST Serveru. Obrázek č. 46 a č. 47 ukazují nutné úpravy v daném flow a JavaScript funkcích, které převáděly data z CSV do zpráv.



obrázek č. 46 – Změny v Node-Red flow pro napojení jiného tématu MQTT brokera pro lokace[46]

Zdroj: Vlastní práce



obrázek č. 47 – Nutné změny v JavaScript funkci, pro zpracování lokací v Java Spring Boot aplikaci[47]

Zdroj: Vlastní práce

Rozšíření řešení základního prototypu za účelem poskytnutí vizualizací dat bylo řešeno skrz instanci platformy Grafana, která po připojení zdrojů dat nabízí různé způsoby, jak dotazovaná data vizualizovat v tzv. „dashboardech“. Více o konfiguraci a jednotlivých ukázkách vizualizací v kapitolách č. 4.4.8 a č. 4.4.9.

4.4.8 Konfigurace rozšířeného prototypu

Rozšíření konfigurace spočívalo v povolení možnosti ukládat do RDBSM databáze i řádky s předem specifikovaným id sloupcem jako primárním klíčem společně už s povoleným automatickým generováním hodnot primárních klíčů.

Jednalo se o změnění konfigurace se jmeném *persistence_idGenerationMode* v konfiguraci HTTP serveru a MQTT brokerovi na hodnotu *ServerAndClientGenerated*. Výchozí hodnota této konfigurace je *ServerGeneratedOnly*.

Dále byly do Docker compose konfigurace přidány následující služby v podobě Docker images:

- InfluxDB
- Grafana
- Java Spring Boot aplikace

InfluxDB konfigurace navíc vyžadovala shell příkazy v následujícím pořadí:

1. `docker run --rm influxdb:latest influxd print-config > config.yml`

2. `docker run --name influxdb -d -p 8086:8086 --volume` `pwd`/influxdb2:/var/lib/influxdb2 --volume` `pwd`/config.yml:/etc/influxdb2/config.yml influxdb:latest`

3. `docker exec influxdb influx setup --bucket BUCKET_NAME -org ORG_NAME --password PASSWORD --username USERNAME -force`

Pro další integrace s jinými systémy je předchozí příkazy nutné ještě doplnit o tento příkaz:

```
docker exec influxdb influx auth list | awk -v
username=USERNAME '$5 ~ username {print $4 " "}'
```

Který vypíše token, který je nutné uvést, při autentizaci a autorizaci mezi jednotlivými službami.

Java Spring Boot aplikace nevyžadovala žádné nestandartní konfigurace navíc oproti základním konfiguracím Spring Boot aplikace, které Spring Boot nastavuje automaticky.

Výjimkou byla nutnost specifikovat URL jednotlivých služeb, se kterými bylo nutné v implementaci integrace komunikovat.

4.4.9 Ukázka vizualizací rozšířeného prototypu

Vizualizace byly řešeny v platformě Grafana, která nabízí velké množství vizualizací dat z více datových zdrojů. Vizualizace byly vytvořeny pro lokace a teplotu.

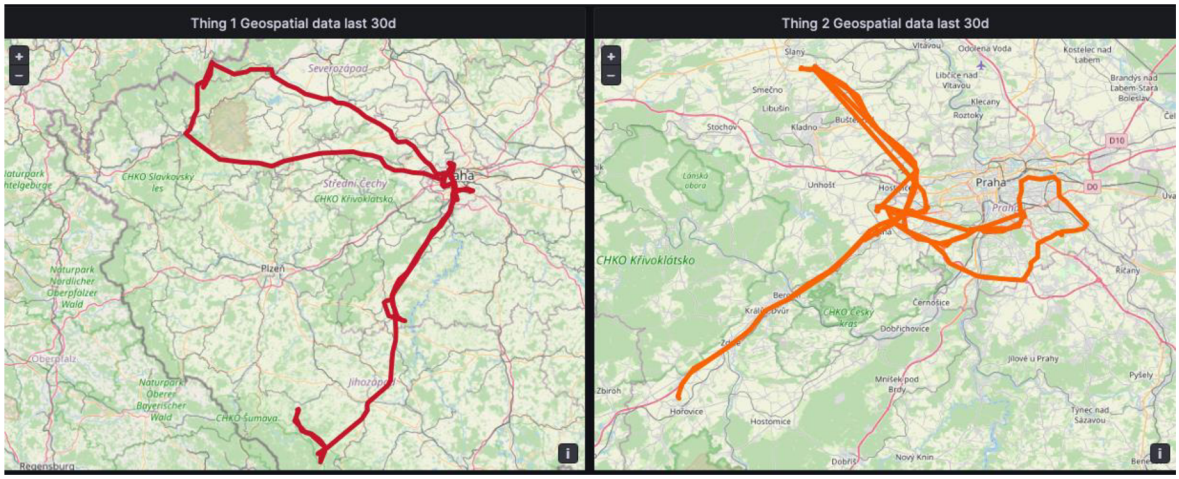
Byla vybrána interní Grafana vizualizace s názvem Geomap, které je nutné předat zdroj dat a vytvořit korektní dotaz vůči databázi. Jako zdroj dat byla vybrána InfluxDB z důvodu korektního předávání časových údajů k jednotlivým lokacím.

Dotaz proti databázi byl vytvořen v jazyce Flux a měl následující formát:

```
import "influxdata/influxdb/schema"
from(bucket: "sensorthings")
  |> range(start: -30d)
  |> filter(fn: (r) => r["_measurement"] == "location")
  |> filter(fn: (r) => r["_field"] == "latitude" or r["_field"] ==
"longitude")
  |> filter(fn: (r) => r["thingId"] == "1")
  |> group()
  |> schema.fieldsAsCols()
```

Dotaz určil, v jakém časovém rozsahu mají být vybrána data, z jakého „measerementu“ mají být vybrána data a pro jaká jednotlivá pole a tagy. Bylo nutné vybrat pole *longitude* a *latitude* a tag *thingId* pro předání GPS koordinací do vizualizace pro konkrétní zařízení.

