TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Test technologie LF Edge na vestavných systémech s Linuxem

Bakalářská práce

Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie

Autor práce: **Tomáš Mejzr**
Vedoucí práce: Ing. Lenka Kosková Třísková, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Test technologie LF Edge na vestavných systémech s Linuxem

Jméno a příjmení: **Tomáš Mejzr**
Osobní číslo: M17000084
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Seznamte se s významem „Edge computing".
2. Seznamte se nástroji z frameworku LF Framework.
3. Srovnejte možnosti instalace Acraino Eliot a EdgeX pro IoT.
4. Zprovozněte alespoň jeden z vybraných nástrojů.
5. Navrhněte a realizujte test funkčnosti zvoleného nástroje pro IoT.

Rozsah grafických prací: dle potřeby dokumentace
Rozsah pracovní zprávy: 30 – 40 stran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] Proceedings of IEEE, Edge Computing, Volume 7, Issue 8, August 2019, ISBN.
[2] A Linux Server Operating System’s Performance Comparison using lmbench, 2016 International Conference on Network and Information Systems for Computers, ISBN: 9781467388382.
[3] On Performance of Kernel Based and Embedded Real-Time Operating System: Benchmarking and Analysis, ICACSIS 2011, ISBN: 9789791421119.

Vedoucí práce: Ing. Lenka Kosková Třísková, Ph.D.
Ústav nových technologií a aplikované informatiky

Datum zadání práce: 19. října 2020
Předpokládaný termín odevzdání: 17. května 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2020

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

16. května 2021 Tomáš Mejzr

Poděkování

Chtěl bych poděkovat své vedoucí bakalářské práce Ing. Lence Koskové Třískové, Ph.D. za její odborné vedení, nekonečnou trpělivost a přístup. Také za její ochotu i v této nelehké koronavirové době.

Dále pak mé rodině, neboť to v době psaní neměla vůbec jednoduché a musela snášet moje výkyvy nálad, nadávání na celý svět a třískáním se vším, co jsem měl po ruce.

Abstrakt

Cílem této práce je otestovat, zjistit funkčnost a porovnat mezi sebou dva frameworky pro Edge Computing z projektu LF Edge. Jelikož tato technologie je nová a teprve se dostává do většího povědomí širší veřejnosti, jedná se o jednu z prvních bakalářských prací na dané téma.

První část je věnována teorii a pojednává o tom, co je Edge Computing, charakterizuje jeho hlavní dělení. Dále seznamuje čtenáře s projektem LF Edge, frameworkem ELIOT od Akraina a EdgeX od EdgeX Foundry. Popisuje použitý hardware a operační systém.

Druhá, praktická část, vyobrazuje jednotlivé kroky instalace frameworků a následně v krátkosti shrnuje výsledek instalace, kde se ukázalo, že ELIOT od Akrania se na rozdíl od EdgeX, zprovoznit nepodařilo. Dále je vytvořena testovací úloha pro zkoumání správné funkčnosti jednotlivých frameworků. Tato úloha spočívá v zasílání jednotlivých dat ze senzorů, jejich exportování do databáze a následné zobrazení ve webovém prohlížeči, na to navazující porovnání frameworků a závěr.

Klíčová slova

Edge Computing, Akraino, ELIOT, EdgeX, EdgeX Foundry, Grafana, MySQL, MQTT, Raspberry Pi, Ubuntu, LF Edge

Abstract

The objective for this work, is to test, to find out the functionality and compare two edge computing frameworks from LF Edge's project. As this technology is new and is only just becoming more widely known, this is one of the first bachelor theses on the subject.

The first part is devoted to theory and discusses what edge computing is and describes its main divisions. It also introduces to the reader the LF Edge project, Akrain's ELIOT framework, and EdgeX from EdgeX Foundry. It describes which hardware and operating system were used.

In the second part which is practical. Describing the installation of both frameworks and then are final installation shortly summarized. ELIOT was not instaled, but EdgeX was. After this is created test task for finding correct functionality. This task is, getting data from sensors to edge, then exporting it via MQTT and storing in a database. From this database, the data is going to be showed in the web browser. In the end, the reader can find and compare these two frameworks and reach a final conclusion.

Key words

Edge Computing, Akraino, ELIOT, EdgeX, EdgeX Foundry, Grafana, MySQL, MQTT, Raspberry Pi, Ubuntu, LF Edge

Obsah

Seznam zkratek

API – Application Programming Interface – Zajišťuje komunikaci mezi dvěma platformami

DC – Direct current – Stejnosměrný proud

ELIOT – Edge Lightweight and IoT Blueprint Family – Framework od Akraina

HTTP – HyperText Transfer Protokol – Protokol pro přenos souborů mezi klientem a serverem

IoT – Internet of Things - Internet věcí

ME – Mobile Edge – Mobilní hrana

REST – Representation state transfer – Architektura rozhraní pro webové API orientovaná datově

SOAP – Simple Object Access Protocol – Protokol na výměnu zpráv založených na XML prostřednictvím HTTP

SQL – Structured Query Language – Standardizovaný strukturovaný dotazovací jazyk

TAC – Technical Advisory Council - Technický poradní spor organizace LF Edge

URL – Uniform Resource Locator - Jednotný lokátor zdroje

Úvod

Tato bakalářská práce se věnuje problematice Edge Computing. Jejím cílem je otestovat funkčnost a použitelnost dvou nástorojů pro Edge Computing, jež jsou součástí frameworku LF Edge. Testovací platformou je počítač Raspberry Pi. Záměr této práce je ověřit připravenost platforem k nasazení, v rámci čehož věnuji pozornost zejména instalaci, použitelnosti frameworku a zprovoznění testovací úlohy s navazujícím rozšířením. Výsledkem je vzájemné srovnání těchto dvou technologií, popsání funkčnosti a doporučení, která platforma je za daných okolností lepší a vhodná k nasazení.

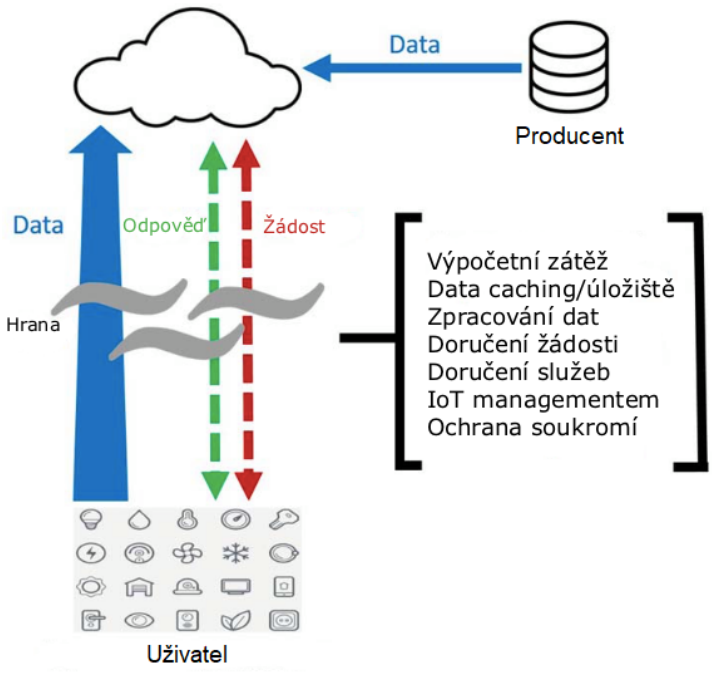
Text této bakalářské práce se postupně věnuje významu Edge Computing, jeho hlavnímu rozdělení a frameworku LF Edge, kde se popisuje, co to je, a prezentují se jejich projekty. Poté se zabývá představení dvou použitých platforem, charakteristikou samotného výzkumu a s tím souvisejícího přínosu. Následují popisy instalací, vyzkoušení technologie, rozšíření testovací úlohy, porovnání výsledků a závěr s doporučením.

1. Edge Computing

Edge Computing je koncept, kdy zařízení internetu věcí (IoT) odešle data právě na zařízení na hraně (uzel edge), kde se provede předem definovaná operace, a ta se poté odešlou do cloudu.

1.1 Definice Edge Computing

Článek [1] jej definuje: "Jako Edge Computing se označují technologie, které umožňují provádět výpočty na hraně mezi místní sítí a připojením do internetu nad daty jménem cloudových služeb a služeb internetu věcí. Hranou je myšleno jakékoliv výpočetní a síťové zařízení mezi IoT a cloudem. Například chytrý telefon (smart phone) je hranou mezi člověkem a cloudem. Brána (gateway) v chytré domácnosti je hranou mezi chytrým domácím vybavením a cloudem. Edge Computing se odehrává v blízkosti zdrojů dat, kde se zaměřuje více na stranu IoT."



Obrázek 1: *Obousměrný provoz, převzato a přeloženo z [1]*

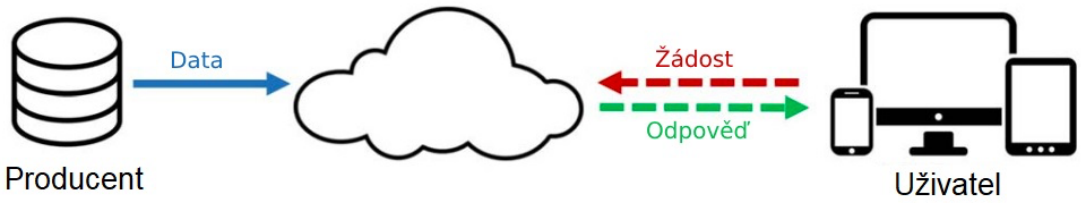
Jak je znázorněno na **Obrázek 1:** Obousměrný provoz, převzato a přeloženo z [1], v konceptu výpočtu na hraně zařízení IoT data vytvářejí, ale i přijímají. Mohou požadovat služby či obsahy z cloudu, zároveň také provádět výpočetní úkony. Dále pak mohou ukládat data, ukládat data do mezipaměti (cache) a zpracovávat je stejně, jako rozdělovat požadavky a doručovat služby z cloudu k uživateli. Kvůli tomu musí být Edge Computing navržen tak,

aby splňoval nároky na spolehlivost, bezpečnost a ochranu soukromí [1]. Edge Computing umožňuje zpracování a ukládání dat blíže ke koncovému bodu použitím známých cloudových technologií. Díky tomu se snižují náklady na vlastnictví, a také se sníží zpoždění aplikací na méně než 20 milisekund [2].

1.2 Důvod vzniku

Důvodem vytvoření tohoto konceptu je skutečnost, že dle [3] se předpokládá, že k roku 2025 bude ke cloudu přistupovat více než 150 miliard IoT zařízení. Také se očekává, že tato zařízení vygenerují více než $1{,}26 \times 10^{11}$ TB dat. Kdyby se tak opravdu stalo, dnešní koncept přímého odesílání dat do cloudu (Obrázek 2: Dnešní koncept přímého odesílání dat do cloudu) již nestačí, a to hned z několika důvodů:

- Zpoždění (Latency) - Nové aplikace IoT vyžadují zpracování dat, odpověď deterministicky a v režimu reálného času. Například autonomní vozidla vyžadují velmi malé zpoždění v rámci milisekund. Pokud by došlo k většímu zpoždění kvůli problémům se sítí, mohlo by to mít tragické důsledky [4].
- Šířka pásma (Bandwidth) - Spojení mezi zařízením IoT a cloudem má určitou šířku pásma, jenž nemusí stačit. Například Boening 787 vygeneruje více než 5 GB dat za vteřinu a jeho spojení se satelity má nedostatečnou šířku pásma pro použití v reálném čase [4].
- Dostupnost (Availability) - Na cloudu nalezneme čím dál tím víc aplikací, ke kterým se miliony zařízení IoT denně připojují. Proto je pro poskytovatele cloudu čím dál tím těžší udržet k nim neustálý přístup [4].
- Energie (Energy) - Veškerá cloudová datacentra spotřebují obrovské množství energie. S neustále přibývajícími zařízeními IoT se spotřeba energie datacenter bude neustále zvyšovat, až se stane omezující pro další rozvoj [4].
- Bezpečnost a soukromí (Security and Privacy) - S prosazením ochrany osobních údajů (GDPR) řeší poskytovatelé cloudu zabezpečení dat. Příkladem může být kamera v domácnosti, která odesílá video data do cloudu a kvůli tomu mohou být tato citlivá data zneužita [5].



Obrázek 2: *Dnešní koncept přímého odesílání dat do cloudu, převzato a přeloženo z [1]*

Všech pět výše uvedených problémů by měl vyřešit již zmiňovaný koncept výpočtu na hraně.

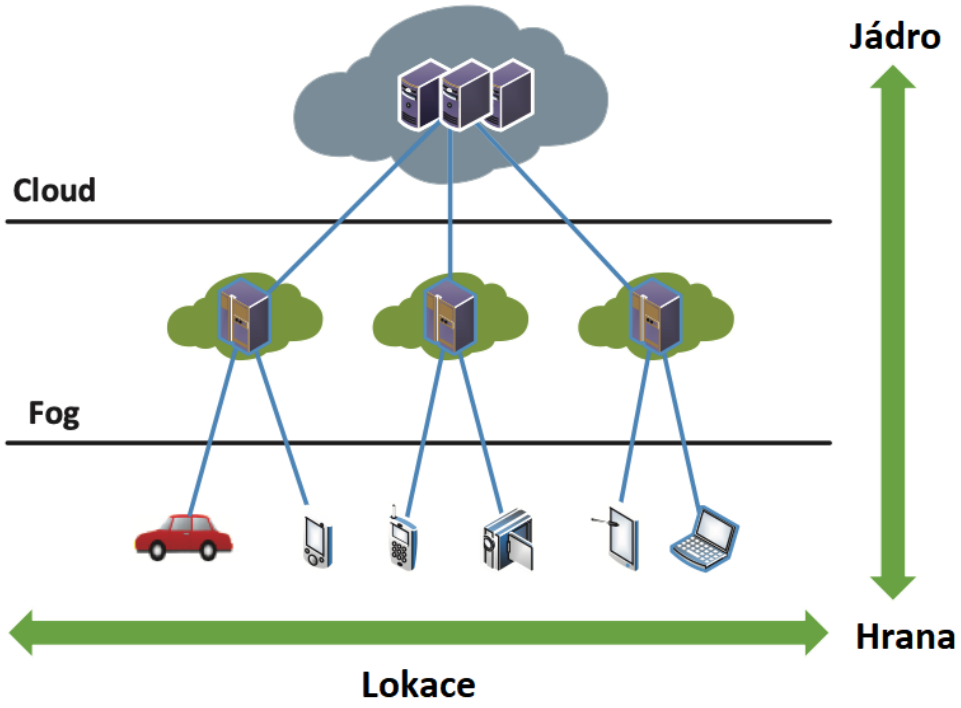
1.3 Edge Computing a jeho hlavní implementace

Podle [6] existují tři implementace, které se od sebe liší architekturou. Dále jsou rozdílné v umístění jejich uzlů (nachází se mezi cloudem a zařízením), poskytování služeb a aplikací. Jednotlivé implementace se nazývají:

- Fog computing
- MEC (Mobile edge computing)
- Cloudlet computing

1.3.1 Fog computing

Fog computing je koncept decentralizované výpočetní sítě založené na uzlech fog computing (FCNs). Tyto uzly jsou heterogenní, proto může být tato síť založena na směrovačích (routers), síťových přepínačích (switch), přístupových bodech (access points), branách internetu věcí (IoT gateways) či set-top boxech. Díky již zmiňované heterogenitě se jednotlivé uzly tváří jako jedna vrstva (Fog layer) Ta poskytuje jednotlivé funkce pro přidělování a monitorování zdrojů, zabezpečení, správy zařízení spolu s úložnými a výpočetními službami. Tyto funkce obsluhuje tzv. Service Orchestration Layer, která přijímá požadavky od uživatelů, vyřizuje je a přiřazuje prostředky. **Obrázek 3:** Zobrazení Fog computing architektury, převzato a přeloženo z [7] ukazuje, jak vypadá Fog computing.



