VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

MĚŘICÍ STANICE S PŘENOSEM DAT PŘES LORA SÍŤ
MEASURING STATION WITH DATA TRANSMISSION VIA LORA NETWORK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
David Nguyen

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Jan Najman

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**David Nguyen**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Jan Najman**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Měřicí stanice s přenosem dat přes LoRa síť

Stručná charakteristika problematiky úkolu:

Vzdálený monitoring zařízení a ukládání naměřených dat z mnoha stanic na server je v dnešní době velmi aktuálním tématem. Cílem této práce je vytvoření jednoduché měřicí stanice, která bude schopná odesílat naměřená data na server. Přenos dat bude realizován přes LoRa síť a bude kladen důraz zejména na nízkou spotřebu, dosah a spolehlivost.

Cíle bakalářské práce:

1. Proveďte rešerši v oblasti dostupných LoRa modulů a přenosu dat ze zařízení na server.
2. Vyberte konkrétní modul a v případě potřeby navrhněte a vyrobte desku s měřicí elektronikou:
– Obvody pro měření napěťových signálů.
– Řídicí člen.
– LoRa modul + vhodná anténa.
3. Vytvořte webovou aplikaci/server pro přijímání, ukládání a zobrazování naměřených dat.

Seznam doporučené literatury:

ZÁHLAVA, Vít. Návrh a konstrukce desek plošných spojů: principy a pravidla praktického návrhu. Praha: BEN - technická literatura, 2010. ISBN 978-80-7300-266-4.

LORA Alliance, [online], Dostupné z: https://www.lora-alliance.org/.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Tato práce se zabývá přenosem dat z měřící stanice přes LoRa síť. Cíl je prozkoumat možné způsoby přenosu dat pomocí technologie LoRa a vybrat vhodný hardware pro vytvoření měřící stanice. Následovně zprovoznit modul s měřící elektronikou a vytvořit webovou aplikaci, která bude data z měřící stanice přijímat, ukládat a zobrazovat. Součásti práce je také otestovat spotřebu, dosah a spolehlivost přenosu vytvořeného modulu.

Summary

This thesis deals with the data transmission from the measuring station via LoRa network. The goal is to research possible solutions of data transmission using a LoRa technology and to select suitable hardware for creating a measuring station. Subsequently, setting up radio module with sensors and developing web application that will collect, store and present the data from the measuring station. This thesis includes tests of power consumption, range and transmission reliability of the end node.

Klíčová slova

LoRa, LoRaWAN, IoT, The Things Network, The Things Stack V3, LoPy4, PySense, MicroPython, Webová Aplikace, Full-stack Vývoj, Mechatronika

Keywords

LoRa, LoRaWAN, IoT, The Things Network, The Things Stack V3, LoPy4, PySense, MicroPython, Web Application, Full-stack Development, Mechatronics

Bibliografická Citace

NGUYEN, David. *Měřicí stanice s přenosem dat přes LoRa síť* [online]. Brno, 2021 [cit. 2021-05-20]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/132450. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Jan Najman.

Čestně prohlašuji, že tuto bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a pod vedením vedoucího.

David Nguyen

Brno . . . .

Děkuji panu Ing. Janu Najmanovi za trpělivost, technické rady a vedení práce.

David Nguyen

Obsah

1 Úvod

Díky růstu počtu chytrých zařízeních od chytrých telefonů, přes chytré vozidla až po chytré zámky, vznikly koncepty jako chytrá města, domácnosti nebo i chytré zemědělství a zdravotnictví. Co definuje daná zařízení, je jejich vzájemná komunikace buďto přes internet nebo jinou bezdrátovou síť. Tato infrastruktura chytrých zařízení dostala název, Internet věcí (Internet of Things, IoT). Rok 2020 se stal prvním momentem, kdy k internetu je připojeno více IoT zařízení než lidí a je očekáváno, že do roku 2025 bude poměr mezi těmito zařízeními vůči lidem na internetu roven čtyřem ku jedné. Jedním z hlavních důvodu tohotu růstu je drastický narůst počtu zařízení nízké spotřeby a širokého pokrytí (low-power wide-area, LPWA), do kterých patří chytré senzory, kontejnery nebo i chytré požární hydranty. Mezi technologie, která pohání narůst LPWA zařízení je LoRa technologie.

LoRa, spolu s dalšími LPWA technologiemi, doplňuje mezeru mezi již známými bezdrátovými technologiemi, jako jsou Wi-FI a LTE tím, že přináší způsob komunikace s dlouhým dosahem a nízkou spotřebou za cenu malé rychlosti přenosu dat. Díky těmto vlastnostem je LoRa technologie vhodná pro zařízení vybavená senzory, které posílá malé množství dat a nemá přístup k elektřině. Příklady využití je monitorování stavu zásilky, kde se produkt nachází, v jaké teplotě je uložena a jestli je váha zásilky zachována. Dále LoRa technologii můžeme vidět v zemědělství pro měření okolních podmínek jako je teplota a vlhkost. Výhodou oproti dalším LPWA technologiím, jako Sig-Fox nebo NB-IoT, je otevřenost LoRa protokolu, kdy uživatelé této technologie nejsou závislí na již existující infrastruktuře. Společnosti si tak mohou vytvořit vlastní privátní nebo otevřenou komunikační síť pro LoRa zařízení.

Tato bakalářská práce se věnuje vývoji měřicí stanice s přenosem dat na server přes LoRa síť. Pro dosažení zadání se využije LoRaWAN protokol, který spojuje LoRa zařízení s internetem pomocí tzv. LoRaWAN gateway. V další části budou stručně představeny koncepty jako způsob přenosu dat pomocí technologie LoRa, architekturu LoRaWAN protokolu a síťový server The Things Network, který nabízí otevřenou infrastrukturu pro LoRa zařízení. Dále se ukážou dostupné technologie pro splnění zadání, moduly pro vývoj měřící stanice, technologie potřebné pro vytvoření webové aplikace, která zobrazuje a ukládá naměřené data.

V praktické části se popíše hotová měřící stanice a webové aplikace, což zahrnuje zapojení hardwaru, strukturu projektu, potřebné knihovny a nakonec implementace kódu v LoRa zařízení a ve webové aplikaci. Nakonec zde budou zobrazeny výsledky experimentů s LoRa zařízením, jejichž cílem je zkoumat dosah a spotřebu zařízení a na čem závisí.

2 Rešerše

2.1 Technologie LoRa

LoRa je bezdrátová technologie pro přenos dat pomocí rádiových vln, patentovaná firmou Semtech. LoRa je zkratkou pro „Long Range“ a typické zařízení využívající tuto technologii má tyto vlastnosti.

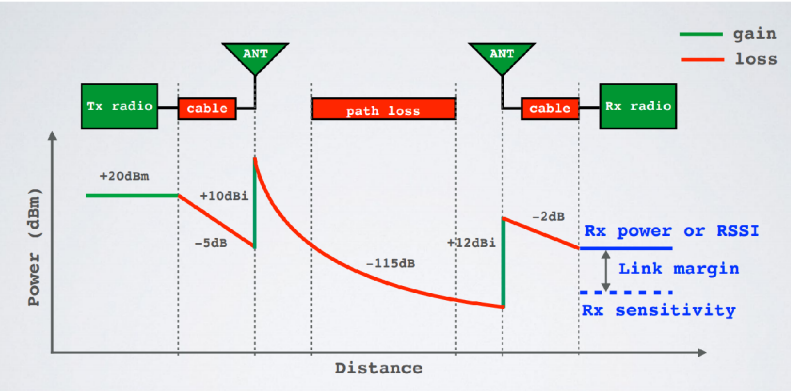
- **Velký dosah**
  LoRa zařízení dokáže přijímat a odesílat signály v městské oblasti 5 kilometrů daleko, ve venkovské oblasti je dosah až 15 kilometrů a pokud je vysílač a přijímač v přímé linii pohledu dosah signálu je pak v desítkách až stovkách kilometrů.

- **Nízká rychlost přenosu dat**
  Rychlost přenosu dat se pohybuje mezi 0,3 a 37,5 kilobitů za sekundu, kdy záleží na nastavení zařízení. LoRa technologie se používá na posílání malých dat o velikosti cca desítky bajtů, které se s časem moc nemění a posílají se v intervalu minuty nebo hodin.

- **Nízká spotřeba**
  Protože data, které LoRa zařízení vysílá jsou malé a časový rozestup mezi zprávami je velký, spotřeba takového zařízení je velice nízká a měří se v jednotkách milli wattů. LoRa vysílač může být tedy napájen z baterie a vydrží desítky let.

2.1.1 Přenos dat

Jestli přijímač zvládne zpracovat zprávu od vysílače určuje síla přijmutého signálu (Received Signal Strength Indicator, RSSI) a citlivost přijímače. RSSI i citlivost přijímače se určují v dBm. Hodnota RSSI musí být větší než hodnota citlivosti přijímače pro zpracování signálu.



Obrázek 2.1: RSSI a Rx sensitivty

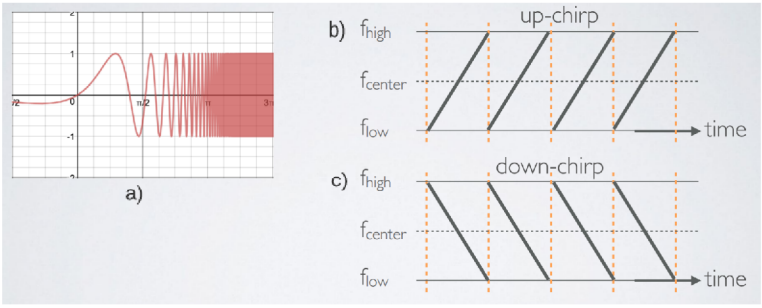
Při vysílání může být signál zesílen nebo utlumen. Zesílit signál můžeme pomocí rádiového modulu, antén a přijímače. Signál může být utlumen v kabelech a cestou vysílání (vzdálenosti, překážky, počasí). Hodnota RSSI je určena součtem těchto zisků a ztrát. Na obr 2.1 můžeme vidět závislost síly signálu na vzdálenosti vysílače od přijímače.

Pro porovnání dvou rádiových komunikací slouží link budget, popisuje nám sílu komunikace mezi vysílačem a přijímačem. Maximální Link Budget se počítá odečtením citlivosti přijimače od zisku vysílače.

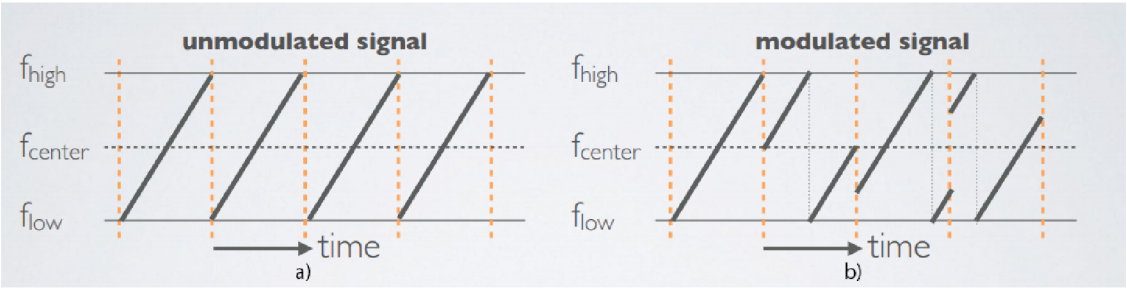
$$MaximumLinkBudget = TransmitterGain - ReceiverSensitivity \tag{2.1}$$

Zisk LoRa vysílačů, je omezen lidskými zákony a pohybuje okolo 20dBm a citlivost LoRa přijímačů může být až -148dBm. Po dosazení těchto hodnot do rovnice (2.1) získáme maximální Link Budget v hodnotě 168dBm. Pro porovnání, Wi-Fi signály mají průměrně 100dBm, důležité je si uvědomit, že závislost mezi link budgetem a vzdálenosti není lineární. Přidáním 6dBm k link budgetu zdvojnásobujeme dosah signálu. LoRa spojení, které má o 68dBm větší link budget než Wi-Fi spojení, má teoreticky 4000 krát větší dosah.

LoRa má vysoké hodnoty link budgetu, kvůli způsobu jak moduluje data. Používá Chirp Spread Spectrum (CSS) technologii pro modulaci dat. Jedná se o způsob kódování informací do tzv. chirps, což je opakovaná lineární změna frekvence, buďto rostoucí nebo klesající. Pokud frekvence s časem lineárně roste do své nejvyšší hodnoty, jedná se o up-chirp (obr. 2.2a,b) a pokud frekvence s časem lineárně klesá do své nejnižší hodnoty, jedná se o down-chirp (obr. 2.2c). Informace je vložena do těchto chirps pomocí skoků mezi frekvencemi (obr. 2.3).



Obrázek 2.2: Chirp a), b) up-chirp c) down-chirp



Obrázek 2.3: Modulace LoRa signálu a) nemodulovaný signál b) modulovaný signál

Počet bitů které můžeme vložit do jednoho chirpu je definován parametrem tzv. Spreading Factor (SF). Jeden chirp v sobě může mít 2^{SF} hodnot nebo-li chipů. Většina LoRa zařízení nabízí spreading factor od SF7 po SF12.

Rozdíl mezí horní a dolní hranicí ve které se frekvence pohybuje se nazývá šířka pásma (Bandwidth, BW). Jednotkou šířky pásma jsou Hertze.