Na obrázku č. 48 lze vidět výsledek vizualizace s vykreslenou cestou lokací obou zařízení za poslední měsíc.



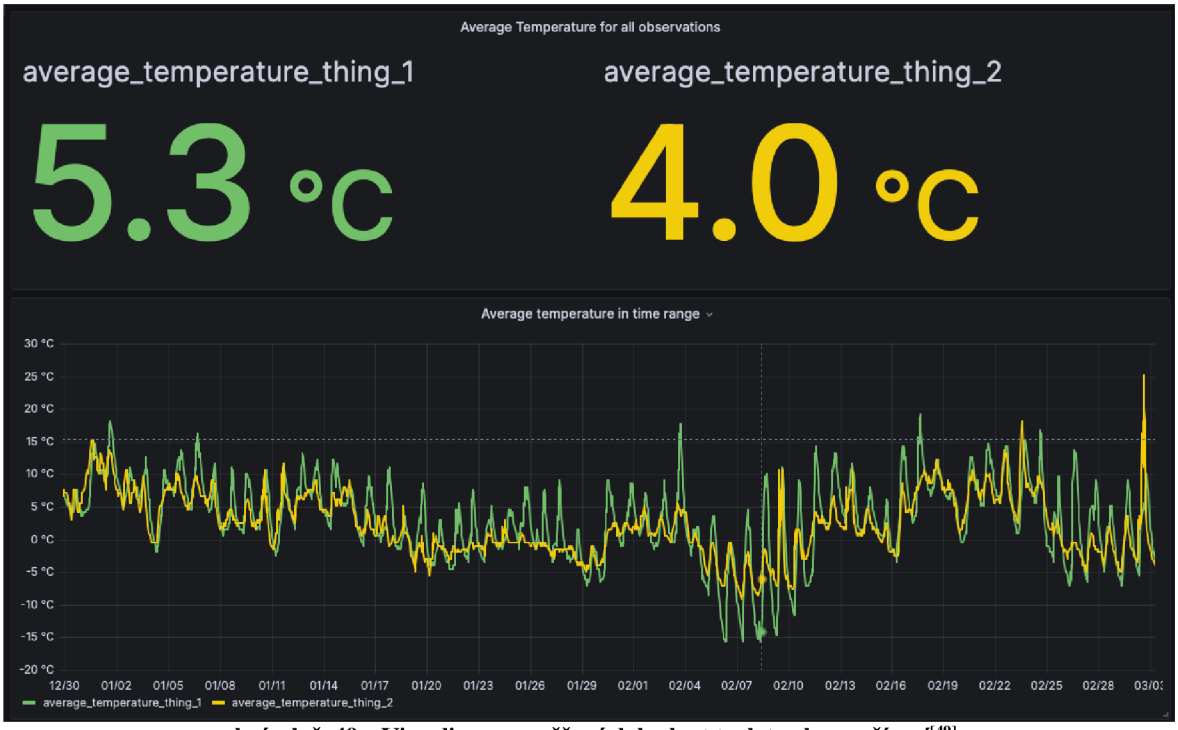
obrázek č. 48 – Vizualizace lokací zařízení za posledních 30 dní[48]

Zdroj: Vlastní práce

U teplot byl jako datový zdroj ponechán výchozí databázový systém FROST Serveru, kdy bylo využito SQL jako dotazovacího jazyka pro výběr dat k zobrazení. Použitý dotaz byl v tomto formátu:

```
SELECT „AVG(„RESULT_NUMBER“) AS average_temperature_thing_1 FROM
„OBSERVATIONS“ WHERE „DATASTREAM_ID“ = 6 GROUP BY „RESULT_TIME“;
```

Ve vizualizaci na obrázku č. 49 lze vidět průměrnou naměřenou teplotu v celkovém časovém období a pro všechna pozorování.



obrázek č. 49 – Vizualizace naměřených hodnot teploty obou zařízení[49]

Zdroj: Vlastní práce

5 Výsledky a diskuse

V praktické části práce byla provedena analýza dostupných implementací SensorThings API standardu za účelem experimentu a analýzy efektivity tohoto standardu pro zpracovávání prostorových dat.

Bylo zjištěno, že z vybraných implementací mají všechny své nedostatky. Hlavními kandidáty na implementaci prototypu za účelem experimentu byly FROST Server a SensorUp, kdy jako vhodnější by byl vybrán SensorUp, pokud by byl dostupný s otevřeným zdrojovým kódem a zdarma.

I přesto, že je SensorUp „vyspělejším“ řešením zmíněného standardu byl nakonec vybrán k dalšímu postupu FROST Server právě díky jeho dostupnosti a otevřenosti řešení.

Provedená analýza zjistila, že hlavními nedostatky tohoto řešení je absence vizualizací zpracovaných dat, náročná integrace s ostatními systémy v sítí Internet a kostrbatý přístup k ukládání prostorových dat.

Následovala implementace vybraného řešení a její rozšíření, které mělo řešit nedostatky základního prototypu. Z analýzy výsledného rozšířeného prototypu vznikly následující poznatky.

Systém zpracovával data ze dvou stejných zařízení, obě zařízení ale zasílala jiné typy zpráv. Bylo zpracováno deset různých typů zpráv:

- Frame Pending – Vyžádání si další zprávy od Gateway v síti
- Event – Událost v zařízení
- Configuration – Požadavek na zaslání konfiguračních hodnot
- Energy Status – Zpráva o energetickém stavu a využívání zařízení
- Shock Detection – Zpráva o nadměrných hodnotách v akcelerometru
- Battery Status – Stav baterie zařízení
- Activity Status – Aktivita zařízení v určitém časovém úseku
- Position – Lokace zařízení
- Temperature – Pozorování okolní teploty
- Heartbeat – Důvod posledního resetu zařízení

Následující tabulka č. 6 zobrazuje množství zpracovaných zpráv pro obě zařízení.