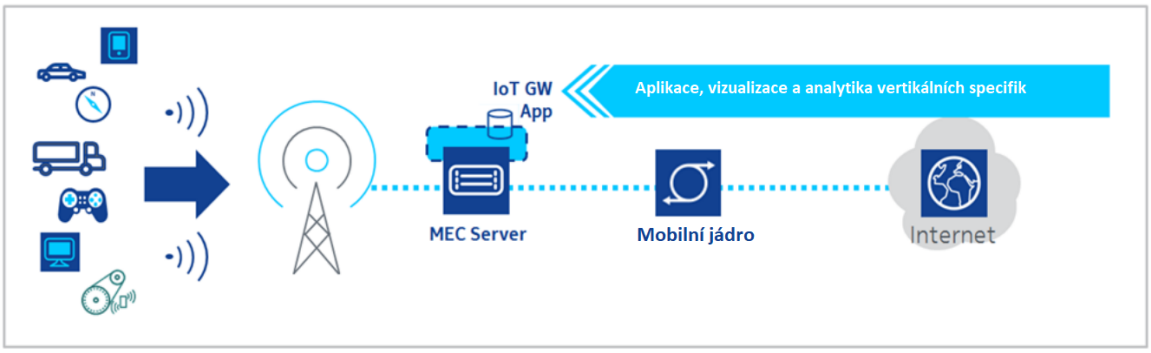
Obrázek 3: *Zobrazení Fog computing architektury, převzato a přeloženo z [7]*

1.3.2 Mobile edge computing

Článek [6] definuje Mobile Edge Computing jako snahu implementovat Edge Computing na okraj rádiové přístupové sítě (Radio Access Network) pro snížení latence. Uzly nebo servery této sítě jsou obvykle umístěny s ovladačem rádiové sítě (Radio Network Controller).

Servery této sítě spouští více instancí hostitelů MEC, které mají kapacitu provádět výpočty a úložiště ve virtualizovaném prostředí. Na tyto hostitele dohlíží Mobile Edge Orchestrator, který zpracovává informace o službách, jež hostitelé MEC nabízejí. Také poskytuje informace o dostupných prostředcích a topologii sítě.

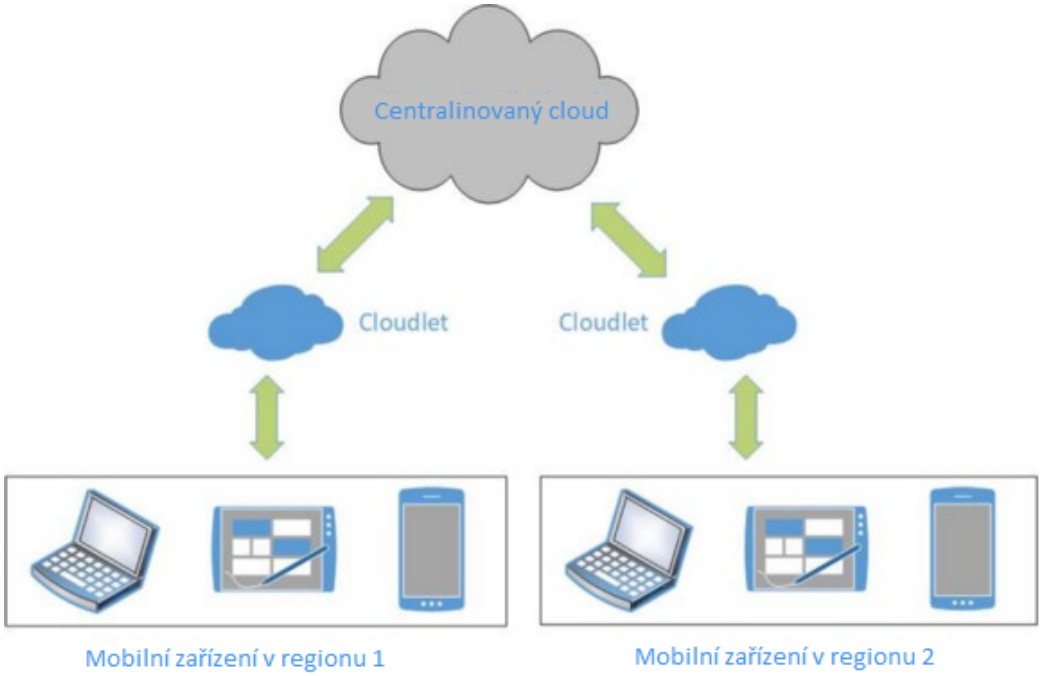
Servery MEC nabízejí informace o síti v reálném čase včetně informací o nich samotných. Tyto informace zahrnují zatížení a kapacitu sítě, počet připojených zařízení k tomuto serveru, jejich polohu a síťové informace.



Obrázek 4: *Ukázka MEC sítě, převzato a přeloženo z [8]*

1.3.3 Cloudlet computing

Cloudlet lze podle [6] definovat jako důvěryhodné seskupení počítačů (cluster) připojených k internetu se zdroji použitelnými pro blízká mobilní zařízení. Lze s ním zacházet jako s datovým centrem běžícím ve virtuálním prostředí, schopným zajišťovat prostředky koncovým zařízením a uživatelům v reálném čase po lokální bezdrátové síti. Cloudlet poskytuje přístup ke svým službám pouze jedním skokem (one-hop access) s velkou šířkou pásma, a díky tomu i s nízkým zpožděním. Na stránce [9] se uvádí, že na Cloudlet se většinou připojují mobilní zařízení a je specifický pro daný region. Pokud mobilní zařízení změní region, musí se změnit i připojení ke clusteru.



Obrázek 5: *Ukázka Cloudlet sítě, převzato a přeloženo z [9]*

1.3.4 Rozdíly mezi jednotlivými implementacemi

Rozdíly mezi architekturou Fog, ME a Cloudlet ukazuje **Tabulka 1:** Rozdíly mezi Computing Fog, ME a Cloudlet, převzato a přeloženo z [6].

Jejich hlavní rozdíly spočívají v síťových uzlech, umístění síťových uzlů, softwarové architektuře, vzdálenosti od síťového uzlu, přístupový mechanismus a komunikace s interním uzlem.

	Komunikace s interním uzlem	Přístupový mechanismus	Vzdálenost	Kontextové povědomí	Softwarová architektura	Umístění síťového uzlu	Síťový uzel
Fog Computing	Podporováno	Bluetooth, wi-fi, mobilní sítě	Jeden nebo více skoků	Střední	For Abstraction Layer	Různě mezi koncovým zařízením a cloudem	Směrovače, interní přepínače, přístupové body, brány
MEC	Částečně	Mobilní sítě	Jeden skok	Vysoké	Mobile Orchestrator	Radiový síťový ovadač	Servery běžící v základnové stanici
Cloudlet computing	Částečně	Wi-Fi	Jeden skok	Malé	Clooudlet Agent	Místní nebo venkovní instalace	Zapouzdřené datové centrum

Tabulka 1: *Rozdíly mezi Computing Fog, ME a Cloudlet, převzato a přeloženo z [6]*

2. LF Edge

LF Edge je zastřešující organizace, která byla založena v lednu 2019 neziskovou organizací Linux Foundation [10]. LF Edge se zaměřuje na vytvoření otevřeného a interoperabilního aplikačního rámce (framework) pro Edge Computing. Tento rámec by měl být nezávislý na operačním systému, cloudu či hardwaru [11].

LF Edge spojuje významné osobnosti oboru a společně vytváří framework pro hardwarové standardy, softwarové standardy a osvědčené postupy pro současnou i budoucí generaci IoT [11]. Mezi hlavní členy LF Edge a zároveň i přispěvatele se řadí společnosti Intel, IBM, Hewlett-Packard, Huawei, Nokia a další[1] [12].

Organizace LF Edge mají již nyní několik (9) rámců ve třech různých úrovních vývoje. Jedná se o postupové úrovně, které se nazývají: “At Large”, “Growth Stage”, “Impact Stage”.

2.1 At large – 1. úroveň

Projekty v této fázi vývoje jsou v úplném začátku přípravy. Odborníci z TAC tomuto projektu věří, že má jistý potenciál a může být dokonce důležitý v ekosystému nejvyšší úrovně nebo pro ekosystém na hraně. Projekt je buď v rané fázi, či dlouhodobý s minimální potřebou zdrojů. Projekty, které nalezneme v této fázi, jsou například: Baetyl, Open Horizont a Secure Device Onboard [13].

- **Baetyl** - Podle [14] Baetyl rozšiřuje cloud, data, služby a umožňuje vytvářet lehké, zabezpečené a škálovatelné aplikace na hranách.
- **Secure Device OnBoard** - Poskytuje snadnější, levnější, rychlejší a bezpečnější nasazení zařízení na hranu [15].
- **Open Horizon** - Je platforma pro správu životního cyklu kontejnerových úloh souvisejících se strojovým učením. Umožňuje autonomní správu aplikací nasazených na uzlech na hraně, aniž by vyžadovaly přístup administrátora [16].

[1] Ke dni 25. 02. 2021 počet členů činí přesně 68.

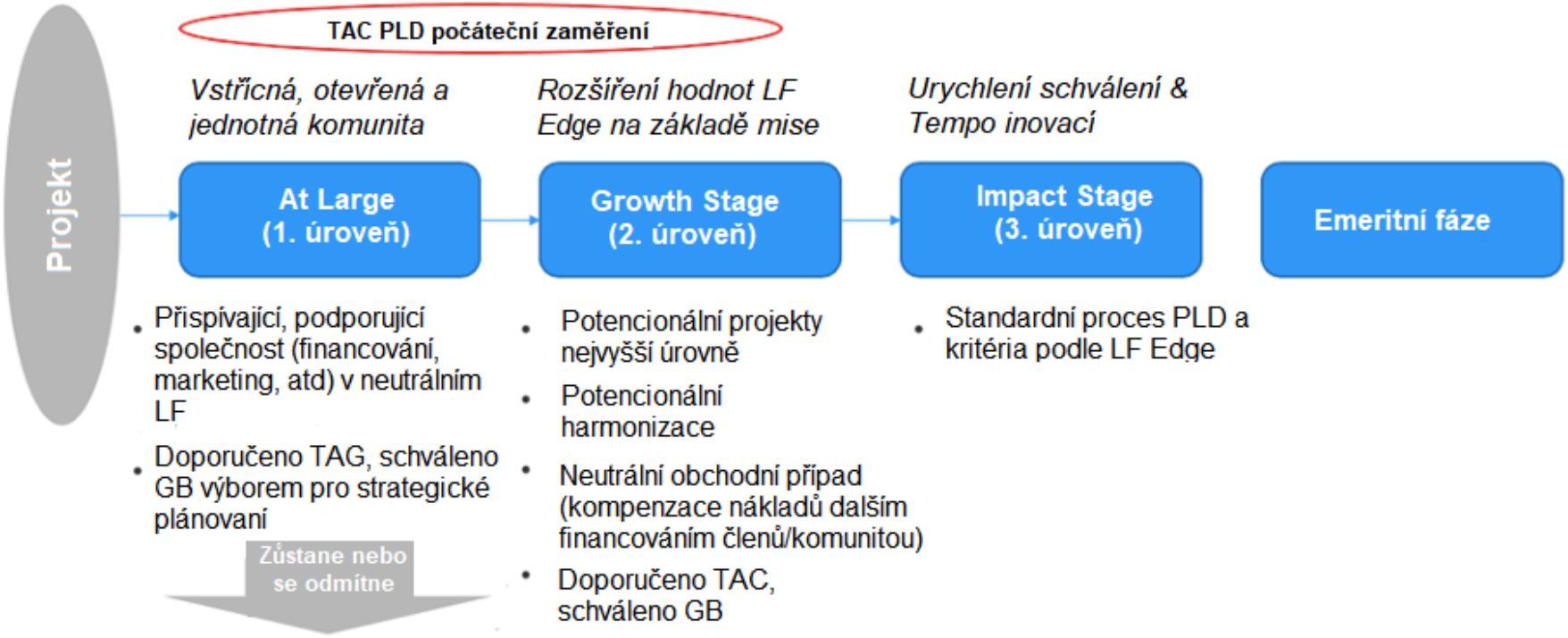
2.2 Growth Stage – 2. úroveň

Růstová fáze je určena pro projekty, které aspirují na dosažení 3. nejvyšší fáze. V úrovni Growth Stage již projekty dostávají mentorství od odborníků z TAC, od projektů se očekává, že budou aktivně rozvíjet svou komunitu uživatelů, správu, projektovou dokumentaci a další faktory ovlivňující úspěch těchto projektů. Aktuálně se v úrovni růstu nachází projekty Home Edge, Fledge, State of the Edge a Edge Virtualization (EVE) [13].

- **EVE** - Projekt EVE vytváří vlastní operační systém založený na Linuxu pro distribuci Edge Computing. Klade si za cíl zjednodušit tak vývoj, orchestraci a zabezpečení všech uzlů na hraně [17].
- **Fledge** - Fledge je otevřený zdrojový rámec a komunita pro průmyslovou hranu. Zaměřuje se na prediktivní službu, kritické operace, bezpečnost a situační povědomí. Integruje průmyslové IoT, senzory a moderní stroje do cloudu a stávajících systémů [18].
- **Home Edge** - Na stránkách [19] popisují HomeEdge jako robustní, spolehlivý a inteligentní ekosystém pro Edge Computing.
- **State of the Edge** - Je dodavatelsky neutrální platfotma pro otevřený výzkum. Věnuje se zrychlení inovací Edge Computingu crowdsourcingem. Projekt vyvíjí bezplatný sdílený výzkum, který se používá k diskuzi o přesvědčivých řešeních, jež nabízí Edge Computing [20].

