ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ
KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

# VYUŽITÍ TECHNOLOGIE IOT

DIPLOMOVÁ PRÁCE

**Bc. Aleš Bubílek**

Vedoucí práce: doc. Ing. Jiří Masopust, CSc.

**Plzeň 2018**

Prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni, 1. června 2018

.....

vlastnoruční podpis

# PODĚKOVÁNÍ

Rád bych touto cestou poděkoval vedoucímu diplomové práce panu doc. Ing. Jiřímu Masopustovi, CSc. za cenné rady a vstřícné vedení mé diplomové práce. Rovněž bych chtěl poděkovat rodinně a přátelům za podporu, trpělivost a motivaci.

Tato práce vznikla za podpory projektu SGS-2018-001.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
**Fakulta elektrotechnická**
Akademický rok: **2017/2018**

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Bc. Aleš BUBÍLEK** |
| Osobní číslo: | **E16N0064P** |
| Studijní program: | **N2612 Elektrotechnika a informatika** |
| Studijní obor: | **Telekomunikační a multimediální systémy** |
| Název tématu: | **Využití technologie IoT** |
| Zadávající katedra: | **Katedra aplikované elektroniky a telekomunikací** |

Zásady pro vypracování:

1. Seznamte se s technologiemi IoT.
2. Popište používaná řešení a nasazené technologie IoT.
3. Zvolte vhodnou technologii IoT pro realizaci pilotní aplikace.
4. Navrhněte vhodnou aplikaci a tuto aplikaci realizujte.
5. Ověřte parametry.

| | |
|---|---|
| Rozsah grafických prací: | **podle doporučení vedoucího** |
| Rozsah kvalifikační práce: | **40 - 60 stran** |
| Forma zpracování diplomové práce: | **tištěná/elektronická** |

Seznam odborné literatury:

**Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.**

| | |
|---|---|
| Vedoucí diplomové práce: | **Doc. Ing. Jiří Masopust, CSc.**<br>Katedra aplikované elektroniky a telekomunikací |
| Datum zadání diplomové práce: | **10. října 2017** |
| Termín odevzdání diplomové práce: | **24. května 2018** |

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 10. října 2017

## ABSTRAKT

Cílem diplomové práce je kromě rešerše používaných řešení a technologií nasazených v IoT také navržení a realizace minimálně jedné aplikace přednostně pro technologie LPWAN. V teoretické části se zabývám technologií IoT a jejími elementy, včetně historie, vývoje a predikce. Popisuji bezdrátové technologie včetně jejich standardů, zabezpečení s následným srovnáním. Cílem v praktické části bylo primární využití sítě Sigfox jak pro aplikace pouhého přenosu informace, tak pro kompletní řešení lokalizace polohy. V práci se teoreticky i prakticky zabývám na několika příkladech využití senzorů pro dálkový přenos informace s modulem FiPy od firmy Pycom a Raspberry PI 3+ s deskou Sense Hat v souvislosti s IoT. Výsledkem diplomové práce je v teoretické rovině rešerše, v praktické rovině alespoň jedna aplikace cílící na bezdrátový přenos informace, s jejíž pomocí spadají věci do kategorie SMART.



## Klíčová slova

Internet věcí, Sigfox, Raspberry PI, Sense Hat, Pycom, Pytrack, FiPy, WiFi, Wia

## ABSTRACT

The main goal of this diploma thesis is to design and implement at least one application that are suitable for LPWAN technology, in addition to the research of the used solutions and technologies used in IoT. Theoretical part deals with IoT technologies and its elements, including history, development and prediction. Then I explain wireless technologies including their standards, security with subsequent comparison. The aim in the practical part was to use network Sigfox not only for application of information transmission, but for complete position tracking solution. In my theoretical and practical parts, I deal with, among other things, several examples of use of remote sensing sensors with FiPy modul from Pycom company in conjunction with IoT. The result of the diploma thesis is a research in the theoretical part, the goal in the practical part is at least one application designed for wireless information transmission, which are used by things in the SMART category.

## Key words

Internet of Things, Sigfox, Raspberry PI, Sense Hat, Pycom, Pytrack, FiPy, WiFi, Wia

# OBSAH

# ÚVOD

Internet věcí je fenoménem dnešní doby. Tento pojem se v současnosti velmi často objevuje ve světě informačních a komunikačních technologií.

Myšlenka propojení věcí, čidel a senzorů je již poměrně stará, ale až nyní zažívá svůj skutečný průlom v podobě internetu věcí (většinou označovaný jako IoT, z anglického Internet of Things). Podle předpovědí odborníku má internet věcí budoucnost. Má potenciál přetransformovat náš svět do něčeho, co jsme schopni si nyní jen stěží představit. Jeho základem je připojování „věcí“ do internetu. Co můžeme považovat za tyto věci, uvedu dále ve své práci. Internet věcí se stává velmi zajímavou oblastí plnou příležitostí pro firmy i pro jednotlivce. Díky tomu se o něj začíná v poslední době zajímat čím dál tím více lidí. Pochopit však, o čem internet věcí je a jakým způsobem na něj nahlížet, se zdá stále obtížné. Problematika internetu věcí je aktuální a velmi rozsáhlá.

Cílem práce je představení technologie jako takové včetně různých odvětví stejně jako praktické ukázky. V první části je proveden teoretický rozbor internetu věcí a záležitosti s ním spojené včetně popsání technických aspektů dvou hlavních sítí k tomu určených. Ve druhé části je popsána realizace několika hotových aplikací na dvou platformách sloužících jako možné řešení v praxi.

# 1 TECHNOLOGIE IOT

Termín internet věcí je z technického hlediska obrovským příslibem budoucnosti. Ačkoli je technologie IoT v podstatě stále v počátku, je počítáno s nesmírnou expanzí v nejbližších letech. Již nyní se můžeme setkat dennodenně s touto technologií v běžném životě a budeme se s ní setkávat stále častěji, budeme na ní více závislí a více jí budeme vyžadovat. Podstata internetu věcí je propojení fyzického světa, kupříkladu senzoru, se světem virtuálním, kupříkladu cloudem. Propojení těchto dvou světů přináší nové možnosti. Objekty spadající pod IoT mohou na základě nasbíraných dat rozhodovat a provádět činnosti zcela autonomně.

Z důvodu přečtení několika odborných článků a zpráv na téma Internet of Things jsem přesvědčen o neexistenci jediné, všeobecně užívané definice tohoto termínu. Různé společnosti používají odlišné definice pro popis nebo šíření určitého pohledu na to, co IoT reprezentuje a jaké jsou jeho základní charakteristiky.

## 1.1 SPECIFIKACE POJMŮ

V rámci této diplomové práce používám pod pojmem internet věcí (Internet of Things) definici globální síťové infrastruktury propojených objektů (věcí). Tato definice je založena na standardech a interoperabilních protokolech, umožňujících výměnu a sdílení dat.

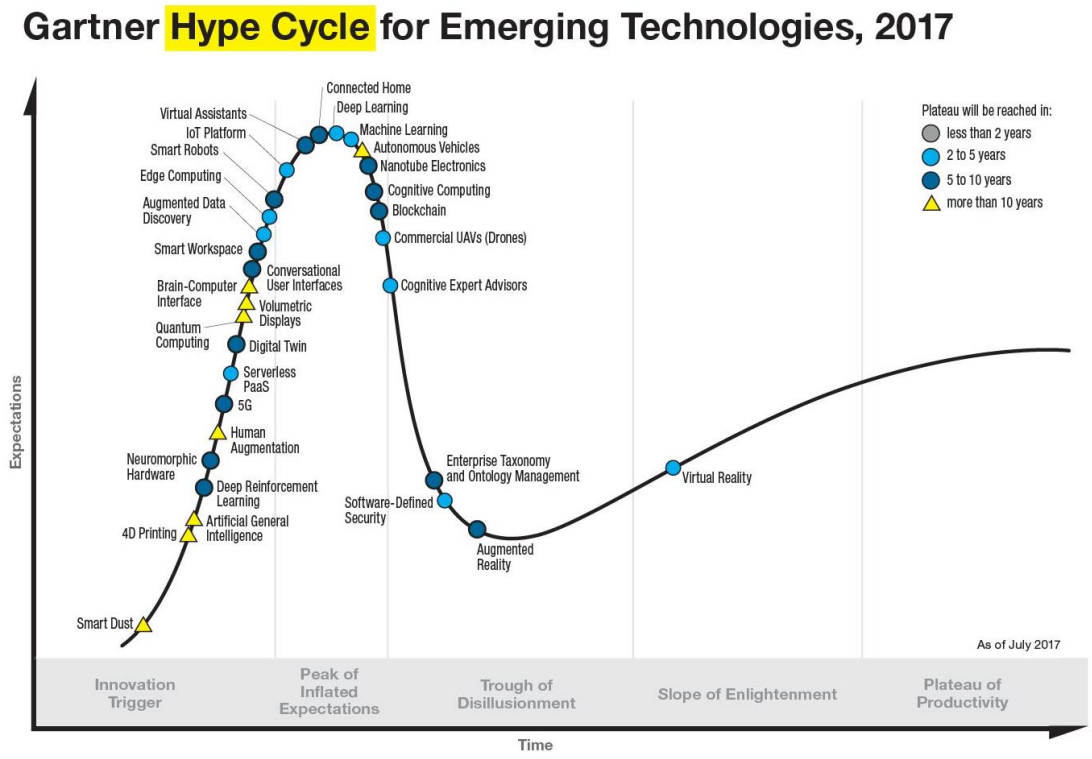
Jelikož je kategorie internetu věcí velice rozsáhlá a existují zde určité rozdíly a nuance, účelově jsem vypsal v dané kapitole základní definice.

Pod pojmem „věcí“ (things) jsou rozuměny jakékoli neživé objekty nebo předměty ať fyzické nebo virtuální. Kupříkladu obyčejná kniha či gramofonová deska jsou příklady fyzických objektů, nicméně elektronická kniha nebo zvukový soubor jsou již objekty virtuálního, či multimediálního světa, jelikož jejich struktura je binární, nikoli založena na fyzické podstatě. Jakákoli věc obsahující elektroniku, software a senzor má unikátní identifikátor a je použitím rozhraní integrována do informační sítě.

Termín „síť“ nemusí znamenat pouze internet, nýbrž může v daném případě jít o lokální síť, v níž mohou věci mezi sebou komunikovat.

Celosvětová analytická společnost Gartner definuje termín internet věcí jako „síť fyzických objektů, obsahující technologie pro komunikace a identifikace nebo schopných měřit parametry vnitřního stavu a stavu vnějšího prostředí.“ [1]

Právě tato společnost od roku 1995 pravidelně zpracovává grafy cyklu dospělosti technologií (tzv. S – křivka, známá též jako hype cycle), kde zachycuje a porovnává vývoj technologií, které se uplatňují na trhu. V posledním vydání z června 2017 zaujala IoT platforma termín 2-5 let, což je oproti loňské verzi s výsledkem 5-10 let značný pokrok. [2]




*OBR. 1: HYPE CYCLE DLE GARTNERA, PŘEVZATO Z [2]*

Pro snazší pochopení pojmu internet věcí lze interpretovat definici společností Accenture a Bankinter Foundation of Innovation. Tyto společnosti ve své publikaci The Internet of Things: In a Connected World of Smart Objects uvádí, že: „Internet věcí se skládá z věcí připojených k internetu kdykoliv a kdekoliv. V technickém smyslu internet věcí začleňuje senzory a zařízení do běžných objektů, které jsou připojeny k internetu přes pevné nebo bezdrátové sítě.“[3]

Jednoznačně definovat termín internetu věcí je velice obtížné, jelikož si jej každá společnost vysvětluje poněkud odlišně, nicméně podstata definic zůstává stejná.

Konceptem internetu věcí je připojení věcí do sítí za účelem vzájemné komunikace mezi sebou i s uživateli. V pozdějších kapitolách popisuji příklady zařízení a možnosti komunikací. Základem internetu věcí nejsou věci jako takové, ale data, která tyto věci poskytují. [4]

## 1.2 HISTORIE INTERNETU VĚCÍ

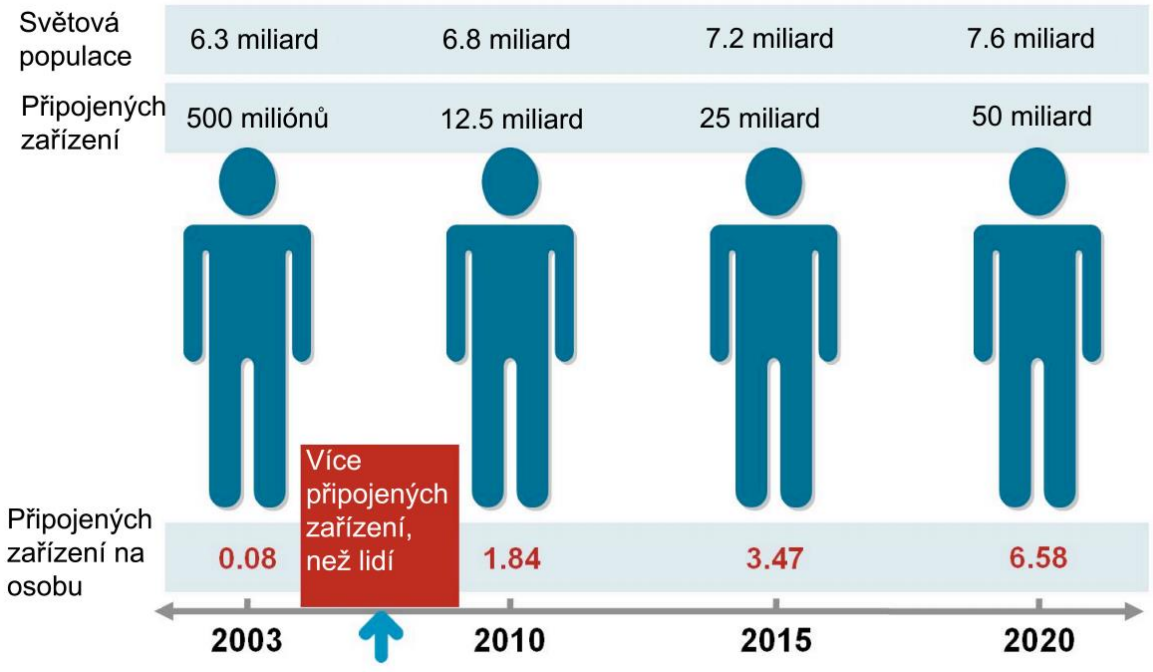
Nikola Tesla pronesl v roce 1926: „Když bude bezeztrátovost perfektně aplikována, celý svět se stane jedním obrovským mozkem, všechny věci se stanou součástí jednoho reálného a rytmického celku. Budeme schopni spolu komunikovat kdykoliv, bez ohledu na vzdálenost. Nejen to, ale prostřednictvím televize a telefonování se budeme vidět a slyšet tak dokonale, jako bychom stáli tváří v tvář, navzdory vzdálenosti tisíců mil, a nástroje, jejichž prostřednictvím bychom toho byli schopni, se budou podobat dnešním telefonům. Člověk bude schopen nosit takový přístroj v kapse vesty.“ [5]

V roce 1989 navrhuje Tim Berners – Lee World Wide Web (WWW) a později roku 1990 je spuštěn první webový server na světě. [6]

Patrně jako první exemplář internetu věci provedl jeden ze zakladatelů protokolů TCP/IP John Romkey v roce 1990, když připojil k síti svůj toastovač. [7]

Myšlenka sítě inteligentních zařízení byla v moderní době poprvé navrhnuta a zpopularizována Kevinem Ashtonem v roce 1999, který přišel s pojmem “Internet věcí”.

Vznik internetu věcí jako takového se datuje mezi roky 2008 a 2009, kdy společnost Cisco odhadla překročení počtu zařízení připojených k internetu počet světové populace, viz obrázek. [8]




*OBR. 2: POČET PŘIPOJENÝCH ZAŘÍZENÍ V ČASE, PŘEVZATO Z [8]*

Velká medializace internetu věcí přišla v roce 2014. Již v roce 2015 bylo údajně připojeno do sítě internetu 4,9 miliard zařízení. Analytici již zmíněné společnosti Gartner předpokládají každodenní nárůst zhruba 5,5 milionu nových zařízení. V roce 2020 odhadují počet připojených zařízení na 20,8 miliard. [9]

Společnost ABI Research uvádí, že v roce 2020 bude počet zařízení dokonce 40,9 miliard. [10]

Cisco předpovídá dokonce až 50 miliard připojených zařízení. [11]

Odhady se mezi jednotlivými společnostmi liší, což je zapříčiněno neurčitostí týkající se toho, jaká zařízení se mezi zařízení internetu věcí řadí. Tento fakt je potvrzen i jednou z těchto společností, konkrétně společností Gartner, která na svých internetových stránkách uvádí, že v jejich statistikách nejsou započteny chytré telefony, tablety ani počítače. [9]

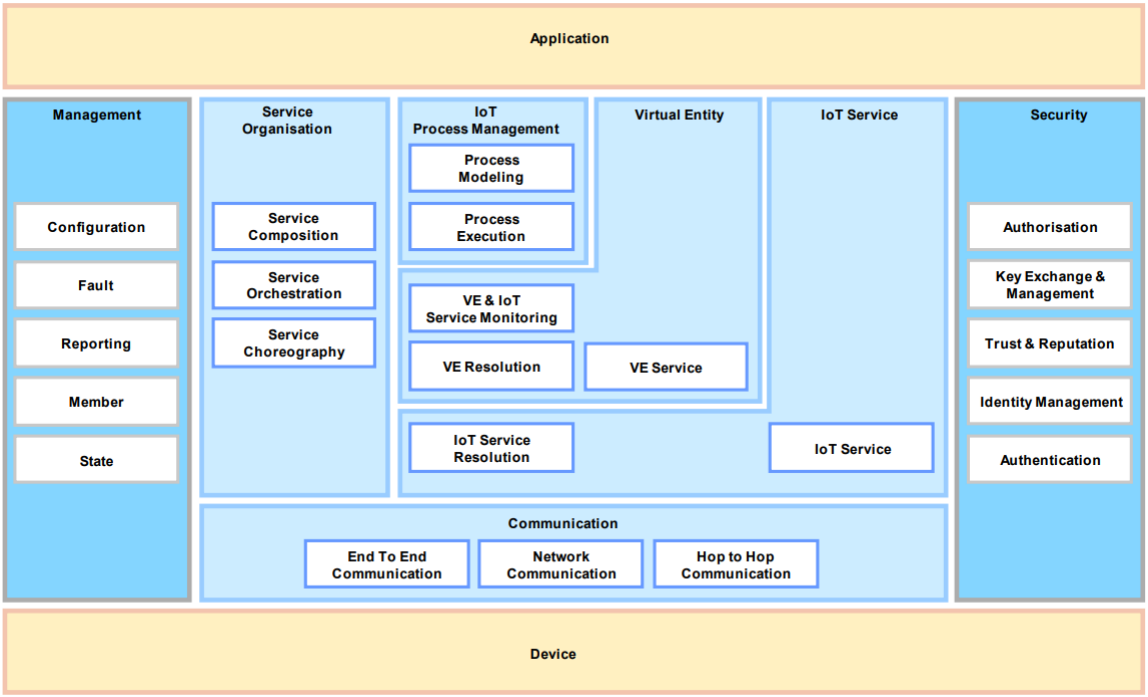
## 1.3 ARCHITEKTURA IOT

U technologií IoT neexistuje jedna univerzální architektura. Vzhledem k otevřenosti a nekompletní definici si mnoho výzkumníků, vývojářů a praktiků vytvořilo vlastní IoT architekturu. V praxi se proto můžeme setkat s několika různorodými architekturami.

Všeobecná představa fungování IoT ale staví na tom, že ekosystém musí obsahovat řadu senzorů, komunikační síť a výpočetní části. Architektura, která byla navržena mezinárodní telekomunikační unií (ITU) [12] obsahuje 5 částí:

- snímanou vrstvu,
- přístupovou vrstvu,
- síťovou vrstvu,
- middleware vrstvu (vrstva umožňující komunikaci a správu dat pro distribuované aplikace),
- aplikační vrstvu

Krom tohoto návrhu vznikl v rámci Evropského výzkumného projektu FP7 IoT-A druhý návrh architektury. Velká část úsilí zde byla věnována právě vytvoření referenčního modelu. Výsledkem je obecný referenční model pod názvem ARM. Jedná se o komplexní návrh, který může být použit k odvození konkrétních architektur.




*OBR. 3: FUNKČNÍ POHLED NA REFERENČNÍ IOT ARCHITEKTURU, NAVRŽENOU VE VÝZKUMNÉM PROJEKTU FP7 IOT-A. PŘEVZATO Z [12]*

V projektu bylo stanoveno, že při návrhu referenční IoT architektury je třeba nejprve odpovědět na širokou škálu otázek z různých oblastí. Odpovědi byly podrobně rozepsány pro tyto oblasti:

- Funkční prvky
- Interakce mezi prvky
- Řízení informací
- Provozní funkce
- Nasazení systému

## 1.4 DATOVÁ KOMUNIKACE

Datová komunikace je samotný přenos digitálního signálu skrze přenosové médium. V IoT využíváme v drtivé většině jako přenosové médium elektromagnetické vlnění. Komunikační rozhraní je rozděleno na vysílač a přijímač.

V současné době využíváme tří základní typy připojení: [13]

- **M2M – Machine to Machine** – je typ komunikace mezi přístroji, která umožňuje vzájemné sdílení dat bez účasti člověka. Tato technologie je rozšířena především v průmyslové automatizaci a logistice. Také se využívá v oblasti dopravy pro různé diagnostiky.
- **M2P – Machine to People** – je komunikace mezi člověkem a strojem, kdy stroj předává generovanou informaci jednotlivým osobám.
- **P2P – Person to Person** – je přímá komunikace mezi lidmi. Ať již přes sms zprávy, mobilním voláním či komunikací přes sociální sítě.

### 1.4.1 TYPY PROPOJENÍ INTERNETU VĚCÍ

V roce 2015 byl vydán Komisí internetové architektury (IAB) směrný dokument popisující architekturu síťového propojení objektů, který definuje čtyři způsoby propojení mezi IoT prvky systému. [14]

- Device to device
- Device to cloud
- Device to gateway
- Back-end data-sharing

Každý model má svá specifika a podmínky použití.

### 1.4.2 MEZI ZAŘÍZENÍMI NAVZÁJEM

Jde o typ komunikace mezi dvěma a více zařízeními, které komunikují přímo mezi sebou, nikoli prostřednictvím mezilehlého aplikačního serveru. Komunikace může probíhat skrze různé druhy sítí. Nejčastěji se využívají protokoly Z-Wave, ZigBee, Bluetooth LE a také IP sítě a internet. Tento komunikační model bývá nejběžněji uplatňován při automatizaci domácnosti, kde není kladen důraz na vysokou přenosovou rychlost a propustnost sítě. Přenosová rychlost je sice nízká, ale jednotlivá zařízení posílají velice krátké informace, tudíž jsou přenášeny pouze malé datové pakety. Výhodou je nízká spotřeba energie.

Příkladem tohoto modelu může být komunikace mezi nositelnými chytrými hodinkami a mobilním telefonem.

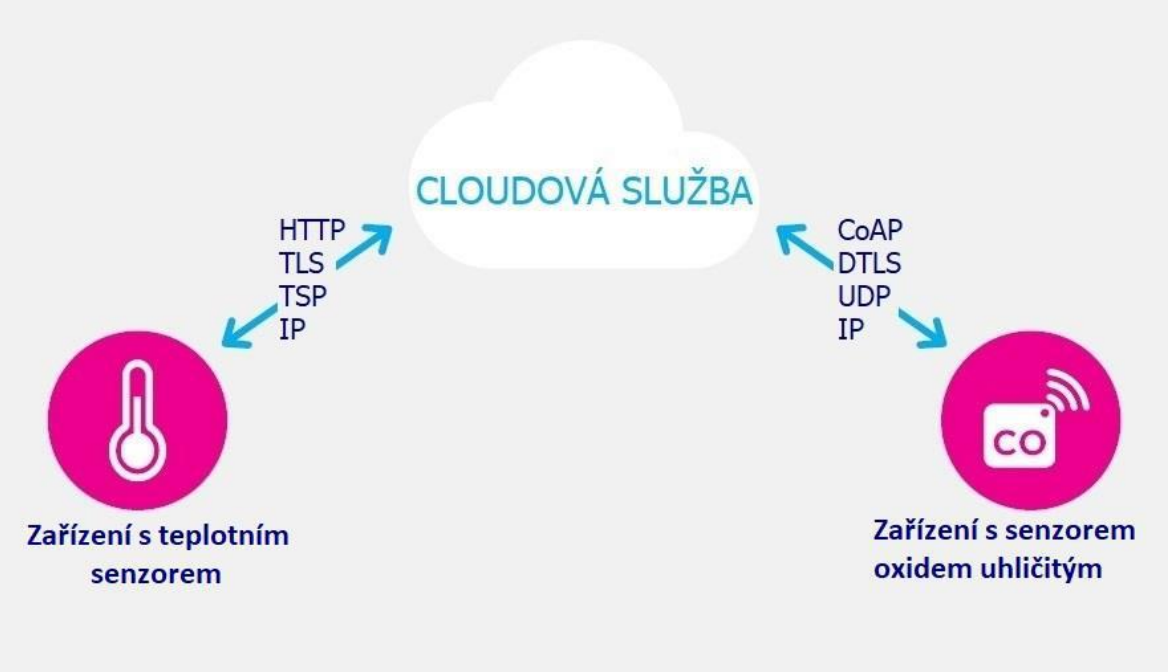

*OBR. 4: MODEL PŘIPOJENÍ MEZI ZAŘÍZENÍMI NAVZÁJEM*

### 1.4.3 OD ZAŘÍZENÍ DO CLOUDU

Zde dochází k připojení zařízení přímo k aplikačním službám konkrétní cloudové služby. Cloudová služba řídí komunikaci a výměnu dat, nebo zajišťuje komunikaci mezi různými systémy. V rámci cloudových systémů lze tyto zařízení vzdáleně ovládat a aktualizovat je.

Jelikož zařízení komunikuje prostřednictvím internetu, využívají se většinou komunikační technologie typu ethernetu, Wi-Fi nebo celulární rádiové sítě. To si žádá větší nároky, když je třeba mít k dispozici dvě pověření, a to pro přístup ke komunikační síti a pro přístup k samotné cloudové službě.

Tento model poskytuje uživateli vzdálený přístup k věcem, může se třeba jednat o komunikaci mezi sledovacím zařízením v automobilu a cloudovou aplikací poskytovatele sledovací služby.

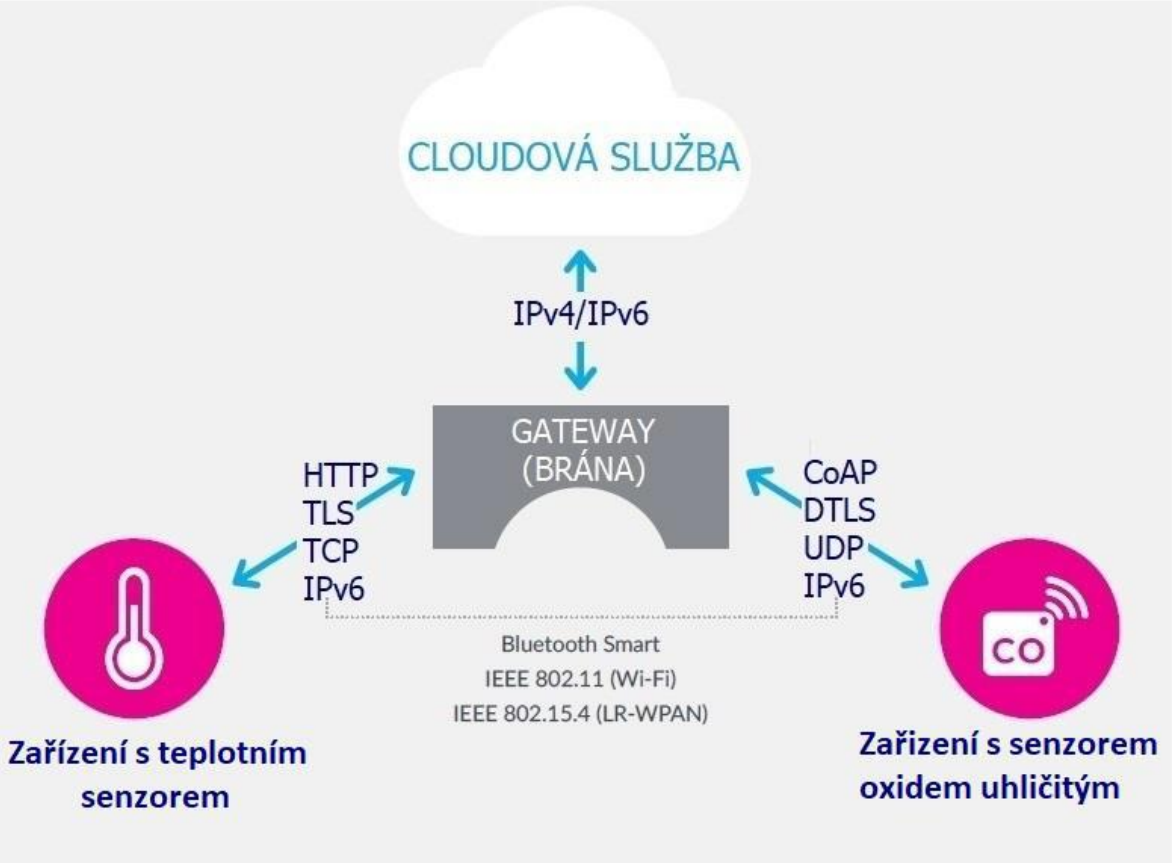



*OBR. 5: MODEL PŘIPOJENÍ OD ZAŘÍZENÍ DO CLOUDU*

### 1.4.4 OD ZAŘÍZENÍ DO BRÁNY

Jedná se o model komunikace využitému v případě podpory komunikačních protokolů jako ZigBee či Bluetooth.