ID zařízení	Typ zprávy	Množství zpracovaných zpráv
20635F0181000B81	Frame Pending	3
	Event	889
	Configuration	34
	Energy Status	54
	Shock Detection	44
	Battery Status	7102
	Position	5905
	Temperature	6972
	Heartbeat	31
20635F0134001777	Event	206
	Battery Status	2653
	Activity Status	60
	Position	2339
	Temperature	2545

tabulka č. 6 – Množství zpracovaných zpráv pro jednotlivá zařízení a typy zpráv[6]

Zdroj: Vlastní práce

Celkem bylo pro zařízení s ID 20635F0181000B81 zpracováno 21034 zpráv a pro zařízení s ID 20635F0134001777 bylo zpracováno 7803 zpráv.

Dále bylo rozšířeno základní řešení o integraci s InfluxDB, kde bylo zpracováno celkem 8244 lokací s časovým razítkem pro obě zařízení.

Na základě těchto zpracování pak byly v platformě Grafana vytvořeny dvě základní vizualizace s dvěma rozdílnými zdroji dat. Pro zdroj prostorových dat byla vybrána InfluxDB, pro svoji schopnost ukládat správné časové údaje a pro teplotu byl vybrán původní zdroj dat, jako demonstrace, že i ze základního řešení bylo možné vytvořit vizualizaci.

Z analýzy těchto výsledků lze vyvést, že standard ve své základní podobě je schopný zpracovávat více rozdílných formátů dat a ukládat, i když poměrně složitým způsobem, i prostorová data. Čeho už základní řešení nebylo schopné byly vizualizace zpracovaných dat a snadná integrace s ostatními systémy.

Pro integraci bylo nutné implementovat rozšíření a následně nastavit vizualizace v další integraci. Toto je do budoucna jedna z největších nevýhod standardu i jeho implementací, pokud nebude bráno v potaz řešení SensorUp.

6 Závěr

Hlavním cílem práce byla analýza a komparace možností sběru, přenosu, integrace a zpracování dat z IoT zařízení. Analýza byla provedena ve formě charakteristiky jednotlivých zmíněných částí.

Na základě analýzy bylo zjištěno několik poznatků, co se týče současných standardů v IoT systémech. Byly zjištěny výhody i nevýhody těchto řešení, současné i budoucí problémy.

Z analýzy vyplívá, že již existuje mnoho technologií, které fungují a podporují současný růst IoT. Největším podnětem pro růst tohoto oboru je především čím dál větší dostupnost a otevřenost hardwarových řešení. Jedná se o mikro počítače a mikrokontrolery, které jsou využívány jak v komerční sféře, tak i například v chytrých domácnostech.

Dále jsou v IoT velmi často využívány již existující a zaběhlé technologie, primárně v sekci přenosu a integrace. Zde jsou často využívány technologie, které již existují řadu let jako je REST a HTTP, MQTT protokoly, XML a JSON pro formátování dat.

Problematikou těchto technologií je fakt, že často nebyly vytvářeny pro zařízení, která nemají primární zdroj napájení a nejsou velmi omezená svými zdroji. Proto je v odborných textech často zmiňováno, že je nutné vyvíjet a vytvářet nové standardy a technologie, které budou dělané přímo na míru IoT technologiím, anebo alespoň upravované pro co nejlepší uplatnění v IoT. Jedním z takových příkladů mohou být LPWAN sítě, Bluetooth Low Energy, nebo například směrovací protokol RPL.

V integraci IoT sítí je momentální problém v nízké schopnosti interoperability mezi jednotlivými systémy a zařízeními, kvůli jejich velké heterogenitě a častému proprietárnímu uzavření koncových uživatelů.

Absentuje společný konsensus nad tím, jakým způsobem by se měli komplexně vyvíjet IoT systémy obecně, a nejen pro konkrétní řešení pro jednoho koncového uživatele.

Existující určité snahy o standardizaci přístupu k IoT systémům, jako například SensorThings API od Open Geospatial Consortia, ale tyto snahy jsou ještě pořád na začátku a chybí zde větší poměr komerční sféry, která by tato řešení aktivně využívala.

Integrace taktéž řeší bezpečnostní problémy a automatizaci rozpoznávání jednotlivých modelů senzorů pro jejich automatickou, ale bezpečnou registraci v jednotlivých IoT systémech. Stále platí, že komplexní řešení tzv. Middleware systémů vyžadují vysokou expertízu a často i velké množství finančních prostředků k investici.

Ani pro zpracování dat z IoT zařízení neexistuje jeden společný pohled, ale zde alespoň existují základní doporučení pro poměrně často se opakující řešení. Doporučení se především týkají, jakým způsobem ukládat a přenášet data v rámci sítí, jestli využívat Cloudových služeb, nebo primárně privátních sítí.

Velmi často se zmiňuje tvz. Edge nebo Fog computing, který kombinuje jak zprocesování dat na „krajích“ privátních sítí a následného zasílání dat, které není nutné mít k dispozici okamžitě a vždy, do Cloudových služeb za účelem dalších analýz.

V předchozí kapitole pak byla shrnuta analýza výsledného experimentu, který spočíval v prototypu, který implementuje standard SensorThings API. Analýza ukázala, že základní prototyp je schopný zpracovávat komplexní a různorodá data a je schopný zpracovávat prostorová data.

Už není schopný vytvářet vizualizace a integrace s jinými systémy, proto bylo představeno řešení v podobě integrace s jednoduchým programem, který řeší zpracování dat do InfluxDB databáze a umožňuje tak snadnější vizualizace dat.

