ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra měření

Bakalářská práce

Bezpečnost komunikace v senzorové síti

Ondřej Maňhal

Květen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Maňhal** Jméno: **Ondřej** Osobní číslo: **466103**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Otevřená informatika**

Studijní obor: **Internet věcí**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Bezpečnost komunikace v senzorové síti

Název bakalářské práce anglicky:

Communication Security in Sensor Network

Pokyny pro vypracování:

Proveďte detailní analýzu bezpečnosti komunikace mezi komponenty senzorové sítě. Zaměřte se zejména na služby pro zprostředkování, ukládání a vizualizaci dat. Analyzované služby zhodnoťte též z ekonomického pohledu. Svá zjištění demonstrujte na testovacím pracovišti sestaveném z vybraných senzorů, PC embedded platformy a analyzovaných služeb.

Seznam doporučené literatury:

[1] Straka, T.: Cloudové služby v prostředí technologické firmy. Praha 2017. Bakalářská práce, ČVUT FEL. Katedra telekomunikační techniky. 2017-05-25.
[2] Weber, R.H. - Weber, R.: Internet of Things, Legal Perspectives. Vol 12. New York, NY, Springer, 2010. ISBN 978-3-642-11709-1.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Bezpalec, Ph.D., katedra telekomunikační techniky FEL

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

Datum zadání bakalářské práce: **12.02.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce:
do konce letního semestru 2020/2021

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

III. PŘEVZETÍ ZADÁNÍ

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

. | Datum převzetí zadání

Podpis studenta

Poděkování / Prohlášení

Dovolte mi poděkovat vedoucímu mé závěrečné práce, panu Ing. Pavlu Bezpalcovi, za předmětné rady, podněty a postřehy, bez kterých by vypracování této práce pro mne nebylo možné. Také bych rád poděkoval všem, kteří se mnou měli během psaní této práce trpělivost a pochopení.

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

Abstrakt / Abstract

Tato práce se zabává zhodnocením bezpečnosti komunikace mezi komponenty v senzorových sítích. V teoretické části je popsána problematika zabezpečení, služeb pro zprostředkování, ukládání a vizualizaci dat s ohledem na praktickou část práce. V té jsou zjištěny dopady zabezpečení komunikace na ekonomickou nákladnost, stejně jako jsou jednotlivé využité služby analyzovány z pohledu bezpečnosti. Toto je provedeno na malém testovacím pracovišti sestávajícím z vybraných senzorů, vývojové desky WeMos D1 mini a miniPC RaspberryPi, stejně jako analyzovaných služeb. Ekonomická stránka je poté zhodnocena v samostatné kapitole na závěr práce.

Klíčová slova: senzorová síť, bezpečnost komunikace, vizualizace dat, ukládání dat, zprostředkování dat

The goal of this thesis is to provide assessment of comunication security between the parts of sensor network. The theoretical part describes the problematic of computer security, visualisation, storage and data transfer services with respect to the practical part. The practical part asses the impact of secure communication on expenses as well as each of the services used is analyzed with regards to computer security. This is achieved using a small testbed made of selected sensors, WeMos D1 mini development board, RaspberryPi miniPC and selected, analyzed, services. The economical impact is the reviewed in it's own chapter at the end of the thessis.

Keywords: sensor network, communication security, data visualisation, data storage, data access

Title translation: Comunication Security in Sensor Network

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Senzorové sítě a z nich vycházející infrastruktura internetu věcí přinesla mnoho nových možností pro podniky a jejich zákazníky a to především v oblasti zdravotnictví, skladování, logistiky, ale samozřejmě i ve vojenských a bezpečnostních otázkách států. Z tohoto masového rozšíření plynou nové nároky a výzvy pro vývojáře, kteří musí zajistit dostatečné zabezpečení jejich projektů. Obzvláště důležité je toto zabezpečení v aplikacích kritické infrastruktury, zdravotnictví a ve vojenských aplikacích, kde sebemenší opomenutí může mít fatální následky. Na druhou stranu zabezpečení přenosu teploty z domácího teploměru do domácí meteostanice již může být spíše kontraproduktivní z důvodu zvyšování složitosti zařízení a diskutabilního přínosu.

Tedy je nutné znát možnosti a cíle dané aplikace, daného projektu a dle něj volit ten správný kompromis. V rámci senzorové sítě jsem se zaměřil na stránku vizualizace sbíraných dat a jejich případného ukládání. Nejsme daleko od doby, ba se v ní již přímo nalézáme, kdy člověk, než vstane z postele, tak vidí na obrazovce svého chytrého telefonu, jak mu kávovar připravuje ranní kávu, trouba se předehřívá na vhodnou teplotu na pečení toastů. V průmyslové sféře se může jednat o monitoring výroby, případně zaměstnanců. Tato data je v naprosté většině případů nutné někam uložit, ať do cloudu nebo do lokální databáze. Vhodné zabezpečení, smysluplné ukládání a správná, přehledná vizualizace je tedy nezbytná. Proto v práci zmiňuji dnes převážně používaný software na vizualizaci, zpracování i ukládání dat.

Jakožto všude v reálném světě, nesmí se zapomenout na finanční stránku věci, a proto v ekonomické části práce zhodnotím přínos jednotlivých technologií a jejich ekonomickou smysluplnost. Přínosem této práce je i odpověď na otázku, zdali se zabezpečení komunikace mezi komponenty senzorové sítě vyplatí, tedy jestli dává po ekonomické stránce smysl.

Tato práce si klade za cíl objasnit možnosti, dostupnost zabezpečení komunikace ve světě bezdrátových senzorových sítí a vizualizace obdržených dat. Cílem je analyzovat stávající komunikační protokoly, užívané standardy zabezpečení a možnosti vizualizace. A to převážně s přihlédnutím k v praxi používaným řešením. Zároveň je kladen důraz na vedlejší produkty zabezpečení, tedy práce se zabývá i negativní stránkou bezpečné komunikace.

Tato negativní stránka bezpečnosti komunikace, tedy vyšší režie je důkladněji analyzována v praktické části práce, kde se zabývám vlivem zabezpečení, potažmo šifrování komunikace, na spotřebu elektrické energie měřících senzorů a celkových nároků na síťovou infrastrukturu. Rozebírány jsou i důsledky nezabezpečené komunikace a její případná pozitiva.

Věřím, že práce bude přínosem odborné veřejnosti, která se zajímá o problematiku bezpečnosti komunikace v rámci senzorových sítích, která by zde mohla nalézt odpovědi na některé své otázky, případně by měla osvětlit, jestli je pro jejich aplikaci zabezpečení komunikace vhodné.

Kapitola 2
Algoritmy počítačové bezpečnosti

Již od počátku věků se lidstvo snažilo zabezpečit vzájemnou komunikaci. Používalo k tomu nejrůznějších šifer a způsobů utajení citlivých informací. V dnešní době digitálních systémů, kdy se téměř veškerá komunikace mezi lidmi a stroji přenáší vzduchem je otázka správného utajení na místě. V této kapitole jsou podrobněji popsány základní zabezpečovací algoritmy, používané v komunikaci v dnešním internetu. Nejdříve uvedu základy zabezpečení a členění šifer, poté v další sekci používané algoritmy.

2.1 Základ zabezpečení a šifry

Původ otázek na zabezpečení komunikace a z nich vyplívajících otázek ochrany soukromí jsou zde od nepaměti. Internet, jako nástroj pro adresování zařízení, jejich lokací a následné připojení přes veřejný komunikační kanál je jednou z největších hrozeb, které lidstvo kdy čelilo. Kdykoliv na internet vstoupíte, kdykoliv se k němu připojíte, vysíláte zároveň i informace o své přítomnosti, a to vzhledem k tomu, že internet je součástí veřejné sféry, ke které má kdokoli přístup, a to s jakýmkoliv účelem není úplně bezpečné. [1] Proto se používají různé formy zabezpečení komunikace, z nichž šifry jsou nejběžnější.

„Od dob druhé světové války se kryptografie a kryptoanalýza staly především matematickými úlohami. Díky široké dostupnosti dostatečně výkonných počítačů a celosvětové síti Internet coby média, se především kryptografie stala běžně využívaným nástrojem a není již nadále spolu s kryptoanalýzou výsadou jednotlivých vlád, nebo podobně velkých společností.“ [2] V roce 1976 byl publikován článek New Directions in Cryptography od W. Diffieho a M. E. Hellmana. Zde byla představena do té doby neznámá metoda přenosu šifrovacího klíče. „Tvrdíme, že je možné vyvinout systém, kde dvě strany komunikují pouze přes veřejný kanál a jen s použitím jen veřejně známých technik jsou schopny navázat zabezpečené spojení.“ [3] Poznatky z tohoto článku vedly k vývoji nového typu šifrovacích algoritmů, který postupně nahradil do té doby používaný typ takzvaných symetrických šifer. Tento nový typ se nazývá asymetrické šifrování, neboť jsou zde na rozdíl od symetrického šifrování, kdy je pro šifrování i dešifrování použit stejný klíč, používány klíče dva. Jeden, zpravidla veřejný, je určen pro zašifrování zpráv pro daného adresáta a druhý, neveřejný, slouží adresátu k jejich dešifrování.

V následujících podsekcích šifry rozdělím, podle principu jejich fungování na symetrické a asymetrické. U každého principu popíši jeho hlavní výhody a nevýhody spolu s tím proč a kde se používá.

2.1.1 Symetrické šifry

V symetrických šifrách, občas také označovány jako šifry s tajným klíčem, je na šifrování i dešifrování použit jen jeden, stejný klíč. Symetrické šifry jsou rychlejší a bývají méně náročné na výpočetní výkon na rozdíl od šifer asymetrických. Klíče jsou v tomto způsobu šifrování jedním z hlavních faktorů ovlivňující bezpečnost celé šifry. Slabé klíče se

dají poměrně jednoduše uhodnout, na rozdíl od složitých, které prolomit je těžké. Symetrické šifry jsou stále běžně používané v případech kdy je kladen velký nárok na alespoň nějaké zabezpečení a zároveň je limitována výpočetní náročnost, v případech, kdy je bezpečná výměna klíčů zaručena nebo jako součást jiných šifrovacích algoritmů.[4]

Jedním z hlavních problémů symetrických šifer je nutnost přenosu šifrovacího klíče mezi subjekty. Tento problém částečně řeší různé algoritmy pro výměnu klíčů, například známý algoritmus Diffie-Hellman. I tak, pokud se nám povede zabezpečeně přenést klíč, stále existuje riziko prolomení klíče. Je-li klíč prolomen třetí stranou, může tato strana zprávy přijímat i odesílat, díky tomu, že jeden šifrovací klíč je použit na obě činnosti. Tedy hrozí zde útok typu *man-in-the-middle*.

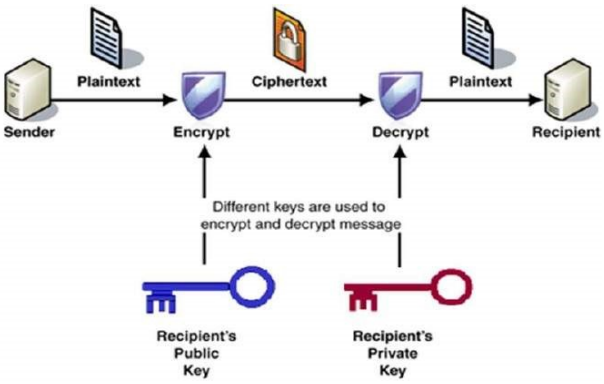
2.1.2 Asymetrické šifry

Asymetrické šifry, jak již z názvu napovídá, používají na rozdíl od šifer symetrických pro šifrování jiný klíč než pro dešifrování. Tato skupina šifer bývá také označována jako skupina šifer s veřejným klíčem, protože šifrovací klíč je zde rozdělen na dva. Veřejný klíč je použit pro šifrování zpráv pro daný subjekt, k jemuž je přidružen. Tento klíč je veřejně dostupný. Privátní klíč, který je použit pro dešifrování zpráv, je oproti klíči veřejnému držen v tajnosti a disponuje jím jen vlastník veřejného klíče. Privátní (soukromý) i veřejný klíč spolu musí být svázány, ovšem podmínkou efektivity šifry je, aby nebylo možné jeden z druhého spočítat. K tomuto jsou využívány různé funkce, u nichž je jejich provedení poměrně jednoduché, ale zpětné získání vstupních parametrů z výsledku téměř nemožné. Příkladem takovéto funkce je třeba násobení velkých prvočísel.

Asymetrických šifer se využívá nejen pro zabezpečení zpráv, ale i pro ověření autora zprávy. Pokud totiž subjekt zašifruje něco svým veřejným klíčem, pak dešifrování autorovým veřejným klíčem potvrzuje platnost takovéto zprávy. Protože zašifrovat ji mohl jen a pouze vlastník privátního klíče a tedy autorem je opravdu ten, jehož veřejný klíč jsme použili pro dekódování.

„Asymetrické šifrovací algoritmy jsou v porovnání se symetrickými obecně výrazně pomalejší. Asymetrické kryptosystémy, kryptografické protokoly i metody digitálních podpisů používají komplikované operace s dlouhými čísly, které by standardnímu PC trvaly příliš dlouho. Proto se často šifruje klasickými symetrickými systémy a asymetrickými systémy se šifrují pouze relativně krátké použité symetrické klíče. “ [5]

Obecné fungování asymetrických šifer, tedy aplikace privátního a veřejného klíče je zobrazena na obrázku 2.1.



Obrázek 2.1. Princip asymetrické šifry [6]

2.2 Používané šifrovací algoritmy a protokoly

Šifrovacích algoritmů a protokolů je obrovské množství různých kvalit a vlastností, a proto v této sekci uvedu ty, pro tuto práci klíčové a zároveň ty nejvíce používané v současné informační infrastruktuře.

