UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2023 Lukáš Laštůvka

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

IoT rozhraní

Bakalářská práce

2023 Lukáš Laštůvka

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lukáš Laštůvka**
Osobní číslo:	**I20025**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Komunikační a mikroprocesorová technika**
Téma práce:	**IoT rozhraní**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování

Cílem práce je vytvoření malé ukázkové sítě využívající některou z typických bezdrátových technologií (LoRa, Zigbee, Sigfox...) s implementací gateway pro předání dat mezi plnohodnotnou TCP/IP sítí a IoT sítí. Teoretická část práce popíše problematiku IoT sítí a jejich specifika (požadavky na spotřebu, dosah..) technologie, protokoly. Praktická část pak provede vytvoření ukázkové sítě (senzorová síť) s gateway a uživatelským rozhraním. Praktická část bude koncipována formou tutoriálu pro tvorbu podobných aplikací se zdůvodněním jednotlivých voleb.

Rozsah pracovní zprávy: **30-50**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1] VÁŇA, V. Mikrokontroléry ATMEL AVR: popis procesoru a instrukční soubor. Praha: BEN technická literatura, 2003.336 s. ISBN 978-80-7300-083-0.
[2] VÁŇA, V. Mikrokontroléry ATMEL AVR: programování v jazyce C. Praha: BEN technická literatura, 2003. 216 s. ISBN 978-80-7300-102-0.
[3] VLACH, J. Řízení a vizualizace technologických procesů. Praha: BEN technická literatura, 2002. 160 s. ISBN 978-80-86056-66-X.
[4] BRTNÍK, B. Základní elektronické obvody. Praha:BEN technická literatura, 2011. 156s. ISBN 978--80-7300-408-8
[5] RIPKA, P.; TIPEK, A. Master Book of Sensors. Praha : BEN, 2003. ISBN 0-12-752184

Vedoucí bakalářské práce: **Ing. Pavel Rozsíval**
Katedra elektrotechniky

Datum zadání bakalářské práce: **15. listopadu 2022**
Termín odevzdání bakalářské práce: **12. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Pidanič, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 31. ledna 2023

Prohlašuji:

Práci s názvem IoT rozhraní jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne …… Lukáš Laštůvka

PODĚKOVÁNÍ

Tímto bych chtěl velice poděkovat vedoucímu bakalářské práce Ing. Pavlu Rozsívalovi za jeho čas, rady a trpělivost při zpracování této práce. Děkuji spolužákovi Lukášovi Poklopovi za propůjčení naměřených dat z jeho bakalářské práce, ochotu a rady. Chtěl bych vyjádřit svou vděčnost vůči rodině a přátelům, kteří mě v průběhu psaní práce podporovali a povzbuzovali.

ANOTACE

Cílem práce je vytvořit ukázkovou IoT síť s využitím bezdrátové technologie LoRa a propojit tuto síť s TCP/IP sítí pomocí implementace IoT brány. Praktická část se zaměřuje na vytvoření senzorové sítě pomocí LoRa, MQTT a nástroje Node-RED, včetně implementace uživatelského rozhraní. V této části bude popsán proces vývoje a implementace sítě v detailu a zdůvodněna rozhodnutí při výběru použitých technologií.

KLÍČOVÁ SLOVA

IoT brána, IoT zařízení, LoRa, MQTT broker, Node-RED, ESP-IDF, mikrokontroler, ESP32

TITLE

IoT Gateway

ANNOTATION

The aim of the thesis is to create a demonstration IoT network using the LoRa wireless technology and to connect this network to TCP/IP using an IoT gateway implementation. The practical part focuses on creating a sensor network using LoRa, MQTT, and the Node-RED tool, including the implementation of a user interface. This part describes the development and implementation process of the network in detail and justifies the decisions made in selecting the technologies used.