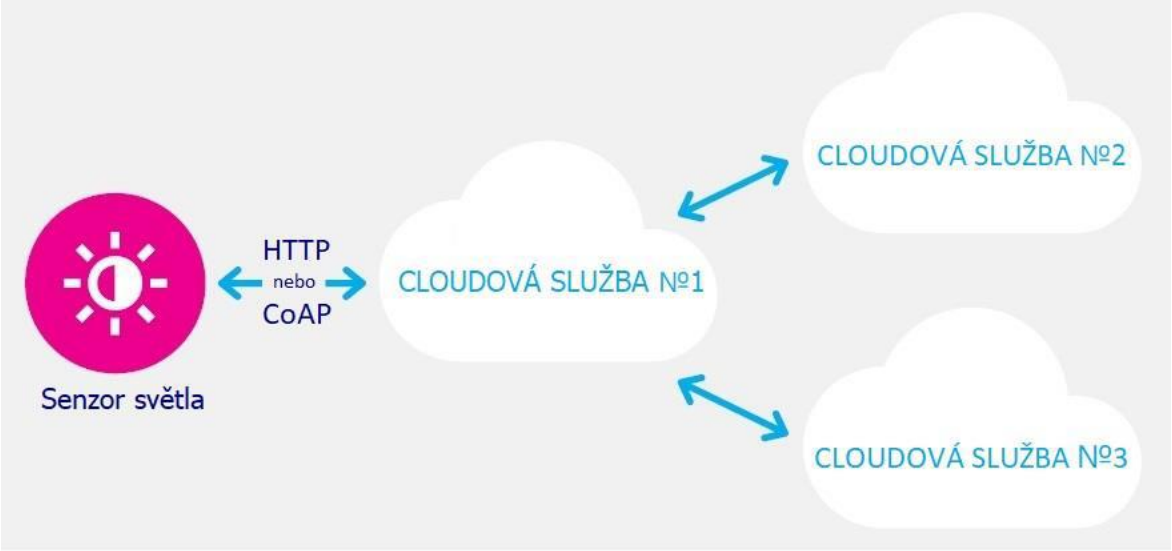
Brány v tomto případě slouží jako prostředek pro spojení lokální sítě se sítí globální. Příkladem využiti tohoto modelu je integrace zařízení podporujících pouze protokol IPv6, poté je brána nutná pro zařízení a služby podporujících pouze protokoly IPv4.




Obr. 6: Model připojení od zařízení do brány

### 1.4.5 MEZI CLOUDY

Jedná se o rozšíření modelu device to cloud o zdokonalené sdílení dat. Data zařízení mohou být přístupná i pro systémy z třetí strany. Uživatelé mohou analyzovat data z několika zařízení, která jsou uložena na různých cloudových službách.




Obr. 7: Model připojení mezi cloudy

## 1.5 POUŽÍVANÁ ŘEŠENÍ A NASAZENÉ TECHNOLOGIE

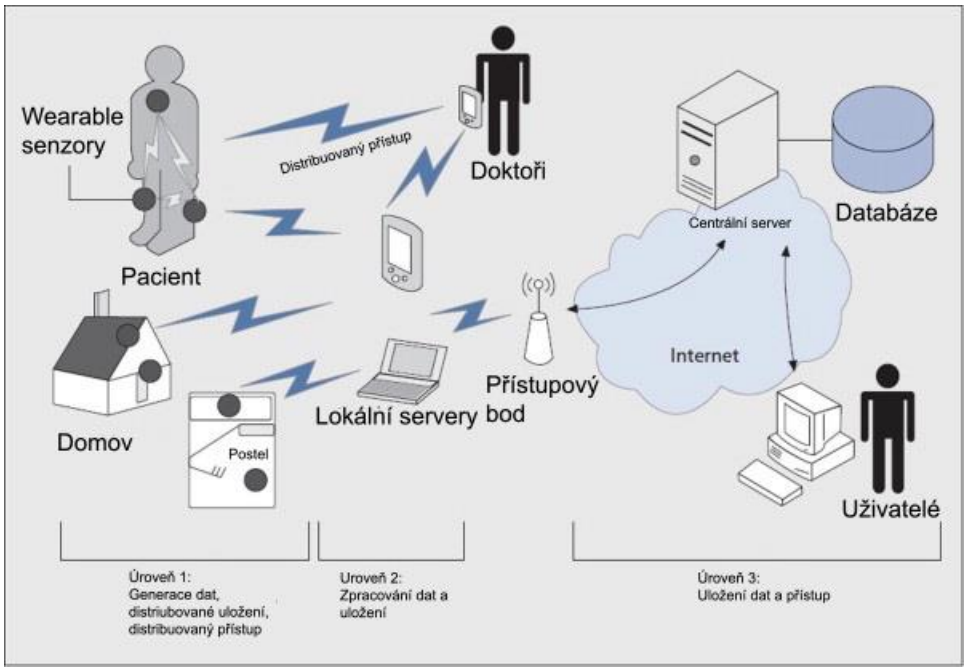
Internet věcí lze rozdělit na dva hlavní segmenty. Prvním segmentem je průmyslový internet věcí, druhým je spotřebitelský internet věcí. [16]

### 1.5.1 PRŮMYSLOVÝ IOT

Zaměřen na usnadnění chodu průmyslového odvětví. Spojuje technologie a procesy z oblastí jako jsou big data, umělá inteligence a M2M komunikace. Mezi odvětví, kde se IoT využívá, se řadí především dopravní průmysl, zdravotnictví a chytrá města.

Zdravotnictví

Ve zdravotnictví se nachází obrovský potenciál internetu věcí. Není to již otázka vzdálené budoucnosti, jelikož již nyní se ve zdravotnictví aplikují zařízení využívající internet věcí. Nemocnice jej využívají ke sledování lékařských přístrojů, personálu i pacientů. Neoddiskutovatelně dochází díky internetu věcí ke zjednodušení zdravotní péče. Zařízení využívající senzory znamenají velký převrat, jelikož díky datům ze senzorů dostaneme téměř okamžitě komplexní přehled o zdravotním stavu pacienta. Díky tomu můžeme včas a přesněji rozpoznat nemoci a začít s léčbou. Za přínos se dá také považovat snížení nákladů. [16]




Obr. 8: Tří úrovňová architektura zdravotní monitorovací sítě, převzato z [20]

Výměna dat mezi senzorem a cloudovým serverem může probíhat přes různé technologie – GPRS, 4G, LPWAN aj. Je třeba uvažovat nad otázkou mobility, datové a energetické náročnosti. Náročnost zdravotní péče a různá technologická omezení představují řadu výzev – nejen pro samotné konstruktéry technologií, ale i pro tvůrce těchto systémů. Tato problematika zasahuje do oblasti:

- politické (např. financování, zákony)
- behaviorální (tj. požadovaná funkčnost)
- fyzické (např. dostupné technologie)
- komunikace (např. dostupné přenosové kanály)
- logiky (např. analytika dat, jazyk, nástroje)
- strukturální (např. designu)
- etnické (např. příslušné zákony daného státu, ochrany soukromí)

Automobilový průmysl

Velké množství změn se zcela jistě bude dít v dohledné době v automobilovém průmyslu. Výrobci automobilů čím dál více směřují k názoru, že nemá smysl vylepšovat automobily rostoucím výkonem, ale mnohem důležitější vlastností se stává konektivita a použití moderních technologií [17]. Již teď některé vozy využívají např. adaptivní tempomat, který sleduje pomocí senzorů vozidlo před ním a automaticky reguluje rychlost.

Budoucností dopravního průmyslu jsou autonomní vozidla. Nejvíce se v současné době angažuje společnost Tesla, která vydala již několikátý model. Vývojem autonomních vozidel se nyní zabývá většina automobilek. Vozidla budou mezi sebou komunikovat, propojení vozů umožní, aby se automobily v dopravě vzájemně sledovaly. Vozidla budou sdílet informace o aktuální poloze, směru i rychlosti. Současně budou automobily komunikovat s chytrými semafory i vozovkami, čímž by se měl zásadně zjednodušit provoz [18].

Energetika

Zavedení technologie IoT v energetickém průmyslu znamená výrazné zvýšení efektivnosti a spolehlivosti odvětví. Dochází ke snížení nákladů jak ze stran výrobců elektřiny, tak i spotřebitelů. Díky dálkovému monitoringu dochází ke zlepšení ovladatelnosti rozvoden, elektrických vedení a jiných elementů sítě, v jejímž důsledku dochází k dalším snížení nákladů.

Na základě platformy Predix od General Electric bylo vytvořeno řešení pro společnost vlastnící více než 80 tisíc km elektrického vedení. Údržba těchto vedení vyžadovala 2 tisíce zaměstnanců, kteří byli nahrazeni drony přenášející data do analytické aplikace [29]. Zavedení těchto systémů umožní podnikům dramatické snížení nákladů.

Obr. 9: Drony pro kontrolu energetických zařízení. Převzato z [19]

Další uplatnění:

[21]

Internet věcí zabývající se životním prostředím se stará o měření koncentrací škodlivých plynů a pevných částic. Také jej můžeme využít při detekci lesních požárů či zemětřesení. V úvahu může připadat i např. měření sněhové pokrývky.

Další model se zabývá měřením, kdy je možné měřit např. průtoky v potrubí, úroveň hladiny v nádržích či energetickou spotřebu a spotřebu vody.

V obchodě je internet věcí využit k managementu dodavatelského řetězce, monitorování skladových zásob, zpracování plateb na základě využívání veřejných dopravních prostředků, kontrola oběhu zboží v regálech a skladech, či automatické doplňování zásob.

V zemědělství lze kontrolovat kvalitu půdy, rostlin, mikroklima ve sklenících, řídit zavlažování. Chov zvířat je zaměřen na monitorování a lokaci chovaných zvířat.

Dalším modelem je bezpečnost osob a majetku. Zjišťovány jsou úrovně radiace, nebezpečných a výbušných plynů. Další možností je kontrola přítomnosti kapalin v důležitých prostorech.

### 1.5.2 SPOTŘEBITELSKÝ IOT

Odvětví IoT zaměřující se na usnadnění každodenního života jednotlivcům. Do tohoto segmentu jsou řazeny mj. chytré domácnosti, nositelná elektronika, chytré zařízení. Za zmínku stojí také chytrý hasičský oblek vyvíjen na ZČU.

Chytré domácnosti

Internet věcí v domácnosti poskytne díky propojení chytrých věcí do společné sítě jednodušší formu bydlení. Systém inteligentní domácnosti typicky řídí elektřinu, osvětlení, vytápění, klimatizaci, bezpečnostní systémy, multimédia (audio a video), dveřní zámky a další. Každý uživatel si může nastavit svou domácnost podle svých potřeb.

Výhody inteligentní domácnosti:

- Snížení spotřeby zdrojů (voda, plyn, elektřina)
- Vysoká úroveň pohodlí
- Snížení pravděpodobnosti výskytu havarijních událostí

Wearables

Nositelná elektronika, tzv. wearables, jsou zařízení připojená k internetu, která může člověk nosit na svém těle. Mezi wearables se řadí chytré hodinky, náramky, brýle atd.

Účelem wearables je osobní užití. Proto tyto produkty nabízejí nejčastěji sledování fyzické aktivity a celkového životního stylu svého nositele. Wearables elektronika se začíná prosazovat i ve zdravotnictví, kde sbírá data o zdravotním stavu svého uživatele a umožňuje tak nepřetržitý monitoring. Toto řešení je pro pacienty pohodlné a dostupné, díky dálkovému monitoringu tak dochází k dalšímu snížení nákladů. [22]

### 1.5.3 NASAZENÉ TECHNOLOGIE

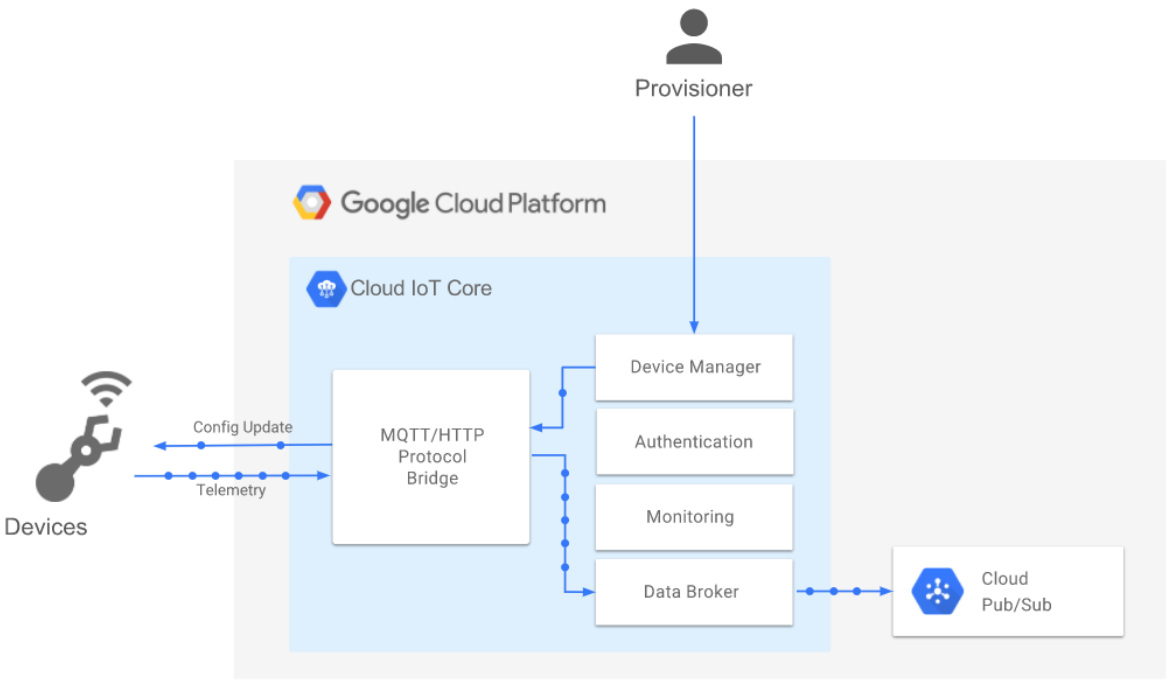
Nejčastěji se IoT systémy v dnešní době využívají v automobilových systémech, inteligentních televizorech a set-top boxech. V průmyslu jde o komerční kamery a inteligentní měřící nástroje jako elektroměry či určování polohy. Očekává se, že rok 2018 bude stěžejní pro směr vývoje inteligentních řízení budov, včetně osvětlení, zabezpečení a správy energií.

Mezi nejvýznamnější technologické firmy investujících do IoT a jejich produkty jsou považovány: [4]

- Amazon – AWS IoT cloud, zařízení pro domácí automatizaci Amazon Echo, nákupní zařízení Amazon dash button
- Apple – HomeKit pro inteligentní dům a HealthKit pro sledování zdraví
- AT&T – Startovací souprava IoT, obchodní model propojených automobilů
- Cisco – Cloud-based IoT platforma, hardware připojení, služby související s IoT, konzultace
- GE – průmyslová platforma založená na technlogiích cloudu a připojených průmyslových strojů Predix IoT
- Google – autonomní auto, domácí automatizace, IoT beacon technologie, práce na IoT standardech, IoT cloud
- IBM – IBM Watson IoT, cloudové služby
- Intel – tvoří hardware pro různé aplikace IoT
- Microsoft – operační systém Windows 10 IoT Core, Azure IoT
- Oracle – cloudová platforma poskytující služby pro IoT
- Samsung – platforma ARTIK pro IoT, inteligentní domácnost a zdravotnické zařízení
- Verizon – cloudová platforma ThingSpace, LTE modemy pro vývojáře IoT

**Cloud IoT Core** je IoT řešení od **Googlu** [23]**.** Základní prvky a terminologie:

- Chytré zařízení, připojitelné zařízení – zařízení připojené k internetu vyměňující si telemetrická data a stav s cloudem
- Telemetrie, telemetrická data – telemetrická data a události jsou zasílána do cloudu. Pro analýzu a vyhodnocení telemetrických dat lze využít „Google Cloud Big data Solutions“. Telemetrická data mohou být strukturovaná, nebo nestrukturovaná
- Stav zařízení – volitelné pole dat, které popisuje aktuální stav zařízení. Data mohou být opět strukturovaná nebo nestrukturovaná. Tato data plynou pouze směrem od cloudu k zařízení
- Registr zařízení – kontejner zařízení se sdílenými vlastnostmi. Každé zařízení je registrací přiřazeno ke službě
- Správce zařízení – bývá služba, která má na starosti monitoring aktivity a update stavu zařízení, autentizaci a pověření
- MQTT – standartizovaný IoT protokol výměny zpráv mezi zařízením a cloudem




*OBR. 10: CLOUD IOT CORE. PŘEVZATO [23]*

Hlavní komponenty Cloud IoT Core jsou správce zařízení a protokolové mosty. Správce zařízení umožňuje registrovat zařízení a asociovat se službou. Samozřejmostí je autentizace, konfigurace a monitoring. Dva protokolové mosty (MQTT a HTTP) umožňují výměnu telemetrických dat. Tato telemetrická data mohou být připojena k platformě „Google Cloud Platform Data“. Tato telemetrická data mohou být použita k vyvolání konkrétní služby na cloudu. Telemetrická data mohou mít rovněž charakter streamu a být analyzována pomocí služby „Cloud Dataflow“.

**Cisco** *[24]* přišlo s obecnější architekturou „Internet of Everything“. Jeho řešení jsou spíše pro velké společnosti v oblastech:

- výrobní sféry
- těžby ropy a plynu,
- dopravy,
- služeb.

Integrované řešení Cisco – „Cisco IoT Cloud Connect“ je variabilní a flexibilní, formou balíčků. Zahrnuje virtualizovaná datová centra na míru, inteligentní a bezpečné síťové prvky, což je základní těžiště této firmy. Rovněž rozšiřuje technologii připojení Internet to Everything a komunikace zařízení M2M a M2H. Výhodou řešení Cisco virualized Packed Core je integrace všech paketových služeb (4G, 3G, 2G, Wi-Fi a malé buňky). Virtualizací těchto služeb je možné optimalizovat kapacity a rychleji zavádět nové služby. Cisco rovněž podporuje IoT integrační snahy související s dopravou, kdy se koncová zařízení pohybují a komunikují mezi sebou.

**PlatformIO**

*[25]*

PlatformIO vychází z myšlenky integrace vývojových systémů pro IoT koncová zařízení do jednoho přenositelného ekosystému. Uživatel má na výběr:

1. Operační systém, na kterém se bude vyvíjet.
2. Editor pro psaní zdrojového kódu. Může to být jednoduchý editor nebo výkonný grafický editor a lze zvolit variantu klasického standardního IDE editor nebo cloudovou variantu.
3. Zaměřit se na vývoj zdrojového kódu, ne na platformu. Tedy lze vyvíjet na stejném ekosystému pro různé koncové MCU.

Tento ekosystém integruje frameworky, toolchainy a debuggery z více, než 200 vývojových desek 15 vývojových prostředí a 10 frameworků. Původně systém vznikal v příkazové řádce. Postupně byly integrovány dva editory Atom a VSCode, ty fungují na jakékoli plarformě. Z důvodu přenositelnosti systému byl zvolen programovací jazyk Python. V současné době se Platform IO skládá z:

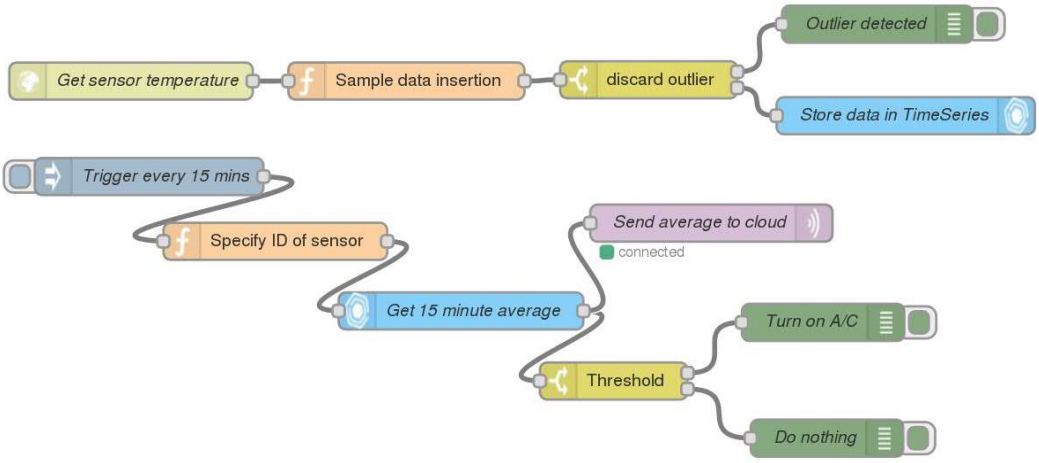
- PlatformIO IDE – integrované vývojové prostředí pro IoT obsahující PIO Unified Debugger, PIO Remote, PIO Unit Testing. Toto IDE rovněž obsahuje prvky jiných vývojových IDE.
- PlatformIO Core – multiplatformní kompilační a sestavovací vývojový systém s multiplatformním manažerem balíčků, knihoven, podporou hledání závislostí „Library Dependency Finder (LDF)“, monitorem přes sériový port a dalšími integračními komponentami
- Program PIO Home umožňuje komunikovat s platformou PlarformIO pomocí moderního a multiplatformních grafických rozhraní. Dále je podporován správa PIO účtů, knihoven, procesorových desek, zařízení.

Tento ekosystém zaštiťuje především část IoT související s návrhem koncových embedded zařízení, kde je roztříštěnost workflow návrhů vývojových nástrojů a frameworků.

**Node – RED**

*[26]*

Zajímavou integrační snahou je projekt Node – RED. Jedná se o programovací nástroj s GUI pro propojení hardwarových zařízení, API a online službě v IoT ve formě blokových diagramů. Tento nástroj umožňuje propojovat terminály entit pomocí vazeb, které definují toky v blokovém diagramu. Tyto toky mohou reprezentovat jak zprávy, tak i složitější struktury.




Obr. 11: Ukázka řešení v systému node-red. Převzato z [26]

Entity, které lze propojovat jsou definovány jako uzly/nody. Programování těchto uzlů musí podporovat události a „non blocking“ mód. Tedy není možné použít dotazování na změnu stavu v cyklu. V současné době může Node-RED běžet na těchto platformách a zařízeních:

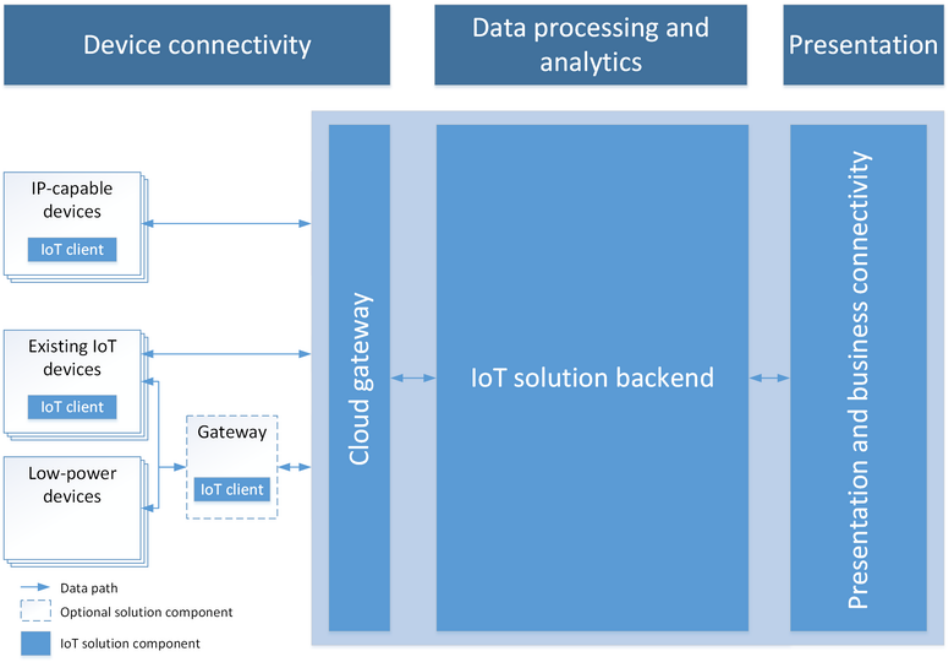
- Lokálně
  - Pc
  - Docker
- Na zařízeních
  - Raspberry PI
  - BeagleBone Black
  - Interacting with Arduino
  - Android
- Na cloudu
  - IBM Bluemix
  - SenseTecnic FRED
  - Microsoft Azure

**Azure IoT Suite**

*[27]*

Azure IoT Suite je vývojový nástroj firmy Microsoft pro návrh, integraci a realizaci řešení IoT. Základní architektura IoT se skládá z bloku konektivity zařízení, bloku zpracování a analýzy telemetrických dat, vrstvy prezentace dat a business konektivity.

- Konektivita zařízení – základním problémem je kvalita připojení, zabezpečení komunikace, různé platformy os v zařízení, nutnost preprocesingu v koncovém zařízení, transportní zpoždění s cloudem nebo bránou
- Zpracování telemetrických dat – zpracování a analýza dat na cloudu, nebo zařízení. Volba závisí na několika faktorech, na kvalitě sítě, transportním zpoždění, nutnosti realtime zpracování. V případě zpracování dat na cloudu hraje rovněž podstatnou úlohu predikce z telemetrických dat
- IoT backend musí rovněž umožňovat registraci, konfiguraci zařízení, správu pověření
- Vrstva prezentace dat a business konektivity umožňuje koncovým uživatelům zobrazit shromážděná a analyzovaná data, integrovat do podnikových systémů a podnikových procesů




Obr. 12: Architektura iot. Převzato z [27]

Předkonfigurovaná řešení pro oblasti průmyslu, dopravy, služeb:

- Azure IoT Hub – zajištění obousměrné výměny zpráv mezi zařízením a cloudem. Služba rovněž umožňuje konfiguraci zařízení. Funguje jako brána ke klíčovým službám IoT Suite
- Azure Event Hubs – tato služba poskytuje investování velkého objemu událostí do cloudu
- Azure Time Series Insights – umožňuje zpracování, analýzu a zobrazení telemetrických dat
- Azure Container Service – hostování a spravování mikroslužeb
- Azure Cosmos DB, Azure Storage – služba databáze a ukládání dat
- Azure Stream Analytics – zpracování telemetrických dat, metadat a reakcí zařízení na zprávy
- Azure Web Apps – hostování kódu aplikací

## 1.6 KOMUNIKAČNÍ STANDARDY

Komunikační standardy definují, co přenášená data reprezentují a jak se s přenášenými daty má nakládat. Z pohledu síťového ISO/OSI modelu se jedná o aplikační vrstvu. V současnosti se vyvíjí řada komunikačních protokolů, které se snaží uplatnit v IoT spektru. Komunikace podle standardu zajišťuje vzájemnou kompatibilitu v komunikaci mezi zařízeními různých výrobců.