Dalšími potenciálními rozšířeními práce by bylo rozšíření možností vizualizací proudových dat s více než jednou pozorovanou hodnotou. Dále řešení postrádá větší důraz na eventuální konzistenci dat a případnou větší synchronizaci mezi stavem dat v obou systémech.

Přínosem práce pak může být představení standardizovaného IoT systému, který by mohl mít i potenciální přínos v komerční sféře, i na základě úspěchů například SensorUp platformy. Bylo prokázáno, že tento systém je schopný zpracovat různorodá data, a to nejen prostorová a je schopný spojit v jednom systému více formátů zpráv, a že jeho rozšířená forma je schopná vytvářet vizualizace pro prostorová data.

7 Seznam použitých zdrojů

[1] MANYIKA, J. et al. *Disruptive Technologies: Advances that Will Transform Life Business and the Global Economy.* San Francisco, CA, USA:McKinsey Global Instit., 2013. [online]. [cit. 2022-26-03]. Dostupné z WWW: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/disruptive-technologies¨

[2] ASHTON, K. *That "Internet of Things" thing*. RFiD Journal (2009). [online]. [cit. 2022-26-03]. Dostupné z WWW: https://www.rfidjournal.com/that-internet-of-things-thing

[3] MARSH-HUNN, D., TRILLES, A., GONZÁLEZ-PÉREZ, J., TORRES-SOSPEDRA a RAMOS, F. *A Comparative Study in the Standardization of IoT Devices Using Geospatial Web Standards.* In IEEE Sensors Journal, Feb.15, 2021, doi: 10.1109/JSEN.2020.3031315. [online]. [cit. 2022-26-03]. Dostupné z WWW: https://ieeexplore-ieee-org.infozdroje.czu.cz/document/9224992

[4] JAYAVARDHANA, G., RAJKUMAR B., SLAVEN a M. MARIMUTHU, P. *Internet of Things (IoT): A vision, architectural elements, and future directions*. Future Generation Computer Systems, ISSN 0167-739X. [online]. [cit. 2022-26-03]. Dostupné z WWW: https://www.sciencedirect.com/science/article/pii/S0167739X13000241

[5] ALUR, R. et al. *Systems computing challenges in the Internet of Things*. In arXiv:1604.02980, 2016. [online]. [cit. 2022-26-03]. Dostupné z WWW: http://arxiv.org/abs/1604.02980

[6] AL-FUQAHA, A., GUIZANI, M., MOHAMMDI, M., ALEDHARI, M. a AYYASH, M. *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications.* In IEEE Communications Surveys & Tutorials, Fourthquarter 2015, doi: 10.1109/COMST.2015.2444095. [online]. [cit. 2022-26-03]. Dostupné z WWW: https://ieeexplore.ieee.org/document/7123563

[7] TSIATSIS, V., HÖLLER, J. a MULLIGAN. *Internet of Things, Technologies and Applications for a New Age of Intelligence.* 2nd Edition. Elsevier, 2019, ISBN 9780128144350. [cit. 2022-28-03].

[8] POHANKA, T. *Distribuované geodatabáze senzorových dat, základ pro integraci a analýzu.* 2020. [online]. [cit. 2022-28-03]. Dostupné z WWW: https://theses.cz/id/uore1t/pohanka2020.pdf

[9] BERTHELSEN, E. et.al. *The global IoT market opportunity will reach usd 4.3 trillion by 2024.* (2015). [online]. [cit. 2022-28-03]. Dostupné z WWW: https://machinaresearch.com/news/the-global-iot-market-opportunity-will-reach-usd43-trillion-by-2024/

[10] ELSAADANY, M., ALI, A. a HAMOUDA, W. *Cellular LTE-A Technologies for the Future Internet-of-Things: Physical Layer Features and Challenges.* In IEEE Communications Surveys & Tutorials, Fourthquarter 2017, doi: 10.1109/COMST.2017.2728013. [online]. [cit. 2022-26-03]. Dostupné z WWW: https://ieeexplore.ieee.org/document/7983347

[11] LIU, X. a ANSARI, N. *Toward Green IoT: Energy Solutions and Key Challenges.* In IEEE Communications Magazine, March 2019, doi: 10.1109/MCOM.2019.1800175. [online]. [cit. 2022-29-03]. Dostupné z WWW: https://ieeexplore.ieee.org/document/8664000

[12] POPLI, S., JHA, R.K. a JAIN, S. *Green IoT: A Short Survey on Technical Evolution & Techniques.* Wireless Pers Commun (2022). https://doi-org.infozdroje.czu.cz/10.1007/s11277-021-09142-3. [online]. [cit. 2022-29-03]. Dostupné z WWW: https://link.springer.com/article/10.1007/s11277-021-09142-3

[13] HERNÁNDEZ-MORALES, C.A., LUNA-RIVERA, J.M. a PEREZ-JIMENEZ, R. *Design and deployment of a practical IoT-based monitoring system for protected cultivations, Computer Communications*. ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2022.01.009. [online]. [cit. 2022-29-03]. Dostupné z WWW: https://www.sciencedirect.com/science/article/pii/S0140366422000159

[14] TRILLES, S., LUJÁN, A., BELMONTE, Ó., MONTOLIU, R., TORRES-SOSPEDRA, J. HUERTA, J. *SEnviro: A Sensorized Platform Proposal Using Open Hardware and Open Standards. Sensors.* 2015. https://doi.org/10.3390/s150305555. [online]. [cit. 2022-29-03]. Dostupné z WWW: https://www.mdpi.com/1424-8220/15/3/5555

[15] *OGC SensorThings API—Sensing*. Jan. 2016. [online]. [cit. 2022-29-03]. Dostupné z WWW: http://www.opengeospatial.org/standards/sensorthings

[16] *FraunhoferIOSB: GitHub - FraunhoferIOSB/FROST-Server: A Server implementation of the OGC SensorThings API*. [online]. [cit. 2022-29-03]. Dostupné z WWW: https://github.com/FraunhoferIOSB/FROST-Server