KEYWORDS

IoT gateway, IoT node, LoRa, MQTT broker, Node-RED, ESP-IDF, microcontroller, ESP32

OBSAH

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

SEZNAM ZKRATEK A ZNAČEK

IoT	Internet of Things
AI	Artificial Intelligence
SQL	Structured Query Language
REST	Representational State Transfer
API	Application Programming Interface
LoRa	Long Range
BLE	Bluetooth Low Energy
NFC	Near Field Communication
RFID	Radio Frequency Identification
ISM	Industrial, Scientific, Medical
CSS	Chirp Spread Spectrum
SF	Spreading Factor
LoRaWAN	Long Range Wide Area Network
LPWAN	Low-power wide area network
M2M	Machine to Machine Communication
Wi-Fi	Wireless Fidelity
IEEE	Institute of Electrical and Electronics Engineers
MQTT	Message Queuing Telemetry Transport
SaaS	Software as a Service
HTTP	Hypertext Transfer Protocol
TLS	Transport Layer Security
TCP	Transmission Control Protocol
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
UART	Universal asynchronous receiver-transmitter
USB	Universal Serial Bus
DoS	Denial of Service
SSL	Secure Sockets Layer
ESP-IDF	Espressif IoT Development Framework
VSCode	Visual Studio Code

GPIO	General-purpose input/output
DIO	Digital input/output
CRC	Cyclic redundancy check
FIFO	First in First out
CAD	Channel Activity Detection
RSSI	Received Signal Strength Indicator
NGP	Node-Gate Protocol
QoS	Quality of Service
FNS	Functional State
NVS	Non-volatile Storage
SNTP	Simple Network Time Protocol
PCB	Printed Circuit Board

ÚVOD

Cílem práce je vytvoření malé ukázkové sítě využívající některou z typických bezdrátových technologií (LoRa, ZigBee, Sigfox, …) s implementací IoT brány pro předání dat mezi plnohodnotnou TCP/IP sítí a IoT sítí.

V úvodní kapitole je vysvětlena problematika IoT, architektura systému a jeho rozdělení. Dále se práce věnuje tématu komunikace v rámci IoT a problematice bezpečnosti v této oblasti. Výhody a nevýhody využívání IoT.

Druhá kapitola se zabývá tvorbou IoT sítě. V této části je popsána představa o vytvoření sítě a výběr komponentů. Jedním z hlavních témat této kapitoly je technologie LoRa a použití modulu ESP32. Tato kapitola obsahuje také popis softwaru, přípravu IDE prostředí a tvorbu programu. Konkrétně jsou popsány různé moduly, jako například LoRa modul, Wi-Fi manažer, MQTT, protokol NGP a GPIO rozhraní. Tato kapitola zahrnuje také ukázky programů pro IoT bránu a IoT zařízení. Posledním tématem této kapitoly je zpracování dat, včetně rozboru dat a použití Node-RED pro vizualizaci.

Třetí kapitola se věnuje konstrukci IoT brány. Popisuje tvorbu PCB, včetně návrhu a osazení PCB. Dále je zahrnut popis tvorby ochranného krytu pro IoT bránu, včetně zakreslení 3D modelu krytu a výtisku 3D modelu pomocí 3D tiskárny.