2.3 Impact Stage – 3. úroveň

Tyto projekty již dosáhly svých růstových cílů a jsou ve stavu soběstačnosti vývoje, údržby a dlouhodobé podpory. V této úrovni jsou projekty také široce používány v produkčním prostředí a mají komunitu, která do projektu přispívá (minimálně ze dvou organizací). Aktuálně se v této fázi vyskytují dva projekty, jež byly vybrány i pro tuto bakalářskou práci. Jedná se o projekty Akraino a EdgeX Foundry (EdgeX), které budou popsány níže [13].



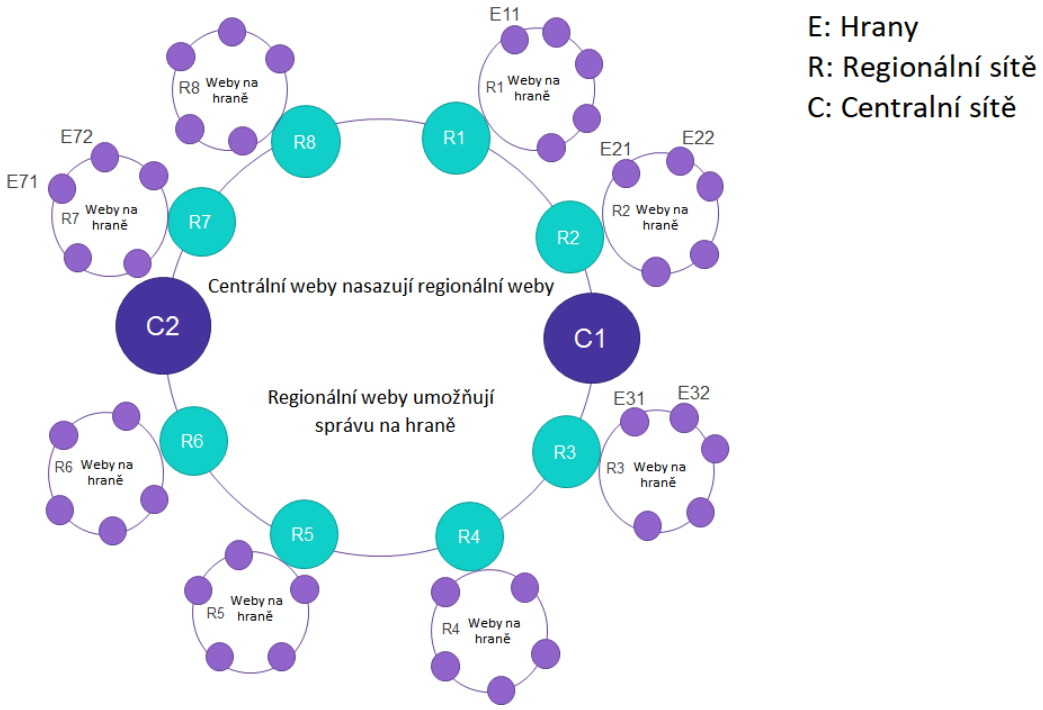
Obrázek 6: Fáze projektů LF Edge, převzato a přeloženo z [13]

3. Akraino

Akraino je soubor otevřených infrastruktur a aplikačních plánů (blueprints), zahrnující velkou škálu použití (například pro 5G, AI, IoT, IaaS/PaaS Edge), a to jak pro poskytovatele síťových infrastruktur, tak pro podnikové okrajové domény. Aplikační plány Akraina jsou deklarativní konfigurací celého zásobníku (cloudové platformy, API a aplikací). Tyto plány se zaměřují čistě jen na hranu ve všech možných podobách. Akraino podporuje chod ve virtualizovaném prostředí (VM), běh v kontejneru (docker) či rovnou nasazené na operačním systému (bare metal) [2][21].

Akraino [2] je projekt iniciovaný společnostmi AT&T a Intel. Cílem je vyvinout infrastrukturu a plně integrované řešení zcela zaměřené na Edge Computing. Tento software s otevřeným zdrojovým kódem (open source) poskytuje kritickou infrastrukturu, která umožňuje vysoký výkon, snižuje zpoždění, zlepšuje dostupnost, snižuje režijní náklady, poskytuje škálovatelnost, řeší zabezpečení a zlepšuje správu poruch.

Hierarchie nasazení Akraina je podobná rozdělení centrálních a regionálních webových aplikací. Centrální web nasazuje kolekci několika regionálních webů a ty nasazují weby na hraně. **Obrázek 7:** Zobrazení nasazení Akraina, převzato a přeloženo z [2] ukazuje centrální servery C1 a C2, které umožňují správu regionálních webů R1, R2, R3 a R4. Regionální weby umožňují správu webů na hraně, které jsou blíže k uživatelům [2].



Obrázek 7: *Zobrazení nasazení Akraina, převzato a přeloženo z [2]*

3.1 Schválené plány Akraina

Na oficiální webové stránce Akraina [22] autoři uvádějí osmnáct plánů, které mají ještě samostatně určité zaměření či podkategorie. Zde jsou vypsány všechny plány/projekty:

- 5G MEC System Blueprint Family
- AI/ML and AR/VR applications at Edge
- Connected Vehicle Blueprint(Aka CVB)
- Edge Video Processing
- Integrated Cloud Native NFV/App stack family (Short term: ICN)
- Integrated Edge Cloud (IEC) Blueprint Family
- IoT Area
- KubeEdge Edge Service Blueprint
- Kubernetes-Native Infrastructure (KNI) Blueprint Family
- MEC-based Stable Topology Prediction for Vehicular Networks
- MicroMEC
- Network Cloud Blueprint Family
- Public Cloud Edge Interface (PCEI) Blueprint Family
- StarlingX Far Edge Distributed Cloud
- Telco Appliance Blueprint Family
- The AI Edge Blueprint Family
- Time-Critical Edge Compute

V této bakalářské práci je pro výzkum použit projekt z IoT Area, přesněji "ELIOT: Edge Lightweight and IoT Blueprint Family", zkráceně ELIOT.

3.2 ELIOT

Plán ELIOT se zaměřuje především na to, aby okrajový uzel byl nenáročný softwarový zásobník, který lze nasadit na okrajové uzly mající omezenou kapacitu hardwaru a omezený operační systém [23].

Kromě toho se ELIOT také zaměřuje na infrastrukturu pro Edge Computing, která umožní vysoký výkon, vysokou dostupnost, zabezpečení a snížení zpoždění [23].

ELIOT rozděluje dva případy použití. V této bakalářské práci je použit pouze jeden, IoT Gateway, a to z důvodu, že je pro tuto práci vhodnější [23].

3.2.1 IoT Gateway

Mnoho různých podnikových aplikací potřebuje jednotnou bránu IoT. IoT Gateway podporuje aplikace snímačů IoT, OPC jednotnou architekturu přes TSN aplikace a kamerové aplikace s umělou inteligencí [18].

4. EdgeX Foundry

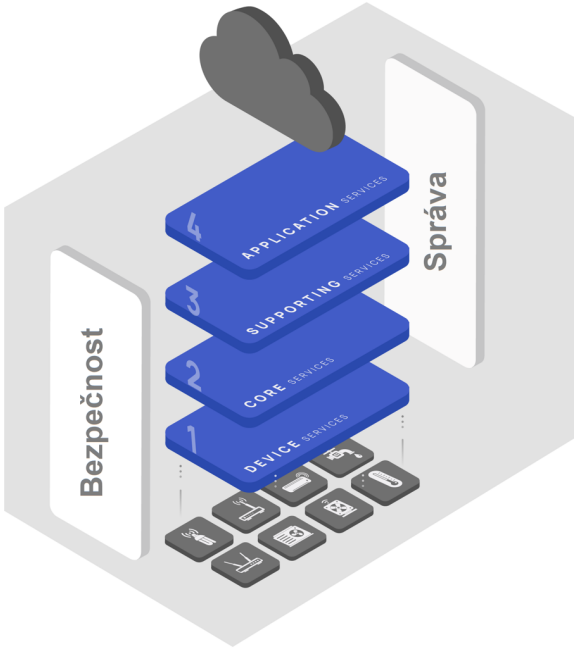
EdgeX je rámec otevřeného kódu, který je vysoce flexibilní a usnadňuje interoperabilitu mezi heterogenními zařízeními a aplikacemi IoT Edge. Díky otevřené platformě se tak vývojářům a poskytovatelům zrychluje čas na uvedení na trh tím, že poskytuje modulární referenční služby pro normalizaci, příjem dat zařízení, podporu nových datových služeb IoT a pokročilých aplikací pro Edge Computing [24].

Tento projekt byl spuštěn v rámci Linux Foundation a iniciativa je zaměřena na zjednodušení a standardizace Edge Computing v průmyslovém trhu IoT. EdgeX je platforma Alpha-grade založená na více než 125.000 řádcích kódu od společnosti Dell s referencemi na další otevřené zdroje [25].

EdgeX [26] překládá a posílá informace/data ze senzorů či jiných zařízení do aplikací přes síťové protokoly ve formátu a strukturách, které požaduje klient. Také umožňuje posílat data z aplikací do zařízení na hraně pro potřeby aktualizací, kontroly a ovládání.

Obsahuje 4 základní vrstvy, také nazvané jako služby:

- Device Services (Služby zařízení)
- Core Services (Služby jádra)
- Supporting Services (Podpůrné služby)
- Application Services (Aplikační služby)



Obrázek 8: *Základní vrstvy EdgeX, převzato a přeloženo z [26]*

4.1 Device Services

Do této služby řadíme konektory, senzory a IoT zařízení na hraně. Jsou to například roboti, drony a kamery. Jsou zde předem definované konektory pro ovládání zařízení a získání nebo odeslání dat. Uživatel si také může vytvořit vlastní službu pomocí SDK. Definováno je MQTT, Modbus, REST, SNMP, Bluetooth, IP kamery, virtuální simulace, BACnet IP a MSTP. Programovací jazyk je buď GO nebo C [26].

4.2 Core Services

Zde je uložena většina znalostí o tom, jak jsou jednotlivá zařízení propojena, jaká data protékají a jak je EdgeX nakonfigurováno v daném nasazení. Dostupné služby v této vrstvě jsou:

- Core Data – Centralizovaná databáze pro data od zařízení a senzorů [26].
- Core Command/Control – Povoluje příkazy/akce na zařízeních jménem jiných aplikací či externích systémů [26].
- Core Metadata - Je využívána ostatními službami ke znalosti daného zařízení a jak s ním komunikovat [26].
- Configuration and Registry – Centralizuje a zjednodušuje konfigurační data služby. Používá projekt Consul, díky kterému může uživatel přistupovat k EdgeX pomocí REST API [26].

4.3 Supporting Services

Zahrnuje mikroslužby, například okrajová analytika a typické softwarové aplikace, jako je přihlašování, plánování a mazání dat. Služby této vrstvy:

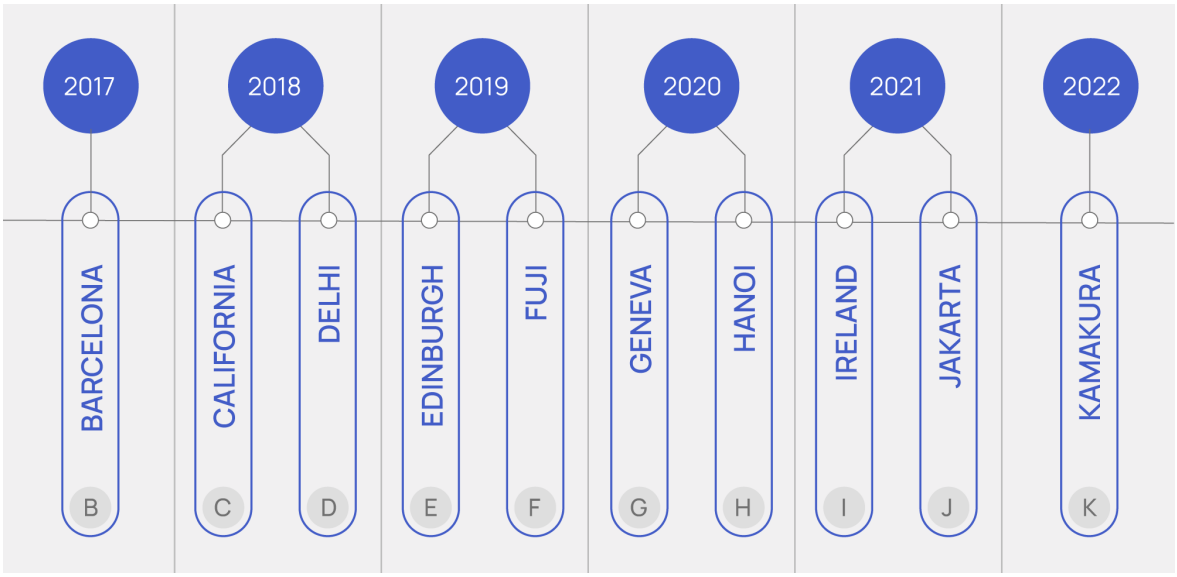
- Alerts and Notification – Poskytuje notifikace systému nebo uživateli [26].
- Rules Engine (Kuiper) – Umožňuje uživatelům realizovat rychlé zpracování dat na hraně a psát vlastní pravidla v SQL [26].
- Scheduling Service – Provádí předem konfigurované operace [26].

4.4 Application Services

Touto vrstvou je myšlena extrakce, zpracování/transformace a odeslání nasnímaných dat do koncového zařízení nebo aplikace. Tím může být analytický balíček, podnik, místní aplikace

nebo cloudový systém jako Azure IoT Hub, AWS IoT nebo Google IoT Core. Služby této vrstvy jsou celkem tři. Configuramble Application Services, Application Services a Additional Services[2] [26].

EdgeX je vyvíjena od roku 2017. Verze tohoto projektu se značí dle anglické abecedy (začátek od B) podle měst. Většinou jsou vydány dvě verze za rok, ačkoliv aktuálně nejpropagovanější verze, alespoň pro Raspberry Pi, je z roku 2020, a to Geneva [27][28].



Obrázek 9: *Verze EdgeX, převzato z [27]*

[2] V době psaní (19.3.2021) mají na stránkách chybně uvedený popis služeb.

5. Realizace úlohy

V následujících dvou podkapitolách je možné vidět, jaký hardware a OS byl použit k instalaci jednotlivých frameworků a k realizování testovací úlohy.

