ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2015-16**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **František Flachs**

Studijní program: **Otevřená informatika**
Obor: **Počítačové systémy**

Název tématu česky: **Návrh a implementace aplikačního protokolu pro přenos senzorových dat**

Název tématu anglicky: **Design and Implementation of an Application Protocol for Sensor-based Data**

Pokyny pro vypracování:

Prostudujte možná existující řešení přenosu aplikačních dat v sítích s velmi nízkou propustností bez podpory protokolů rodiny TCP/IP. Analyzujte omezení a požadavky plynoucí z vlastností specifických pro senzorová data v tzv. konceptu Internetu věcí (IoT). Navrhněte jednoduchý aplikační protokol, případně upravte již existující, který bude optimalizován pro omezenou velikost zpráv používanou u konkrétních přenosových technologií v IoT. Navržený protokol implementujte na vybrané přenosové technologii a ověřte jeho vlastnosti v demonstračním nasazení.

Seznam odborné literatury:

[1] Bormann, Carsten, Angelo P. Castellani, and Zach Shelby. "Coap: An application protocol for billions of tiny internet nodes." IEEE Internet Computing 2 (2012): 62-67.

[2] Mainetti, Luca, Luigi Patrono, and Antonio Vilei. "Evolution of wireless sensor networks towards the internet of things: A survey." Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on. IEEE, 2011.

[3] Lerche, Christoph, Klaus Hartke, and Matthias Kovatsch. "Industry adoption of the internet of hings: a constrained application protocol survey." Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on. IEEE, 2012.

[4] Sheng, Zhengguo, et al. "A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities." Wireless Communications, IEEE 20.6 (2013): 91-98.

Vedoucí bakalářské práce: Ing. Tomáš Hégr (K13132)

Datum zadání bakalářské práce: 27. ledna 2016

Platnost zadání do[1]: 30. září 2017

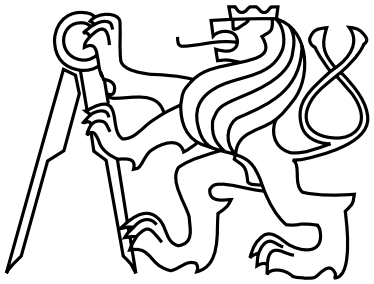
Doc. Ing. Jan Holub, Ph.D.
vedoucí katedry

Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 27. 1. 2016

[1] Platnost zadání je omezena na dobu tří následujících semestrů.

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra měření



Bakalářská práce

Návrh a implementace aplikačního protokolu pro přenos senzorových dat

František Flachs

Vedoucí práce: Ing. Tomáš Hégr

Studijní program: Otevřená informatika, Bakalářský

Obor: Počítačové systémy

25. 5. 2017

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Tomášovi Hégrovi za inspirativní a technické rady a jeho ochotu, kterou mi věnoval. Rád bych také chtěl poděkovat svým blízkým za pomoc a trpělivost při mém studiu.

Prohlášení

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

V Praze dne 25. 5. 2017 .....

Abstract

This thesis offers survey of using sensors, data models and way how to transmit data in IoT (Internet of Things). First main point is comparison of the data models which are comparing with designed scenarion. Second main point is design of own data model and protocol on application layer.

There is complete pattern for testing data models after severity of transmitions data witch each data model in IoT. Witch this scenario is tested many data models including the designed one.

Keywords: Internet of Things, data serialization, data model, senzors, testing

Abstrakt

Tato práce nabízí přehled používaných senzorů, datových modelů a způsobů přenosu dat v IoT. Hlavními body této práce jsou porovnání datových modelů na základě navržených testovacích scénářů a návrh vlastního aplikačního protokolu včetně datového modelu na aplikační vrstvě.

Práce nabízí možnost kompletního vzoru pro testování datových modelů podle náročnosti datových přenosů v IoT. Tímto scénářem je testováno několik datových modelů včetně nově navrženého v této práci.

Klíčová slova: Internet věcí, serializace dat, datový model, senzory, testování

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V posledních pár letech došlo k velkému rozšíření chytrých zařízení z oblastí nositelné elektroniky, zdravotnictví, průmyslu, logistiky, dopravy a z dalších odvětví. Důsledkem tohoto masivního rozšíření vznikl nový pojem Internet of Things, který má české označení internet věcí. Součástí IoT jsou senzorické sítě, které nám umožňují sběr aktuálních stavů senzorů na zařízeních připojených do sítě. Tato zařízení jsou označována jako koncová. Koncové zařízení chápeme jako modul obsahující kokrokontrolér a prostředky pro bezdrátové připojení do sítě.

Téma internetu věcí je v dnešní době velmi aktuální a do budoucna se očekává velký rozvoj připojení fyzických zařízení do internetu. Implementace velkého množství zařízení do sítě bude mít velký dopad na různé životní, pracovní a ekonomické procesy, které nelze přehlížet.

Cílem této práce je otestovat způsob serializace dat a jejich následné zaslání pomocí bezdrátové technologie pro IoT. Serializace je způsob uložení dat takovým způsobem, aby jejich přenos byl co nejefektivnější. Existuje celá řada serializačních nástrojů. Pro testování budou vybrány ty z nich, které převádí data do binární podoby a do podoby čitelné člověkem.

U těchto nástrojů bude testována doba serializace a deserializace, velikost používané paměti RAM, velikost paměti pro uložení serializovaných dat a dobu přenosu z jednoho zařízení na jiné včetně serialializace před samotným přenosem a deserializace bezprostředně po přenosu dat.

Z mnou známých technologií a dostupných informací nabývám domněnky, že se zvětšující se velikostí dat bude narůstat doba serializace/deserializace, velikost využité paměti a čas pro přenos informace z jednoho zařízení na druhé. U binárních serializačních nástrojů předpokládám pomalejší nárůst využité kapacity procesoru a paměti při zvětšujícím se počtu serializovaných dat.

Práce je strukturována do několika kapitol, kde každá z nich popisuje určité logické celky, kterými jsou: rešerše, analýza, testování včetně implementace a zhodnocení testů. Rešerše obsahuje rozdělení senzorů do oblastí využitelných pro internet věcí a výčet senzorů pro tyto oblasti. Následuje přehled IoT modelů, komunikačních standardů a přenosových technologií. V této kapitole je obsažen úvod do bezpečnosti v IoT. V kapitole analýzaje popsán přehled nástrojů pro popis datových modelů. Kapitola testování obsahuje návrh testovacích scénářů

a implementaci testovaného datového modelu. V závěru této práce je v kapitole vyhodnocení testů popsáno a graficky znázorněno, které nástroje jsou vhodné pro serializaci dat v internetu věcí.

Kapitola 2

Rešerše

2.1 Koncept internetu věcí

Internet je spojen se vznikem počítačů a počítačových sítí, což je datováno po roce 1945. Jedná se o komplexní propojení zařízení a počítačů do logických celků tak, aby spolu mohla komunikovat. S postupným rozvojem elektronických zařízení došlo ke snižování výrobní ceny a současně k miniaturizaci. S tímto novým proudem elektronických zařízení došlo k stále rostoucí integraci chytrých zařízení.

Do existujících zařízení byla přidána elektronika pro řízení funkčnosti zařízení. Nová vyráběná zařízení již mnohdy tuto elektroniku obsahují. Tato elektronika je použita pro sběr informací ze senzorů a jejich následné předání do internetu popř. do domácí sítě.

S postupným rozšiřováním této chytré sítě se objevují i některé problémy, které je potřeba řešit. K těmto problémům patří např. identifikace zařízení, diagnostika a správa celé sítě nebo automní správa.

Příchod internetu věcí můžeme chápat jako spojení fyzických věcí a okolního digitální světa. Cílem tohoto propojení je integrace stavů předmětů, které mohou být prospěšné pro různé aplikace a nasazení. Propojení objektů do světa internetu představuje výzvy v zabezpečení chytrých sítí.

Internet vyžaduje určitá pravidla, kterými se budou řídit designéři při návrhu nových zařízení. Je nemyslitelné, aby zařízení, jako například pračka prádla, nebylo možné otevřít v případě výpadku proudu nebo výpadku spojení do sítě. Zařízení, která vznikla již před érou IoT, by měla zachovávat svou funkčnost a jejich připojení k síti by nemělo ovlivňovat základní účel zařízení.

Celá myšlenka IoT by měla být v budoucnu završena pojmem "smart world", kde budou věci každodenní potřeby připojeny do sítě internetu a ekonomika a další podpůrné systémy budou pracovat jednodušeji a efektivněji.

2.2 Dělení senzorů

Ve světě IoT se objevuje celá řada senzorů, které lze integrovat do "hloupých"elektrospotřebičů, a tak z nich udělat zařízení, která je možné připojit do internetu a vzdáleně je ovládat nebo z

nich vyčítat informace. Senzor je snímač/čidlo/převodník/detektor, který se stará o snímání fyzikální, chemické či biologické veličiny a převádí ji na napěťový, proudový, číslicový nebo jiný signál.

Některé elektrospotřebiče vyžadují nízkou spotřebu elektrické energie. Pro takovéto typy se využívá architektury procesorů ARM. Tyto procesory mají spotřebu elektrické energie tak nízkou, že je lze použít na místa, kde není možné použít velkokapacitní baterii nebo ji často nabíjet.

Senzory je možné dělit do několika kategorií a skupin. Nejčastější dělení senzorů je podle snímané veličiny, z čehož vyplývá dělení do těchto základních skupin: [22]

- mechanické
- tepelné
- elektrické
- magnetické
- intenzita vyzařování
- chemické
- biologické

Senzory dále dělíme podle charakteru výstupní veličiny a to na analogové a digitální. Výstupní veličinou může být:

- elektrický signál
- optický signál (změna barvy, intenzity)
- mechanický pohyb (posun předmětu)

Další dělení senzorů definujeme podle styku s měřeným prostředím na dotykové a bezdotykové. Dělení senzorů je mnoho a s přibývajícími aplikacemi jsou další a další senzory vyvíjeny pro konkrétní aplikace a účely.

Pro výběr konkrétního senzoru k dané aplikaci je potřeba znát několik konkrétních parametrů, které ovlivní samotný výběr. Tyto parametry mohou být závislé na daném prostředí, typických podmínkách pro měření nebo dalšími vlivy jako je například lidský faktor. Senzory vybíráme podle následujících vlastností (uvádím základní z nich): [21]

- měřená přesnost
- rozsah měření
- rozlišovací schopnost
- citlivost

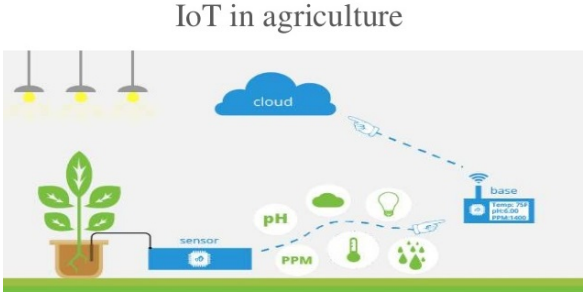
- šum
- výstupní impedance
- zkreslení
- podmínky prostředí (teplota, vlhkost)
- způsob kalibrace
- cena

Na senzory je možné nahlížet z pohledu využití pro průmysl nebo v domácím prostředí. Na senzory v průmyslu se klade mnohem větší důraz na přesnost, stabilitu a spolehlivost. Na senzory v domácím prostředí nejsou kladeny takto náročné požadavky, a to se odráží v jejich pořizovací ceně. Senzory je možné klasifikovat podle potřeby elektrické energie pro svou funkčnost, a to na aktivní nebo pasivní.

2.3 Senzory používané v IoT

Svět IoT zahrnuje řadu senzorů, které lze v konkrétních případech velmi efektivně využít. Existuje několik skupin průmyslových odvětví, kde využití senzorů dovoluje zvýšit efektivitu práce nebo výroby na násobky díky propojení daného odvětní s IoT. Přehled používaných senzorů je v příloze práce.

2.3.1 Zemědelství



Obrázek 2.1: IoT v zemědělství[27]

V zemědělství pomocí připojení senzorů do sítě sledujeme různé jevy, které obvykle vyžadují velké množství času, které vynakládá člověk. Díky úspoře jak časové, tak finanční, lze zemědělství udělat efektivnější z pohledu ušetřených financí. V tomto průmyslu můžeme najít senzory pro GPS (Global Positioning System) navigaci a další pomocné prvky k mapování povrchu jako je například GIS (Geographic Information System) nebo družicový průzkum země pro mapování půdy a pozemků. Najdeme zde několik oblastí, kde lze IoT uplatnit, hlavní z nich jsou tyto:

- hlídání objektů a ploch
- hlídání a sledování zdravotního stavu skoru
- měření vlhkosti a pH (Potential of Hydrogen) půdy
- lokální meteostanice
- sledování průtoků a znečištění vodních toků
- monitoring zemědělských strojů

Dalším důležitým bodem pro zemědělství jsou drony, které se budou starat o hlídání objektů ze vzduchu, pravidelné fotografické snímky z oblastí nebo rozhazování hnojiv na potřebná místa.

2.3.2 Smart City

Obrázek 2.2: IoT ve městě [20]

Proniknutí internetu věcí do měst přinese mnoho zefektivnění. Pomůže nám k většímu komfortu, životní úrovni, vyššímu bezpečí nebo k úspoře financí. Najdeme několik oblastí, ve kterých se internet věcí velmi zasazuje o zlepšení a zvýšení komfortu použití.

Infrastruktura

- řízení pouličního osvětlení
- zabezpečení objektů
- energetický management
- kontrola vibrací budovy a konstrukcí

Doprava

- inteligentní řízení dopravy
- sdílení dopravních prostředků
- řízení volných parkovacích míst

- použití pro MHD (vytíženost vozidel, přesnější jízdní řády)

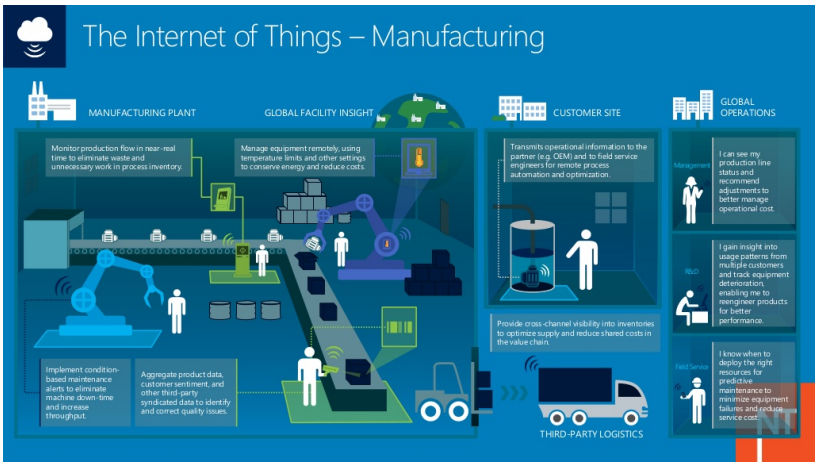
Životní prostředí

- sledování znečištění ovzduší ve městě
- monitoring toku řek
- kontrola městské zeleně
- optimalizace využití energií

2.3.3 Energetika

V energetice se jeví jako prospěšné umístit čidla do rozvodných stanic, na stožáry vysokého napětí a na důležití body, které slouží k přenosu elektrické energie. Tato čidla umožní komplexní informovanosi o závadách a o jejich charakteru, což pomůže v budoucnu předcházet těmto poškozením. Jeví se zde i vhodné použít čidla pro detailní regulování toku energií. Tato čidla budou sbírat informace z měřených veličin a výpočetní jednotka informace následně vyhodnotí.

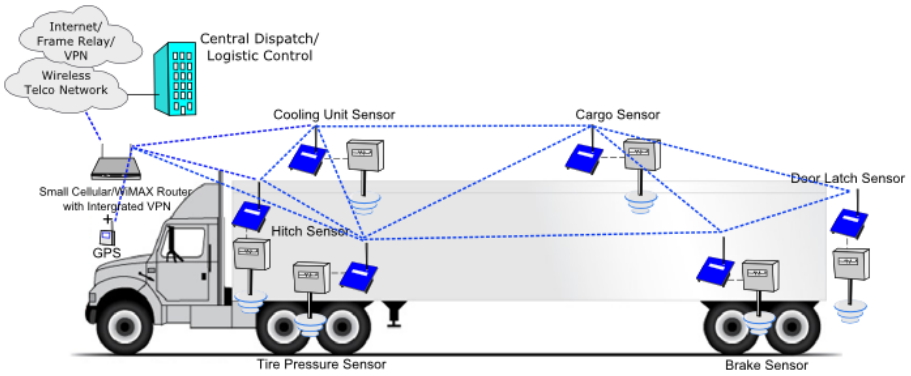
2.3.4 Průmysl



Obrázek 2.3: IoT v průmyslových strojích[37]

Průmysl využije internet věcí k efektivnějším inovacích výrobních linek nebo průmyslových strojů. Díky snímání různých veličiny se získají detailní informace o přehledu výroby a bude zde možnost detailnějšího plánování výrobních postupů a procesů. Čidla pomohou ke stanovení intervalů údržby nebo ke včasnému odhalení závady, což bude mít za následek minimalizaci výrobních prostojů.

2.3.5 Doprava a logistika



Obrázek 2.4: IoT ve městě[26]

Pro IoT je doprava a logistika velkou výzvou. Některé pokusy o integraci IoT můžeme pozorovat již dnes. Jedná se autonomní vozidla, navigační systémy s přístupem do databáze dopravních omezeních, sledování zásilek, navigační systémey apod. V budoucnosti očekáváme další průnik IoT do dopavy a logistiky a to v těchto bodech:

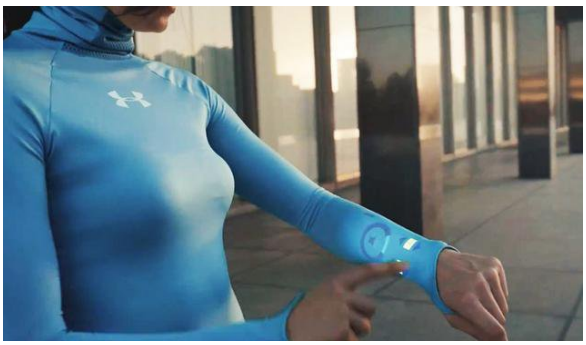
- plynulost dopravy
- snížování počtu nehod
- real-rime zásilky na mapovém podkladu
- senzory pro zabránění řízení pod vlivem alkoholu nebo proti mikrospánku
- při něhodě dojde k automatickému přivolání pomoci a ke zjištění lokace a rozsahu škod
- rozšířená virtuální realita ve smartglasses

2.3.6 Zdravotnictví

IoT proniká do oboru zdravotnictví a zaměřuje se na kontrolu zdravotního stavu člověka. Je zde celá řada oblastí, které lze monitorovat, jsou to například krevní tlak, hladina cukru v krvi, srdeční tep, hmotnost nebo pravidelnost a kvalita spánku. Tyto oblasti jsou zatím jen začátkem.