[17] *Seznam OGC certifikovaných implementací SensorThings API.* [online]. [cit. 2022-29-03]. Dostupné z WWW: https://www.ogc.org/resource/products?display_opt=1&specid=772

[18] *Datová platforma Life 4.0 Sciences*. Česká zemědělská univerzita v Praze, Katedra informačních technologií. [online]. [cit. 2022-29-03]. Dostupné z WWW: https://ls40.pef.czu.cz/datova-platforma

[19] POSS, E., BASSI, A., a HORN, G. *"Internet of Things in 2020: A roadmap for the future"*. Eur. Commission: Inf. Soc. Media. Sep. 2008. [online]. [cit. 2022-25-07]. Dostupné z WWW: https://docbox.etsi.org/erm/Open/CERP%2020080609-10/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v1-1.pdf

[20] MARTIKKALA, A., LOBOV, A., LANZ, M. a ITUARTE, I. F. *Towards the Interoperability of IoT Platforms: A Case Study for Data Collection and Data Storage.* IFAC-PapersOnLine. ISSN 2405-8963. https://doi.org/10.1016/j.ifacol.2021.08.134. [online]. [cit. 2022-25-07]. Dostupné z WWW: https://www.sciencedirect.com/science/article/pii/S2405896321008958

[21] TSCHOFENIG, H., ARKKO, J. a THALER, D. aj. *Architectural Considerations in Smart Object Networking.* Technická zpráva. March 2015. doi:10.17487/rfc7452. [online]. [cit. 2022-25-07]. Dostupné z WWW: https://www.rfc-editor.org/info/rfc7452

[22] DÂMASO, A., ROSA, N., a MACIEL, P. *Reliability of Wireless Sensor Networks.* Sensors 2014. https://doi.org/10.3390/s140915760. [online]. [cit. 2022-25-07]. Dostupné z WWW: https://www.mdpi.com/1424-8220/14/9/15760

[23] HEJLOVÁ, V. *Experimentální bezdrátová senzorová síť pro monitoring znečištění ovzduší ve středu města Olomouce (E-BOSS).* Univerzita Palackého v Olomouci, 2017. [online]. [cit. 2022-25-07]. Dostupné z WWW: https://theses.cz/id/ln5quo/disertacni_prace-final2.pdf

[24] TECHTARGET. *What is microcontroller?* [online]. [cit. 2022-19-04]. Dostupné z WWW: https://www.techtarget.com/iotagenda/definition/microcontroller

[25] TECHTARGET. *Microcontroller usage.* [online]. [cit. 2022-19-04]. Dostupné z WWW: https://www.techtarget.com/iotagenda/definition/microcontroller

[26] TECHOPEDIA. *Single-board computers explained.* [online]. [cit. 2022-19-04]. Dostupné z WWW: https://www.techopedia.com/definition/9266/single-board-computer-sbc

[27] EPC-RFID. EPC Information. [online]. [cit. 2022-25-07]. Dostupné z WWW: https://www.epc-rfid.info/

[28] UID CENTER. *What is ucode?* [online]. [cit. 2022-25-07]. Dostupné z WWW: http://www.uidcenter.org/learning-about-ucode/what-is-ucode

[29] SCIENCE DIRECT. *Radio Frequency Identification.* [online]. [cit. 2022-25-07]. Dostupné z WWW: https://www.sciencedirect.com/topics/computer-science/radio-frequency-identification

[30] PELAEZ, A., UBIDOTS. *What is IoT data collection?* [online]. [cit. 2022-30-07]. Dostupné z WWW: https://ubidots.com/blog/iot-data-collection

[31] HOJLO, J., IDC. *Future of Industry Ecosystems: Shared Data and Insights.* January 2021. [online]. [cit. 2022-30-07]. Dostupné z WWW: https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/

[32] MAHANTHAPPA, S., CHANDAVARKAR, B.R. Data Formats and Its Research Challenges in IoT: A Survey. In Suma 2021. https://doi.org/10.1007/978-981-15-5258-8_47. [online]. [cit. 2022-30-07]. Dostupné z WWW: https://link.springer.com/chapter/10.1007/978-981-15-5258-8_47

[33] INTUZ. *Top 25 IoT Development Boards in 2022 And How To Choose The Right One.* [online]. [cit. 2022-30-07]. Dostupné z WWW: https://www.intuz.com/guide-on-top-iot-development-boards

[34] YOUNGWONKS. *Top 10 IoT Boards for development and prototyping in 2022.* [online]. [cit. 2022-30-07]. Dostupné z WWW: https://www.youngwonks.com/blog/Top-10-IoT-boards-for-2019

[35] HASHSTUDIOZ. *Top IoT Development Boards & Steps To Select The Right One For Your Project.* [online]. [cit. 2022-30-07]. Dostupné z WWW: https://hashstudioz.com/blog/top-iot-development-boards-how-to-select-the-right-one-for-your-project/

[36] HOLOGRAM. *The 13 best IoT development boards & shields in 2022.* [online]. [cit. 2022-30-07]. Dostupné z WWW: https://www.hologram.io/blog/top-development-boards-shields-iot-projects

[37] SILICONITHUB. *Top IoT Boards to Kickstart Project Development in 2022.* [online]. [cit. 2022-30-07]. Dostupné z WWW: https://siliconithub.com/top-iot-boards/

[38] VERYTECHNOLOGY. *Top 10 IoT Boards for Development & Prototyping in 2022.* [online]. [cit. 2022-30-07]. Dostupné z WWW: https://www.verypossible.com/insights/top-10-iot-boards-for-development-prototyping-in-2022

[39] ELECTRONICPRODUCTS. *Top 10 development boards for IoT.* [online]. [cit. 2022-30-07]. Dostupné z WWW: https://www.electronicproducts.com/top-10-development-boards-for-iot/