2.2.1 Diffie-Hellman

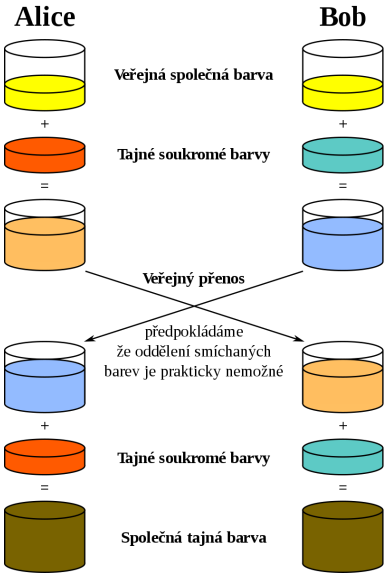
Diffie-Hellman protokol vznikl v roce 1976, dle návrhu W. Diffieho a M. E. Hellmana. Jedná se o protokol, který umožňuje zabezpečenou výměnu šifrovacího klíče přes veřejně dostupné prostředky. Zároveň se však nejedná o šifrovací algoritmus v pravém slova smyslu, neboť není určen na posílání zabezpečených zpráv, nýbrž jen a pouze na bezpečnou výměnu klíčů. Zabezpečení je založeno na problému výpočtu diskrétního logaritmu

„Matematický postup pro stanovení klíče je jednoduchý. Nejdřív se obě strany dohodnou na velkém prvočísle n, jedna strana je navrhne a druhá strana vyjádří souhlas. Toto prvočíslo je nazýváno modulem. Poté je nutné aby se dohodly na dalším čísle (g), vygenerovaném generátorem, které tvoří základ modulu n, pro číslo g platí $1 \leq g \leq n-1$. Není požadováno, aby g bylo co největší číslo, může být klidně i jednomístné. Prvočíslo n musí být z hlediska bezpečnosti vybráno tak, aby bylo dostatečně veliké a aby hodnota $(n-1)/2$ byla také prvočíslo. Tyto dvě celá čísla n a g jsou veřejná, nemusí být tedy tajná. “ [7]

Postup přenosu tajného klíče mezi dvěma subjekty je tedy následovný:

- Subjekty si určí prvočíslo n a nějaké přirozené číslo g
- Dále si subjekty zvolí každý jedno libovolné přirozené číslo, tedy a a b
- Subjekt I spočítá číslo $A = (g * a) \bmod n$ a odešle ji
- Subjekt II spočítá číslo $B = (g * b) \bmod n$ a odešle ji
- Subjekt I spočítá klíč k, kde $k = (B * a) \bmod n$
- Subjekt II spočítá klíč k, kde $k = (A * b) \bmod n$
- Oba subjekty mají nyní stejný klíč k

Tento princip je znázorněn na následujícím obrázku 2.2



Obrázek 2.2. Diffie-Hellman výměna klíčů, znázorněna pomocí barev [8]

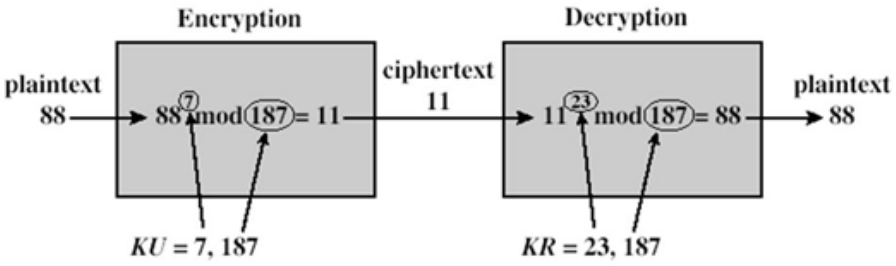
2.2.2 RSA

Algoritmus RSA je jedním z prvních algoritmů, který je vhodný jak pro šifrování tak pro vytváření digitálních podpisů. Jedná se o algoritmus popsaný v roce 1977 a pojmenovaný dle počátečních písmen jeho vynálezců, pánů Rona Rivesta, Adiho Shamira a Lena Adlemana. Jedná se o šifru s veřejným šifrovacím klíčem. Bezpečnost RSA je postavena na matematickém problému faktorizace velmi velkých čísel. „V roce 2005, největší číslo faktoru univerzálními metodami bylo 663 bitů dlouhé, použitím distribučních metod. RSA klíče jsou typicky dlouhé 1024-2048 bitů. Někteří experti věří, že klíče 1024 bitů se mohou v blízké době stát prolomitelnými, ačkoli toto je sporné. Proto se obecně předpokládá, že RSA je bezpečný jestliže n je dostatečně velké. Jestliže n je 256 bitů nebo kratší, může být za pár hodin faktorizován na osobním počítači, za použití volně dostupného softwaru. V současné době je doporučováno, aby n bylo alespoň 2048 bitů dlouhé. V roce 1993 publikoval Peter Shor Shorův algoritmus, který ukazoval, že by kvantový počítač mohl v principu vykonávat faktorizaci v polynomiálním čase, což by učinilo RSA a příbuzné algoritmy zastaralými. Realizace principů kvantového počítání se však v současnosti potýká s takovými praktickými problémy, že se o bezpečnost zašifrovaných dat zatím není třeba bát.“ [9]

Postup výroby privátního a veřejného klíče a následného šifrování a dešifrování je následovný:

- Zvolíme dvě prvočísla p a q a vynásobíme je, tedy spočteme $m = p * q$
- Spočteme hodnotu Eulerovy funkce $\varphi(m) = (p-1) * (q-1)$
- Zvolíme celé čísla e a d takové, že $e < \varphi(m)$ a zároveň e je s $\varphi(m)$ nesoudělné a d takové, že platí $d * e \equiv 1 (\mathrm{mod} \varphi(m))$
- Veřejným klíčem je pak dvojice (m, e) a privátním klíčem dvojice (n, d), kde e se také označuje jako šifrovací exponent a d jako dešifrovací.
- Chceme-li zašifrovat zprávu (v číselném formátu) z, provedeme $s = z^e \bmod m$, kde s je zašifrovaná zpráva
- Dešifrování poté provedeme jako $z = s^d \bmod m$

Tento postup je pro lepší ilustraci znázorněn na obrázku 2.3.



Obrázek 2.3. Princip fungování RSA [10]

2.2.3 SHA

„Šifrování a dešifrování velkého množství dat může být poměrně časově náročné, a tak se v některých případech využívají tzv. hašovací funkce. Hašovací funkce vytvoří z dokumentu otisk (haš, hash) o mnohem kratší délce, než má samotný dokument. Musí být přitom zajištěno, že i malá změna v původním dokumentu se projeví výraznou změnou haše.“ [11]

Algoritmus SHA (*Secure Hashing Algorithm*) je velmi používaná hašovací funkce, vytvářející haše pevné délky. Jedná se o algoritmus navržený americkou vládní organizací NSA, který byl standardizován, ve verzi označované jako SHA-0, v roce 1993. V dnešní době se používá rodina algoritmů označovaných jako SHA-2. Jednotlivé algoritmy této rodiny se mezi sebou liší jen délkou výstupu, kde 256 bitový výstup je nejběžnější a použitý algoritmus se často označuje jako SHA-256.

SHA je dnes použito téměř v každé bezpečnostní aplikaci, díky jeho schopnosti vytvářet bezpečné (ve verzi SHA-2, případně SHA-3, která je stále ve vývoji) otisky což je důležité pro jakékoliv ověřování. Díky otisku je možné ověřovat, jestli uživatel zadal správné heslo, bez nutnosti fyzického uložení takovéhoto hesla v databázi, jestli stažený soubor je opravdu ten, který vydavatel uvolnil ke stažení a podobně. Zároveň je ale SHA důležitou součástí tvorby digitálních certifikátů a jednou z klíčových součástí algoritmu TSL/SSL, neboť tento algoritmus (SSL) ověřuje autentičnost protistrany pomocí digitálních certifikátů. Více o SSL v sekci 2.2.4.

Jednotlivé verze algoritmu SHA jsou porovnány na obrázku 2.4, kde jsou zobrazeny jejich základní vlastnosti. Algoritmy SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 a SHA-512/256 jsou součástí rodiny SHA-2.

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Obrázek 2.4. Přehled vlastností Secure Hash Algoritmu (SHA) [12]

2.2.4 SSL a TLS

Protokoly SSL (Secure Sockets Layer) a TLS (Transport Layer Security) jsou dnes všeobecně akceptovaným standardem pro zabezpečenou komunikaci klienta a serveru. Protokoly používají kombinaci symetrického a asymetrického šifrování. Symetrické pro jeho nenáročnost a rychlost vůči asymetrickému a asymetrické pro jeho lepší vlastnosti při autentizaci dvou zařízení. [13] Historie TLS a jeho předchůdce SSL se začala psát již v počátcích internetu v 90. letech. V roce 1994 vytvořila společnost Netscape protokol SSL, v roce 1995 byla vydána SSL v3, později jako IETF (Internet Engineering Task Force) RFC 6101. V roce 1999 se z SSL v3 stává TLS v1.0, publikováno jako RFC 2246. [14] Ke změně jména došlo díky spojení Netscape a Microsoftu a dohodě, že IETF protokol zastřeší. Tato změna působí dodnes zmatky, jelikož protokoly nepracují na transportní vrstvě, jak by napovídal nový název a i proto, že SSL již bylo zažité označení. [15] Navázání spojení vždy začíná takzvaným *SSL handshakem*, sérií zpráv,

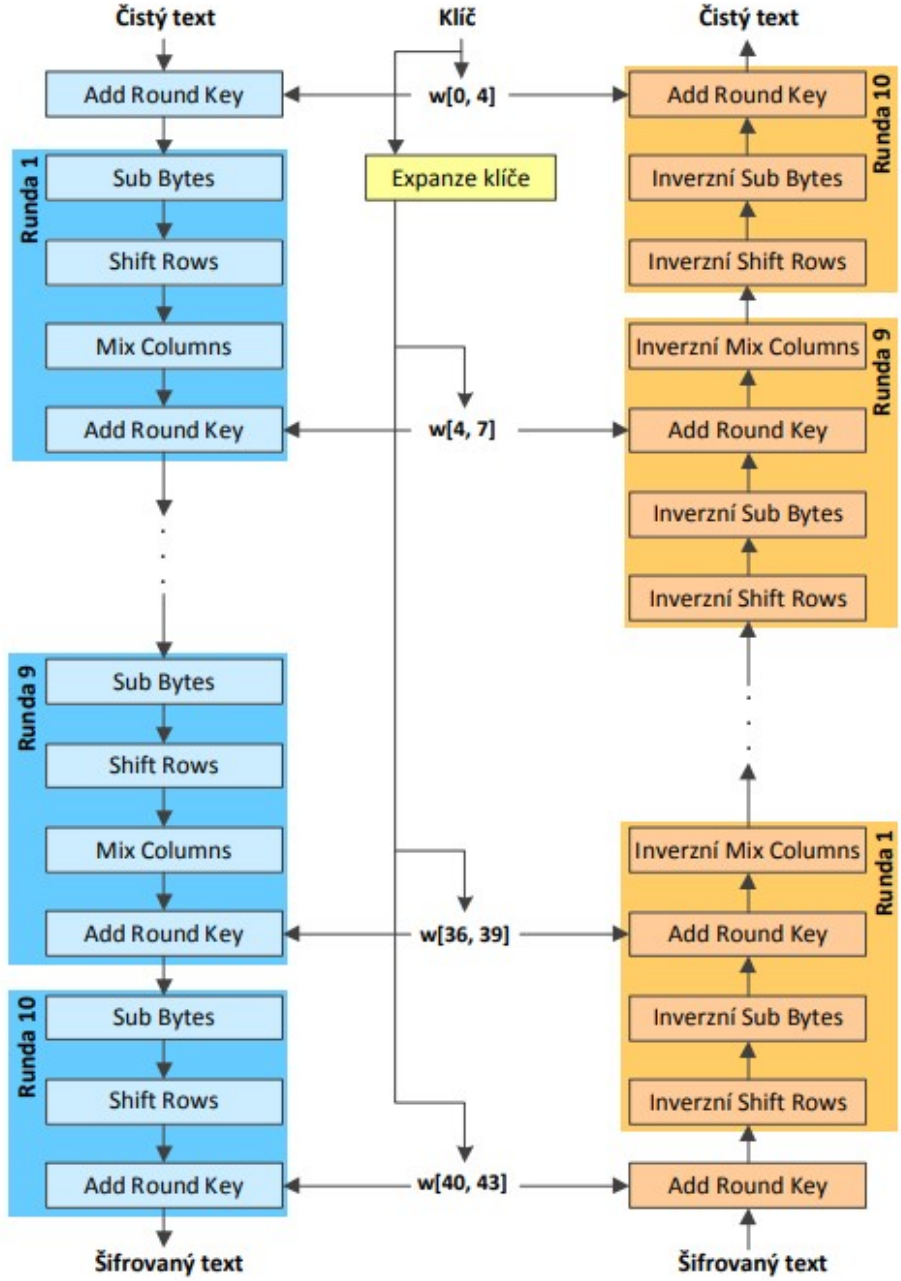
které slouží pro navázání zabezpečené komunikace klienta a serveru. Klient se obrátí na server s žádostí o zabezpečené spojení spolu se seznamem podporovaných šifer. Z tohoto seznamu si server vybere a o své volbě informuje klienta. Poté se za pomoci buď RSA nebo nějakého jiného algoritmu na výměnu klíčů (např. Diffie-Hellman) dohodne náhodný klíč, který poté zůstává platný po celou dobu SSL/TLS spojení a všechny zprávy jsou jím šifrovány. Na obrázku 2.5 je podrobné schéma této HELLO inicializační části protokolu.

Obrázek 2.5. Princip fungování SSL/TLS [16]

2.2.5 AES

Advanced Encryption Standard (AES) je symetrická, bloková šifra, aplikovaná na data s pevně danou velikostí bloku, 128 bitů, kde šifrovací klíč je o délce 128, 192 nebo 256 bitů. Algoritmus byl představen světu v roce 2000, kdy vyhrál výběrové řízení amerického NIST (*National Institute of Standards and Technology, Národní institut standardů a technologie - USA*) na náhradu zastarávajícího algoritmu DES. AES byla vyvinuta s ohledem na využití hardwaru i softwaru a tedy je všeobecně použitelná. [17]

Díky optimalizaci algoritmu bylo dosaženo velké efektivity výpočtu a tudíž je tato šifra poměrně masově rozšířená. Její nejznámější využití nalezneme jako součást standardu WPA2-PSK (AES), kde slouží k zabezpečenému připojování do sítě WiFi. Proces šifrování a následného dešifrování je znázorněn na následujícím obrázku 2.6.