Lékaři budou mít díky přítomnosti IoT v lékařství možnost provádět diagnózu mnohem více komplexnější. Zvolená léčba bude cílenější a bude možné sledovat průběh indispozice. Pojišťovny zajisté také využijí tyto možnosti, zejména u výpočtu pojistného. V tuto chvíli přecházíme do oblasti osobních údajů a dat, kdy se ne každému bude líbit, že je sledován každý jeho krok. V tuto chvíli si bude muset každý vybrat mezi efektivnější léčbou a sledováním informací o jeho stavu a pohybu osoby.

2.3.7 Nositelná elektronika



Obrázek 2.5: Nositelná elektronika [41]

Nositelná elektronika je část elektronických zařízení, která jsou přímo uzpůsobena pro nošení na lidském těle. Tato elektronika je jak svou velikostí, hmotností, tak i použitým materiálem vhodná ke každodennímu nošení. Získané údaje mohou být použity k efektivnímu sportování nebo odhalí počínající nemoci popř. rovnou zavolají lékařskou pomoc. Nositelná elektronika může interagovat s okolím a reagovat na aktuální rozpoložení člověka.

Pokud přejdeme elektroniku, kterou standardně používáme dnes (mobilní telefon, mp3/mp4 přehrávač, primitivní krokoměr), přichází do této oblasti mnohem oblíbenější a zajímavejší zažízení mezi které patří:

- chytré náramky (hodinky, fitness)
- brýle
- senzory určující zdravotní stav člověka
- outdoor kamery a videotechnika
- GPS systémy
- oblečení

2.3.8 Elektrospotřebiče v datové síti

Obrázek 2.6: IoT v elektrospotřebičích [39]

IoT přichází v dalším stupni s připojením elektrospotřebičů do datové sítě. Pro některé starší generace je nepochopitelné připojení všech elektrospotřebičů do internetu. Připojení TV do sítě je dnes takřka standardem, ale připojení pračky se jeví jako nesmysl. Myšlenkou připojení všech elektrospotřebičů je vzájemná interakce mezi spotřebiči, vzdálené ovládání, komunikace s cloudem a interakce s člověkem.

Vize elektrospotřebičů v síti jsou v těchto oblastech:

- lednice si sama objednává potraviny nebo rozpozná kazící se jídlo
- mikrovlná trouba rozpozná druh a váhu jídla a poté sama nastaví počet minut ohřevu
- trouba navrhne jaké jídlo upéci podle surovin v domácnosti
- vytvoření světelné atmosféry podle okolních podmínek či pořadu v TV

2.3.9 Inteligentní budovy

Obrázek 2.7: IoT v budovách [40]

Výrobci se předhání, kdo z nich udělá inteligentnější elektroinstalaci tak, aby uživatel mohl ovládat ideálně všechny senzory a zařízení a ideálně odkud koliv v bytě nebo mimo něj. Všechny senzory a snímače jsou pomocí proprietárního protokolu propojeny mezi sebou a s řídící jednotkou (bránou), pomocí které jsou data odesílána do internetu - privátní cloud. Základní myšlenka IoT zde není dodržena, protože každý prvek nekomunikuje přímo do internetu, ale přes jiné zařízení. Každopádně tak nebo tak nám inteligentní domácnosti přinášejí mnoho výhod.

- kamerový systém a zabezpečení objektu
- topení, klimatizace

- vyhřívání a čištění bazénu
- ovládání osvětlení
- zavlažování zahrady
- multimediální zařízení
- úspora energií

2.4 IoT modely

Komunikaci mezi prvky v síti IoT rozdělujeme do čtyř základních kategorií, které jsou charakteristické typem přenosu dat a daným použitím specifikovaným každou aplikací. [19]

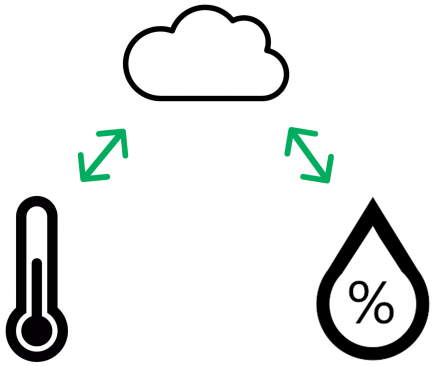
2.4.1 Device-to-device

Obrázek 2.8: Device-to-device

Tento model představuje komunikaci mezi dvěma zařízeními, které jsou v poměrně blízké vzdálenosti od sebe. Přenáší se malý obsah dat při nízkých přenosových rychlostech. Dvě zařízení mohou spolu komunikovat přes mnoho protokolů včetně IP. Většina zařízení používá protokoly jakou jsou Bluetooth, Z-Wave nebo ZigBee.

Model device-to-device se používá v domácnostech, kde není zapotřebí přenášet velké množství dat. Tento model je vhodný např. pro nositelnou elektroniku nebo osobní zařízení určeného k měření osobních životních funkcí. Jelikož není nutné sdílet data mezi více zařízeními, stačí propojení bod-bod. Tento model je založený na nízké spotřebě elektrické energie, malé velikosti a nízké ceně. Použití je např. systém elektrického vypínače a žárovky, kde se spojí dvě zařízení mezi sebou.

2.4.2 Device-to-cloud

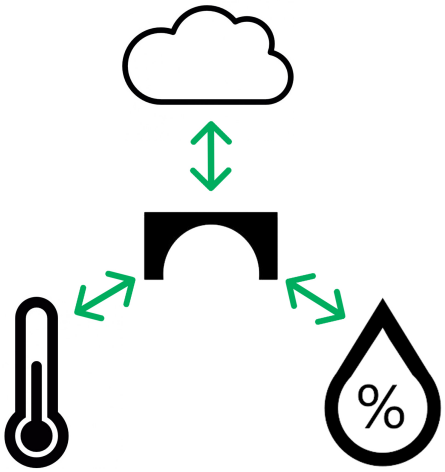


Obrázek 2.9: Device-to-cloud

Model device-to-cloud slouží k přístupu a zasílání dat ze zařízení do cloudu na aplikaci poskytovatele služby nebo k výměně dat mezi dvěma rozdílnými systémy. Přístup z cloudu nabízí možnost vzdáleného ovládání zařízení z cloudu a nahrávání nových firmwarů.

Z bezpečnostního pohledu se jedná o mnohem komplikovanější model než je device-to-device z důvodu nutnosti připojení koncového zařízení do místní sítě. Pokud se jedná o wifi je nutné znát klíč a přístupové údaje do cloudového prostoru. Příklad použití je webová kamera sdílející obraz do cloudového prostoru. Tohoto typu komunikace je využito např. přenosové technologii LoRa.

2.4.3 Device-to-gateway

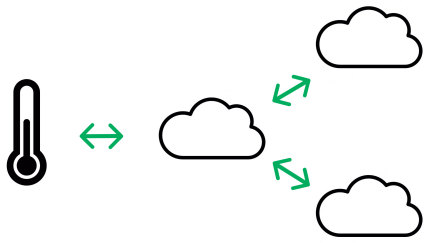


Obrázek 2.10: Device-to-gateway

V tomto modelu se všechna koncová zařízení připojují k místní gateway, která je spojená s cloudovým prostorem nebo aplikací. Na *gateway* je nejčastěji umístěn software, který dovoluje zprostředkování dat mezi koncovým zařízením a cloudem.

Brána poskytuje bezpečný přenos dat nebo protokolový překlad. Současně je možné propojit několik zařízení, která jsou připojená pomocí různých protokolů. Příkladem použití může být sportovní náramek, který komunikuje se serverem skrze mobilní telefon, který je připojen přes operátora nebo wifi do internetu.

2.4.4 Back-end data-sharing



Obrázek 2.11: Backend-data-sharing

Tento model umožňuje uživateli porovnávat data získaná ze senzorů uložená na několika serverech. Jedná se o rozšíření modelu davice-to-cloud. Model back-end data-sharing dovoluje získávání dat z jednotlivých zařízení agregovat a anlyzovat.

2.5 Komunikační standardy

Protokol je standard, který definuje způsob výměny dat v sítích jako jsou LAN, Internet, Intranet a další. Každý protokol má definováno jak jsou data formátována před odesláním, jaká je komprese dat a jakým způsobem se kontrolují a opravují vzniklé chyby. Jeden z nejznámějších protokolů pro výměnu dat v Internetu je HTTP (HyperText Transfer Protocol). [12]

2.5.1 HTTP

HTTP (Hypertext Transfer Protocol) je protokol aplikační vrstvy, který je používaný ve WWW (World Wide Web) od roku 1990. Dovoluje uživatelům a zařízením výměnu informací na webových stránkách. Při prohlížení webové stránky prohlížečem, je na začátku URL adresy schéma, což je použitý protokol pro předávání dat mezi zařízením (prohlížeč, aplikace) a serverem. Schéma je charakteristické použitím písmen, číslic a znaků. Zápis je závislý na velikosti písmen, následuje dvojtečka a dvě lomítka. Tento zápis není vyžadován u všech druhů schémat.

V současné době většina webových prohlížečů má defaultně přednastaveno použití protokolu HTTP. Tento protokol má vyhrazený port 80. Pro použití šifrované metody HTTPS (Hypertext Transfer Protocol Secure) je k dispozici port 443. Tato podoba protokolu zabraňuje odposlouchávání. Používá se např. pro finanční transakce nebo pro přenos citlivých údajů na webu. [11]

Protokol HTTP je definovaný v několika verzích, které jsou zaneseny jako standardy RFC. [36]

- HTTP/0.9 - první verze HTTP protokolu, představena v letech 1990
- HTTP/1.0 - RFC 1945, představen v roce 1996
- HTTP/1.1 - RFC 2616, uvolněn pro použití v roce 1997
- HTTP/2 - RFC 7540 (RFC 7541 specifikuje formát komprimace hlaviček)

Nejnovější verzí protokolu HTTP je verze 2. Jedná se o verzi, která byla standardizována v roce 2015 jako RFC 7540. Protokol zrychluje vzájemnou komunikaci mezi klientem a serverem a minimalizuje zátěž obou stran. Přidává koncepci, kdy klient naváže pouze jedno TCP spojení, přes které spravuje všechny datové proudy paralelně. Založení a ukončení datového proudu HTTP/2 nezahrnuje žádnou režii. Jednotlivé proudy jsou nezávislé a TCP spojení se mohou míchat. Tento přenos je ideální pro větší množství souborů [36]. Tuto verzi v únoru 2017 již podporovalo 12 % webových stránek.

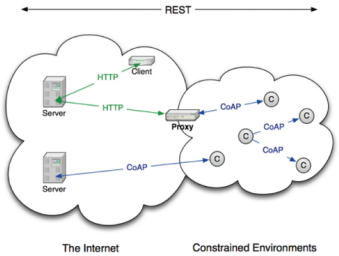
Pro zjištění stavu komunikace pomocí HTTP slouží stavové kódy [11]. Jejich přehled je v následující tabulce:

Tabulka 2.1: HTTP status codes

Code	**Definice**	**Code**	**Definice**
201	Created	406	Not Acceptable
202	Accepted	407	Proxy Authentication
203	Non-Authoritative Information	408	Request Time-out
204	No Content	409	Conflict
205	Reset Content	410	Gone
206	Partial Content	411	Length Required
300	Multiple Choices	412	Precondition Failed
301	Moved Permanently	413	Request Entity Too Large
302	Found	414	Request-URI Too Large
303	See Other	415	Unsupported Media Type
304	Not Modified	416	Requested range not satisfiable
305	Use Proxy	417	Expectation Failed
307	Temporary Redirect	500	Internal Server Error
400	Bad Request	501	Not Implemented
401	Unauthorized	502	Bad Gateway
402	Payment Required	503	Service Unavailable
403	Forbidden	504	Gateway Time-out
404	Not Found	505	HTTP Version not supported

2.5.2 CoAP

CoAP (Constrained Application Protocol) je protokol navržen na aplikační vrstvě modelu ISO/OSI. Je navržen pro využití zařízeními s menší pamětí a malou elektrickou spotřebou. Tento protokol je podporován většinou zařízení, které mohou používat UDP (je možné implementovat i potvrzovanou podobu TCP s ACK). CaAP lze používat pro IoT sítě nebo pro topologii typu device-to-device.

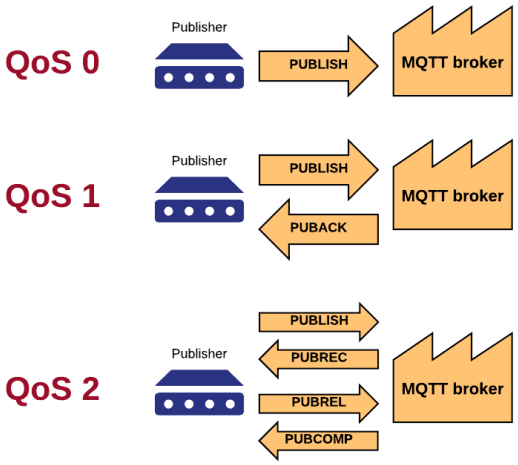


Obrázek 2.12: Topologie sítě s využitím protokolu CoAP [28]

Je navržen jako efektivní protokol používající REST, což je rozhraní pro distribuované prostředí orientované na data, ideální protokol pro sítě tvořící omezená zařízení na výkon a spotřebu energie. Je zde implementována jednoduchá konverze s protokolem HTTP. CoAP je asynchronní model pro přenos dat. Zabezpečení je řešeno pomocí PSK, RPK nebo certifikátu.

Ve srovnání s HTTP, které využívá TCP spojení, CoAP používá UDP. CoAP nebyl navržen aby nahradil HTTP, každý z protokolů má své charakteristické využití. CoAP je primárně určen pro sítě s méně výkonými zařízeními, proto také jeho hlavička je menší. HTTP předpokládá použití zařízení, která nejsou kapacitně a výkonostně omezená a spotřeba elektrické energie nehraje roli.

2.5.3 MQTT



Obrázek 2.13: MQTT QoS [30]

MQTT (Message Queuing Telemetry Transport) je protokol založený na jednoduchosti a nenáročnosti předávání zpráv mezi klientem a centrálním bodem (broker). Díky jeho nenáročnosti je možná implementace i pro jednodušší procesory.

Komunikace v tomto protokolu probíhá pomocí TCP a používá návrhový vzor *publisher-subscriper*. Zprávy jsou děleny to témat (topic). Pro každé téma může mít zařízení v módu

publish, tzn. že posílá informace brokeru, který se dále stará o distribuci dalším přihlášeným zařízením k odběru. Zařízení může být i přihlášeno k odběru zpráv (subscribe) a potom mu broker zasílá informace z daného tématu.

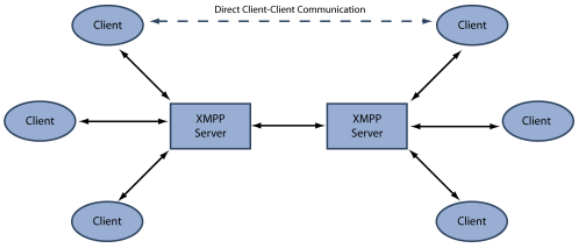
V protokolu MQTT existují tři úrovně QoS (Quality of Service). V nulté úrovni pošle *publisher* zprávu PUBLISH *brokeru* a stejně tak *broker* pošle zprávu všem *subscriberům* daného tématu. Žádné zařízení se o další komunikaci více nestará.

Na první úrovni *publisher* pošle zprávu PUBLISH *brokeru* a čeká. *Broker* rozesílá zprávu všem *subscriberům* a jakmile dostane alespoň od jednoho (nebo ode všech - závisí na konkrétní implementaci), odesílá *broker* zprávu PUBACK *publisheru* o potvrzení doručení k *subscriberům*.

Na druhé úrovni posílá *publisher* zprávu PUBLISH *brokeru*, ten potvrdí přijetí zprávou PUBREC a přeposílá zprávu *subscriberům*. *Publisher* odpovídá *borkeru* zprávou PUBREL a *broker* posílá zpět zprávu PUBCOMP. [30]

MQTT-SN (MQTT fot Sensor Networks) je zjednodušená verze MQTT, která nahrazuje v síti koncových čidel TCP. Tento protokol optimalizuje komunikaci tak, aby bylo přenášeno co nejmenší množství dat např. místo názvů témat se přenáší jejich číselné označení. Další výhodou je automatizace v nalezení *brokera* v síti. Tato zjednodušená verze také přináší úsporu energie pro zařízení, která jsou schopna se uspat.

2.5.4 XMPP



Obrázek 2.14: XMPP topologie [29]

XMPP (Extensible Messaging and Presence Protocol) je protokol, který zasílá zprávy a zjišťuje stav zařízení. Protokol vznikl pro IM (Instant Messaging) síť Jabber. Později se ukázalo, že je implementovatelný i pro vzájemnou komunikaci dvou zařízení pro automatizované služby. XMPP je implementací jazyka XML.[35]

Tato síť je založena na architektuře klient-server. Síť je decentralizovaná, což znamená, že není jeden dedikovaný server.

Pomocí tohoto protokolu je možné posílat data pro real-time komunikaci. Obsahuje set standardů pro komunikaci mezi systémy. Indikátor prezence dovoluje zaslat informaci o stavu přítomnosti uživatel k příjímání/odesílání zpráv. Protokol byl navržel jako otevřený a rozšiřitelný. [16]

2.6 Prostředky pro popis datových modelů

Při vybírání správného prostředku pro popis datového modulu je velmi důležité rozhodnout se pro takový nástroj, který nám bude vyhovovat formou výpisu. Tyto nástroje dělíme podle toho, v jaké podobě je serializovaný objekt. Existují nástroje, které produkují serializované objekty do binární podoby a do podoby *human-readable*, neboli čitelné člověkem. Nástore produkující formát čitelný člověkem jsou jednodušeji implementovatelné, protože jim je snáz rozumět a při ladění chyb je možné konkrétně označit chybnou část kódu. Binární nástroje jsou rychlejší a jednodušší pro použití zařízením, protože není potřeba mezi kroku mezi převodem do podoby, která je určena pro výpočetní jednotku. Pro binární formáty je zapotřebí nějakého editoru, pomocí kterého je možné serializované objekty zobrazovat popř. upravovat při ladění programu.