[40] RESTFUL API. *What is REST.* [online]. [cit. 2022-15-08]. Dostupné z WWW: https://restfulapi.net/

[41] RESTFUL API. REST *Architectural Constraints.* [online]. [cit. 2022-15-08]. Dostupné z WWW: https://restfulapi.net/rest-architectural-constraints/

[42] RESTFUL API. REST *Resource Naming Guide.* [online]. [cit. 2022-15-08]. Dostupné z WWW: https://restfulapi.net/resource-naming/

[43] Object Management Group. *OMG Standards Development Organization.* [online]. [cit. 2022-15-08]. Dostupné z WWW: https://www.omg.org/

[44] JARA, A. J., MARTINEZ-JULIA, P. a SKARMETA, A. *"Light-weight multicast DNS and DNS-SD (lmDNS-SD): IPv6-based resource and service discovery for the web of things".* Proc. 6th Int. Conf. IMIS Ubiquitous Comput, 2022. [online]. [cit. 2022-15-08]. Dostupné z WWW: https://ieeexplore.ieee.org/document/6296945

[45] KLAUCK, R. a KIRSCHE, M. *"Chatty things—Making the Internet of Things readily usable for the masses with XMPP".* Proc. 8th Int. Conf. CollaborateCom, 2012. [online]. [cit. 2022-15-08]. Dostupné z WWW: https://ieeexplore.ieee.org/abstract/document/6450893

[46] IETF. *DNS Terminology – RFC 8499.* 2019. [online]. [cit. 2022-15-08]. Dostupné z WWW: https://datatracker.ietf.org/doc/rfc8499/

[47] IETF. *The Internet Engineering Task Force.* [online]. [cit. 2022-15-08]. Dostupné z WWW: https://www.ietf.org/

[48] PALATTELLA, M. R., et al. *"Standardized protocol stack for the Internet of (important) things".* IEEE Commun. Surveys Tuts., 3rd Quart. 2013. [online]. [cit. 2022-15-08]. Dostupné z WWW: https://ieeexplore.ieee.org/document/6380493

[49] HUI, J. W. a CULLER, D. E. *"Extending IP to low-power wireless personal area networks".* IEEE Internet Comput, Jul./Aug. 2008. [online]. [cit. 2022-15-08]. Dostupné z WWW: https://ieeexplore.ieee.org/document/4557977

[50] FRANK, R., BRONZI, W., CASTIGNANI, G. a ENGEL, T. *"Bluetooth low energy: An alternative technology for VANET applications".* Proc. 11th Annu. Conf. WONS, 2014. [online]. [cit. 2022-15-08]. Dostupné z WWW: https://ieeexplore.ieee.org/document/6814729

[51] DECUIR, J. *"Introducing Bluetooth smart: Part 1: A look at both classic and new technologies".* IEEE Consum. Electron. Mag., Jan. 2014. [online]. [cit. 2022-15-08]. Dostupné z WWW: https://ieeexplore.ieee.org/document/6685914

[52] SIEKKINEN, M., HIIENKARI, M., NURMINEN, K a NIEMINEN, J. *"How low energy is Bluetooth low energy? Comparative measurements with ZigBee/802.15.4".* Proc. IEEE WCNCW, 2012. [online]. [cit. 2022-30-08]. Dostupné z WWW: https://ieeexplore.ieee.org/document/6215496

[53] MINOLI, D. *Building the Internet of Things With IPv6 and MIPv6: The Evolving World of M2M Communications.* New York, NY, USA, Wiley, 2013. [online]. [cit. 2022-30-08]. Dostupné z WWW: https://ieeexplore.ieee.org/book/8040289

[54] GOMEZ. C a PARADELLS, J. *"Wireless home automation networks: A survey of architectures and technologies".* IEEE Commun. Mag., Jun. 2010. [online]. [cit. 2022-03-12]. Dostupné z WWW: https://ieeexplore.ieee.org/document/5473869

[55] CLAUSEN, T., HERBERG, U. a PHILIPP, M. *"A critical evaluation of the IPv6 routing protocol for low power and lossy networks (RPL)".* Proc. IEEE 7th Int. Conf. WiMob, 2011. [online]. [cit. 2022-03-12]. Dostupné z WWW: https://ieeexplore.ieee.org/abstract/document/6085374

[56] ACCETTURA, N., GRIECO, L. A., BOGGIA, G. a CAMARDA, P. *"Performance analysis of the RPL routing protocol".* Proc. IEEE ICM, 2011. [online]. [cit. 2022-03-12]. Dostupné z WWW: https://ieeexplore.ieee.org/document/5971218

[57] ANCILLOTTI, E., BRUNO, R. a CONTI, M. *"RPL routing protocol in advanced metering infrastructures: An analysis of the unreliability problems".* Proc. SustainIT, 2012. [online]. [cit. 2022-03-12]. Dostupné z WWW: https://ieeexplore.ieee.org/document/6388038

[58] ENJIAN, B. a XIAOKUI, Z. *"Performance evaluation of 6LoWPAN gateway used in actual network environment".* Proc. ICCECT, 2012. [online]. [cit. 2022-12-12]. Dostupné z WWW: https://ieeexplore.ieee.org/document/6413761

[59] OLYAEI, B. B., PIRSKANEN, J., RAEESI, O., HAZMI, A. a VALKAMA, M. *"Performance comparison between slotted IEEE 802.15.4 and IEEE 802.1 lah in IoT based applications".* Proc. IEEE 9th Int. Conf. WiMob, 2013. [online]. [cit. 2022-12-12]. Dostupné z WWW: https://ieeexplore.ieee.org/document/6673381

[60] WITHANAGE, C., ASHOK, R., YUEN, C a OTTO, K. *"A comparison of the popular home automation technologies".* Proc. IEEE ISGT Asia, 2014. [online]. [cit. 2022-12-12]. Dostupné z WWW: https://ieeexplore.ieee.org/document/6873860