Při výběru vhodného komunikačního protokolu je potřeba zvážit tyto faktory:

- Množství přenášených dat
- Způsob komunikace
- Zabezpečení přenosu dat
- Zabezpečení proti neoprávněnému přístupu k datům
- Účel přenášených dat

### 1.6.1 HTTP

[28]

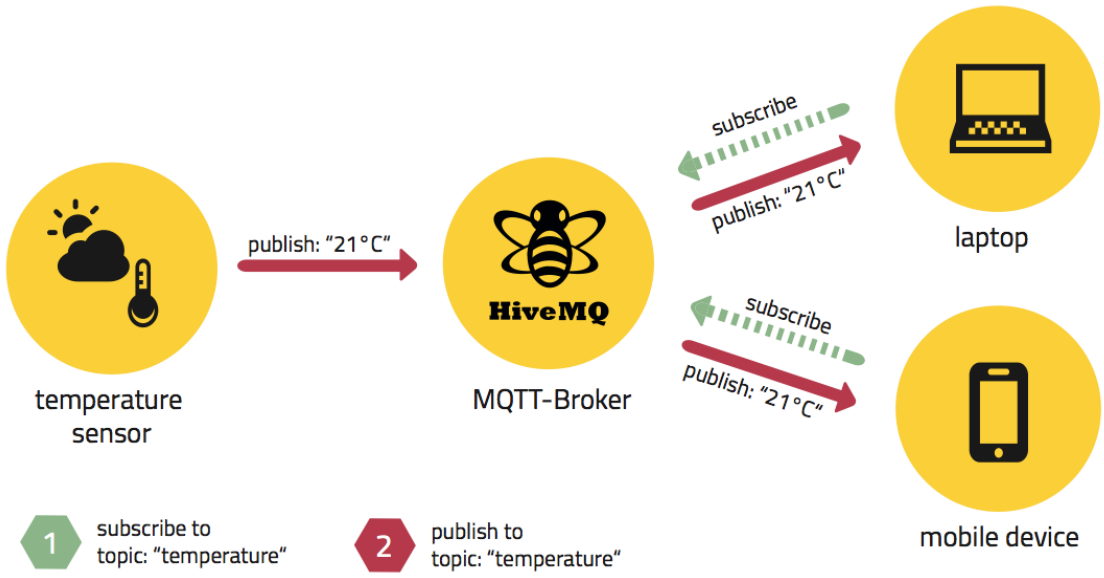
Hyper Text Transfer Protocolu (zkratka HTTP) je jeden z nejdůležitějších protokolů Internetu pro přenos dat mezi serverem a klientem. HTTP je jednoduchý protokol implementující malou množinu příkazů, schopný přenášet data určená URL adresou. Využívá standardů MIME, díky čemuž se dá rozšiřovat o formáty různých médií. Umožňuje efektivní využití síťových prostředků s nižším zpožděním a také paralelní výměnu dat v jednom spojení. Protokol HTTP komunikuje na portu 80 TCP protokolu, je ovšem možno zadat i jiný port. Protokol HTTP není závislý na protokolu TCP a může pracovat i s jinými spolehlivými protokoly. Tento protokol má v IoT ale několik nevýhod. Je náročnější na implementaci a pro vzájemnou komunikace je třeba nainstalování serverové aplikace. Protokol sám o sobě neřeší QoS úrovně.

### 1.6.2 MQTT

[29]

MQTT je jednoduchý a otevřený protokol pro zasílání zpráv mezi klienty přes brokera (centrální řídící prvek) navržen tak, aby byl co nejlépe implementovaný [25]. V IoT se jedná o velmi používaný protokol pro zařízení s malou spotřebou energie. Původně vznikl ve společnosti IBM, dnes je však otevřeným standardem. Je postaven na modelu klient – server. Server je MQTT broker, který se stará o výměnu zpráv mezi klienty. Klient je senzor/koncové zařízení, který se připojuje k serveru přes TCP protokol. MQTT broker může vyžadovat při spojení od klientů jméno a heslo a spojení může být šifrováno pomocí SSL/TLS.

Klienti mohou publikovat zprávy (publish) nebo se mohou přihlásit k odběru (subscribe) konkrétních tematických zpráv. Veškeré zprávy jsou tříděny do témat (topic). Broker zajišťuje přijímání zpráv od publikujících klientů, a provádí jejich směrování konkrétním klientům, kteří jsou v daném tématu přihlášeni k odběru. Klient může v nějakém tématu publikovat a v jiných může být přihlášen k odběru.

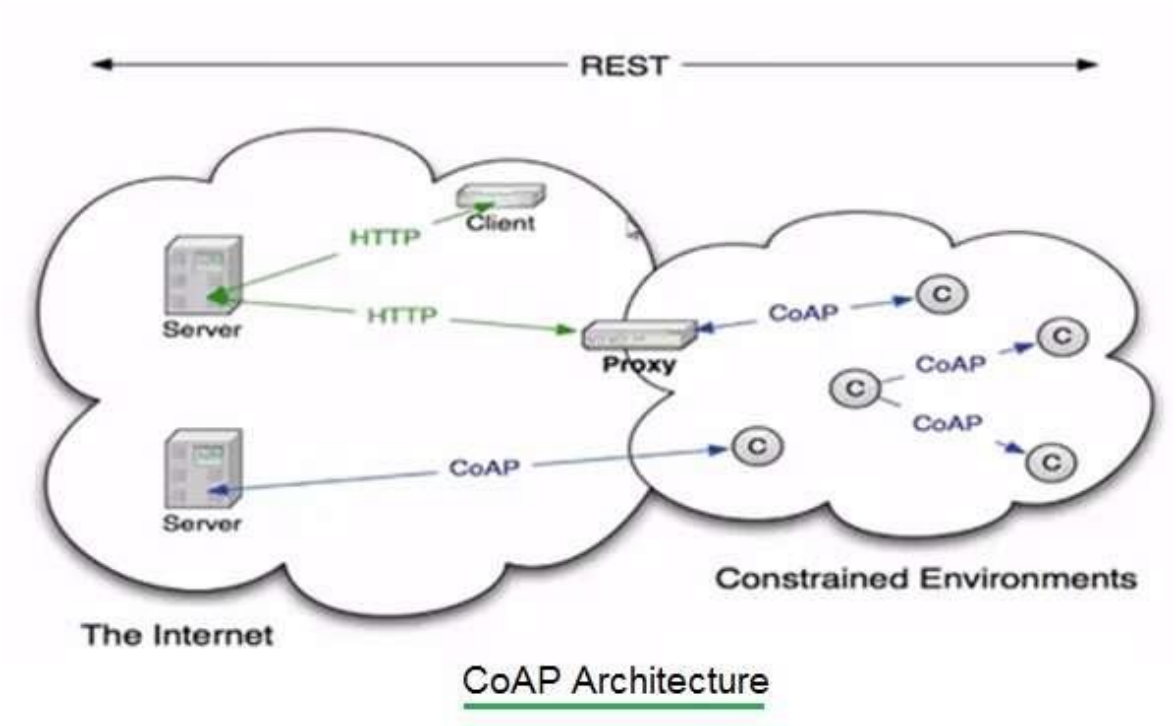



*OBR. 13: PŘÍKLAD ARCHITEKTURY MQTT. PŘEVZATO Z [30]*

### 1.6.3 CoAP

[31]

CoAP (Constrained Application Protocol) vychází z HTTP. Má za cíl zjednodušit implementací v mikrokontrolérech a snížit množství přenášených dat. Jedná se o otevřený standard umožňující asynchronní komunikaci. Stejně jako http je postaven na modelu klient – server (klient posílá požadavky, server mu odpovídá). Je navržen tak, aby byl umožněn snadný překlad zpráv mezi http a CoAP protokoly. Také umožňuje REST rozhraní.




*OBR. 14: ARCHITEKTURA COAP. PŘEVZATO Z [31]*

Primárně je využíván pro komunikace device-to-device. CoAP protokol má vestavěnou podporu pro označování zpráv pomocí metadat, která slouží k určení typu zprávy. Využívá se UDP protokol namísto TCP. CoAP navíc řeší QoS úrovně, zprávy se označují příznakem pro potvrzení/nepotvrzení po doručení. Zabezpečení je řešeno na transportní vrstvě s podporou RSA a AES.

## 1.7 BEZPEČNOST V PROSTŘEDÍ INTERNETU VĚCÍ

[32]

S expanzivním nástupem internetu věcí souvisí otázky zabezpečení. Internet věcí nachází využití jak v průmyslové a veřejné sféře, tak i pro osobní použití. V každé z těchto oblastí by mělo být hlavní prioritou zamezení neoprávněného přístupu a manipulace se zařízeními v síti. Zneužití jednotlivých zařízení může vést k finančním ztrátám, ale je zde také možnost, že cíleně pozměněný chod přístrojů může mít fatální následky na zdraví uživatele. Bezpečnost internetu věcí by měla splňovat kritéria kybernetické bezpečnosti.

Pro různá odvětví internetu věcí jsou zapotřebí různé stupně zabezpečení dat proti jejich zneužití. V oblasti osobního využití se může jednat o „chytrou“ domácnost, kde jsou jednotlivé věci připojeny k síti a komunikují s uživatelem přes aplikace v telefonu nebo přes centrální prvek pro inteligentní domácnost. Věci jsou nejčastěji připojeny k síti přes Wi-Fi. Právě tyto sítě jsou na trhu delší dobu, a proto je k dispozici větší množství materiálů popisující slabé stránky této technologie.

Hugh Ujhazy, analytik specializující se na IoT ve společnosti IDC, uvádí, že řada vývojových firem přináší nová a inovativní řešení IoT technologií. Na druhou stranu se ale již příliš nezabývají problematikou bezpečnosti. Důležité je zabezpečení především koncových zařízení. Důraz je kladen na jejich monitorování, eliminaci bezpečnostních chyb a neustálý rozvoj bezpečnostních mechanismů. Monitoring zařízení by měl řešit případy, kdy nastanou neočekávané bezpečnostní komplikace nebo anomálie. Systém by pak měl umět na tyto situace vhodně reagovat. Jen dobře zabezpečené zařízení je stále přínosné pro byznys.

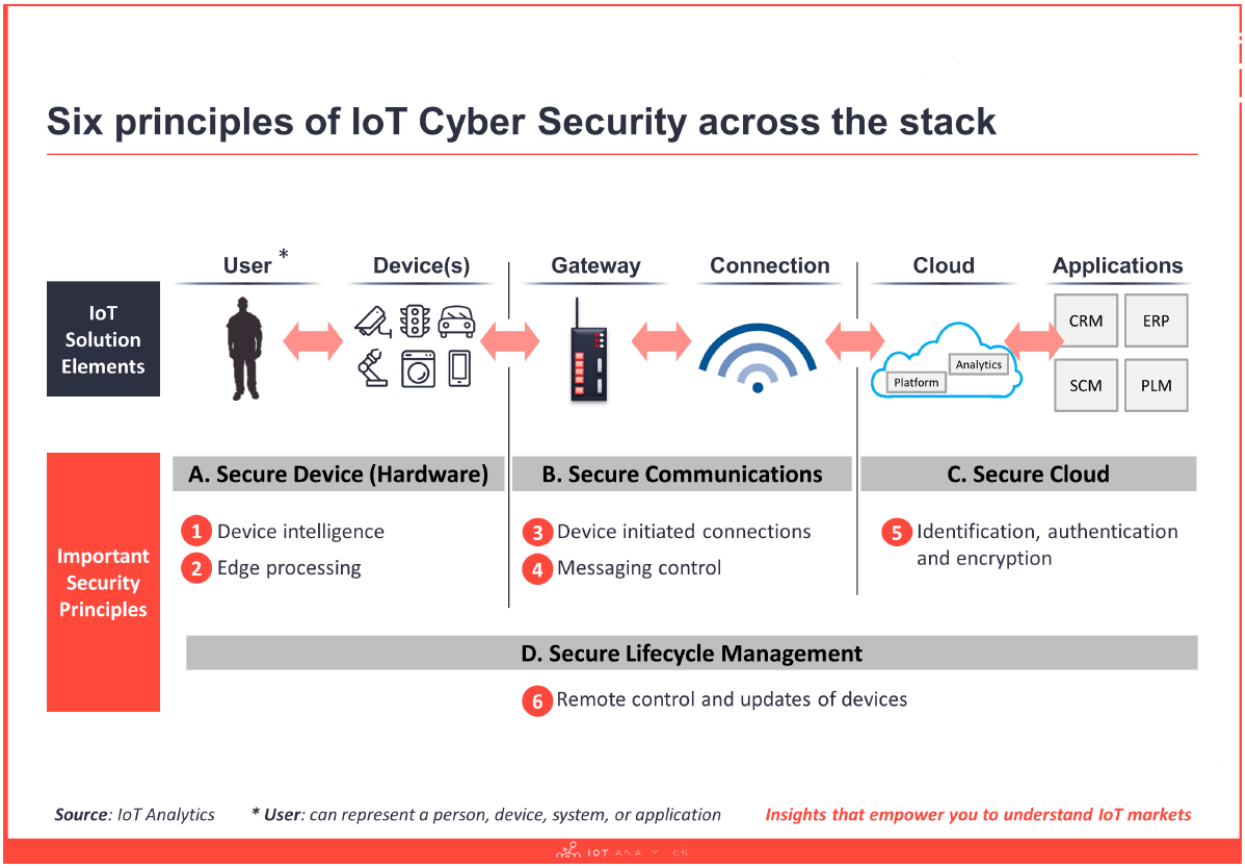
Dle analytiků společnosti IDC lze bezpečnost koncových IoT zařízení rozdělit na tři oblasti:

- fyzická bezpečnost koncových zařízení - zamezení neautorizovaného přístupu
- bezpečnost přístupu zařízení k síti - autorizovaný a řízený přístup
- bezpečnost přenášených dat - zajištění bezpečného spojení zařízení a cloudové aplikace, šifrování

## PRINCIPY IOT BEZPEČNOSTI

George Cora, generální ředitel společnosti Ardexa, zaměřující se na bezpečnost IoT, stanovil šest klíčových zásad zabezpečení IoT. Ty se dají rozdělit do následujících čtyř oblastí:

1. zabezpečení zařízení,
2. zabezpečení komunikace,
3. zabezpečení cloudu a cloudové aplikace,
4. bezpečnostní management životního cyklu IoT systému




*OBR. 15: ŠEST KLÍČOVÝCH ZÁSAD BEZPEČNOSTI. PŘEVZATO Z [33]*

### OBLAST ZABEZPEČENÍ ZAŘÍZENÍ

Cílem je, aby výrobci IoT zařízení integrovali do svých výrobků bezpečnostní prvky. Může se jednat o prvky, které zajišťují:

- ochranu proti fyzickému přístupu k zařízení a znemožnění modifikace zařízení
- důvěryhodný přístup k datům a ochranu citlivých informací
- řízení ochrany spouštěných aplikací

### OBLAST ZABEZPEČENÍ KOMUNIKACE

Komunikační vrstva se využívá k propojení IoT zařízení s cloudovou aplikací. Dle ISO/OSI modelu je potřeba řešit bezpečnost na jednotlivých vrstvách - fyzické (Ethernet, WiFi, LPWAN sítě), síťové (IP, Modbus, OPC-UA) a aplikační (MQTT, CoAP, HTTP). Nezabezpečená komunikace je totiž náchylná k různým útokům, např. útoku typu man-in-the middle.

Důležité funkce zabezpečení komunikace:

- bezpečné šifrování dat,
- vybudovaný systém prevence průniku (IPS) a aktivní firewall

## Zabezpečení cloudu a cloudové aplikace

[32]

Cloud je považován za backendové řešení, které přijímá veškerá data z uživatelských zařízení. Data analyzuje, zpracovává a interpretuje v požadované uživatelské formě. Většina cloudových platforem má bezpečnost jako jedno z hlavních témat. Zákazník se může rozhodovat mezi on-premis (na vlastních systémech) řešením a cloudovou platformou právě na základě bezpečnostní politiky dané firmy. Většina firem již má široce propracovanou bezpečnostní politiku a má nastaveny účinné mechanismy pro udržení kvality a dostupnosti svých služeb, kterou aplikuje již v základu.

- Veškerá data, která jsou uložena v cloudu, je potřeba účinně šifrovat, aby nedošlo k zneužití těchto dat útočníkem nebo poskytovatelem cloudové platformy. Je třeba analyzovat, zdali k datům přistupují i aplikace třetí strany, které mají nižší úroveň zabezpečení. Přes tyto aplikace se totiž může potencionální útočník dostat k jinak zabezpečeným informacím.
- Identifikace a autorizace přístupu k datům by měla probíhat na základě ověření digitálním certifikátem.
- Je potřeba ověřovat kvalitu a integritu platforem třetích stran, které se snaží komunikovat a přistupovat k datům v cloudu. Především z důvodu ochrany dat a před případnými škodlivými aktivitami těchto platforem.

### BEZPEČNOSTNÍ MANAGEMENT ŽIVOTNÍHO CYKLU IOT SYSTÉMU

V rámci celého životního cyklu provozu IoT systému je potřeba mít vhodně nastavený bezpečnostní management. Ten bude především pokrývat nová rizika, bezpečnostní a výkonové problémy a vylepšení. Cílem je stále udržet celý systém zabezpečený. Je potřeba identifikovat různé bezpečnostní problémy a kontrolovat aktuálnost zabezpečení jednotlivých prvků v systému. Také je podstatné provádět pravidelné audity a dodržovat bezpečnostní politiku. Důležité je rovněž nastavení procesu implementace nových zařízení do systému, jejich provozu, údržby, vyřazení a následné likvidace. Dobře nastavená bezpečnostní politika je prvním krokem k trvalému a udržitelnému chodu IoT systému.

## Důležité bezpečnostní funkce:

- Monitorovat a zaznamenávat veškeré aktivy, hledat podezřelé chování a upozornit na ně.
- V pravidelných intervalech kontrolovat vydávaná upozornění na nalezené zranitelnosti, instalovat bezpečnostní aktualizace a posílit tak odolnost proti útoku.
- Zabezpečit dálkové ovládání prvků sítě.
- Pravidelně obměňovat již nepodporované infrastruktury za nové.

### 1.7.1 PŘEHLED NEJČASTĚJŠÍCH ÚTOKŮ V IOT

Útoky na zařízení spadající do internetu věcí jsou podobné již známým způsobům z prostředí počítačových sítí.

**Man-in-the-middle attack** – Útočník odposlouchává a manipuluje s komunikací mezi dvěma zařízeními. Útočník je schopen vytvářet falešné zprávy, které přesvědčí obě strany, že komunikují mezi sebou. Nejčastěji se tento útok provádí v sítích s nedostatečně ověřeným spojení. Primárním opatřením před tímto typem útoku by tedy mělo být spolehlivé ověření na zabezpečeném kanále.

**Replay attack** – Při tomto typu útoku dochází k úmyslnému zpoždění nebo opakování reálných zpráv od jednoho zařízení s cílem zmatení druhého. Útočník je schopen vynucení si nesprávné, či duplicitní sekvence odpovědí, ze které může získat informace k přístupu k zařízení. Účinnou ochranou proti tomuto útoku je zahrnutí dalších kontrolních informací, které jsou platné pouze pro jedno použití nebo zavedení synchronizace mezi zařízeními.

**Denial of service (DoS)** – Úmyslné vyřazení zařízení z provozu pomocí přetížení velkého množství zpráv. Nutnou podmínkou je znalost IP adresy zařízení. Ochrany proti tomuto útoku se většinou implementují na síťové prvky. V IoT se ovšem setkáváme s jiným problémem. IoT zařízení se díky svému vysokému počtu a nízkému stupni zabezpečení staly častým cílem jiných druhů útoky (nejčastěji malware) se záměrem jejich infikováni, aby byly následně využity jako zdroj síťového provozu pro útoky DoS.

**Malware** – Zřejmě nejznámější pojem z předešlých. Zkratka pro malicious software, tedy software určený k narušení chodu infikovaného zařízení. Jedním z největších problémů malware útoku je skutečnost, že škodlivý software může zůstat aktivní v zařízeních po dlouhou dobu bez povšimnutí.

**Útoky na lokální sítě** – Dostane-li se útočník do lokální sítě obsahující inteligentní zařízení, může se většinou zmocnit plné kontroly. Toho je možné dosáhnout, protože jedním z rozšířených nedostatků je přenos zpráv v místní síti ve volném textu. U zařízení, která ještě komunikují s cloudovou službou musí útočník ještě provést Man-in-the-Middle útok, pro získání kontroly nad tímto zařízením.

**Útoky na Cloud** – Někteří výrobci IoT poskytují cloudová řešení pro své výrobky (existují i případy kdy je připojení do cloudu vynucené). Cloud je většinou dobře zabezpečen před útoky na svou strukturu, ať už funguje na hardwaru třetích stran nebo u výrobce technologie. V kontrastu k této skutečnosti, většina služeb není dobře chráněná před neoprávněným vniknutím. Opět se zde setkáváme s problematikou nevynucování silných hesel v kombinaci s nedostatečnou ochranou před opakovanými neúspěšnými pokusy o přihlášení [34]**.** Některé ze současných cloudových služeb také mají nedostatečně vyřešené algoritmy pro obnovu zapomenutých hesel, kdy při procesu systém poskytuje citlivé informace o majiteli účtu, nebo dokonce nevyžaduje autorizaci pro dokončení změny hesla.

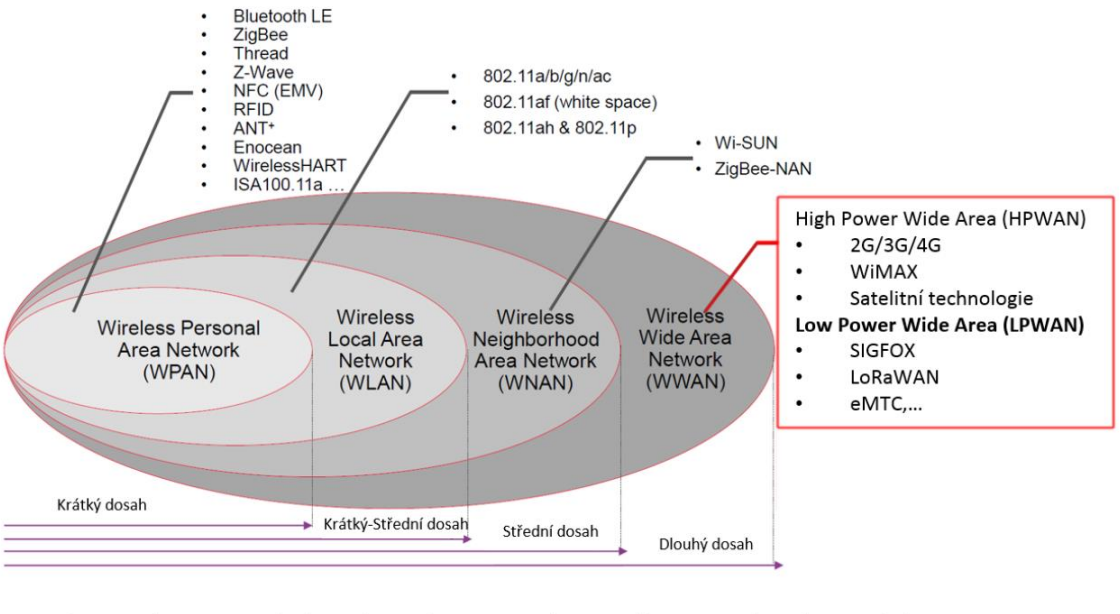
**Direct Access** – Téměř bez výjimky platí, že veškerá zařízení jsou náchylnější k útokům, pokud k nim má útočník fyzicky přístup. V těchto případech mohou vznikat zadní vrátka pro přístup systému nebo může dojít k pozměnění chodu zařízení. Nutnou podmínkou pro tento útok je dostatečně dlouhá doba fyzického přístupu k zařízení pro provedení jeho modifikace.

# 2 KOMUNIKAČNÍ TECHNOLOGIE

Sítě pro IoT jsou zpravidla bezdrátové a jejich komunikace je prováděna na různých kmitočtech, řádově od stovek MHz po jednotky GHz. K dispozici je několik druhů komunikačních technologií. Pro účely zjednodušení rozdělím bezdrátovou komunikace dle výkonu na nízkovýkonovou a standartní. Do kategorie nízkovýkonových bezdrátových sítí, značených LPWAN nebo 6LoWpan, patří především technologie LoRa a SIGFOX.

| Proximity | WPAN | WLAN | WNAN | WWAN |
|---|---|---|---|---|
| NFC | Bluetooth LE | 802.11a/b/g/n/ac | Wi-SUN | 2G/3G/4G |
| RFID | ZigBee | 802.11af | ZigBee-NAN | LTE |
| | Thead | 802.11ah | | 5G |
| | Z-Wave | 802.11p | | SigFox |
| | WirlessHART | | | LoRa |
| | ISA100.11a | | | Telensa |
| | | | | PTC |
| | | | | 802.22 |
| | | | | WiMax |

*TAB. 1: ROZDĚLENÍ BEZDRÁTOVÝCH TECHNOLOGIÍ DLE DOSAHU [35]*




Obr. 16: Přehled dosahu bezdrátových technologií, inspirováno [35]

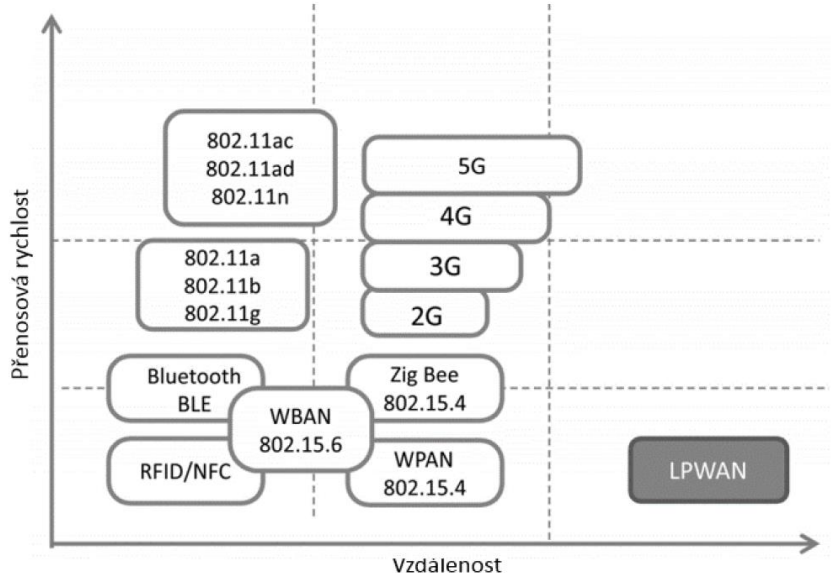
## 2.1 SÍTĚ VELKÉHO DOSAHU

Nízkovýkonové sítě (LPWAN - Low Power Wide Area Network) jsou sítě navrženy vyloženě za účelem minimální spotřeby energie. Vyznačují se také nízkými přenosovými rychlostmi. Zásadním aspektem této technologie je možnost velkého dosahu vzhledem k nároku na výkon. V zastavěné oblasti můžeme mít dosah v řádu kilometrů, v případě přímé viditelnosti až desítky kilometrů. Další nespornou výhodou je možnost dlouhodobého provozu vzhledem k právě minimální spotřebě.