Obrázek 2.6. Princip šifrování a dešifrování pomocí AES [7, 18]

Kapitola 3
Prostředky pro zprostředkování a vizualizaci dat

Svět IoT a senzorových sítí je celý založen na získávání a zužitkování dat reálného světa, v reálném čase. Již nyní se blížíme do doby, kdy téměř každé zařízení je zdrojem dat a všechna tato data by byla k ničemu, pokud bychom je neuměli sesbírat, pochopit a využít. Sběrem dat, respektive jejich přenosem, jsme se zabývali výše; pro jejich pochopení a využití je ale nutné umět data i zobrazit, vizualizovat. Kombinací mnoha zdrojů dat s dobrou vizualizační platformou můžeme dosáhnout zisku nových dat, důležitých pro kritická rozhodnutí v reálném čase. Můžeme monitorovat průběh výroby, řídit bezpečnost na fotbalovém stadionu; téměř vše je možné díky správné kombinaci získávání, analýze a zobrazení dat.

Ve svém testovacím pracovišti jsem použil nástroje Grafana Dashboard a Node-Red a proto je níže zmiňuji. Zároveň jakožto další běžný nástroj pro vizualizaci dat ze senzorových sítí se používá Freeboard.io a tedy je zde také zmíněn.

3.1 Grafana Dashboard

V rámci této práce jsem pracoval s jedním z mnoha nástrojů, které jsou na zobrazování dat z bezdrátových senzorů vhodné a tedy zde uvádím jeho hlavní přednosti. Jedná se o projekt Grafana Dashboard, což je Open-Source dashboard a editor grafů. Aplikace obsahuje vlastní web server, standardně běžící na portu 3000. Grafana podporuje několik datových zdrojů, mimo jiné MySQL, PostgreSQL, MSSQL databáze, Graphite a další. Pro datové zdroje, které nejsou podporovány existují pluginy, zásuvné moduly, které rozšiřují možnost připojení Grafany téměř k nekonečnu. Základní sada nástrojů, je také rozšiřitelná pomocí pluginů a pokrývá téměř veškeré možnosti zobrazení dat, jako jsou tabulky, grafy, budíky a další. Díky architektuře open-source je možné, v případě nutnosti, si i vlastní možnost zobrazení dodělat. Na obrázku 3.1 jsou zobrazeny základní možnosti zobrazení dat, zahrnující klasický graf, sloupcový graf, budík a klasický text.

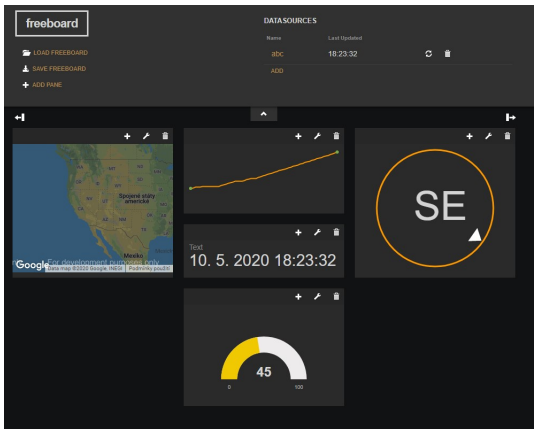
Co se zabezpečení týče, Grafana má vlastní webový server s integrovanou podporou HTTPS. Tedy jsou použity SSL certifikáty, pro zabezpečení samotného přenosu. Aplikace jako taková poté nabízí rozšířenou správu uživatelských účtů a s tím spojené zabezpečení přístupu různých uživatelů ke zdrojům dat.

Obrázek 3.1. Grafana Dashboard, ukázka možností [19]

3.2 Freeboard.io

Freeboard.io je open-source aplikace, založená na HTML a speciálně vyvinutá pro zobrazování dat formou dashboardů. Na každý dashboard se umisťují jednotlivé widgety, okénka, na kterých se zobrazují data. Rozšiřitelnost a *customizovatelnost* aplikace se provádí přes pluginy, zásuvné moduly, díky kterým je zde například možnost zobrazovat data přímo z databáze. Bohužel takováto možnost není nativní a je třeba si ji doprogramovat. Narozdíl od výše zmíněné Grafany, Freeboard nemá integrovaný webserver a tedy jsou pro jeho provoz dvě možnosti. Buď lokálně na nějakém zařízení s webserverem a nebo v cloudu, kdy je přímo freeboard hostem běžící instance. Zde je pak poplatek od 12 do 100 USD za měsíc. [20] Možnosti zabezpečení komunikace u Freeboardu jsou pak tedy otázkou použitého webserveru a programátorských schopností vývojáře v případě použití *custom* pluginů pro získání zobrazovaných dat.

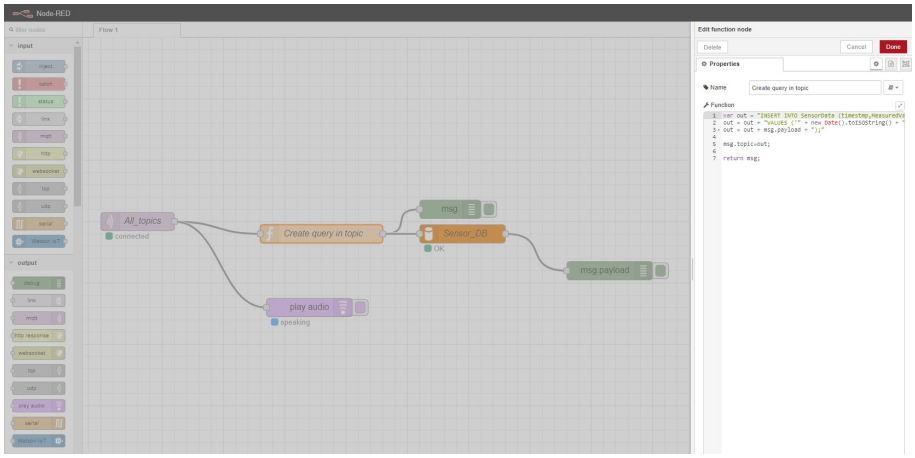
Obrázek 3.2 zobrazuje základní možnosti Freeboard.io. Bohužel, pokud by uživatel chtěl cokoli více, v 99 % případů je potřeba aby si nové zásuvné moduly doprogramoval sám.



Obrázek 3.2. Freeboard.io, ukázka možností

3.3 Node-Red

Zpracování surových dat může probíhat mnoha způsoby, v této práci jsem se rozhodl zmínit *flow-based* programovací nástroj, původně vyvinutý IBM a nyní součást JS Foundaton, Node-Red. Jedná se o jeden z nejpoužívanějších nástrojů pro zpracování dat ve světě IoT, neboť jednoduchým způsobem umožňuje nastavovat a upravovat toky dat a to propojováním tzv. nodů. Příklad *flow* (pracovní plochy) je na obrázku 3.3. Tento *flow* (tok) je použit pro přeposílání dat z MQTT do MariaDB databáze. V levé části obrázku 3.3 je vidět panel s jednotlivými nody, základními funkčními bloky, které se pak na základní ploše (uprostřed) spojují do *flow* a tedy tak tvoří *tok* dat. V pravé části je poté panel možností vybraného node.



Obrázek 3.3. NodeRED, pracovní prostředí

3.4 Databázové systémy

Těžko je možné zpracovávat a vizualizovat data bez jejich ukládání a archivace. K tomuto jsou v naprosté většině případů využívány nejrůznější databázové systémy. Samozřejmě je možné data ukládat jen tak do souborů v textové formě, ale to má pro jejich další použití více nevýhod než výhod. Z databázových systémů jsou pro archivaci tohoto typu dat vhodné relační SQL databáze. Níže popíšu dva vybrané zástupce tohoto typu databází, OracleDB a MariaDB, která vychází z MySQL.

3.4.1 Oracle Database

Oracle Database je relační databázový systém vyvinutý v 70. letech minulého století. Jedná se o jeden z nejpoužívanějších a nejdůvěryhodnějších databázových systémů na trhu.

Klíčovou vlastností, odlišující Oracle od konkurence, je rozdělení architektury na logickou a fyzickou část. To znamená velkou a poměrně jednoduchou škálovatelnost aplikací a velkou vhodnost pro tzv. grid výpočty. [21] Oracle také nabízí velmi rozsáhlé možnosti zabezpečení, počínaje standardním řízením přístupu pomocí uživatelských účtů, přes *privileges* po kompletní šifrování dat pomocí *DBMS_CRYPTO* a Oracle proprietárního programovacího jazyka *PL/SQL*.

Bohužel Oracle databáze není zkompilovaná pro procesory s ARM architekturou, a tedy jsem ji nemohl využít v mém testovacím pracovišti. Také je nutno dodat, že se nejedná o open-source systém, ale je zde nutno platit licenční poplatky.

3.4.2 MariaDB

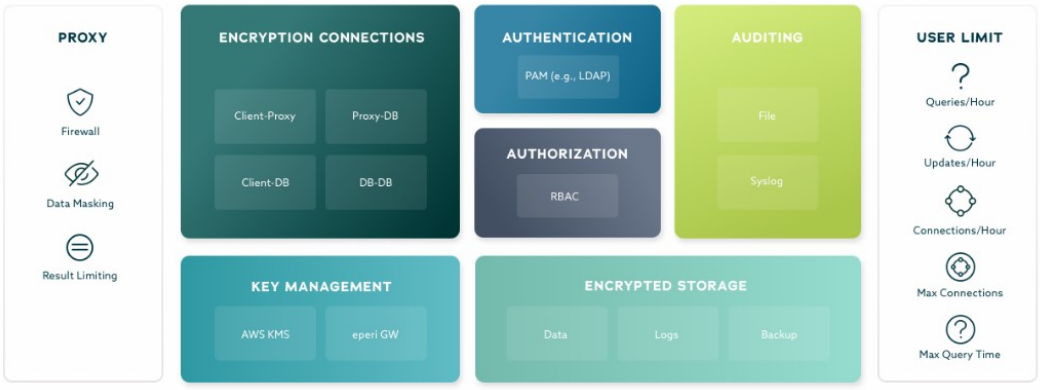
MariaDB Server je jedna z nejpopulárnějších open-source relačních databází. Je vyvíjena původním týmem, který měl na starosti MySQL a je garantováno, že zůstane open-source, na rozdíl od MySQL, které koupil Sun Microsystems a posléze Oracle. Až do verze 5.5 byla MariaDB plně kompatibilní a tedy v podstatě záměnná s MySQL, v současnosti již toto neplatí.

Systém nabízí různé možnosti rozšíření, pomocí různorodých pluginů. Jeden z častěji používaných, firewall plugin řeší bezpečnost systému, neboť nabízí široké možnosti konfigurace. Na základě tohoto je schopen zachytávat a blokovat dotazy, například za účelem prevence škodlivých SQL injection útoků, které by například mohly smazat veškeré řádky tabulky, nebo se dostat k neveřejným datům. [22]

Předchozí odstavec by měl, doufám že dostatečně, demonstrovat, proč je důležité mít správně nastavená přístupová práva uživatelů. Existuje-li uživatel jen za účelem čtení dat, což je perfektně dostačující například pro vizualizační systémy, pak by takovýto uživatel měl mít jen a pouze možnost čtení z příslušných tabulek. Takovéto omezení se řeší pomocí tzv. *permissions*, kde mezi základní oprávnění nakládání s daty patří *SELECT, INSERT, DELETE, UPDATE a EXECUTE*. Oprávněních je samozřejmě mnohem více, zde jsou vyjmenována pouze základní pro manipulaci s daty v již vytvořené databázi.

Spojení mezi klienty, proxy servery a databází může být šifrováno pomocí TLS, zatímco tabulky a serverové logy mohou být chráněny pomocí AES šifrování. Pomocí pluginů MariaDB nabízí také podporu LDAP autentizace, SSH, jednorázových hesel, či dvoufázového ověření pomocí služby Google Authenticator. [22] Obecné možnosti zabezpečení v databázovém systému MariaDB jsou zobrazeny na obrázku 3.4

V současnosti se jedná o výchozí databázový systém linuxových distribucí a zároveň je to jeden z mála databázových systémů, který podporuje ARM architekturu procesorů. Proto jsem MariaDB zvolil jako databázový systém v mém testovacím pracovišti.



Obrázek 3.4. Možnosti zabepečení u MariaDB [22]

Kapitola 4
Senzorová síť, komunikační mechanismy

Na otázku, co je senzorová síť se dá odpovědět, že senzorová síť je skupina malých, většinou bateriemi napájených senzorů, které jsou většinou bezdrátově propojeny a monitorují nějakou věc. Tyto senzory jsou pak připojeny k nějakému většímu celku, ať se jedná o internet nebo nějakou průmyslovou síť, ve které se nacházejí systémy pro analýzu a uložení dat ze senzorů. Tyto stroje samozřejmě můžeme propojit napřímo pomocí drátů, respektive propojit jejich senzory, což je samozřejmě ale většinou nepraktické. Proto používáme bezdrátové připojení a připojení přes internet. Řešením je posílat nějaké dohodnuté zprávy, v dohodnutém formátu. V případě připojení k internetu nám rodina protokolů TCP/IP zajistí dodání zprávy z jednoho zařízení na druhé, ovšem na vytvoření spojení je potřeba nějaká aplikace. Níže zmíněné protokoly jsou vše protokoly aplikační vrstvy, které většinou fungují nad protokolem TCP. Pro účely této práce budu uvažovat protokoly pro přenos zpráv na úrovni aplikační vrstvy, kde můžeme rozlišit dva základní přístupy. Prvním je sbírání dat v jednom centrálním místě (třeba cloud). Pro to jsou vhodné protokoly s centrálním brokerem, jako je třeba níže zmíněné MQTT. Někdy je ale potřeba, aby spolu více zařízení komunikovalo napřímo, každý s každým a k tomu je vhodné použít distribuovaný protokol, například DDS. Samozřejmě svět senzorů a IoT je obrovský, a ne vždy úplně standardní, kde v průmyslovém prostředí jsou často použity proprietární řešení na míru šitá dané aplikaci.