[61] TRILLES, S., GONZÁLEZ-PÉREZ, A. a HUERTA, J. *An IoT Platform Based on Microservices and Serverless Paradigms for Smart Farming Purposes.* Sensors. 2020. https://doi.org/10.3390/s20082418. [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.mdpi.com/1424-8220/20/8/2418

[62] AMAZON. *AWS IoT.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://aws.amazon.com/iot/

[63] MICROSOFT. *Azure Event Hubs.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://azure.microsoft.com/en-us/products/event-hubs

[64] IBM. *IBM IoT Cloud.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.ibm.com/cloud/internet-of-things

[65] ORACLE. *Oracle Internet of Things.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.oracle.com/internet-of-things/

[66] GOOGLE. *Google Cloud IoT Core.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://cloud.google.com/iot-core

[67] THINGSBOARD. *ThingsBoard Open-source IoT Platform.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://thingsboard.io/

[68] SENSORUP. *SensorUp IoT Platform.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://sensorup.com/platform/

[69] DEVICEHIVE. *DeviceHive IoT Made Easy – Open Source IoT Data Platform.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://devicehive.com/

[70] DOMÍNGUEZ-BOLAÑO, T., CAMPOS, O., BARRAL, V., ESCUDERO, C. J. a GARCÍA-NAYA, J. A. *An overview of IoT architectures, technologies, and existing open-source projects.* Internet of Things, Volume 20, 2022, ISSN 2542-6605. https://doi.org/10.1016/j.iot.2022.100626. [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.sciencedirect.com/science/article/pii/S254266052200107X

[71] ZHANG, J., MA, M., WANG. P a SUN, X. *Middleware for the Internet of Things: A survey on requirements, enabling technologies, and solutions. Journal of Systems Architecture.* 2021, ISSN 1383-7621. https://doi.org/10.1016/j.sysarc.2021.102098. [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.sciencedirect.com/science/article/pii/S1383762121000795

[72] ECLIPSE. *Eclipse Kapua.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://projects.eclipse.org/projects/iot.kapua

[73] HOME ASSISTANT. *Home Assistant.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.home-assistant.io/

[74] MAINFLUX. *Mainflux Labs Open source IoT Platform.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://mainflux.com/

[75] OPENEMS. *Open Energy Management System Platform.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://openems.github.io/openems.io/openems/latest/introduction.html

[76] OPENHAB. *openHAB empowering the smart home.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.openhab.org/

[77] OPENREMOTE. *openremote Open Source IoT Platform.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.openremote.io/

[78] SITEWHERE. *SiteWhere the open platform for Internet of Things.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://sitewhere.io/en/

[79] GOST. *Go-SensorThings IoT Platform.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://github.com/gost/server

[80] ICCS. *Institute of Communication and Computer Systems SensorThings API certified implementation.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.ogc.org/resources/product-details/?pid=1554

[81] RAJ, P., DEKA, G. C., GUPTA,N. a AGRAWAL, R. *A Deep Dive into NoSQL Databases: The Use Cases and Applications.* Advances in Computers, Volume 109, 2018, ISSN: 0065-2458. [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www-sciencedirect-com.infozdroje.czu.cz/science/article/abs/pii/S0065245818300032

[82] POSTGRESQL. *PostgreSQL: The world's most advanced open source relational database.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.postgresql.org/

[83] MYSQL. *MySQL.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.mysql.com/

[84] MARIADB. *MariaDB Server: The open source relational database.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://mariadb.org/

[85] SQLITE. *SQLite database engine.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.sqlite.org/index.html

[86] APACHE. *Apache CASSANDRA, Open Source NoSQL Database.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://cassandra.apache.org/_/index.html

[87] APACHE. *Apache CouchDB.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://couchdb.apache.org/

[88] MONGODB. *MongoDB: Build the next big thing.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.mongodb.com/

[89] INFLUXDATA. *InfluxDB: Is the smart data platform for time series.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.influxdata.com/products/

[90] TIMESCALE. *Timescale: High-performance PostgreSQL for time-series and analytics.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.timescale.com/products

[91] OPENTSDB. *OpenTSBD: The scalable time series database.* [online]. [cit. 2022-12-12]. Dostupné z WWW: http://opentsdb.net/

[92] POSTGRESQL. *PostgreSQL License.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.postgresql.org/about/licence/

[93] GNU. *GNU General Public License, version 2.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

[94] APACHE. *Apache License, version 2.0.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.apache.org/licenses/LICENSE-2.0

[95] MONGODB. *MongoDB, Server Side Public License, SSPL.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.mongodb.com/licensing/server-side-public-license

[96] OPENSOURCE. *Open source initiative, The MIT license.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://opensource.org/license/mit/

[97] GNU. *GNU General Public License, version 3.* [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.gnu.org/licenses/gpl-3.0.en.html

[98] PRAMANIK, P. K. D. a CHOUDHURY, P. *IoT Data Processing: The Different Archetypes and Their Security and Privacy Assessment.* 1st Edition. River Publishers, 2018, ISBN 9781003338642. [cit. 2022-12-12].