Jak poukazuje ETSI, sítě LPWAN vyplňují niku mezi standartními mobilními sítěmi WWAN a technologiemi s krátkým dosahem (WPAN,WLAN)[37]. Jsou speciálně navržené pro komunikaci v IoT a ačkoliv vznikla řada protokolů, standardů a technologií, které se zaměřují na určitý výsek z balíku služeb, sdílejí společně následující hlavní cíle a vlastnosti:

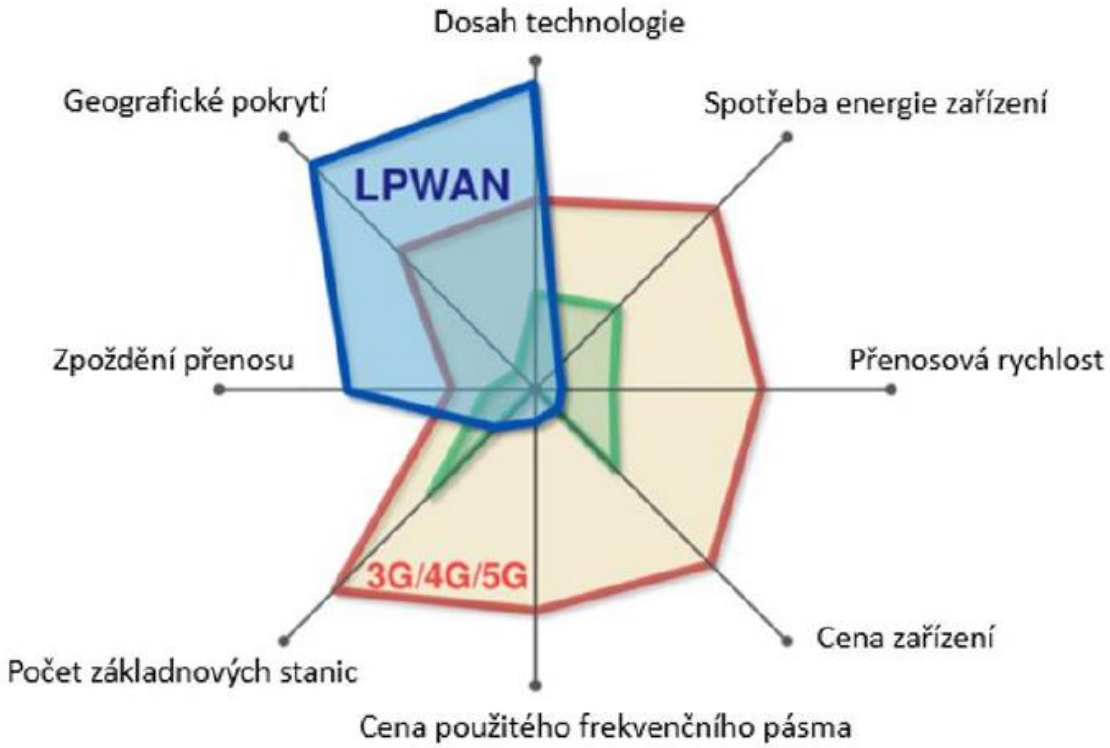
- Nízké přenosové rychlosti (~100(k)b/s)
- Maximalizace dosahu (~50km)
- Nízká spotřeba energie koncových zařízení (~20uAh)
- Velmi nízká cena - jak z pohledu koncových zařízení (KZ), tak z pohledu nákladů na výstavbu a provozování sítě

Na tyto aspekty poukazuje Egli [36] v následujícím diagramu, ve kterém je znázorněna pozice LPWAN v závislosti na přenosové rychlosti:




Obr. 17: Pozice lpwan v závislosti na předpokládané přenosové rychlosti. Převzato z [36]

Iniciativa LPWAN tedy spojuje několik desítek mobilních operátorů, výrobců modulů, infrastruktury i čipů za účelem zrychlení vývoje a komerčního nasazení LPWAN řešení. Iniciativa uvádí, že široké spektrum požadavků není možné splnit jednou technologií, a proto se zaměřuje na tři komplementární 3GPP standardy – evoluci LTE, evoluci GSM, popř. zcela novou technologii.




*OBR. 18: POZICE 3GPP SÍTÍ VŮČI LPWAN. PŘEVZATO Z [38]*

Na následujících plenárních zasedáních GERAN CP-67 (08/2015), GERAN CP-69 (09/2015), GERAN CP-70 (12/2015) byla posuzována řada návrhů bezdrátových technologií určených pro IoT [39][40][41]. Přesněji se nejprve jednalo o návrh a analýzu dvou variant evoluce GPRS technologie a čtyři úplně nové radikálně zjednodušené návrhy. Všechny návrhy měly společnou snahu o přepoužití licencované části spektra zakoupené mobilními operátory. Významného pokroku bylo dosaženo na zasedání GERAN CP-69 (02/2016), kde byly schváleny dvě z nových úzkopásmových řešení, splňující cíle stanovené GERAN CP-62, využívající techniky popsané v 3GPP TR 45.820 [39]. Prvním bylo NB-CIoT (Narrow Band Cellular Internet of Things), podporované především Huawei/Neul, Vodafone, T-Mobile a společností Qualcomm. Druhým NB-LTE (Narrow Band Long Term Evolution) podporované především společností Ericsson, Nokia a Intel. Dalším z důležitých kroků jednání bylo schválení EC-GSM-IoT (Extended Coverage GSM IoT) jako jediné technologie vhodné pro evoluci GSM a EGPRS (Enhandced General Packet Radio Service). Zároveň došlo ke změně předchozího pojmenování EC-EGPRS (Extended Coverage EGPRS) na EC-GSM-IoT [42].

V lednu 2016 byl oznámen zánik technické skupiny GERAN a přesun jejích pracovních podskupin (Working Group) WG1-WG3 do RAN TSG do podskupin RAN TSG WG5 a WG6.

Až na jednání CP-70 (05/2016) bylo stanoveno, že jediným řešením bude NB-IoT (NarrowBand IoT). Dále bylo rozhodnuto o zařazení NB-IoT jako funkcionality LTE, tudíž lze nyní NB-IoT nalézt ve specifikacích LTE/E-UTRAN řady 36.xxx (i přes fakt, že NB-IoT lze provozovat jak v LTE, tak ve 2G rádiové přístupové síti) [42]. Standardizace NB-IoT byla úspěšně dokončena v červnu 2016 zařazením cca 150 změnových požadavků do specifikací Release 13 (LTE Advanced Pro) [43].

### 2.1.1 NB-IoT, EC-GSM-IoT

Jedním z možných řešení je možnost využít stávajících mobilních sítí. Od roku 2013 se touto myšlenkou zaobírala již neexistující pracovní skupina GERAN. V zápise z jednání č. 62 GP-140421 skupina uvedla následující hlavní cíle studie hledající řešení pro buňkový systém internetu věcí (Cellular-IoT, CIoT) [44]

- Rozšíření pokrytí uvnitř budov (o 20 dB vůči GPRS a datovou rychlostí 160 b/s)
- Podpora masivního počtu zařízení určených pro komunikaci mezi stroji
- Různá citlivost na zpoždění (od zaslání jednou za 24h až po každých 10 min)
- Nízká spotřeba (výdrž s baterií až 10 let při kapacitě 10 Wh)
- Optimalizovaná architektura

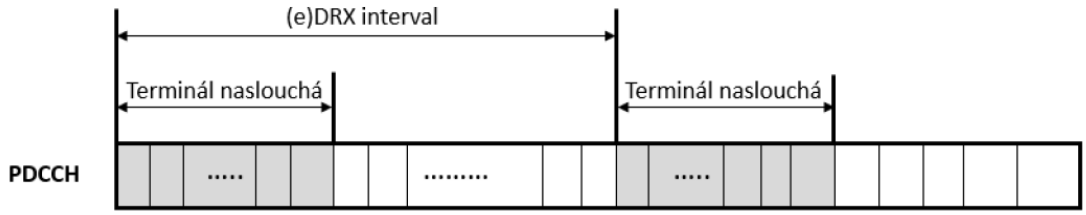
**Specifika EC-GSM-IoT**

Spotřeba energie byla minimalizována díky následujícím funkcionalitám:

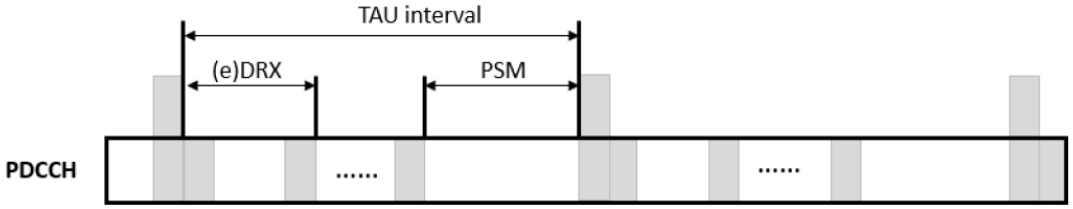
- Použití eDRX (extended Discontinuous Reception) – jedná se o prodloužení intervalu, kdy termín nenaslouchá na přijímací straně (není dekódován PDCCH (Psychical Downlink Control Channel)) [48]. Interval prodloužen z 2560 ms na 5,12 s a 10,24 s v připojeném stavu a přibližně 3 h v klidovém režimu. [45][46][47]
- Použití PSM (Power Saving Mode) [46] – umožnění zařízení v RRC_Idle (Radio Resource Control Idle) periodicky spouštět proceduru aktualizace sledovací oblasti RAU/TAU (Routing/Tracking Area Update), přičemž v mezičase je zařízení nečinné v hlubokém spánku. V zařízení zároveň zůstává aktivní PDN konektivita.

Další důležité vlastnosti [49]

- Výkonová balance spoje:
  - Až 164 dB při efektivním vyzářeném výkonu 33 dBm
  - Až 154 dB při efektivním vyzářeném výkonu 23 dBm
- Přenosová rychlost:
  - Při použití GMSK: 70-350 kb/s
  - Při použití 8PSK: 240 kb/s
- Až 50 tisíc zařízení v jedné buňce
- Šířka kanálu: 200 kHz na kanál, 2,4 MHz celkem
- Mód: HD-FDD (Half-Duplex Frequency Division Duplexing)
- Metoda mnohonásobného přístupu: TDMA (Time Divison Multiple Access)/FDMA (Frequency Division Multiple Access)




(a) Znázornění intervalu (e)DRX

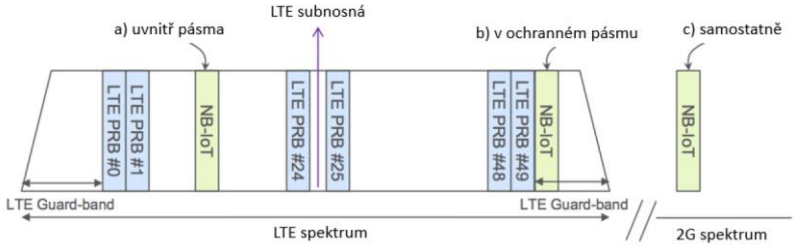



Obr. 19: Optimalizace rádiového prostředí edrx a psm. Převzato z [48]

**NB-IoT (LTE Cat NB1)**

[50],[51],[52]

Výhodou tohoto řešení je možnost být provozován v rádiovém pásmu 2G i LTE:




Obr. 20: Spektrum nb-iot

Příchozí směr:

- Metoda mnohonásobného přístupu: OFDMA (Orthogonal Frequency-Division Multiple Access)
- V časové oblasti je přenos rozdělen do slotů s trváním 0,5 ms a každý dále na 7 OFDM symbolů. Data v kmitočtové oblasti jsou dále kódována na 12 sub-nosných s šířkou pásma 15 kHz [50]. Dva sloty (s délkou 0,5 ms) jsou dále namapovány do 10 sub-rámců a ty do jednoho rámce s délkou trvání 10 ms
- Přenosová rychlost: až 250 kb/s

Odchozí směr:

- Metoda mnohonásobného přístupu: SC-FDMA (Single Carrier - Orthogonal Frequency-Division Multiple Access)
- V časové oblasti je přenos rozdělen do pěti slotů s trváním 0,5 ms a každý dále na 7 OFDM symbolů. Data v kmitočtové oblasti jsou ale dále kódována buď stejně jako v příchozím směru (12 sub-nosných), nebo na 48 sub-nosných s šířkou pásma 3,75 kHz a dobou trvání 2 ms
- Přenosová rychlost: 20/250 kb/s
- Vysílací výkon: 23 dBm

Použití eDRX[46]:

- C-eDRX: 5,12 s a 10,24 s
- I-eDRX: až cca. 3 h

**eMTC (LTE Cat M1)**

[53],[54]

LTE MTC (Machine Type Communication) - základem je snaha přidat do LTE podporu pro MTC. První verze byla vydána v 3GPP Rel 8 na základě kategorie 1 (CAT 1). Tato verze ale ještě nesplňovala zvýšené nároky na LPWAN sítě a až 3GPP Rel 12 a nová kategorie Cat 0 umožňuje zlevnit KZ např. opuštěním techniky MIMO a snížením přenosové rychlosti (na 5 Mb/s). Další výrazné zjednodušení přineslo 3GPP Rel 13, kde došlo k dalšímu ponížení přenosové rychlosti (1 Mb/s) zavedením poloviční duplexu a možnost snížení vysílacího výkonu na 20 dBm. Zařízení může být provozováno pouze v LTE pásmu.

Příchozí směr:

- Metoda mnohonásobného přístupu: OFDMA, detaily viz NB-IoT
- Přenosová rychlost: 1 Mb/s

Odchozí směr:

- Metoda mnohonásobného přístupu: SC-FDMA (Single Carrier - Orthogonal Frequency-Division Multiple Access)
- V časové oblasti je přenos rozdělen opět do slotů s trváním 0,5ms a každý dále na 7 OFDM symbolů. Data v kmitočtové oblasti jsou ale dále kódována stejně jako v příchozím směru (12 sub-nosných)
- Přenosová rychlost: 1 Mb/s
- Vysílací výkon: 23 dBm

Použití eDRX:

- C-eDRX: 5,12 s a 10,24 s
- I-eDRX: cca. 44 min

### 2.1.2 LORA

[55] [56]

Původně navržená LPWAN technologie firmou Semtech, která je nyní dále rozvíjena a standardizována neziskovou organizací LoRa Alliance [57]. Podstatu technologie tvoří dvě části – LoRa modulace a LoRaWAN (Long Range Widw Area Network) síťový protokol [58]. Technologie je určena pro IoT aplikace v ISM (Industry, Science and Medicine) pásmu. Díky protokolu LoRaWAN je umožněn transparentní a zabezpečený přenos dat (algoritmus AES s 128 bitovým klíčem) mezi koncovým zařízením a LoRa Gateway na principu hvězdicové jednoskokové topologie [55],[59]. Sítě LoRa pracují na různých kmitočtech, dle oblasti používání. V Evropě je nelicencované kmitočtové pásmo v oblasti okolo 868 MHz.

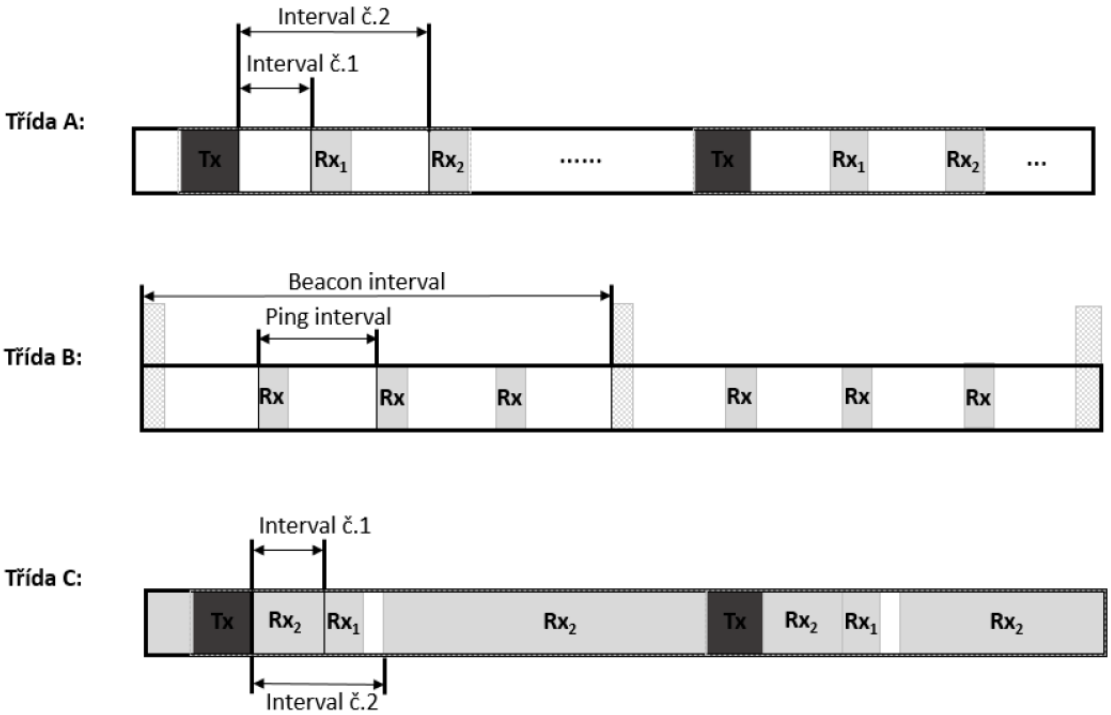
| Třída datového toku | Konfigurace | Bitový tok [bps] |
|---|---|---|
| 0 | LoRa: SF12 / 125 kHz | 250 |
| 1 | LoRa: SF11 / 125 kHz | 440 |
| 2 | LoRa: SF10 / 125 kHz | 980 |
| 3 | LoRa: SF9 / 125 kHz | 1760 |
| 4 | LoRa: SF8 / 125 kHz | 3125 |
| 5 | LoRa: SF7 / 125 kHz | 5470 |
| 6 | LoRa: SF7 / 250 kHz | 11000 |
| 7 | FSK: 50kbps | 50000 |
| 8 … 15 | | |

*TAB. 2:ZAŘÍZENÍM VYSÍLANÝ DATOVÝ TOK V SÍTI LORA, PŘEVZATO Z [60]*

Protokol nám dává několik možností pro každé koncové zařízení. Můžeme zvolit jiné datové rychlosti, vysílací výkon a maximalizovat tedy výdrž baterií. Datovou rychlost lze zvolit v rozmezí 0,3 – 50 kb/s. V LoRa zařízeních můžeme nastavit následující přenosové parametry [56]:

Třídy koncových zařízení:

- Třída A: Princip obousměrné komunikace, kdy může být zpráva vyslána v libovolném čase (na principu metody ALOHA [57]), a zároveň definuje časové intervaly, kdy bude zařízení naslouchat. Tato třída je nezbytná pro veškerá zařízení.
- Třída B (Beacon): Nepovinná implementace takové třídy, kdy zařízení může opět komunikovat obousměrně, ale již s využitím předem definovaných časových oken. Tato okna jsou synchronizována použitím krátkých synchronizačních rámců tzv. Beacon frames.
- Třída C (Continuous): Obdoba třídy B s tím rozdílem, že zařízení naslouchá vždy, když právě nevysílá.




*OBR. 21: TŘÍDY KONCOVÝCH ZAŘÍZENÍ LORAWAN*

Další důležité vlastnosti:

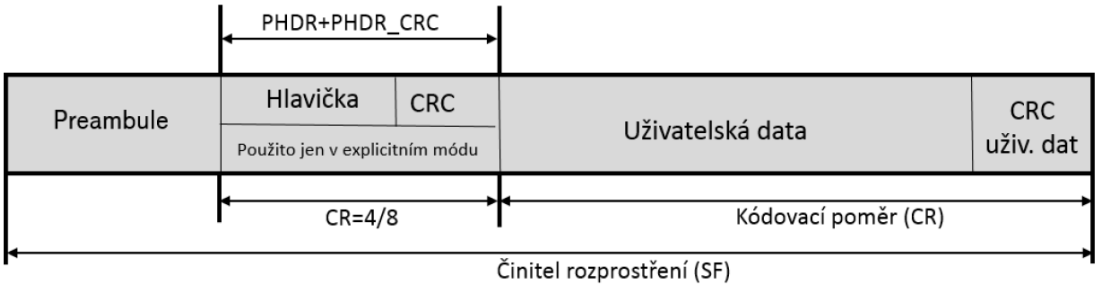
- **Vysílací výkon**: koncové zařízení může být nastaveno v rozsahu -4 dBm až 20 dBm s 1 dB krokem, z důvodu regulatorních požadavků je rozsah většinou limitován na 2 až 14 dBm (parametr ovlivňující přímo úměrně dosažitelnou vzdálenost mezi vysílačem a přijímačem a nepřímo úměrný výdrži baterie)
- **Frekvence:** centrální frekvence v rozmezí 137-1020 MHz s krokem 61 Hz
- **Činitel rozprostření (SF, Spreading Factor):** indikace, kolik bitů je zakódováno do každého symbolu, hodnoty v rozmezí 6-12 (přímo úměrný SNR, tedy i dosažitelné vzdálenosti, ale nepřímo úměrný přenosové rychlosti a výdrži baterie)
- **Šířka pásma (BW, Bandwidth):** může být nastavena v rozmezí 7,8-500 kHz, typicky však 125, 250 nebo 500 kHz (parametr přímo úměrný datové rychlosti a odolnosti vůči interferencím, nepřímo úměrný citlivosti)

- **Kódovací poměr (CR, Coding Rate):** množství nadbytečných dat určených pro dopřednou korekci chyb (FEC, Forward Error Correction), vyšší hodnota představuje lepší odolnost vůči interferencím, ale prodlužuje dobu vysílání zprávy, viz následující tabulka:

| CR | Čas nutný k odeslání zprávy [ms] |
|---|---|
| 1 (4/5) | 123,9 |
| 2(4/6) | 132,1 |
| 4(4/8) | 148,5 |

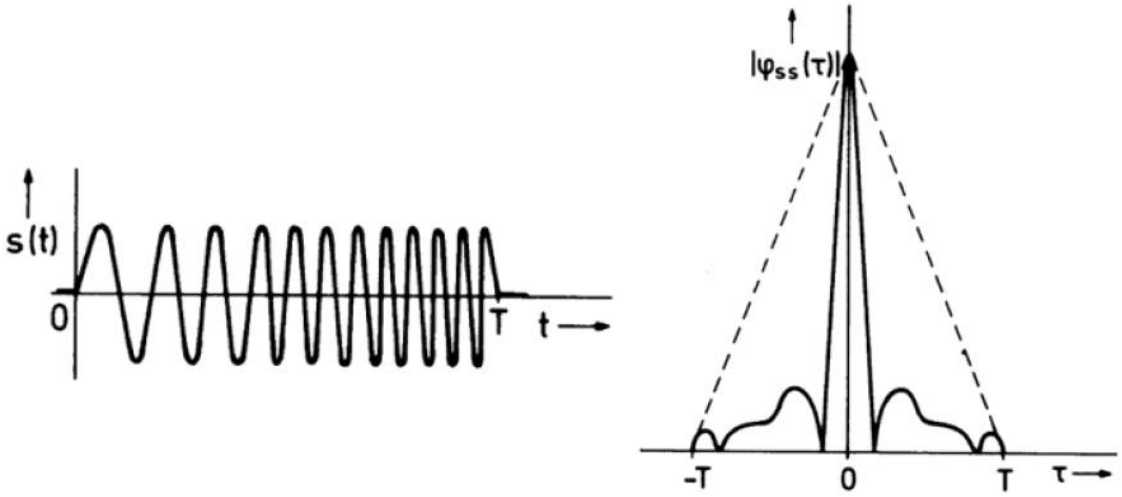
*TAB. 3: HODNOTY KÓDOVACÍCH POMĚRŮ PŘI SF=10 A BW=250KHZ*

**Struktura LoRa paketu** [61]**:** preambule (6-65535 symbolů), volitelná fyzická hlavička (PHDR, LoRa physical header) a 16 b CRC hlavička indikující, zda má být použito CRC uživ. dat (PHDR_CRC, LoRa fyzická Cyclic Redundancy Check hlavička), uživatelská data (1-255 B), volitelná 16 b CRC uživ. dat.




*OBR. 22: STRUKTURA LORAWAN PAKETU KONCOVÉHO ZAŘÍZENÍ. PŘEVZATO Z [59]*

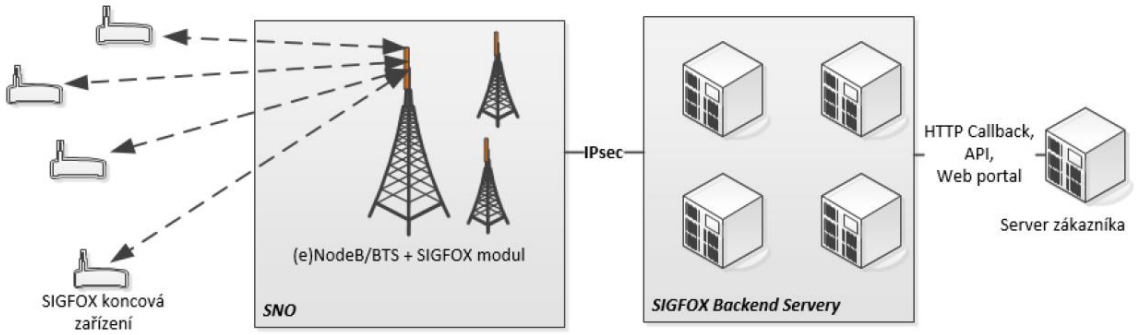
**Modulace** – specifikace umožňuje klíčování frekvenčním posuvem (FSK, Frequency-shift Keying), ale pro snížení chybovosti při větších vzdálenostech je použita technika rozprostření signálu použitím chirp modulace (CSS, Chirp Spread Spectrum) [60]. Tato lineární frekvenční modulace používá binární klíčování a signál s rostoucí frekvencí reprezentující symbol 1 a signál s klesající frekvencí symbol -1 (viz následující obrázek).




*OBR. 23: UKÁZKA CHIRP SIGNÁLU A JEHO AUTOKORELAČNÍ FUNKCE. PŘEVZATO Z [57]*

### 2.1.3 SIGFOX

Společnost SIGFOX je francouzský technologický startup, který byl založen v roce 2009. Ačkoliv je tato technologie považována za proprietární, firma úzce spolupracuje na vytváření LTN (z anglického Low Throughput Networks) standardů definovaných ETSI. Lze proto očekávat i významnou podobnost s budoucím standardizovaným řešením [62]. Za úspěchem SIGFOX technologie lze hledat vyjimečný obchodní model [63]. Síť je postavená na jedno skokové hvězdicové topologii sestávající z koncových zařízení, základnových stanic (tzv. Gateway/base station) a z SIGFOX back-endu. Pokrytí velkého území signálem se provádí pomocí tzv. národních SIGFOX operátorů, kdy se umisťují základnové stanice na věže s vysílači mobilních operátorů [64]. Pro vybudování celoplošné sítě ve Francii řádově stačilo 1000 základnových stanic za cenu 4 miliony USD, což je přibližně 100× méně, než obdobná síť na bázi GSM/CDMA [72].




*OBR. 24: SÍŤOVÁ ARCHITEKTURA*

Technologie

Stejně jako LoRa, i Sigfox využívá bezlicenční pásmo ISM. V Evropě se pracuje na frekvenci kolem 868 MHz, v USA 902 MHz. Technologie využívá úzkopásmového přenosu signálu (UNB, Ultra Narrow Band). UNB technologie nám přináší vysokou citlivost umožňující větší dosah, než standartní technologie, což na druhou stranu ale znamená zvýšení požadavků na přístupové metody kvůli velkému počtu zařízení v jedné buňce. Díky použití úzkého signálu je i hodnota šumu velice nízká, přibližně -150 dBm při 290 K a výkonová bilance spoje může tedy dosáhnout až 160 dB [66][67].

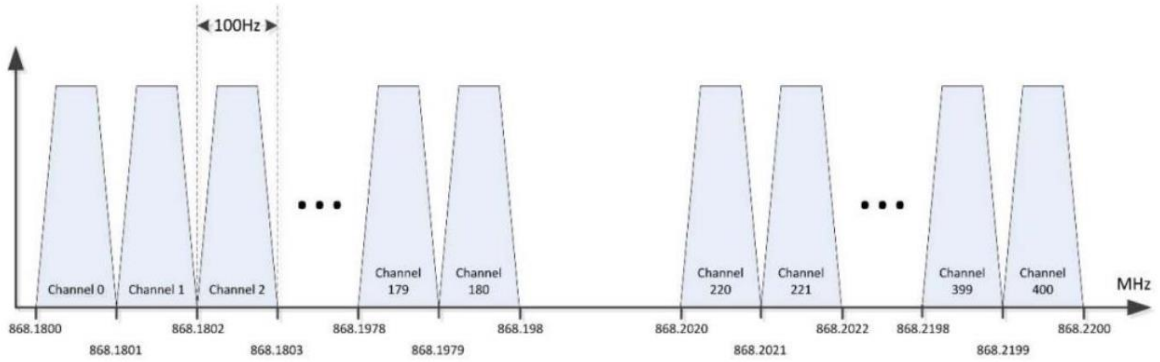
Protokol aktuálně existuje ve dvou základních verzích. Ta první pro uplink používá 48 kHz makro kanál na 868,2 MHz. Druhá verze, z roku 2014, již využívá 192 kHz makro kanál na frekvenci 868,13 MHz. Česká republika jako člen European Conference of Postal & Telecommunications Administrations (CEPT) alokuje tuto část frekvencí pro zařízení s krátkým dosahem (z anglického short range devices, SRD), přesněji pásmo h1 a h3, viz Tabulka 4 [68].

| Označení | Kmitočtové pásmo [MHz] | Vyzářený výkon [mW] | Klíčovací poměr [%] |
|---|---|---|---|
| h | 863,0-870,0 | 25 | < 0,1 nebo použití LBT |
| h1 | 868,0-868,6 | 25 | < 1 nebo použití LBT |
| h2 | 868,7-869,2 | 25 | < 0,1 nebo použití LBT |
| h3 | 869,4-869,65 | 500 | < 10 nebo použití LBT |
| h4 | 869,7-870,0 | 25 | |
| h5 | 869,7-870,0 | 25 | |

*TAB. 4: FREKVENČNÍ REGULACE V PÁSMU 868MHZ DLE ČTÚ*

I když SIGFOX nepoužívá technologii LBT (Listen Before Talk – vysílání pouze po vyžádání na základě příjmu), platí pro technologii omezený klíčovací poměr 1 % s maximálním vyzářeným výkonem (ERP, Effective Radiated Power) 500 mW (27 dBm) pro downlink a 25 mW (14 dBm) pro uplink [69][70].