Senzorové sítě dnes našly uplatnění ve velmi mnoha různých aplikací nejen v průmyslu, ale třeba při měření znečištění ovzduší. V naprosté většině případů jsou takovéto senzorové sítě realizovány bezdrátově, kde pro bezdrátový přenos se používá různých technologií jako jsou WiFi, Bluetooth, Zigbee, LoRaWAN nebo GSM. Níže popíši jednotlivé technologie, jejich výhody, nevýhody a hlavní oblasti použití.

4.1 Senzory připojené fyzickým médiem

Mezi senzory připojené (propojené) fyzickým médiem řadím senzory, které jsou buď mezi sebou nebo s centrální sítí propojeny kabelem, ať metalickým nebo optickým. Mezi nejčastěji používaná rozhraní, pro přenos dat, u tohoto typu senzoru patří Ethernet nebo sériová linka (RS-232, RS-422 či RS-485). Co se aplikačních protokolů týče, tak v případě připojení přes Ethernet jsou možnosti velmi široké, často používané jsou IoT messaging protokoly, jako je třeba MQTT. V průmyslové aplikaci je rozšířený protokol Modbus, pro které je typické připojení přes sériovou linku, ovšem přes Ethernet funguje také.

Hlavní výhodou kabelem připojených senzorů je eliminace bateriového napájení. Díky tomu mohou být takovéto senzory použity i v místech kde je potřeba vyšší výpočetní výkon a s tím související vyšší spotřeba energie. Pokud je z nějakého důvodu nutno eliminovat elektromagnetické záření senzoru, pak jsou drátové senzory jasnou volbou. Jejich další výhodou je vyšší přenosová rychlost dat a nižší pořizovací cena.

Hlavní nevýhodou je pak nutnost fyzického připojení, s tím související složitější instalace a omezené možnosti přenositelnosti.

4.2 Bezdrátové senzory

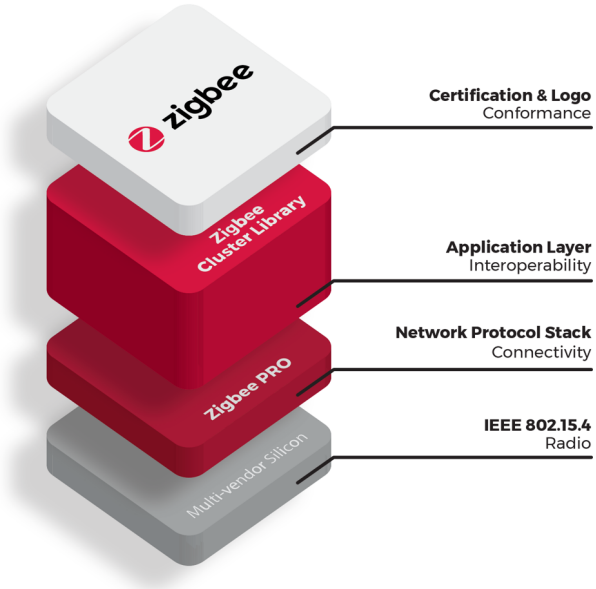
Bezdrátové senzory, jak již z jejich názvu vyplívá, ke své funkčnosti nepotřebují připojení kabelem, a tedy nabízejí jistou svobodu při jejich aplikacích. Jejich hlavní nevýhodou je nutnost vlastního pohonu, nejčastěji realizovaného bateriovým článkem. Přestože v dnešní době je technologie baterií na velmi dobré úrovni, stále je u tohoto typu senzorů třeba myslet na odběr energií, a tedy v zájmu zachování co nejdelší výdrže omezenou výpočetní kapacitu.

Díky tomuto omezení vzniklo mnoho nových protokolů, které se snaží minimalizovat energetickou náročnost těchto zařízení. „V posledních dvou letech, právě společně s rozšířením označení IoT, se pak s potřebou jednoduché, energeticky nenáročné, a hlavně bezdrátové komunikace s velkým dosahem, která by se však dala velmi miniaturizovat do velikosti čipů, vznikly a vyvinuly technologie nové kategorie LPWAN (Low-Power Wide-Area Network). Mezi nimi jsou zatím nejznámější technologie LoRaWAN a Sigfox s přenosovým dosahem až 15 km. Existují však i další systémy jako ZigBee, Z-Wave, 6LoWPAN, Thread, Cellular a dalším, které lze také do `rodiny` IoT zahrnout a obvykle slouží buď lokální několika desítek či stovek metrové vzdálenosti nebo naopak dálkový mnoha kilometrový přenos.“ [23]

V následující podsekci popíšu vybrané, jednotlivé protokoly spolu s jejich přednostmi. Záměrně zde neuvádím WiFi síť, neboť jejímu použití se věnuje kapitola 5.

4.2.1 ZigBee

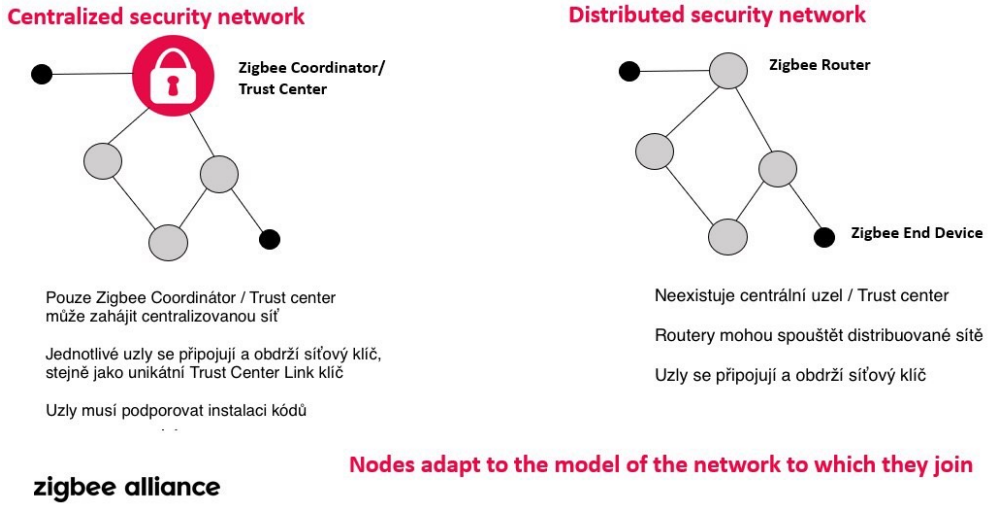
Zigbee je poměrně nový protokol vystavěný na standardu *IEEE 802.15.4* vytvořený speciálně pro účely bezdrátového propojení zařízení s nízkým výpočetním výkonem na malé vzdálenosti - jedná se o síť PAN (*Personal Area Network*). Dle popisu od *The ZigBee Aliance* se jedná o jediné kompletní řešení pro IoT, nabízí vše od fyzické sítě po univerzální aplikační jazyk, který umožňuje chytrým objektům pracovat společně. Tento standard tedy nabízí pokrytí celého ISO/OSY modelu, tedy od fyzické po aplikační vrstvu. [24] Tato prostupnost všemi vrstvami je znázorněna na obrázku 4.1



Obrázek 4.1. ZigBee architektura, filosofie standardu [24]

Pro bezdrátový přenos dat na fyzické vrstvě je použit standard *IEEE 802.15.4* , který v Evropě využívá pásmo 868.3 MHz, v Americe 902 - 928 MHz a ve zbytku světa 2.4 GHz. Podporovány jsou tři různé topologie sítě, hvězda, strom a mesh, kdy mesh je v aplikacích nejpoužívanější. ZigBee síť je složena ze třech základních konstrukčních prvků, PAN Coordinatora, routerů a koncových zařízení. PAN Coordinátor je *vlastník* celé sítě, síť vytváří, jednotlivým prvkům přiřazuje adresy a řídí přístup ostatních prvků do sítě, většinou je napájen drátem z elektrické sítě. Router bývá také napájen z elektrické sítě a jeho hlavní funkcí je směrování zpráv v síti, což ale neznamená, že zároveň nemůže fungovat jako měřící senzor. Koncovým prvkem je pak samotný senzor nebo čidlo, kde komunikace probíhá pouze s jeho přidruženým routerem nebo koordinátorem a tedy neřeší routování paketů v síti. Tato koncová zařízení jsou pak, v souladu se svým účelem poháněna bateriemi. [24]

Co se bezpečnosti týče, ZigBee, na úrovni síťové vrstvy nabízí dva modely, centralizovaný a decentralizovaný, lišící se v oprávnění jednotlivých routerů přijímat nové klienty do sítě. V obou případech je ale komunikace šifrovaná pomocí 128-bit klíčů a šifer založených na AES. Zabezpečení na aplikační úrovni v rámci jedné sítě je také možné, vytváří vlastně virtuální linky mezi jednotlivými zařízeními v síti. Používá také 128-bit verzi AES. [25] Rozdíl mezi centralizovaným a distribuovaným modelem je znázorněn na obrázku 4.2. Jak je vidět z obrázku, architektura sítě je u obou modelů zabezpečení stejná, liší se jen absencí *trust centra* u distribuovaného modelu a z toho vyplývajících omezení (za účelem zvýšení bezpečnosti) u centralizovaného modelu.



Obrázek 4.2. ZigBee rozdíly mezi modely zabezpečení [24]

Celý projekt je postaven na velmi nízké energetické náročnosti koncových zařízení a proto je velmi vhodný a používaný na různé automatizační projekty světa IoT a obecně aplikace, kde je potřeba co nejdelší životnost na baterii. Bohužel, přestože je ZigBee zaštítěn ZigBee Aliancí, jejíž členové jsou přední výrobci IoT zařízení, z ekonomických důvodů mají svá zařízení uzpůsobená tak, že komunikují jen se zařízeními stejného výrobce. Což do jisté míry limituje možnosti aplikace takto vybavených zařízení, ovšem v průmyslové sféře, kdy je většinou řešen celý projekt kompletně to není zas až takový problém.

4.2.2 Bluetooth

Bluetooth bylo vytvořeno v roce 1994 Jaapem Haartsenem a firmou Ericsson jako bezdrátová alternativa k tehdy standardně používanému sériovému portu, RS-232. Jedná se o technologii pracující, stejně jako WiFi na frekvenci 2.4 GHz. [26] Přestože se jedná o technologii, která je dnes převážně využívána pro propojení dvou zařízení, například telefonu s náhlaví soupravou, má dnes využití i v senzorových sítích. S uvedením Bluetooth Low-Energy přišla technologie, která Bluetooth pustila do světa nízko výkonných zařízení, za cenu velmi mála omezení oproti standardní verzi. Jedná se například o nemožnost přenosu audio signálu, což ale u senzorů většinou není potřeba.

Jako možnost zabezpečení Bluetooth Low-Energy nabízí 128-bit AES a v tomto ohledu je tedy plně srovnatelné s dříve zmíněným ZigBee. Narozdíl od něj se ale jedná jen o technologii prvních třech vrstev ISO/OSY modelu a tedy případné zabezpečení na aplikační úrovni je závislé čistě na aplikaci.

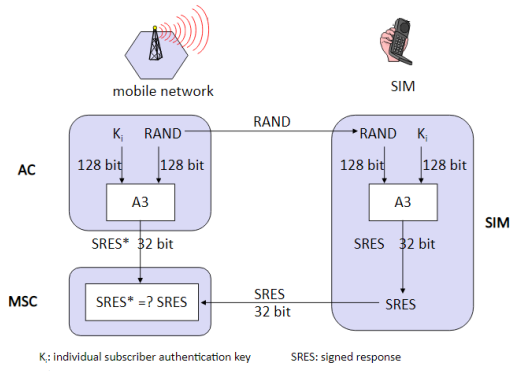
4.2.3 GSM

GSM (Groupe Spécial Mobile) je tradiční mobilní síť, tak jak ji známe a je tedy vhodná pro komunikaci se zařízeními na větší vzdálenosti. Speciálně pro potřeby světa IoT a tedy propojenosti senzorů byl vyvinut nový standard EC-GSM-IoT. Jedná se o *low power* technologii založenou na GPRS a kompatibilní s klasickou technologií GSM.

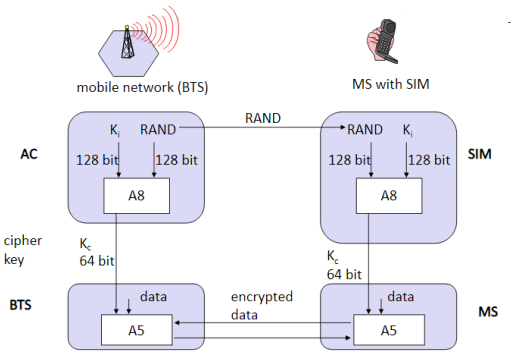
Nevýhodou této sítě je ale poměrně velká cena za přenesená data, stejně tak jako nutnost spoléhat se na třetí stranu - operátora jakožto provozovatele sítě. Nespornou výhodou je pak téměř neomezený dosah.