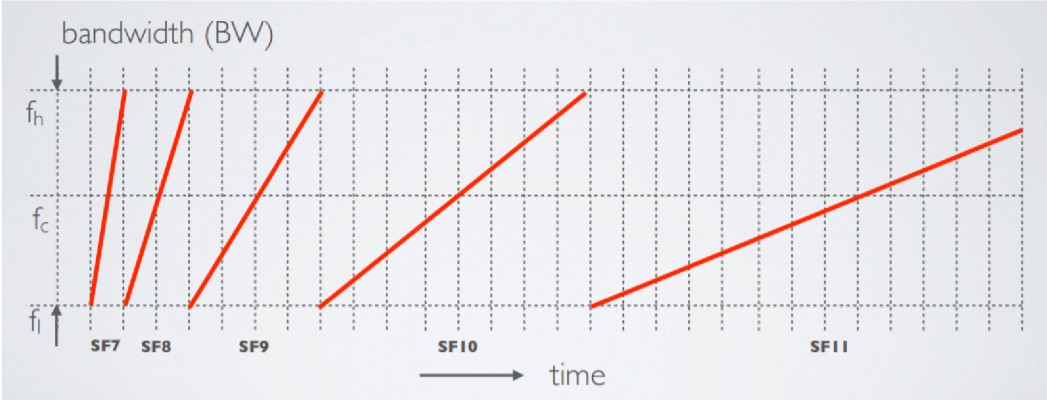
Rychlost přenosu dat (Data Rate, DR) závisí na šířce pásma, faktoru rozšiřování a coding rate (CR), který má hodnotu od 1 do 4. Vztah pro určení DR:

$$DR = SF * \frac{BW}{2^{SF}} * \frac{4}{4 + CR} \tag{2.2}$$

Další důležitou veličinou je čas posílání. Je-li přijímač velice daleko od vysílače, můžeme zvětšit dosah zvětšením času vysílání. Čas posílaní jednoho Chipu se počítá pomocí vztahu:

$$T_s = \frac{2^{SF}}{BW} \tag{2.3}$$

Ve většině případů je hodnota BW konstantní a mění se pouze SF. Ze vztahu (2.2) můžeme vidět, že zvětšením SF zmenšíme rychlost přenosu dat. Podle vztahu (2.3), zvětšením SF zvětšíme čas vysílání, tudíž i dosah signálu. Existují legislativní zákony a pravidla provozovatele sítě (viz. kapitola 2.2.3 a 2.4.2), které limitují čas vysílání LoRa zařízení, tudíž zvětšením SF také limitujeme, jak často můžeme zprávy posílat.

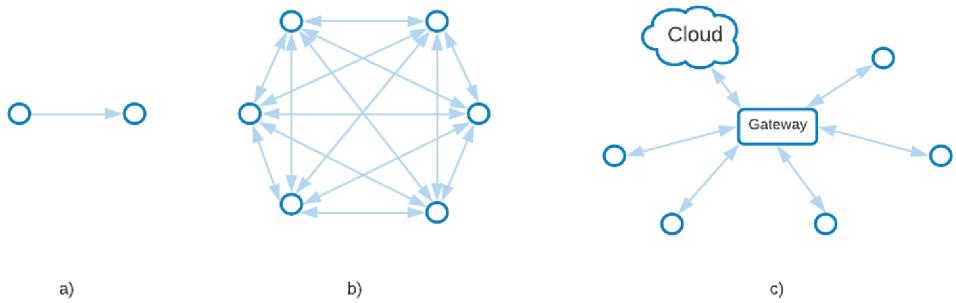


Obrázek 2.4: Chirp při různých SF

2.1.2 LoRa a IoT

LoRa komunikace může být buďto one-to-one (obr. 2.5a), tedy jedno zařízení je vysílač a druhé je přijímač. Dále LoRa Mesh (obr. 2.5b), což je síť LoRa zařízení, které komunikují mezi sebou. A nebo LoRaWAN (obr. 2.5c), což je komunikační protokol, kdy zařízení, tzv. LoRaWAN Gateway, přijímá zprávy od několik LoRa zařízení, které posílá na síťový server, jedná se o hvězdicovou topologii. V praktické části se bude používat LoRaWAN protokol.

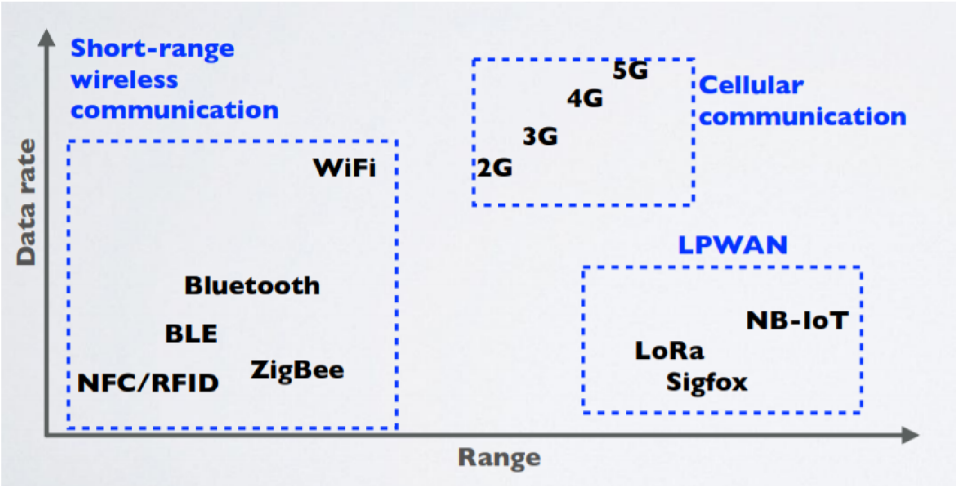
LoRa zařízení pracující na LoRaWAN protokolu spadají do kategorie Internet věcí (Internet of Things, IoT), což je síť zařízení, „věcí“, ,které jsou vybavené senzory, softwarem,



Obrázek 2.5: Druhy LoRa komunikací a) one-to-one b) LoRa Mesh c) LoRaWAN

dalšími technologiemi. Tyto zařízení musí mít síťovou konektivitou, která jim umožňuje vzájemnou komunikaci a výměnu dat přes internet.

Technologie, které spadají do IoT (obr. 2.6 a tab. 2.1) můžou být bezdrátové technologie krátkého dosahu (Wi-Fi, Bluetooth), bezdrátové technologie středního dosahu (3G, 4G, 5G) a bezdrátové technologie dlouhého dosahu, kam spadá LoRaWAN, konkrétněji do kategorie Low Power Wide Area Network (LPWAN).



Obrázek 2.6: Rozdělení IoT technologií podle dosahu [5]

Bezdrátová technologie	Dosah [m]	Spotřeba [mW]
Bluetooth	~50	~2,5
WiFi	~50	~80
3G/4G	~5000	~500
LoRa	2000-5000 (městká oblast) 5000-15000 (venkovská oblast) >15000 (přímá linie pohledu)	20

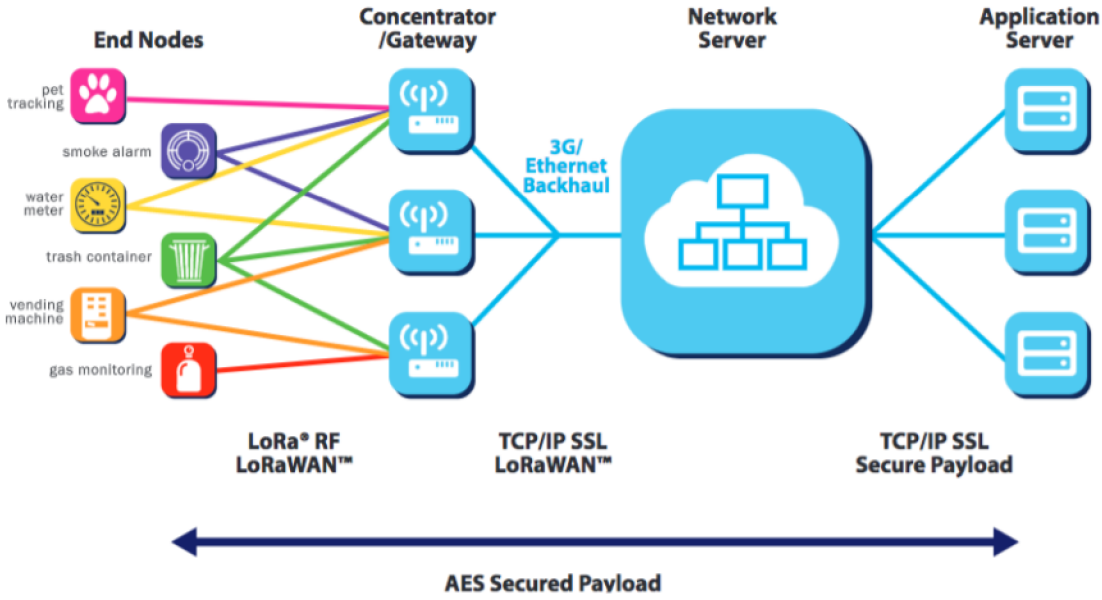
Tabulka 2.1: Porovnání různých bezdrátových technologií

2.2 LoRaWAN Protokol

LoRaWAN je komunikační protokol pro LoRa zařízení, který vytvořila asociace LoRa Alliance. Na jedné straně máme hardware, který posílá data pomocí technologie LoRa a na druhém konci máme IoT aplikaci nasazenou na serveru. LoRaWAN slouží jako oboustranná komunikace, tudíž zařízení může posílat data na server a zároveň můžeme posílat ze serveru data do hardwaru.

Využití LoRaWAN architektury můžeme vidět v chytrém městě (ovládání světla, parkování, monitorování znečistění), průmyslu (sledování zásilek), zemědělství (monitorování teploty, vlhkosti, hladiny vody), chytré domácnosti (automatizace domácích spotřebičů) nebo i ve zdravotnictví (monitorování zdraví skrze nositelné technologie).

2.2.1 Topologie



Obrázek 2.7: Topologie LoRaWAN protokolu [1]

Na obr 2.7 můžeme vidět architekturu LoRaWAN, která je tvořena z:

- **End node**
  End node, nebo také end device, je zařízení vybavené senzory a LoRa rádiovým modulem pomocí kterého může posílat data a zároveň může přijímat data. Zpráva, kterou end node pošle, se nazývá uplink (ty se dělí ještě na uplinky potvrzené a nepotvrzené) a zpráva kterou end node přijme se nazývá downlink. End node slouží, v některých aplikacích, jako aktuátor (ovládání světla, zámku, ventilů atd.).

- **Gateway**
  Zařízení připojené k internetu, které dokáže dekódovat rádiový signál z LoRa end node a posílá ho na síťový server. Slouží jako přístupový bod mezi LoRa zařízeními a internetem, něco jako Wi-Fi router. LoRaWAN gateway přijímá data od všech blízkých end nodes a end node posílá data na všechny možné gatewaye.

- **Síťový server**
  Síťový server se stará o zpracovávání dat. Jelikož end device posílá zprávy na všechny možné gatewaye, vznikají duplikátní zprávy, které síťový server filtruje. Dešifruje příchozích zprávy a šifruje odchozích zprávy. Vybírá nejvhodnější gateway pro poslání downlinku. Přeposílá uplinky na aplikační server.

- **Aplikační server**
  Server, kde se nachází samotná IoT aplikace, která pracuje s daty od end device. Tento server může být na privátním nebo na veřejném cloudu. Tato aplikace se stará o vytváření downlinku, které přepošle skrze síťový server na end node.

Výhodou LoRaWAN protokolu je jeho škálovatelnost. Nejlevnější druh gatewaye může zpracovat statisíce zpráv za den. Pokud by jeden LoRa end node posílal 10 zpráv za den, tak by zmíněný gateway dokázal přijímat zprávy od 10 000 end nodes. Kdyby bylo potřebné zpracovat více zpráv, stačí přidat další gateway.

Další výhodou je cena pokrytí. Jelikož se jedná o hvězdicovou strukturu, kdy se jeden gateway postará o několik end nodes a tyto zařízení mají dosah v kilometrech, stačí relativně málo gatewayes, aby se vytvořilo rozsáhlé pokrytí pro LoRa zařízení v okolí. Cena jednoho gateway se pohybuje v řádech tisíců korun.

2.2.2 Rozdělení zařízení podle tříd

Výše již bylo řečeno, že LoRa end node dokáže posílat zprávy a zároveň dokáže přijímat zprávy. Existují ale případy, kdy chceme využít end node pouze na posílání dat a nechceme tedy plýtvat baterií tím, že end node bude čekat na příchozí zprávy. Podle daného režimu provozu dělíme LoRa end nody do třech tříd:

- **Třída A**
  Zařízení třídy A může přijímat downlinky pouze, když odesílá uplinky. Toto zařízení je většinu času v režimu spánku. Zařízení se vzbudí, pokud uběhne čas nastaveného spánku, nebo pokud je nastavené, aby se vzbudilo při změně monitorované veličiny. Toto zařízení nemůže vzbudit IoT aplikace, kvůli této limitaci se tyto zařízení nehodí jako aktuátory, ale mají nejnižší spotřebu ze všech dalších možných tříd. Všechny LoRa end nodes mohou být zařízení třídy A.

- **Třída B**
  End node, který podporuje nastavení třídy B, má možnost často přijímat zprávy. Nastaví se čas, po kterém bude zařízení naslouchat, zda nejsou dostupné downlinky. Kvůli této vlastnosti se zařízení třídy B hodí i jako aktuátory.

- **Třída C**
  Poslední možné nastavení zařízení je podle třídy C. Tento end node vždy čeká, zda nejsou dostupné dowlinky, tudíž je časové okno mezi poslanou zprávou z aplikace a přijmutí zprávy zařízením minimální. Zařízení třídy C se nehodí, aby bylo napájeno pomocí nízkokapacitní baterie.

2.2.3 Regulace

LoRa Alliance vytvořila dokument *LoRaWAN Regional Parameters*[7], kde je specifikované, v jakém frekvenčním pásmu může LoRa end node pracovat. Tohle záleží, v jakém

Region	**Frekvence [MHz]**
Asie	433
Evropa, Rusko, India, části Afriky	863-870
USA	902-928
Australie	915-928
Kanada	779-787
Čína	779-787, 470-510

Tabulka 2.2: Frekvenční plán pro různé regiony [8]

regionu se zařízení nachází. Pro end node umístěný v Evropě, může vysílat na frekvencích od 863MHz do 870MHz (Tab 2.2).

Zařízení má také omezený pracovní cyklus (duty cycle) na 1% nebo 0,1%, kdy záleží na použitém kanálu. Zařízení s 1% duty cycle mohou vysílat 36 sekund za hodinu.

Dále existují omezení pro uplinky a dowlinky pro LoRaWAN end nodes v Evropě. Přenosový výkon nesmí přesáhnout 25mW pro uplinky a 0,5W pro downlinky.

Kromě těchto pravidel můžou dále existovat restrikce zavedené jednotlivými síťovými servery (viz. kapitola 2.4.2).

2.2.4 Zabezpečení

Před tím, než bude end device moci komunikovat pomocí LoRaWAN sítě, musí být toto LoRa zařízení aktivováno. Aktivované zařízení má v sobě tyto hodnoty:

- **Device address (DevAddr)**
  DevAddr je 32 bitové číslo, které identifikuje end device s používaným síťovým serverem. Je to hodnota přidělena zařízení síťovým serverem.