Serializace je vytvoření proudu symbolů vhodných pro přenos. Z těchto symbolů je možné zpětně sestavit původní objekty ve stavech, v jakých byly serializovány. Jedním z důvodů serializace je uložení dat do trvalého úložitě, když nepotřebujeme k uložení všechna data včetně velkého množství prázdných míst.

Takto se nepotřebná data pro uložení odstraní a zpětná rekonstrukce již prázdné bloky dat k sestavení do původní podoby nepotřebuje. Serializace se obdobně používá i pro přenos dat po síti, kdy je nezbytné ušetřit elektrickou energii. Čím déle se vysílají data, tím více je spotřebovávána elektrická energie. Na druhé straně proběhne deserializace, která rekonstruuje data do původní podoby. Čím kratší dobu deserializace probíhá, tím méně je spotřebováno elektrické energie.

2.7 Přenosové technologie

V této kopitole jsou popsány nejvyužívanější přenosové technologie, které jsou navrženy pro používání v LPWAN sítích. Všechny tyto technologie umožňují přenos senzorických dat na větší vzdálenosti a zařízení přenášející data jsou schopna fungovat na baterii několik let bez vnějšího zásahu. Tyto přenosové technologie využívaní volných pásem pro přenos dat. Tato pásma jsou odlišná mezi kontinenty:

- Asie - 433 MHz
- Evropa - 868 MHz
- Amerika - 915 MHz

Využívání těchto pásem je řízeno vládou každé země, kde se definuje maximální počet zpráv vyslaných z/do zařízení, jaký je maximální vysílací výkon a jak velká zpráva může být. Více informací je možné dohledat v zákoně č. 500/2004 Sb. Více informací o rádiových přenosech je k dispozici na stránkách ČTU (Český telekomunikační úřad) ve všeobecných oprávněních [3].

2.7.1 6LoWPAN

6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) je protokol pro přenos dat bezdrátovým způsobem. Topologie sítě pro tento protokol je typu mesh. 6LoWPAN je „low-power“, což je uzpůsobení funkčnosti pro zařízení, která potřebují málo energie pro svůj chod. Každý uzel v této síti má svou vlastní IPv6 adresu, což mu umožňuje komunikovat přímo do internetu pomocí otevřených standardů. 6LoWPAN může implementovat i routovací protokol, který zajišťuje směrování v síti. Bohužel tento protokol neimplementuje funkcionalitu „sleeping“ módu, což je nezbytné pro nízkonákladová zařízení v síti IoT.

Jedna z nevýhod tohoto protokolu je obtížnější implementace do zařízení z důvodu využívaní IPv6. Výhodou je možnost adresovat každý uzel v topologii, kde žádná brána není nezbytně nutná. Tato síť je velmi rubustní a rozšiřitelná. Tuto technologii je možné používat na frekvencích 868, 915 a 2400 MHz. Pro využívání těchto frekvencí není potřeba licenceo

V následujících kapitolách jsou popsány nejrozšířejší přenosové technologie pro odečet senzorických dat.

2.7.2 LoRa

Technologie LoRa zahrnuje jak LoRa modulace, tak komunikační protokol LoRaWAN . Samotná technologie je navržena pro frekvence 433, 868 a 915 MHz. LoRa dokáže přenáše data až na vzdálenost 40 km ve volném prostoru a na vzdálenost až 3 km v zastavěné ploše.

Technologie LoRa popisuje proprietární návrhy modulací, které mají rozšířené spektrum patentovaní společností Semtech. Komunikační protokol LoRaWAN je optimalizovaný pro sítě LPWAN a současně je uzpůsoben pro zařízení napájená baterií, která mají vydržet několik let. Tento protokol vznikl díky neziskové organizaci LoRa-Alliance, která sdružuje několik členských organizací zabývajících se rozvojem komunikačních standardů v IoT sítích.

LoRaWAN sítě fungují na principu síťového modelu device-to-gateway. Zařízení pošle informaci bráně a ta se postará o přeposlání na aplikační server, kde s informací dále naložena dle potřeby. Brány komunikují se serverem pomocí TCP/IP protokolu. Komunikace mezi zařízením-bránou-aplikačním serverem je obousměrná. Směrem na aplikační server jsou zasílána data (sebrané informace ze senzoru) a zpět k zařízení jsou směrovány příkazy k nastavení snímání senzoru (frekvence snímání, rozlišení hodnoty a další).

Komunikace mezi koncovým zařízením a bránou může být rozprostřena do různých kanálů a rychlostí, aby bylo efektivně využito kapacity sítě z důvodu úspory energie nebo minimalizaci chybových přenosů. Jsou zde k dispozici rychlosti 0,3 kbps - 50 kbps.

LoRaWAN definuje tři třídy, které jsou specifické chováním v odesílání a přijímání dat:

- Třída A - koncové zařízení naslouchá na síti velmi krátkou dobu po odeslání dat, velmi malá spotřeba energie
- Třída B - koncové zařízení naslouchá na síti krátkou dobu po odeslání dat v předem určených intervalech
- Třída C - koncové zařízení naslouchá na síti nepřetržitě, po odeslání zprávy zařízení nenaslouchá, nejvyšší spotřeba energie

2.7.3 SigFox

Přenosová technologie SigFox je produktem ze stejnojmenné firmy. Jedná se o další přenosovou technologii, která byla navržena pro LPWAN sítě, které jsou náročné na nízkou spotřebu elektrické energie. SigFox je úzkopásmová technologie navržená pro přenos malých datových celkům na velké vzdálenosti. Frekvence zasílání zpráv je malá, což je dostačující při vyčítání informací z koncových senzorů pouze v předem stanovené časové intervaly.

U této technologie je možné posílat data pouze na frekvencích 868 MHz a 915 MHz, což jsou volné frekvence IoT pro Evropu a Ameriku. Pro odchozí komunikaci je využito binárního klíčování a s využití DBPSK a pro příchozí komunikace se používá GFSK.

Přenos dat v topologii sítě SigFox probíhá obousměrně. Při komunikaci dvou zařízení se nevyužívá navázání komunikace jako např. u TCP, ale data jsou vyslána a nejbližší zařízení, které je na dosah, data přijme a odešle je směrem k aplikačnímu serveru.

Maximální velikost zprávy je 12 bytů, kde v každé zprávě je zasláno ID zařízení a časová značka. Přenosová rychlost je 100 bps a přenosové pásmo je šířky 100 Hz, díky které je technologie velmi odolná vůči rušení.

2.7.4 IQRF

Technologie IQRF byla vyvinuta českou firmou Microrisc. Bezdrátový přenos je zprostředkováván pomocí transceiver modulů, které obsahují microcontroler a paměť typu EEPROM. Tato technologie využívá pásma 433, 868 a 915 MHz pro svůj přenos. Transceivery obsahují operační systém, který má v sobě implementovanou podporu bezdrátové komunikace v topologii mesh/peer-to-peer a zbytek ovládání modulu. Aplikaci, kterou lze uploadovat do modulu je možné napsat v jazyce C.

IQRF umožňuje přenášet až 64 bytů. Maximální přenosová vzdálenost je definovaná na 500 metrů. Na modulu je tištěná anténa, která má vysílací výkon 2,5 mW. Přenosová rychlost je definovaná na 20 kbps. Při využívání topologie sítě typu mesh se může doba přenášení dat ve vzduchu dostat až na jednotky vteřin. Tuto topologii je možné propojit s ostatními technologiemi pomocí brány, která zajišťuje připojení např. pomocí USB (Universal Serial Bus), ethernetu nebo GSM (Global System for Mobile Communications).

2.7.5 RFID

RFID (Radio Frequency Identification) je technologie, která dokáže identifikovat čipy pomocí rádiové frekvence. Jedná se o další generaci čipů a vylepšení technologie NFC (Near Field Communication). Slouží k bezkontaktní komunikaci na krátkou vzdálenost. Využívají se frekvence 125 a 134 kHz; 13,56, 868 a 915 MHz. RFID čipy se dělí na pasivní a aktivní. Pasivní čip využije vysílanou elergii z elektromagnetického pole, nabije svůj napájecí kondenzátor a odešle odpověď. Odpovědí může být číslo nebo kód určený již při výrobě čipu. Pasivní čipy se využívají v těchto aplikacích:

- identifikace předmětu v obchodě
- přístupy do uzavřených objektů

- bezhotovostní platby

Aktivní čipy jsou oproti pasivním čipům dražší a dokáží vysílat informaci, kterou v sobě mají uloženou. Tyto čipy mají v sobě uloženou celou řadu informací, které dokáží odesílat spolu se svou identifikací.

RFID čipy obsahují 96 bitové číslo EPC (Electronic Product Code), které je přiděleno každému kusu zboží pro příklad obchodu. Je možné využít číslo EPC ve velikosti 64 nebo 128 bitů, což pohybuje s cenou čipu. Služba ONS (Object Name Service) slouží k přiřazování každému číslu EPC předem určených informací ve formátu XML. V tomto formátu jsou uchovávány informace o zboží a dá se s nimi dále pracovat.

2.7.6 Mesh ZigBee

ZigBee patří mezi další beztrátové technologie, které se vyznačují nízkou spotřebou elektrické energie, menší přenosovou rychlostí a uzpůsobením pro zařízení a senzory v síti IoT. ZigBee dokáže přenášet data až na vzdálenost 75 m. Použití ad-hoc směrování umožňuje komunikaci i na větší vzdálenost. Přenosovou rychlost je možné dle aplikace nastavit na 20, 40 nebo 250 kbps. ZigBee pracuje v bezlicenčním pásmu 868, 915 MHz a 2,4 GHz.

Signál se pro přenos moduluje metodou BPSK. Pro přístup k fyzickému médiu je využita metoda CSMA/CA.

Tato technologie se nasazuje tam, kde bluetooth není vhodný. Nachází uplatnění hlavně v oboru automatizace a strojírenství. Význačnými vlastnostmi této technologie jsou spolehlivost jednoduchá implementace, nízká spotřeba elektrické energie a nízká cena.

Ve standardu definuje tři základní režimy přenosu informací:

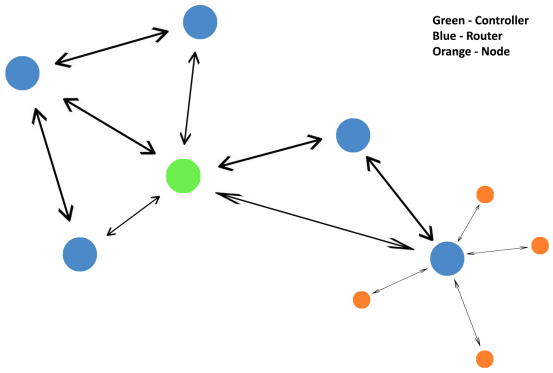
- periodicky se opakující přenosy dat (vyčítání senzorů)
- opakující se přenosy se zpožděním (externí periferie)
- nepravidelné přenosy dat (reakce na událost - stisk tlačítka, změna teploty)

Z důvodu úspory elektrické energie na zařízení jsou jednotlivé koncové body uspávány a probouzeny v pravidelných intervalech, během kterých jsou přeneseny všechny potřebné informace.

Tato technologie pracuje ve třech základních topologiích, a to hvězdicová, stromová a mesh. Ve všech třech typech je využit tzv. koordinátor, který řídí provoz v síti.

V síti existují čtyři typy přenášených rámců:

- data - přenos payload
- ack - potvrzování komunikace
- beacon - uvádění zařízení do spánkového režimu
- mac command - nastavování a řízení koncových zařízení

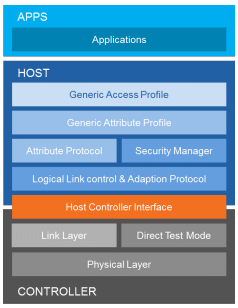


Obrázek 2.15: Zigbee network

Každé zařízení v síti je identifikováno 16 bitových číslem PAN ID, které přiděluje koordinátor řídící celou síť. Síťová vrstva se stará o připojení k síti, zabezpečení, směrování, synchronizaci a přiřazování adres nově připojeným zařízením. V síťové vrstvě se defaultně využívá zabezpečení typu AES-128b.

V aplikační vrstvě jsou definovány dvě podvrstvy APS (Application Support Sublayer) a ZDO (ZigBee Device Object). APS je zodpovědná za párování zařízení k požadavkům senzorů a aplikací. ZDO zajišťuje tři role zařízení v síti - koordinátor, směrovač, koncová zařízení, vyhledává nová zařízení a definuje zabezpečení a způsob ověřování.

2.7.7 Bluetooth Smart



Obrázek 2.16: Bluetooth stack [10]

Bluetooth je další z přenosových technologiích typu LPWAN. Bluetooth byl vytvořen jako náhrada za sériové drátové rozhraní RS-232. Bluetooth vytváří privátní síť (PAN) mezi dvěmi zařízeními. Tato technologie je specifická tím, že propojuje zařízení, která jsou většinou blízko sebe. Většinou se jedná o zařízení pro osobní potřebu či pro denní využití.

Jelikož mnoho zařízení propojených přes bluetooth přenáší soukromé inforvamce, je potřeba řešit šifrování dat a popř. i ověření připojení zařízení. K tomu slouží tzv. párování zařízení. Jedná se o uložení stejného pin kódu v obou zařízeních, která spolu komunikují. Tento společný tajný klíč je nutné mít v obou zařízeních. Pokud by byl na jedné straně

smazán, zařízení nepřenáší data dokud nedojde k dalšímu párování. Zařízení typu headset mají v sobě pin uložený továrně a tudíž není možné ho změnit.

Komunikace probíhá v pásmu 2,4 GHz. Bluetooth ve verzi 5.0 dokáže data přenášet až na vzdálenost 240 m při přenosové rychlosti 2 Mbps. Byla zde navržena funkcionalita uspání zařízení, co během nepřenášení žádných dat dokáže výrazne uspořit energii. Bluetooth není vhodný pro použití na větší vzdálenosti a ani do míst s větším množstvím připojených uzlů v síti. [10]

2.8 Datová komunikace a její bezpečnost

Datová komunikace je přenos digitálního nebo zdigitalizovaného analogového signálu. Médiem pro přenos dat může být metalický kabel, optický kabel nebo bezdrátový přenos. Pro přenos dat definujeme vysílač (Tx) a přijímač (Rx). Datová komunikace zahrnuje i přípravu přenosu a jeho řízení a následně procesy pokračující s informací po jejím úspěšném přijetí.

Data se přenáší pomocí modulování nosného signálu pomocí modulujícícho signálu.

Modulaci dělíme z hlediska modulované veličiny na následující druhy:

- amplitudová
- frekvenční
- fázová
- šířková

Dělení podle nosného signílu je následující:

- harmonický nosný signál
- impulsní nosný signál

Dělení modulace podle modulačního signálu obsahuje tyto skupiny:

- analogová
- číslicová

Analogová modulace používá za základní nosný signál takový, který je harmonický v čase, což je např. sinusoida. Modulačním signálem je v tomto případě analogový signál. Tato modulace je charakteristická pro potřebu přenosu úrovně napětí či proudu. Pro číslicovou modulaci je použito nosného signálu s harmonickým průběhem (stejně jako u analogové modulace) a modulační signál zastupuje digitální signál. Této podoby modulace se využívá při přenosu hodnoty zakódované do binární podoby.

Digitální signál je charakterizován rychlostí přenosu. Přenosová rychlost informace závisí na počtu stavů a rychlosti změn těchto stavů, což je tzv. modulační rychlost. Čím více stavů

bude signál nabývat, a čím rychleji je bude měnit, tím bude dosaženo větší přenosové rychlosti. Při přenosu informace jsou limitující vlastnosti přenosového kanálu (šířka kmitočtového pásma, rušení). [25]

Datový přenos můžeme dělit z několika pohledů. Dělení podle směru přenosu dat je jednosměrné, obousměrné (střídavé, současné). Uvedené dělení uvádí, jakým směrem jsou data přenášena mezi odesílatelem a příjemcem a zda oba členové komunikace využívají společný kanál a jakým způsobem.

Dělení podle počtu současně využívaných kanálů rozlišujeme na sériové a paralelní. U sériovéhe přenosu jsou data přenášena postupně v čase za sebou. U paralelního přenosu je přenáseno několik bitů po skupinách. Paralelní přenos je zpravidla rychlejší, ale je zapotřebí řídících signálů pro komunikaci.

Dalším dělením přenosu dat je způsob vkládání signálů na přenosové médium. Rozlišujeme sériový a paralelní přenos. U sériového přenosu přenášíme data sekvenčně. Tento typ přenosu nepotřebuje tolik práce se signálem jako paralelní přenos, a proto je zde eliminována většina chyb. Paralelní přenos se vyznačuje simultáním přenosem několika dat současně po několika různých vodičích. Zde je zapotřebí provoz řídit, a proto se mohou vyskytovat chyby v přenosu. Hlavní výhodou je velká přenosová rychlost, které se pomocí sériového přenosu nemůže z principu nikdy dosáhnout. [32]

2.8.1 Bezpečnost přenosu

V dnešní době je většina informací vytvářena, přenášena a udržována v digitální podobě, a proto je nutné je chránit, protože mohou být cílem různých útoků. Cíle, aby systém manipulující s digitálními daty byl důvěryhodný, určuje kryptografie.

Definujeme tři základní pojmy: kryptologie, kryptografie, kryptoanalýza. Kryptologie je vědní obor, který zahrnuje kryptografii a kryptoanalýzu. Kryptografie se zabývá návrhem a konstrukcí kryptografických algoritmů a jejich využíváním. Kryptoanalýza zkoumá odolnost a zranitelnost šifer a pokouší se prolomit zašifrovaný text bez znalosti klíče.

V terminologii kryptologie definujeme dva typy šifer, a to symetrickou a asymetrickou. Symetrická šifra používá pro šifrování a dešifrování stejný klíč. Asymetrická šifra používá odlišné klíče, jeden pro šifrování a jiný pro dešifrování. [38]

2.8.2 Kódování a dekódování

Kódování je zobrazení, které každému informačnímu slovu přiřadí slovo kódové. Kódování je dáno volbou báze v podprostoru. Každý kód tudíž může mít více způsobů, jak informační slova zakódovat. Informační znaky tedy jsou souřadnice kódového slova vůči konkrétní bázi. Zvolením této báze a určitých znaků nám zaručuje, že zobrazení je vzájemně jednoznačné.