5.1. Hardware

Pro účely této bakalářské práci je použito Raspberry Pi 3 model B+ z roku 2017. Tento „počítač“ je osazen procesorem Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC@ a nataktován na 1.4GHz. Operační paměť má 1GB a jde o typ LPDDR2 SDRAM. ROM pamětí není osazeno, nicméně má vstup na SD kartu, která je v tomto případě Kingston 32GB microSD HC. Disponuje bezdrátovým připojením Wi-Fi 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac a Bluetooth 4.2. Dále HDMI portem, čtyřmi USB 2.0 porty, ethernetovou přípojkou o maximální rychlosti 300 Mbps, CSI konektor pro připojení Raspberry Pi kamery a DSI portem pro připojení Raspberry Pi displeje. Dále na Raspberry Pi nalezneme 40-pinový GPIO port a zdířku pro napájení 5V/2.5A DC [29].

Pro zobrazení je přes DSI port připojen sedmipalcový dotykový displej od Raspberry Pi. Jeho rozlišení je 800x480 pixelů a podporuje dotyk až deseti prstů naráz [30].

5.2. Operační systém

Operační systém pro Raspberry Pi jsem kvůli EdgeX vybral Ubuntu 20.10 (Groovy Gorilla) založený na linuxovém jádře Linux 5.8.0-1017-raspi pod architekturou arm64. Pro ELIOT jsem vybral Ubuntu 16.04. Jelikož k instalaci ELIOTu je zapotřebí více počítačů, pro tyto virtuální počítače byl opět použit systém Ubuntu 16.04.

```
pi@pi-desktop:~$ cat /etc/os-release
NAME="Ubuntu"
VERSION="20.10 (Groovy Gorilla)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.10"
VERSION_ID="20.10"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=groovy
UBUNTU_CODENAME=groovy
pi@pi-desktop:~$
```

Obrázek 10: *Operační systém v Raspberry Pi*

Ačkoliv pro Raspberry Pi existuje jejich vlastní operační systém Raspbian Pi OS [31], pro účely této bakalářské práce není relevantní, protože ne všechny služby a knihovny jsou pro něj dostupné, a tak ho nelze použít. Proto jsem zvolil systém od Ubuntu, neboť je podporovaný pro oba frameworky a je doporučován.

6. Instalace frameworků

V níže uvedených podkapitolách popisuji instalaci zvolených frameworků pro Edge Computing. Prvně je popsána instalace ELIOTu a závěr instalace. Dále pak instalace EdgeX a shrnutí celé instalace. Následující kapitoly líčí přípravu pro nasazení na úlohu, rozšíření testovací úlohy, srovnání použití a celkové porovnání společností a jejich frameworku subjektivním pohledem. Celá práce je poté závěrem s doporučením k použití.

6.1 Instalace ELIOTu

Instalace ELIOTu je rozdělena podle verzí. Aktuálně se na jejich stránkách [32] nacházejí čtyři verze (Release 1-4). Pro účely této práce a výzkum jsem vyzkoušel všechny verze tohoto projektu.

6.1.1. Release 1

Podle [33] je k instalaci potřeba zprovoznit ELIOT Manager (osobní počítač s virtuálním operačním systémem Ubuntu) a ELIOT Edge Node, což představuje Raspberry Pi.

Minimální hardwarové požadavky na instalaci ukazuje **Tabulka 2**: Minimální hardwarové požadavky Release 1, převzato a přeloženo z [33]

Požadavky/Hardware	ELIOT Manager	ELIOT Edge Node
CPU	Minimálně 1 socker x86_AMD64 nebo ARM64 s podporou virtualizace	Minimálně 1 socker x86_AMD64 nebo ARM64
RAM	4 GB	1 GB
Disk	120 GB - 512 GB	20 GB - 256 GB
Síťová karta	1	1

***Tabulka 2**: Minimální hardwarové požadavky Release 1, převzato a přeloženo z [33]*

Tyto požadavky použitý hardware splňuje. Softwarových požadavků před samotnou instalací je celkem devět:

- Virtuální počítač s předinstalovaným Ubuntu 16.04

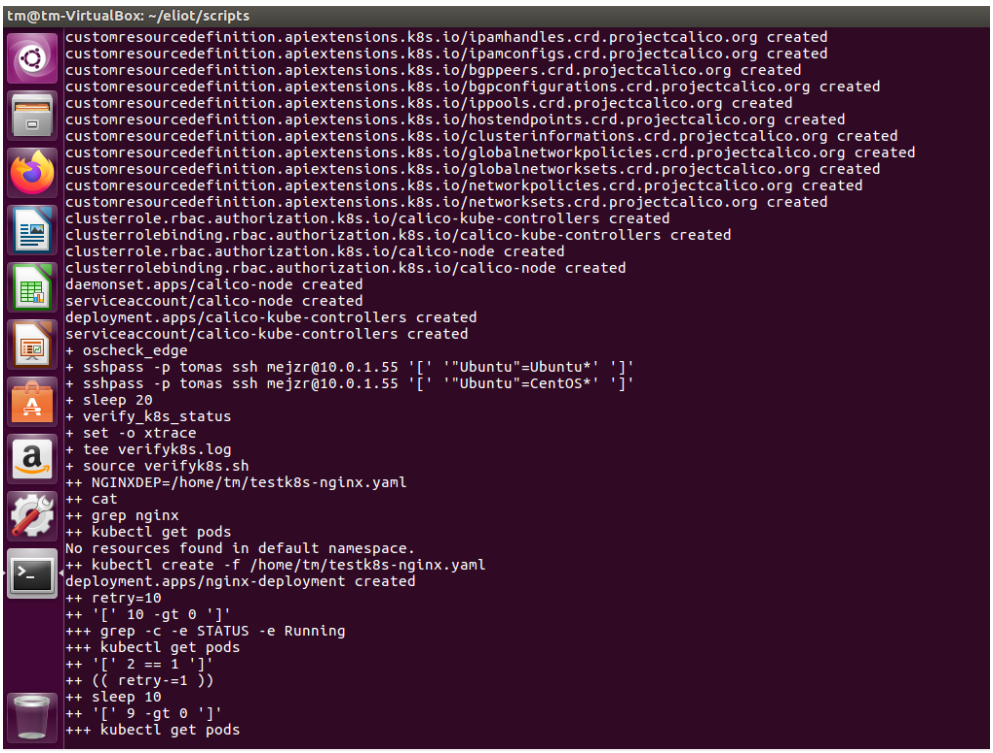
- Root uživatel vytvořen jak v ELIOT Manageru, tak v Node
- SSH server na obou zařízeních
- Sshpass nainstalovaný na Manageru
- SCP na oběma zařízeními
- Programovací jazyk GO na obou zařízeních
- Git
- Internetové připojení jak na Manageru, tak na Node
- Obě zařízení jsou na stejné síti a vidí se

Tyto potřebné softwarové požadavky jsem splnil. Akraino dává na výběr, zda chce uživatel použít instalaci přes Kubernetes nebo KubeEdge. Zvolil jsem instalaci přes Kubernetes, jelikož je popsána jako první. Podle návodu jsem naklonoval ELIOT z gerritu a ve složce **scripts** jsem upravil soubor **nodelist**. Tento soubor v sobě obsahuje jméno, heslo a ip adresu Raspberry Pi a je v instalačním skriptu použit na instalaci ELIOT Edge Node. Po upravení a uložení jsem spustil samotný skript **setup.sh**, který slouží pro instalaci. Instalace trvala několik minut a občas vyžadovala administrátorské heslo. Zde se nicméně objevila první komplikace. Instalace ELIOT Manageru vyžaduje alespoň dvě jádra procesoru. Změnil jsem tedy nastavení virtuálního počítače z jednoho jádra na šest a pomocí spuštění dvou předem připravených skriptů jsem vše smazal a spustil instalaci znovu. Ani po zvýšení jader se instalace nepovedla. Dle instalačního skriptu se na konci instalace má objevit „Success!“, což se v mém případě nestalo. **Obrázek 11:** Instalace Release 1, pokus 1 zobrazuje můj instalační pokus.

```
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
serviceaccount/calico-node created
deployment.apps/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
+ setup_k8sworkers
+ set -o xtrace
+ SETUP_WORKER_COMMON='sudo rm -rf ~/eliot &&                    git clone https://gerrit.akraino.org
/r/eliot &&                    cd eliot/blueprints/iotgateway/scripts/ && source common.sh'
+ SETUP_WORKER='cd eliot/blueprints/iotgateway/scripts/ && source k8sworker.sh'
++ kubeadm token create --print-join-command
W0214 10:46:30.730269   29160 validation.go:28] Cannot validate kube-proxy config - no validator is avai
lable
W0214 10:46:30.730318   29160 validation.go:28] Cannot validate kubelet config - no validator is availab
le
+ KUBEADM_TOKEN='kubeadm join 10.0.1.67:6443 --token 0x46yo.x4ehc714kscazerf     --discovery-token-ca-ce
rt-hash sha256:d574faca3477be4267a30aea55ef9624f2dccced88ff4a6f64d77d9809c36e71 '
+ KUBEADM_JOIN='sudo kubeadm join 10.0.1.67:6443 --token 0x46yo.x4ehc714kscazerf     --discovery-token-c
a-cert-hash sha256:d574faca3477be4267a30aea55ef9624f2dccced88ff4a6f64d77d9809c36e71 '
```

Obrázek 11: *Instalace Release 1, pokus 1*

Důvod, proč se mi ELIOT nenainstaloval korektně, byl ten, že se vynechala instalace ELIOT Edge Node. Přepsal jsem tedy instalační skript, kde jsem odstranil ověřování operačního systému a nastavil jsem instalaci Edge Node rovnou. Také jsem nastavil jméno, heslo a IP adresu Raspberry Pi přímo do skriptu. Instalace se opět nezdařila, nicméně skript zobrazil jiný výsledek, který je znázorněn na **Obrázek 12:** Instalace Release 1, pokus 2.



Obrázek 12: *Instalace Release 1, pokus 2*

Jelikož se mi nepodařilo odstranit příčinu špatné instalace, rozhodl jsem se poslat email Khemendrovi Kumarovi, který je na stránkách [32] veden jako jeden z hlavních členů vývojářského týmu ELIOT. Obdržená odpověď ve zkratce říkala, že Release 1 je již zastaralý a mám zkusit nejnovější Release 4. Během čekání na odpověď jsem nicméně vyzkoušel Release 2/3 (Release 3 je přesměrován na Release 2).

6.1.2. Release 2/3

Jak již bylo uvedeno výše, Release 2 a 3 jsou vlastně stejné, neboť třetí verze odkazuje na druhou. Většina úkonů je zde shodná jako u Release 1. Minimální požadavky na instalaci jsou totožné, softwarové požadavky jsou stejné, dokonce i klonování souborů z gerritu je identické pod stejnou adresou. První změna je tedy až v cestě, kde se nejde rovnou do složky **scripts**, ale do **blueprints/iotgateway/scripts**. Zde jsem opět přepsal obsah souboru **nodelist** (jméno, heslo a ip adresa ELIOT Edge Node). Jelikož veškeré hardwarové i softwarové požadavky byly splněny, začal jsem s instalací.

Stejně jako u Release 1, i zde instalace neproběhla správně a skončla chybně. Při pokusu o zjištění, co se nainstalovalo korektně, napsáním příkazu

```
kubectl get nodes
```

se zobrazil text, ze kterého bylo možné zjistit, že opět neproběhla instalace ELIOT Edge Node a chybově je nainstalován i ELIOT Manager. I přes mnohočetné opakování byl výsledek totožný.

```
tom@tom-VirtualBox:~/eliot/blueprints/iotgateway/scripts$ kubectl get nodes
NAME            STATUS     ROLES    AGE   VERSION
tom-virtualbox  NotReady   master   26s   v1.17.2
tom@tom-VirtualBox:~/eliot/blueprints/iotgateway/scripts$
```

Obrázek 13: *Release 2/3 zobrazení uzlů*

6.1.3. Release 4

Požadavky na instalaci Release 4 jsou jiné než u ostatních. ELIOT Manager se mění na ELIOT Master, ELIOT Edge Node se mění na ELIOT Iot Gateway node a přidává se Deployment node, také nazvaný jako Jump Host.

Minimální požadavky byly napsány pouze k ELIOT Masteru a Iot Gateway a zobrazuje je **Tabulka 3:** Minimální požadavky Release 4, převzato a přeloženo z [34].

Požadavky/Hardware	ELIOT Master Node	IOTGateway Node
CPU	x86_AMD64 nebo ARM64	x86_AMD64 nebo ARM64
Počet CPU	8	4
RAM	8 GB	1 GB
Disk	120 GB - 512 GB	20 GB - 256 GB
Síťová karta	1	1

Tabulka 3: *Minimální požadavky Release 4, převzato a přeloženo z [34]*

Softwarové požadavky byly následující:

- Virtuální stroje s předinstalovaným Ubuntu 16.04/18.04
- Vytvořený root uživatel na všech strojích
- SSH server spuštěný na všech strojích
- Ansible> 2.5 a git nainstalovaný na Jump Hostu
- Stažený kubespray
- Programovací jazyk GO nainstalovaný na Jump Hostu
- Všechny stroje mají internetové připojení
- ELIOT Master Node a IoT Gateway jsou ve stejné síti a navzájem se vidí

Veškeré softwarové i hardwarové požadavky byly splněny, kromě jednoho. Můj počítač nemá dostatečný počet jader pro Eliot Master Node (fungují pouze fyzická, nikoliv logická).

Samotná instalace začíná prvně na Jump Hostu, kde se jako první musí vygenerovat veřejný ssh klíč. Ten se poté nakopíruje na ELIOT Master Node. Dále se musí provést konfigurace již zmíněného kubespraye: Nejprve se otevře složka se staženým kubesparyem a vytvoří se kopie složky **sample**, kde se bude nastavovat cluster. Po tomto kopírování přichází na řadu již zmíněná konfigurace, během které se musí upravit dva soubory:

- **inventory/mycluster/group_vars/all/all.yml**
- **inventory/mycluster/group_vars/k8s-cluster/k8s-cluster.yml**

Po jejich upravení se musí nakonfigurovat soubor **config.yml,** který by se měl nacházet pod cestou **eliot/blueprints/iotgateway/playbooks/config.yml**. Jenže tento soubor v mé instalaci chybí a nevěděl jsem, jak přesně nakonfigurovat předchozí dva soubory.

6.1.4. Komunikace a závěr instalace ELIOTu

Má komunikace se zmíněným Khemedrou Kumarem neměla dlouhého trvání. Po dvou dnech jsem dostal odpověď, že je rád, že se o ELIOT zajímám a přeposílá dotaz na kolegu Srinivasana Selvama. Ten odpověděl po šesti dnech strohou odpovědí, že Release 1 už není aktuální a ať přejdu na Release 4. Po dvou dnech jsem odepsal, že nevím, co přesně upravit v souborech **k8s-cluster.yml** a **all.yml**, a že soubor **config.yml** mi zcela chybí. Odpověď přišla po 2 dnech, kde v podstatě jinak interpretoval návod na [34] a neodpověděl na otázku, co a kde přesně se musí změnit. Po opětovném tázání mi již žádná odpověď nepřišla.