- **Network session key (NwkSKey)**
  Jedná se o klíč, který využívá end device a síťový server při vzájemné komunikaci. Používá se pro kódování odchozích zpráv a dekódování příchozích zpráv.

- **Application session key (AppSKey)**
  Tento klíč se používá při end-to-end komunikaci, tedy při komunikaci mezi end device a aplikačním serverem. Opět se používá pro kódování a dekódování příchozích a odchozích zpráv. Zprávy mezi aplikačním serverem a end device musí projít síťovým serverem, který může ovlivnit průchozí zprávy, ale nemůže nahlédnout do obsahu. Síťové servery se považují za důvěryhodné.

End device můžeme aktivovat dvěma způsoby, Over The Air Activation (OTAA) a Activation By Personalization (ABP). Kdy OTAA je bezpečnější a preferovaná forma aktivace. ABP se používá především při vývoji aplikace.

Over The Air Activation (OTAA)

Před tím, než se zařízení pokusí aktivovat, musí v sobě mít uložené tři hodnoty. DevEUI, což je 64 bitový unikátní identifikátor zařízení, většina zařízení má DevEUI v sobě už zabudované. AppUEI, což je indentifikátor aplikačního serveru na který chce end device

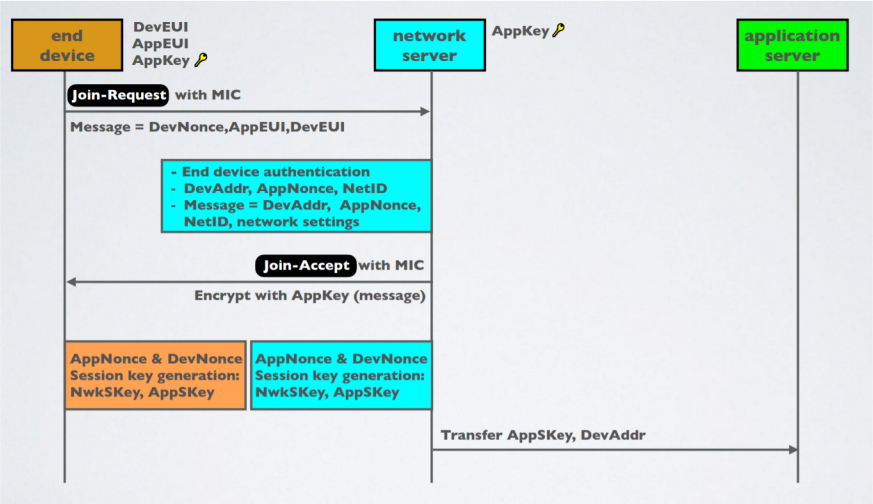
posílat zprávy. A AppKey, klíč, který musí mít jak zařízení, tak síťový server stejný, využívá se pro ověření integrity zpráv.

End device vytvoří Join-Request, který obsahuje DevEUI, AppEUI, a k tomu ještě dvě nové hodnoty. DevNonce je náhodně vygenerované číslo, které si zařízení i síťový server uloží. Využívá se při opakovaném připojení, aby zařízení, které bylo probuzené ze spánku nemuselo projít celou aktivací znovu. Další hodnota je Message Integrity Code (MIC), která se stará o integritu zprávy a je vygenerována pomocí AppKey.

Jakmile síťový server příjme Join-Request, podívá se jestli již DevNonce nebyl kdysi použit, pokud ano, šlo by o znovu připojení. Pokud se jedná o nový DevNonce, ověří síťový server MIC hodnotu pomocí svého AppKey.

Jestli Join-Request projde úspěšně ověřením, vytvoří síťový server Join-Accept. Join-Accept obsahuje DevAddr, který zmapuje DevEUI zařízení do kratší (32 bitové) adresy pamětí síťového serveru. Dále AppNonce nebo-li JoinNonce, což je náhodně vygenerované číslo. Pak v Join-Accept zprávě najdeme identifikátor a nastavení sítě. Join-Accept je zašiforvána pomocí AppKey.

Po úspěšném přijmutí zprávy Join-Accept zařízením, sdílí LoRa zařízení a síťový server AppNonce a DevNonce, který se využije na vygenerování NwkSKey a AppSKey, síťový server předá AppsKey a DevAddr aplikačnímu serveru, což umožní komunikaci jak mezi end device a síťovým serverem, tak mezi end device a aplikačním serverem. Proces aktivace zařízení pomocí OTAA můžeme vidět na obr 2.8.



Obrázek 2.8: Aktivace pomocí OTAA [5]

Access By Personalization (ABP)

Při ABP aktivaci se přeskočí proces Join-Request a Join-Accept. Místo toho jsou hodnoty DevAddr, AppSKey a NwkSKey vloženy přímo do end node. Síťovému serveru jsou hodnoty DevAddr a NwkSKey přednastaveny a aplikačnímu serveru jsou hodnoty DevAddr a AppSKey přednastaveny také.

ABP urychluje aktivaci zařízení tím, že se nečeká na Join-Accept od síťového serveru. Také může zvýšit dosah zařízení. Pokud by se gateway, co posílá Join-Accept při OTAA aktivaci nacházel daleko od zařízení, je možné, že by Join-Accept nedošel v čas, kdy

zařízení čeká na příchozí zprávy.

Problém ABP je, že snižuje bezpečnost celé aplikace. Při OTAA aktivaci se každému zařízení vygeneruje unikátní DevAddr, AppSKey a NwkSKey. Při ABP aktivaci jsou tyto citlivé hodnoty vloženy přímo do zařízení, která komunikují s naším aplikačním serverem a mají tyto hodnoty stejné. Proto ohrožení jednoho zařízení může ohrozit všechny ostatní, které komunikují se stejným serverem.

Další nevýhodou ABP je fixní hodnota DevAddr, která umožňuje end node komunikaci pouze s definovaným serverem. Pokud bychom měnili nastavení aplikačního serveru, museli bychom přeprogramovat všechny zařízení, které s tímto serverem komunikují.

Dále síťový server i zařízení mají v sobě integrovaný frame counter, který počítá kolikátá zpráva se posílá. Hodnoty těchto počítačů musí být synchronizovány, aby byl uplink přijat serverem. Pokud se něco stane se zařízením a restartuje se mu frame counter, způsobí to odmítnutí uplinku. OTAA aktivace synchronizuje frame countery při Join-Request procesu na rozdíl od ABP.

2.3 LoRa End Node

Hardware

Všechny LoRa end nody musí mít LoRa rádiový modul, mikrokontroler a anténu. Dalším běžným hardwarem jsou senzory a aktuátory.

LoRa rádiový modul nebo LoRa čip je hardware, co se stará o vysílání a přijímání dat. Zvolený čip určuje maximální link budget, zisk vysiláče, citlivost přijímače, maximální možnou přenosovou rychlost, možnou třídu provozu a potřebnou knihovnu na programování čipu.

Miktrokontroler se stará o zpracování odchozích dat, které nejčastěji získává z nějakého senzoru a příchozích dat. Výběr miktrokontroleru má největší vliv na spotřebu celého LoRa end node a definuje programovací jazyk.

Anténa přidává gain a tím pádem zvětšuje link budget.

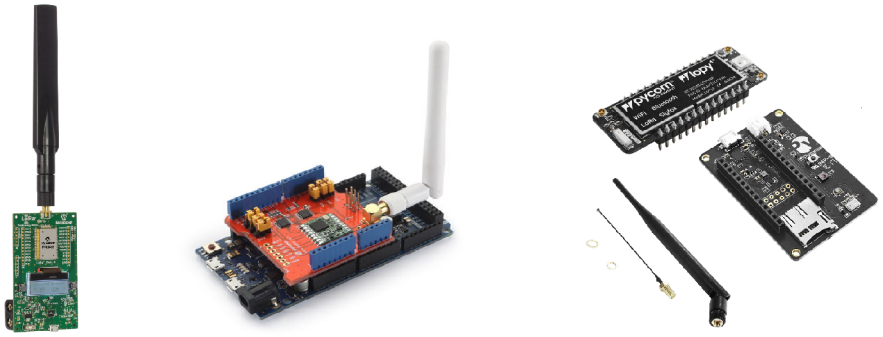
LoRa vývojové desky

Existuje mnoho LoRa vývojových desek, které už mají většinu hardwarových součástí integrovaných. Do praktické části byly zvažovány tři vývojové desky.

RN2483 LoRa Technology Mote (obr. 2.9a) od Microchip, který v sobě má zabudovaný LoRa rádiový modul, mikrokontroler, konektor pro externí anténu, senzory, USB port pro programování desky a konektor pro zdroj napájení. Desce chybí je programátor mikrokontroleru PICkit.

Dragino LoRa shield (obr. 2.9b) má LoRa rádiový modul, konektor pro externí anténu a senzory. Této desce chybí mikrokontroler, USB port pro programování desky a zdroj napájení. Deska je kompatibilní s mikrokontrolerem Arduino UNO, který v sobě má všechny chybějící komponenty.

LoPy4 (obr. 2.9c) od firmy Pycom, tato deska v sobě má zabudovaný LoRa rádiový modul, mikrokontroler a konektor pro externí anténu. K této desce by bylo nutné přikoupit desku PySense 2.0 X, která má chybějící komponenty, senzory, USB port pro programování desky a konektor pro zdroj napájení.



Obrázek 2.9: LoRa vývojové desky a) RN2483 LoRa Mote b) Dragino LoRa shiled a ArduinoUNO klon c) LoPy4 a PySense

Porovnání vývojových desek

Vývojová deska	**RN2483 Mote**		**LoPy4**		**Dragino shield**	
Extra hardware	PICkit		PySense		ArduinoUNO	
LoRa RF modul	RN2483		SX1276		RFM95W	
Max. Link Budget [dB]	N/A		168		164	
Gain [dBm]	14		14		14	
Citlivost přijímače [dBm]	-146		-148		-144	
Max. přenosová rychlost [kbps]	300		300		300	
Možné třídy provozu	A		A, C		N/A	
Mikrokontroler	PIC18LF45K50		ESP32		ArduinoUNO	
Programovací jazyk	C		MicroPython		C, C++	
Konektor antény	SMA		IPEX MHF		SMA	
Možné zdroje napájení	USB, AAA baterie		USB, LiPo baterie		USB	
Senzory	Teploty, světla		Teploty, vlhkosti, světla, tlaku, zrychlení		Teploty	
Spotřeba v režimu spánku [µA]	Total	16	LoPy4 PySense Total	1 9 10	Dragino shield ArduinoUNO Total	1 31 32
Další funkce	LED display, tlačítka		WiFi, Bluetooth, SigFox, Micro SD card slot		Spájené pin headers pro output piny	

Tabulka 2.3: Porovnání vývojových desek [12][9][10][11][13]

V tabulce 2.3 můžeme vidět vlastnosti desek získané z datasheetu. Do praktické části byla zvolená deska LoPy4 s PySense, proto je popis tohoto produktu rozsáhlejší oproti zbývajícím.

- **RN2483 LoRa Mote**
  LoRa Mote od firmy Microchip má LoRa čip RN2483 vyvinutý také firmou Microchip. K produktu nenalezneme mnoho komunitní podpory na internetu. Na druhou stranu, firma Microchip má všechny hardwarové součástky velice dobře dokumentované. Desku je možné napájet pomocí AAA baterie. Mikrokontroler se programuje v jazyce C, což dává vývojáři velkou kontrolu nad produktem.

- **Dragino LoRa shield**
  Dragino LoRa shield s čipem RFM95 od firmy HOPERF má výhodu v tom, že se používá s Arduinem UNO nebo jeho klonem, což je jeden z nejpopulárnějších mikrokontrolerů a najdeme na internetu spoustu podpory, jak od komunity, tak od vývojářů produktu. Další výhodou jsou již spájené header piny, tudíž bychom měli přístup k mnoho input a output pinům, které na Arduinu najdeme a to by umožnilo využít tento produkt i jako akční člen, který by mohl pracovat i s mechanickými součástkami. Nevýhodou je vysoká spotřeba způsobená mikrokontrolerem a jeho nekompatibilita s nízkonapěťovými bateriemi, což je jedna z typických vlastností LoRa zařízení.

- **LoPy4**
  Poslední varianta, kombinace LoPy4 a PySense of firmy Pycom, využívá LoRa čip SX1276 od firmy Semtech, která vlastní technologii LoRa. SX1276 má ze všech zmíněných modulů nejlepší link budget, dosah, citlivost přijímače a jako jediný modul nabízí třídu provozu C.

  Deska PySense nabízí širokou škálu senzorů, kterou je možné napájet pomocí LiPo baterie. LoPy4 a PySense mají nejnižší spotřebu v režimu spánku. LoPy4 nabízí také bezdrátovou komunikaci pomocí WiFi, Bluetooth a SigFox, což rozšiřuje IoT využití a nabízí velkou flexibilitu s daty.

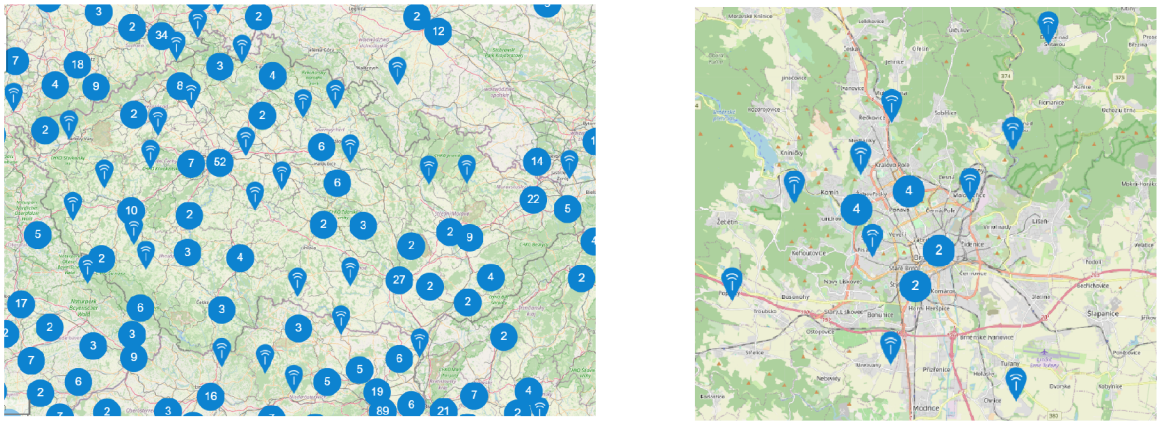
  Deska se programuje v jazyce MicroPython. Na internetu najdeme bohatou podporu komunity a Pycom svoje knihovny, které se využívají na programování desky, dobře dokumentuje. MicroPython je interpretovaný jazyk a ne kompilovaný jazyk jako C a C++, tudíž při vývoji můžeme používat příkazový řádek REPL pro exekuci kódu, což dává vývojáři možnost rychlého testování kódu a pomáhá při debugování. Nevýhodou interpretovaných jazyků je, že jsou náročnější na paměť zařízení, která je limitovaná při práci s mikrokontrolery. PySense nabízí možnost vložení SD karty pro rozšíření paměti. Další nevýhodou MicroPythonu je, že programátor ztrácí kontrolu nad hardwerem, kterou by měl při využívání jazyku C.