Dekódování slova je inverzní operací ke kódování. Spočívá v tom, že nalezneme souřadnice kódového slova vůči původní bázi, která je v řádcích matice použité při zakódování. [17]

Pro velkou množinu lidí představuje internet velkou nezabezpečenou skupinu počítačů, kde si může kdokoliv vytvořit anonymní identitu a vystupovat zcela anonymně bez jakýchkoliv postihů za porušování zákona. Mnoho lidí dnes slepě důvěřuje všem informacím umístěných na internetu, bohužel i těm, které nejsou ani zdaleka pravdivé. Lidé se diví, že data

posílaná z jednoho zařízení na druhý lze číst po cestě kýmkoliv, avšak existují způsoby, jak tomuto zamezit. Jedná se o využití elektronického podpisu a šifrování obsahu zprávy.

Pro vytvoření bezpečné šifrované zprávy se v dnešní době využívá dvojice klíčů (veřejný a privátní), což je tzv. asymetrická kryptografie. Přenos dat vypadá následovně: odesílatel zašifruje data věřejným klíčem příjemce a odešle je na adresu příjemce. Ten si pomocí privátního klíče zprávu rozšifruje. Podmínkou tohoto typu šifrování je, že odesílatel zná veřejný klíč příjemce.

Elektronický podpis nahrazuje ve světe IT vlastnoruční podpis na papíře. Elektronickým podpisem je zajištěna:

- autenticita podeposující osobou
- integrita zprávy
- odpovědnost odesílatele

Elektronický podpis je přidaná informace k odesílaným datům, aby identifikovala odesílatele. Pomocí *hashovací* funkce se nejprve spočítá kontrolní součet zprávy a následně se zašifruje privátním klíčem odesílatele. Připojením k originálním datům pouze zašifrovaného kontrolnímu součtu se zvýší velikost původní zprávy pouze o několik bitů. Pro výpočet kontrolního součtu se dnes používají algoritmy SHA1 a SHA2. Každý, kdo zná veřejný klíč odesílatele si může ověřit zprávu, zda není porušená nebo modifikovaná.

K veřejnému klíči zadává certifikační autorita (CA) certifikát, který uvádí totožnost majitele veřejného klíče. V certifikátu jsou uvedeny informace jako je např. verze certifikátu, sériové číslo, specifikace algoritmu použitého pro veřejný klíč, platnost certifikátu, identifikace CA a uživatele. Certifikační autority jsou organizace, které vystavují certifikáty a ověřují totožnost žadatelů.

Přínosem šifrování elektronických zpráv je ochrana citlivých dat, které si vyměňuj odesílatel s příjemce. Současně omezuje i zneužití dat. Elektronický podpis lze použít i pro podepsání webových stránek s citlivými údaji nebo pro podepsání jakéhokoliv souboru.

Kapitola 3

Analýza

Serializace dat je velmi výkonný nástroj, který při správném použití dokáže šetřit jak kapacitu úložných prostor, tak kapacitu přenosového média při posílání dat mezi zařízeními na síti. Serializace je proces hodně používaný u síťové komunikace nebo v archivaci a ukládání dat. Běžně používáme framework, který provádí serializaci za nás. Problém těchto nástrojů je neznalost, co přesně se děje na pozadí.

V mnoha případech je vyvinout nástroj, který bude serializovat objetky a data sám. V komunikačních nástrojích si můžeme většinou vybrat metody serializace a způsobu přenosu dat, proto je důležité znát, jakým způsobem jaká metoda přistupuje k serializaci dat a zda je vhodná pro konkrétní řešení daného projektu.

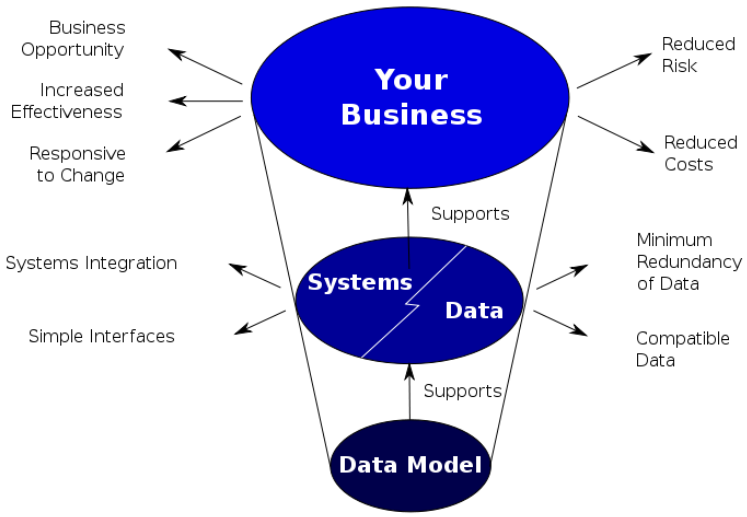
3.1 Nástroje pro popis datového modelu

Datový model poskytuje strukturu použitou pro data v daném informačním systému, která poskytuje specifikaci a formát dat. Pokud je stejný datový model používán napříč různými informačními systému, není problém v přenosu dat mezi nimi. Pokud jsou použity stejné struktury pro práci s daty, mohou různé aplikace sdílat tato data současně.

3.1.1 Protocol Buffers

Co je Protocol Buffers? Dokumentace říká následující: “Protocol Buffers are a way of encoding structured data in an efficient yet extensible format.”

Protocol Buffers je efektivní a automatizovaný systém pro serializaci strukturovaných dat. Protocol Buffers je nástroj, který je oproti jiným (XML, JSON) méně náročný na kapacitu v operační paměti, disponuje jednodušší implementací a rychlejším zpracováním dat (serializace/deserializace). Pro definování stačí jednou zvolit strukturu a následně lze použít generovaný zdrojový kód pro jednoduchý zápis a čtení strukturovaných dat z data streamů s použitím různých jazyků. Aktualizace podoby struktury je možná i za běhu. Pro začátek je potřeba definovat jak mají být serializovaná data strukturovaná, což se definuje v souborech zprávy typu *.proto*. Každá protocol buffer zpráva je malý logický záznam informace obsahující několik dvojic typu jméno-typ. Příklad *.proto* zprávy definující informace o člověku.



Obrázek 3.1: Výhody datových modelů [4]

Porovnání s XML má Protocol Buffers tyto výhody:

- jednodušší
- 3-10x menší
- 20-100x rychlejší
- jednoznačný
- generuje přístupové třídy, které jsou jednoduché použít v programu

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phone = 4;
}
```

Obrázek 3.2: Procotol Buffers - definice třídy Person [15]

Protocol Buffers není vhodné použít pro HTML, protože nelze propojit strukturu s textem. XML je “human-readable” a “human-editable”, ale protocol buffers ve svém výchozím

nastavení není. XML je taktéž velmi jasný, popisuje sám sebe, oproti tomu protocol buffers potřebuje definici zprávy v souboru ".proto". V současné chvíli byla představena verzi proto3, která rozšiřuje počet podporující programovacích jazyků s protocol buffers. [15]

3.1.2 FlatBuffers

FlatBuffers je efektivní platforma pro serializaci mnoha programovacích jazyků. Původně tato platforma byla vytvořena pro herní průmysl a kritické aplikace s důrazem na výkon.

Uchovává data v binární podobně, což dává možnost přístupu kdykoliv bez nutnosti parsování nebo další úpravy dat. Jediná nutná paměť pro tuto platformu je pro vyrovnávací paměť, ve které jsou data uložena. Nepotřebuje žádnou další alokaci paměti. Používá se v aplikacích, kde se klade velký důraz na vysokou rychlost přenosu a rychlost přístupu k datům. Serializace dat nemá u FlatBuffers velkou režii, a proto je možné přirovnat rychlost zpracování dat k době přístupu k datům přímo v operační paměti (RAM). Pro další rozšíření použitelnost jsou připravena volitelná pole pro zajištění např. zpětné kompatibility. Minimální verze FlatBuffers obsahuje velmi malé množství generovaného kódu a malou jednoduchou hlavičku, což dává možnost jednoduché integrace. Flatbuffers oproti ProtoBuffers nepotřebuje parsování nebo další předzpracování dat před jejich použitím. Samotný kód je kratší a zabírá méně paměti.

Ve shrnutí se jedná o binární zásobník, který obsahuje struktury, tabulky, vektory a další. Prvky v zásobníku jsou indexovány bitovým zápisem a s použitím ukazatele jsou adresovány a procházeny velmi efektivně. FlatBuffers používá např. Facebook pro komunikaci server-client (android aplikace), Cocos2d-x pro open-source mobilní herní nástroje k serializaci dat nebo Fun Propulsion Labs používá Google ve všech knihovnách a hrách. [18]

3.1.3 XML

XML (eXtensible Markup Language) byl vytvořen členy sdružení W3C. Byl navržen tak, aby podporoval co největší množství aplikací a byl jednoduchý na použití v internetu. Základní myšlenkou je také čitelnost člověkem (human-legible) a jednoduchost vytvoření. V současné době má XML velké použití a mnoho webových služeb ho používá.

Existuje několik verzí tohoto standardu. XML je kritizován za velké množství informací, které je uváděno i datech. Současně podpora pro mobilní telefony není daleko ve vývoji.

3.1.4 IPSO

IPSO (Internet Protocol for Smart Objects) byl navržen pro využití protokolu CoAP ke komunikaci mezi zařízením a serverem nebo mezi zařízeními navzájem. První představení bylo pro použití senzorů pro pračky prádla. Tyto senzory byly např. senzor vlhkosti, teploty, hladiny vody, elektrická pumpa, časovač a další.[24]

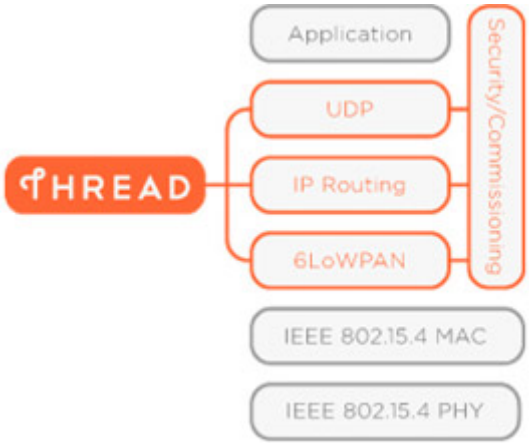
Použití IP protokolu pro zařízení s menší pamětí než 16 kB umožňuje použití nových možností propojení širšího spektra elektronických zařízení. Tato zařízení používají IPv6 a 6LowPAN pro komunikaci v síti.

3.1.5 Protokol Thread

Základní vlastnosti tohoto protokolu jsou spolehlivost, zabezpečení, efektivní spotřeba a nízká cena. Protocol Thread patří do skupiny IEEE 802.15.4 (low-rate wireless personal area networks). Tento protokol pracuje na frekvenci 2,4 GHz. Thread protocol byl vyvinut a schválen společností Thread Group, která sdružuje několik technologických firem, servisních společností a dalších. Výhody tohoto protokolu jsou

- jednoduché nastavení a používání
- zabezpečení
- nízká spotřeba elektrické energie
- otevřený protokol, který pracuje s IPv6
- robustní mesh topologie bez žádného SPOF (single point of failure)
- pracuje na standardu 802.15.4
- podporuje velkou řadu koncových zařízení

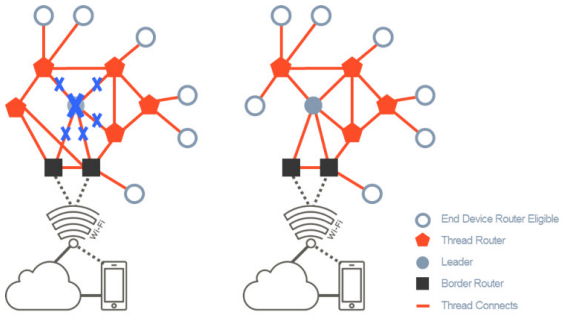
Síťová vrstva implementuje IPv6 adresaci s 6LoWPAN hlavičkou. Pro maximální efektivitu v routování a záložní linky je vyvinuta topologie mesh. Další výhodou tohoto protokolu je oddělenost aplikační vrstvy od fronty protokolu Thread. Výhodou je možnost diagnostiky a možnost nezávislého použití jiných technologií a standardů. Tento protokol není rozšířen na aplikační vrstvu. O aplikační vrstvu se nestará a umožňuje tak dalším nástrojům tuto vrstvu obsadit a pro spodní vrstvy použít Thread.



Obrázek 3.3: Protocol Thread diagram1 [23]

V topologii typu mesh, kterou protocol Thread podporuje, jsou jednotlivé uzly propojené s dalšími několika cestami. Každý uzel je zodpovědný sám za sebe, co se týká registrování a odpojování ze sítě. Pokud uzel opustí síť, celá síť je rekonfigurována a jsou přepočítány routovací cesty. V topologii Thread mesh existuje několik routerů, které jsou propojené mezi sebou. Díky možnostem mesh topologie v této implementaci je možné přejít od topologie

typu star do topologie typu mesh pouze přidání několika routerů. Tato flexibilita zvyšuje spolehlivost celého systému díky možnosti posílání zpráv na další spoje a uzly, odpadá zde problém SPOF.[23]



Obrázek 3.4: Protocol Thread diagram2 [23]

3.1.6 MessagePack

MessagePack je další z binárních formátů pro serializaci dat. Není sice tak známý jako BSON, ale v mnoha ohledech je velmi zajímavý. Zde je několik příkladů jeho kvalit:

- navržen pro efektivní komunikaci využívající kabelové spojení
- menší režie než BSON, může serializovat velmí malé objekty frekventovaně v čase
- podporuje kontrolu *static-typing*, což znamená, že typ proměnné je znám při kompilaci kódu
- umožňuje streamování díky *streaming API*, což je velmi efektivní v síťové komunikaci
- umožňuje vzdálené volání procedur

3.1.7 Frictionless Data

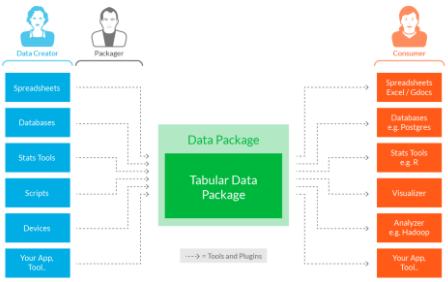
Frictionless Data je způsob odstranění nadbytečných informací, která nejsou nezbytně nutná zaslat. Srdcem tohoto nástroje je datový balíček, což je souhrn informací, jak popsat a zpracovat data pro další použití.

Základní hesla této technologie jsou:

- jednoduchost - všechna nadbytečná data jsou odstraněna, zůstavají pouze nezbytné informace
- rozšiřitelnost - umožňuje navázání nových technologií na základy Frictionless Data
- editovatelnost člověkem, čitelnost strojem
- není zaměřený na jednu konkrétní technologii - podporuje více programovacích jazyků, technologií a infrastruktur

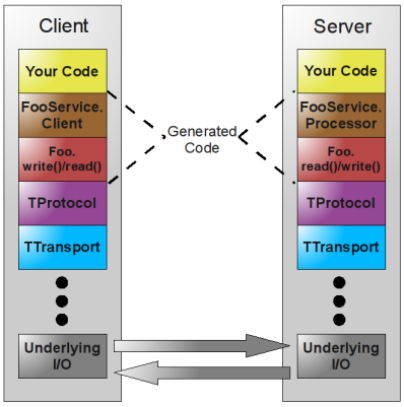
Výše popisovaná technologie se velmi úzce zaměřuje na zabalování dat a jejich serializaci. Jeden ze specifických typů dat je "tabular". Toto řešení je založeno na distribuovaném systému, který nemá SPOF nebo žádnou další závislost, na které by byl systém napadnutelný proti selhání.

Systém je napsán jako opensource, aby ho mohl každý kdo chce jednoduše použít. Integruje se zde velmi jednoduše za využití již existujících nástrojů. Např. za využití formátu *.csv*, který přečte takřka každý nástroj dnešní doby [14].



Obrázek 3.5: Frictionless Data - tabular[33]

3.1.8 Apache Thrift



Obrázek 3.6: Apache Thrift - stack [34]

Apache Thrift je softwarový framework, který je škálovatelný napříč různými programovacími jazyky. Byl vyvinut ve Facebooku pro zefektivnění a škálování služeb na pozadí. Jedná se jak o interface pro různé programovací jazyky, tak o binární komunikační protokol. I když byl vyvinut ve společnosti Facebook, nyní se jedná o opensource.

Thrift dovoluje programátorům definovat datové typy a servisní rozhraní v jazykově neutrálním prostředí a vygenerovat všechny nezbytné kódy pro RPC klienty a servery.

Výhodami Thriftu jsou např. oddělitelnost formátu pro serializaci dat a pro aplikační část. Předdefinované způsoby serializace obsahují binární část, HTTP a tzv. "Compact binary".[31]

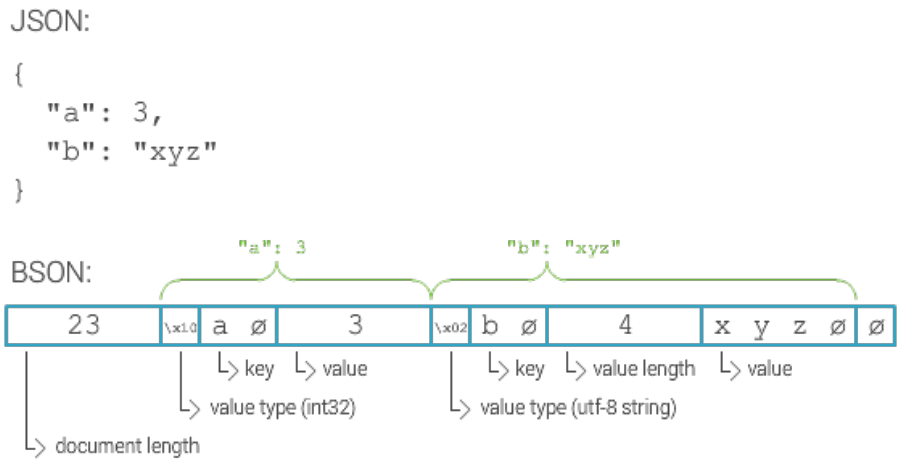
3.1.9 BSON

BSON (Binary JSON) je binární podoba JSONu (JavaScript Object Notation). BSON obsahuje rozšíření o datové typy, které nejsou podporované v JSON specifikaci (např. datový typ "Date"a "BinData"). V porovnání s Protofol Buffers je BSON více "schema-less", což nám dává výhodu ve flexibilitě protokolu, ale je trochu nevýhodný v zabírané kapacitě prostoru, BSON má režijní data např. pro jména polí v serializovaných datech.

BSON má tři základní charakteristické vlastnosti [1]:

- *lightweight* - udržuje režijní informace na minimální velikosti, což je důležité pro použití na síti
- *traversable* - procházení dat dává velkou výhodu při použití v MongoDB, kde zastupuje primární formát pro uchovávání dat
- *efficient* - kódování a dekódování z/do formátu BSON je velmi rychlé při použití jazyků odvozených od typu C

Hodnoty v tomto formátu jsou ukládány v podobě *key-value*. Tyto entity nazýváme dokumentem. K dokumentu přidává doplňující informace, např. délku textových řetězců nebo objektů, což umožňuje rychlejší procházení dat [8].



Obrázek 3.7: Apache Thrift - zásobník [13]

Kapitola 4

Testování

Tato kapitola popisuje porovnání mezi vybranými nástroji pro serializace objektů. Pro testování byly vybrány nástroje MessagePack, Protocol Buffer, JSON a XML. Tyto nástroje jsou popsány výše.

4.1 Porovnání datových modelů

Testování nástrojů pro popis datových modelů se bude zaměřovat na výkonnost daného nástroje. Testovány budou parametry: náročnost na využití operační paměti, rychlost serializace/deserializace a využitá kapacita v úložišti. Serializace je v mnoha případech používána zařízeními pro IoT, které jsou vyrobena většinou s malou výpočetní kapacitou (operační paměť, výkon procesoru, úložiště apod.).

Rychlost serializace je velmi důležitý parametr pro volbu datového modelu pro danou aplikaci. Jelikož aplikace může být navázána na kritické informační systémy, musí být všechny používané nástroje bezchybné, rychlé a stabilní. Na serializaci/deserializaci se klade velký důraz a nesmí zbytečně brzdit zpracování a přenos informací. Rychlost zpracování dat je silně spojena s výpočetní rychlostí procesorové jednotky daného zařízení.

Dalším důležitým parametrem pro datové modely je velikost dat, která zabírají prostor v operační paměti a v interní paměti. Jelikož používáme pro IoT sítě zařízení s nízkou spotřebou elektrické energie, mají tato zařízení omezenou výpočetní kapacitu a současně i paměť. Některá zařízení je možné rozšířit o externí paměťové médium, ale některá tuto možnost nemají. Jsou zde hraniční hodnoty i pro množství odesílaných dat a frekvenci přístupu na přenosové médium. Jelikož každé zapnutí zařízení, zpracování dat a jeho odeslání zvyšuje spotřebu elektrické energie, je důležité provést implementaci velmi důmyslně.

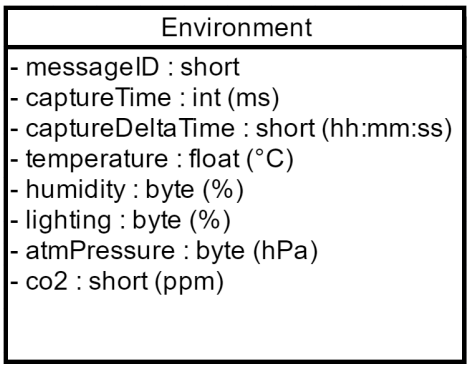
Použitelnost datových modelů hraje velkou roli v následné rozšiřitelnosti na další systémy nebo servery. Požadavkem na datový model je jednoduchá implementovatelnost. Data musí být možné zaslat na různé webové nebo aplikační servery, ať už přímo nebo pomocí dalšího zařízení (brány). Pro co nejširší použití musí být datový protokol multiplatformní [9].

Tabulka 4.1: Porovnani datovych modelu

Název	Standardizace	Binarní	Čitelné člověkem	Podpora serializace
Protocol Buffers	ano	ano	částečně	ano
JSON	ano	ne	ano	ne
CSV	částečně	ne	ano	ne
FlatBuffers	ne	ano	ano	ano
XML	ano	částečně	ano	ano
Protocol Thread	ano	ano	ano	ano
MessagePack	ano	ano	ne	ano
Frictionless Data	ne	ano	ano	ano
Apache Thrift	no	ano	částečně	ano
BSON	ano	ano	ne	ano

4.2 Implementace vlastního datového modelu

Datový model byl vybrán na základě prostředí chytré domácnosti. V tomto prostředí měříme parametry jako jsou např. teplota, osvětlení, tlak apod. Na základě této volby vznikl následující datový model, který je popsán v UML diagramu, který je nezávislý na programovacím jazuku.



Obrázek 4.1: Defincie datového modelu Environment

Datový model obsahuje dva typy konstruktorů. Objekt je možné vytvořit buď zadáním všech hodnot, které jsou uvedeny. Je možné vytvořit objekt, který nemá zadanou žádnou hodnotu, a všechny atributy jsou nastaveny na nulu. Poté je možné každou atributu dodatečně naplnit, nebo ji nechat nastavenou ve výchozím stavu. Tohoto stavu je dosaženo z důvodu ohledu na typ veličiny, spotřeby energie jednotlivých čidel a variabilního intervalu odečtů hodnot.

Hodnota scanningTime určuje čas, kdy byly změřeny ty hodnoty, které jsou různé od nuly. Může být změřena jedno a více hodnot ve stejném čase, poté je možné pomocí navrženého modelu přenést tyto hodnoty současně.

- messageID - 0-1140, 11 bitů

- captureTime [ms] - měsíc 4 bity, den 5 bitů, hodina 5 bitů, minuta 6 bitů, sekunda 6 bitů, celkem 26 bitů
- captureDeltaTime [mm:ss] - přesnost sekundy, časový rozdíl mezi posledním a aktuálním výčtem hodnot ze senzorů, 12 bitů
- temperature [°C] - přesnost 0.1, rozsah hodnot -40.0 - +40.0, 10 bitů
- humidity [%] - přesnost 1%, rozsah hodnot 0-100, 7 bitů
- lighting [%] - přesnost 1%, rozsah hodnot 0-100, 7 bitů
- atmPreasure [hPa] - přesnost 1%, rozsah hodnot 960-1100, 8 bitů
- co2 [ppm] - přesnost 1%, rozsah hodnot 350-5000, 13 bitů

U výše popsaného modelu se sběr dat předpokládá alespoň 1x za hodinu, tzn. alespoň 24x za den a maximálně 1x za minutu.

4.3 Popis testování

V následujícím experimentu jsou testovány nástroje pro popis datových modelů. Tyto nástroje jsou buď v menší nebo ve větší míře využívány v sítích IoT.

Testované nástroje jsou:

- ProtocolBuffer
- MessagePack
- XML
- JSON

ProtocolBuffer a MessagePack jsou nástroje, které ukládají data v binární podobě. Narozdíl XML a JSON jsou nástroje, které ukládají data v podobě čitelné člověkem (human-readable). Všechny čtyři nástroje je možné použít na zařízeních osazených mikrokontrolérem. Pro JSON existuje např. minimalistická verze JSMN ("jasmine"), kterou lze implementovat na velmi jednoduchá zařízení s omezeným výkonem.

Následující testování bude rozděleno do dvou částí. V první části testování bude kladen důraz na velikost paměti, kterou zabírají serializované objekty, dobu zpracování objektů při serializaci a při deserializaci.

Ve druhé části testování jsou výše zmíněné technologie použity s technologií IQRF, pomocí které jsou serializované objekty přenášeny a následně deserializovány. Měří se čas od začátku serializace objektů, přes přenos na jiné zařízení, až po ukončení deserializace posledního přenášeného objektu.

Počet serializovaných objektů je 1, 10, 100 a 1000. Každá sada objektů je zpracována 500x a následně statisticky vyhodnocena.

Testování probíhá na notebooku, který má následující parametry:

Tabulka 4.2: HW a SW parametry notebooku

CPU: Intel Core i7-3630QM 2.40 GHz
RAM: 16 GB
SSD: ADATA SX900
OS: Windows 10 x64
verze javy: 1.8.0_131
verze .NET: 4.7

4.3.1 Testování 1

V tomto testování jsou nástroje ProtocolBuffer, MessagePack, XML a JSON testovány pro:

- závislost využití operační paměti RAM na počtu zpracovávaných objektů
- závislost využití paměti datového úložiště na počtu zpracovávaných objektů
- čas pro serializaci/deserializaci objektů na počtu zpracovávaných objektů

Používané knihovny pro testování jsou tyto:

- jackson-databind-2.6.3 (837 KB)
- jackson-annotations-2.6.0 (48 KB)
- jackson-core-2.6.3 (265 KB)
- javassist-3.18.1-GA (501 KB)
- json-simple-1.1.1 (17 KB)
- msgpack-0.6.12 (276 KB)
- protobuf-java-2.5.0 (270 KB)

Měření délky trvání zpracování úseků v kódu v Javě je měřeno pomocí *System.currentTimeMillis()*. Před začátkem meřené části programu je uložena hodnota aktuálního času v milisekundách. Tato hodnota je zjištěna i po ukončení měřené části. Zmíněné dvě uložené hodnoty jsou od tebe odečteny, a tím je získán čas, po který byla v chodu měřená část programu.