V otázce zabezpečení přenosu dnes asi nenajdeme lepší síť než GSM. Prvně síť ověřuje identitu mobilní stanice (přijímače - mobilního telefonu, senzoru) pomocí 128 bitového náhodného klíče a 32 bitové odpovědi. Tento algoritmus se jmenuje A3. Algoritmy A5 a A8 poté zajišťují, v pořadí, dešifrování a šifrování odesílaných dat. [27]

Schéma autentizace nové stanice v síti je znázorněno na obrázku 4.3 a schéma kódování dat již autentizované stanice je na obrázku 4.4



Obrázek 4.3. GSM schéma autentizace



Obrázek 4.4. GSM schéma zakódování dat

Kapitola 5
WiFi senzorové sítě a messaging brokery

Nejjednodušší způsob bezdrátového propojení senzorů je v dnešní době připojení WiFi, které je vhodné jak do malých soukromých aplikací, tak do velkých průmyslových aplikací. Rodina protokolů TCP/IP nám zajistí dodání zprávy z jednoho zařízení na druhé, ovšem na vytvoření spojení je potřeba nějaká aplikace. Níže zmíněné protokoly jsou vše protokoly aplikační vrstvy, které většinou fungují nad protokolem TCP. Pro účely této práce budu uvažovat protokoly pro přenos zpráv na úrovni aplikační vrstvy, kde můžeme rozlišit dva základní přístupy. Prvním je sbírání dat v jednom centrálním místě (třeba cloud). Pro to jsou vhodné protokoly s centrálním brokerem, jako je třeba níže zmíněné MQTT. Někdy je ale potřeba, aby spolu více zařízení komunikovalo napřímo, každý s každým a k tomu je vhodné použít distribuovaný protokol, například DDS. Samozřejmě svět senzorů a IoT je obrovský, a ne vždy úplně standardní. Proto je někdy potřeba použít protokoly vytvořené na míru s prvky obou výše zmíněných přístupů.

5.1 Komunikační protokoly

Aplikačních protokolů pro komunikaci dataloggeru se senzory je nepřeberné množství, ve světě internetu Věcí, jakožto nejčastější aplikaci senzorových sítí připojených na internet jsou ovšem nejběžnější message oriented protokoly, jako je MQTT. Ty spoléhají na architekturu centrálního brokeru, přes který jde veškerá komunikace. Naproti tomu jsou decentralizované protokoly, ze kterých uvedu DDS, kde žádné centrální řízení ve formě brokeru neexistuje.

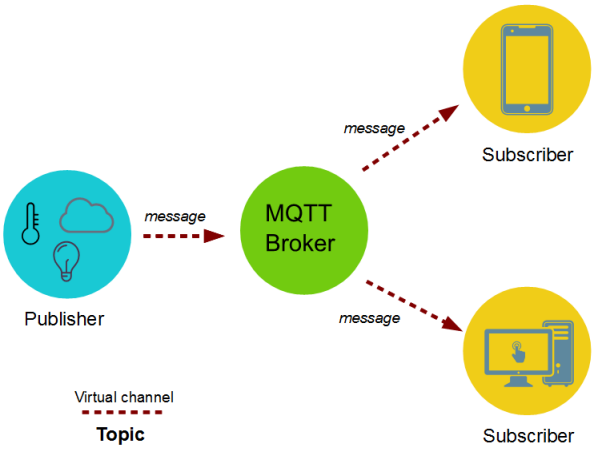
5.1.1 MQTT

Protokol MQTT (MQ Telemetry Transport, dříve Message Queuing Telemetry Transport) je jednoduchý a nenáročný IoT komunikační protokol vhodný na posílání dat od senzoru do centrálního dataloggeru přes centrální broker. Zdroj zpráv (senzor) je označován jako publisher a jejich příjemce (datalogger) jako subscriber. Tento princip je znázorněn na obrázku 5.1

Jedná se o binárně založený protokol, kde ovládací elementy jsou bity, nikoli textové řetězce. Ke každému požadavku (command) je přidružena potvrzující (ACK) zpráva. Jména témat, ID klientů a hesla jsou zakódována jako UTF-8 textové řetězce, vlastní přenášená data – náklad (payload) jsou binární data a jejich struktura záleží na dané aplikaci protokolu a aplikaci, která ho používá.

MQTT nabízí tři úrovně QoS (*Quality of Service*). QoS se vztahuje k samotnému přenosu dat, neřeší zabezpečení jako takové. Jedná se spíše o pasivní bezpečnost přenosu dat, respektive úroveň QoS určuje, s jakou přesností víme, že data dorazila do cíle. Přehledné schéma všech tří úrovní QoS je na obrázku 5.2.

Na úrovni 0 (*fire and forget*) posílá publisher zprávu PUBLISH a dále se nezajímá, byla-li doručena. Broker stejným způsobem zprávu přeposílá subscriberům. Tedy jedná se o nejjednodušší možnost, potvrzení o doručení dat neexistuje.

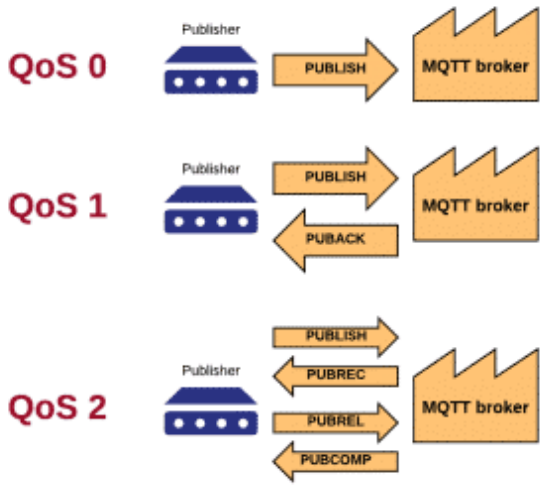


Obrázek 5.1. Funkční princip MQTT [28]

Při použití QoS úrovně 1 (*at least once*) pošle publisher zprávu brokeru a čeká na potvrzení, že zpráva byla doručena. Broker zprávu rozesílá se stejným, nebo nižším QoS subscriberům a potvrzení publisherovi posílá pouze, pokud obdržel ACK od všech, nebo většiny subscriberů. Definice všech, a nebo většiny ale záleží na implementaci brokeru.

Úroveň 2 (*exactly once*), je v podstatě úroveň 1 doplněná ještě o jedno potvrzení. Broker na zprávu od publishera odpovídá potvrzením o přijetí zprávy, publisher odpovídá potvrzením o přijetí potvrzení a na toto ještě jednou odpovídá broker. Poté končí relaci.

MQTT sám o sobě nepoužívá žádné zabezpečení pro přenos dat a je tedy nutno použít zařízení nižších vrstev ISO/OSY modelu, jmenovitě zabezpečení na úrovni TCP/IP protokolu, tedy použití TLS/SSL. Dále je možné, díky tomu, že MQTT nezkoumá, co přenáší za data, data šifrovat ještě před zasláním externě v aplikaci. V každém případě jednoduchost MQTT spočívá i v absenci jakéhokoli vlastního zabezpečení.



Obrázek 5.2. Quality-Of-Service, možnosti MQTT [29]

5.1.2 DDS

DDS je založen na real-time publish-subscribe standardu, tak jak ho definuje OMG (Object Management Group). Na rozdíl od MQTT je to protokol decentralizovaný, takže nemá žádného centrálního brokera, komunikace je zde založena čistě na peer-to-peer principu. Tento fakt přispívá ke zvýšení spolehlivosti, díky absenci single-point-of-failure, čímž centrální broker je. Publishers a subscribers zde komunikují napřímo, skrze sdílenou sběrnici, kde každý poslouchá jen to, co ho zajímá. Dalším rozdílem proti výše zmíněným protokolům je, že DDS je data-centric protokol. To znamená, že protokol ví, co přenáší za data a typ přenášených dat přímo určuje, jak bude komunikace vypadat. Protokol standardně používá jako transportní protokol UDP, ale podporuje i TCP. Jednou z výhod DDS je definice mnoha QoS pravidel, a tedy vysoká spolehlivost přenosu dat.

Co se týká zabezpečení přenosu, DDS nabízí o mnoho více možností než ostatní protokoly. Je-li použito TCP jako transportní protokol, pak se může použít TLS/SLL, což je jinde téměř jediná možnost. Při použití UDP se nabízí DTLS a OMG specifikuje DDS Security service Plugin Interface (SPI), kde bude dodefinováno zabezpečení přímo na úrovni protokolu. Bohužel i přes své přednosti, se DDS ve světě IoT příliš nepoužívá, což by se mohlo změnit s novými open-source implementacemi, jako je třeba OpenDDS.

5.2 Brokery

Message broker umožňuje komunikaci mezi více zařízeními, aplikacemi. Slouží pro validaci, transformování a směrování zpráv mezi klienty. Díky tomu je možné minimalizovat uvědomění jedné aplikace, nebo zařízení o té druhé. Například čidlo teploty venku na zahradě nemusí vědět, že zasílá data chytrému telefonu a meteostanici, stačí, že je zašle brokeru, ten se pak již postará o jejich další rozšíření.

Úkolem brokeru je tedy přijmout zprávu od aplikace nebo zařízení a něco s ní provést, ať se jedná o její přeposlání někam dále, uložení, různé chybové hlášky a podobně.

Brokery se dnes vyskytují jak v cloudové, tak místní instalaci, většina jich podporuje obě varianty, ovšem v této práci se zabývám pouze místní instalací brokerů.

Vzhledem k tomu, že pro aplikaci v senzorových sítích a IoT se nejvhodnější protokol MQTT, který byl pro tuto aplikaci přímo vytvořen, věnuji se brokerům, které jsou určeny jen pro něj.

5.2.1 Mosquitto

Jedná se o open-source implementaci MQTT protokolu od Eclipse Foundation, dnes považovanou jako téměř standardní broker pro většinu IoT aplikací využívajících MQTT protokol.

Skládá se ze tří hlavních částí, jmenovitě samostatného mosquitto serveru, publisher a subscriber klientů. Tyto je možné použít pro komunikaci s hlavním serverem, například pro otestování funkčnosti. Třetí hlavní částí je MQTT klient knihovna pro C++, kterou je možno použít pro lightweight embedded klienty, například postavené na Arduino platformě. Mosquitto jako aplikace nemá žádné GUI (Graphical User Interface) a její kompletní konfigurace tedy probíhá z příkazové řádky a v konfiguračním souboru.

Broker nabízí několik možností zabezpečení, tou nejjednodušší je nepoužívat žádné. MQTT zajišťuje ověření pomocí jména a hesla, která jsou bohužel přenášena v otevřené textové formě, z tohoto důvodu je nutné mít zabezpečenou síť stejně jako použít TLS/SSL. Duhou možností je použít ověření založené na certifikátech, kde je buď možné certifikát vyžadovat, nebo ho jen akceptovat. Pokud je certifikát vyžadován tak

jsou k dispozici nastavení *use identity as username* a *use subject as username* kde obojí mění přístupové údaje klienta, tak jak je uloží server. Poslední možností je použití PSK (pre-shared-key) šifrování, kde klient musí dodat platný identity key (může být username) a klíč, jinak broker odmítne připojení. Všechny výše zmíněné možnosti lze kombinovat, a tedy broker má široké možnosti zabezpečení připojení. [30]

Na následujícím obrázku (5.3) je uveden příklad konfigurace Mosquitto brokeru, jmenovitě vytvoření dvou nových uživatelů, jejich zabezpečení heslem a kontola uložení takovéhoto hesla. Jak je vidět, hesla jsou uložena jako haše, po aplikaci SHA-256 funkce.

```
ondra@ThinkPad-E490-O:~/mosquitto$ ls
ondra@ThinkPad-E490-O:~/mosquitto$ mosquitto_passwd -c p1.txt uzivatel1
Password:
Reenter password:
ondra@ThinkPad-E490-O:~/mosquitto$ ls
p1.txt
ondra@ThinkPad-E490-O:~/mosquitto$ cat p1.txt
uzivatel1:$6$XnClkf4KwHw/uGJp$5jFLHeTzGzYdk+OwIkQ/Nou0md7whbLvtRxDZHneicdOy1K/apMSS5kwHddfkVa5OU4
qfeZpysvZMPBbScQjtA==
ondra@ThinkPad-E490-O:~/mosquitto$ mosquitto_passwd -b p1.txt uzivatel2 heslouzivatele2
ondra@ThinkPad-E490-O:~/mosquitto$ cat p1.txt
uzivatel1:$6$XnClkf4KwHw/uGJp$5jFLHeTzGzYdk+OwIkQ/Nou0md7whbLvtRxDZHneicdOy1K/apMSS5kwHddfkVa5OU4
qfeZpysvZMPBbScQjtA==
uzivatel2:$6$U8Bi5KaRDXHHer+c$1R6zniCCQXjAsAQmhCMxIFeLA+WxxQS5s51NBid8I5DKqgqmi8jsXzgNG0bKPajyKqm
ZcAoO+1dVDIOAKcts6g==
ondra@ThinkPad-E490-O:~/mosquitto$
```

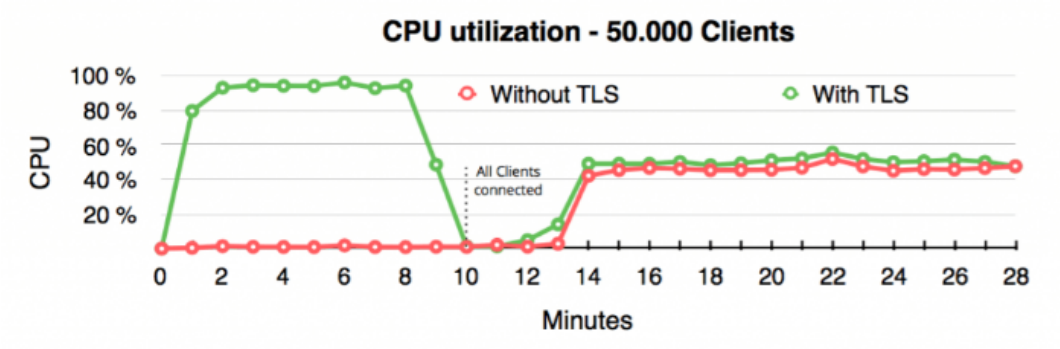
Obrázek 5.3. Ukázka konfigurace brokeru, vytvoření uživatele s heslem

5.2.2 HiveMQ

HiveMQ je enterprise zaměřený IoT broker psaný v jazyce Java. Jako takový implementuje základní požadavky MQTT, stejně jako Mosquitto, ale i něco navíc. Broker byl od základů vyvíjen s důrazem na maximální škálovatelnost a bezpečnostní koncepty vhodné pro velké firmy.