  LoPy4 nenabízí SMA konektor k anténě jako zbývající desky. Anténu nelze připojit přímo k desce, místo toho se musí použít koaxiální kabel, který způsobí zeslabení signálu.

2.4 The Things Network TTN

The Things Network (TTN) je infrastruktura pro IoT. TTN nabízí LoRaWAN síťový server The Things Stack (TTS), který je otevřený, decentralizovaný a volně k využití. Všechny gatewaye zaregistrované na TTS jsou volně přístupné pro všechny. TTN také nabízí nástroje pro vývojáře LoRaWAN aplikací, jako gatewaye, LoRa end nody a aplikační servery.

V praktické části se bude využívat The Things Stack V3 (TTS V3), což je server co podporuje všechny třídy provozu (A, B a C) a všechny regionální parametry. Nabízí hlubokou bezpečnost jak síťového, tak aplikačního serveru. Dále je možná integrace mezi vlastní aplikací a aplikačním serverem běžící na TTS V3. Můžeme připojit vlastní gatewaye na síťový server TTS V3. V poslední řadě můžeme monitorovat skrze konzoli TTN zprávy, které projdou skrze naše gatewaye, připojení našeho end node, příchozí uplinky od našich LoRa zařízení a připravené downlinky, které se pošlou na naše zařízení.



Obrázek 2.10: Gatewaye nasazené pomocí TTN a) v České Republice b) v Brně

V České Republice najdeme široké pokrytí LoRaWAN sítě pomocí gatewayů, které jsou zaregistrované na The Things Network. Na obr. 2.10 můžeme vidět, přibližné umístění LoRaWAN gatewayů v České Republice a v Brně.

2.4.1 Služby TTN

Vytvoření aplikace

Po přihlášení do konzole The Things Stack V3, můžeme založit aplikační server, kterému zvolíme Application ID.

Možné funkce aplikace:

- **End devices**, zde nalezneme naše zařízení nebo můžeme přidat nové.
- **Live data** je konzole, kde můžeme monitorovat chod dat aplikace.
- **Payload formatters** nám umožní formátovat příchozí i odchozí data (například z Hex do Json).

- **Integrations** slouží pro integraci s naší vlastní aplikací, buďto pomocí MQTT, Webhooks, Storage Integration, AWS IoT a nebo LoRa Cloud.
- **Collaborators**, místo kde můžeme přidat spolupracovníky, co se budou podílet na vývoji aplikace a přidělit jim různé práva.
- **API keys**, tady vytvoříme klíč, pomocí kterého budeme moct přistoupit do aplikačního serveru skrze externí aplikace.

Registrace LoRa end device

Při registraci našeho LoRa zařízení máme dvě možnosti jak registrovat zařízení, buďto využijeme existující zařízení z LoRaWAN Device Repository a nebo manuálně registrujeme zařízení. V této práci se bude popisovat manuální postup registrace.

- Vybereme jakým způsobem chceme zařízení aktivovat: OTAA, ABP, multicast (variace ABP) a nebo nekonfigurovat.
- Formulář sám nastaví síťový, aplikační i join server, ale máme možnost použít i jiný.
- Zvolíme unikátní identifikátor, End device ID, pro naše zařízení. Nastavíme frekvenční plán podle umístění end device.
- Pomocí datasheetu zařízení zvolíme DevEUI, podporovanou LoRaWAN verzi, verzi regionálního parametru a třídy provozu.
- Zbytek registrace se liší podle způsobu aktivace. Pokud jsme zvolili OTAA, můžeme určit hodnoty AppEUI a AppKey, nebo je necháme vygenerovat. Pokud jsme zvolili ABP, můžeme určit hodnoty DevAddr, NwkSKey a AppSKey, nebo je necháme vygenerovat.

Zobrazování dat na TTN

Konzole TTS V3 slouží především k monitorování průchozích dat síťovým serverem. Vidíme, kdy zprava přišla na server, o jaký typ zprávy se jedná a náhled na obsah. Můžeme vidět zpracování join-request zprávy, přeposlání join-accept, zpracování uplinku a připravení downlinku.

2.4.2 Omezení

Kromě regulací určené LoRaWAN protokolem (2.2.3), existují regulace a omezení síťového serveru a to platí i při využívání síťového serveru TTN, který využívá tzn. *Fair Use Policy* [15].

TTN udává velikostní limit pro odesílané uplinky a downlinky. Tento limit závisí především na zvoleném SF.

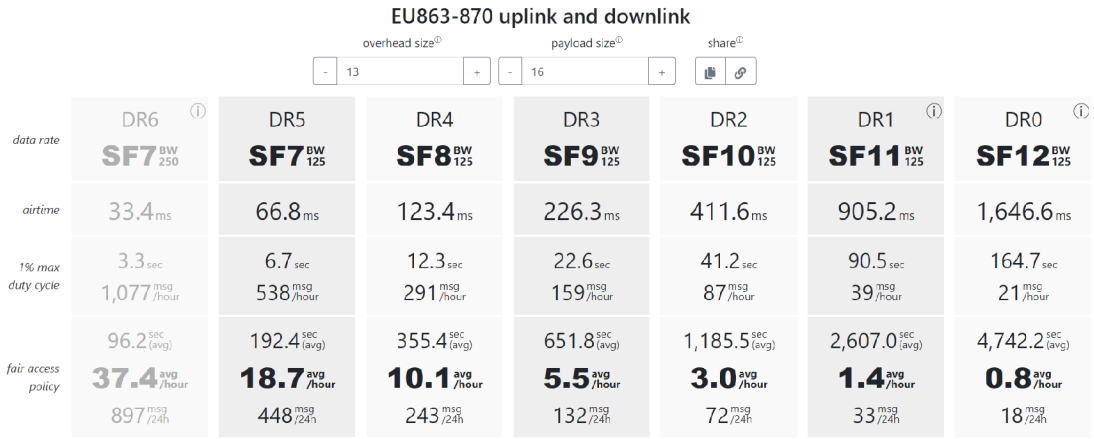
- Pro S7 a SF8 při BW 125kHz je horní limit 222 bytů.
- Pro SF9 při BW 125kHz je horní limit 115 bytů.
- Pro SF10, SF11 a SF12 při BW 125kHz je horní limit 51 bytů.

Je důležité si uvědomit, že LoRaWAN protokol přidává každé zprávě minimálně 13 bytů (DevAddr, MIC, port sítě a informace o nastavení sítě).

Dále TTN povoluje pouze 10 downlinků za 24 hodin a to včetně potvrzení uplinků.

Jedno zařízení má omezený čas vysílání na 30 sekund za 24 hodin. Vztah(2.3), nám říká, že čas vysílání závisí na velikosti zprávy a zvoleném SF při konstantním BW. LoRa zařízení musíme naprogramovat tak, aby časové rozestupy mezi vysíláním byli dostatečně velké, aby celkový čas vysílání za 24 hodin nepřekročil hranici 30 sekund.

Existuje online kalkulačka[16], do které vložíme velikost zprávy a ta nám vypočítá minimální časový rozestup mezi vysíláním pro různé SF. Na obr. 2.11 je nastavená velikost zpráv na 16 bytů, v druhé řadě vidíme čas vysílání jedné zprávy a v třetí minimální rozestup mezi zprávami. Toto platí pro LoRaWAN protokol, v poslední řadě máme „fair access policy“, který využívá TTN a určuje maximální čas vysílání za den. Podle online kalkulačky, kdybychom vysílali celý den a posílali 16 bytů s SF7, můžeme poslat 448 zpráv za den a časový rozestup musí být minimálně 192,4 sekund.



Obrázek 2.11: Online kalkulačka pro výpočet času vysílání [16]

2.5 IoT Aplikace

TTN umožňuje získat data z LoRa end node. Pokud ale chceme dále pracovat s informací, které zařízení LoRa odesílá, potřebujeme samotnou IoT aplikaci, která data z aplikačního serveru TTN získá a bude s nimi dále pracovat.

Nejpopulárnější formy IoT aplikce jsou desktopové, mobilní nebo webové. V praktické části se bude tvořit webová aplikace, proto se v této sekci budeme soustředit na tuto formu.

Webová aplikace

Jediné, co uživatel potřebuje pro spuštění webové aplikace je internetový prohlížeč a přístup k internetu. Webová aplikace se skládá ze dvou částí backend a frontend.

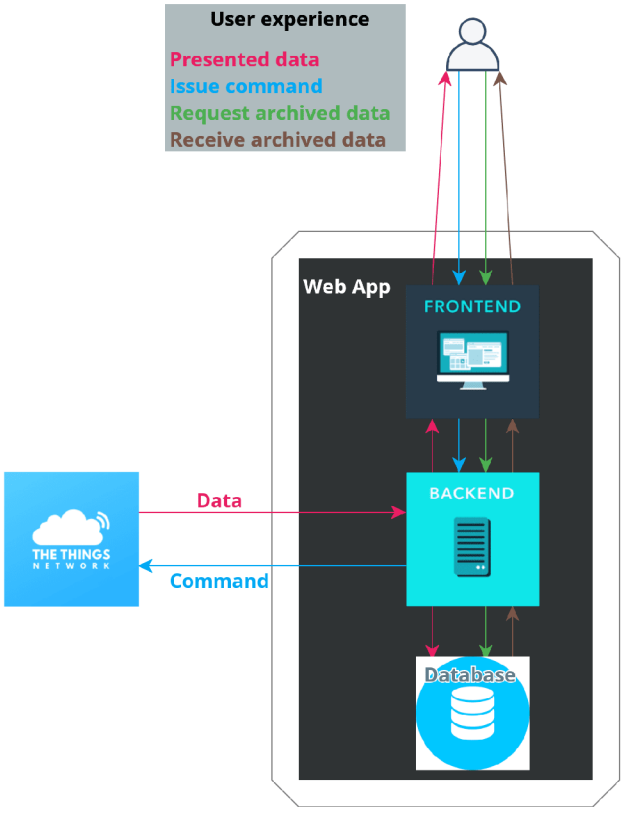
Backend je část, která se stará o server aplikace, o samotnou funkcionalitu aplikace, o práci s daty, databázi a více. Backend je část webové aplikace, která bude komunikovat s TTN a zpracovávat data. Typické nástroje pro vývoj backendu jsou programovací jazyky

jako PHP, Ruby, Java, Python, Golang a více. Dále databázové technologie jako SQL, NoSQL, MongoDB, Oracle a další.

Frontend je část webové aplikace, se kterou interaktuje uživatel. Frontend se stará o text, obrázky, menu, tlačítka atd. Práce frontendu bude prezentovat získané data uživateli a přeposílat požadavky uživatele backendu. Pro vývoj frontendu je potřebné znát HTML, CSS a Javascript, další pomůcky při programování frontendu můžou být Bootstrap, jQuery, Ajax atd.

Jelikož LoRaWAN nabízí oboustrannou komunikaci, budeme moci na webové aplikaci implementovat jak zobrazování získaných dat, tak komunikaci s LoRa end node, kdy uživatel bude moci poslat nějaký příkaz. Uživateli také můžeme nabídnout komunikaci s databází pro získání starších dat, které nechceme zobrazovat na frontendu.

Přibližnou architekturu webové aplikace můžeme vidět na obr. 2.12.



Obrázek 2.12: Topologie webové aplikace

Jak už bylo výše zmíněno, backend se stará o spuštění serveru aplikace. Programovací jazyky pro vývoj backendu nám nabízí spoustu možností, jak spustit lokální server na kterém aplikace bude běžet. Pokud, ale chceme, aby webová aplikace byla přístupná nejen na lokálním zařízení, musí se aplikace nasadit na cloud. Jednou z možných řešení je model cloudového computingu zvaný Platform as a Service (PaaS). Do modelu se vloží celá infrastruktura webové aplikace a PaaS poté vytvoří veřejný server podle tohoto modelu.

2.6 Zvolená technologie

V praktické části se používá komunikace pomocí LoRaWAN protokolu, pro funkčnost byly zvoleny následující technologie:

LoRa end node

Jako LoRa zařízení bylo zvoleno LoPy4 a PySense od firmy Pycom. Důvodem byla široká nabídka senzoru od desky PySense, velký link budget LoRa rádiového modulu SX1276, nízká spotřeba obou zařízení, možnosti napájení skrze baterii a také programovací jazyk MicroPython. S jazykem Python již mám zkušenosti, který je MicroPythonu velice podobný.

LoRa end node bude pracovat jako třída A, frekvence bude nastavena podle Evropského regionu, tudíž na 868MHz. Pro aktivaci použije OTAA, která má na rozdíl od ABP vyšší bezpečnost. Bude posílat data ze senzoru, konkrétně okolní teplotu, barometrický tlak a napětí z baterie, jako nepotvrzený uplink. Uživatel bude moct měnit barvu LED diody a hodnotu SF. Podle nastavené hodnoty SF se nastaví i čas režimu spánku, aby end node neporušil omezení nastavené síťovým serverem.

Pycom k LoPy4 nabízí anténu, která nakonec byla zvolena, jelikož nabízela ze všech dostupných antén největší gain.

Síťový server

The Things Stack V3 od The Things Network byl vybrán jako síťový server. Jedná se o bezplatnou službu, kde všichni uživatele mohou používat LoRaWAN gatewaye od všech uživatelu, což vytváří celosvětové pokrytí LoRaWAN. Dále TTN nabízí vytvoření aplikačního serveru, ve kterém můžeme integrovat komunikaci s IoT aplikací.

IoT aplikace

IoT aplikace byla vytvořena jako webová aplikace.