```
startTime = System.currentTimeMillis();
/* process code */
endTime = System.currentTimeMillis();
totalTime += endTime - startTime;
```

Měření délky trvání běhu kód v C# je měřeno pomocí třídy Stopwatch.

```
Stopwatch sw = Stopwatch.StartNew();
sw.Restart();
/* process code */
sw.Stop();
totalTime += sw.ElapsedMilliseconds;
```

Měření náročnosti na využití paměti RAM v Javě probíhá pomocí využití *Runtime.getRuntime()*. Na začátku měřené části programu je zjištěna volná pamět ve virtuálním stroji, která je odečtena od celkové kapacity. Výsledkem je využitá kapacita spuštěných programem. Stejný postup je na konci měřené části programu. Tyto dvě hodnoty jsou odečteny, a tím je získáno množství využitíé paměti částí programu.

```
Runtime runtime = Runtime.getRuntime();
usedMemoryBefore = runtime.totalMemory() - runtime.freeMemory();
/* process code */
usedMemoryAfter = runtime.totalMemory() - runtime.freeMemory();
memoryIncreased = usedMemoryAfter - usedMemoryBefore;
```

Měření využití paměti RAM v C# probíhá pomocí třídy PerformanceCounter.

```
PerformanceCounter ramCounter = new PerformanceCounter();
usedMemoryBefore = ramCounter.NextValue();
/* process code */
usedMemoryAfter = ramCounter.NextValue();
memoryIncreased += usedMemoryAfter - usedMemoryBefore;
```

Testování 1 je naimplementováno pro dva programovací jazyky a to Java a C#. Pro jazyk Java jsem využil Netbeans ve verzi 8.2. Pro jazyk C# byl

C# je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft. Později byl schválen standardizačními komisemi ECMA (ECMA-334) a ISO (ISO/IEC 23270). C# lze využít k tvorbě databázových programů, webových aplikací a stránek, webových služeb, formulářových aplikací ve Windows, softwaru pro mobilní zařízení (PDA a mobilní telefony) atd. V současné době je tento jazyk ve verzi 5.0. Microsoft Visual Studio je oficiální vývojové prostředí od společnosti Microsoft pro tvorbu aplikací v jazece C#. [2]

Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems a představila v roce 1995. Jde o jeden z nejpoužívanějších programovacích jazyků na světě. Díky své přenositelnosti je používán pro programy, které mají pracovat na různých systémech. V polovině roku 2007 Sun uvolnil zdrojové kódy Javy (cca 2,5 miliónů řádků kódu) a Java je dále vyvíjena jako open source. Společnosti Google a Android si zvolily použití Javy jako klíčový pilíř při tvorbě stejnojmenného open source mobilního operačního systému určeného

pro chytré telefony a tablety. I přesto, že je Android postaven na Linuxovém jádře a byl napsán převážně v programovacím jazyce C, tak SDK Androidu používá jazyk Java jako základ pro svoje aplikace. Java je však používána pouze pro syntaxi, nikoliv pro svoji knihovnu tříd. [7]

4.3.2 Testování 2

V testování 2 jsou serializační nástroje Protocol Buffer, MessagePack, XML a JSON podrobeny testování spolu s přenosovou technologií IQRF. K této technologii existuje celá řada návodů a nástrojů, které jsou umístěny na webových stránkách společnosti Microrisc.

Pomocí technologie IQRF je vytvořen P2P (point-to-point) spoj mezi dvěma uzly typu TR-72DA v2.12. Tyto dva uzly slouží pro přijetí a odeslání dat. Síť P2P je vytvořena pomocí IQRF IDE, které je zdarma dostupné z webových stránek IQRF [5]. Tento nástroj slouží k vytvoření sítě, k jejímu nastavení, obsluze a servisu.

Testování bude probíhat následovně:

- odeslání informace pro začátek odpočtu
- serializace objektu nebo série objektů
- odeslání serializovaných dat pomocí technologie IQRF
- přijetí dat druhou stranou
- deserializace dat
- ukončení odpočtu

Výše uvedené body jsou opakovány 500x a získaná data jsou následně statisticky vyhodnocena.

Při tomto měření je sledována doba od začátku serializace dat až po ukončení deserializace posledního přijatého packetu.

Měření délky trvání zpracování úseků je měřeno pomocí *System.currentTimeMillis()*.