Díky orientaci na firmy je ovšem HiveMQ vysokovýkonným (highperformance) brokerem a je zamýšlen pro provoz na serverovém hardwaru. Broker je sice schopen fungovat i na malých embedded zařízeních, jako je RaspberryPi, ale díky většímu overheadu nebude tak rychlý a stabilní jako Mosquitto.

Zabezpečení přenosu dat, je zde řešeno stejně jako u Mosquitta přes SSL/TLS. Z Pohledu MQTT je jediným rozdílem mezi Mosquittem a HiveMQ jiná cílová skupina, jiné uplatnění na trhu. Oba brokery plně implementují poslední verze MQTT standardu, a tedy obě používají TLS, je-li nakonfigurováno. HiveMQ je díky zacílení na enterprise podniky náročnější na výpočetní výkon, ale to kompenzuje vyšší stabilitou a výkonem při větším počtu připojených klientů. Jeho bonus oproti standardu MQTT je implementace WebSockets (integrovaný webserver), a tedy schopnost zasílat MQTT zprávy přes HTTP protokol, a PROXY protokolu, který je velmi užitečný, pokud je připojení řešeno přes load-balancer, to znamená rozloženo na proxy servery. Vliv využití procesorového času na server s běžící instancí HiveMQ brokeru a větším počtem připojených klientů je zobrazen na obrázku 5.4. Zde je patrný výrazně vyšší požadavek na výpočetní výkon v počáteční fázi připojování klientů při použití TLS zabezpečení, oproti variantě bez. O ekonomické stránce tohoto jevu je více v kapitole 7.



Obrázek 5.4. CPU zátěž při připojení 50 000 klientů na HiveMQ brokeru [31]

Kapitola 6
Testovací pracoviště

6.1 Konstrukce

Demo se skládá z použití WiFi desky Wemos D1, mini-PC RaspberryPi 3B+ a senzoru teploty DHT11 jakožto vzorového senzoru se stabilní spotřebou elektrické energie.

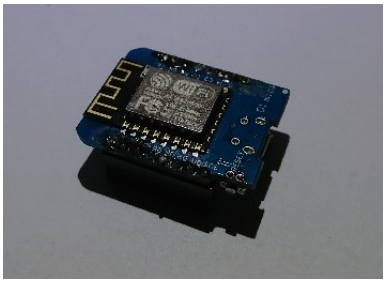
Jednou z hlavních částí dema je miniPC Raspberry Pi (na obrazku 6.1). Jako operační systém je použit Raspbian, linuxový OS založen na Debianu. Deska je napájená 5V přes micro-USB port, v testovacím prostředí je použit síťový adaptér z 230V, ale je možné použít i powerbanku. To ale v tomto případě není vhodné, neboť je Raspberry použito jako centrální server a tedy se od něj neočekává energetická nenáročnost.

Obrázek 6.1. RaspberryPi 3B+

Název	RASPBERRY Pi 3 Model B+ Broadcom Quad-Core BCM2837B0 1.4 GHz 64bit VideoCore IV 3D 1GB Wi-Fi 802.11 b/g/n, Bluetooth 4.2
Funkce v Demu	Wi-Fi modul, MQTT broker NodeRED server, MySQL databáze Grafana Webserver
Cena	1 1137 Kč [32]

Tabulka 6.1. Údaje o RaspberryPi

Druhou hlavní částí je deska WeMos D1 mini (na obrázku 6.2) , což je malá deska založená na platformě Arduino s integrovaným WiFi modulem. Napájení je stejně jako u Raspberry řešeno pomocí 5V micro-USB portu. Ten je v tomto případě ale připojen přes ampérmetr do powerbanky. Přestože toto propojení není ideální, co se energetické náročnosti týče, po domluvě s vedoucím práce jsem se rozhodl pro tuto cestu, místo připojení desky přímo na 3.3V vstup.



Obrázek 6.2. Wemos D1 mini

Název	ESP8266 ESP-12E OTA WeMos D1 CH340 WiFi Arduino IDE UNO R3
Funkce v Demu	Wi-Fi modul datalogger napájení pro senzory
Cena	148 Kč [33]

Tabulka 6.2. Údaje o WeMos D1 mini

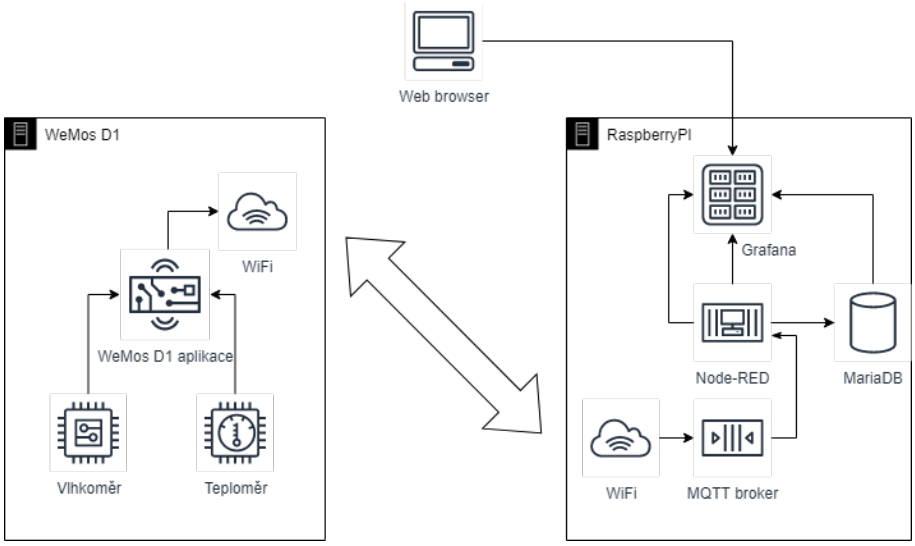
Jako příklad senzoru je využit teploměr a vlhkoměr DHT11. Jedná se o senzor, který měří teplotu s přesností na $\pm 1°C$ a vlhkost s přesností $\pm 4\%$, tedy je naprosto dostačující pro běžné orientační měření teploty. Tento senzor jsem vybral pro jeho stálý odběr elektrické energie a tedy jistou stabilitu následujícího měření, neboť senzor je aktivní pořád a jen se vzorkují data v čase, kdy je potřeba je získat.

Název	DHT11
Funkce	Teploměr a vlhkoměr
Cena	54 Kč [34]

Tabulka 6.3. Údaje o DHT11

6.2 Funkce

Celé pracoviště je sestaveno z RaspberryPi počítače a WeMos D1 desky, které spolu komunikují přes WiFi síť. Jako router je použit obyčejný domácí směrovač od firmy ASUS s podporou WPA-PSK, tedy síť je zabezpečena, čímž je alespoň základně řešena bezpečnost přenosu dat, a to v podstatě *zadarmo*. Zapojení testovacího pracoviště, po funkcionální stránce, je znázorněno na blokovém diagramu 6.3. Jednotlivým částem se poté věnují následující podsekce.



Obrázek 6.3. Blokové schéma zapojení testovacího pracoviště

6.2.1 Část na WeMos D1 mini

Deska D1 je použita jako datalogger a WiFi modul pro připojené senzory, kde získaná data od nich posílá přes protokol MQTT do RaspberryPi, jako do centálního brokeru. Napájení je řešeno powerbankou, přes 5V USB port. Na digital pin 6 desky D1 je připojen senzor DHT11, tedy teploměr a vlhkoměr. Data z tohoto senzoru jsou pravidelně každých 10 sekund vyčítaná a přes MQTT zasílaná do Raspberry. Dle módu, který jsem zrovna testoval v mezičase, kdy nebylo potřeba využít procesorový čas byla deska buď nečinná nebo v tzv. *deep sleep* módu. To znamená, že napájen je pouze obvod reálného času, zbytek desky je odpojen od napájení za účelem úspory elektrické energie.

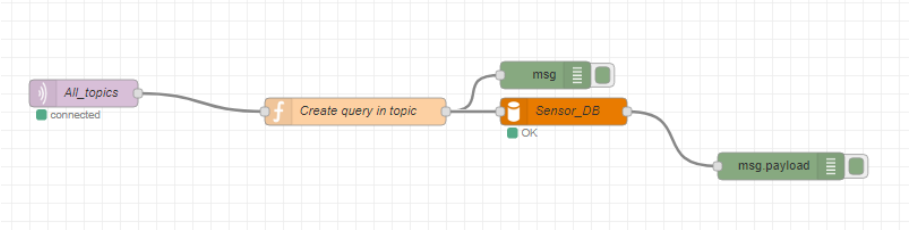
Struktura zasílaného paketu je následovná. Topic je všude stejný, v případě více desek D1 by odlišoval jejich lokalitu. Data jsou zasílaná jako pole znaků, kde separátorem je čárka. Formát je změřená hodnota, identifikační číslo senzoru a násobič hodnoty. Ten jsem použil pro eliminaci potřeby zasílat desetinná čísla. Identifikátor senzoru pohybu je 1, teploty 2 a vlhkosti 3. Tyto identifikátory se odvíjí od jejich konfigurace v databázi, která je popsána v následující podsekci.

Pro odeslání MQTT paketu jsem použil veřejně dostupnou knihovnu *PubSubClient.h*. Publikování zprávy je tedy provedeno následovně:

```
client.connect(clientID, mqtt_username, mqtt_password); // připojí se
                                                        // k brokeru
delay(100); // zpoždění nutné k správnému proběhnutí připojení
client.publish(topic, payload); // publish payloadu v daném topic
```

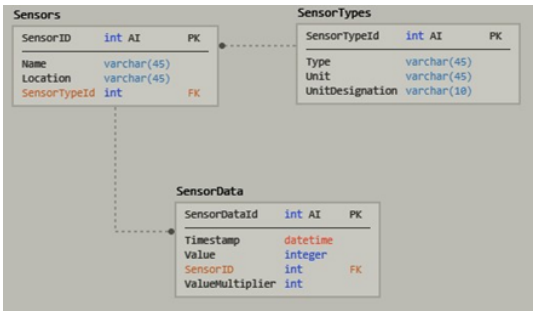
6.2.2 Část na RaspberryPi

MiniPC Raspberry Pi je v projektu použito jako centrální server. Jako hlavní zde běží centrální MQTT broker, dále NodeRED, SQL databáze (MariaDB) a webový server s aplikací Grafana. Jako MQTT broker je použito Mosquitto pro jednoduché použití a plnou optimalizaci. Zkoušel jsem i HiveMQ, které je konfiguračně i funkčně složitější, a po funkční stránce jsem nepozoroval rozdíl. Oba brokery mají 100 % implementaci MQTT a tedy jsou funkčně shodné. Jako vyhodnocovací aplikaci jsem zvolil NodeRED, převážně pro její plnou připravenost na takovýto typ využití a podporu MQTT klienta. Jedná se o *industry standard* aplikaci.



Obrázek 6.4. Schéma Node-RED aplikace

NodeRED je připojená jako subscriber na broker, kde poslouchá všechny topicy. Jediným používaným a očekávaným topicem v rámci této práce je *Test*. Příchozí zprávy jsou formátovány v souladu s formátováním odeslaným z D1 desky. NodeRED zpracuje příchozí zprávy, vytvoří SQL insert příkaz a vloží data do MariaDB SQL databáze. Nad kterou poté běží vizualizační software Grafana, který má do databáze přímý přístup přes vlastní connector. Tok dat přes NodeRED je znázorněn na obrázku 6.4, schéma architektury databáze je poté na obrázku 6.5.



Obrázek 6.5. Schéma architektury databáze

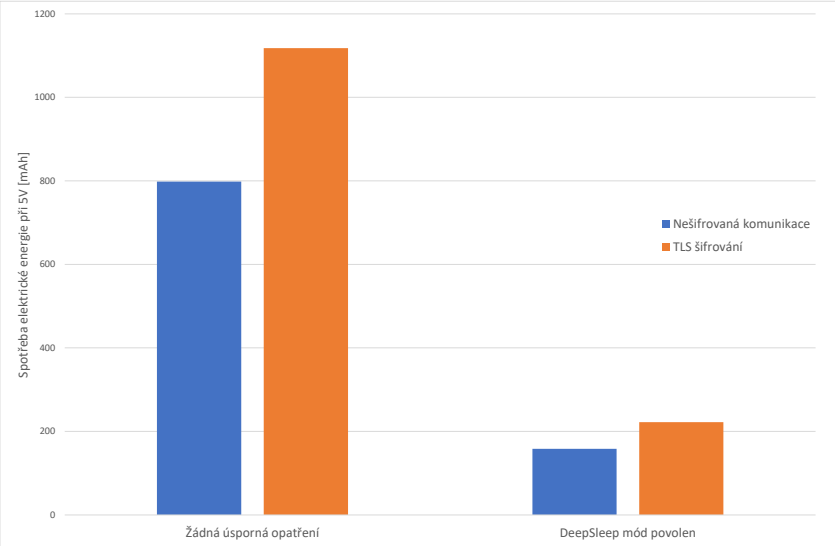
6.3 Naměřená data

V rámci této práce jsem se zabýval i velikostí vlivu šifrování přenosu dat na energetickou náročnost aplikace. Testovací pracoviště, zapojené dle popisu v sekci 6.2.1 jsem nechal vždy 24 hodin měřit tepotu a vlhkost v pokoji a data přenášet přes protokol MQTT do RaspberryPi. Zabývám se i možností optimalizace power managementu desky D1 za účelem snížení odbytu energie. V tabulce 6.4 jsou naměřená data spotřeby elektrické energie. V grafu 6.6 pro lepší znázornění zobrazena tabulka 6.4. Je vidět, že závislost spotřeby na zvoleném způsobu zabezpečení je lineární, zabezpečené spojení má o cca 40%větší spotřebu.