Pro šířku kanálu 100 Hz je k dispozici tedy teoreticky 480 slotů pro první a až 1920 slotů pro druhou verzi. Někteří však zároveň pro první verzi uvádějí, že kanály 181-219 jsou rezervovány [62]:




*OBR. 25: FREKVENČNÍ ALOKACE KANÁLŮ TECHNOLOGIE SIGFOX. PŘEVZATO Z [62]*

Pro výrobce zařízení je důležitý certifikační proces, který je povinný pro jednotlivá SIGFOX zařízení podle hodnoty efektivního vyzářeného výkonu viz Tabulka 5.

| Třída | 0u | 1u | 2u | 3u |
|---|---|---|---|---|
| ERP [dBm] | 14 > P > 12 | 13 > P > 7 | 7 > P > 0 | P < 0 |

*TAB. 5: SIGFOX CERTIFIKAČNÍ TŘÍDY*

Díky zúžené frekvenční šířce okupovaného pásma lze při dané cílené chybovosti dosáhnout na přijímací straně menšího příspěvku šumu. Modulační technikou použitou pro vzestupný směr (uplink) je diferenciální klíčování fázovým zdvihem DBPSK (Differential Binary Phase-Shift Keying), pro sestupný směr (downlink) pak Gaussovo klíčování frekvenčním zdvihem (GFSK, Gaussian Frequency-Shift Keying) [71]. Z důvodu dodržení regulatorního omezení vysílacího výkonu a vzhledem k požadavku na maximalizaci dosahu byla přenosová rychlost omezena na 100 b/s pro vzestupný směr a pro nově doplněný sestupný směr až 600 b/s [71].

## 2.2 POROVNÁNÍ TECHNOLOGIÍ

Právě technologie LoRa a Sigfox patří k nejčastěji citovaným komerčním řešením. LoRa je unikátní právě tím, že využívá patentově chráněnou modulaci, která rozprostírá signál do celého vysílacího kanálu, ale na rozdíl od tradičních řešení tak dělá lineární změnou (viz XX obrázek chirp modulace). To umožňuje snížení nákladů na lokální oscilátor kvůli menším nárokům na stabilitu kmitočtu. Jednou z největších limitací protokolu je jeho lokální nasazení v rámci jednotlivých firemních řešení. Pokud ale uživatel předpokládá použití napříč rozsáhlým územím, např. mezi více státy, musí zvolit jiné řešení. Alternativa v řešení pokrývající desítky států se dá najít právě v technologii Sigfox. Z teoretického rozboru vyplývá velký potenciál protokolu stát se zároveň plnohodnotným standardizovaným řešením pro LPWAN sítě [75][76]. Ač v současné době probíhá výstavba sítě v řadě států, SIGFOX technologie díky svému kontinuálnímu vývoji a částečné uzavřenosti doposud nebyla uspokojivě prozkoumána či optimalizována.

Shrnutí hlavních parametrů výše zmíněných protokolů je součástí Tabulky č.6.

| | Sigfox | Lora |
|---|---|---|
| Pásmo | ISM <1 Ghz | ISM <1 Ghz |
| Šířka pásma [kHz] | 192 | až 500 |
| Vysílací výkon UL/DL [dBm] | až 14 | až 14/27 |
| Přenosová rychlost příchozí směr [kb/s] | 0,6 | 0,3-50 |
| Přenosová rychlost odchozí směr [kb/s] | 0,1 | 0,3-50 |
| Link budget [dB] | 160 | 154/162 |
| Přístup k médiu | UNB/R-FTDMA | DSSS/TDMA |
| Citlivost BTS/KZ [-dBm] | 160 | 157 |
| Délka zprávy [s] | 2 | 40m-1,2 |
| Zpráva UL/DL [B] | 12-Aug | 250 |
| Max. počet zpráv/den | 144/4 | neomezeno |
| MAC | prostá ALOHA | prostá ALOHA |
| Modulace UL/DL | UNB DBPSK | LoRa CSS |
| Kanál UL/DL [Hz] | 100 | 125000 |
| Fyzický vrstva | UNB | CSS |
| Dosah | až 50 km | až 40 km |
| Velikost payloadu | maximálně 12 bytů | až 255 bytů |
| Zpětný kanál | velmi omezený | ano |
| Potvrzení doručení | velmi omezené | ano |
| Mobilní komunikace | velmi omezená | ano |
| Open source | ne | ano |

*TAB. 6: VÝSLEDNÉ ZHODNOCENÍ TECHNOLOGIÍ*

Porovnání energické náročnosti, údaje pro LoRu získány z F8L10D-N LoRa Module [23], pro Sigfox z Atmel ATA8520D [73]

| | Přijímání [mA] | Vysílání [mA] | Režim spánku s buzením [µA] | Režim spánku [µA] |
|---|---|---|---|---|
| LoRa | 22 | 127-129 | 3 | 2 |
| Sigfox | 10,4 | 32.7 | 50 | 5 |

*TAB. 7: SROVNÁNÍ ENERGICKÉ NÁROČNOSTI*

# 3 REALIZACE APLIKACÍ

Tato část práce se zabývá jednotlivými aplikacemi a vlastním řešením projektů. Jednotlivé části se zaměřují na hardwarové i softwarové požadavky. Především jde o řešení bezdrátového přenosu přes WiFi sítě a síť Sigfox.

## 3.1 ZVOLENÝ MODEL INTERNETU VĚCÍ

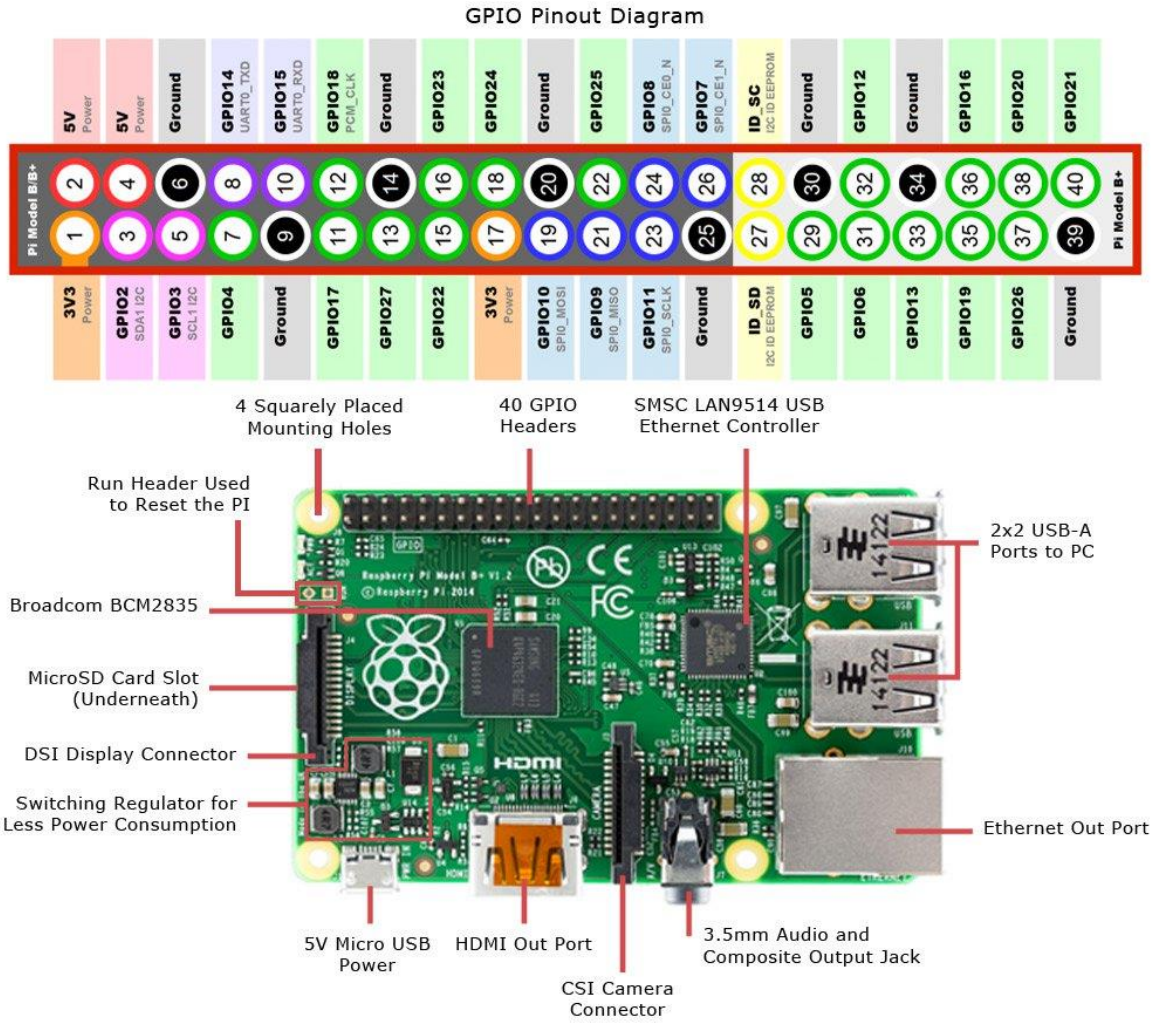
Prvotním námětem byla lokalizace pomocí bezdrátových sítí. V práci jsou řešeny různé možnosti tohoto využití včetně následného zpracování. Nad rámec zadání vznikly doplňující aplikace jako například zveřejnění stavu na Twitteru.

## 3.2 HARDWARE

V této kapitole je popsán použitý hardware. Konkrétně jde o vývojové sady Raspberry Pi 3 model B+ se shieldem Sense Hat a Pytrack s modulem FiPy.

### 3.2.1 RASPBERRY PI 3 A SENSE HAT

Pro základní využití a fundamentální projekty byl využit vývojový kit Raspberry Pi 3 model B+, který poskytuje grafický čip VideoCore IV s taktem 300 MHz, WiFi podporu 2.4 GHz i 5 GHz standardu 802.11ac včetně Bluetooth 4.2. Dále poskytuje 4 porty USB 2.0 a 40 pinový GPIO header, vše na základě ARM Cortex procesoru s frekvencí 1.4 GHz.



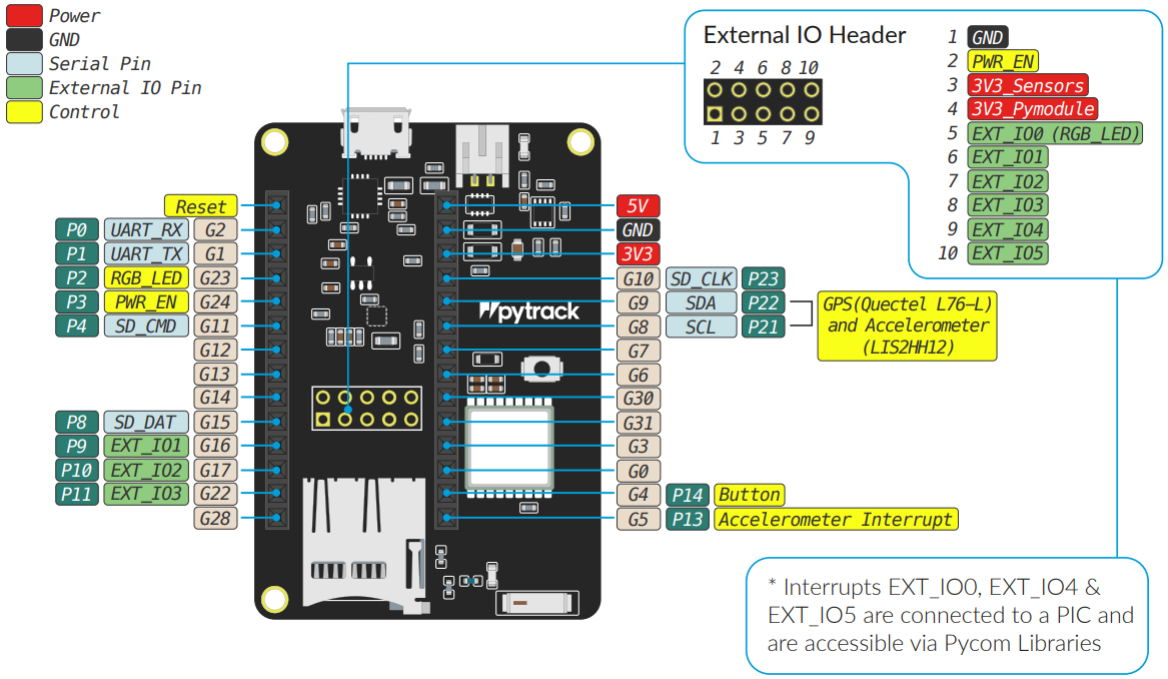

*OBR. 26: PINOUT DIAGRAM RASPBERRY PI 3 MODEL B+*

**Sense Hat** je rozšiřující deska se senzory a 8x8 maticovým displejem. Mezi poskytnutými senzory jsou akcelerometr, gyroskop, magnetometr, teploměr, tlakoměr a vlhkoměr.

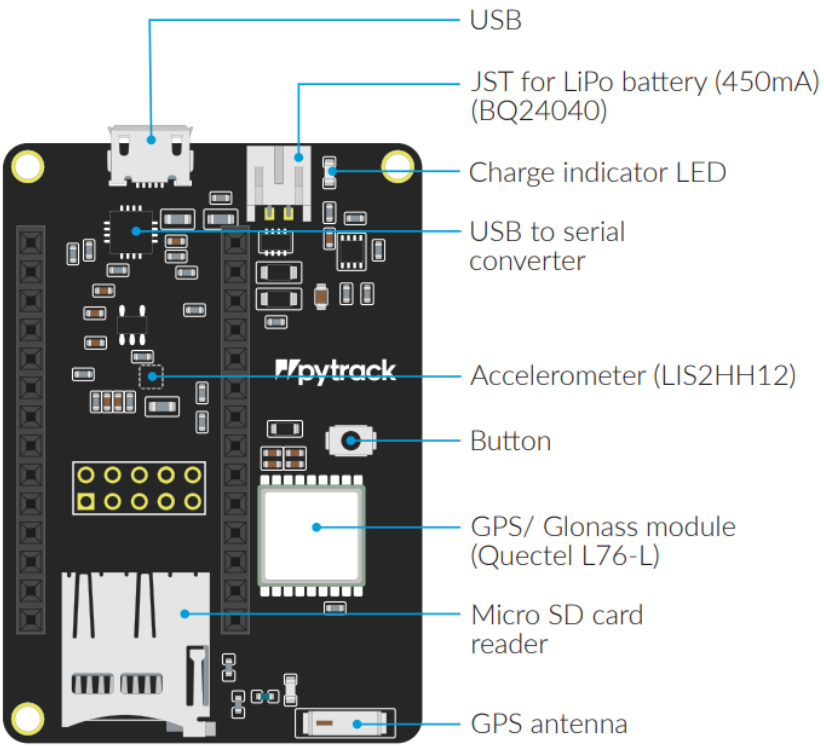

Obr. 27: Zapnutý sense hat

### 3.2.2 PYTRACK S MODULEM FIPY

Pro primární aplikaci byla školou na mojí žádost zakoupena vývojová deska, neboli senzorový štít, Pytrack, od firmy Pycom. Tento kit je poměrně neznámý a nový, proto nebylo jednoduché získat některé informace. V době objednání ani nebyl k prodeji, k dispozici byl o tři měsíce později. Pytrack je deska poskytující možnost využití několika síťových modulů. Využít lze velice přesného lokalizačního systému GNSS s ruským Glonass, trojosého 12 bitového akcelerometru, USB portu s přístupem přes sériový port, LiPo baterii, kompatibilitu s MicroSD kartami a především režim hlubokého spánku při spotřebě kolem 1 µA. Tato extrémně nízká spotřeba byl hlavní důvod k zakoupení. Deska má rozměry 55 x 35 x 10  a dokáže operovat v rozmezí teplot od -40 do +85 °C, váží 11g.

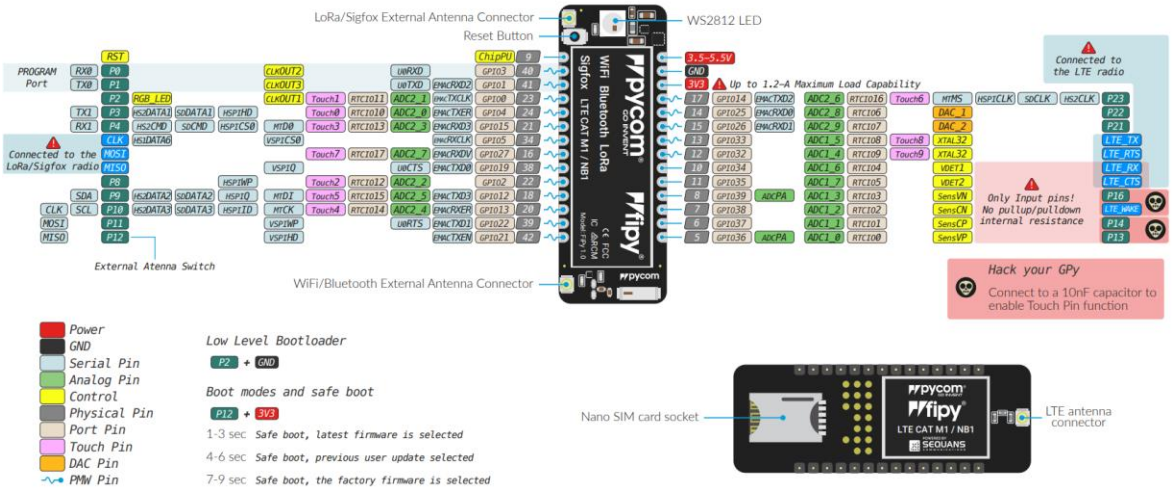



*OBR. 28: PINOUT DIAGRAM DESKY PYTRACK*




Obr. 29: Popis umístění jednotlivých modulů

Modul **FiPy** poskytuje podporu pro pět sítí, konkrétně pro WiFi, Bluetooth, Lora, Sigfox a LTE CAT M1/NB1. Poskytuje tak možnost připojení přes všechny světově komerční LPWAN sítě. Modul má implementován MicroPython. FiPy má velice výkonný, dvou jádrový CPU Xtensa a WiFi dosahuje až jednoho kilometru. Poskytuje 2 x UART, 2 x SPI a podporu micro SD karty. Kódování a dekódování lze provádět pomoci SHA, MD5, DES i AES hashů.




*OBR. 30: PINOUT DIAGRAM MODULU FIPY*

## 3.3 SOFTWARE

V této kapitole je zmíněn software potřebný k vytvoření aplikací. K tvoření na Raspberry byl využit operační systém Raspbian bez dodatečného softwaru. Ke konkrétním projektům byly pouze staženy a nainstalovány určité knihovny, které popisuji u konkrétních aplikací. S deskou Pytrack a modulem FiPy bylo pracováno na operačním systému Windows 10. Nejprve bylo třeba pomocí programu **Zadig** nainstalovat různé drivery pro konkrétní situace, poté updatovat firmware pomocí programu od mateřské firmy Pycom. Tvorba programu probíhala ve vývojovém prostředí **Atom** s pluginem **Pymakr**.

## 3.4 APLIKACE

Bylo docíleno zhotovení několika aplikací na dvou platformách. Zmiňované aplikace jsou podrobně vysvětleny včetně zdrojových kódů.

### 3.4.1 RASPBERRY PI & GOOGLE ASSISTANT

Pro první demonstraci aplikace v rámci internetu věcí byl vytvořen Google Assistant na Raspberry Pi. Google Assistant je v podstatě obdoba známého Amazonu Echo či aplikace Siri na mobilech iPhone. Google Assistant může fungovat na systémech Android či jako provozní program v zařízení Google Nest.

Přes terminál v Raspbianu byla ověřena funkčnost reproduktorů příkazem:

```
speaker-test -t wav
```

Poté bylo třeba ověřit funkčnost mikrofonu. Toho bylo docíleno následujícím příkazem, který nahrával po dobu pěti vteřin a uložil jej pod názven test.raw:

```
arecord --format=S16_LE --duration=5 --rate=16000 --file-type=raw
test.raw
```

Ověření nahrané ukázky příkazem:

```
aplay --format=S16_LE --rate=16000 test.raw
```

Pro možné nastavení přehrávání či nastavení je možno otevřít mixér a nastavit zde vhodné hodnoty, mixér spustíme příkazem:

```
alsamixer
```

Po ověření funkčnosti bylo třeba zaregistrovat Google Assistant API pro získání přístupu. K tomu je zapotřebí již vlastnit účet na Googlu. Pro úsporu prostoru nejsou přiloženy obrázky registrace, jelikož nebudou z podstaty věci zcela zřejmé či z hlediska přehledu a pochopení fundamentální.

Na následující adrese se po přihlášení dostaneme na Dashboard, čili domovskou stránku platformy Cloudu od Googlu.

```
https://console.cloud.google.com/project
```

Po kliknutí na „Create Project“ zadáme jméno projektu a klikneme na „Create“. Po krátké odmlce způsobené tvořením projektu na hlavní stránce klikneme na právě vytvořený projekt, který nasměruje na novou stránku. Zde otevřeme menu a z nabídky vybereme „API's and Service“. Nyní povolíme správné API, nejdříve musíme kliknout na „Enable APIS and SERVICES“ aby se globálně povolili API na tomto účtu. Poté napíšeme do hledacího boxu „Google Assistant“ a z výsledků vybereme „Google Assistant API“. API zaktivujeme a v menu se dostaneme do položky „Credentials“. Zde musíme přepnout do „OAuth consent screen“, zde již bude náš vybrán projekt a klikneme na „Save“. Poté je třeba se vrátit do „Credentials“ a při tvoření údajů vybrat „OAuth client ID“. V nabídce „Application type“ vybereme „Other“ a klikneme na „Create“. Nyní potřebujeme vytvořené údaje stáhnout, čehož docílíme kliknutím na ikonu „Download“. V poslední řadě je ještě třeba přednastavit nastavení v Google Accountu, jelikož nutné služby jsou defaultně vypnuty.

```
https://myaccount.google.com/activitycontrols
```

Zaktivujeme Web & App activity, Location History, Device Information a Voice & Audio Activity.

Poté se vrátíme zpět do terminálu na Raspberry. Nejdříve je třeba updatovat software, toho docílíme zadáním příkazů:

```
sudo apt-get update
sudo apt-get upgrade
```

Jakmile jsou procedury hotovy, máme vše připravené. Vytvoříme složku „googleassistant“ a v ní soubor „credentials.json“, který rovnou otevřeme k editaci:

```
mkdir ~/googleassistant
```

```
nano ~/googleassistant/credentials.json
```

Nyní otevřeme stažený soubor z minulých kroků a překopírujeme údaje z tohoto souboru právě sem. Uložíme stisknutím Ctrl + X, Y a potvrdíme enterem. Poté je třeba nainstalovat Python3 a Python 3 Virtual Environment:

```
sudo apt-get install python3-dev python3-venv
```

Následně povolíme python3 jako naše virtuální prostředí:

```
python3 -m venv env
```

Než pokročíme k samotnému asistentu, je třeba se ujistit o přítomnosti nejaktuálnější verze **pip** a **setuptools**.

```
env/bin/python -m pip install --upgrade pip setuptools --upgrade
```

Abychom se dostali do prostředí Pythonu, zadáme:

```
source env/bin/activate
```

Nyný, když máme připraveny všechny věci k instalaci knihovny Google Assistant, spustíme samotnou instalaci:

```
python -m pip install --upgrade google-assistant-library
```

Abychom nejdříve mohli pustit Google Assistanta, je třeba nainstalovat utilitu pro autorizování.

```
python -m pip install --upgrade google-auth-oauthlib[tool]
```

Dále je třeba získat autentizační údaje, po zadání následujícího příkazu dostaneme URL odkaz, který je třeba otevřít v prohlížeči:

```
google-oauthlib-tool --client-secrets
~/googleassistant/credentials.json --scope
https://www.googleapis.com/auth/assistant-sdk-prototype --save --
headless
```

Na této adrese se přihlásíme pod svým účtem a dostaneme k dispozici kód, který je třeba zkopírovat a vložit do terminálu. Pokud je procedura provedena správně, mělo by se objevit v terminálu „credentials saved: …“

Nyní spustíme aplikaci a otestujeme:

```
google-assistant-demo
```

Do mikrofonu proneseme „Ok Google“ nebo „Hey Google“ a proneseme dotaz. V mém případě jsem vznesl do mikrofonu „Where is West Bohemia University?“. Odpovědí je kompletní adresa na borech.

```
ON_CONVERSATION_TURN_STARTED // zapnutí asistenta na pokyn „Hey Google“

ON_END_OF_UTTERANCE // oznámení konce dotazu

ON_RECOGNIZING_SPEECH_FINISHED:
{'text': 'where is West Bohemia University'} // rozeznaný text

ON_RESPONDING_STARTED:
{'is_error_response': False} // oznámení, že nevznikl error během algoritmu
a vyslovení odpovědi na dotaz

ON_RESPONDING_FINISHED // oznámení konce odpovědi

ON_CONVERSATION_TURN_FINISHED:
{'with_follow_on_turn': False}
```

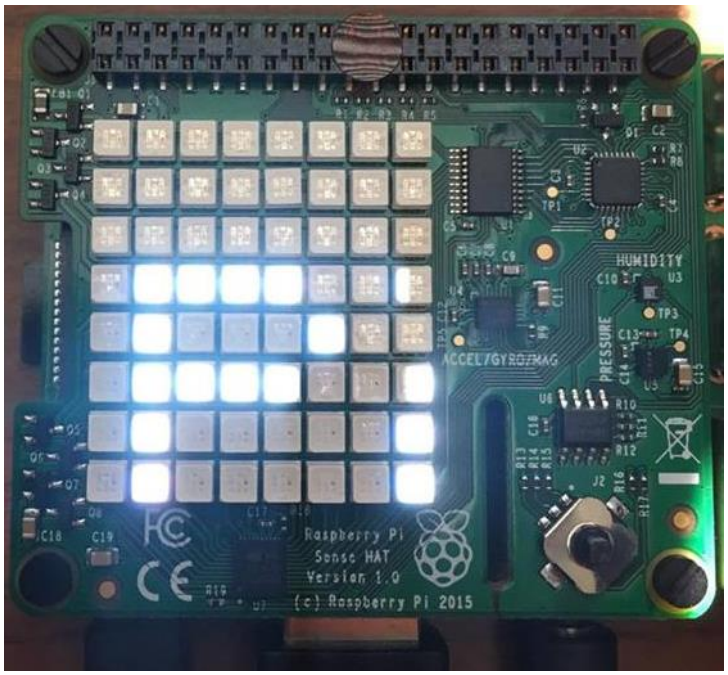
Toto je ukázka zcela jednoduché aplikace s využitím cloudových služeb a API od Googlu.

V dalším případě byla vytvořena meteo stanice s využitím modulu **Sense Hat** a cloudové služby **Initial State**, která kromě cloudových služeb poskytuje také uložiště a analýzu dat. Některé stejné kroky jako v předchozím projektu, proto již nejsou vysvětleny.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install sense-hat
sudo reboot
sudo nano ~/sensehat_test.py
```

Dojde k otevření nového okna, kam zadáme následující příkazy, které budou znamenat, že se na maticové obrazovce modulu Sense Hat proběhne text „Diplomová práce“:

```
from sense_hat import SenseHat
sense = SenseHat()
sense.show_message("Diplomová práce")
```




Obr. 31: Ukázka z probíhající textu „diplomová práce“

Abychom mohli využít služeb Initial State, je třeba si nejdříve vytvořit účet a poté se přihlásit.

```
www.initialstate.com
```

Na Dashboard stránce klikneme na náš účet a v dolní části stránky v kolonce „Streaming Access Keys“ klikneme na „Create a New Key“. Nyní nainstalujeme Initial State python streamer do našeho zařízení přímo ze stránek Initial State:

```
curl -sSL https://get.initialstate.com/python -o - | sudo bash
```

Nyní nastavíme stanici. Vytvoříme soubor weather_script.py, do kterého vložíme knihovnu sense_hat, pomocí které komunikujeme se samotnou deskou Sense Hat a čteme data. Knihovna time poskytuje mnoho funkcí, zde ale využijeme pouze funkci sleep. Knihovna sys nám dává přístup k několika proměnným a funkcím, které jsou poskytovány samotnou deskou. Knihovna Streamer nám poskytuje možnost utvoření streamu na cloud.