Backend aplikace byl napsán v Pythonu, kvůli vlastní zkušenosti s tímto jazykem. Server se vytvořil pomocí Flask, což je Python framework pro vytvoření webové aplikace. Flasku nabízí možnost rychlého nasazení, což je výhodou při tvoření menší aplikace na které se podílí malé množství lidí.

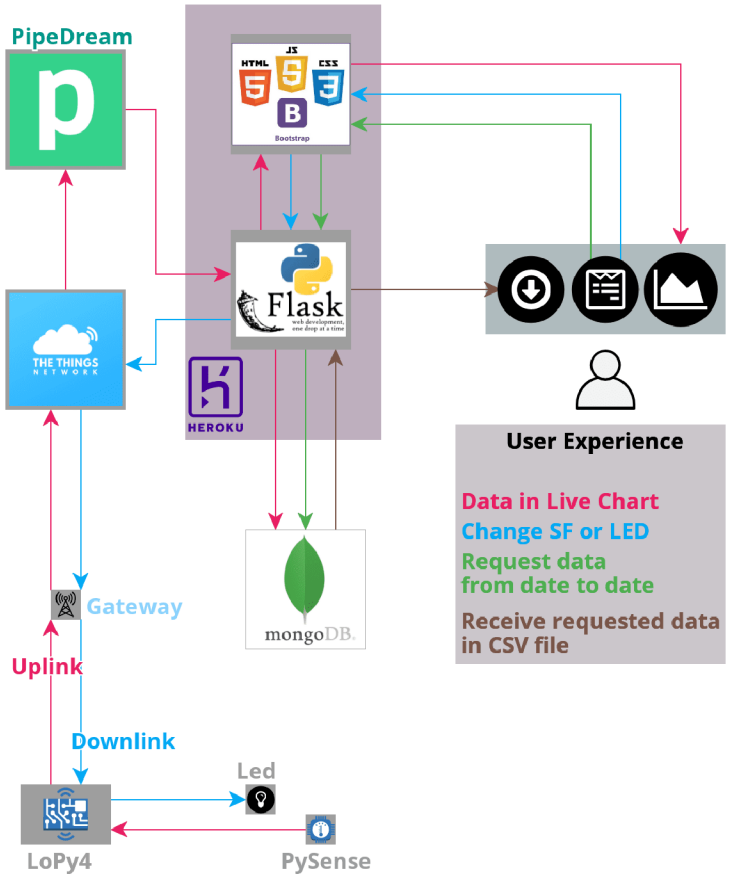
Jako databáze byla zvolena MongoDB kvůli předešlé zkušenosti z touto databází.

Frontend byl napsaný ve HTML, CSS a JavaScriptu, jelikož se jedná o nejpopulárnější metodu na tvoření frontendu s velkou škálou návodu na internetu.

Webová aplikace byla nasazena na cloud pomocí Heroku. Heroku je cloud PaaS, která nabízí bezplatnou službu a podporuje programující jazyk Python.

3 Praktická část

3.1 Architektura projektu



Obrázek 3.1: Architektura projektu

Na obr. 3.1 můžeme vidět architekturu projektu. Červená čára značí tok dat z LoPy4 k uživateli webové aplikace, modrá čára značí příkaz poslán uživatelem do LoPy4, zelená čára značí žádost uživatele o data z databáze a hnědá čára značí cestu souboru z databáze pro uživatele.

Můžeme vidět, že tok červené čáry začíná u senzoru PySense a pokračuje do LoPy4. LoPy4 vytvoří uplink, který pošle na TTN skrze LoRaWAN gateway.

Uplink nejde přímo z TTN do webové aplikace, ale projde ještě bločkem PipeDream. Jeden z hlavních způsobů, jak spolu aplikace na internetu komunikují, je Rest API, kdy jedna strana pošle HTTP Request a pokud druhá strana request přijme, odešle zpátky

vyžadovaný response. V době psaní této práce nebyla TTS V3 API plně funkční, proto se využilo třetí strany. PipeDream nabízí službu RequestBin, což je krátkodobé uložiště pro HTTP requesty. Aplikační server na TTN šlo nastavit, aby při příchodu nového uplinku vytvořil HTTP Post request a přeposlal uplink na PipeDream. Odtud bylo možné, aby backend převzal uplink pomocí PipeDream API.

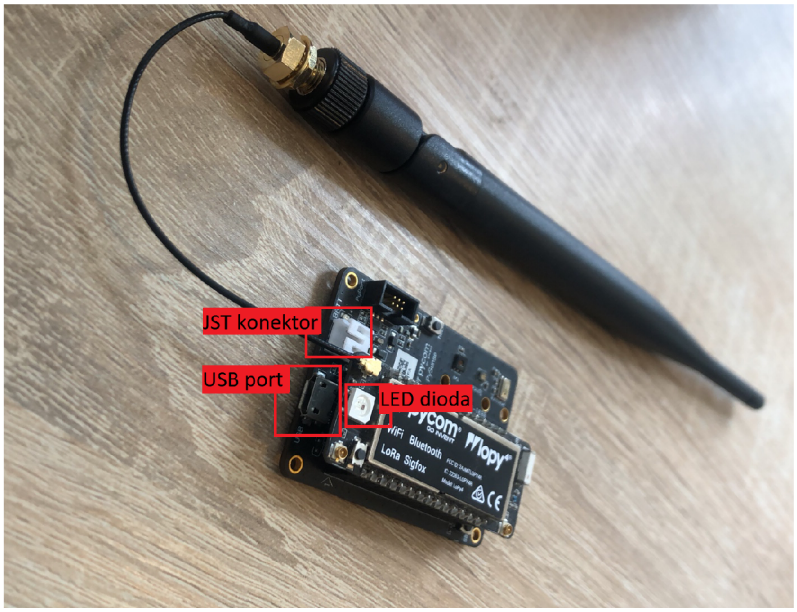
Frontend zobrazí získané data ve formě grafu a informačních bloků. Dále uživatel bude mít přístupný formulář, který umožňuje komunikaci s LoRa end node a databází. Bude moci změnit barvu LED diody a hodnoty SF. Pokud bude chtít uživatel archivované data, zadává časový interval ve kterém chce získat data a po zaslání žádosti se stáhne CSV soubor s požadovanými daty.

3.2 LoPy4 a PySense

3.2.1 Zprovoznění

Hardware zapojení

LoPy4 a PySense k sobě zapojíme podle obrázku 3.2, kdy LED dioda LoPy4 je umístěná nad PySense USB port. Desku můžeme napájat pomocí LiPo baterie s JST PHR-2 konektorem nebo pomocí mikro USB. USB port se používá i na programování desky.



Obrázek 3.2: Zapojení hardwaru

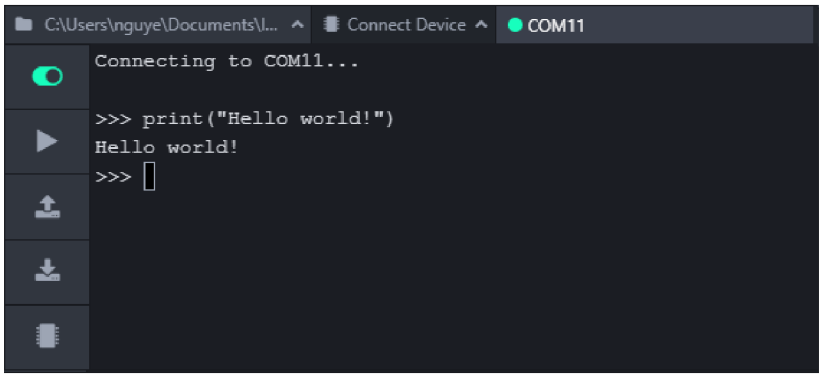
Příprava vývojového prostředí

Pro programování LoPy4 je potřeba Pymakr, což je plugin pro editory kódu, jako Atom a VS code, který umožňuje komunikaci s Pycom vývojovou deskou.

V této práci se bude používat Pymakr s editorem kódu Atom. Pomocí vyhledávače balíčku v Atomu nainstalujeme nejnovější verzi Pymakr.

Po instalaci pymakru přibyde do Atomu konzole REPL (obr. 3.3). Pymakr nám umožní

připojit Pycom zařízení s Atomem, spustit kód v mikrokontroleru, nahrát kód do zařízení nebo ho z něj stáhnout. Dále máme možnost získat informace o připojené desce, jako typ vývojové desky, jeho verzi firmwaru, využitou paměť a více. Nakonec můžeme do konzole psát kód v MicroPythonu, který se ihned spustí.



Obrázek 3.3: Pymakr a REPL

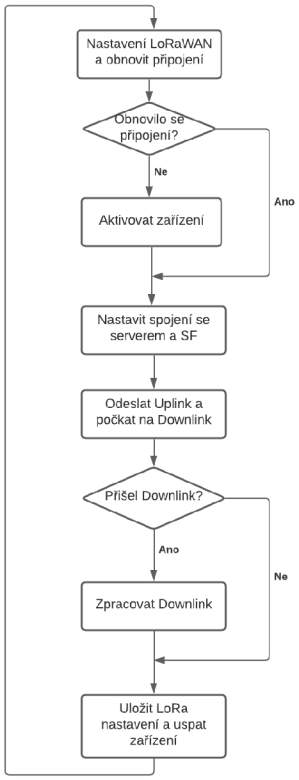
3.2.2 Návrh a implementace

Struktura projektu

Projekt, který vložíme do LoPy4 musí být strukturovaný tak, že soubory *boot.py* a *main.py* jsou umístěny ve složce projektu. Soubor *boot.py* je skript, který se spustí jako první, nejčastěji se do něj vkládá kód, který inicializuje nastavení. Skript *main.py* se spouští buďto po *boot.py* nebo pokud *boot.py* neexistuje, tak jako první. Do *main.py* se píše náš projekt. Adresář *lib* je adresář pro všechny knihovny, které importujeme do našeho projektu.

```
|--Project folder
    |--lib
        |--library1.py
        |--library2.py
    |--boot.py
    |--main.py
```

Vývojový diagram



Obrázek 3.4: Vývojový diagram projektu

Použité knihovny

```
1  import array
2  import crypto
3  import pycom
4  import struct
5  import time
6  import ubinascii
7  # LoRaWAN libraries
8  import socket
9  from network import LoRa
10 # Sensor libraries
11 from SI7006A20 import SI7006A20
12 from MPL3115A2 import MPL3115A2, PRESSURE
13 from pysense import Pysense
```

Implementace

Na obr. 3.4 vidíme vývojový diagram projektu. Ještě před, tím než se kód uvnitř LoPy4 spustí je nutné nastavit konfigurační hodnoty do napěťově nezávislé paměti (Non-volatile random-access memory, NVRAM). Při spuštění programu se zapne LED dioda a zajistí se spojení s TTS V3. Dále se odešle uplink. Po odeslání uplinku se čeká, pokud nepříjde downlink, který se popřípadě zpracuje. Nakonec se nová konfigurace, pokud nějaká je, ukládá do NVRAM paměti a zařízení se uspí pomocí deep sleep, který vypne LED diodu a vymaže všechny deklarované proměnné.

- **Konfigurace**
  LoPy4 má v sobě NVRAM paměť, která udrží informaci i po odpojení napájení nebo po uspání. Je vhodné do ní zapisovat citlivé hodnoty (AppEui a AppKey) a také konfigurační hodnoty (SF, čas spánku, barvu LED diody a LoRaWAN nastavení). Přístup do NVRAM paměti umožňuje knihovna *pycom*.

- **Spojení se síťovým serverem TTN**
  Níže máme ukázaný skript pro aktivaci zařízení.

```
1    # initialise LoRa in LORAWAN mode and tries to restore connection
2    lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868)
3    lora.nvram_restore()
4    if not lora.has_joined():
5        print("Fristly join")
6        # creates an OTAA authentication parameters
7        app_eui = ubinascii.unhexlify(pycom.nvs_get("app_eui"))
8        app_key = ubinascii.unhexlify(pycom.nvs_get("app_key"))
9        # join a network using OTAA (Over The Air Activation)
10       lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key), timeout=0)
11   else:
12       print("Connection restored")
```

  Knihovna *LoRa* se využije pro nastavení zařízení podle LoRaWAN protokolu a nastavení vysílací frekvence podle daného regionu (řádek 2). Pokusí se obnovit připojení z NVRAM paměti (řádek 3). Pokud znovupřipojení neuspělo, aktivuje se zařízení pomocí OTAA (řádek 7-10). Po úspěšné aktivaci, se nastaví hodnota SF a vytvoří se spojení mezi zařízením a serverem pomocí knihovny *socket*.

- **Odeslání uplinku**
  Knihovna *Pysense* inicializuje I^2C komunikační protokol mezi senzory a mikrokontrolerem. Ze senzorů získáme okolní teplotu, tlak a napětí na baterii.

  Kromě dat ze senzoru má v sobě poslaný uplink ještě unikátní id. Ten využije webová aplikace, aby rozpoznala zda se jedná o nový uplink.

  Unikátní id se vygeneruje pomocí knihovny *crypto*. Do mikrokontroleru vložíme prázdný textový soubor do kterého se budou zapisovat již vygenerované identifikátory. Crypto bude generovat náhodné 4 bity, které se porovnají s existujícími identifikátory v textovém souboru, dokud nevytvoří unikát.

Nakonec se vytvoří bytové pole, které se pošle jako uplink. Uplink má hodnotu 16ti bytů, kdy první 4 jsou id uplinku a zbývajících 12 bytů jsou data ze senzoru.

- **Zpracování downlinku**
  Po odeslání uplinku bude LoPy4 chvíli čekat, zda nepříjde downlink. Je třeba implementovat callback funkci, která se zavolá, pokud se něco stane na vysílači nebo na přijímači LoRa modulu.

  Tato callback funkce se spustí pokud se odešle uplink, pouze jako potrvzení toho, že modul vysílá. A také se spustí, když na přijímači modulu přijde downlink, který funkce pošle na zpracování.

  Příchozí downlinky mají hodnotu 4 bytů. První dva určují změnu SF, kdy změna SF určuje i změnu času spánku a poslední dva byty určují změnu barvu LED diody. Změny se zapíšou do NVRAM paměti.

- **Uložení nastavení a uspání zařízení**
  Parametry pro LoRaWAN připojení se uloží do NVRAM paměti a nakonec se zařízení připraví k deep sleep. Deep sleep uspí skoro všechny funkce, tudíž probuzení je jako kdybychom celé zařízení restartovali. Zapomenou se všechny deklarované proměnné, proto musíme konfigurační hodnoty zapisovat do NVRAM paměti.