Ačkoliv jsem se o instalaci snažil relativně dlouhou dobu (s pauzami od prosince do března), nepodařilo se mi to. Ptal jsem se na oficiálním fóru i na různých skupinách, zda se někomu zmíněný ELIOT povedlo nainstalovat, ale odpovědi se mi nedostalo. Také jsem se pokoušel o instalaci přes KubeEdge, ale také bez výsledku. Po poradě s vedoucí této bakalářské práce jsem se rozhodl v tomto projektu dále nepokračovat.

6.2 Instalace EdgeX

Oproti ELIOTu je EdgeX rozdělena na instalaci pomocí dockeru a pomocí snapu.

Ačkoliv nasazení EdgeX přes snap by vše značně ulehčilo, pro instalaci na Raspberry Pi je instalace popsána přes docker pro verzi Geneva. I přesto, že podle [27] Geneva není nejnovější verzí, v návodech je stále doporučována a je nejvíce popsána. I to je důvod, proč se v této práci používá právě tato verze.

Na [35] je instalace sice pro Raspberry Pi 4, nicméně pro tyto účely by mělo stačit i Raspberry Pi 3B+. Instalace je rozdělena do čtyř návodů, přičemž důležitý je druhý a třetí. První návod obsahuje instrukce pro instalaci balíčků potřebných pro nainstalování EdgeX a druhý je již samotná instalace EdgeX.

6.2.1. Instalace požadovaných balíčků

Návod [36] uvádí, že jako první se musí nastavit hostname jméno a správná časová zóna. Provede se restart a přes sudo apt se aktualizují balíčky. Poté je zapotřebí nainstalovat základní balíčky nezbytné pro správný chod a kontrolu úspěšné instalace EdgeX. Jedná se o tyto balíčky

- jq
- vim
- git
- tmux
- curl
- tree
- make
- libzmq3-dev
- gnupg-agent
- build-essential
- ca-certificates
- apt-transport-https
- software-properties-common

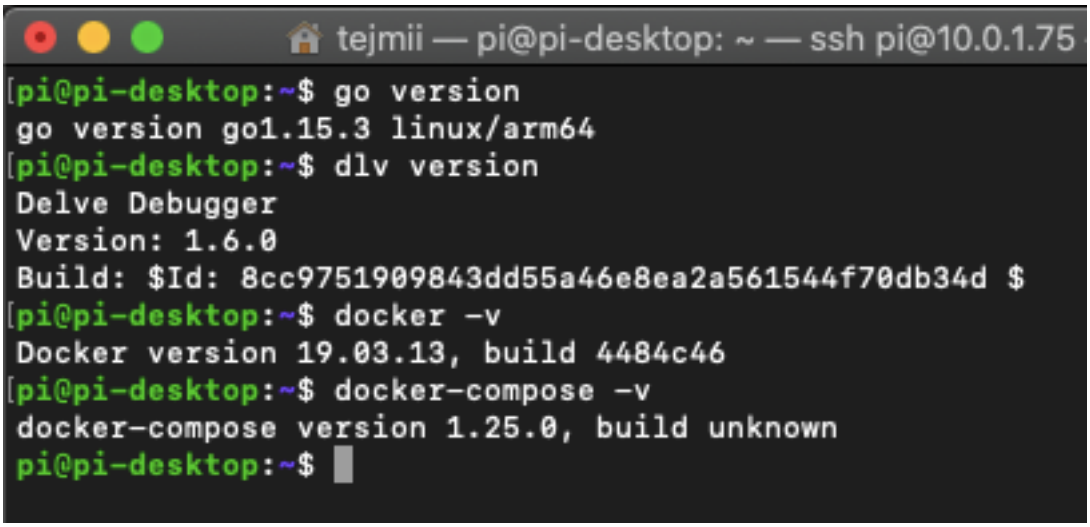
Po úspěšném nainstalování všech výše zmíněných balíčků se musí nainstalovat programovací jazyk GO, debugovací nástroj Delve, samotná kontejnerová platforma Docker a její rozšíření Docker-compose. Podle návodu jsem vytvořil složku **go**, přes **wget** stáhl potřebnou verzi programovacího jazyku GO a rozbalil ji do zmíněné složky. Dalším krokem bylo nastavení cest pro jazyk GO a instalace Delve. To se provedlo příkazy:

```
echo "export PATH=$PATH:/usr/local/go/bin:$HOME/go/bin" >> ~/.bashrc
echo "export GOROOT=/usr/local/go" >> ~/.bashrc
echo "export GOPATH=$HOME/go" >> ~/.bashrc
source ~/.bashrc
go get -u github.com/go-delve/delve/cmd/dlv
```

Nástroje Docker a Docker-compose se podle návodu [36] musí nejprve nainstalovat, povolit, spustit, přidat práva a restartovat počítač. V příkazech to vypadá takto:

```
sudo apt install -y docker.io docker-compose
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker ${LOGNAME}
sudo reboot
```

Jelikož se mi vše povedlo, přesunul jsem se na samotnou instalaci EdgeX podle [28].



Obrázek 14: *Kontrola správně nainstalovaných balíčků EdgeX*

6.2.2. Samotná instalace EdgeX

Jako první se podle [28] musí vytvořit složka, která bude sloužit jako repozitář všech dostupných verzí, které jsou uloženy v developerských skriptech. Ty se získají pomocí klonování z platformy git. Po korektním naklonování jsem vstoupil do složky se všemi

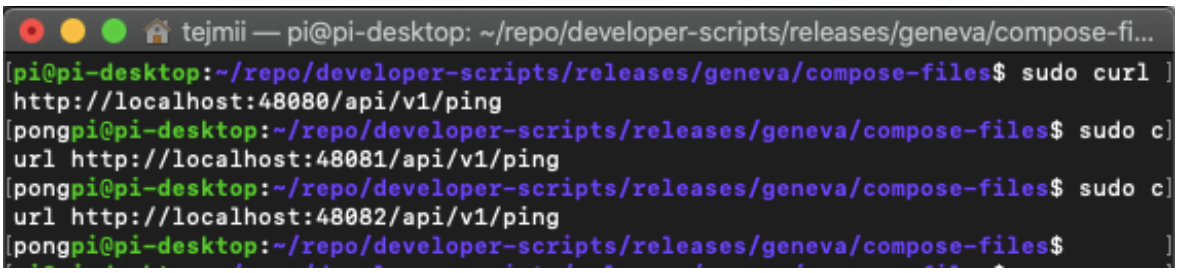
developerskými skripty **/repo/developer-scripts/releases/geneva/compose-files**. Zde se nachází celkem jedenáct souborů, z nichž jsou podle návodu důležité především dva:

- docker-compose-geneva-redis-no-secty-arm64.yml
- docker-compose-portainer.yml

U prvního, jak již název napovídá, se jedná o verzi Geneva. Redis je databáze, no secty znamená, že nemá žádné zabezpečení (z důvodu omezeného výkonu Raspberry) a ARM64 je architektura. Druhý skript je nástroj pro zjednodušenou a grafickou práci s dockerem. Spuštění těchto skriptů se provede dvěma příkazy:

```
docker-compose -f docker-compose-geneva-redis-no-secty-arm64.yml up -d
docker-compose -f docker-compose-portainer.yml up -d
```

Zde se mi vyskytl problém označený COMPOSE_HTTP_TIMEOUT. Jelikož Raspberry Pi nemá dostatek výkonu na provedení celého skriptu v časovém limitu, který je automaticky definován na 60 vteřin. Přidání **COMPOSE_HTTP_TIMEOUT=600** před výše zmíněné příkazy tento omezený čas změnilo na 600 vteřin, a díky tomu se vše nainstalovalo. O korektní instalaci jsem se přesvědčil postupným zasláním příkazů, kdy každý příkaz vrátil pong. Tyto příkazy a jejich odpovědi zobrazuje **Obrázek 15:** Kontrola správného nainstalování EdgeX



Obrázek 15: *Kontrola správného nainstalování EdgeX*

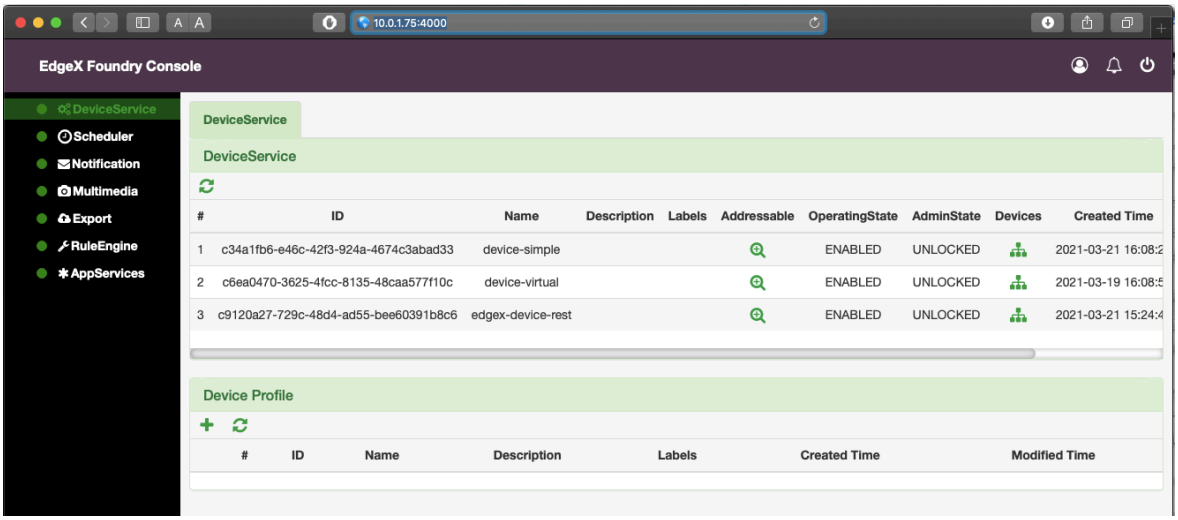
6.2.3. Úprava instalace

Ačkoliv samotná instalace se mi podařila relativně bez problému, má dva zásadní nedostatky. Prvním je, že připojení z Raspberry Pi je sice v pořádku, ale z jiného počítače na lokální síti se na budoucí potřebné adresy nedostanu. Druhým nedostatkem je pak absence grafického rozhraní, kterým EdgeX disponuje [37].

První nedostatek jsem vyřešil tak, že v původním skriptu **docker-compose-geneva-redis-no-secty-arm64.yml** jsem přepsal IP adresu všech potřebných služeb ze 127.0.0.1 na

0.0.0.0. Díky této adrese je zajištěn přístup ze všech počítačů na lokální síti. To stejné muselo být provedeno i u portaineru.

Druhý nedostatek řeší skript **docker-compose-geneva-ui-arm64.yml**, kde jsem také musel přepsat adresu 127.0.0.1 na 0.0.0.0. Tento skript jsem pak jen spustil a po provedení ověřil, zda se webové rozhraní správně nainstalovalo a jestli je spuštěné.



Obrázek 16: *Grafické rozhraní EdgeX*

6.2.4. Závěr instalace EdgeX

Instalace samotného EdgeX proběhla v pořádku bez větších komplikací. Jejich návody k instalaci jsou psané přehledně a vysvětlují, co se vlastně v následujícím kroku bude dít. Návody je možno nalézt jak na samotných oficiálních stránkách, tak na githubu, kde je možnost přečíst si i komentáře ostatních uživatelů.

Třebaže dle návodů je všechno v pořádku, ve webovém rozhraní po kliknutí na Notification, Multimedia, Export a RuleEngine tyto služby nejsou dostupné. Pravděpodobně se jedná o potíže na různých architekturách a instalací. Toto jsem si ověřil nainstalováním EdgeX na virtuální stroj, a díky tomu vím, že na architektuře x86 funguje například Export, ale nefunguje AppServices.

7. Základní testovací úloha

Testovací úlohou, kterou jsem zvolil v této bakalářské práci, je možnost odesílání dat, například ze senzoru do frameworku. Senzor může být pro představu teplotní či vlhkostní. Záměr byl odesílat a zpětně číst zaslaná naměřená data. Jelikož jsem nedisponoval senzory, které by teplotu a vlhkost měřily, napsal jsem vlastní skript v pythonu. Ten náhodně generuje teploty od 20 do 35 stupňů celsia a vlhkost od 60 do 81 %. Data odesílá každých 5 vteřin. Tento a další skripty či jiné potřebné soubory jsou k nalezení na mém github repozitáři (dále jen repozitář), který je dostupný pod touto adresou: **https://github.com/tejmii/bakalarska_prace**.

Také jsem odebíral data zaslaná z reálných senzorů, které mi posílal kolega David Dlouhý v rámci spolupráce s jeho ročníkovou prací. Jedná se o data ze senzoru vibrací a PIR senzoru (pohybové čidlo).

Jelikož se nepodařilo nainstalovat ELIOT, v dalších kapitolách a podkapitolách se text této práce bude věnovat pouze samotnému EdgeX.

7.1. Zprovoznění testovací úlohy na platformě EdgeX

Zprovoznění testovací úlohy vychází z návodu [38]. Tato instalace se rozděluje na čtyři hlavní kroky:

- **Vytvoření deskriptoru hodnot** – Říká EdgeX v jakém formátu data přijdou a jak jsou označena.
- **Vytvoření profilu zařízení (profil senzoru)** – Profil zařízení je v podstatě šablona, která popisuje jeho datové formáty a podporované příkazy. Musí být napsaný ve formátu .yaml.
- **Vytvoření samotného zařízení (senzoru)** – Jedná se o vytvoření samotného zařízení. Po vytvoření je již možné zasílat jednotlivá data.
- **Zasílání a získání dat**

K prvnímu, třetímu a čtvrtému kroku je zapotřebí použít software zvaný Postman. Díky Postmanu uživatel může rychle a lehce použít REST a SOAP operace [39].