```
sudo nano ~/weather_script.py
#!/usr/bin/python
from sense_hat import SenseHat
import time
import sys
from ISStreamer.Streamer import Streamer
```

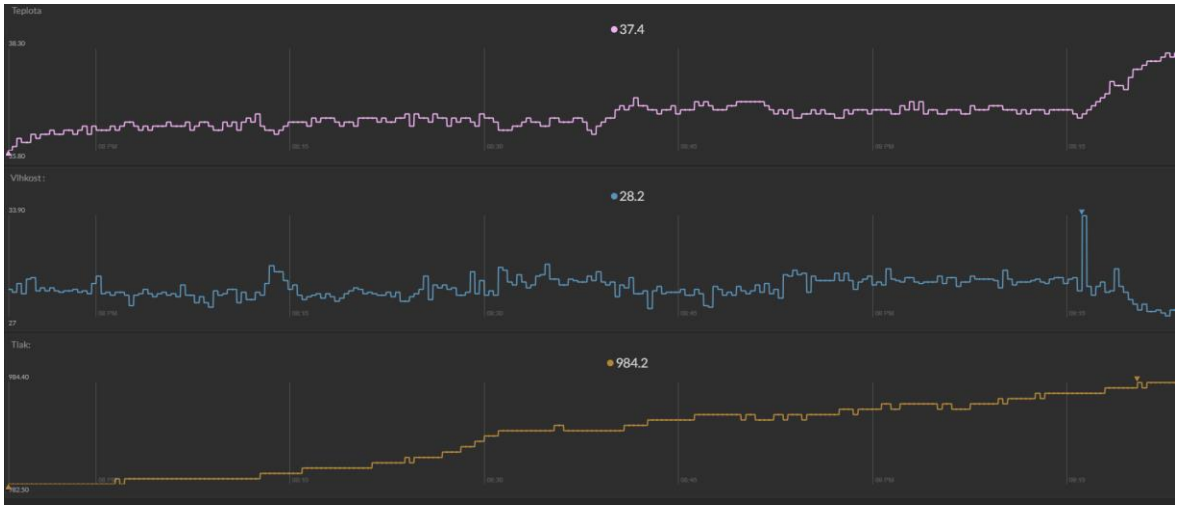
Pokračujeme:

```
sense = SenseHat() // Inicializace knihovny

logger = Streamer(bucket_name="Sense Hat Sensor Data", access_key="
iMcZZvBAjWPsawjKAuwjQxRJydJsKudR") // zde nutno zadat svůj klíč
vygenerovaný na stránkách Initial State v předchozích krocích

sense.clear() // nastavení LED matice do základních hodnot
```

Jádro scriptu:

```
try:

      while True:
            temp = sense.get_temperature()// získání teploty

            temp = round(temp, 1) // nastavení do stupnice Celsia

            logger.log("Teplota:",temp) // nalogování výsledku do Cloudu

             humidity = sense.get_humidity()

             humidity = round(humidity, 1)

             logger.log("Vlhkost:",humidity)

             pressure = sense.get_pressure()

             pressure = round(pressure, 1)

             logger.log("Tlak:",pressure)

            time.sleep(1) // "uspání" skriptu na jednu vteřinu
```

Tímto jsme spustili script. Na dashboardu v Initial State můžeme vlevo vidět přenos dat. Po kliknutí na tuto položku máme na výběr možnost zpracování dat, vybereme Tiles a nastavíme si okna tak, jak vyhovují. Hodnoty zde jsou streamované, tudíž nejaktuálnější možné dle zpracovaného kódu.




*OBR. 32: UKÁZKA VÝSLEDKŮ Z INITIAL STATE*

Další možností je spuštění skriptu automaticky při zapnutí zařízení. K tomu je třeba balíku dos2unix a inicializačního souboru, do kterého se vloží kód následující po příkazech:

```
sudo apt-get install dos2unix

sudo nano /etc/init.d/weatherstation
```

```
#!/bin/bash
### BEGIN INIT INFO
# Provides:          weatherstation
# Required-Start:
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start/stops the weatherstation
# Description:       Start/stops the weatherstation
### END INIT INFO

DIR=/home/pi
DAEMON=$DIR/weather_script.py
DAEMON_NAME=weatherstation

DAEMON_USER=root

PIDFILE=/var/run/$DAEMON_NAME.pid

. /lib/lsb/init-functions

do_start () {
    log_daemon_msg "Starting system $DAEMON_NAME daemon"
    start-stop-daemon --start --background --pidfile $PIDFILE --make-pidfile --
user $DAEMON_USER --chuid $DAEMON_USER --startas $DAEMON
    log_end_msg $?
}
do_stop () {
    log_daemon_msg "Stopping system $DAEMON_NAME daemon"
    start-stop-daemon --stop --pidfile $PIDFILE --retry 10
    log_end_msg $?
}

case "$1" in
    start|stop)
        do_${1}
        ;;

    restart|reload|force-reload)
        do_stop
        do_start
        ;;

    status)
        status_of_proc "$DAEMON_NAME" "$DAEMON" && exit 0 || exit $?
        ;;
    *)

        echo "Usage: /etc/init.d/$DAEMON_NAME {start|stop|restart|status}"
        exit 1
        ;;
esac
exit 0
```

Soubor uložíme a spustíme soubor jako dos2unix.

```
sudo dos2unix /etc/init.d/weatherstation

sudo chmod 755 /home/pi/weather_script.py

sudo chmod +x /etc/init.d/weatherstation // navýšíme práva

sudo update-rc.d weatherstation defaults // vytvoříme spojení mezi bash
scriptem a rc.d složkami

sudo service weatherstation start // spustíme službu
```

Další možností využití je zveřejňování stavů na **Twitteru**. To je s následujícím programem možné jak na kitu Raspberry Pi, tak i na Pytracku.

Nejdříve je třeba podobně jako v předchozích případech API údajú. Konkrétně Twitter API nám umožní využití REST rozhraní. Podobně jako v ostatních případech, i zde je třeba účtu. Zde následuje vytvoření aplikace v rámci Twitteru. Začneme tedy na následujícím odkazu:

```
https://apps.twitter.com/app/new
```

Vyplníme údaje dle uvážení, kolonku Callback URL necháme prázdnou. Poté klineme na „Create your Twitter application“ a na následující stránce zamíříme na „Keys and Access Tokens“. Zde si uložíme k pozdějšímu použití „Consumer Key (API Key)“ a „Concumer Secret (API Secret). Po uložení klikneme na „Create my access token“ a následující údaje si taktéž uložíme. Následuje sekvence příkazů:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python-setuptools
sudo easy_install pip
sudo pip install twython
mkdir ~/twitterbot
cd ~/twitterbot
sudo nano TwitterBot.py
```

Dojde k otevření okna souboru TwitterBot.py, kam vložíme následující kód:

```
#!/usr/bin/env python
import os
from twython import Twython

CONSUMER_KEY = 'brrqDM5Q4EMnTDieaYnzXd5aA'
CONSUMER_SECRET = 'YQwqBivMst19drIV38mdc2RlK0ha37Vw1lY0ZnEvCKjavGGZMu'
ACCESS_KEY = '2278088191-mVufxGlxRgqYsCqlSMCevpHmeqPEfQkCL20bDHz'
ACCESS_SECRET = '9lJAJPOMN2WANkU9pO1FHmtCm1U4IIJiSsaUXSrzh8Qli'

api = Twython(CONSUMER_KEY,CONSUMER_SECRET,ACCESS_KEY,ACCESS_SECRET)

cmd = '/opt/vc/bin/vcgencmd measure_temp'
line = os.popen(cmd).readline().strip()
temp = line.split('=')[1].split("'")[0]
api.update_status(status=Současná teplota CPU je: '+temp+' C')
```

Zde je uveden konkrétní příklad se mými údaji. Následuje přiřazení práv a spuštění skriptu:

```
sudo chmod +x TwitterBot.py
python TwitterBot.py
```

Výsledek můžeme vidět níže.

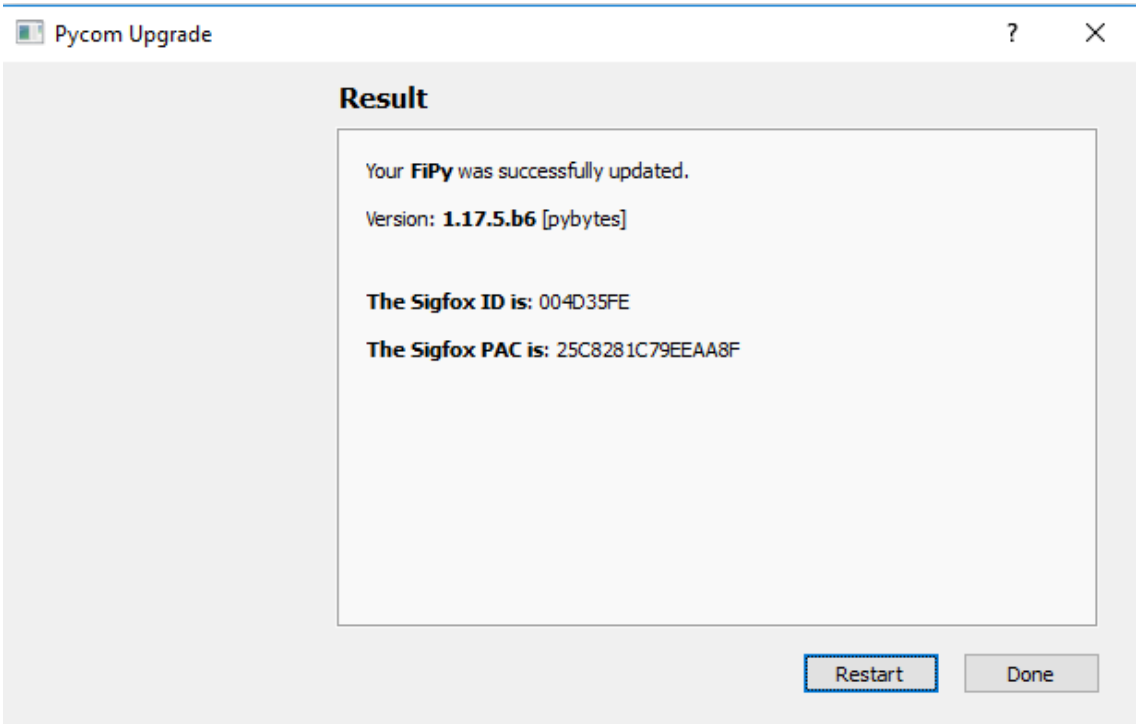

Obr. 33: Ukázka propojení RPi a twitteru

### 3.4.2 PYTRACK S FIPY

Pro tvoření aplikací na **Pytracku** bylo nejdříve třeba poněkud složitěji aktualizovat firmware. Popsaný postup je již ověřený a spolehlivě funguje, nicméně v oficiálních dokumentech je popsán nedostatečně a špatně.

Postup aktualizace firmwaru:

1. Nejprve fyzicky připneme FiPy modul do GPIO headeru desky Pytrack
2. Připravíme si redukci microUSB-USB, která nebyla součástí balíku
3. Spustíme program Zadig, v Options zaškrtneme „List all devices“
4. Připojíme kabel k Pytracku, na desce najdeme resetovací tlačítko.
5. Zmáčkneme a podržíme resetovací tlačítko a zároveň připojíme desku k PC, stále je třeba držet reset
6. V nabídce programu Zadig se objeví nové zařízení „Unknown device #1”, na to klikneme a rychle najdeme v nabídce USB Serial (CDC), následně klikneme na Replace Driver, poté můžeme pustit resetovací tlačítko
7. Windows se nyní připojí na Pytrack přes sériový port. Jaký konkrétní port můžeme zjistit přes Device Manager, záložka Ports (COM & LPT). V mém případě jde o port COM6
8. Z oficiálních stránek si stáhneme updater. V mém případě byl stažen pycom_firmware_update_1.14.4 a spustíme.
9. Program nainstalujeme a spustíme
10. První okno má informační charakter, informuje o nejaktuálnější verzi a případně přiloží i odkaz, ze které se dá nejnovější update stáhnout. Klikneme dvakrát na „Continue“, ve třetím okně vybereme správný COM port a typ firmwaru vybereme pybytes. V dalším okně proběhne samotná instalace firmwaru. Na konci této instalace dostaneme **Device ID** a **PAC number, což jsou fundamentální údaje pro síť Sigfox!** Viz. Následující obrázek.




Obr. 34: Potřebné údaje pro práci se sítí sigfox

Pokud bychom na desku nenahráli nejnovější firmware, nedostali bychom unikátní kódy do sítě Sigfox a nemohli tak zařízení registrovat. Jelikož není platforma Sigfoxu open-source, je třeba zařízení registrovat. Registrace nebývá zadarmo a v České republice je provozována společností SimpleCell Networks a.s. V tomto případě ale není externí registrace nutná, jelikož spolu firmy Pycom a Sigfox spolupracují a jejích zařízení jsou již předregistrovány. Ke kompletní registraci postačí vyplnění registračního formuláře na této stránce:

`buy.sigfox.com/activate`

Zde vybereme Českou republiku a klikneme na „Next“, zde poskytneme námi již získaný Device ID a PAC Number a základně popíšeme projekt. V dalších oknech si vytvoříme účet a potvrdíme jej. Následně se přihlásíme a začneme pracovat na této adrese:

`backend.sigfox.com/welcome/news`

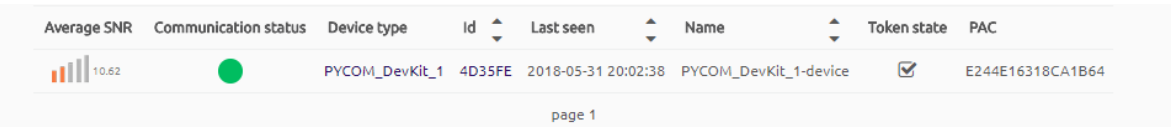
Nyní je třeba nastavit primární věc, a to Keep-alive parametr. Ten je totiž defaultně nastaven na 0 a to znamená vypnutí komunikace. V záložce „Device Type“ klikneme na „PYCOM_DevKit_1“ pod kolonkou „Name“. V následujícím okně klikneme na „edit“ a dále změníme právě parametr Keep-alive na 30.

Stejně jako v případě Raspberry PI, i zde jsou uvedeny fundamentální projekty až po složitější konstrukce.

**První** aplikace je klasické poslání zprávy „Hello World“ přes síť Sigfox na jejich cloud. Abychom mohli využít konektivity Sigfoxu, je **nutné připojení antény tomu konstruované, jinak může dojít k nevratnému poškození desky**. Projekt byl napsán v programu Atom s pluginem Pymakr. Instalace těchto programů a doplňků není považována za téma této práce a dá se najít spoustu návodů na internetu, proto je jejich instalaci přeskočena. V globálních podmínkách nastavíme připojení přes COM6, ostatní necháme. Po kliknutí na Connect se připojíme k zařízení. Vytvoříme novou složku a v ní nový soubor pojmenovaný main.py, který následně otevřeme pro editaci a vložíme následující kód:

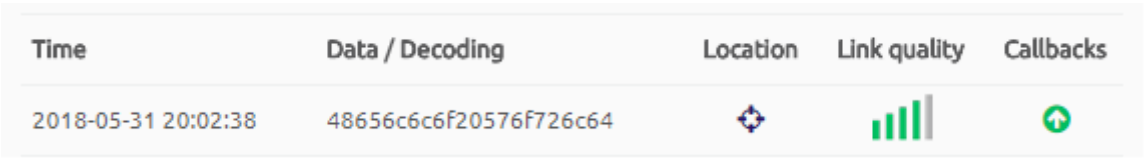
```
from network import Sigfox
import socket

# inicializace Sigfoxu pro evropský region (RCZ1)
sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ1)

# vytvoření Sigfox socketu
s = socket.socket(socket.AF_SIGFOX, socket.SOCK_RAW)

# následné uvedení socketu do blokujícího stavu
s.setblocking(True)

# nastavení socketu pouze jako uplink
s.setsockopt(socket.SOL_SIGFOX, socket.SO_RX, False)

# poslání zprávy
s.send("Hello World")
```

Program nahrajeme na desku kliknutím na „Upload“ a následně pustíme přes „Run“. Program nyní pošle zprávu „Hello World“ přes síť Sigfox. K oveření využijeme backend Sigfoxu, kde by se pod kolonkou „Device“ měla změnit barva „Communication status“ na zelenou.




*OBR. 35: AKTUÁLNÍ STAV ZAŘÍZENÍ V PŘIPOJENÍ SE SIGFOXEM*

V dalším kroku klikneme na ID zařízení a v levém menu vybereme „Messages“. Zde se nachází přehled všech poslaných, či přijatých zpráv.




Obr. 36: Ukázka odeslané zprávy

Jako obsah zprávy je uvedeno „48656C6C6F20576F726C64“. To je způsobeno tím, že Sigfox automaticky dekóduje zprávy jako hexadecimální a v tomto formátu jej posílá dále. Správnost zprávy lze ověřit v jakémkoli konvertoru hexadecimální zprávy na formát string.


Obr. 37: Převedení zprávy zpět do stringu

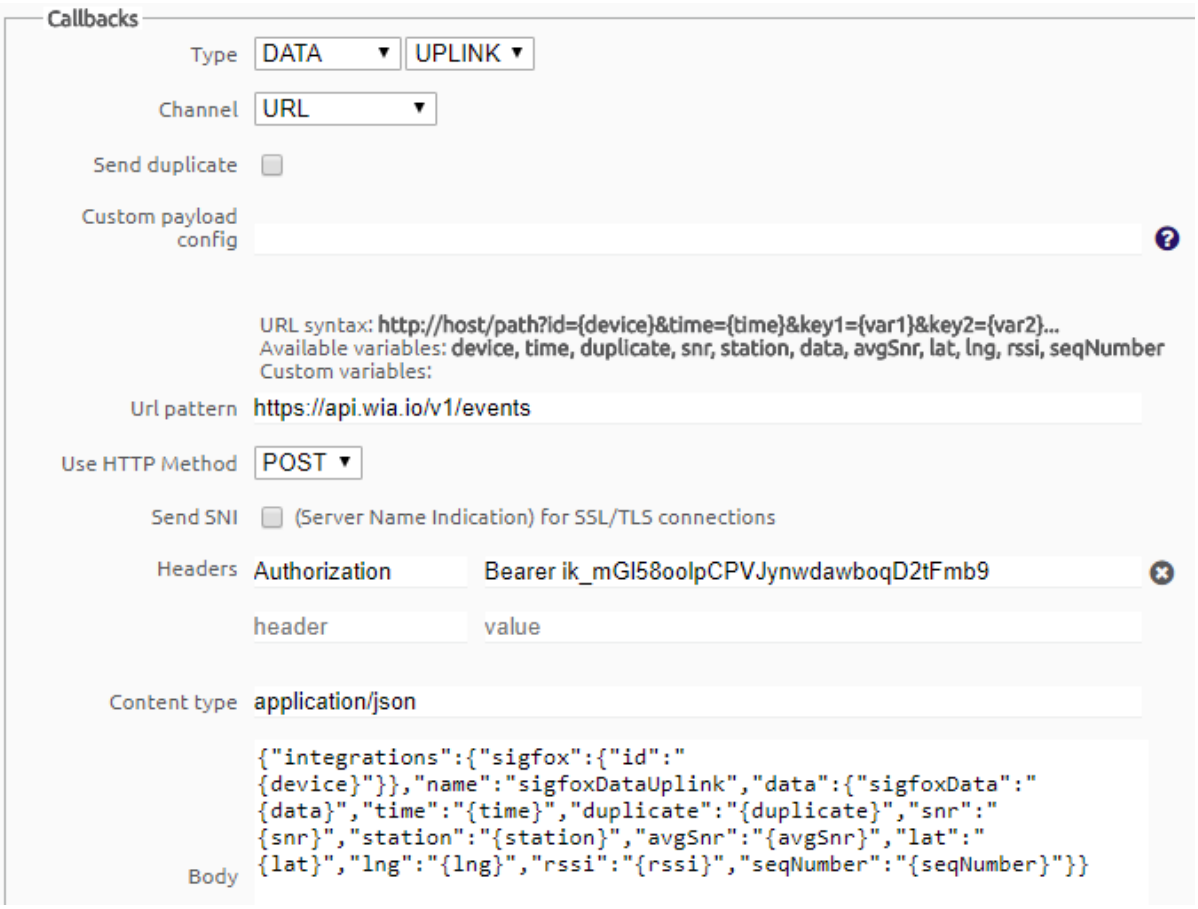
V **druhém** projektu je již zahrnut přenos na jinou cloudovou službu. V následující ukázce je využito služby **Wia**, které je charakteristická rozmanitou podporou zařízení a programovacích jazyků. Nejprve je třeba vytvoření účtu:

```
dashboard.wia.io/signup
```

Po vytvoření účtu vytvoříme „Space“. Po vytvoření jdeme do „Settings“ a klikneme na „Integrations“. Zde nastavíme integraci se Sigfoxem kliknutím na „Setup“. Wia po nás nyní chce API Login a API Password, to získáme z dashboardu Sigfoxu. Jdeme postupně do „Group“ – „Název skupiny (v mém případě West Bohemia University)“-„Api Access“. Abychom vygenerovali potřebné klíče, klikneme na „New“ v pravém horním rohu, zadáme „Wia“ jako jméno a práva přiřadíme „DEVICE MANAGER [R]”. Klíče zkopírujeme, vložíme a klikneme na „Create Integration“. V dalším kroku je třeba nastavit Sigfox callback. Tedy dorozumívací funkci mezi cloudy. V sigfoxovém dashboardu vybereme „Device Type“ a klikneme na jméno zařízení. V levém menu vybereme „Callbacks“ a vytvoříme nový.

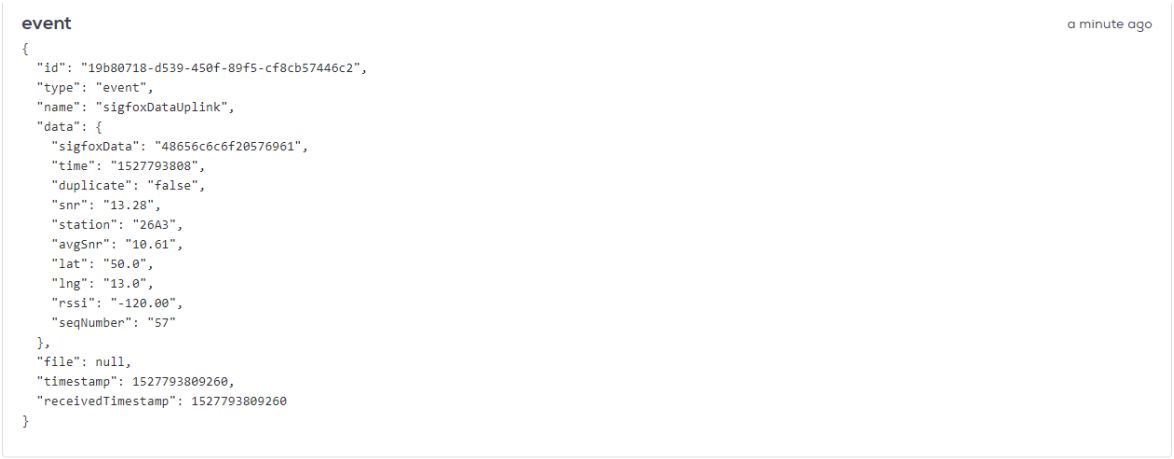
Nastavíme podle následujících parametrů:

- Type: DATA UPLINK
- Channel: URL
- Url pattern: https://api.wia.io/v1/events
- Use HTTP method: POST
- Headers
- header: Authorization
- value: Bearer [YOUR INBOUND KEY]
- Note: inbound klíč ik_
- Content Type: application/json
- Body:{"integrations":{"sigfox":{"id":"{device}"}},"name":"sigfoxDataUplink","data":{"sigfoxData":"{data}","time":"{time}","duplicate":"{duplicate}","snr":"{snr}","station":"{station}","avgSnr":"{avgSnr}","lat":"{lat}","lng":"{lng}","rssi":"{rssi}","seqNumber":"{seqNumber}"}}




Obr. 38: Nastavení callbacku pro wiu

Poté se zpět ve Wie automaticky vytvoří zařízení ve skupině “Devices”, toto zařízení vybereme, z horního menu vybereme “Debugger” a spustíme program. Ve Wie vyskočí v tomto okně “event” s následujícími údaji:



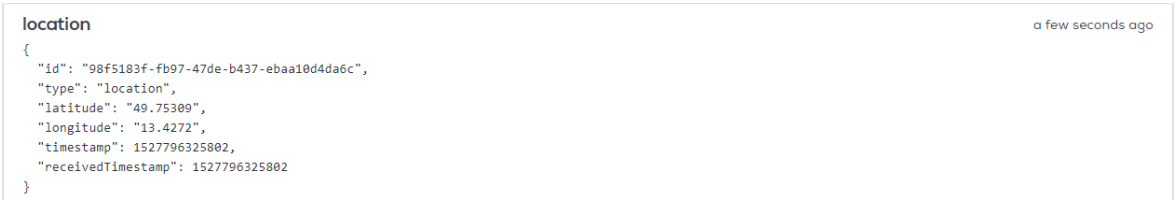

*OBR. 39: ODEZVA CALLBACKU VE WIE*

Kde zpráva “Hello Wia” je přenesena v proměnné “sigfoxData” opět v hexadecimální soustavě s hodnotou 48656c6c6f20576961. Tímto projektem jsme dokázali pomocí Sigfox sítě přenést informaci do externího cloudu. Se zpracováním informace se zabývám v dalším projektu.

Ve **třetím** projektu je třeba využít knihoven. Proto si vytvoříme složku lib a do ní vložíme soubory L76GNSS.py, LIS2HH12.py, pycoproc.py, pytrack.py a urequests.py. Sobory jsou v příloze níže. V projektu jsou získány aktuální souřadnice zařízení a přes rozhraní WiFi poslány na cloudovou službu Wia, ze které jsou následně poslány informace o poloze na mobilní telefon přes stejnojmennou aplikaci volně ke stažení. Kód vypadá následovně:

```
from network import WLAN
from pytrack import Pytrack
import urequests as requests
from L76GNSS import L76GNSS
import socket
import time
import pycom
py = Pytrack()
gps = L76GNSS(py, timeout=30)
# WIFI údaje
WIFI_SSID = 'bubilek_upc'
WIFI_KEY = 'kesskess1'
# Secret Key z Wia dashboardu
DEVICE_SECRET_KEY = 'd_sk_RNpmOe0n1dgEKPVCMIZEBT6u'
wlan = WLAN(mode=WLAN.STA)
nets = wlan.scan()
# pripojení k WIFI
for net in nets:
    if net.ssid == WIFI_SSID:
        print('Sit nalezena!')
        wlan.connect(net.ssid, auth=(net.sec, WIFI_KEY), timeout=5000)
        print('Pripojuji...')
        while not wlan.isconnected():
            machine.idle() #
        print('WLAN pripojeni uspesne!')
        break
# poslani eventu do Wii
def post_location(latitude, longitude):
    try:
        url = "https://api.wia.io/v1/locations"
        headers = {"Authorization": "Bearer " + DEVICE_SECRET_KEY, "Content-Type": "application/json"}
        json_data = {"latitude": str(latitude), "longitude": str(longitude)}
        if json_data is not None:
            req = requests.post(url=url, headers=headers, json=json_data)
            if req.status_code is not 200:
                machine.reset()
            else:
                print(json_data)
            return req.json()
        else:
            pass
    except:
        pass
lat = 53
lng = -6
# main loop
while True:
    # Získání souradnic z pytracku
    coord = gps.coordinates(debug=True)
    if not coord == (None, None):
        lat, lng = coord
    post_location(lat, lng)
    time.sleep(15)
```

V tomto případě jde o zaslání struktorovaných json dat přímo na cloud Wii přes rozhraní WiFi, proto zde není žádné volání Sigfoxu. Na Wiu dorazí data v následující struktuře:

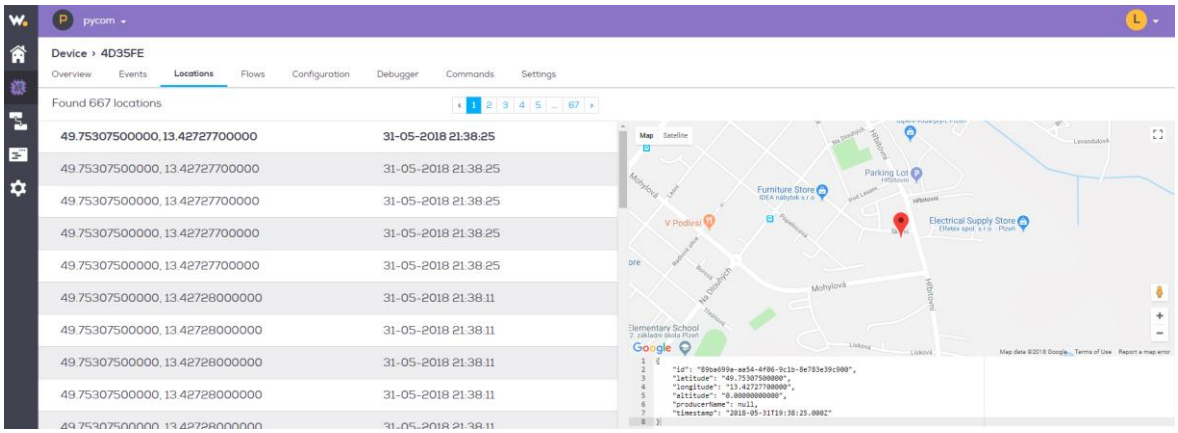



Obr. 40: Struktura lokalizace ve Wie

Jako další krok je vytvoření Flow. Ve Flow studio nyní potřebujeme přetáhnout „Location“ z nabídky „Triggers“ na kanvas. Poté přetáhnemé z nabídky „Actions“ položku „Notification“ a tyto položky spojíme. V konfiguraci „Location“ zaškrtneme naše zařízení a do konfigurace „Notifications“ vložíme následující:

```
GPS souřadnice: ${input.body.latitude}, ${input.body.longitude}.
```

Výsledkem programu je oznámení o poloze na mobilní telefon každých 15 vteřin.