3.3 Webová aplikace

Struktura projektu

```
|--Project folder
    |--templates
        |--index.html
    |--app.py
    |--config.py
    |--mongo_db_communication.py
    |--ttn_communication.py
```

Nainstalované balíčky

```
click==7.1.2
Flask==1.1.2
gunicorn==20.0.4
itsdangerous==1.1.0
Jinja2==2.11.3
MarkupSafe==1.1.1
Werkzeug==1.0.1
ttn==2.1.5
requests==2.25.1
pymongo==3.11.3
dnspython==1.16.0
```

Implementace

Výše je popsána struktura projektu, kde najdeme většinu souborů, které se starají o chod webové aplikace. *index.html* se použije pro vykreslení frontendu. V *ttn_communication.py* najdeme funkce, které se starájí o komunikaci s TTS V3. Funkce které komunikují s databázi najdeme v *mongo_db_communication.py*. V souboru *app.py* se nachází backend webové aplikace a v *config.py* jsou jeho konfigurační hodnoty.

V *app.py* se vytvoří instance třídy Flask, která se stará o inicializaci webového serveru na kterém aplikace poběží. Pomocí Flasku jsou implementované routy, které spouští různé funkce aplikace: vykreslení frontendu, stahování uplinků, vytváření downlinků a vytvoření CSV souboru, který se pošle uživateli.

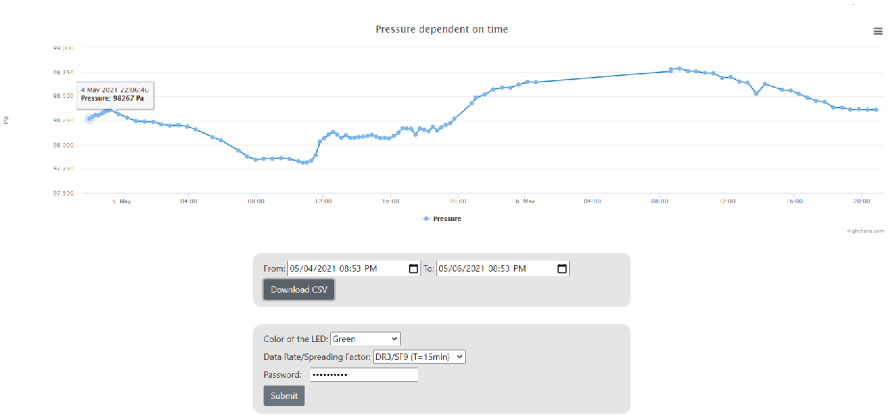
- **Vykreslení frontendu**
  Při navštívení domény webové aplikace, backend zavolá soubor *index.html*, který vykreslí potřebné komponenty. Tyto komponenty jsou nadpis, informační okna o modulu (stav baterie a nastavený SF), živé grafy a webové formuláře. Při prvním spuštění aplikace může načítání chvíli trvat, zobrazí se proto na frontendu načítací ikonka, která zmizí jakmile se všechny komponenty načtou.

  Knihovna *highcharts* se použije na vytvoření interaktivních grafů na zobrazení teploty a tlaku v závislosti na čas. Graf se živě aktualizuje. Při přejetí myší přes body grafu se zobrazí jejich přesná hodna. V menu grafu můžeme graf zvětšit, vytisknout nebo stáhnout zobrazené hodnoty v různých formátech. Frontend opakovaně vyžaduje od backendu nejnovější data, aby mohl aktualizovat graf.

  Dále se vytvoří informační okna s momentálním napětím na baterii a nastavenou hodnotou SF, pomocí *bootstrapu*.

  Poslední funkcí frontendu jsou dva webové formuláře. Jeden slouží pro stahnutí dat ve specifikovaném datovém intervalu. Další pro změnu barvy LED diody a spreading factoru modulu, pro změnu konfigurace modulu musí uživatel zadat heslo. Pokud by byl formulář vyplněn nesprávně (špatné heslo, neexistující interval), zobrazí se zprává s chybou.

  Graf *highcharts* a webové formuláře jsou zobrazeny na obr. 3.5.



Obrázek 3.5: Graf a webové formuláře

- **Stahování uplinků**
  Backend získává uplinky z TTN tím, že pošle HTTP GET Request na PipeDream server. Response serveru je pole jsonů, ve kterém najdeme námi specifikovaný počet nejnovějších uplinků. Data posláné z LoPy4 najdeme v *frm_payload*. TTN zakóduje uplink pomocí Base64 kódování, které backend dekóduje před posláním frontendu. Ukázková odpověď od PipeDream:

```
"uplink_message": {
    "frm_payload": "TV9+4ADQYEOArsBH6voEQpovmUA=",
    "rx_metadata": {
      "gateway_ids": "eui-b827ebfffec6e5d0",
      "rssi": -116,
      "snr": -11,
    },
    "settings": {
        "bandwidth": 125000,
        "spreading_factor": 10
    },
    "frequency": "867900000"
}
```

- **Vytvoření downlinku**
  Backend pošle na TTN server HTTP Post Request pro vytvoření downlinku. Do těla requestu vložíme zprávu pro LoRa zařízení, port a typ zprávy. Zpráva musí být zakódovaná pomocí Base64. Port na downlinky má hodnotu 1. V čase psaní této práce nebyla API od TTS V3 plně funkční a musí se nastavit typ zprávy jako potvrzený downlink. Tělo Post Requestu:

```
{"downlinks":[{"frm_payload":downlink, "f_port":1, "confirmed": True}]}
```

- **Získávání dat z databáze**
  Komunikaci s databází MongoDB, umožňuje knihovna *pymongo*. Data v MongoDB jsou reprezentovány v jsonu. Pro vyhledávání a získávání dat z databáze se používají tzv. Query, což je formát jak se strukturuje data request. Příklad Query, který žádá o data z databáze:

```
1  # finds data with spreading factor SF12
2  results = collection.find({'sf': 12})
3  # finds data between specified timestamps and sorts them from highest to lowest
4  results = collection.find(
5      {'time': {'$gte': from_timestamp, '$lte': to_timestamp}}).sort([('time', -1)])
```

- **Ukládání dat do databáze**
  Funkce co se stará o ukládání do databáze se podívá na id nového uplinku a porovná ho s existujícími identifikátory v databázi uplinků. Pokud se jedná o unikátní id, tak

uplink do databáze uloží. Tato funkce je zavolána pomocí vytvoření nového vlákna, které běží souběžně s hlavním programem. Funkce je pak nezávislá na webové aplikaci.

- **Nasazení na cloud**
  Pro nahrání webové aplikace na cloud pomocí Heroku, potřebujeme vytvořit Procfile soubor. Ten řekne Heroku jaký server hosting má použít a jaký skript má spusit. Pro vytvoření serveru na cloudu se použije knihovna *gunicorn*. Skript, který se má spustit je *app.py*, ve kterém je vytvořen backend webové aplikace. Procfile vypadá takto:

```
web: gunicorn app:app
```

  V projektu musí být textový soubor *requirements.txt*, což je seznam nainstalovaných knihoven. Heroku tento seznam využije pro instalaci potřebných závislostí. Obsah *requirements.txt* je výše popsán v „Nainstalované balíčky“.

  Vytvoří se Git repositář s projektem a hlavní větev se nahraje do Heroku aplikace.

4 Experimenty

4.1 Testování dosahu

LoRa je vyhlášená svým dosahem, který se měří v kilometrech. Toto měření se bude zabývat parametry ovlivňující úspěšnost připojení, jako spreading factor a umístění zařízení. Dále vliv zvolené antény, uzemňovacího efektu a vzdálenosti gatewaye na sílu přijmutého signálu, měřeného v RSSI (viz. 2.1.1).

Při posílaní uplinku na TTN je možné zjistit, které gatewaye zprávu slyší. Níže vidíme zjednodušenou ukázku zprávy na serveru TTN, když přijde uplink. Můžeme vidět pole gatewayí, které zařízení slyší a s jakým RSSI zprává na gateway došla. Vidíme také, id gatewaye, když na server TTN pošleme HTTP Get Request s tímto id, přijde response, kde je uložena i lokace gatewaye. Tudíž jsme schopni změřit vzdálenost od gatewaye, které naše zařízení slyší, sílu přijatehé signálu a při opakovaném vysílání i úspěšnost připojení s daným gateway.

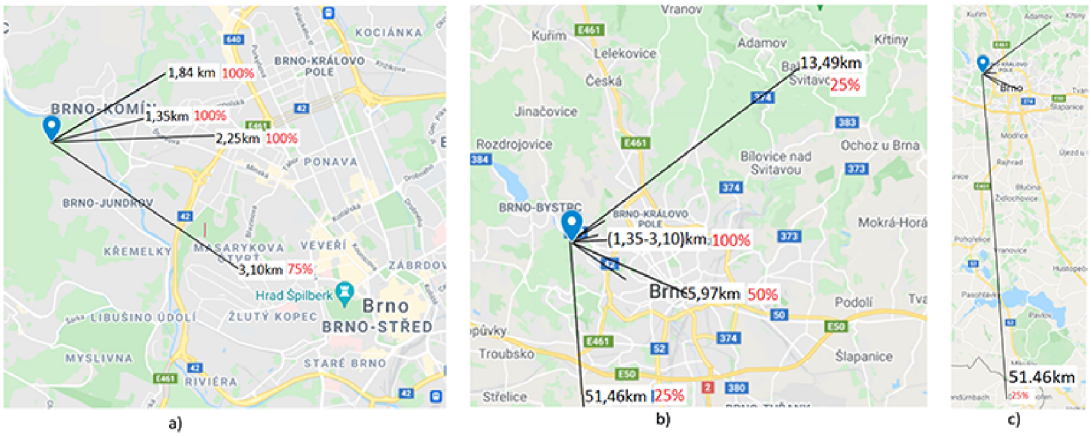
```
{
    "payload_raw": "UvHrRQBF",
    "gateways": [
    {
      "gtw_id": "eui-313532352a004e00",
      "rssi": -115,
    },
    {
      "gtw_id": "eui-58a0cbfffe802a45",
      "rssi": -47,
    }
    ]
}
```

Závislost dosahu a úspěšnost připojení na SF

- **Postup měření**
  Implementuje se skript do LoPy4, který postupně bude zvyšovat hodnotu SF a při každé hodnotě spreading factoru pošle čtyři uplinky. Tyto uplinky v sobě mají uloženou informaci o lokalitě ze které vysíláme (např.: *00 01* - Brněnská přehrada).

  Dále se implementuje Python skript, který tyto uplinky ze serveru TTN bude sbírat podobným způsobem jako v praktické části (viz. 3.3). Z uplinku získá potřebné informace: hodnotu SF, umístění zařízení, na které gatewaye signál přišel a s jakým RSSI přišel.

- **Získané závislosti**



Obrázek 4.1: a) SF7 b,c) SF12

- **Vyhodnocení měření**
  Zvětšením spreading factoru se zvětšuje čas vysílání, toto zvyšuje šanci dekódování signálu gatewayem a odolnost signálu vůči ztrátám cestou. Z obr. 4.1 můžeme vidět, že s SF7 můžeme vysílat na gatewaye vzdálené okolo 3km, s SF12 je zde šance dosáhnout až na gatewaye 50km daleko.

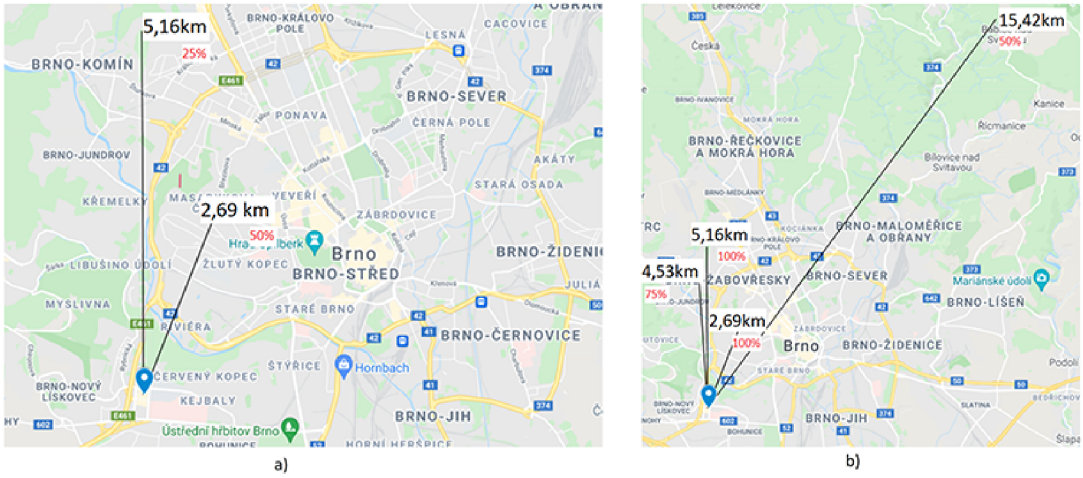
Závislost dosahu a úspěšnost připojení na umístění

- **Postup měření**
  LoPy4 bude vysílat se stejným spreading factorem na různých místech.

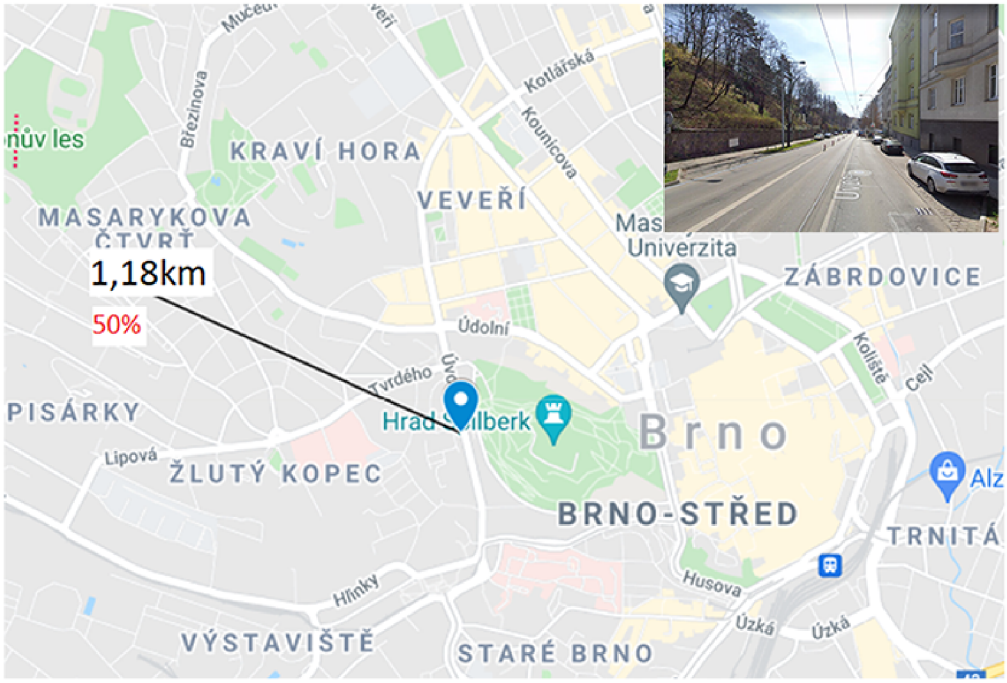
  Nejprve se bude porovnávat dosah a úspěšnost připojení při vysílání ve vnitřních prostorech a ve venkovních prostorech (uvnitř OC Campusu a před vchodem do OC Campusu).