7.1.1. Vytvoření deskriptoru hodnot

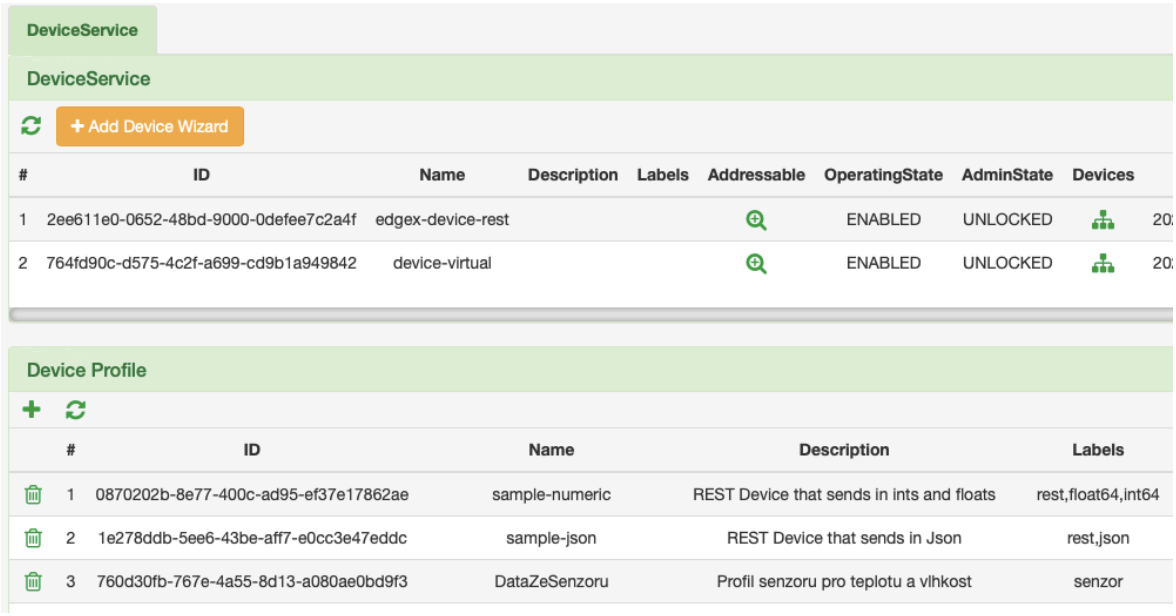
V aplikaci Postman jsem zvolil metodu POST, URL je adresa EdgeX s přidaným portem a cestou. Raspberry Pi mělo URL **http://10.0.1.75:48080/api/v1/valuedescriptor**. Dále jsem v záložce body nastavil „raw“ a „JSON“. Jako data se vloží již samotný deskriptor hodnot. Je potřeba zaslat deskriptor celkem čtyřikrát (pro teplotu, vlhkost, vibrace a pir). Níže je zobrazen jeden deskriptor, všechny jsou k nalezení v repozitáři.

```
{
"name": "vlhkost",
"description": "Cas + vlhkost v procentech",
"type": "String",
"uomLabel": "vlhkost",
"defaultValue": "0",
"formatting": "%s",
"labels": [
"senzor",
"senzory"
]
}
```

Pokud vše proběhne v pořádku, zobrazí se kód „Status 200 OK“, což byl i můj případ.

7.1.2. Vytvoření profilu zařízení

Vytvoření profilu zařízení je možné provést dvěma způsoby. První je opět přes aplikaci, druhý je nahrání profilu přes webové rozhraní. Já jsem zvolil druhý způsob, tedy nahrání přes webové rozhraní. Nejprve jsem otevřel webové rozhraní přes internetový prohlížeč zadáním příslušné URL adresy a portu (URL Raspberry Pi byla **http://10.0.1.75:4000**). Poté jsem v levém menu vybral „DeviceService“, v něm jsem pod kategorií „Device Profile“ klikl na + a zvolil konfigurační soubor s koncovkou .yaml. Tento soubor je k zobrazení v mém repozitáři. Po správném nahrání se profil zobrazil pod „Device Profile“, jak ukazuje autorský **Obrázek 17:** Zobrazení správného nahrání profilu zařízení.



Obrázek 17: *Zobrazení správného nahrání profilu zařízení*

7.1.3. Vytvoření samotného zařízení

K vytvoření samotného zařízení byla opět použita aplikace Postman. Znovu se jako u deskriptoru hodnot vybrala metoda POST, v záložce body „raw“ a „JSON“. Za URL adresu Raspbery Pi se dopíše :48081/api/v1/device. Data pro vytvoření samotného zařízení jsem tedy zasílal na adresu **http://10.0.1.75:48081/api/v1/device**. Samotné zařízení v datech vypadalo takto:

```
{
"name": "Senzory",
"description": "Data zaslana pres URL",
"adminState": "unlocked",
"operatingState": "enabled",
"protocols": {
"example": {
"host": "dummy",
"port": "1234",
"unitID": "1"
}
},
"labels": [
"senzor",
"senzory"
],
```

```
"location": "Liberec",
"service": {
"name": "edgex-device-rest"
},
"profile": {
"name": "DataZeSenzoru"
}
}
```

Po jejich zaslání se mi opět zobrazil kód „Status 200 OK“.

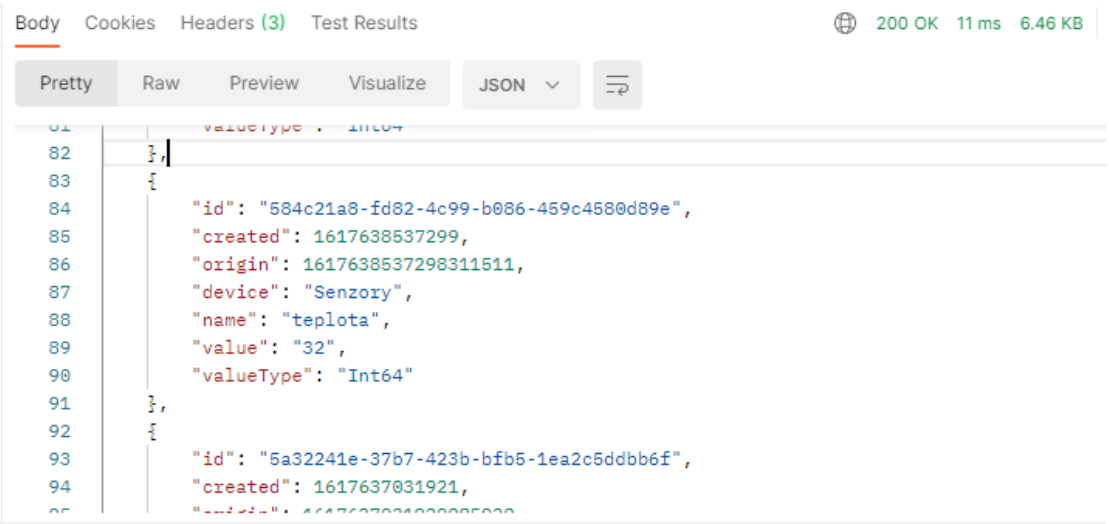
7.1.4. Zasílání dat a opětovné získání

K zasílání dat jsem využil již zmiňovaný software Postman. K zasílání jednotlivých dat ze senzoru se musí zvolit metoda POST, body se nastaví na „raw“ a „text“. Samotná URL adresa slouží k identifikaci jednotlivého senzoru.

- **http://10.0.1.75:49986/api/v1/resource/Senzory/teplota**
- **http://10.0.1.75:49986/api/v1/resource/Senzory/vlhkost**
- **http://10.0.1.75:49986/api/v1/resource/Senzory/vibrace**
- **http://10.0.1.75:49986/api/v1/resource/Senzory/pir**

Zpráva již obsahuje jednotlivé informace ze senzorů, v mém případě je to buď teplota, vlhkost, vibrace nebo pir. Tato zaslaná data jsem pak doplnil o datum a čas, z důvodu, který je popsán v kapitole Rozšíření testovací úlohy.

Pro opětovné získání dat, neboli čtení, se v aplikaci Postman nastavuje pouze metoda a URL. Místo POST se použije GET a URL se přepíše na patřičný port a cestu. Adresa čtení z Raspberry Pi byla **http://10.0.1.75:48080/api/v1/reading**. Pokud vše funguje správně, zobrazí se veškerá data tak, jak ukazuje můj **Obrázek 18:** Zobrazení dat v aplikaci Postman.



Obrázek 18: *Zobrazení dat v aplikaci Postman*

Dat, která se vyskytují pod touto adresou, je mnoho, neboť se generují i náhodná data z důvodu, že EdgeX má od samotného základu zapnuté generování náhodných čísel, booleanů a dalších. Proto je potřeba použít funkci hledání, která se dá zapnout kombinací kláves **ctrl+f**.

K získání dat se nicméně nemusí využívat žádná aplikace. Stačí do internetového prohlížeče zadat patřičnou URL adresu a internetový prohlížeč zobrazí data ve formátu JSON sám.

8. Rozšíření testovací úlohy

Rozšířením testovací úlohy byl můj záměr pokusit se exportovat (odebírat) již získaná data o teplotě, vlhkosti, vibraci a pohybu tak, aby se pro zobrazování dat nemusela manuálně použít žádná aplikace, ale data se zobrazila automaticky. K připojení se může dle [40] použít aplikace Eclipse Mosquitto. Vycházet budu z předchozího návodu [38].

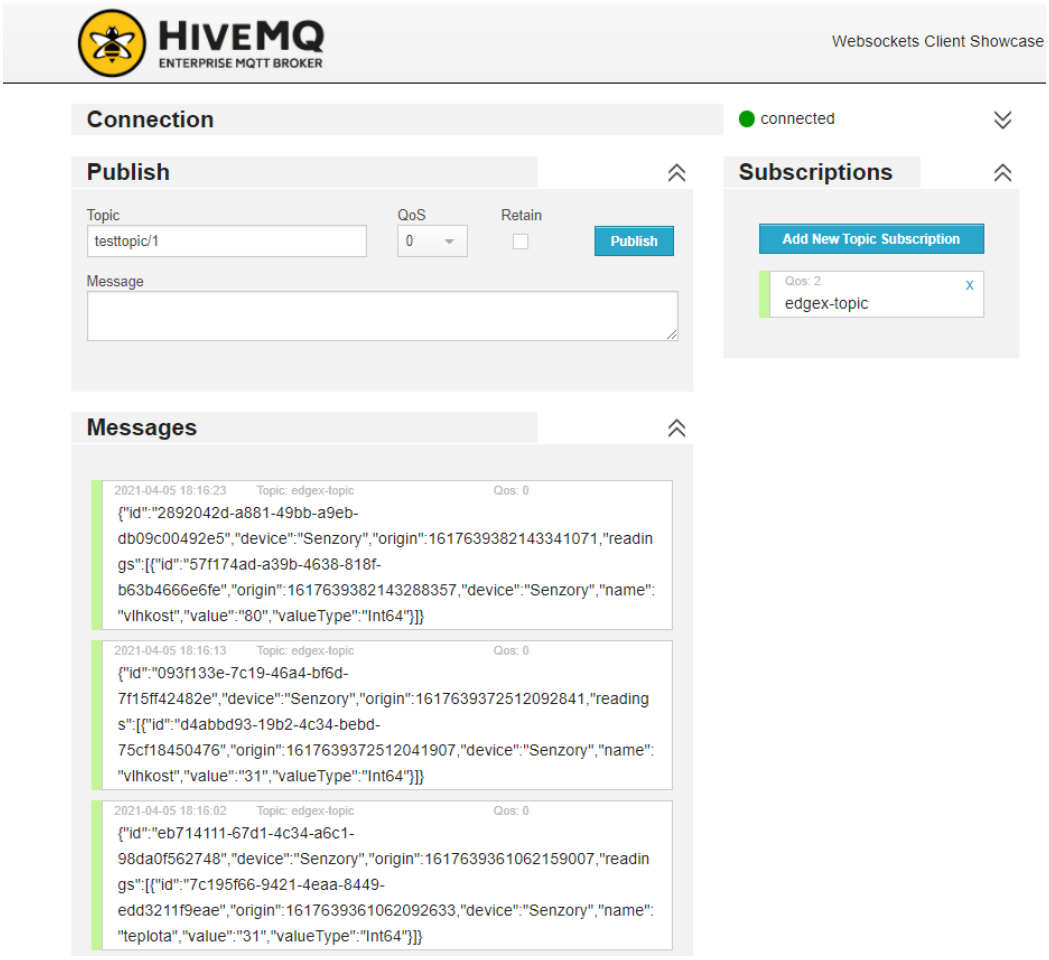
Eclipse Mosquitto je zprostředkovatel zpráv s otevřeným kódem. Implementuje MQTT protokol, který je nenáročný a může být použit na všech zařízeních od nízkoenergetických jednodeskových počítačů po plné servery. Protokol MQTT funguje na modelu publikovatel/odběratel, kde publikovatel zprávu posílá a odběratel ji přijímá [41].

8.1. Exportování dat

Pro exportování dat jsem musel přidat do původní konfigurace **docker-compose-geneva-redis-no-secty-arm64.yml** kus kódu, kterým se nastavil export přes MQTT. Kvůli jeho délce je k nalezení v repozitáři. Oproti původnímu kódu z [38] jsem musel pozměnit **„image“**, aby fungoval na procesoru architektury ARM64. Správnou změnu tohoto image jsem nikde nenašel a přišel na ní až dlouhodobým zkoumáním a používáním. Dále jsem změnil **„topic“**, díky kterému bude moct aplikace HiveMQ odebírat zaslaná data. Po uložení pozměněné konfigurace se musí skript opětovně spustit.

Po úspěšném spuštění jsem přešel na webovou stránku http://www.hivemq.com/demos/websocket-client/, kde se ponechala původní konfigurace. Připojil jsem se a pro odebírání dat jsem vložil stejný topic, jaký jsem nastavil v konfiguraci.

Pak už jsem jen spustil skript na generování hodnot teploty a vlhkosti. Tento skript byl zmíněn v kapitole Základní testovací úloha. Jelikož vše fungovalo korektně, mohl jsem vidět všechna odeslaná data.



Obrázek 19: *Zobrazení exportovaných (odebíraných) dat*

Tato odebíraná data jsou ve formátu JSON. Výhodou je, že data o naměřené vlhkosti, teplotě, pohybu či vibrací si můžeme zobrazit odkudkoliv, neboť HiveMQ je pro soukromé účely zdarma. Nevýhodou je, že k těmto datům je volný přístup a může tak hrozit jejich zneužití, což ale pro mou potřebu nevadilo. Doporučuji však používat vlastní MQTT server. Dále je vhodné zdůraznit, že export v tomto kontextu není celá historie měření, nýbrž data od doby, kdy jsme začali odebírat tyto naměřené hodnoty.

Pro odebírání dat nemusíme používat pouze oficiálního klienta HiveMQ, ale i doplněk do jednoho z nejpoužívanějších internetových prohlížečů, Google Chrome. Tento doplněk se jmenuje MQTTLens a funguje velmi podobně, jako právě již zmíněný oficiální klient. Také je možné si napsat vlastní skript, který tato data odebírá. Osobně jsem si takový skript napsal a je použit v následující kapitole.

8.2. Grafické zobrazení

Tato podkapitola je věnována zobrazení exportovaných dat o teplotě, vlhkosti, vibracích a pohybu v nějakém grafickém zobrazení. K tomu je potřeba použít databázi, a tu připojit ke grafickému rozhraní. K tomuto úkolu není žádný návod, a proto je popisování jednotlivých kroků čistě autentické.