1 PROBLEMATIKA

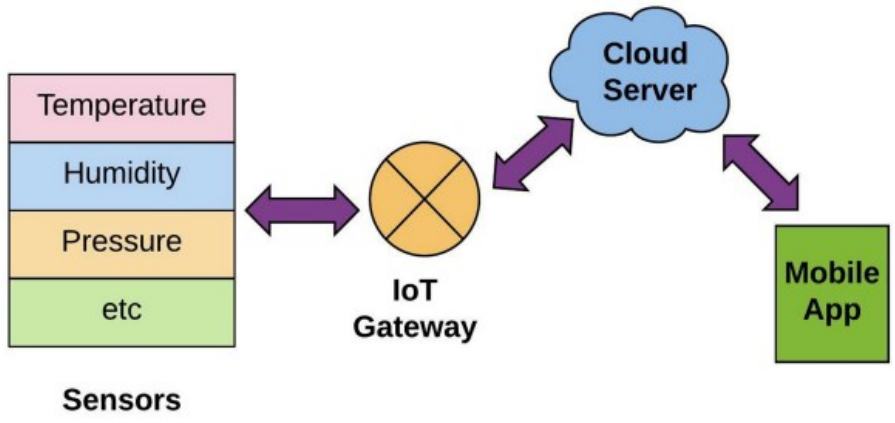
Internet věcí (IoT) vytváří hlavní globální infrastrukturu mezi lidmi a zařízeními. Nabízí důležité zázemí pro vývoj komplexních systémů a napomáhá k příchodu stále více oblíbené umělé inteligence (AI). Dnes se s takovouto infrastrukturou setkáváme téměř všude, často o ní, na první pohled, ani nevíme, ale hraje důležitou roli např. ve zdravotnictví od služeb až po výrobu nebo v zemědělství či moderním průmyslu. [1] Můžeme se s ním setkat i ve smart mobilech či počítačích nebo tabletech. Zkrátka všude tam, kde je zapotřebí získávat různá data a následně je monitorovat a pracovat s nimi. Tato kapitola shrne celkový přehled vlastností IoT, architekturu IoT sítě, implementaci a výhody a nevýhody používání IoT.

1.1 Co je IoT

IoT je v oblasti dnešních technologických trendů velice významné. Vychází z dřívějších metod budování infrastruktur pomocí senzorické sítě, vestavěných systémů a informačních technologií. Velice často se objevuje jako výkonná infrastruktura v mnoha oblastech. Díky jeho provázanosti se použití IoT sítě, jeví jako robustní, snadné a kompletní řešení. Schopnost rozsáhlé konfigurace a volnost pro vytvoření sítě je jedním z klíčových parametrů. [1]

1.2 Architektura IoT

Architekturou nebo také IoT sítí můžeme nazvat uspořádání jednotlivých dílčích částí do jednoho celku. Patří sem koncová zařízení (node), tzn. senzory, tlačítka, světla, výstražná světla, aj., dále pak IoT brána (gateway), databáze, často označována jako vzdálené uložiště dat, a zařízení pro monitoring a správu jednotlivých nodů, typicky mobilní, nebo počítačová aplikace. [1]



Obrázek 1 IoT architektura [1]

1.2.1 IoT zařízení (nodes)

Hlavním úkolem je v rámci celé architektury odesílání naměřených dat, pomocí předem stanoveného protokolu, IoT bráně. Samotná zařízení se jeví jako komplexní a ucelený systém, avšak nijak neovlivňující samotnou IoT síť.

1.2.2 IoT brána (gateway)

Je centralizované rozhraní mezi jednotlivými nody a hlavní databází dat. Slouží pro komunikaci pomocí různých protokolů a zastřešuje kompletní komunikační most včetně základní správy dat a autentizaci. Ve větších sítích se může použitím více IoT bran dosáhnout lepší stability celkové infrastruktury, kdy každá z nich zpracuje jednotlivý úsek dat a následně odešle do hlavní databáze dat.

1.2.3 Hlavní databáze dat

Pro uchování dat v organizované podobě využíváme databázových řešení. Často se může jednat o vzdálenou (cloudovou) databázi dat. Pro přístup k datům se často používá standardizovaného komunikačního protokolu SQL, případně takovou databází může být webové REST API, které může obsahovat algoritmus, jak s daty pracovat. Hlavním úkolem databáze je uchovávání dat a následný přístup k datům. Databáze patří k „mozku“ celé IoT sítě, bez databáze by nebylo možné data monitorovat.

1.2.4 Aplikace

Ve chvíli, kdy chceme s daty pracovat, je zapotřebí, aby byla data co možná nejlépe uspořádána pro snadné porozumění člověkem. K tomu slouží mobilní nebo počítačová aplikace, která propojuje databázi, s velkým množstvím dat, a zprostředkovává je v rámci různých tabulek, textových sloupců či grafů do, na první pohled, čitelné podoby. Je velice důležité, aby byla data na první pohled snadno čitelná. S tím souvisí snadné zjištění nebezpečných stavů nebo vyladění nejrůznějších procesů, na kterých IoT síť funguje.