*OBR. 41: LOKACE VE WIE*


Obr. 42: Oznámení na mobilním telefonu

Ve **čtvrtém** projektu šlo o stejný princip, ale s využitím sítě Sigfox. Zde byl opět využit již dříve využitý callback. V programu je jen úprava nahrazení připojení k WiFi síti a poslání dat na Wiu tímto blokem:

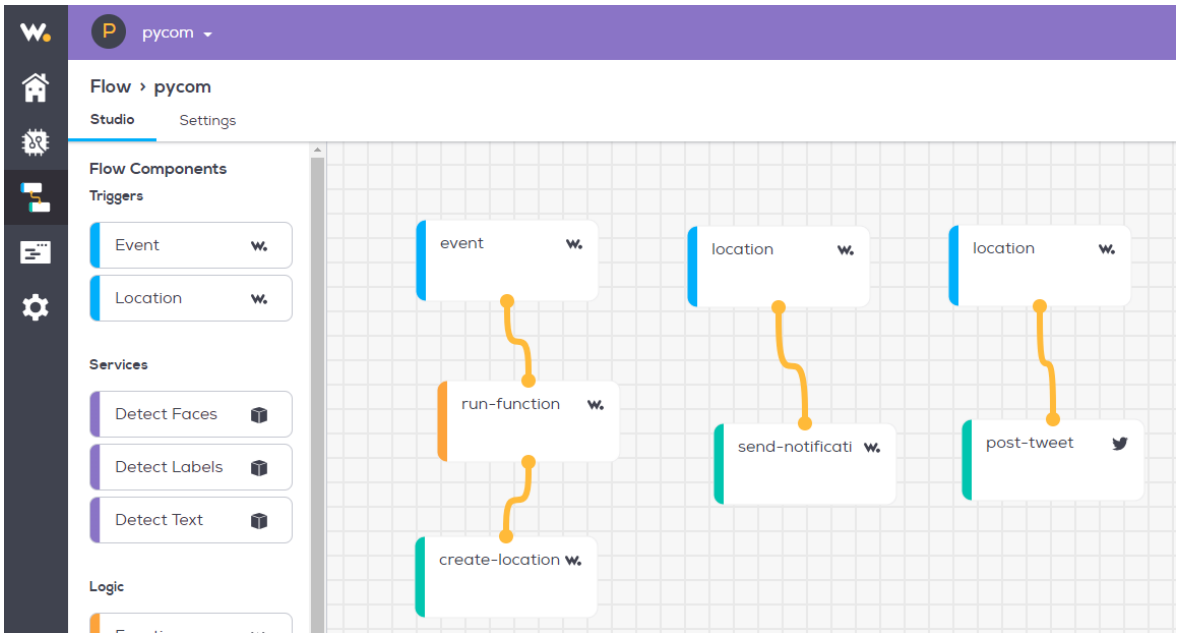
```
print("pripojovani do Sigfoxu")
# init Sigfox for RCZ1 (Europe)
sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ1)
s = socket.socket(socket.AF_SIGFOX, socket.SOCK_RAW)
s.setblocking(True)
s.setsockopt(socket.SOL_SIGFOX, socket.SO_RX, False)

# Poslani lokace do Wii pres Sigfox backend
def post_location(latitude, longitude):
    try:
        print(str(latitude), ":", str(longitude))
        s.send(struct.pack('f',float(latitude)) +
struct.pack('f',float(longitude)))
    except:
        pass
```

Jedná se zde o specifické poslání lokace, jelikož Sigfox má omezenou délkou zprávy, je třeba informaci vhodně strukturovat a dekódovat. S kódováním a dekódováním mi pomohl hlavní vývojář Wii. V programu se lokace převede ze stringu do floatu a Sigfox jej nadále převádí do hexadecimálního tvaru přímo, nikoli přes UTF kódování. To je podstatná informace vzhledem k dekódování, které je provedeno přes javascript v samotné Wie. Aby mohl program fungovat, je třeba upravit Flow. Z „Triggers“ nabídky přetáhneme na kanvas „Event“. Ten je třeba pojmenovat „sigfoxDataUplink“, neboť je tak pojmenován v callbacku a vybereme naše zařízení. Dále je třeba „Function“ z nabídky „Logic“. Právě v této funkci proběhne zpětné dekódování, do konfigurace zadáme kód:

```
if (input.body) {
  let latLong = parseSigfoxLocation(input.body.data.sigfoxData);
  output.body.latitude = latLong.latitude;
  output.body.longitude = latLong.longitude;
}

function parseSigfoxLocation(sigfoxData) {
  let latHex = sigfoxData.slice(0, 8);
  let longHex = sigfoxData.slice(8);
  let result = {
    latitude: Buffer(latHex, 'hex').readFloatLE(0).toFixed(6),
    longitude: Buffer(longHex, 'hex').readFloatLE(0).toFixed(6)
  };

  return result;
}
```

Data ze Sigfoxu přijdou ve formátu 12 bajtového hexadecimálního stringu, každá z GPS souřadnic má 4 bajty dat. Jsou to dva znaky, které se vejdou do 1 bajtu dat, proto jsme rozdělili hexadecimální string osmi. Poté jsme převedli hex na formát float little-endian. Výsledkem je správné dekódování polohy. Poté přidáme blok „Location“ z podnabídky „Actions“. Eventuelně je možnost přidat posílání lokace na Twitter přidáním „Location“ z nabídky „Triggers“ a „Tweet“ z nabídky „Actions“, kterou nakonfigurujeme stejně jako položku „Notification“. Výsledkem je opět upozornění o poloze na telefon každých 15 vteřin, nyní obohaceno o zveřejnění stavu na Twitteru.


*OBR. 43:OZNÁMENÍ NA TWITTERU*




*OBR. 44: POHLED NA FINÁLNÍ KONSTRUKCI FLOW*

# 4 OVĚŘENÍ PARAMETRŮ

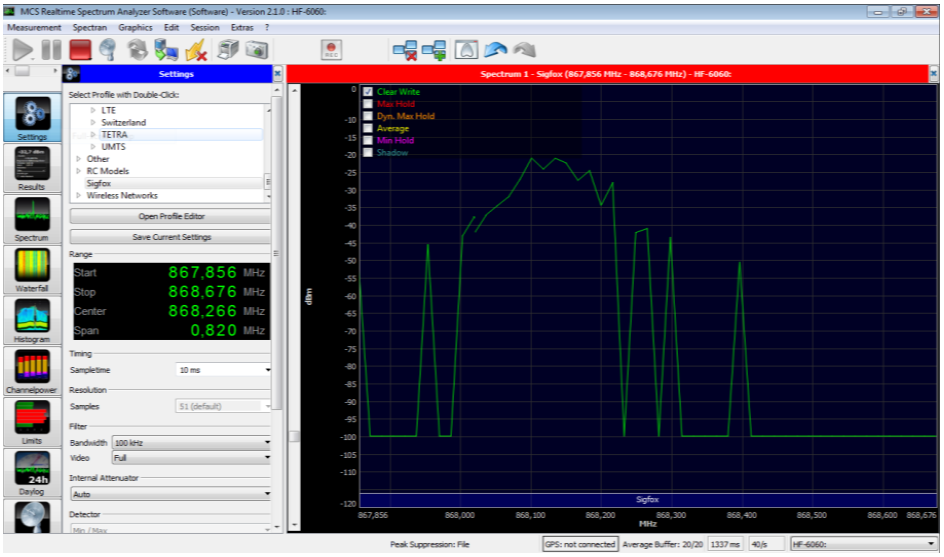
Sigfox poskytuje na svém backendu základní přehled o kvalitě signálu v podobě parametrů SNR a RSSI. RSSI parametr je zkratka pro Received Signal Strength Indicator a reprezentuje celou přijatou energii na přijímači včetně požadované energie z buňky stejně jako veškerý kanálový výkon a další zdroje šumu.

| | Venkovní prostředí | V místnosti s okny | Garáž | Chodba budovy | Automobil |
|---|---|---|---|---|---|
| SNR [dB] | 12.91 | 12.07 | / | 11.51 | 10.77 |
| RSSI [dBm] | -112.82 | -115.48 | / | -119.86 | -115.34 |
| Chybovost [%] | 0 | 0 | 88 | 4 | 2 |

*TAB. 8: NAMĚŘENÉ PARAMETRY Z BACKENDU SIGFOX*

V případě např. WiFi by hodnoty RSSI byly naprosto nepoužitelné, jelikož se počítá s hodnoty RSSI ideálně kolem -50 dBm. Ovšem na takové sítě jsou kladeny naprosto jiné požadavky. V případě LPWAN sítí se za dobrý signál může považovat signál s hodnotou RSSI -110 dBm. SNR parametr je naopak relativně v normě v porovnání s WiFi. Parametr chybovosti vyjadřuje procentuální ztrátu očekávaných zpráv. Všechny výše uvedené parametry jsou zprůměrovány 25 vzorky.

Technologie Sigfox je navržena pro využití ve sdíleném pásmu ISM. Byly ověřeny útlumové parametry kompletně zapojené desky při provozu. Použit byl přenosný analyzátor RF Spectrum Analyzer SPECTRAN HF-6060 V4 připojený k počítači. Ukázka výstupu analyzátoru SPECTRAN je uvedena níže.




*OBR. 45: VÝSTUP ZE SPEKTRÁLNÍHO ANALYZÁTORU*

# ZÁVĚR

Cílem této práce bylo v teoretické části seznámení se problematikou technologií IoT, popis používaných řešení a nasazených technologií. Po počátečním specifikování pojmů v první kapitole je v práci řešena historie internetu věcí a následně architektura přístrojů a vymezení základních typů datové komunikace. V dalším kroku jsou představeny oblasti, kde lze internet věcí aplikovat nebo se již aplikuje. Internet věcí je velmi obsáhlé téma, které zasahuje téměř do všech oblastí lidského života. Z důvodu širokého rozsahu tématu nebylo možné uvést veškerá řešení. Nezbytné je popsání komunikačních standardů, neboť právě ty jsou základem komunikace v internetu věcí. Následně je popsána velmi často opomíjená bezpečnost v IoT.

Internet věcí je pokládán za jednu z klíčových součástí čtvrté průmyslové revoluce neboli průmyslu 4.0. Tímto tématem se zabývají vědci, akademici, ale i technologické firmy po celém světě. Hlavním principem průmyslu 4.0 je pokročilá automatizace v rámci výrobních závodů a průmyslových podniků. Díky internetu věcí, tedy spojení co nejvíce zařízení mezi sebou a zároveň k internetu, je možné řešit problémy novými způsoby, automatizovat a „udělat chytřejšími“ věci a úkony, které by v minulosti byly nepředstavitelné.

Z použitých zdrojů je patrná všeobecně nedostatečná definice pojmů napříč spektrem internetu věcí. Kupříkladu neexistence jediné, obecně užívané definice pojmu internet věcí. Téměř každá mezinárodní firma si vytváří vlastní standardy, vlastní protokoly, vlastní představy a vlastní rozhraní. To vede k nejednoznačnosti, matení a brzdění ze všech stran hlásaného předpokládaného masivního využití internetu věcí. I z těchto důvodů není na trhu dostatek adekvátních řešení.

Druhá kapitola je zaměřena na detailní popis LPWAN sítí, které se vyznačují nízkou spotřebou a schopností přenášet informace na vzdálenosti až několik desítek kilometrů. Hlavními zástupci takových technologií jsou Sigfox, Lora a celulární sítě.

Z praktického hlediska byly využity dvě platformy. Konkrétně Raspberry Pi 3 Model B+ s rozšiřující deskou Sense Hat a Pytrack s modulem FiPy. Řešení s Raspberry bylo zvoleno z důvodu komplexnosti zařízení a jeho důkladné podpory. Aplikace vytvořené na této platformě byly využity jako odrazový můstek ke složitější práci s platformou od firmy Pycom.

Při práci s Pytrackem byly zjištěny problémy s nedostatečnou a nejednoznačnou dokumentací. Jelikož je firma Pycom relativně nová a jejich výrobky nejsou příliš známé a komerčně využívané, není k dispozici dostatečně kvalitní podpora. Přes počáteční komplikace se podařilo splnit zadání v plném rozsahu. Především jde o pilotní aplikaci a to sice vytvoření programu pro lokalizaci polohy pomocí globálního družicového polohovacího systému GLONASS, následného přenesení informace na sigfoxový backend, prvotní zpracování informace a zaslání na cloudovou službu Wia, kde je naprogramováno další zpracování dat. To vše přes LPWAN síť Sigfox. Výstupy aplikace jsou tři. Prvním je zobrazení lokace na mapě na webovém rozhraní, druhým je zaslání souřadnic na mobilní telefon a třetí je publikování polohy na sociální sítě, v tomto případě na Twitter. Stejně jako v případě Raspberry, i zde byly vytvořeny doplňující aplikace. Nepodařilo se zprovoznit režim hlubokého spánku tak, jak je inzerován. Pytrack se sice dostane do režimu hlubokého spánku na určenou dobu, poté se zapne, provede úkon a opět jde do režimu hlubokého spánku, nicméně to vše probíhá při plné spotřebě energie, kterou se nepovedlo vyřešit. Zástupci firmy Pycom poskytli pouze vyhýbavé odpovědi a nadále nekomunikovali. Jiné zdroje popisují špatnou podporu hlubokého spánku na konkrétních verzích firmwaru či modulu FiPy. Tyto neoficiální informace nejsou příliš relevantní, nicméně po důkladném průzkumu nebyl nalezen jediný případ, který by značil funkční režim hlubokého spánku v konfiguraci Pytracku s modulem FiPy.

Poslední bod zadání se věnuje ověření parametrů. Ověřována byla především výkonová úroveň signálu mezi zařízením a přijímačem v různém prostředí a útlumová charakteristika. Z měření vyplývá, že technologie Sigfox není zcela vhodná do vnitřního prostředí budov. Zatímco v otevřeném prostoru či za jednou zdí bylo dosaženo velice přívětivých výsledků, v automobilu nebo v chodbě budovy je již úroveň signálu velmi nízká a začínají se objevovat výpadky přenosu zpráv. Při pokusu o měření v garáži rodinného domu byla ztrátovost zpráv 88%.

## SEZNAM LITERATURY

[1] Internet of Things. *Gartner, Inc.* [online]. [cit. 2018-05-30]. Dostupné z: http://www.gartner.com/it-glossary/internet-of-things

[2] Top Trends in the Gartner Hype Cycle for Emerging Technologies, 2017. *Gartner, Inc.* [online]. [cit. 2018-05-30]. Dostupné z: https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017

[3] The Internet of Things In a Connected World of Smart Objects. *Fundacion Innovacion Bankinter* [online]. [cit. 2018-05-30]. Dostupné z: https://www.fundacionbankinter.org/documents/20183/137558/Publicacion+PDF+IN+FTF_IOT.pdf/2783707e-b729-45b2-98eb-1ba52b652b37

[4] POHANKA, Pavel. *Internet věcí* [online]. [cit. 2018-05-30]. Dostupné z: http://i2ot.eu/internet-of-things/

[5] NOVAK, Mat. Nikola Tesla's Incredible Predictions For Our Connected World [online]. [cit. 2. 8. 2017]. Dostupné z: http://paleofuture.gizmodo.com/nikola-teslas-incredible-predictions-for-our-connected-1661107313

[6] World Wide Web. History of the Web [online]. *World Wide Web Foundation* [online]. [cit. 3. 8. 2017]. Dostupné z: https://webfoundation.org/about/vision/history-of-the-web/

[7] DEORAS, Sritshi. First ever IoT device- „The internet Toaster“ [online]. [cit. 2. 8. 2017]. Dostupné z: http://iotindiamag.com/2016/08/first-ever-iot-device-the-internet-toaster/

[8] EVANS, Dave. The Internet of Things how the next evolution of the Internet is changing everything. *Cisco Systems, Inc*. [online]. [cit. 2018-05-30]. Dostupné z: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[9] Gartner Says 6.4 Billion Connected "Things"Will Be in Use in 2016, Up 30 Percent From 2015. *Gartner, Inc.* [online]. [cit. 2018-05-30]. Dostupné z:http://www.gartner.com/newsroom/id/3165317

[10] The Internet of Things Will Drive Wireless Connected Devices to 40.9 Billion in 2020. *ABI Research.* [online]. [cit. 02. 11. 2015]. Dostupné z: https://www.abiresearch.com/press/the-internet-of-things-will-drive-wireless-connect/

[11] Internet of Things (IoT). *Cisco Systems, Inc*. [online]. [cit. 28. 12. 2015]. Dostupné z: http://www.cisco.com/web/solutions/trends/iot/overview.html

[12] Internet of Things – Architecture. *IoT Forum* [online]. [cit. 2018-05-30]. Dostupné z: https://iotforum.org/wp-content/uploads/2014/09/D1.5-20130715-VERYFINAL.pdf

[13] Special Issue of the International Journal on Network Management (IJNM) on Management of the Internet of Things and Big Data. *University of Nicosia* [online]. [cit.

2018-05-30]. Dostupné z:
http://www.cs.unic.ac.cy/cmavrom/IJNM_SI_Internet_of_Things_and_Big_Data.pdf

[14] ARKKO J., THALER D. and MCPHERSON D. Internet architecture board (IAB). 2011. [cit. 2018-05-30]. ISSN: 2070-1721.

[15] The internet of things: an overview. *ISOC* [online]. [cit. 2018-05-30]. Dostupné z: https://cdn.prod.internetsociety.org/wp-content/uploads/2017/08/ISOC-IoT-Overview-20151221-en.pdf

[16] 5 key benefits of IoT for healthcare organizations. *[x]cube LABS* [online]. [cit. 2018-05-30]. Dostupné z: http://www.xcubelabs.com/our-blog/benefits-iot-in-healthcare/

[17] Myslíme na budoucnost. *Volvo* [online].[cit. 2018-05-30]. Dostupné z: http://www.volvocars.com/cz/o-nas/nase-pribehy/inovace-pro-lidi/myslime-nabudoucnost

[18] TRCÁLEK, Antonín. *Google není jediný, kdo vyvíjí chytrá auta budoucnosti* [online].[cit. 2018-05-30]. Dostupné z: http://www.zive.cz/clanky/google-nenijediny-kdo-vyviji-chytra-auta-budoucnosti/sc-3-a-170245/default.aspx

[19] ČEZ testuje drony pro kontrolu energetických zařízení [online]. [cit. 2018-05-30]. Dostupné z: http://www.solarninovinky.cz/?zpravy/2017081402/cez-testujedrony-pro-kontrolu-energetickych-zarizeni

[20] A novel and efficient user access control scheme for wireless body area sensor networks. *ResearchGate* [online]. [cit. 2018-05-30]. Dostupné z: https://www.researchgate.net/figure/A-general-three-tier-architecture-of-WBAN-Li-et-al-2010_fig10_259157645

[21] 50 Sensor Applications for a Smarter World. *Libelium*. [Online] [Citace: 2018-05-30]. Dostupné z:
http://www.libelium.com/resources/top_50_iot_sensor_applications_ranking/

[22] Představte si budoucnost v propojeném světě internetu
Věcí. *IDC IOT Forum 2015*. [online].[cit. 2018-05-30]. Dostupné z: http://idc-czech.cz/cze/oidc/tiskove-zpravy/63043-idc-iot-forum-2015-predstavte-si-budoucnost-v-propojenemsvete-internetu-veci

[23] Google Cloud Platform Documentation. *Google*. [Online] [Citace: 2018-05-30]. Dostupné z: https://cloud.google.com/docs/

[24] IoT Networking. *Cisco Systems, Inc*. [Online] [Citace: 2018-05-30]. Dostupné z: https://www.cisco.com/c/en/us/solutions/internet-of-things/iot-network-connectivity.html

[25] Getting started. *PlatformIO*. [Online] [Citace: 2018-05-30]. Dostupné z: https://platformio.org/get-started

[26] Getting started. *Node - RED*. [Online] [Citace: 2018-05-30]. Dostupné z: https://nodered.org/docs/getting-started/

[27] What is Azure IoT Hub?. *Microsoft*. [Online] [Citace: 2018-05-30]. Dostupné z: https://docs.microsoft.com/en-us/azure/iot-hub/about-iot-hub

[28] Hypertext Transfer Protocol. *World Wide Web Consortium*. [Online] [Citace: 2018-05-30]. Dostupné z: https://www.w3.org/Protocols/rfc2616/rfc2616.html

[29] 5 Things to Know About MQTT – The Protocol for Internet of Things. *IBM*. [Online] [Citace: 2018-05-30]. Dostupné z: https://www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_mqtt_the_protocol_for_internet_of_things?lang=es

[30] IoT MQTT prakticky v automatizaci. *Automatizace.HW*. [Online] [Citace: 2018-05-30]. Dostupné z: https://automatizace.hw.cz/iot-mqtt-prakticky-v-automatizaci-1dil-uvod.html

[31] What is CoAP IoT protocol | CoAP Architecture,message header. RF Wireless World. [Online] [Citace: 2018-05-30]. Dostupné z: https://automatizace.hw.cz/iot-mqtt-prakticky-v-automatizaci-1dil-uvod.html

[32] ČERMÁK, Petr a Tomáš Kramný. TECHNOLOGIE IOT [online]. [cit. 2018-05-30]. Dostupné z: http://mvso.cz/wp-content/uploads/2018/02/Technologie-IoT-studijn%C3%AD-text.pdf

[33] Understanding IoT Security – Part 1 of 3: IoT Security Architecture on the Device and Communication Layers. *IOT ANALYTICS*. [Online] [Citace: 2018-05-30]. Dostupné z: https://iot-analytics.com/UNDERSTANDING-IOT-SECURITY-PART-1-IOT-SECURITY-ARCHITECTURE

[34] Insecurity in IoT. Symantec. *Symantec* [online] [cit. 2018-05-30]. Dostupné z: https://www.symantec.com/content/dam/symantec/docs/white-papers/insecurity-in-the-internet-of-things-en.pdf

[35] VOJÁČEK, Antonín**.** Základní úvod do oblasti internetu věcí (IoT). *automatizace hw* [online]. [cit. 2018-05-30] http://automatizace.hw.cz/zakladni-uvod-do-oblasti-internetu-veci-iot.html.

[36] EGLI, P. R. LPWAN Technologies for Internet of Things (IoT) and M2M Scenarios [online]. [cit. 2018-05-30]. Dostupné z: http://fr.slideshare.net/PeterREgli/lpwan

[37] ETSI. Low Throughput Networks. *Etsi* [online]. [cit. 2018-05-30]. Dostupné z: http://www.etsi.org/technologies-clusters/technologies/lowthroughput-networks

[38] EGLI, P. R.. LPWAN Technologies for Internet of Things (IoT) and M2M Scenarios. *SlideShare* [online]. [cit. 2018-05-30]. Dostupné z: https://www.slideshare.net/PeterREgli/lpwan

[39] 3GPP: GERAN#67, 2015 [online]. [cit. 2018-05-30]. Dostupné z: http://www.3gpp.org/ftp/tsg_geran/TSG_GERAN/GERAN_67_Yinchua

[40] 3GPP: GERAN#69, 2015 [online]. [cit. 2018-05-30]. Dostupné z: http://www.3gpp.org/ftp/tsg_geran/TSG_GERAN/GERAN_69_Malta

[41] 3GPP: GERAN#70, 2016 [online]. [cit. 2018-05-30]. Dostupné z: http://www.3gpp.org/ftp/tsg_geran/TSG_GERAN/GERAN_70_Nanjing

[42] 3GPP: GERAN#69: Narrowband IOT, 2015 [online]. [cit. 2018-05-30]. Dostupné z: ftp://ftp.3gpp.org/tsg_ran/TSG_RAN/TSGR_69/Docs/RP-151621.zip

[43] 3GPP: List of Change Requests that allow NB-IoT to be implemented in Release 13, 2016 [online]. [cit. 2018-05-30]. Dostupné z: http://www.3gpp.org/images/PDF/R13_IOT_rev3.pdf

[44] 3GPP GERAN#63: Cellular System Support for Ultra Low Complexity and Low Throughput Internet of Things [online]. [cit. 2018-05-30]. Dostupné z: http://www.3gpp.org/ftp/tsg_geran/TSG_GERAN/GERAN_62_Valencia /Docs/GP-140421.zip

[45] XINHUI, Wang. 3GPP TSG GERAN. GIS MOTS Convention Center Taipei, Taiwan [online]. [cit. 2018-05-30]. Dostupné z: http://www.3gpp.org/ftp/workshop/2015-11-15_Taipei_summit_5G/Docs/2_5_2015_ITRI_3GPP_summit_GERANv05.pdf

[46] 3GPP TR 23.770 v13.0.0 Study on system impacts of extended Discontiuous Rception (DRX) cycle for power consumption optimization (Release 13) [online]. [cit. 2018-05-30]. Dostupné z: http://www.3gpp.org/ftp/Specs/archive/23_series/23.770/23770-d00.zip

[47] CHANG, Chia-Wei a Jyh-Cheng CHEN. Adjustable Extended Discontinuous Reception Cycle for Idle-State Users in LTE-A. IEEE Communications Letters [online]. [cit. 2018-05-30]. DOI: 10.1109/LCOMM.2016.2602200. ISSN 1089-7798. Dostupné z: http://ieeexplore.ieee.org/document/7549050/

[48] DA SILVA, Icaro Leonardo, Gunnar MILDH, Mikko SAILY a Sofonias HAILU. A novel state model for 5G Radio Access Networks. In: 2016 IEEE International Conference on Communications Workshops (ICC) [online]. IEEE, 2016, s. 632-637 [online]. [cit. 2018-05-30]. DOI: 10.1109/ICCW.2016.7503858. ISBN 978-1-5090-0448-5. Dostupné z: http://ieeexplore.ieee.org/document/7503858/

[49] 3GPP. LPWA June 2016 [online].[cit. 2018-05-30]. Dostupné z: http://www.3gpp.org/ftp/Information/presentations/presentations_2016/3GPP_Standards_for_IoT.pdf

[50] 3GPP. RP-151621: Narrowband IOT. *3GPP* [online]. [cit. 2018-05-30]. Dostupné z: ftp://ftp.3gpp.org/tsg_ran/TSG_RAN/TSGR_69/Docs/RP-151621.zip

[51] SCHLIENZ, J., RADDINO, D. Narrowband Internet of Things (Rohde&Schwarz Whitepaper) [online]. [cit. 2018-05-30]. Dostupné z: https://cdn.rohdeschwarz.com/pws/dl_downloads/dl_application/application_notes/1ma 266/1MA266_0e_NB_IoT.pdf

[52] WANG Y.-P., et al. A Primer on 3GPP Narrowband Internet of Things (NBIoT) [online]. [cit. 2018-05-30]. Dostupné z: https://arxiv.org/ftp/arxiv/papers/1606/1606.04171.pdf

[53] Elnashar, A. Building IoT Network for Smart City. Forum on Internet of Things: Empowering the New Urban Agenda. [online]. [cit. 2018-05-30]. Dostupné z: https://www.itu.int/en/ITU-T/Workshops-andSeminars/iot/20151019/Documents/S1P2_Ayman_ElNashar.pptx

[54] Nokia. LTE evolution for IoT connectivity. *Nokia* [online]. [cit. 2018-05-30]. Dostupné z: http://resources.alcatel-lucent.com/asset/200178

[55] LoRa Alliance: LoRaWAN Specification. *LoRa Alliance* [online]. [cit. 2018-05-30]. Dostupné z: https://lora-alliance.org/DesktopModules/Inventures_Document/FileDownload.asp%20x?ContentID=1398

[56] VOIGT, T., bor M., Roedig, U., Alonso a J. MITIGATING: Inter-network Interference in LoRa Networks [online]. arXiv:1611.00688v1 [cs.NI]. [cit. 2018-05-30]. Dostupné z: https://arxiv.org/pdf/1611.00688v1.pdf

[57] GEORGIOU, O., RAZA, U. Low power wide area network analysis: Can LoRa scale? [online]. 2016 [cit. 2018-05-30]. Dostupné z: arXiv preprint arXiv:1610.04793

[58] HAUSNER, V. Bezpečnost v sítích LPWAN/LPN pro aplikace v IoT [online]. [cit. 2018-05-30]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/64726/F3-BP-2016-Hauser-VojtechBezpecnost%20v%20sitich%20LPWAN%3fLPN%20pro%20aplikace%20v%20IoT.pdf?sequence=-1&isAllowed=y

[59] Designer's Guide AN1200.13 SX1272/3/6/8: LoRa Modem. *Semtech Corporation* [online]. [cit. 2018-05-30]. Dostupné z: https://www.semtech.com/images/datasheet/LoraDesignGuide_STD.pdf

[60] LoRaWan 1.0.1 Specification. *LoRa Alliance.* [Online]. [cit. 2018-05-30]. Dostupné z: https://enablingsupport.zendesk.com/hc/enus/articles/205928502-LoRaWan-1-0-1-specification

[61] SIKORA, Axel, AREF, Mohamed. Free Space Range Measurements with Semtech LoRaTM Technology [online]. [cit. 2018-05-30]. Dostupné z: http://ieeexplore.ieee.org/document/6954616/

[62] G. Margelis, R. Piechocki, D.K. a Thoma, P. *Low throughput networks for the iot: Lessons learned from industrial implementations*, s. 181-186.