  Nakonec se bude zkoumat dosah a úspěšnost připojení v závislosti na vyvýšení zařízení (ulice Úvoz blízko hradu Špilberk a na hradě Špilberk).

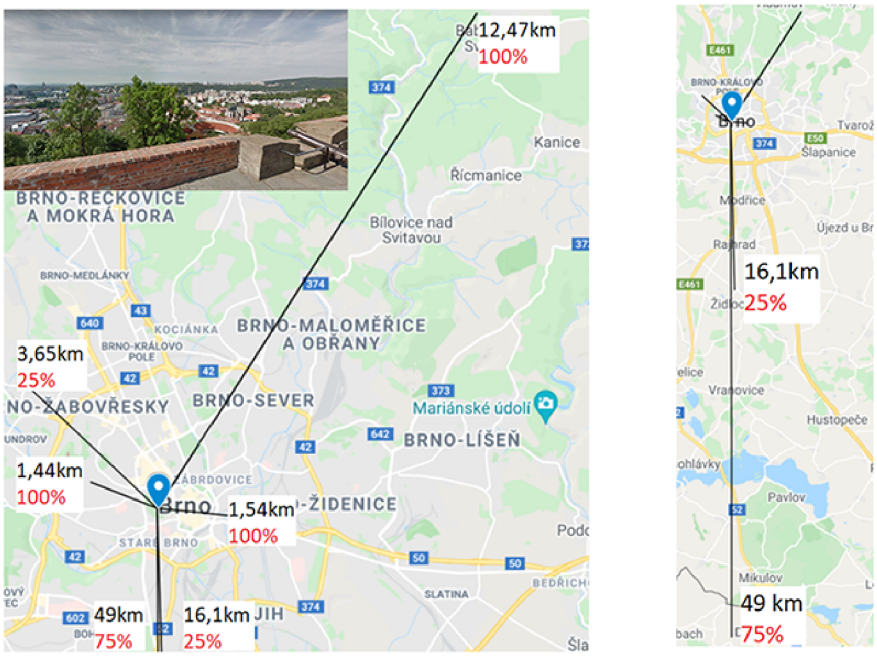
- **Získané závislosti**



Obrázek 4.2: a) uvnitř b) venku



Obrázek 4.3: Nižší nadmořská výška



Obrázek 4.4: Vyšší nadmořská výška

- **Vyhodnocení měření**
  Snížením počtu překážek drasticky zvýšíme dosah a spolehlivost signálu.

  Na obr. 4.2 vidíme, že při vysílání uvnitř budovy dosáhne signál s poloviční úspěšnosti na dva gatewaye, zatím co signál umístěn ve stejné lokalitě mimo budovu dosáhne na tyto gatewaye spolehlivě a má šanci poslat uplink na gateway 15km daleko.

  Na obr. 4.3 a 4.4, můžeme vidět, že zařízení v nižší nadmořské výšce, obklopené budovami a kopci má velkou nespolehlivost a malý dosah. Vyvýšením zařízení a zbavením se překážek opět drasticky zvětšíme dosah, kdy end node se 75% úspěšností posílá uplinky na gateway vzdálený 49 kilometrů daleko.

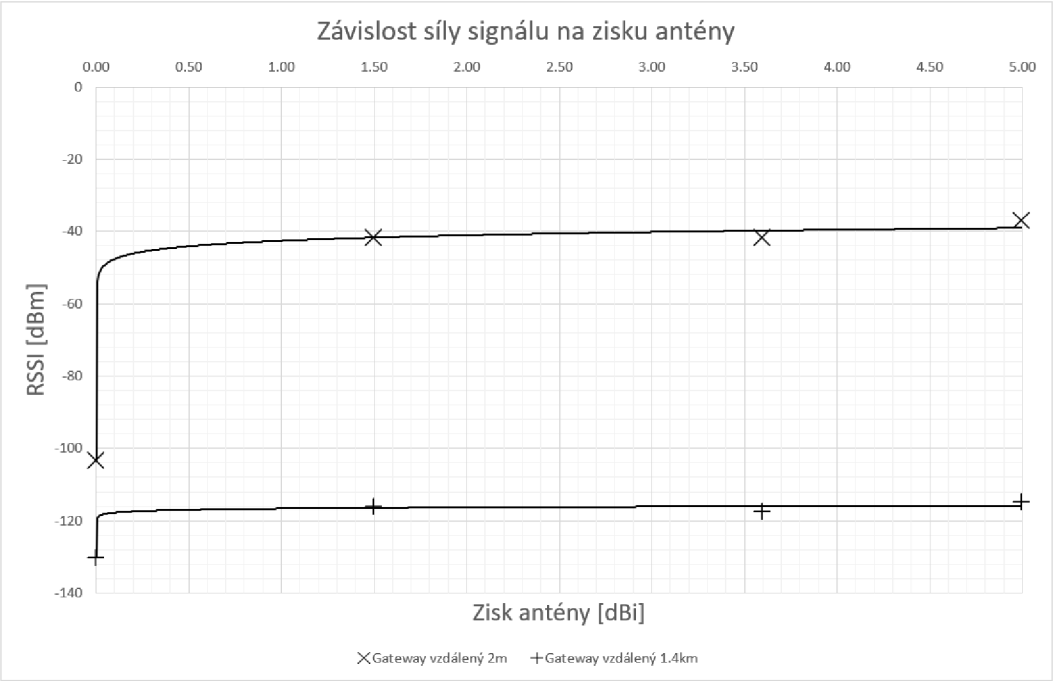
Závislost síly signálu na anténě

- **Postup měření**
  Měření bude probíhat se třemi antény s různým posílením signálu a bez antény. Z LoPy4 budeme posílat čtyři uplinky, pokaždé s jinou anténou. Budeme sledovat sílu signálu a úspěšnost připojení v závislosti na zvolené anténě.

- **Získané závislosti**

	Gateway distance			
	d = 2m		**d = 1.4 km**	
Antenna Gain [dBi]	**Success rate**	**avg(RSSI) [dBm]**	**Success rate**	**avg(RSSI) [dBm]**
0.00	100%	-103.2	0%	-130
1.50	100%	-41.8	20%	-116
3.60	100%	-42	80%	-117.5
5.00	100%	-36.8	100%	-114.8

Tabulka 4.1: Naměřené hodnoty při různých anténách

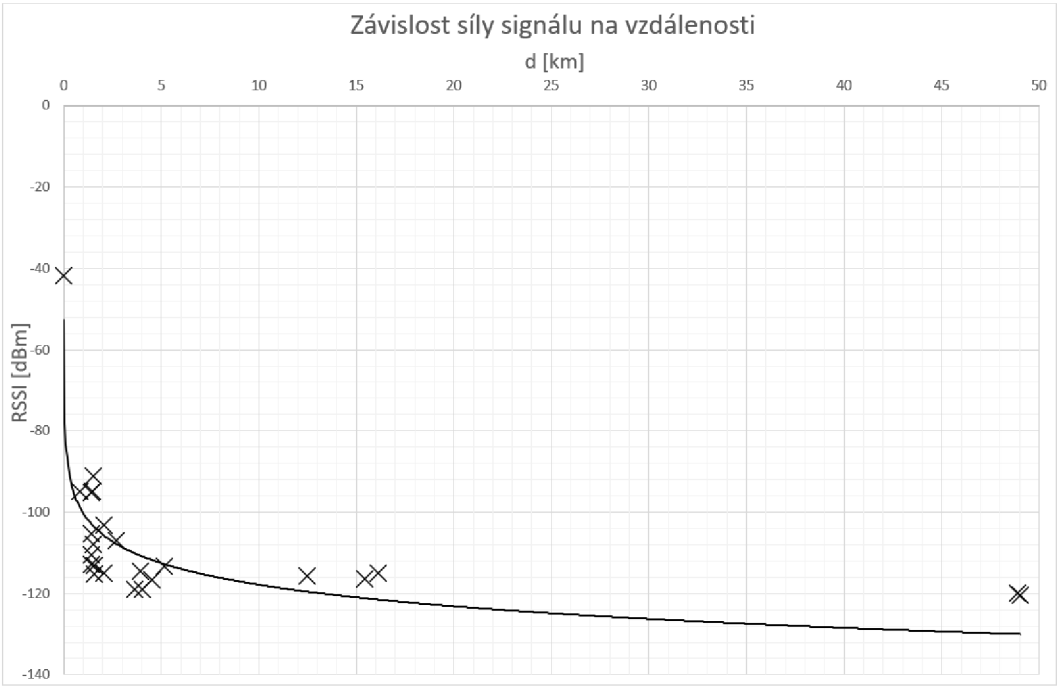


Obrázek 4.5: Závislost síly signálu na zisku antény

- **Vyhodnocení měření**
  Z tabulky 4.1 a obr. 4.5 můžeme vidět, že vybraná anténa nemá velký vliv na sílu signálu, ale vysílání bez antény je téměř nemožné, pokud se gatewaye nenachází pár metrů od end node.

Závislost síly signálu na vzdálenosti gatewaye

- **Získané závislosti**



Obrázek 4.6: Závislost vzdálenosti gatewaye na sílu signálu

- **Vyhodnocení měření**
  Z obr. 4.6 můžeme vidět, že síla signálu rychle poklesne při zvětšení vzdálenosti, ale rozdíl signálu mezi gatewayem vzdáleným 10 kilometrů a 50 kilometrů není příliš viditelný.

Závislost síly signálu na uzemnění vysílače

- **Získané závislosti**

	i	1	2	3	4	avg
RSSI [dBm]	**In the air**	-84	-78	-81	-79	-80.5
	Grounded	-77	-78	-77	-73	-76.25

Tabulka 4.2: Naměřené hodnoty při uzemnění antény a při volném prostoru

- **Vyhodnocení měření**
  Z tabulky 4.2, můžeme vidět, že uzemnění LoRa zařízení může zvýšit zisk signálu o pár jednotek dBm.

Vyhodnocení experimentu

Z měření lze usoudit, že na dosah LoRa zařízení má největší vliv umístění end node, kdy venkovní vyvýšený volný prostor mají nejlepší vliv na vysílací signál.

Dále zvětšením spreading factoru zvětšujeme dosah a houževnatost signálu, ale musíme brát na vědomí, že tím omezujeme jak často můžeme posílat data.

Větší vliv na dosah signálu mají spíše samotné gatewaye než konfigurace end nodu. LoRa moduly mají limitované kolik dBi mohou přidat signálu (v Evropě je maximum 15dBi), gatewaye mají tyto limity podstatně menší a tudíž je dosah více závislý na citlivosti LoRaWAN gatewayů.

4.2 Testování spotřeby

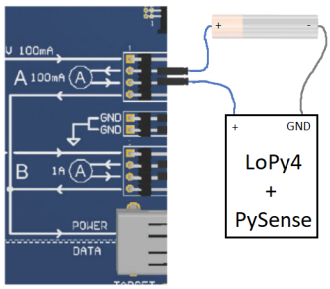
LoRa zařízení, pracující v třídě provozu A, posílá data co minuty nebo někdy co hodiny, tudíž většinu času je zařízení v režimu spánku. LoPy4 lze uspat třemi způsoby.

- **Idle mode** je mód, kdy zařízení čeká specifikovaný čas, než zpracuje další příkaz. V tomto režimu jsou všechny funkce mikrokontroleru zapnuté a není nutné obnovovat připojení po probuzení.
- **Ligh sleep** uspí všechny rádiové moduly, CPU a RAM mikrokontroleru stále běží. Po probuzení je nutné obnovit LoRaWAN připojení.
- **Deep sleep** je režim, kdy se vypnou téměř všechny funkce, vypne se CPU, RAM a rádiové moduly, jediné co běží dál jsou CPU hodiny. Po probuzení se zařízení chová, jak kdyby bylo restartováno, tudíž je nutné obnovit LoRaWAN připojení.

Toto měření se bude zabývat spotřebou LoPy4 a PySense při různých režimech spánku, při posílání uplinku a při bootování zařízení.

Popis měření

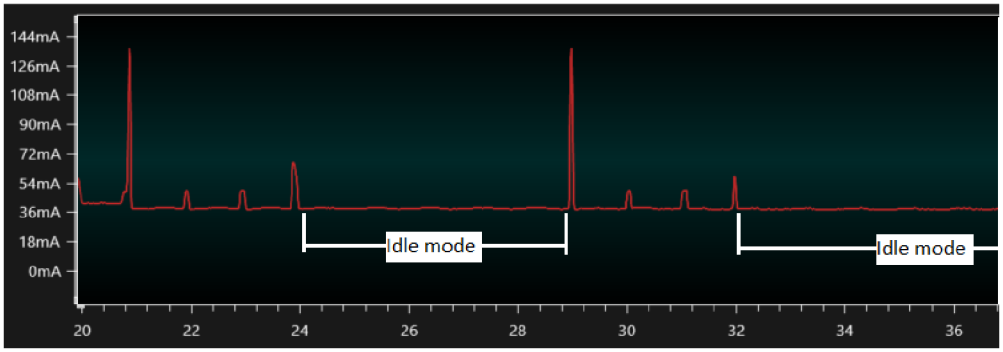
Pro měření spotřeby byl využit Power Debugger od Microchipu, který v sobě má zabudovaný nízkoproudový snímač proudu. Desku Power Debugger, lze zapojit do počítače a pomocí Data Visualizeru můžeme pozorovat průběh proudu deskou. Na obr. 4.7 můžeme vidět zapojení LoPy4 s Power Debuggerem.