```
startTime = System.currentTimeMillis();
/* process code */
endTime = System.currentTimeMillis();
totalTime += endTime - startTime;
```

Serializace objektů probíhá na osobním počítači, Data jsou předána technologii IQRF pro odeslání. Po přijetí packetů je provedena deserializace taktéž na osobním počítači. V této fázi testování bude použit balíček IQRF SDK [6], který umožňuje propojení aplikace v Javě spolu s uzly IQRF. Komunikace RPI a IQRF nodu bude probíhat přes USB.

Kapitola 5

Vyhodnocení testů

5.1 Testování 1

V testování 1 byl použit datový model, který je uveden v implementaci. Jedná se o třídu Environment, který byla navržena pro informace získáváné z prostředí domu. Testovány byly čtyři nástroje Protocol Buffer, MessagePack, XML a JSON.

Každé testování bylo prováděno 500x a následně statisticky vyhodnoceno pomocí grafu typu box plot.

Testovány byly veličiny čas, velikost využité paměti a počet objektů. Každý nástroj byl použit pro všechna tato testování:

- rostoucí velikost využívané paměti RAM při zvyšujícím se počtu serializovaných/deserializovaných objektů
- rostoucí velikost využívané paměti v datovém úložišti při zvětšujícím se počtu serializovaných objektů
- rostoucí doba při serializování/deserializování objektů

V tomto testování se prokázalo vhodné pro IoT použít nástroje MessagePack nebo ProtocolBuffer. Jejich společnou vlastností je binární serializace dat, která je jak rychlá, tak úsporná na využívanou kapacitu serializovaných souborů, tak v úsporném využití operační paměti RAM.

Všechny nástoje, které byly použity, jsou v testování použity ve výchozím nastavení. Je pravděpodovné, že při různých úpravách dílčích nastaveních nástrojů by mohlo dojít k odlišným výsledkům měření.

Testování 1 bylo naimplementováno pro dva programovací jazyky a to Java a C#. Naměřené hodnoty mězi těmito dvěma programovacími jazyky jsou patrné. Více je popsáno v následující kapitolách.

5.1.1 Závislost počtu objektů na času serializace

U všech testovaných nástrojů se projevila rostoucí doba serializace v závislosti na zvětšujícím se počtu serializovaných objektů. V jazyce Java v tomto testu nejlépe obstál nástroj Protocol Buffer. Nejhůře se projevil nástroj XML. Pro jazyk C# je zde značný rozdíl. Jako nejhorší se projevil protokol JSON a oproti tomu protokol XML se projevil velmi podobně jako Protocol Buffer a MessagePack. XML v současné době se již nenasazuje do nových systémů kvůli své časové a paměťové náročnosti. Ve stávajících systémech, ve kterých je nasazen se jeho použití buď udržuje nebo postupně přechází na jiný z nástrojů, jako je např. JSON.

5.1.2 Závislost počtu objektů na času deserializace

Pro Javu, v tomto testování všechny testované nástroje vykazují rostoucí čas při zvyšujícím se počtu deserializovaných objektů. Nejhůře dopdl XML, obdobně jako u testování času serializace. Nejlepší výsledky u deserializace objektů vykazuje nástroj JSON, který se dnes ve velké míře používá např. pro webové služby.

Při porovnání stejných nástrojů v jazyce C# dostáváme podobné výsledky. Nejlépe se projevil protokol JSON v rozmezí od 1 do 100 objektů vykazuje několikanásobně menší čas pro serializaci objektů. Od počtu 100 objektů jeho doba serializace mírně klesá. Ostatní nástroje se chovají velmi obdobně. Od počtu 100 objektů je nárůst doby pomalejší než do počtu 100 objektů.

5.1.3 Závislost počtu objektů na využívané paměti RAM při serializaci

Při rostoucím počtu objektů se projevuje u třech ze čtyř nástrojů vyšší využití paměti RAM. U jediného nástroje XML výsledky ukázaly, že při rostoucím počtu objektů zabírá menší množství paměti. Tento nástroj se ukazuje jako rychlý pro větší množství objektů, avšak pro IoT, kdy je stěžejní zaslání jednoho objektu za delší časový úsek, se nestává velmi vhodným kandidátem. Tyto výsledky jsou pro jazyk Java.

Stejná implementace těchto serializačních nástrojů pro jazyk C# ukazuje rozdílné výsledky. Nástroj XML vykazuje větší využití operační paměti při rostoucím počtu serializovaných objektů. Vyšší nárůst nastává v počtu objektů 10. Oproti tomu nástroj JSON se ukazuje jako nejméně náročný a velikost využívané paměti při počtu objektů 100 začně klesat.

5.1.4 Závislost počtu objektů na využívané paměti RAM při deserializaci

Pro jazyk Java jsou výsledky následující. Při deserializaci objektů se ukázal jako nejméně náročný na využití operační paměti RAM nástroj MessagePack pro menší počet objektů. Nástroje Protocol Buffer a JSON se ukázaly stejně náročné ve využití pro objekty 1-10. Nástroj XML při využití pro menší počet objektů zůstává velmi náročný.

U jazyku C# dostáváme výsledky zcela opačné. Nástroj JSON se u předchozích testování jevil jako nadpůrměrný, ale jeho využívání paměti při deserializaci objektů je vyšší než u ostatních měřených nástrojů. Ostatní nástroje pozvolna navyšují využívanou paměť při rostoucím počtu objektů.

5.1.5 Závislost počtu objektů na velikost využívané paměti

Všechny nástroje byly testovány pro velikost využité paměti v paměťovém úložišti. Z grafu je patrné, že binární nástroje Protocol Buffer a MessagePack využívají mnohem méně paměti než *human-readable* nástroje. Velikost využívané paměti roste přímoúměrně s počtem serializovaných objektů. Nejmenší velikost jednoho objektu má nástroj MessagePack, následuji Protocol Buffer. JSON a XML generují objekty několikanásobně větší.

Co se týká porovnání obou programovacích jazyků, tak nárůst je takřka shodný. V následujících tabulkách jsou uvedeny hodnoty pro velikosti serializovaných objektů uložených v paměti.

Tabulka 5.1: Velikosti serializovaných objektů v Javě (hodnoty jsou uvedeny v bytech)

nástroj/počet objektů	1	10	100	1000
XML	509	5090	50900	509000
JSON	116	1160	11600	116000
Protocol Buffer	23	230	2300	23000
MessagePack	22	220	2200	22000

Tabulka 5.2: Velikosti serializovaných objektů v C# (hodnoty jsou uvedeny v bytech)

nástroj/počet objektů	1	10	100	1000
XML	367	3670	36700	367000
JSON	101	1010	10100	101000
Protocol Buffer	23	230	2300	23000
MessagePack	22	220	2200	22000

5.2 Testování 2

Při testování 2 byla vytvořena síť P2P mezi nody IQRF TR-72DA. Tato síť byla vytvořena pomocí IQRF IDE verze 4, která je volně dostupná z webových stránek IQRF. Při prvním testování funkčnosti sítě byla zaslána testovací data. Tato testovací data jsou dostupná ke stažení v balíčku IQRF IDE.

Další fází testování bylo propojení IQRF SDK se stávající aplikací vytvořenou v testování 1. V tomto kroku se vyskytly problémy s propojením tohoto SDK. Implementace IQRF SDK do stávající aplikace se nepovedla z důvodu chyby propojení IQRF nodu přes USB. Přes všechny dostupné možnosti se mi nepodařilo propojit danou technologii s IQRF SDK. Vyzkoušel jsem instalaci různých operačních systémů, reinstalaci ovladačů pro IQRF nodu, přeflashování firmwaru v IQRF nodech a další možná úskalí těchto problémů. Přes všechny neúspěšné pokusy jsem kontaktoval podporu IQRF. Po výměně několika emailů jsme nedošli k závěru, jak technologii zprovoznit. V komunikaci s IQRF podporou byl v kopii vedoucí práce. Emailová komunikace je umístěna na přiloženém CD.

5.3 Zhodnocení testování

Z výše uvedených výsledků je patrné, že nelze vybrat nejlepší a nejhorší serializační nástroj. Každý z nástrojů je vhodný pro jinou aplikaci. Co se týká IoT, tak se jeví jako nejvhodnější použití nástrojů binárních, které jsou pro menší množství objektů rychlejší a nezabírají tak velké množství paměti.

V IoT jsou používaná zařízení s malým výpočetním výkonem a menší pamětí. Tato zařízení mají menší spotřebu elektrické energie a je možné je bez výměny nebo nabíjení baterií používat několik let. Tato zařízení mají výhodu dlouhé životnosti, ale nevýhodu menšího výpočetního výkonu. Při implementaci různých knihoven je potřeba vybírat takové nástroje, které jsou paměťově a výkonnostně nenáročné.

Při rozhodování nad zvolením serializačního nástroje je dalším důležitým parametrem způsob serializace objektů do podoby binární nebo čitelné. Pokud data není potřeba číst, editovat a upravovat po jejich serializaci, je vhodnější použít binární formát dat, který je paměťově a výpočetně méně náročný.

Při porovnání serializačních nástrojů v rámci více jazyků je zapotřebí brát zřetel na funkčnost samotného jazyka. Jakým způsobem je přidělována paměť pro jednotlivé procesy nebo jak je řízeno přidělování procesorového času.

Naměřené hodnoty jsou umístěny na přiloženém CD.

Závěr

Ve svém bakalářské práci jsem shrnul hlavní myšlenku internetu věcí. Jelikož je tento tren na začátku svého vývoje, nelze jednoznačně říci, jakým směrem se bude vyvíjet technologie za pár let. V práci jsem uvedl příklady využití internetu věcí v několika hlavním oblastech, ve kterých lze očekávat velký technologický rozmach.

Uvedl jsem několik přenosových technologií internetu věcí, které patří v dnešní době mezi nejpoužívanější. Zejména jsou to technologie LoRa, SigFox a IQRF. Tyto technologie jsou význačné svými vlastnostmi pro přenos dat, pracují v bezlicenčním pásmu a odlišují se od ostatních technologií svými nízkými pořizovacími náklady. I když tato zařízení mohou vysílat v bezlicenčním pásmu, existuje zde celá řada omezení jako je např. omezení vysílacího času.

Hlavním cílem mé práce bylo otestovat nástroje pro popis datových modelů, které ve velké míře disponují i serializačními prostředky. Tyto nástroje se dělí podle výstupu serializovaného objektu na binární a *human-readable*. Po analýze trhu a z porovnání jednotlivých nástrojů vyšlo nejefektivnější porovnání nástrojů Protocol Buffer, MessagePack, XML a JSON. Tyto nástroje jsou ve velké míře používány. Některé z nich více, některé méně. Protocol Buffer a MessagePack jsou nástroje s binárním výstupem a XML s JSONem s výstupem *human-readable*.

Zmíněné nástroje jsem podrobil testování. Testování probíhalo ve dvou programovacích jazycích, a to v Javě a v C#. První testování bylo navrženo pro 500 opakujících se cyklů, abych předešel statistické chybě testování. Nástroje byly testováno pro dobu serializace a deserializace, velikosti využívané paměti RAM a velikost využívané paměti serializovanými objekty. Při testech bylo využito logaritmicky rostoucí množství objektů počínaje jedním objektem a konče tisícem objektů.Jak jsem předpokládal před samotným měřením, se zvyšujícím se počtem objektů bude narůstat jak doba serializace/deserializace, tak množství využité paměti RAM a paměti pro ukládání serializovaných objektů. Z testů vyplynulo, že efektivnější pro internet věcí jsou nástroje binární, jelikož jejich nárok na implementaci a používání je mnohonásobně menší než u nástrojů *human-readable*.