[63] Davies, A. On lpwans: Why sigfox and lora are rather different, and the importance of the business model [online]. [cit. 2018-05-30]. Dostupné z: http://rethinkiot.com/2015/03/20/on-lpwans-why-sigfox-and-lora-are-ratherdifferent-and-the-importance-of-the-business-model/

[64] Link Labs, I.. A comprehensive look at low power, wide area networks [online]. [cit. 2018-05-30]. Dostupné z: http: http://info.link-labs.com/lpwan

[65] Emmerson, B. License-Free Spectrum Goes Cellular. [online]. [cit. 2018-05-30]. Dostupné z: http://www.nojitter.com/post/240168055/licensefreespectrum-goes-cellular

[66] Minh-Tien Do, Claire Goursaud, J.M.G.: Interference modelling and analysis of random fdma scheme in ultra narrowband networks [online]. [cit. 2018-05-30]. Dostupné z: https://hal.archives-ouvertes.fr/hal-01096493

[67] SIGFOX: Sigfox specics [online]. [cit. 2018-05-30]. Dostupné z: http://makers.SIGFOX.com/#about

[68] Všeobecné oprávnění vo-r/10/05.2014-3 k využívání rádiových kmitočtů u a k provozovaní zařízení krátkého dosahu. *Český telekomunikační úřad* [online]. [cit. 2018-05-30]. Dostupné z: http://www.ctu.cz/cs/download/oop/rok_2014/vo-r_10-05_2014-03.pdf#page=2

[69] Galati, G. 100 Years of Radar. Springer eBook. 2016. ISBN 9783319005843.

[70] Brown, R. H. Robert Alexander Watson-Watt, the father of radar. Engineering Science and Education Journal. 1994, Volume: 3, Issue: 1. Pages: 31 - 40. DOI: 10.1049/esej:19940108

[71] SimpleCell Networks a.s.: *Technologie SIGFOX* [online]. [cit. 2018-05-30]. Dostupné z: https://www.simplecell.eu/technologie_sigfox

[72] Emmerson, B. License-*Free Spectrum Goes Cellular* [online]. [cit. 2018-05-30]. Dostupné z: http://www.nojitter.com/post/240168055/licensefreespectrum-goes-cellular

[73]Datasheet: ATA8520 [online].[cit. 2017-05-22]. Dostupné z: http://www.atmel.com/Images/Atmel-9372-Smart-RF-ATA8520_Datasheet.pdf

[74] Technologie SIGFOX. *SimpleCell Networks a.s.:* [online]. [cit. 2018-05-30]. Dostupné z: https://www.simplecell.eu/technologie_sigfox

[75] Low Throughput Networks. *ETSI* [online]. [cit. 2018-05-30]. Dostupné z: http://www.etsi.org/technologies-clusters/technologies/lowthroughput-networks
[76] RAZA, U., KULKARNI, P., SOORIYABANDARA, M. Low Power Wide Area Networks: An overview. *IEEE* [online]. [cit. 2018-05-30] Dostupné z: https://arxiv.org/pdf/1606.07360.pdf

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# PŘÍLOHY

### Příloha A: Knihovna pytrack.py

```
from pycoproc import Pycoproc

__version__ = '1.4.0'

class Pytrack(Pycoproc):

    def __init__(self, i2c=None, sda='P22', scl='P21'):
        Pycoproc.__init__(self, i2c, sda, scl)
```

### Příloha B: knihovna pycoproc.py

```
from machine import Pin
from machine import I2C
import time
import pycom

__version__ = '0.0.2'

""" PIC MCU wakeup reason types """
WAKE_REASON_ACCELEROMETER = 1
WAKE_REASON_PUSH_BUTTON = 2
WAKE_REASON_TIMER = 4
WAKE_REASON_INT_PIN = 8

class Pycoproc:
    """ class for handling the interaction with PIC MCU """

    I2C_SLAVE_ADDR = const(8)

    CMD_PEEK = const(0x0)
    CMD_POKE = const(0x01)
    CMD_MAGIC = const(0x02)
    CMD_HW_VER = const(0x10)
    CMD_FW_VER = const(0x11)
    CMD_PROD_ID = const(0x12)
    CMD_SETUP_SLEEP = const(0x20)
    CMD_GO_SLEEP = const(0x21)
    CMD_CALIBRATE = const(0x22)
    CMD_BAUD_CHANGE = const(0x30)
    CMD_DFU = const(0x31)

    REG_CMD = const(0)
    REG_ADDRL = const(1)
    REG_ADDRH = const(2)
    REG_AND = const(3)
    REG_OR = const(4)
    REG_XOR = const(5)

    ANSELA_ADDR = const(0x18C)
    ANSELB_ADDR = const(0x18D)
    ANSELC_ADDR = const(0x18E)
```

```
ADCON0_ADDR = const(0x9D)
ADCON1_ADDR = const(0x9E)

IOCAP_ADDR = const(0x391)
IOCAN_ADDR = const(0x392)

INTCON_ADDR = const(0x0B)
OPTION_REG_ADDR = const(0x95)

_ADCON0_CHS_POSN = const(0x02)
_ADCON0_ADON_MASK = const(0x01)
_ADCON1_ADCS_POSN = const(0x04)
_ADCON0_GO_nDONE_MASK = const(0x02)

ADRESL_ADDR = const(0x09B)
ADRESH_ADDR = const(0x09C)

TRISC_ADDR = const(0x08E)

PORTA_ADDR = const(0x00C)
PORTC_ADDR = const(0x00E)

WPUA_ADDR = const(0x20C)

WAKE_REASON_ADDR = const(0x064C)
MEMORY_BANK_ADDR = const(0x0620)

PCON_ADDR = const(0x096)
STATUS_ADDR = const(0x083)

EXP_RTC_PERIOD = const(7000)

def __init__(self, i2c=None, sda='P22', scl='P21'):
    if i2c is not None:
        self.i2c = i2c
    else:
        self.i2c = I2C(0, mode=I2C.MASTER, pins=(sda, scl))

    self.sda = sda
    self.scl = scl
    self.clk_cal_factor = 1
    self.reg = bytearray(6)
    self.wake_int = False
    self.wake_int_pin = False
    self.wake_int_pin_rising_edge = True

    # Make sure we are inserted into the
    # correct board and can talk to the PIC
    try:
        self.read_fw_version()
    except Exception as e:
        raise Exception('Board not detected: {}'.format(e))

    # init the ADC for the battery measurements
    self.poke_memory(ANSELC_ADDR, 1 << 2)
```

```
        self.poke_memory(ADCON0_ADDR, (0x06 << _ADCON0_CHS_POSN) |
_ADCON0_ADON_MASK)
        self.poke_memory(ADCON1_ADDR, (0x06 << _ADCON1_ADCS_POSN))
        # enable the pull-up on RA3
        self.poke_memory(WPUA_ADDR, (1 << 3))
        # make RC5 an input
        self.set_bits_in_memory(TRISC_ADDR, 1 << 5)
        # set RC6 and RC7 as outputs and enable power to the
sensors and the GPS
        self.mask_bits_in_memory(TRISC_ADDR, ~(1 << 6))
        self.mask_bits_in_memory(TRISC_ADDR, ~(1 << 7))

        if self.read_fw_version() < 6:
            raise ValueError('Firmware out of date')


    def _write(self, data, wait=True):
        self.i2c.writeto(I2C_SLAVE_ADDR, data)
        if wait:
            self._wait()

    def _read(self, size):
        return self.i2c.readfrom(I2C_SLAVE_ADDR, size + 1)[1:(size
+ 1)]

    def _wait(self):
        count = 0
        time.sleep_us(10)
        while self.i2c.readfrom(I2C_SLAVE_ADDR, 1)[0] != 0xFF:
            time.sleep_us(100)
            count += 1
            if (count > 500):  # timeout after 50ms
                raise Exception('Board timeout')

    def _send_cmd(self, cmd):
        self._write(bytes([cmd]))

    def read_hw_version(self):
        self._send_cmd(CMD_HW_VER)
        d = self._read(2)
        return (d[1] << 8) + d[0]

    def read_fw_version(self):
        self._send_cmd(CMD_FW_VER)
        d = self._read(2)
        return (d[1] << 8) + d[0]

    def read_product_id(self):
        self._send_cmd(CMD_PROD_ID)
        d = self._read(2)
        return (d[1] << 8) + d[0]

    def peek_memory(self, addr):
        self._write(bytes([CMD_PEEK, addr & 0xFF, (addr >> 8) &
0xFF]))
        return self._read(1)[0]
```

```
    def poke_memory(self, addr, value):
        self._write(bytes([CMD_POKE, addr & 0xFF, (addr >> 8) &
0xFF, value & 0xFF]))

    def magic_write_read(self, addr, _and=0xFF, _or=0, _xor=0):
        self._write(bytes([CMD_MAGIC, addr & 0xFF, (addr >> 8) &
0xFF, _and & 0xFF, _or & 0xFF, _xor & 0xFF]))
        return self._read(1)[0]

    def toggle_bits_in_memory(self, addr, bits):
        self.magic_write_read(addr, _xor=bits)

    def mask_bits_in_memory(self, addr, mask):
        self.magic_write_read(addr, _and=mask)

    def set_bits_in_memory(self, addr, bits):
        self.magic_write_read(addr, _or=bits)

    def get_wake_reason(self):
        """ returns the wakeup reason, a value out of constants
WAKE_REASON_* """
        return self.peek_memory(WAKE_REASON_ADDR)

    def get_sleep_remaining(self):
        """ returns the remaining time from sleep, as an interrupt
(wakeup source) might have triggered """
        c3 = self.peek_memory(WAKE_REASON_ADDR + 3)
        c2 = self.peek_memory(WAKE_REASON_ADDR + 2)
        c1 = self.peek_memory(WAKE_REASON_ADDR + 1)
        time_device_s = (c3 << 16) + (c2 << 8) + c1
        # this time is from PIC internal oscilator, so it needs to
be adjusted with the calibration value
        try:
            self.calibrate_rtc()
        except Exception:
            pass
        time_s = int((time_device_s / self.clk_cal_factor) + 0.5)
# 0.5 used for round
        return time_s

    def setup_sleep(self, time_s):
        try:
            self.calibrate_rtc()
        except Exception:
            pass
        time_s = int((time_s * self.clk_cal_factor) + 0.5)  #
round to the nearest integer
        if time_s >= 2**(8*3):
            time_s = 2**(8*3)-1
        self._write(bytes([CMD_SETUP_SLEEP, time_s & 0xFF, (time_s
>> 8) & 0xFF, (time_s >> 16) & 0xFF]))

    def go_to_sleep(self, gps=True):
        # enable or disable back-up power to the GPS receiver
        if gps:
```

```
            self.set_bits_in_memory(PORTC_ADDR, 1 << 7)
        else:
            self.mask_bits_in_memory(PORTC_ADDR, ~(1 << 7))
        # disable the ADC
        self.poke_memory(ADCON0_ADDR, 0)

        if self.wake_int:
            # Don't touch RA3, RA5 or RC1 so that interrupt wake-
up works
            self.poke_memory(ANSELA_ADDR, ~((1 << 3) | (1 << 5)))
            self.poke_memory(ANSELC_ADDR, ~((1 << 6) | (1 << 7) |
(1 << 1)))
        else:
            # disable power to the accelerometer, and don't touch
RA3 so that button wake-up works
            self.poke_memory(ANSELA_ADDR, ~(1 << 3))
            self.poke_memory(ANSELC_ADDR, ~(1 << 7))

        self.poke_memory(ANSELB_ADDR, 0xFF)

        # check if INT pin (PIC RC1), should be used for wakeup
        if self.wake_int_pin:
            if self.wake_int_pin_rising_edge:
                self.set_bits_in_memory(OPTION_REG_ADDR, 1 << 6) #
rising edge of INT pin
            else:
                self.mask_bits_in_memory(OPTION_REG_ADDR, ~(1 <<
6)) # falling edge of INT pin
            self.mask_bits_in_memory(ANSELC_ADDR, ~(1 << 1)) #
disable analog function for RC1 pin
            self.set_bits_in_memory(TRISC_ADDR, 1 << 1) # make RC1
input pin
            self.mask_bits_in_memory(INTCON_ADDR, ~(1 << 1)) #
clear INTF
            self.set_bits_in_memory(INTCON_ADDR, 1 << 4) # enable
interrupt; set INTE)

        self._write(bytes([CMD_GO_SLEEP]), wait=False)
        # kill the run pin
        Pin('P3', mode=Pin.OUT, value=0)

    def calibrate_rtc(self):
        # the 1.024 factor is because the PIC LF operates at 31
KHz
        # WDT has a frequency divider to generate 1 ms
        # and then there is a binary prescaler, e.g., 1, 2, 4 ...
512, 1024 ms
        # hence the need for the constant
        self._write(bytes([CMD_CALIBRATE]), wait=False)
        self.i2c.deinit()
        Pin('P21', mode=Pin.IN)
        pulses = pycom.pulses_get('P21', 100)
        self.i2c.init(mode=I2C.MASTER, pins=(self.sda, self.scl))
        idx = 0
        for i in range(len(pulses)):
            if pulses[i][1] > EXP_RTC_PERIOD:
```

```
                idx = i
                break
        try:
            period = pulses[idx][1] - pulses[(idx - 1)][1]
        except:
            period = 0
        if period > 0:
            self.clk_cal_factor = (EXP_RTC_PERIOD / period) *
(1000 / 1024)
        if self.clk_cal_factor > 1.25 or self.clk_cal_factor <
0.75:
            self.clk_cal_factor = 1

    def button_pressed(self):
        button = self.peek_memory(PORTA_ADDR) & (1 << 3)
        return not button

    def read_battery_voltage(self):
        self.set_bits_in_memory(ADCON0_ADDR,
_ADCON0_GO_nDONE_MASK)
        time.sleep_us(50)
        while self.peek_memory(ADCON0_ADDR) &
_ADCON0_GO_nDONE_MASK:
            time.sleep_us(100)
        adc_val = (self.peek_memory(ADRESH_ADDR) << 2) +
(self.peek_memory(ADRESL_ADDR) >> 6)
        return (((adc_val * 3.3 * 280) / 1023) / 180) + 0.01    #
add 10mV to compensate for the drop in the FET

    def setup_int_wake_up(self, rising, falling):
        """ rising is for activity detection, falling for
inactivity """
        wake_int = False
        if rising:
            self.set_bits_in_memory(IOCAP_ADDR, 1 << 5)
            wake_int = True
        else:
            self.mask_bits_in_memory(IOCAP_ADDR, ~(1 << 5))

        if falling:
            self.set_bits_in_memory(IOCAN_ADDR, 1 << 5)
            wake_int = True
        else:
            self.mask_bits_in_memory(IOCAN_ADDR, ~(1 << 5))
        self.wake_int = wake_int

    def setup_int_pin_wake_up(self, rising_edge = True):
        """ allows wakeup to be made by the INT pin (PIC -RC1) """
        self.wake_int_pin = True
        self.wake_int_pin_rising_edge = rising_edge
```

**Příloha C: knihovna L76GNSS.py**

```
from machine import Timer
import time
import gc
import binascii


class L76GNSS:

    GPS_I2CADDR = const(0x10)

    def __init__(self, pytrack=None, sda='P22', scl='P21',
timeout=None):
        if pytrack is not None:
            self.i2c = pytrack.i2c
        else:
            from machine import I2C
            self.i2c = I2C(0, mode=I2C.MASTER, pins=(sda, scl))

        self.chrono = Timer.Chrono()

        self.timeout = timeout
        self.timeout_status = True

        self.reg = bytearray(1)
        self.i2c.writeto(GPS_I2CADDR, self.reg)

    def _read(self):
        self.reg = self.i2c.readfrom(GPS_I2CADDR, 64)
        return self.reg

    def _convert_coords(self, gngll_s):
        lat = gngll_s[1]
        lat_d = (float(lat) // 100) + ((float(lat) % 100) / 60)
        lon = gngll_s[3]
        lon_d = (float(lon) // 100) + ((float(lon) % 100) / 60)
        if gngll_s[2] == 'S':
            lat_d *= -1
        if gngll_s[4] == 'W':
            lon_d *= -1
        return(lat_d, lon_d)

    def coordinates(self, debug=False):
        lat_d, lon_d, debug_timeout = None, None, False
        if self.timeout is not None:
            self.chrono.reset()
            self.chrono.start()
        nmea = b''
        while True:
            if self.timeout is not None and self.chrono.read() >=
self.timeout:
                self.chrono.stop()
                chrono_timeout = self.chrono.read()
                self.chrono.reset()
```

```
                self.timeout_status = False
                debug_timeout = True
            if not self.timeout_status:
                gc.collect()
                break
            nmea += self._read().lstrip(b'\n\n').rstrip(b'\n\n')
            gngll_idx = nmea.find(b'GNGLL')
            if gngll_idx >= 0:
                gngll = nmea[gngll_idx:]
                e_idx = gngll.find(b'\r\n')
                if e_idx >= 0:
                    try:
                        gngll = gngll[:e_idx].decode('ascii')
                        gngll_s = gngll.split(',')
                        lat_d, lon_d =
self._convert_coords(gngll_s)
                    except Exception:
                        pass
                    finally:
                        nmea = nmea[(gngll_idx + e_idx):]
                        gc.collect()
                        break
            else:
                gc.collect()
                if len(nmea) > 410: # i suppose it can be safely
changed to 82, which is longest NMEA frame
                    nmea = nmea[-5:] # $GNGL without last L
            time.sleep(0.1)
        self.timeout_status = True
        if debug and debug_timeout:
            print('GPS timed out after %f seconds' %
(chrono_timeout))
            return(None, None)
        else:
            return(lat_d, lon_d)
```

**Příloha D: knihovna LIS2HH12.py**

```
import math
import time
import struct
from machine import Pin


FULL_SCALE_2G = const(0)
FULL_SCALE_4G = const(2)
FULL_SCALE_8G = const(3)

ODR_POWER_DOWN = const(0)
ODR_10_HZ = const(1)
ODR_50_HZ = const(2)
ODR_100_HZ = const(3)
ODR_200_HZ = const(4)
ODR_400_HZ = const(5)
ODR_800_HZ = const(6)

ACC_G_DIV = 1000 * 65536


class LIS2HH12:

    ACC_I2CADDR = const(30)

    PRODUCTID_REG = const(0x0F)
    CTRL1_REG = const(0x20)
    CTRL2_REG = const(0x21)
    CTRL3_REG = const(0x22)
    CTRL4_REG = const(0x23)
    CTRL5_REG = const(0x24)
    ACC_X_L_REG = const(0x28)
    ACC_X_H_REG = const(0x29)
    ACC_Y_L_REG = const(0x2A)
    ACC_Y_H_REG = const(0x2B)
    ACC_Z_L_REG = const(0x2C)
    ACC_Z_H_REG = const(0x2D)
    ACT_THS = const(0x1E)
    ACT_DUR = const(0x1F)

    SCALES = {FULL_SCALE_2G: 4000, FULL_SCALE_4G: 8000,
FULL_SCALE_8G: 16000}
    ODRS = [0, 10, 50, 100, 200, 400, 800]

    def __init__(self, pysense = None, sda = 'P22', scl = 'P21'):
        if pysense is not None:
            self.i2c = pysense.i2c
        else:
            from machine import I2C
            self.i2c = I2C(0, mode=I2C.MASTER, pins=(sda, scl))

        self.odr = 0
        self.full_scale = 0
```

```
        self.x = 0
        self.y = 0
        self.z = 0
        self.int_pin = None
        self.act_dur = 0
        self.debounced = False

        whoami = self.i2c.readfrom_mem(ACC_I2CADDR ,
PRODUCTID_REG, 1)
        if (whoami[0] != 0x41):
            raise ValueError("LIS2HH12 not found")

        # enable acceleration readings at 50Hz
        self.set_odr(ODR_50_HZ)

        # change the full-scale to 4g
        self.set_full_scale(FULL_SCALE_4G)

        # set the interrupt pin as active low and open drain
        self.set_register(CTRL5_REG, 3, 0, 3)

        # make a first read
        self.acceleration()

    def acceleration(self):
        x = self.i2c.readfrom_mem(ACC_I2CADDR , ACC_X_L_REG, 2)
        self.x = struct.unpack('<h', x)
        y = self.i2c.readfrom_mem(ACC_I2CADDR , ACC_Y_L_REG, 2)
        self.y = struct.unpack('<h', y)
        z = self.i2c.readfrom_mem(ACC_I2CADDR , ACC_Z_L_REG, 2)
        self.z = struct.unpack('<h', z)
        _mult = self.SCALES[self.full_scale] / ACC_G_DIV
        return (self.x[0] * _mult, self.y[0] * _mult, self.z[0] *
_mult)

    def roll(self):
        x,y,z = self.acceleration()
        rad = math.atan2(-x, z)
        return (180 / math.pi) * rad

    def pitch(self):
        x,y,z = self.acceleration()
        rad = -math.atan2(y, (math.sqrt(x*x + z*z)))
        return (180 / math.pi) * rad

    def set_register(self, register, value, offset, mask):
        reg = bytearray(self.i2c.readfrom_mem(ACC_I2CADDR,
register, 1))
        reg[0] &= ~(mask << offset)
        reg[0] |= ((value & mask) << offset)
        self.i2c.writeto_mem(ACC_I2CADDR, register, reg)

    def set_full_scale(self, scale):
        self.set_register(CTRL4_REG, scale, 4, 3)
        self.full_scale = scale
```

```
    def set_odr(self, odr):
        self.set_register(CTRL1_REG, odr, 4, 7)
        self.odr = odr

    def set_high_pass(self, hp):
        self.set_register(CTRL2_REG, 1 if hp else 0, 2, 1)

    def enable_activity_interrupt(self, threshold, duration,
handler=None):
        # Threshold is in mg, duration is ms
        self.act_dur = duration

        if threshold > self.SCALES[self.full_scale]:
            error = "threshold %d exceeds full scale %d" %
(thresold, self.SCALES[self.full_scale])
            print(error)
            raise ValueError(error)

        if threshold < self.SCALES[self.full_scale] / 128:
            error = "threshold %d below resolution %d" %
(thresold, self.SCALES[self.full_scale]/128)
            print(error)
            raise ValueError(error)

        if duration > 255 * 1000 * 8 / self.ODRS[self.odr]:
            error = "duration %d exceeds max possible value %d" %
(duration, 255 * 1000 * 8 / self.ODRS[self.odr])
            print(error)
            raise ValueError(error)

        if duration < 1000 * 8 / self.ODRS[self.odr]:
            error = "duration %d below resolution %d" % (duration,
1000 * 8 / self.ODRS[self.odr])
            print(error)
            raise ValueError(error)

        _ths = int(127 * threshold / self.SCALES[self.full_scale])
& 0x7F
        _dur = int((duration * self.ODRS[self.odr]) / 1000 / 8)

        self.i2c.writeto_mem(ACC_I2CADDR, ACT_THS, _ths)
        self.i2c.writeto_mem(ACC_I2CADDR, ACT_DUR, _dur)

        # enable the activity/inactivity interrupt
        self.set_register(CTRL3_REG, 1, 5, 1)

        self._user_handler = handler
        self.int_pin = Pin('P13', mode=Pin.IN)
        self.int_pin.callback(trigger=Pin.IRQ_FALLING |
Pin.IRQ_RISING, handler=self._int_handler)

        # return actual used thresold and duration
        return (_ths * self.SCALES[self.full_scale] / 128, _dur *
8 * 1000 / self.ODRS[self.odr])

    def activity(self):
```

```
        if not self.debounced:
            time.sleep_ms(self.act_dur)
            self.debounced = True
        if self.int_pin():
            return True
        return False

    def _int_handler(self, pin_o):
        if self._user_handler is not None:
            self._user_handler(pin_o)
        else:
            if pin_o():
                print('Activity interrupt')
            else:
                print('Inactivity interrupt')
```

**Příloha E: knihovna urequests.py**

```
import usocket

class Response:

    def __init__(self, f):
        self.raw = f
        self.encoding = "utf-8"
        self._cached = None

    def close(self):
        if self.raw:
            self.raw.close()
            self.raw = None
        self._cached = None

    @property
    def content(self):
        if self._cached is None:
            try:
                self._cached = self.raw.read()
            finally:
                self.raw.close()
                self.raw = None
        return self._cached

    @property
    def text(self):
        return str(self.content, self.encoding)

    def json(self):
        import ujson
        return ujson.loads(self.content)


def request(method, url, data=None, json=None, headers={},
stream=None):
    try:
```

```
        proto, dummy, host, path = url.split("/", 3)
    except ValueError:
        proto, dummy, host = url.split("/", 2)
        path = ""
    if proto == "http:":
        port = 80
    elif proto == "https:":
        import ussl
        port = 443
    else:
        raise ValueError("Unsupported protocol: " + proto)

    if ":" in host:
        host, port = host.split(":", 1)
        port = int(port)

    ai = usocket.getaddrinfo(host, port)
    addr = ai[0][-1]

    s = usocket.socket()
    try:
        s.connect(addr)
        if proto == "https:":
            s = ussl.wrap_socket(s, server_hostname=host)
        s.write(b"%s /%s HTTP/1.0\r\n" % (method, path))
        if not "Host" in headers:
            s.write(b"Host: %s\r\n" % host)
        # Iterate over keys to avoid tuple alloc
        for k in headers:
            s.write(k)
            s.write(b": ")
            s.write(headers[k])
            s.write(b"\r\n")
        if json is not None:
            assert data is None
            import ujson
            data = ujson.dumps(json)
            s.write(b"Content-Type: application/json\r\n")
        if data:
            s.write(b"Content-Length: %d\r\n" % len(data))
        s.write(b"\r\n")
        if data:
            s.write(data)

        l = s.readline()
        protover, status, msg = l.split(None, 2)
        status = int(status)
        #print(protover, status, msg)
        while True:
            l = s.readline()
            if not l or l == b"\r\n":
                break
            #print(l)
            if l.startswith(b"Transfer-Encoding:"):
                if b"chunked" in l:
                    raise ValueError("Unsupported " + l)
```

```
            elif l.startswith(b"Location:") and not 200 <= status
<= 299:
                raise NotImplementedError("Redirects not yet
supported")
    except OSError:
        s.close()
        raise

    resp = Response(s)
    resp.status_code = status
    resp.reason = msg.rstrip()
    return resp


def head(url, **kw):
    return request("HEAD", url, **kw)

def get(url, **kw):
    return request("GET", url, **kw)

def post(url, **kw):
    return request("POST", url, **kw)

def put(url, **kw):
    return request("PUT", url, **kw)

def patch(url, **kw):
    return request("PATCH", url, **kw)

def delete(url, **kw):
    return request("DELETE", url, **kw)
```