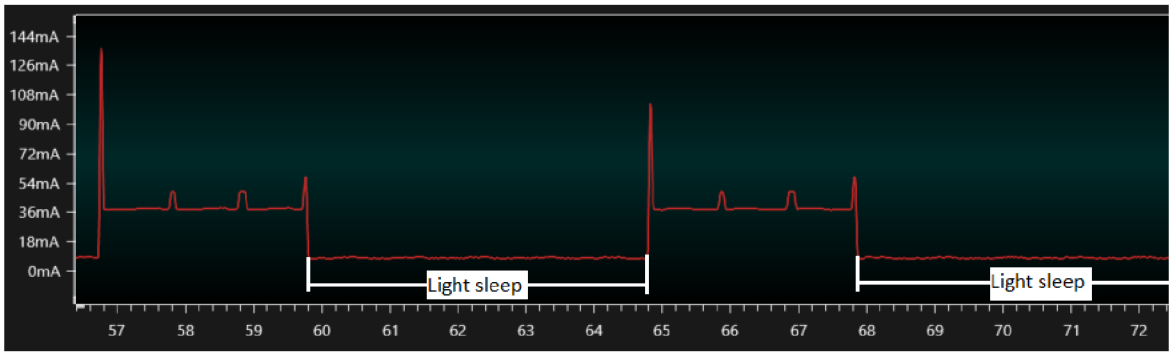
Obrázek 4.7: Zapojení

Výsledek měření

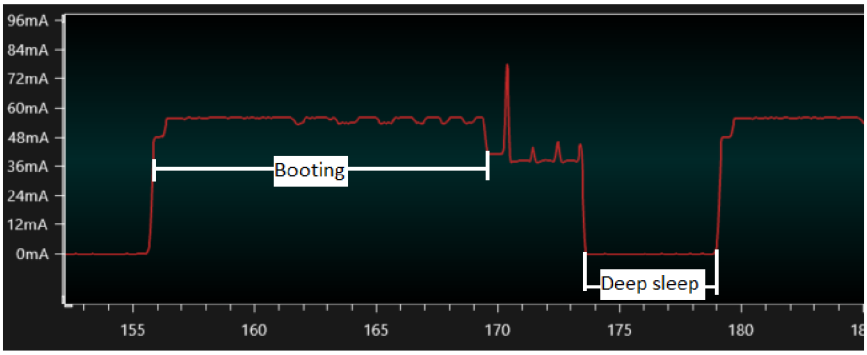
Na obr. 4.8 až obr. 4.11 najdeme výsledky měření, kde na ose x je hodnota času v jednotkách sekund.



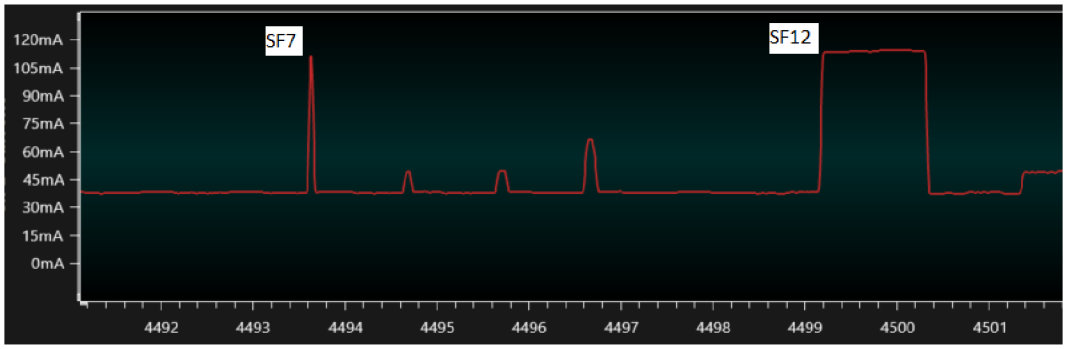
Obrázek 4.8: Spotřeba idle modu



Obrázek 4.9: Spotřeba light sleep modu



Obrázek 4.10: Spotřeba deep sleep modu



Obrázek 4.11: Spotřeba SF7 a SF12

	Idle	Light sleep	Deep sleep	Sending uplink	Booting
Current	37.9 mA	7.77mA	11.1 µA	102 mA	56 mA

Tabulka 4.3: Naměřené hodnoty

Výpočet minimální doby deep sleep

$$t_{boot} = 17.6s$$

$$I_{light} \cdot t_{sleep} = I_{deep} \cdot t_{sleep} + I_{boot} \cdot t_{boot}$$

$$t_{sleep} = \frac{I_{boot} \cdot t_{boot}}{I_{light} - I_{deep}} = \frac{56 \cdot 10^{-3} \cdot 17.6}{7.77 \cdot 10^{-3} - 11.1 \cdot 10^{-6}} = 127.03s$$

Vyhodnocení měření

Z tab. 4.3 můžeme vidět spotřebu při idle režimu, light sleep, deep sleep, posílání uplinku a zapínání zařízení.

V datasheetu LoPy4[10] je uvedeno, že spotřeba při deep sleep je 1µA a v datasheetu PySense[11] je uvedena spotřeba při deep sleep 9µA. Naměřili jsme 11,1µA, což je velice blízko uvedeným hodnotám.

Z obr. 4.11 můžeme vidět, že zvolený spreading factor neovlivňuje velikost odebíraného proudu, ale zvětšením SF zvětšíme čas odebírání proudu.

Průměrná hodnota odebraného proudu při bootování je 56mA a čas bootování vyšel 17,6s. Tyto hodnoty patří softwaru, který byl využit v praktické části (viz. 3.2). Zapínání zařízení má velkou spotřebu, pokud bychom uspávali zařízení na krátkou dobu, bylo by lepší použít režim light sleep. Minimální doba spánku, aby byla varianta deep sleep efektivnější je 127s, což většina projektů využívající LoRa technologii bude mít.

5 Závěr

Cílem této bakalářcé práce bylo vybrat potřebný hardware pro vytvoření měřící stanice ze které se posílají data na webovou aplikace skrze LoRa síť.

Zvolený hardware pro vytvoření měřící stanice byla vývojová deska LoPy4 s LoRa modulem SX1276, kdy k desce byl připojen štít Pysense, který byl vybaven senzory snímající okolní teplotu a tlak. Využila se anténa nabídnutá k desce LoPy4. LoPy4 je vybaven mikrokontrolerem ESP32, který se programoval pomocí jazyku MicroPython.

Pro přenos dat pomocí LoRa technologie byl zvolen protokol LoRaWAN a síťový server The Things Stack V3.

Backend webové aplikace, který data ze síťového serveru The Things Stack V3 získával a ukládal, byl vytvořený pomocí Flask framework, který se programoval v jazyce Python. Frontend, napsaný v HTML, CSS a Javascriptu, zobrazoval data skrze knihovnu Highcharts. Databáze pro ukládání dat byla MongoDB.

Přenos dat z měřící stanice na webovou aplikaci proběhl úspěšně, kdy LoRa zařízení, napájené z baterie operovalo spolehlivě několik dní bez jakéhokoliv dozoru.

Z testování dosahu zařízení se ukázalo, že největší vliv na spolehlivost a dosah spojení má umístění LoRa vysílače a přijímače. Minimalizováním překážek mezi LoRa end node a LoRa gateway může výrazně zvýšit spolehlivý přenos dat při velkých vzdálenostech. Chceme-li zvětšit dosah a spolehlivost LoRa signálu, pak volba LoRa gatewaye má větší vliv na výsledek než LoRa end nody. Gatewaye mohou přispět více link budgetu než end nody, kvůli méně striktním limitacím zákona.

Při měření hodnoty odebíraného proudu uspaného zařízení v režimu deep sleep byla naměřeno 11µA, což se velice blíží hodnotám uvedeným v datasheetu. LoRa end node odebírá nejvíce proudu při vysílání, kdy odebere 102mA a nastavený spreading factor určuje délku spotřeby. Dále při probuzení z deep sleepu zařízení odebírá 56mA, kdy čas spotřeby je definován programem v modulu.

V této bakalářské práci byly splněny všechny body zadání. Za účelem rozvoje této práce se rozšířila komunikace webové aplikace s LoRa zařízením. Implementovala se oboustranná komunikace, kdy bylo možné posílat příkazy z webové aplikace do LoRa zařízení. Příkazy měnily rozšiřující faktor vysílaných zpráv, interval mezi vysíláním a barvu LED světla. Uživatel měl také přístup k archivovaným datům pomocí webového formuláře, kdy bylo potřebné zadat časový interval vyžadovaných dat.

Na bakalářskou práci je možné navázat změnou provozu LoPy4 z třídy A do třídy C, kdy by zařízení mohlo sloužit i jako akční člen nebo implementovat sledování lokality zařízeni. Další možností je lepší optimalizace programu uloženého v desce LoPy4, momentálně je čas bootování relativně dlouhý, což zvyšuje spotřebu desky.

Seznam příloh

1. LoPy4 skript
2. Program webové aplikace
3. Data naměřené při testování dosahu
4. Skripty použité při testování dosahu
5. Data naměřené při testování spotřeby

Seznam použitých zkratek

ABP	Access By Personalization
API	Application Programming Interface
AppSKey	Application Session Key
BW	Bandwidth
CPU	Central Processing Unit
CR	Coding Rate
DevAddr	Device Address
DR	Data Rate
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
I2C	Inter-Integrated Circuit
JSON	JavaScript Object Notation
LPWA	Low-Power Wide-Area
LPWAN	Low-Power Wide-Area Network
NVRAM	Non-Volatile Random-Access Memory
NwkSKey	Network Session Key
OTAA	Over The Air Activation
PaaS	Platform as a Service
RAM	Random-Access Memory
REPL	Read-Eval-Print-Loop
RSSI	Received Signal Strenght Indicator
SF	Spreading Factor
TTN	The Things Network
TTS	The Things Stack

Literatura

[1] *LoRa and LoRaWAN: A Technical Overview* [online]. 2019, 26 [cit. 2021-5-14]. Dostupné z: *https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf*

[2] *AN1200.22 LoRa™ Modulation Basics* [online]. 2015, 26 [cit. 2021-5-14]. Dostupné z: *https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf*

[3] *Glossary* [online]. 2020, [cit. 2021-5-14]. Dostupné z: *https://www.thethingsindustries.com/docs/reference/glossary/*

[4] LUETH, Knud Lasse. *State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time* [online]. 2020, [cit. 2021-5-14]. Dostupné z: *https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/*

[5] LIE, Robert. *LoRaWAN: LoRa/LoRaWAN tutorial* [online]. 2018, [cit. 2021-5-14]. Dostupné z: *https://www.mobilefish.com/developer/lorawan/lorawan_quickguide_tutorial.html*

[6] YEGIN a SELLER. *LoRaWAN® L2 1.0.4 Specification (TS001-1.0.4)*. 2020.

[7] YEGIN a SELLER. *RP002-1.0.2 LoRaWAN® Regional 40 Parameters* [online]. 2020 [cit. 2021-5-14]. Dostupné z: *https://lora-alliance.org/wp-content/uploads/2020/11/RP_2-1.0.2.pdf*

[8] *Frequency Plans* [online]. [cit. 2021-5-14]. Dostupné z: *https://www.thethingsnetwork.org/docs/lorawan/frequency-plans/index.html*

[9] *Semtech SX1276: LoRa Core™ 137MHz to 1020MHz Long Range Low Power Transceiver* [online]. [cit. 2021-5-14]. Dostupné z: *https://www.semtech.com/products/wireless-rf/lora-core/sx1276*

[10] *Lopy4: Datasheet* [online]. 2020, 33 [cit. 2021-5-14]. Dostupné z: *https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_LoPy4_v2.pdf*

[11] *Pysense 2.0 X: Datasheet* [online]. 2020, 15 [cit. 2021-5-14]. Dostupné z: *https://docs.pycom.io/gitbook/assets/PySense2X_specsheet.pdf*

[12] *RN2483: Low-Power Long Range LoRa® Technology Transceiver Module* [online]. 2020, 22 [cit. 2021-5-14]. Dostupné z: *https://ww1.microchip.com/downloads/en/devicedoc/50002346c.pdf*

[13] *RFM95/96/98(W) - Low Power Long Range Transceiver Module* [online]. 2020, 123 [cit. 2021-5-14]. Dostupné z: *https://ww1.microchip.com/downloads/en/devicedoc/50002346c.pdf*

[14] *Technology Stack: LoRaWAN Network Server* [online]. [cit. 2021-5-14]. Dostupné z: *https://www.thethingsnetwork.org/tech-stack#section1*

[15] *Duty Cycle* [online]. [cit. 2021-5-14]. Dostupné z: *https://www.thethingsnetwork.org/docs/lorawan/duty-cycle/index.html#maximum-duty-cycle*

[16] Avbentem/airtime-calculator *Github* [online]. [cit. 2021-5-14]. Dostupné z: *https://avbentem.github.io/airtime-calculator/ttn/eu868/16*

[17] *Pycom: Docs* [online]. [cit. 2021-5-14]. Dostupné z: *https://docs.pycom.io/*

[18] Pycom/pycom-libraries *Github* [online]. [cit. 2021-5-14]. Dostupné z: *https://github.com/pycom/pycom-libraries*

[19] *The Things Stack: Docs* [online]. [cit. 2021-5-14]. Dostupné z: *https://www.thethingsindustries.com/docs/*

[20] *Flask: Docs* [online]. [cit. 2021-5-14]. Dostupné z: *https://flask.palletsprojects.com/en/2.0.x/*

[21] *HTTP Request Methods* [online]. [cit. 2021-5-14]. Dostupné z: *https://www.w3schools.com/tags/ref_httpmethods.asp*

[22] *Highcharts: Docs* [online]. [cit. 2021-5-14]. Dostupné z: *https://www.highcharts.com/docs/index*

[23] Soumilshah1995/Flask-Charts-Youtube-Tutorials- *Github* [online]. [cit. 2021-5-14]. Dostupné z: *https://github.com/soumilshah1995/Flask-Charts-Youtube-Tutorials-*

[24] *HTML Forms* [online]. [cit. 2021-5-14]. Dostupné z: *https://www.w3schools.com/html/html_forms.asp*

[25] *MongoDB: Docs* [online]. [cit. 2021-5-14]. Dostupné z: *https://docs.mongodb.com/*