[99] KRISHNAMURTHI, R., KUMAR, A., GOPINATHAN, D., NAYYAR, A. a QURESHI, B. *An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques.* Sensors, 2020. https://doi.org/10.3390/s20216076. [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.mdpi.com/1424-8220/20/21/6076

[100] SHAH, N., SHAH, S., JAIN, P. a DOSHI, N. *Overview of Present-Day IoT Data Processing Technologies.* Procedia Computer Science, Volume 210, 2022, ISSN 1877-0509. https://doi.org/10.1016/j.procs.2022.10.150. [online]. [cit. 2022-12-12]. Dostupné z WWW: https://www.sciencedirect.com/science/article/pii/S1877050922016076

[101] CARNEGIE MELLON. Carnegie Mellon University. [online]. [cit. 2022-15-12]. Dostupné z WWW: https://www.cmu.edu/

[102] ARCGIS. *ArcGIS Online.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://www.arcgis.com/index.html

[103] ESRI. *Esri: Advancing the power of geography.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://www.esri.com/en-us/about/about-esri/overview

[104] NATO. *NATO NCI Agency Announces Winners of Third Annual Defence Innovation Challenge.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://www.ncia.nato.int/about-us/newsroom/nci-agency-announces-winners-of-third-annual-defence-innovation-challenge.html

[105] CIO APPLICATIONS. CIO Applications Announces Top Internet of Things Solution Provider of 2021. [online]. [cit. 2023-20-02]. Dostupné z WWW: https://internet-of-things.cioapplications.com/vendor/sensorup-simplifying-data-collection-from-connected-devices-cid-5564-mid-362.html

[106] SENSORUP. *About the Department of Homeland Security.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://sensorup.com/dhs/

[107] DHS GOV. *News Release: DHS Partners with Industry for Operational Experimentation in Houston, Texas.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://www.dhs.gov/science-and-technology/news/2018/11/20/news-release-dhs-partners-industry-houston

[108] SENSORUP. *SensorUp Joins the Esri Startup Programme.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://sensorup.com/blog/sensorup-joins-the-esri-startup-programme/

[109] IOSB FRAUNHOFER. Fraunhofer Institute of Optronics, System Technologies and Image Exploitation. [online]. [cit. 2023-20-02]. Dostupné z WWW: https://www.iosb.fraunhofer.de/en.html

[110] GRAFANA. *FROST SensorThings API Plugin.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://grafana.com/grafana/plugins/iosb-sensorthings-datasource/

[111] MEDIUM. *Introduction to InfluxDB: A time-series database.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://anurag-nair.medium.com/introduction-to-influxdb-a-time-series-database-6a32460c0554

[112] INFLUXDATA. InfluxData: It's About Time. Build on InfluxDB. [online]. [cit. 2023-20-02]. Dostupné z WWW: https://www.influxdata.com/

[113] KX. *Developing with kdb+ and the q language.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://code.kx.com/q/

[114] KX. KX: *Elegantly simple. Universally compatible.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://kx.com/about/

[115] PROMETHEUS. *Prometheus: From metrics to insight.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://prometheus.io/

[116] CNCF. *Cloud Native Computing Foundation: Make Cloud native ubiquitous. [online].* [cit. 2023-20-02]. Dostupné z WWW: https://www.cncf.io/

[117] GRAPHITEAPP. *Graphite does three things: Kick Ass. Chew bubblegum. Make it easy to store and graph metrics.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://graphiteapp.org/

[118] ORBITZ. Orbitz: *All travelers are welcome here.* [online]. [cit. 2023-20-02]. Dostupné z WWW: https://www.orbitz.com/lp/b/about

[119] ABEEWAY. *Abeeway Industrial Tracker.* [online]. [cit. 2023-04-03]. Dostupné z WWW: https://www.abeeway.com/industrial-tracker/

[120] NODERED. *Node-Red: Low-code programming for event-driven application.* [online]. [cit. 2023-04-03]. Dostupné z WWW: https://nodered.org/

[121] LIQUIBASE. *Liquibase: Fast database change. Fluid delivery.* [online]. [cit. 2023-04-03]. Dostupné z WWW: https://www.liquibase.org/

[122] OGC. *Open Geospatial Consortium.* [online]. [cit. 2022-04-03]. Dostupné z WWW: https://www.ogc.org/

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam použitých zkratek

ACID	Atomic, Consistent, Isolated, Durable
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
B	byte
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
CSV	Comma-seperated values
DAO	Destination Advertisment Object
DAO-Ack	Destination Advertisment Object Acknowledgement
DB	Database
DCPS	Data-Centric Publish-Subscribe

DDS	Data Distribution Service
DIO	Destination Oriented Directed Acyclic Graph Information Object
DIS	Destination Oriented Directed Acyclic Graph Information Solicitation
DLRL	Data-Local Reconstruction Layer
DNS	Domain Name System
DNS-DS	Domain Name System Service Discovery
DRAM	Dynamic random-access memory
DODAG	Destination Oriented Directed Acyclic Graph
DTLS	Datagram Transport Layer Security
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPC	Electronic Product Code
EXI	Efficient Extensible Markup Language Interchange
GB	gigabyte
GHz	Gigahertz
GPS	Global Positioning System
GOST	Go-SensorThings
GPLv2	General Public License v2.0
GPLv3	General Public License v3.0
HTTP	Hypertext Transfer Protocol
ICCS	Institute of Communication and Computer Systems
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IM	Instant Messaging
IoT	Internet of Things
IP	Internet Protocol
IPSO	Internet Protocol for Smart Objects
IPv6	Internet Protocol version 6
ISM	Industrial, Scientific, and Medical
JSON	JavaScript Object Notation
kb	kilobit
KB	kilobyte
LPWAN	Low-power, wide-area network

LTE-A	Long Term Evolution Advanced
MB	megabyte
mDNS	Multicast Domain Name System
MHz	Megahertz
MIME	Multipurpose Internet Mail Extensions
MIT	Massachusetts Institute of Technology
MQTT	Message Queuing Telemetry Transport
mW	milliwatt
NoSQL	Not Only Structured Query Language
ODBC	Open Database Connectivity
OGC	Open Geospatial Consortium
OMG	Object Management Group
ONS	Object Name Service
PaaS	Platform as a service
PIR	Passive Infrared Sensor
QoS	Quality of Service
RAM	Random-access memory
RDBMS	Relational database management system
REST	Representational state transfer
RFC	Request for Comments
RFID	Radio Frequency Identification
RPL	Routing Protocol for Low Power and Lossy Networks
SaaS	Software as a service
SBC	Single-board computer
SD	Secure Digital
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator

XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
YAML	Yet Another Markup Language