Kvůli této úloze byl původní skript generující náhodně teplotu a vlhkost doplněn o datum a čas. Jelikož jsem chtěl data vizualizovat a zobrazovat tak, jak šla za sebou, potřeboval jsem mít informaci o tom, kdy se data vytvořila.

Pro grafické rozhraní jsem zvolil aplikace Grafana. Ta je zmíněna v [38] a její zobrazování vypadá velmi pěkně. Pro instalaci této aplikace bylo využito již zmíněného návodu a k nainstalování do dockeru jsou použity dva terminálové příkazy:

```
docker pull grafana/grafana
docker run -d --name=grafana -p 3000:3000 grafana/grafana
```

Pro ukládání dat do databáze jsem vybral známé MySQL. K instalaci této databáze není použit docker, nýbrž se instaluje do systému. Hlavní důvod, proč není použit docker, je ten, že MySQL nemá verzi přeloženou pro procesory architektury ARM. Instalace a konfigurace jsem provedl dvěma příkazy:

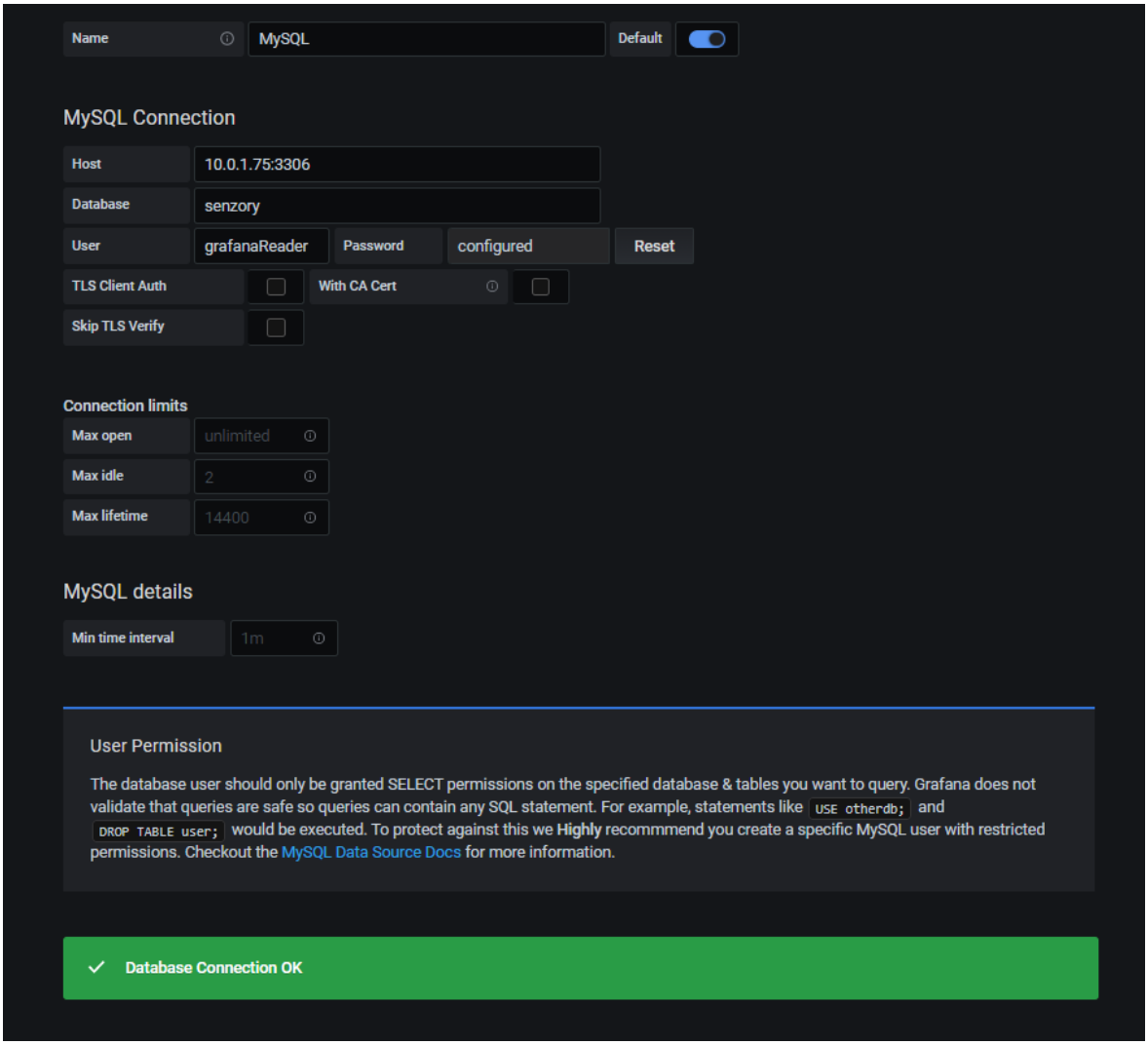
```
sudo apt install mysql-server
sudo mysql_secure_installation
```

V samotné aplikaci MySQL jsem vytvořil dva uživatele a databázi se čtyřmi tabulkami. Dva uživatele jsem vytvořil z důvodu, že jeden je pouze pro čtení a druhý pro čtení/zápis do databáze. Uživatel pro čtení slouží pro přístup do databáze pro Grafanu, jelikož s jinými právy se Grafana do databáze nepřipojí. Čtyři tabulky jsem vytvořil proto, aby každý senzor měl vlastní. Příkazy vypadaly takto:

```
CREATE USER ‘tom‘ IDENTIFIED BY ‘123456789‘;
CREATE USER 'grafanaReader' IDENTIFIED BY 'password';
CREATE DATABASE senzory;
CREATE TABLE teplota (id int NOT NULL AUTO_INCREMENT, date TIMESTAMP,
teplota  VARCHAR(255), PRIMARY_KEY (id));
CREATE TABLE vlhkost (id int NOT NULL AUTO_INCREMENT, date TIMESTAMP,
vlhkost VARCHAR(255), PRIMARY_KEY (id));
CREATE TABLE pir (id int NOT NULL AUTO_INCREMENT, date TIMESTAMP, pir
VARCHAR(255), PRIMARY_KEY (id));
CREATE TABLE vibrace (id int NOT NULL AUTO_INCREMENT, date TIMESTAMP,
vibrace VARCHAR(255), PRIMARY_KEY (id));
GRANT ALL PRIVILEGES ON senzory.* TO ‘tom‘;
```

```
GRANT SELECT ON senzory.* TO 'grafanaReader';
```

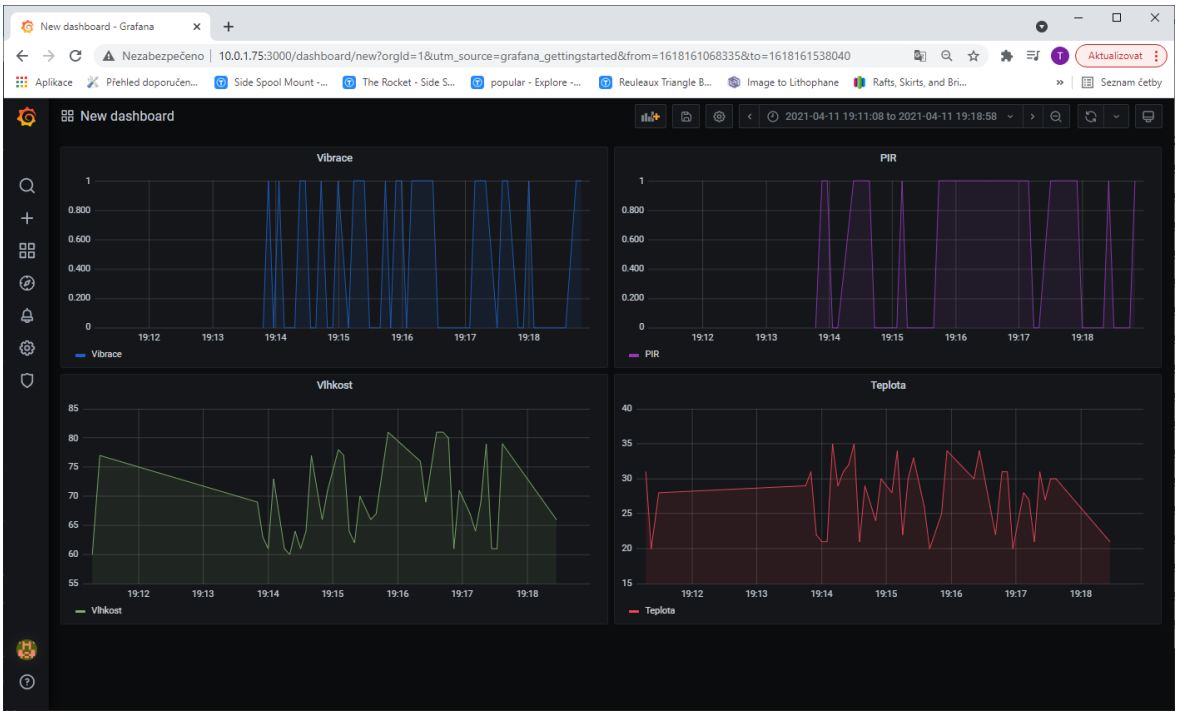
Po úspěšném provedení těchto příkazů jsem propojil Grafanu s MySQL Nejprve je potřeba přihlásit se do webového rozhraní Grafany. Ta byla dostupná na adrese **http://10.0.1.75:3000**. Přihlásil jsem se pod jménem „admin“ a heslem „admin“. Pak jsem vybral „Add data source“, zvolil MySQL. Vyplnil jsem potřebné údaje, kterými byly host (URL adresa), jméno databáze, jméno uživatele a jeho heslo. Jak je psáno výše, zde se napíše uživatel pouze s právy na čtení. Pokud uživatel bude mít jiná práva, než jen na čtení, grafana se s MySQL nespojí. Po správném připojení se zobrazil nápis se zeleným pozadím, který je zobrazený níže.



Obrázek 20: *Propojení MySQL s Grafanou*

Dále se ve webovém rozhraní musí smazat veškerá zařízení, která generují vlastní náhodné hodnoty, které byly popsány v podkapitole Zasílání dat a opětovné získání

Jako poslední krok je potřeba spustit dva skripty zmíněné v této podkapitole a v podkapitole Zasílání dat a opětovné získání. Prvním je generování teploty a vlhkosti. Druhý skript je pro odebírání zaslaných dat doplněný o ukládání jich do příslušné databáze. Oba dva skripty jsou autorské. Data uložená v databázi se pak zobrazily v aplikaci Grafana, jak je znázorněno v **Obrázek 21:** Zobrazení dat v Grafaně níže.



Obrázek 21: *Zobrazení dat v Grafaně*

Můžeme zde vidět i data o pohybu a vibracích od kolegy Dlouhého. Pokud proběhla nějaká vibrace nebo pohyb, je to znázorněno číslem 1 a pokud nikoliv, tak číslem 0.

9. Srovnání ELIOTu a EdgeX

Toto srovnání je čistě subjektivní a je rozděleno na tři podkapitoly. Jako první je porovnána podpora, komunita, výsledek instalace a vyhledávání všeobecných informací a návodů. Druhá se věnuje porovnání popisu instalace od vývojářů. Ve třetí, tedy poslední podkapitole, čtenář nalezne porovnání funkčnosti.

9.1. Základní porovnání ELIOTu a EdgeX

První, co je potřeba zmínit, je, že oba projekty patří do poslední skupiny ze tří z Frameworku LF Edge. Vše by tedy mělo být funkční, s dobrou podporou a vypracovanou komunitou. Toto bohužel neplatí pro platformu Akraino, konkrétně jejich ELIOT. Nehledě na to, že se mi nepodařilo framework nainstalovat, jejich podpora je na velmi slabé úrovni. Po obtížném hledání kontaktní osoby zabývající se danou problematikou jsem byl sice schopen získat odpověď, ale ta se zdála strojová, neboť ze dvou různých emailů s podobným dotazem mi byla zaslána stejná odpověď. Komunita zabývající se frameworkem ELIOT není aktivní. I přes značné pátrání se nepodařilo najít žádné uživatele. Pokud se uživatel pokusí hledat na vyhledávacím serveru www.google.com „Akraino ELIOT“, zobrazí se pouze stránky LF Edge, stránky Akraina, prezentace, můj dotaz na jednom fóru a několik dalších, pro tuto práci nerelevantních, výsledků. Samozřejmě počítám s tím, že Google pracuje se svým algoritmem, proto jsem pro hledání použil anonymní režim a proxy pro zobrazení výsledků z „jiné země“. Výsledky hledání byly téměř totožné. Dále jsem se pokusil vyhledat ELIOT na serveru www.youtube.com za účelem zjistit, zda neexistují video návody nebo záznamy reálného použití. Bohužel byly nalezeny pouze přednášky, kde se představuje samotný ELIOT.

Zato EdgeX od EdgeX Foundry je na tom o poznání lépe. Oproti ELIOT nemám žádné reálné zkušenosti s podporou, ale s komunitou ano. Například jsem se snažil konzultovat část poinstalačního kroku s jedním z členů komunity z Indie, i když výsledky konzultace nebyly zdárné. Co se týče funkčnosti, EdgeX oproti ELIOTu lze nainstalovat a spustit. Pokud se rozhodneme vyhledat „EdgeX“ pomocí Google a YouTube, výsledky hledání jsou na rozdíl od ELIOTu mnohem lepší. Získané výsledky vyhledávače Google jsou v podobě návodů na instalaci, které nepocházejí jen od vývojářů, ale i od samotného Ubuntu. Dále jsou dohledatelné práce zabývající se EdgeX a další užitečné odkazy. Výsledky na portálu YouTube zobrazují také přednášky, jako po vyhledávání ELIOTu, ale ukazují i reálné použití a nasazení od různých lidí, návody na instalaci a doplňky.

9.2. Porovnání popisu instalací

Ačkoliv se na první pohled popis instalace od Akraina zdá přehledný, opak je pravdou. Na stránkách Akraina je popis instalace všech verzí ELIOTu, jenže nikde není psáno, že starší se nemají používat, neboť již nefungují. Samotný popis nejnovější instalace[3] je chaotický. Na rozdíl od EdgeX sice uvádí minimální HW požadavky, za to ale používají starší verze OS. K samotné instalaci jsou zapotřebí minimálně 3 počítače, tedy více oproti druhé platformě. Jejich instrukce a popis jednotlivých zařízení je také matoucí, například v návodu jsou názvy „Jump Host“, „Deployment Host“, „OCD Host“ a „One Click Deployment Node“ pro stejné zařízení. Instrukce typu „zkontrolujte a změňte parametry v souboru“ je nicneříkající a nesrozumitelná.