Druhé testování bylo navrženo na serializaci objektů pomocí dříve zmíněných čtyř vybraných nástojů, následné odeslání serializovaných dat pomocí technologie IQRF, přijetí dat a deserializování. Toto testování však nebylo zrealizováno z důvodu nefunkčního spojení modulů IQRF s SDK balíčkem. Problématiku jsem konzultoval s technickým oddělením IQRF, s vedoucím práce a s dalšími kolegy z Fakulty elektrotechnické. Odhad pro měření 2 je stejný jako u měření 1, kdy se projevují binární nástroje jako efektivnější pro využití v síti internetu věcí.

Z celkého měření vyplává, že použití binárních serializačních nástrojů je mnohem efektivnější z pohledu menší náročnosti na výpočetní výkon mikrokontroléru a menší náročnost na

využívanout paměť. Jelikož zažízení internetu věcí jsou velmi závislá na spotřebovávané energii, jsou vyráběna s co nejmenším odběrem. Binární nástroje vytěžují procesorovou jednotku mnohem méně než nástroje *human-readable*.

Přínosem této práce je v porovnání serializačních nástrojů, které jsou nezbytné pro efektivní přenos dat v síti internetu věcí. V porovnání se stávajícími nástroji např. pro www, jsou nástroje jako je Protocol Buffer nebo MessagePack mnohem efektivnější, rychlejší a méně náročné na výkon procesorové jednotky.

I když se nepovedlo propojit IQRF SDK s vytvořenou aplikací, přínos práce shledávám v porovnaných serializačních nástrojích. Vytvořené porovnání je dále možné použít při vybírání vhodného serializačního nástroje k implementaci aplikace pro IoT.

Zdrojové kódy využité k testování jsou na přiloženém CD.

Literatura

[1] *BSON Spec* [online]. [cit. "21. 5. 2017"]. Dostupné z: <http://bsonspec.org/spec.html>.

[2] *C#* [online]. Wikipedia. [cit. "21. 5. 2017"]. Dostupné z: <https://cs.wikipedia.org/wiki/C_Sharp>.

[3] *Všeobecné oprávnění č. VO-R/10/11.2016-13 k využívání rádiových kmitočtů a k provozování zařízení krátkého dosahu* [online]. Český telekomunikační ústav, 2016. [cit. "21. 5. 2017"]. Dostupné z: <http://www.ctu.cz/sites/default/files/obsah/ctu/vseobecne-opravneni-c.vo-r/10/11.2016-13/obrazky/vo-r10-112016-13.pdf>.

[4] *Data modeling* [online]. Wikipedia. [cit. "21. 5. 2017"]. Dostupné z: <https://en.wikipedia.org/wiki/Data_model>.

[5] *IQRF IDE* [online]. MICRORISC s.r.o., IQRF, . [cit. "21. 5. 2017"]. Dostupné z: <http://www.iqrf.org/technology/iqrf-ide>.

[6] *IQRF SDK* [online]. MICRORISC s.r.o., IQRF, . [cit. "21. 5. 2017"]. Dostupné z: <http://www.iqrf.org/technology/iqrf-sdk>.

[7] *Java (programovací jazyk)* [online]. Wikipedia. [cit. "21. 5. 2017"]. Dostupné z: <https://cs.wikipedia.org/wiki/Java_(programovaci_jazyk)>.

[8] *JSON and BSON* [online]. MongoDB, Inc. [cit. "21. 5. 2017"]. Dostupné z: <https://www.mongodb.com/json-and-bson>.

[9] AUDIE SUMARAY, S. K. M. *A Comparison of Data Serialization Formats For Optimal Efficiency on a Mobile Platform* [online]. Christopher Newport University, Lamar University. [cit. "21. 5. 2017"]. Dostupné z: <http://dl.acm.org/citation.cfm?id=2184810>.

[10] BLUETOOTH SIG, I. *Bluetooth Core Specification* [online]. [cit. "21. 5. 2017"]. Dostupné z: <https://www.bluetooth.com/specifications/bluetooth-core-specification>.

[11] COMPUTERHOPE. *HTTP* [online]. ComputerHOPE, 2017. [cit. "21. 5. 2017"]. Dostupné z: <http://www.computerhope.com/jargon/h/http.htm>.

[12] COMPUTERHOPE. *Protocol definition* [online]. ComputerHope, 2017. [cit. "21. 5. 2017"]. Dostupné z: <http://www.computerhope.com/jargon/p/protocol.htm>.

[13] CROCODILLON. *MongoDB for DBAs: Introduction* [online]. 2013. [cit. "21. 5. 2017"]. Dostupné z: <http://crocodillon.com/images/blog/2013/mongodb-for-dbas__bson.png>.

[14] DATA, O. G. *Frictionless Data Spec* [online]. [cit. "21. 5. 2017"]. Dostupné z: <http://specs.frictionlessdata.io>.

[15] DEVELOPERS, G. *Protocol Buffers - Developer Guide* [online]. Google, Inc. [cit. "21. 5. 2017"]. Dostupné z: <https://developers.google.com/protocol-buffers/docs/overview>.

[16] FOUNDATION, X. O. *An Overview of XMPP* [online]. [cit. "21. 5. 2017"]. Dostupné z: <https://xmpp.org/about/technology-overview>.

[17] GOLLOVA, A. *Kódování* [online]. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2012. [cit. "21. 5. 2017"]. Dostupné z: <https://math.feld.cvut.cz/gollova/tik/tik.pdf>.

[18] GOOGLE, I. *FlatBuffers* [online]. [cit. "21. 5. 2017"]. Dostupné z: <https://google.github.io/flatbuffers/>.

[19] HAMILTON, D. *The Four Internet of Things Connectivity Models Explained* [online]. WHIR, 2016. [cit. "21. 5. 2017"]. Dostupné z: <http://www.thewhir.com/web-hosting-news/the-four-internet-of-things-connectivity-models-explained>.

[20] HERNANDEZ, C. *IoT Smart City – What is a Smart City?* [online]. INFINITE INFORMATION & TECHNOLOGY, 2016. [cit. "21. 5. 2017"]. Dostupné z: <http://www.infiniteinformationtechnology.com/iot-smart-city-what-is-smart-city>.

[21] HUSAK, M. *Parametry senzorů* [online]. České vysoké učení technické v Praze, Fakulta elektrotechnická, . [cit. "21. 5. 2017"]. Dostupné z: <https://moodle.fel.cvut.cz/mod/resource/view.php?id=28167>.

[22] HUSAK, M. *Úvod - proč senzory* [online]. České vysoké učení technické v Praze, Fakulta elektrotechnická, . [cit. "21. 5. 2017"]. Dostupné z: <https://moodle.fel.cvut.cz/mod/resource/view.php?id=28164>.

[23] INC., D. I. *THREAD WIRELESS NETWORKING PROTOCOL* [online]. 2015. [cit. "21. 5. 2017"]. Dostupné z: <https://www.digi.com/resources/Onlines-and-technologies/thread-networking-protocol>.

[24] IPSO-ALLIANCE. *IPSO Smart Object Guideline* [online]. 2015. [cit. "21. 5. 2017"]. Dostupné z: <https://github.com/IPSO-Alliance/pub/blob/master/docs/IPSO-Smart-Objects-Expansion-Pack.pdf>.

[25] JARES, P. *Moderní modulační metody a jejich aplikace* [online]. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2013. [cit. "21. 5. 2017"]. Dostupné z: <http://data.cedupoint.cz/oppa_e-learning/2_KME/163.pdf>.

[26] JHINGRAN, P. *Reducing food spoilage using IoT* [online]. 2016. [cit. "21. 5. 2017"]. Dostupné z: <https://www.linkedin.com/pulse/reducing-food-spoilage-using-iot-prashant-jhingran>.

[27] JOHNSON, A. *IoT App Development Areas And Major Challenges* [online]. [cit. "21. 5. 2017"]. Dostupné z: <https://www.slideshare.net/astoria0128/iot-app-development-areas-and-major-challenges>.

[28] KONTOPOULOS, G. *Resource constrained protocols for IoT: 6LoWPAN, MQTT & CoAP* [online]. LinkedIn, 2016. [cit. "21. 5. 2017"]. Dostupné z: <https://www.linkedin.com/pulse/resource-constrained-protocols-iot-6lowpan-mqtt-coap-kontopoulos>.

[29] LTD., I. *Isode's Presence, Real Time Messaging and XMPP Strategy* [online]. ISODE. [cit. "21. 5. 2017"]. Dostupné z: <https://www.isode.com/whitepapers/xmpp.html>.

[30] MALY, M. *Protokol MQTT: komunikační Online pro IoT* [online]. Seriál BigClown se představuje, 2016. [cit. "21. 5. 2017"]. Dostupné z: <https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/>.

[31] MARK SLEE, A. A. – KWIATKOWSKI, M. *Thrift: Scalable Cross-Language Services Implementation* [online]. Facebook, Inc. [cit. "21. 5. 2017"]. Dostupné z: <https://thrift.apache.org/static/files/thrift-20070401.pdf>.

[32] NOVAK, J. *Typy datových přenosů, multiplexování, metody řízení přístupu ke sdílenému médiu* [online]. České vysoké učení technické v Praze, Fakulta elektrotechnická. [cit. "21. 5. 2017"]. Dostupné z: <http://measure.fel.cvut.cz/system/files/files/cs/vyuka/predmety/A3B38DSY/7_Multiplex_MAC_cz.pdf>.

[33] POLLOCK, R. *Frictionless Data: making it radically easier to get stuff done with data* [online]. OPEN KNOWLEDGE INTERNATIONAL BLOG, 2013. [cit. "21. 5. 2017"]. Dostupné z: <https://blog.okfn.org/2013/04/24/frictionless-data-making-it-radically-easier-to-get-stuff-done-with-data/>.

[34] PRUNICKI, A. *Apache Thrift* [online]. Software Engineering Tech Trends. [cit. "21. 5. 2017"]. Dostupné z: <http://jnb.ociweb.com/jnb/jnbJun2009.html>.

[35] SAINT-ANDRE, P. *Extensible Messaging and Presence Protocol (XMPP), RFC 6120* [online]. IETF. [cit. "21. 5. 2017"]. Dostupné z: <https://tools.ietf.org/html/rfc6120>.

[36] SATRAPA, P. *Jak funguje nový protokol HTTP/2* [online]. Root.cz, 2015. [cit. "21. 5. 2017"]. Dostupné z: <https://www.root.cz/clanky/jak-funguje-novy-protokol-http-2/>.

[37] TOZON, A. *NTK 2015: Internet of things track (IoT) - Smart Home* [online]. SlideShare. [cit. "21. 5. 2017"]. Dostupné z: <`https://www.slideshare.net/andrejt/ntk-2015-internet-of-things-track-iot-smart-home`>.

[38] VANEK, T. *Úvod do kryptologie, základní pojmy* [online]. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2015. [cit. "21. 5. 2017"].

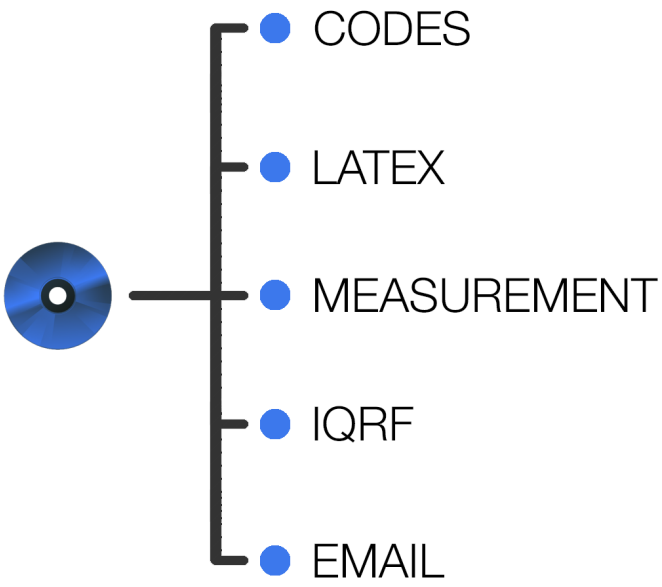
[39] VORISEK, L. *Kde najde IoT největší využití? 10 oblastí, které pravděpodobně ovlivní nejvíce* [online]. CDR server s.r.o., 2016. [cit. "21. 5. 2017"]. Dostupné z: <`http://cdr.cz/sites/default/files/internet-of-things-data-protection-issues.jpg`>.

[40] VORISEK, L. *Kde najde IoT největší využití? 10 oblastí, které pravděpodobně ovlivní nejvíce* [online]. CDR server s.r.o., 2016. [cit. "21. 5. 2017"]. Dostupné z: <`http://cdr.cz/sites/default/files/future-internet-of-things.png`>.

[41] VORISEK, L. *Kde najde IoT největší využití? 10 oblastí, které pravděpodobně ovlivní nejvíce* [online]. CDR server s.r.o., 2016. [cit. "21. 5. 2017"]. Dostupné z: <`http://cdr.cz/sites/default/files/underarmorshirt.jpg`>.

Příloha A

Obsah přiloženého CD



Obrázek A.1: Obsah přiloženého CD

Příloha B

Seznam použitých zkratek

- ACK - Acknowledgement
- AES - Advanced Encryption Standard
- API - Application Programming Interface
- APS - Application Support Sublayer
- BPSK - Binary-Phase Shift Keying
- CA - Certificate Authority
- CPU - Central Processing Unit
- CSMA/CA - Carrier Sense Multiple Access with Collision Avoidance
- CSMA/CD - Carrier Sense Multiple Access with Collision Detection