Funkce	Spotřeba [mAh]
Nešifrované spojení, žádné úsporné opatření	798
Šifrované spojení (TLS v1.2), žádné úsporné opatření	1118
Nešifrované spojení, DeepSleep mód povolen	158
Šifrované spojení (TLS v1.2), DeepSleep mód povolen	222

Tabulka 6.4. Spotřeba elektřiny na desce D1

Rozdíl mezi šifrovanou a nešifrovanou komunikací je samozřejmě i v objemu přenesených dat. Objem dat nutných pro navázání komunikace a přenesení data o teplotě a



Obrázek 6.6. Závislost spotřeby energie na způsobu zabezpečení a úsporném módu

No.	Time	Source	Destination	Protocol	Length	Info
3050	31.770494548	192.168.1.185	192.168.1.6	MQTT	96	Connect Command
3052	31.771208237	192.168.1.6	192.168.1.185	MQTT	58	Connect Ack
3061	32.054281155	192.168.1.185	192.168.1.6	MQTT	76	Publish Message [Test]
3066	32.161679523	192.168.1.185	192.168.1.6	MQTT	76	Publish Message [Test]

Obrázek 6.7. Nezabezpečené spojení, testovací pracoviště

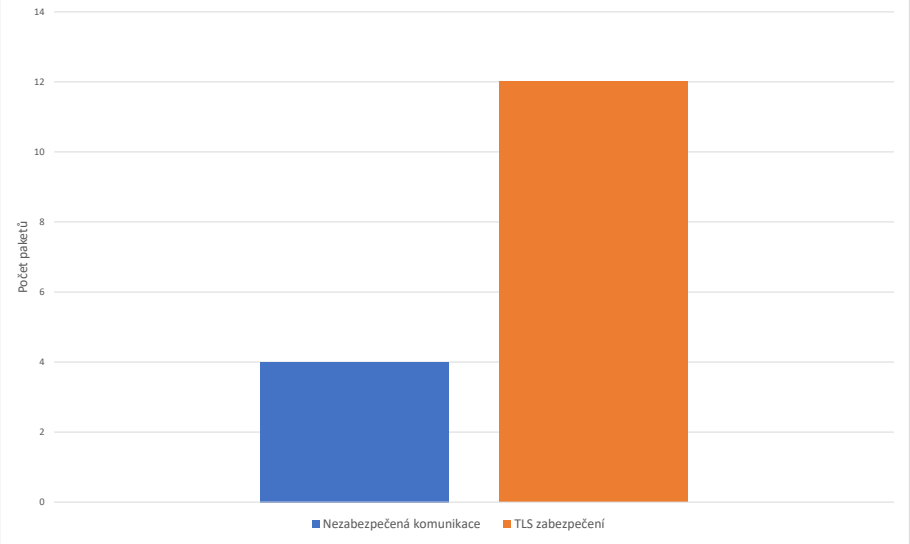
No.	Time	Source	Destination	Protocol	Length	Info
431	1.585456055	192.168.1.185	192.168.1.6	TLSv1.2	270	Client Hello
452	1.610860112	192.168.1.6	192.168.1.185	TLSv1.2	590	Server Hello
461	1.657664076	192.168.1.6	192.168.1.185	TLSv1.2	590	[TCP Window Full] , Certificate [TCP segment of a reassembled PDU]
463	1.657861054	192.168.1.6	192.168.1.185	TLSv1.2	82	Server Key Exchange, Server Hello Done
682	2.124136588	192.168.1.185	192.168.1.6	TLSv1.2	96	Client Key Exchange
774	2.177366253	192.168.1.185	192.168.1.6	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
776	2.178039011	192.168.1.6	192.168.1.185	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
791	2.197727103	192.168.1.185	192.168.1.6	TLSv1.2	125	Application Data
792	2.197972206	192.168.1.6	192.168.1.185	TLSv1.2	85	Encrypted Alert
794	2.198116945	192.168.1.6	192.168.1.185	TLSv1.2	87	Application Data
892	2.479895829	192.168.1.185	192.168.1.6	TLSv1.2	105	Application Data
936	2.638961183	192.168.1.185	192.168.1.6	TLSv1.2	105	Application Data

Obrázek 6.8. Zabezpečené spojení, testovací pracoviště

vlhkosti při použití nešifrovaného spojení je vidět na obrázku 6.7. Objem dat pro přenesení té samé informace, ovšem zabezpečené pomocí TLS v1.2 je vidět na obrázku 6.8. Pro názornost je rozdíl mezi těmito dvěma způsoby zabezpečení přenosu dat znázorněn na grafu 6.9.

6.4 Výsledky z naměřených dat

Jak je možno vidět z tabulky 6.4 a grafů 6.6, 6.9, v případě použití zabezpečeného přenosu pomocí TLS je spotřeba elektrické energie o 40 procent větší, než v případě nezabezpečeného přenosu. Také vytížení síťové infrastruktury je 3x vyšší. S ohledem na tyto skutečnosti je tedy vždy potřeba zvážit jestli daná aplikace vyžaduje zabezpečení, případně jestli není dostatečné jen zabezpečení WiFi sítě, například pokud je síť určená

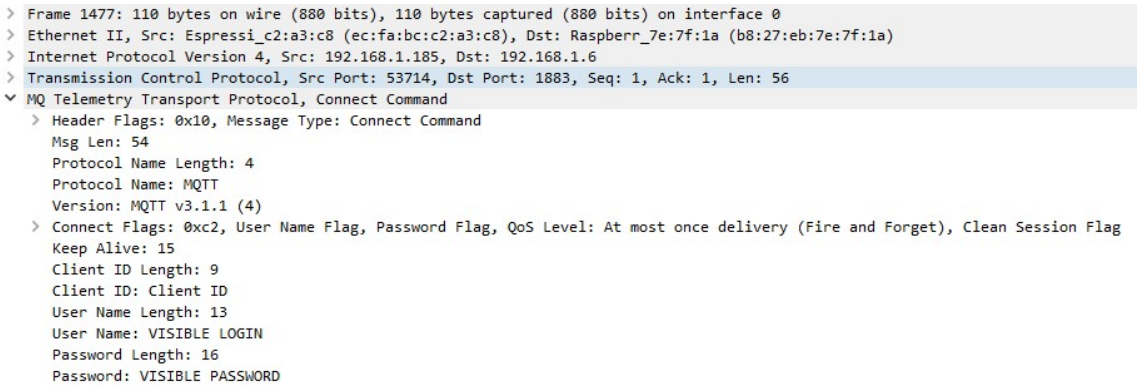


Obrázek 6.9. Závislost počtu přenesených paketů na způsobu zabezpečení

jen a pouze pro komunikaci mezi senzory bez přístupu do Internetu. Více o ekonomickém vlivu zabezpečení na aplikaci je uvedeno v kapitole 7.

Dále je nutno upozornit na architekturu protokolu MQTT, který v nezabezpečeném režimu posílá heslo uživatele MQTT brokeru v *plain textovém* formátu. Toto je možno pozorovat na obrázku 6.10, což je odchycená část autentizace klienta k MQTT brokeru. V případě, že síť je přístupná více zařízením, nebo je jinak náchylná k připojení neautorizovaných uživatelů, existuje riziko, že si toto heslo může přečíst kdokoli. To není dobrá situace, bohužel praxe ukazuje, že toto se děje a je zneužíváno. Pro ilustraci lze uvést případ, popsaný v internetovém článku, kdy přes vyhledávač Shodan byl nalezen nezabezpečený MQTT broker a během chvíle měl útočník kompletní přehled o dění v rodinném domě včetně real-time polohy majitele domu. Článek je převzat z [35].

Z obrázků 6.7 a 6.8 je také možné pozorovat zvýšení doby přenosu dat z dataloggeru do centrálního brokeru v případě použití TLS zabezpečení. Přenos dat od navázání komunikace po poslední datový paket trval v nezabezpečeném případě 0.39 sec. V případě zabezpečeném pomocí TLS se jednalo ale již o 1.05 sec pro stejná data. Toto zpoždění přičítám, jak nutnosti přenést větší počet paketů tak samozřejmě i výrazně vyššímu výpočetnímu nároku na desku D1 během zahajovací fáze komunikace. Pro aplikace, kde je co nejnižší latence nutností a zároveň není možno využít drátových senzorů je tedy volba zabezpečené komunikace nevhodná. Bohužel v praxi jsou takovéto aplikace většinou aplikacemi kritickými, kde je zabezpečení nutné. A tedy jde vždy o nějaký kompromis, případně jsou místo bezdrátových senzorů voleny na instalaci složitější drátové senzory.



Obrázek 6.10. Zranitelnost protokolu MQTT, nezabezpečené heslo

Kapitola 7
Ekonomické zhodnocení

Tato kapitola se zabývá vlivem zjištění z testovacího pracoviště na ekonomickou stránku aplikace. První sekce této kapitoly řeší otázku pořizovacích nákladů na WiFi senzorovou síť s vizualizací naměřených dat, tedy prvotní investici. Druhá sekce se poté věnuje provozním nákladům, především pak nákladům na energie. Upozorňuji však, že veškeré ceny a náklady níže zmíněné jsou ceny surového materiálu, tedy bez nákladů na práci.

7.1 Nákupní cena, prvotní náklady

Vzhledem k dostupnosti použitého hardwaru, jmenovitě RaspberryPi a WeMos D1 mini desky, je sestavení podobného zařízení pro enterprise (podnikové) využití poměrně levnou záležitostí. Veškerý použitý software je *open-source* a tedy odpadají náklady spojené s jeho pořízením. V tabulce 7.1 je odhad nákladů pro pořízení systému na sledování výroby s jedním centrálním brokerem na RaspberryPi spolu s vizualizací na obrazovku 4K televizoru (s přístupem na internet). V mnou sestaveném testovacím pracovišti jsem potřebu televize, jakožto zobrazovače naměřených hodnot, obešel zobrazováním dat na monitoru mého PC. Náklad za televizi je tedy možné z kalkulace vypustit. Tato kalkulace vychází z aktuálních tržních a katalogových cen jednotlivých komponent, dle dostupnosti na českém trhu a byly převzaty z [32, 36, 33–34, 37].

Zboží	Počet	Cena / Kus	Cena celkem
RaspberryPi	1	1137 Kč	1137 Kč
128 Gb paměťová karta pro RPi	1	794 Kč	794 Kč
WeMos D1 mini (ESP 8266)	100	148 Kč	14800 Kč
Senzory, čidla (DHT11)	100	54 Kč	5400 Kč
Televizor LG 60UM7100PLB	1	13718 Kč	13718 Kč
		Celkem, bez TV	22131
		Celkem, včetně TV	35849

Tabulka 7.1. Ceny HW pro enterprise aplikaci

Předpokládám nákup jednoho sta dataloggerů (D1 desky), ke každému přísluší jeden senzor. Jako běžný senzor v průměrné cenové hladině beru DHT11, tedy senzor teploty a vzdušné vlhkosti. Samozřejmě je možné pořídit jak výrazně levnější senzory, tak také výrazně dražší. V případě pořízení takovýchto senzorů by bylo třeba tuto kalkulaci upravit. Televizor jsem zvolil v průměrné ceně a velikosti, spíše jako odhad z důvodu velké závislosti na požadavcích klienta. Existenci dostatečně výkonné WiFi infrastruktury již předpokládám, a tedy nezahrnuji do výpočtů. Dále v ceně není zahrnuto úložiště pro sbíraná data, možno je buď řešit cestou lokálního uložení na paměťovou kartu RaspberryPi, nebo externě na disk.

V případě aplikace pro průmyslové prostředí, tedy aplikací velkého rozsahu, předpokládám nižší procento využití *open-source* software. Tím by samozřejmě výrazně vzrostly náklady, jak na prvotní pořízení tak na roční provoz.

7.2 Zabezpečení a spotřeba energií

V návaznosti na kapitolu 6.3 je třeba zdůraznit vliv šifrování na spotřebu elektrické energie, při průměrné ceně za 1 kWh pro rok 2020, 4.76 Kč [38]. Toto znázorňuje tabulka 7.2, kde počítám s non-stop provozem D1 desky, po dobu jednoho roku.

Zabezpečení	Úsporný mód	Spotřeba	Spotřeba 100 desek za rok	Cena spotřebované energie za rok
Ne	Ne	0.167 Wh	146292 Wh	696.3 Kč
TLS	Ne	0.233 Wh	203232 Wh	967.4 Kč
Ne	DeepSleep	0.032 Wh	29908 Wh	137.6 Kč
TLS	DeepSleep	0.046 Wh	40296 Wh	191.8 Kč

Tabulka 7.2. Spotřeba a cena elektrické energie, vliv zabezpečení

Jak je možno pozorovat z tabulky 7.2, úspora energie při nepoužití zabezpečeného připojení je stále 40%, ale její absolutní výše není až taková, při současné ceně elektrické energie. Mnohem větší úsporu vyrobí použití úsporných opatření zabudovaných přímo do desky WeMos D1, jmenovitě použití DeepSleep módu. Nutno ale upozornit, že tyto hodnoty se vztahují jen a pouze ke spotřebě samotné desky. Zabezpečené připojení klade několikanásobně větší nároky na síťovou infrastrukturu i výpočetní výkon serveru s brokerem. Toto je při případné instalaci do firemního prostředí také třeba zvážit. Dále je nutno upozornit, že Deska D1 a senzory na ní připojené jsou zamýšleny jako bezdrátová čidla, tedy dá se předpokládat, že budou umístěna na hůře přístupných místech, možná bez přístupu k elektrické energii. V tu chvíli je nutné bateriové napájení, kde je cena za každou kilowatthodinu mnohonásobně vyšší než při přímém odběru z rozvodné sítě.

Kapitola 8
Vyhodnocení

Cílem této práce je analyzovat služby pro zprostředkování, ukládání a vizualizaci dat, a bezpečnosti komunikace mezi komponenty senzorové sítě. Tohoto jsem dosáhl nejdříve analýzou služeb a jimi využívaných algoritmů pčítačové bezpečnosti. Poté, praktickým ověřením zjištění na testovacím pracovišti, kde jsem se věnoval ekonomickým důsledkům zabezpečené komunikace. Během psaní této práce jsem se nesetkal s nedostatkem kvalitních zdrojů, ze kterých jsem mohl čerpat. To přikládám i velkému zájmu odborné veřejnosti o téma bezpečnosti komunikace, převážně v dnešním stále rostoucím světě IoT technologií.