EdgeX Foundry na popis instalace jdou, na rozdíl od Akraina, zcela rozdílně. Mají několik návodů na samotnou instalaci, kde to rozdělují na „Rychlý start“, „Nasazení přes Docker“ a „Nasazení přes Snap“. První „Rychlý start“ čistě popisuje, jak co nejrychleji nainstalovat a použít EdgeX bez samotného pochopení, jak vlastně funguje. Zbylé dvě přehledně popisují, jak má uživatel postupovat. Liší se jen typem, přes co se instaluje. Samotný popis instalace na Raspberry Pi je velmi přehledný, rozdělen na čtyři kapitoly, kde je přesně popsáno, co k čemu slouží. Dále oproti ELIOTu na stránkách nalezneme návody na připojení různých zařízení a použití.

9.3. Porovnání funkčnosti

Jak bylo uvedeno v kapitole o instalaci, ELIOT se nepodařilo zprovoznit, proto se tato podkapitola bude věnovat pouze EdgeX.

Ačkoliv v podkapitole výše je zmíněno, že na stránkách se dají nalézt návody na připojení různých zařízení, tak ty, které jsem zkoušel:

- MQTT (zde se nejedná o použité MQTT, ale o [42])
- custom_services [43]

nefungují. Samotný EdgeX také nefunguje stoprocentně. Spuštění příkazu, který zde již byl zmíněn:

```
COMPOSE_HTTP_TIMEOUT=600 docker-compose -f docker-compose-ui-arm64.yml up
-d
```

[3] V době psaní této práce byl aktuální Release 4

se ne vždy spustí všechny potřebné služby a je potřeba ho spustit několikrát (maximální počet opakovaných pokusů byl v mém případě pět). Někdy, když jsem se pokusil spustit EdgeX, docker vyhodil chybu s obsazeností portů a bylo potřeba veškeré docker kontejnery smazat. Tento krok nebyl vždy úspěšný, a tak jsem byl nucen smazat všechno, včetně instalačních skriptů, odinstalovat docker a docker-compose a poté vše nainstalovat znovu. Jednou se mi stalo, že ani toto nezabralo a byl potřeba přeinstalovat operační systém (příčinou bylo zobrazení výše uvedené chyby s použitými porty, ale nikdo je nepoužíval). Také se stalo, že se mi náhodně ukončila MQTT služba, která byla použita pro exportování dat, dále služba, která shrnuje zařízení a jiné. Většinou se dá vše vyřešit příkazem pro znovuspuštění. Také je potřeba zmínit informaci napsanou v kapitole ohledně instalace. Některé služby jsou již od samotného začátku nefunkční, a nelze je spustit ani po opakovaných pokusech.

Závěr

Edge computing je nová technologie, která podle mě teprve dostane svůj prostor. Přijde mi to podobné, jako to bylo s USB-C, které bylo známo již delší dobu, a až po několika letech se z něj teprve stal standard.

Projekt LF Edge má své opodstatnění, ale dle výsledků výzkumu se zdají být nevhodně nastavené úrovně, kam jednotlivé projekty postupují. Třetí úroveň by, dle mého názoru, rozhodně neměla být finální. Důkazem toho je již zmíněný ELIOT od Akraina, jehož zpracování 3. úrovně nedosahuje. Podpora tohoto frameworku je velmi slabá, odpověď na položené otázky je nedostačující a zdá se být strojová. Komunita je slabá, což dokazuje i absence odpovědí jak na různých skupinách, tak na oficiálním fóru, kde to dokazuje fakt, že od roku 2019 je položeno pouze osm otázek[4] [44]. Framework se nepovedlo nainstalovat. Poinstalační kroky zcela chybí a není tedy možné zjistit, jak Framework v případě úspěšné instalace použít. Z těchto důvodů ELIOT nedoporučuji. Jelikož se jedná o novou technologii, dá se předpokládat, že v dohledné době budou vydány nové releasy a instalace tak bude proveditelná.

EdgeX od EdgeX Foundry je na tom, jak bylo zmíněno v předchozí kapitole, o poznání lépe. Zde sice nemám zkušenosti s podporou, spojil jsem se však s jedním členem komunity. Samotný EdgeX se dá nainstalovat a i přes občasné potíže používat. Jak jsem již psal, občas náhodně spadnou výše zmíněné služby EdgeX, ne vždy se spustí vše na první pokus a nějaké služby nefungují vůbec (také vypsané výše). Na základní úlohy, jako byla ta má testovací, je však dostačující. Dá se proto říct, že tato technologie je použitelná, ale s výhradami. Mezi tyto výhrady patří zejména to, že na použití EdgeX má Raspberry Pi 3B+ slabý výkon a také je potřeba počítat i s komplikacemi spojených s procesorem ARM64, kde právě to je občas důvod, proč některé služby nefungují. Je třeba i zmínit to, že exportování dat přes MQTT ne vždy zcela stoprocentně odešle všechna přijatá data. I zde si myslím, že v průběhu času se vše zlepší a použití bude mnohem příjemnější .

Zkráceně se tedy dá říct, že ELIOT k nasazení a používání nedoporučuji. EdgeX k nasazení a používání doporučuji, ale pouze pro základní úlohy. Avšak není vhodný k provozu na Raspberry Pi či jiném ARM procesoru.

[4] Ke dni 9.4. 2021

Seznam zdrojů

[1] SHI, Weisong, Jie CAO, Quan ZHANG, Youhuizi LI, Lanyu XU. *Edge computing: Vision and challenges*. IEEE Internet Things J., vol. 3, no. 5, pp. 637–646. [online] 2016 [cit. 2020-11-24] Dostupné z: https://ieeexplore.ieee.org/abstract/document/7488250

[2] ARUTPERUNJOTHI, Raj. *Akraino Edge Stack*. Akraino. [online] 2020 [cit. 2020-11-24] Dostupné z: https://wiki.akraino.org/display/AK/Akraino+Edge+Stack

[3] SHI, Weisong, George PALLIS, Zhiwei XU. Edge computing. Proceedings of the IEEE | Vol. 107, No. 8. [online] srpen 2019 [cit. 2020-11-24] ISBN: Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8789742

[4] FINNEGAN, Matthew. Boeing 787s to Create Half a Terabyte of Data Per Flight, Says Virgin Atlantic. [Online] 2016 [cit. 2020-11-24] Dostupné z: https://datafloq.com/read/self-driving-cars-create-2-petabytes-data-annually/172 - dořešit

[5] VOIGT, Paul, Axel Von Dem BUSSCHE. *The EU general data protection regulation (GDPR): A Practical Guide,* 1st ed. Cham, Switzerland: Springer, [online] 2017 [cit. 2020-11-24]. ISBN: 978-3-319-57959-7. Dostupné z: https://www.researchgate.net/publication/321515854_The_EU_General_Data_Protection_Regulation_GDPR_A_Practical_Guide

[6] DOLUI, Koustabh, Soumya Kanti DATTA. *Comparison of edge computing implementations: fog computing, cloudlet and mobile edge computing*. Global IOT Summit. Geneva, Switzerland. [online] ©2017 [cit. 2020-12-03]Dostupné z:https://www.eurecom.fr/publication/5193

[7] STOJMENOVIC, Ivan, Sheng WEN. *The Fog Computing Paradigm: Scenarios and Security Issues*. Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 1–8. [online] 2014 [cit. 2020-12-03] DOI: 10.15439/2014F503. Dostupné z: http://faratarjome.ir/u/media/shopping_files/store-EN-1519284237-8059.pdf

[8] HU, Yun Chao, Milan PATEL, Dario SABELLA, Nurit SPRECHER a Valerie YOUNG. *Mobile Edge Computing A key technology towards 5G*. ETSI. [online] 2015 [cit. 2020-12-03] ISBN: 979-10-92620-08-5 Dostupné z: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[9] SHETH Viral. *Fog / Edge-computing and Cloudlets*. Medium. [online] 2019 [cit. 2020-12-03] Dostupné z: https://medium.com/@virral/fog-edge-computing-and-cloudlets-c4db6cc9b15a

[10] LF Edge. *Enterprise Mobility Exchange: The Future of Edge Computing*. LF EDGE. [online] 2019 [cit. 2020-12-09] Dostupné z: https://www.lfedge.org/2019/02/08/enterprise-mobility-exchange-the-future-of-edge-computing

[11] *What is lf edge.* LF Edge. [online] ©2020 [cit. 2020-11-24] Dostupné z: https://www.lfedge.org/

[12] *Members*. LF EDGE. [online] 2019 [cit. 2021-02-25] Dostupné z: https://www.lfedge.org/members

[13] *Projects.* LF Edge. [online] ©2020 [cit. 2020-11-24] Dostupné z: https://www.lfedge.org/projects/

[14] *Baetyl*. LF EDGE. [online] ©2020 [cit. 2021-04-22] Dostupné z: https://www.lfedge.org/projects/baetyl/

[15] *Secure Device OnBoard*. LF EDGE. [online] ©2020 [cit. 2021-04-22] Dostupné z: https://www.lfedge.org/projects/securedeviceonboard/

[16] *Open Horizon*. LF EDGE. [online] ©2020 [cit. 2021-04-22] Dostupné z: https://www.lfedge.org/projects/openhorizon/

[17] *EVE*. LF EDGE. [online] ©2020 [cit. 2021-04-22] Dostupné z: https://www.lfedge.org/projects/eve/

[18] *Fledge*. LF EDGE. [online] ©2020 [cit. 2021-04-22] Dostupné z: https://www.lfedge.org/projects/fledge/

[19] HomeEdge. LF EDGE. [online] ©2020 [cit. 2021-04-22] Dostupné z: https://www.lfedge.org/projects/homeedge/

[20] *State of the Edge*. LF EDGE. [online] ©2020 [cit. 2021-04-22] Dostupné z: https://www.lfedge.org/projects/stateoftheedge/

[21] *Akraino*. LF Edge. [online] ©2020 [cit. 2020-11-24] Dostupné z: https://www.lfedge.org/projects/akraino/

[22] TSOU, Tina. *Approved blueprints*. Akraino. [online] 2010 [cit. 2021-01-10] Dostupné z: https://wiki.akraino.org/display/AK/Approved+blueprints

[23] DASGUPTA, Abhijit. *ELIOT Architecture Document*. Akraino. [online] 2019 [cit. 2021-01-10] Dostupné z: https://wiki.akraino.org/display/AK/ELIOT+Architecture+Document

[24] *EdgeX Foudry*. LF Edge. [online] ©2020 [cit. 2020-11-24] Dostupné z: https://lfedge.org/projects/edgexfoundry

[25] gabrielle. *EdgeX Foundry Project Wiki*. EdgeX Wiki. [online] 2020 [cit. 2020-11-24] Dostupné z: https://wiki.edgexfoundry.org

[26] *EdgeX Foundy platform*. EdgeX Foundry. [online] ©2021 [cit. 2021-03-19] Dostupné z: https://edgexfoundry.org/software/platform/

[27] *Release Cadence*. EdgeX Foundry. [online] ©2021 [cit. 2021-03-22] Dostupné z: https://edgexfoundry.org/software/releases

[28] WHITE, Jim. *3. How to install EdgeX.* 30_install_edgex.md. GitHub [online] 2021 [cit. 2021-03-19] Dostupné z: https://github.com/edgexfoundry/edgex-examples/blob/master/deployment/raspberry-pi-4/30_install_edgex.md

[29] *Raspberry Pi 3 Model B+*. Raspberry Pi. [online] [cit. 2021-03-22] Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[30] *Raspberry Pi Touch Display*. Raspberry Pi. [online] [cit. 2021-03-22] Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-touch-display/

[31] *Raspberry Pi OS*. Raspberry Pi. [online] [cit. 2021-04-18] Dostupné z: https://www.raspberrypi.org/documentation/raspbian/

[32] GRIMBERG, Andrew. *Welcome to the Akraino Wiki*. Akraino. [online] 2020 [cit. 2021-03-18] Dostupné z: https://wiki.akraino.org

[33] DASGUPTA, Abhijit. *ELIOT Installation Guide*. Akraino. [online] 2020 [cit. 2021-03-18] Dostupné z: https://wiki.akraino.org/display/AK/ELIOT+Installation+Guide

[34] Selvam, Srinivasan. *ELIOT R4 - IoT Gateway Installation Guide*. Akraino. [online] 2021 [cit. 2021-03-18] Dostupné z: https://wiki.akraino.org/display/AK/ELIOT+R4+-+IoT+Gateway+Installation+Guide

[35] WHITE, Jim, bus710. *EdgeX on Raspberry Pi*. Raspberry-pi-4. GitHub [online] 2021 [cit. 2021-03-25] Dostupné z: https://github.com/edgexfoundry/edgex-examples/tree/master/deployment/raspberry-pi-4

[36] WHITE, Jim. 2. *How to install package required for EdgeX development*. 20_install_packages.md. GitHub [online] 2021 [cit. 2021-03-25] Dostupné z: https://github.com/edgexfoundry/edgex-examples/blob/master/deployment/raspberry-pi-4/20_install_packages.md

[37] *Getting Started with Docker*. EdgeX Foundry Documentation. [online] 2020 [cit. 2021-03-25] Dostupné z: https://docs.edgexfoundry.org/1.2/getting-started/Ch-GettingStartedUsers/

[38] WERNER, Jonas. *EdgeX Foundry: A hands-on tutorial. A practical guide to getting started with open source IoT*. [online] 2020 [cit. 2021-03-26] Dostupné z:

https://docs.edgexfoundry.org/1.2/examples/LinuxTutorial/EdgeX-Foundry-tutorial-ver1.0.pdf

[39] *What is Postman?.* Postman. [online] 2021 [cit. 2021-03-25] Dostupné z: https://www.postman.com

[40] MQTT. EdgeX Foundry Documentation. [online] 2020 [cit. 2021-03-27] Dostupné z: https://docs.edgexfoundry.org/1.2/examples/Ch-ExamplesAddingMQTTDevice/

[41] *Eclipse Mosquitto™. An open source MQTT broker*. Mosquitto. [online] [cit. 2021-03-26] Dostupné z: https://mosquitto.org

[42] *MQTT. EdgeX - Edinburgh Release*. Mosquitto. [online] ©2020 [cit. 2021-03-26] Dostupné z:

https://docs.edgexfoundry.org/1.2/examples/Ch-ExamplesAddingMQTTDevice/

[43] WHITE, Jim. *4. How to develop custom device and app services*. 40_custom_services.md. GitHub [online] 2021 [cit. 2021-03-27] Dostupné z:

https://github.com/edgexfoundry/edgex-examples/blob/master/deployment/raspberry-pi-4/40_custom_services.md

[44] *Otázky*. Akraino. [online] 2021 [cit. 2021-04-09] Dostupné z: https://wiki.akraino.org/display/AK/questions/all?&filter=recent

Seznam obrázků

Seznam tabulek