- DBPSK - Differential Binary Phase Shift Keying
- EPC - Electronic Product Code
- GFSK - Gaussian frequency-shift keying
- GIS - Geographic Information System
- GPS - Global Positioning System
- GSM - Global System for Mobile Communications
- HTTP - Hypertext Transfer Protocol
- IM - Instant Messaging
- IoT - Internet of Things
- IP - Internet Protocol
- MAC - Media Access Control Address

- NFC - Near Field Communication
- ONS - Object Name Service
- OS - Operating System
- P2P - Peer-to-peer
- PAN - Personal Area Network
- PH - Potential of Hydrogen
- PSK - Pre-shared Key
- RAM - Random Access Memory
- RFC - Request For Comments
- RFID - Radio Frequency Identification
- RPK - Rapid Public Key
- RS-232 - Recommended Standard 232
- RX - Receiver
- SHA1/2 - Secure Hash Algorithm
- SPOF - Single Point of Failure
- SSD - Solid State Drive
- TCP - Transmission Control Protocol
- TX - Transmitter
- UDP - User Datagram Protocol
- url - Uniform Resource Locator
- USB - Universal Serial Bus
- W3C - The World Wide Web Consortium
- wifi - Wireless Fidelity
- WWW - World Wide Web
- ZDO - ZigBee Device Object

Příloha C

Seznam použitých pojmů

- agregace - parametr určující metody kombinování síťových připojení pro účely zvýšení datové propustnosti a spolehlivosti připojení
- autenticita - ověření vlastností subjektu
- bit - základní a současně nejmenší jednotkou dat, používanou především v číslicové a výpočetní technice a v teorii informace
- byte - jednotka dat, která je dlouhá 8 bitů a v informatice nejčastěji označuje znak, číslo nebo symbol
- datový model - struktura dat v daném IS jako celku
- dedikovaný server - Pronájem hardwaru, který je umístěn do serverovny poskytovatele. Pro server je poskytnuto zálohované napájení a připojení do sítě Internetu. Server zůstává ve vlastnictví poskytovatele.
- deserializace - opačný proces serializace
- field gateway - Agregační bod nebo sběrné místo dat pro senzory v dané lokalitě.
- human-readable - eprezentace dat v informatice, která mohou být čitelná člověkem
- identifikace - porovnání nezaměnitelných charakteristik předmětů s následným určením nebo vyloučením shodnosti
- Jabber - Jabber je otevřený protokol pro instant messengery – programy, které umí zprostředkovat online komunikaci mezi uživatel.
- key-value - způsob ukládání dat do databáze označovaná jako slovník
- modulace - nelineární proces, kterým se mění charakter vhodného nosného signálu pomocí modulujícího signálu
- non-human-readable - eprezentace dat v informatice v binární podobě, která nelze číst člověkem bez převodu do čitelné verze

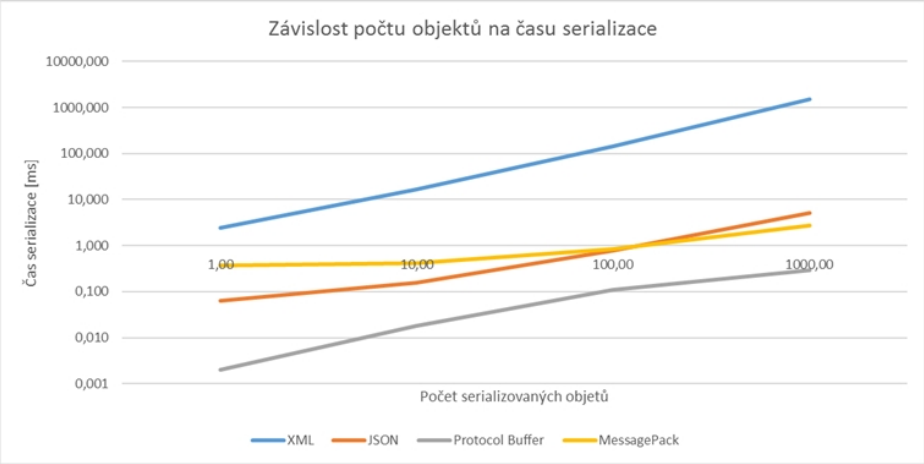
- paralelní přenos - proces přenosu dat, kdy je několik bitů posíláno najednou
- port - číslo v rozsahuj 0 až 65535, které slouží v počítačových sítích při komunikaci pomocí protokolů TCP a UDP k rozlišení aplikace v rámci počítače
- receiver - prvek, který převádí přijatá data ze sítě a převádí je do použitelné podoby
- sémantika - Obor zabývající se důsledným matematickým popisem významu programovacího jazyka.
- serializace - způsob uchování stavu objektu v podobě proudu bytů a následné uložení do souboru, databáze nebo paměti
- seriový přenos - proces přenosu dat postupně po jednotlivých bitech pomocí komunikačního kanálu nebo sběrnice
- server - obecné označení pro počítač, který poskytuje nějaké služby, nebo počítačový program, který tyto služby realizuje
- static-typing - způsob programování, u kterého je typ proměnných znám při kompilaci kódu
- tranceiver - prvek, který převádí data z jedné sítě do jiné sítě

Příloha D

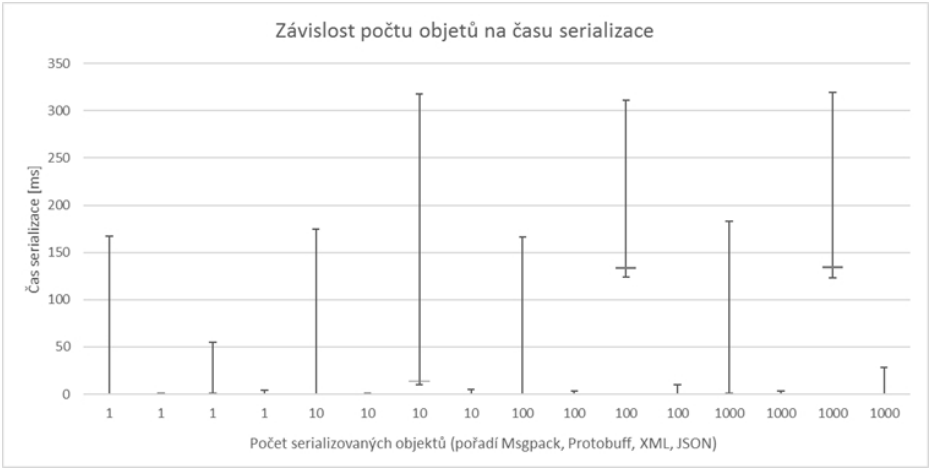
Výsledky měření - grafy

D.1 Serializace a deserializace dat v jazyce Java

D.1.1 Závislost počtu objektů na času serializace

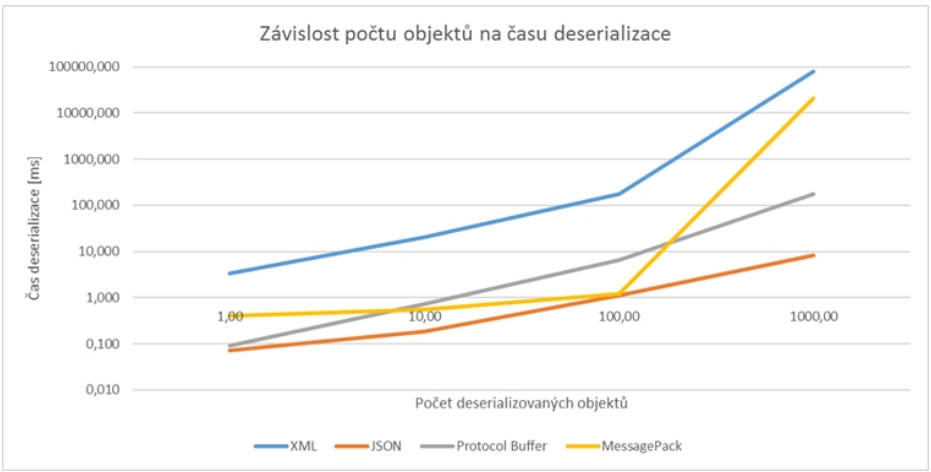


Obrázek D.1: Závislost počtu objektů na času serializace - spojnicový graf

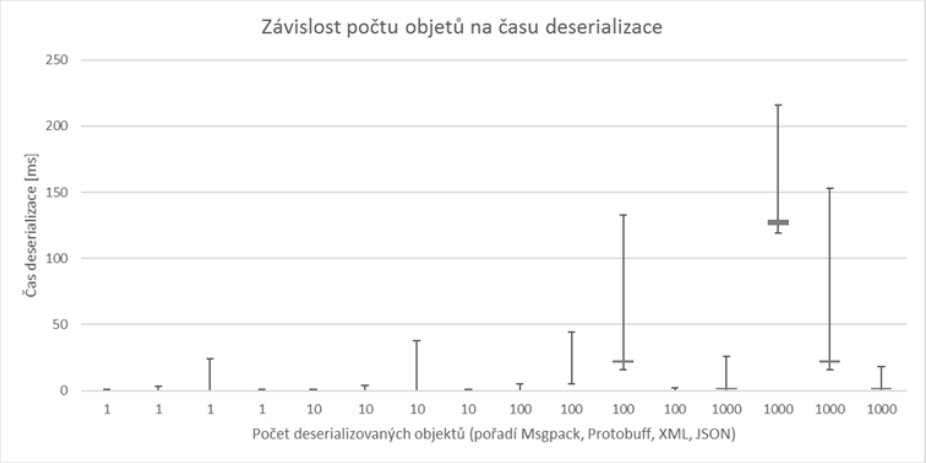


Obrázek D.2: Závislost počtu objektů na času serializace - krabicový graf

D.1.2 Závislost počtu objektů na času deserializace

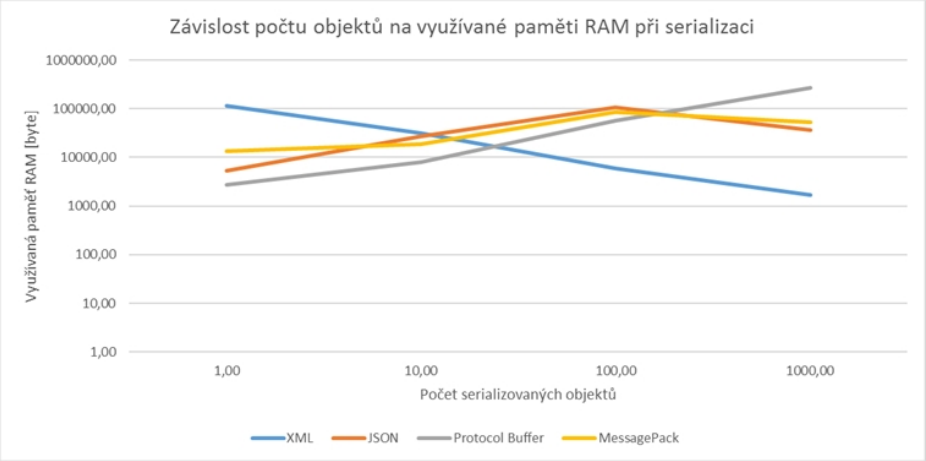


Obrázek D.3: Závislost počtu objektů na času deserializace - spojnicový graf

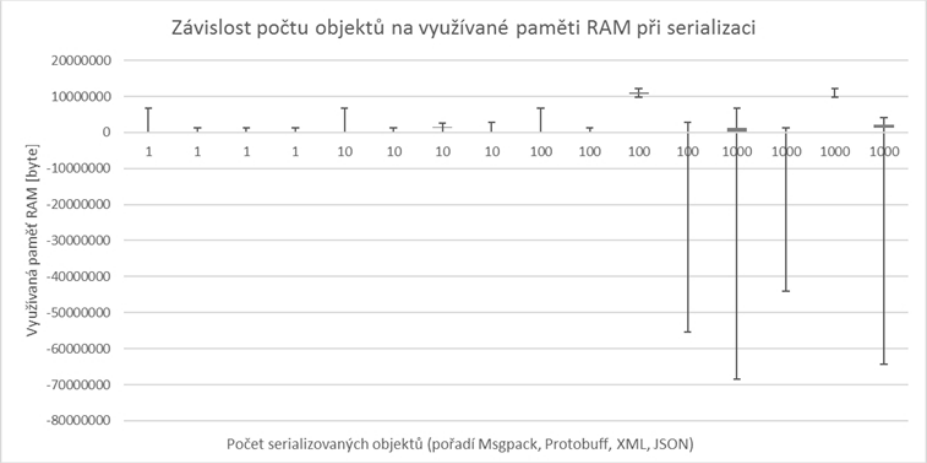


Obrázek D.4: Závislost počtu objektů na času deserializace - krabicový graf

D.1.3 Závislost počtu objektů na využívané paměti RAM při serializaci

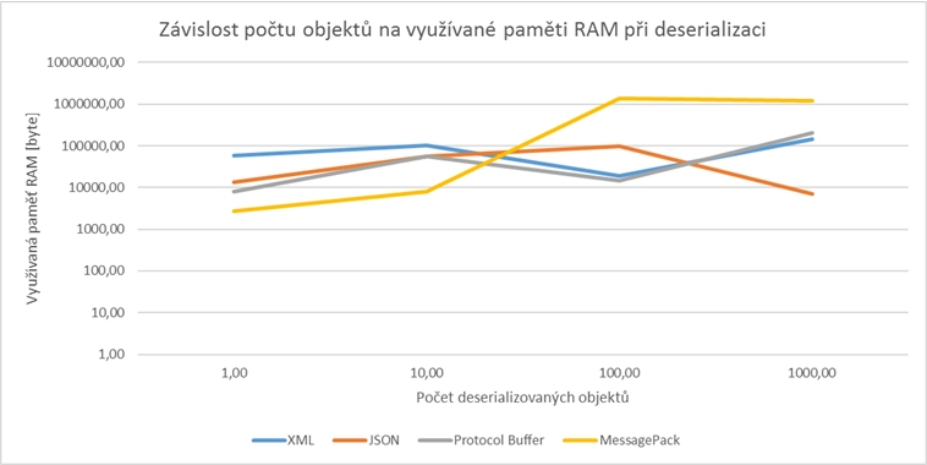


Obrázek D.5: Závislost počtu objektů na využívané paměti RAM při serializaci - spojnicový graf

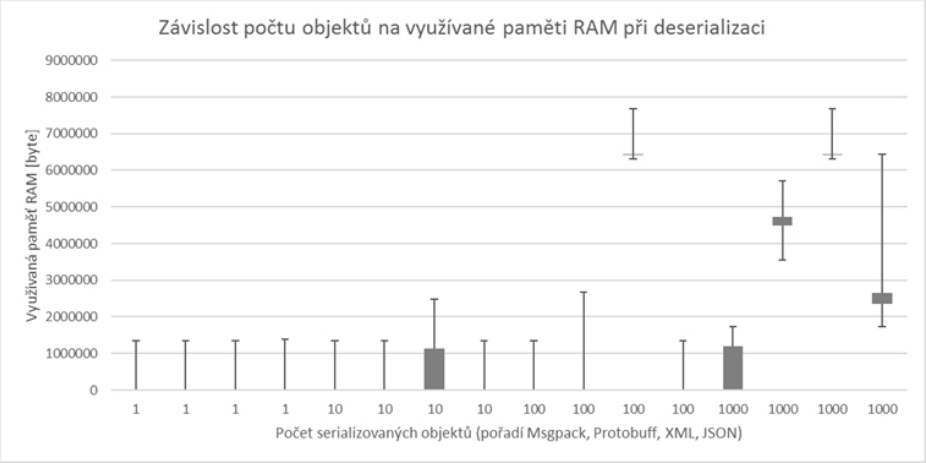


Obrázek D.6: Závislost počtu objektů na využívané paměti RAM při serializaci - krabicový graf

D.1.4 Závislost počtu objektů na využívané paměti RAM při deserializaci

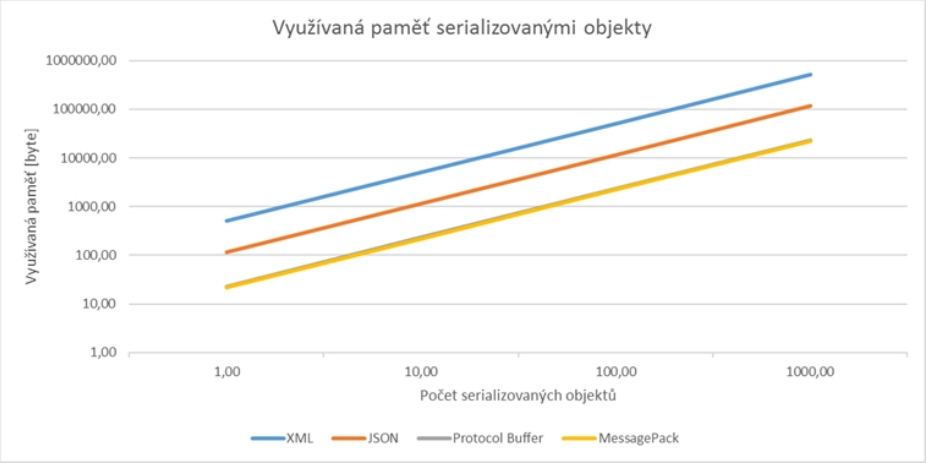


Obrázek D.7: Závislost počtu objektů na využívané paměti RAM při deserializaci - spojnicový graf



Obrázek D.8: Závislost počtu objektů na využívané paměti RAM při deserializaci - krabicový graf

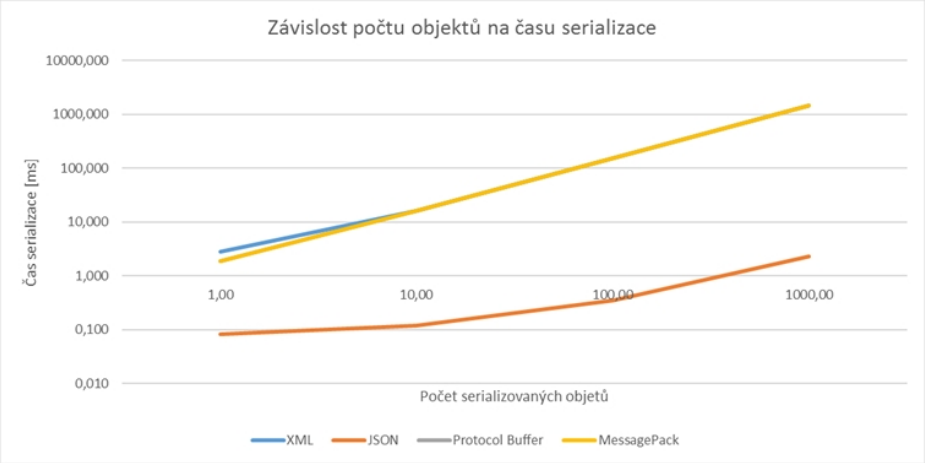
D.1.5 Závislost počtu objektů na velikost využívané paměti



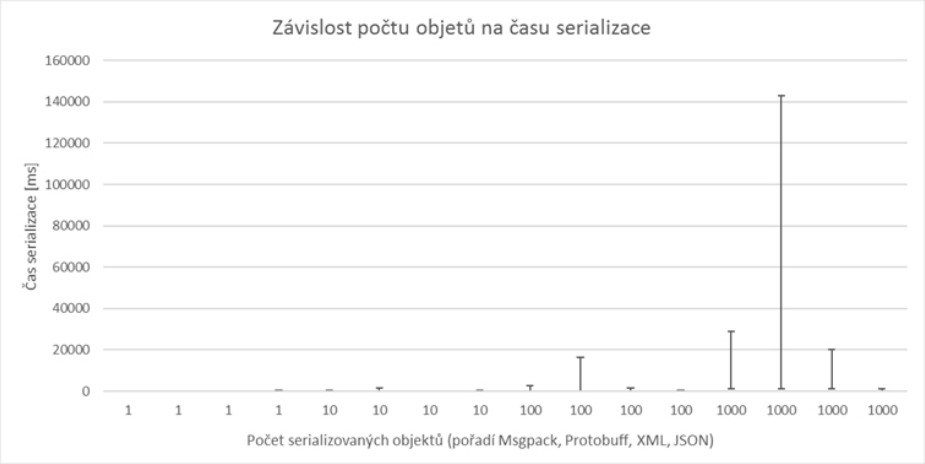
Obrázek D.9: Využívaná paměť serializovanými objekty - spojnicový graf

D.2 Serializace a deserializace dat v jazyce C#

D.2.1 Závislost počtu objektů na času serializace

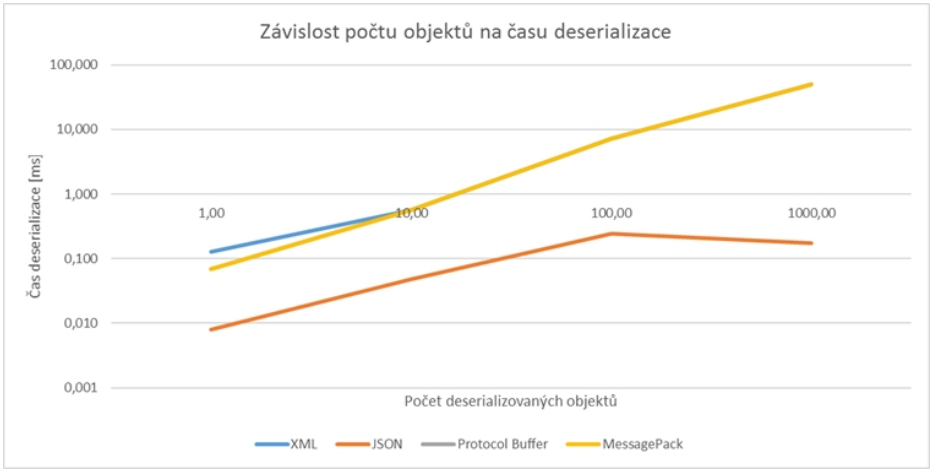


Obrázek D.10: Závislost počtu objektů na času serializace - spojnicový graf

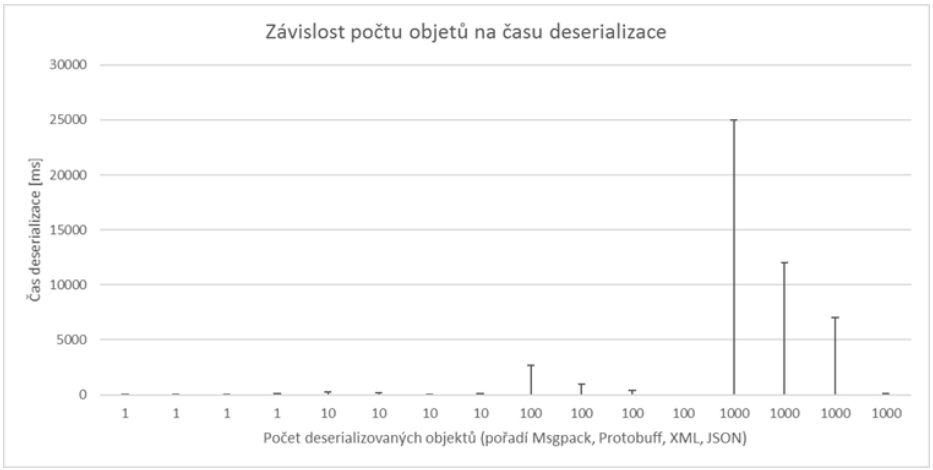


Obrázek D.11: Závislost počtu objektů na času serializace - krabicový graf

D.2.2 Závislost počtu objektů na času deserializace

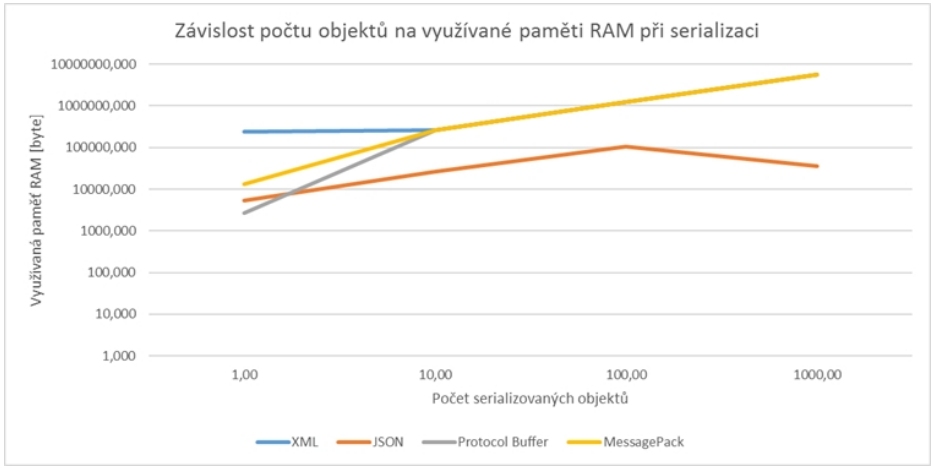


Obrázek D.12: Závislost počtu objektů na času deserializace - spojnicový graf

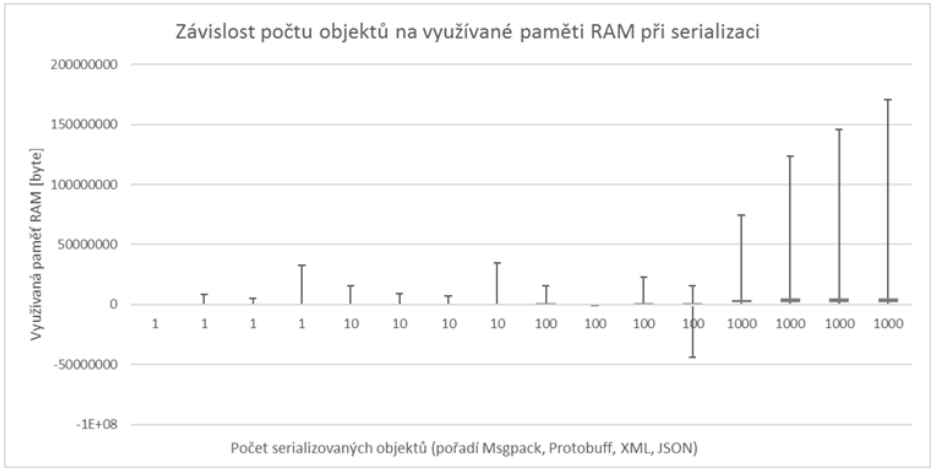


Obrázek D.13: Závislost počtu objektů na času deserializace - krabicový graf

D.2.3 Závislost počtu objektů na využívané paměti RAM při serializaci

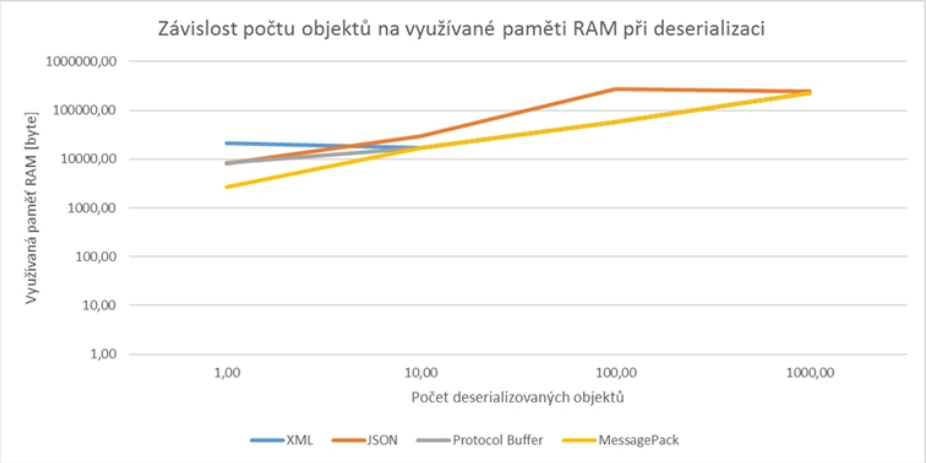


Obrázek D.14: Závislost počtu objektů na využívané paměti RAM při serializaci - spojnicový graf

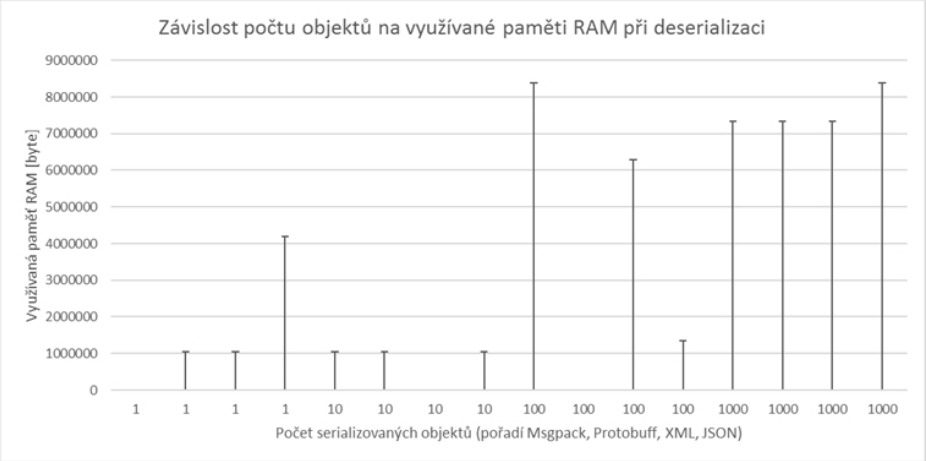


Obrázek D.15: Závislost počtu objektů na využívané paměti RAM při serializaci - krabicový graf

D.2.4 Závislost počtu objektů na využívané paměti RAM při deserializaci

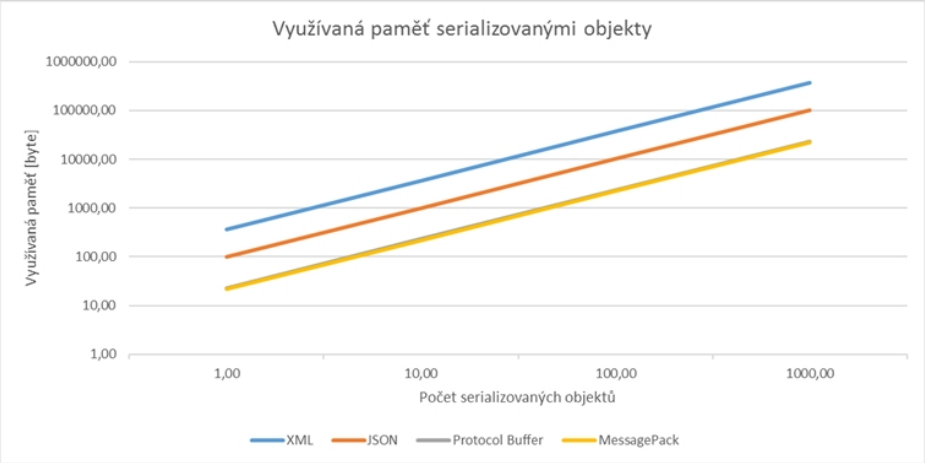


Obrázek D.16: Závislost počtu objektů na využívané paměti RAM při deserializaci - spojnicový graf



Obrázek D.17: Závislost počtu objektů na využívané paměti RAM při deserializaci - krabicový graf

D.2.5 Závislost počtu objektů na velikost využívané paměti



Obrázek D.18: Využívaná paměť serializovanými objekty - spojnicový graf

Příloha E

Přehled používaných senzorů a měřených veličin

E.1 Hlídání objektu

- pohybové čidlo
- CMOS senzor
- reflexní optické senzory

E.2 Zdravotní stav skotu

- snímač srdečního tepu
- GPS

E.3 Měření vlastností půdy (on-the-go senzory)

- pH glass electrode
- senzor vlhkosti půdy
- elektrické a elektromagnetické senzory (elektrický odpor, kapacita, vodivost půdy)
- optické a radiometrické senzory (elektromagnetické vlny)
- mechanické senzory (tahová síla)
- akustické senzory zvuky (zvuky při práci nářadí v půdě)
- pneumatické senzory (schopnost vzduchu pronikat do půdy)
- elektrochemická čidla (koncentrace vodíku, draslíku, dusíku apod.)

E.4 Lokální meteostanice

- rychlost větru
- směr větru
- teplotní čidlo
- senzor vlhkosti vzduchu
- snímač hladiny - množství srážek
- měření množství sněhu
- barometr

E.5 Sledování průtoků a znečištění vodních toků

- senzor průtoku vody
- senzory přítomnosti cizích látek (toxické látky, tuky a oleje, dusík, fosfor, pesticidy, rtuť)
- průhlednost vody
- tvrdost
- vodivost
- salinita
- rezistivita
- obsah kyslíku
- dusičnany, fosfáty, amonium

E.6 Monitoring zemědělských strojů

- senzor polohy GPS

E.7 Smart city - infrastruktura

- řízení pouličního osvětlení - senzor intenzity osvětlení
- zabezpečení objektů - optické závory, PIR detektor, magnetický detektor, detektor náklonu nebo otřesu
- energetický management - teplotní senzor, regulátor topení

- kontrola vibrací budovy a konstrukcí - vibrační senzor, senzor otřesu
- inteligentní budovy - senzory pro měření vlhkosti, kvality vzduchu, teploty a průtoku vody, senzor tlaku, detektory kouře, senzor intenzity osvětlení, pohybový senzor
- inteligentní řízení dopravy - indukční smyčky, pneumatické senzory, piezoelektrické detektory, magnetometry, ultrazvukové detektory, mikrovlnné detektory, pasivní/aktivní infračervené detektory, optické detektory, videodetekce
- sdílení dopravních prostředků - magnetické senzory, bezpečnostní senzory, GPS
- řízení volných parkovacích míst - ultrazvukové senzory, pasivní infračervené senzory (PIR)
- použití pro MHD (vytíženost vozidel, přesnější jízdní řády) - GPS, infračervený senzor
- sledování znečistění ovzduší ve městě - elektrochemické senzory
- monitoring toku řek - rychlost průtoku, kvalita vody (kyslík, pH, teplota, vodivost, zákla, amoniak, draslík, dusičnany, chloridy, ...)
- kontrola městské zeleně - senzor vlhkosti půdy
- optimalizace využití energií - senzor osvětlení, senzor CO2, GPS

E.8 Doprava a logistika

- plynulost dopravy - RFID, inductive loop, CMOS,
- real-time zásilky na mapovém podkladu - GPS
- senzory pro zabránění řízení pod vlivem alkoholu nebo proti mikrospánku - alkohol tester, gyroskopický senzor
- automatické přivolání pomoci či lokace a rozsah škod - GPS, senzor nárazu
- rozšířená virtuální realita ve smartglasses - pohybové senzory, akcelerometr, gyroskop

E.9 Zdravotnictví

- tlakoměr
- senzory pro EKG
- okysličení krve (senzor kyslíku v krvi)
- senzor teploty

E.10 Nositelná elektronika

- srdeční tep
- tělesný tlak
- GPS
- krokoměr
- magnetometr
- světelný senzor
- CMOS čidla
- akcelerometr
- gyroskop

E.11 Inteligentní budovy

- kamerový a zabezpečovací systém - PIR senzory, optická závora, senzor hluku
- topení, klimatizace - senzor teploty a vlhkosti, rychlost proudění vzduchu, otáčky ventilátoru, tlak
- vyhřívání a čištění bazénu - teplota vody, pH vody, cirkulace vody,
- ovládání osvětlení - senzor stmívání
- zavlažování zahrady - vlhkost půdy
- multimediální zařízení - senzor hluku, pohybový senzor
- úspora energií - senzor spotřeby vody, plynu, elektrické energie