V kapitole 2 jsem se zabýval běžně používanými algoritmy počítačové bezpečnosti. Zaměřil jsem se na rozdělení používaných algoritmů včetně jejich popisu. Zaměřil jsem se i na charakteristické vlastnosti jednotlivých algoritmů, které definují jejich uplatnitelnost v jednotlivých segmentech světa IoT a převážně poté jejich použití v bezdrátových senzorových sítích.

Na v praxi běžně používané prostředky pro zprostředkování, ukládání a vizualizaci dat jsem se zaměřil v kapitole 3. V první části kapitoly jsem analyzoval možnosti prostředků pro zpracování a vizualizaci dat. Zabýval jsem se především na charakteristickými vlastnostmi těchto systémů a porovnáním vybraných systémů. Druhou část kapitoly jsem zaměřil na databázové systémy, jakožto systémy pro ukládání dat.

V kapitole 4 jsem se zaměřil na definici senzorových sítí, jejich členění a použití. Zmíněny jsou také typické vlastnosti a aplikace jednotlivých typů sítí spolu s charakteristikou jejich bezpečnostní filozofie. Sítě jsem analyzoval i pohledem požadavku na co největší energetickou nenáročnost.

Kapitola 5 je zaměřená na WiFi senzorové sítě. Důkladně jsem se zaměřil na vybrané komunikační protokoly, rozebral jsem jejich možnosti. Vzhledem k dominanci protokolu MQTT ve světě bezdrátových sítí jsem se poté zaměřil na dva nejpoužívanější MQTT brokery, a to sice Mosquitto a HiveMQ. Věnoval jsem se charakteristickým vlastnostem těchto brokerů a jejich typickým aplikacím.

Praktické poznatky demonstrované na testovacím pracovišti sestaveném z vybraného senzoru a služeb jsem poté demonstroval v kapitole 6. První část kapitoly jsem věnoval konstrukčnímu a funkčnímu popisu testovacího pracoviště. Snažil jsem se, aby demo odpovídalo modelovému použití v běžné aplikaci, a to včetně použití modelového senzoru DHT11, který se dá po stránce ekonomické považovat za univerzální senzor. Druhá část byla věnována vlastním měřeným datům, jejich zhodnocení a interpretování z praktické části. Ekonomickému zhodnocení naměřených dat, jakožto ekonomickému zhodnocení celé práce se poté věnuje kapitola 7, kde je rozebírán ekonomický smysl zabezpečení. Je nutno podotknout, že ačkoli zabezpečení komunikace s sebou přináší až trojnásobnou režii, oproti komunikace bez zabezpečení, jeho přínos je v určitých momentech k nezaplacení. Pokud jsou data přenášena alespoň z části nezabezpečeným kanálem (např. WiFi síť nezabezpečená heslem nebo se slabým heslem) a hrozí tedy riziko odposlechnutí paketu, u protokolu MQTT je použití zabezpečení téměř nutností, bez ohledu na náklady. Máme-li aplikaci, kdy je veškerý přenos dat řešen v zabezpečené síti, nejlépe

bez připojení do internetu a zároveň se nejedná o přenos tajných dat, pak je možno za zvýšené opatrnosti zabezpečení vynechat a tím šetřit náklady.

Věřím tomu, že stejným přínosem jako pro mne bude tato práce i pro veřejnost, která přemýšlí o aplikaci bezdrátových senzorových sítí, nebo systému pro chytrou domácnost, či jiné IoT aplikaci a není jim lhostejná bezpečnost komunikace. V budoucnu bych na tuto práci rád navázal a věnoval se bezpečnostím vylepšením protokolu MQTT.

Literatura

[1] Internet Privacy - Cyber Security. *Le-vpn.com* [online]. online: online, 2018 [cit. 2020-03-13]. Dostupné z: https://www.le-vpn.com/internet-privacy-cyber-security/

[2] Historie kryptografie. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-04-13]. Dostupné z: https://cs.wikipedia.org/wiki/Historie_kryptografie

[3] DIFFIE, Whitfield a Martin E. HELLMAN. New Directions in Cryptography. *IEEE Transactions on Information Theory* [online]. **1976** (IT-22), 11 [cit. 2020-03-13]. Dostupné z: https://ee.stanford.edu/ hellman/publications/24.pdf

[4] ISLAM, Mazhar, Mohsin SHAH, Zakir KHAN, Toqeer MAHMOOD a Muhammad Jamil KHAN. A New Symmetric Key Encryption Algorithm Using Images as Secret Keys. *2015 13th International Conference on Frontiers of Information Technology (FIT)* [online]. IEEE, 2015, 2015, , 1-5 [cit. 2020-04-13]. DOI: 10.1109/FIT.2015.12. ISBN 978-1-4673-9666-0. Dostupné z: http://ieeexplore.ieee.org/document/7420966/

[5] ČLUPEK, Vlastimil. *Kryptografický protokol výměny klíčů Diffie-Hellman* [online]. Brno, 2010 [cit. 2020-05-08]. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/6539/final-thesis.pdf?sequence=6&isAllowed=y
Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Jiří Sobotka.

[6] Princip asymetrické šifry. In: *Tutorialspoint.com* [online]. online: online, online [cit. 2020-05-10]. Dostupné z: https://www.tutorialspoint.com/cryptography/images/public_key_cryptography.jpg

[7] ŠKODA, Martin. *Implementace symetrické blokové šifry AES na moderních procesorech.* [online]. Brno, 2014 [cit. 2020-05-08]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=84166
Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Ondřej Rášo.

[8] VINCK, A.J. Han. Diffieho-Hellmanova výměna klíčů znázorněna pomocí míchání barev. In: *Wikimedia Commons* [online, obrázek]. 19 July 2016 [cit. 2020-05-10]. Dostupné z: https://commons.wikimedia.org/wiki/File:Diffie-Hellman_Key_Exchange_(cs).svg

[9] MATĚJOVÁ, Lucie. *RSA* [online]. [cit. 2020-04-18]. Dostupné z: http://www.kryptografie.wz.cz/data/RSA.htm

[10] LIU, Jen-Chang. RSA encryption/decryption example: Chap. 8,9: Introduction to number theory and RSA algorithm. In: *Slideplayer* [online]. 2004 [cit. 2020-05-10]. Dostupné z: https://slideplayer.com/slide/5023333/

[11] DURČÁK, Pavel. Symetrické a asymetrické šifrování. In: *NaPočítači.cz* [online]. Praha: Dashöfer Holding, 2018 [cit. 2020-05-15]. Dostupné z:

https://www.napocitaci.cz/33/symetricke-a-asymetricke-sifrovani-uniqueidgOkE4NvrWuNY54vrLeM677jX7sp3Lu-ZpLpGVMy1prA/

[12] Secure Hash Standard (SHS). In: *National Institute of Standards and Technology — NIST* [online]. Gaithersburg (MD): National Institute of Standards and Technology, 2015 [cit. 2020-05-14]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[13] 3.2. SSL/TLS, ECC, and RSA. *RedHat* [online]. 2018 [cit. 2020-04-17]. Dostupné z: https://access.redhat.com/documentation/en-US/Red_Hat_Certificate_System/8.0/html/Deployment_Guide/SSL-TLS_ecc-and-rsa.html

[14] SSL, TLS and PKI History. *Feistyduck* [online]. [cit. 2020-04-17]. Dostupné z: https://www.feistyduck.com/ssl-tls-and-pki-history/

[15] Security Standards and Name Changes in the Browser Wars. In: *Tim Dierks* [online]. 2014-04-23 [cit. 2020-04-17]. Dostupné z: http://tim.dierks.org/2014/05/security-standards-and-name-changes-in.html

[16] Certifikát SSL: Jak funguje SSL zabezpečení? - OVH. In: *Webhosting, Cloud a Dedikované servery - OVH* [online]. [cit. 2020-05-10]. Dostupné z: https://www.ovh.cz/images/ssl/schema_ssl_subs.jpg

[17] SIMMONS, Gustavus J. AES: Cryptology. In: *Encyclopædia Britannica* [online]. Chicago (IL): Encyclopædia Britannica, 2014 [cit. 2020-05-07]. Dostupné z: https://www.britannica.com/topic/AES

[18] STALLINGS, William. *Cryptography and Network Security: Principles and Practices*. Fourth Edition. Upper Saddle River (NJ): Prentice Hall, 2005. ISBN 0131873164.

[19] Demo / Grafana Dashboard. In: *Grafana: The open observability platform* [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://play.grafana.org/d/3SWXxreWk/grafana-dashboard?orgId=1

[20] Freeboard. *Freeboard - Dashboards For The Internet Of Things* [online]. New York (NY): BugLabs [cit. 2020-05-07]. Dostupné z: https://freeboard.io/

[21] Oracle Database (Oracle DB). In: *Technopedia* [online]. 2017 [cit. 2020-05-07]. Dostupné z: https://www.techopedia.com/definition/8711/oracle-database

[22] MariaDB Security Features. In: *MariaDB* [online]. Espoo: MariaDB, c2020 [cit. 2020-05-11]. Dostupné z: https://mariadb.com/database-topics/security/

[23] VOJÁČEK, Antonín. Základní úvod do oblasti internetu věcí (IoT). In: *Automatizace.hw.cz* [online]. Praha: HW server, 2016 [cit. 2020-05-07]. Dostupné z: https://automatizace.hw.cz/zakladni-uvod-do-oblasti-internetu-veci-iot.html

[24] Zigbee: The Sandard for the IoT. In: *Zigbee alliance* [online]. Davis (CA): zigbee alliance, 2019 [cit. 2020-05-08]. Dostupné z: https://zigbeealliance.org/wp-content/uploads/2019/12/Zigbee-Technical-2019.pptx

[25] FAN, Xueqi, Fransisca SUSAN, William LONG a Shangyan LI. *Security Analysis of Zigbee* [online]. In: . Cambridge (MA): Massachusetts Institute of Technology, 2017 [cit. 2020-05-08]. Dostupné z: https://courses.csail.mit.edu/6.857/2017/project/17.pdf

[26] HAARTSEN, J.C. The Bluetooth radio system. *IEEE Personal Communications* [online]. 2000, **7** (1), 28-36 [cit. 2020-05-08]. DOI: 10.1109/98.824570. ISSN 1070-9916. Dostupné z: http://ieeexplore.ieee.org/document/824570/

[27] GSM - Security and Encryption. In: *Tutorialspoint* [online]. Telangana: Tutorials Point, c2020 [cit. 2020-05-08]. Dostupné z: https://www.tutorialspoint.com/gsm/gsm_security.htm

[28] VOJÁČEK, Antonín. IoT MQTT prakticky v automatizaci - 1.díl - úvod. *Automatizace.hw.cz* [online]. 2017-01-21 [cit. 2020-05-10]. Dostupné z: https://automatizace.hw.cz/iot-mqtt-prakticky-v-automatizaci-1dil-uvod.html

[29] MQTT - Komunikační standard pro IoT. In: *ROOT.cz* [online]. 2016 [cit. 2020-05-10]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/

[30] Mosquitto.conf man page. In: *Eclipse Mosquitto* [online]. [cit. 2019-05-23]. Dostupné z: https://mosquitto.org/man/mosquitto-conf-5.html

[31] HiveMQ TLS Benchmark. In: *HiveMQ* [online]. [cit. 2020-05-10]. Dostupné z: https://www.hivemq.com/img/blog/tls_benchmark_screenshot-1024x333.png

[32] RASPBERRY Pi 3 Model B+. In: *Alza.cz* [online]. Praha: Alza.cz, 2020 [cit. 2020-05-13]. Dostupné z: https://www.alza.cz/raspberry-pi-3-model-b-d5284636.htm

[33] WeMos D1 Mini ESP8266 WiFi modul. In: *LASKARDUINO.cz — by Makers for Makers* [online]. Rychnov nad Kněžnou: laskaarduino.cz, 2020 [cit. 2020-05-13]. Dostupné z: https://www.laskarduino.cz/wemos-d1-mini-esp8266-wifi-modul/

[34] ASAIR senzor teploty a vlhkosti vzduchu DHT11, modul. In: *LASKARDUINO.cz — by Makers for Makers* [online]. Rychnov nad Kněžnou: laskaarduino.cz, 2020 [cit. 2020-05-13]. Dostupné z: https://www.laskarduino.cz/arduino-senzor-teploty-a-vlhkosti-vzduchu-dht11–modul/

[35] ČÍŽEK, Jakub. Odposlouchávali jsme děravou domácnost: Katka z Brightonu právě sedí doma, zatímco její manžel hraje asi golf. In: *Živě.cz: O počítačích, internetu, vědě a technice* [online]. Praha: CZECH NEWS CENTER, 2018, 19. srpna [cit. 2020-05-18]. Dostupné z: https://www.zive.cz/clanky/odposlouchavali-jsme-deravou-domacnost-katka-z-brightonu-prave-sedi-doma-zatimco-jeji-manzel-hraje-asi-golf/sc-3-a-194628/default.aspx

[36] Samsung MicroSDXC 128GB EVO Plus UHS-I U3 + SD adaptér. In: *Alza.cz* [online]. Praha: Alza.cz, 2020 [cit. 2020-05-13]. Dostupné z: https://www.alza.cz/samsung-micro-sdxc-128gb-evo-plus-sd-adapter-d4846086.htm

[37] 60in LG 60UM7100PLB. In: *Alza.cz* [online]. Praha: Alza.cz, 2020 [cit. 2020-05-13]. Dostupné z: https://www.alza.cz/60-lg-60um7100plb-d5625317.htm

[38] Cena elektřiny za kWh opět zdražila. V roce 2020 stojí 4,76 Kč. In: *Elektřina.cz - vše co potřebujete vědět v oblasti energetiky a technologií* [online]. Praha: Ušetřeno.cz, 9. března 2020 [cit. 2020-05-13]. Dostupné z: https://www.elektrina.cz/cena-elektriny-za-kwh-2020-cez-eon-pre-bohemia-centropol-a-dalsi