1.3 Rozdělení systému

Pro správné fungování IoT sítí je zapotřebí komunikace mezi jednotlivými částmi celé infrastruktury. K tomu nám slouží různé komunikační systémy. V závislosti na použití můžeme jednotlivé systémy rozdělit do několika skupin. [2]

1.3.1 Spotřeba energie

Často se jedná o zařízení, které funguje na baterie s krátkou životností či kapacitou. Mezi taková zařízení můžeme označit např. nositelnou elektroniku.

1.3.2 Přenosová rychlost

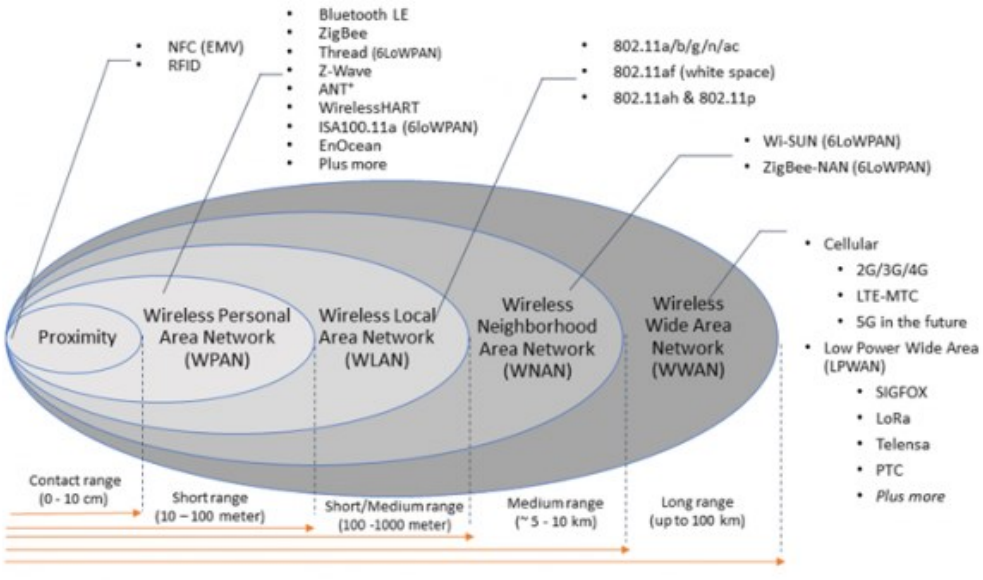
Někdy zařízení slouží pouze pro odesílání jedné informace v dlouhém intervalu, např. teplotní senzor, vlhkoměr. Taková zařízení nepotřebují vysokou přenosovou rychlost. Naopak zařízení, která odesílají velké množství dat v krátkém časovém úseku, potřebují dostatečnou kapacitu přenosu.

1.3.3 Vzdálenost

Dalším rozhodujícím faktorem komunikace je dosah zařízení a brány. Kdy data mohou být předávána na vzdálenost jen několika málo metrů nebo kilometrů.

1.3.4 Frekvence

Posledním faktorem samotné komunikace systému je frekvenční oblast. Často tuto oblast volíme podle předchozího výběru. Důležitou součástí výběru je vzdálenost a přenosová rychlost.



Obrázek 2 Rozdělení komunikačního systému podle vzdálenosti [2]

1.4 Komunikace

Pro správnou komunikaci, mezi jednotlivými částmi sítě, je zapotřebí výběru správného komunikačního systému a implementace protokolu. Na každou situaci se hodí jiný protokol, přičemž samotných protokolů je celá řada. Zaměříme se tedy na nejpoužívanější protokoly pro IoT sítě. [3]

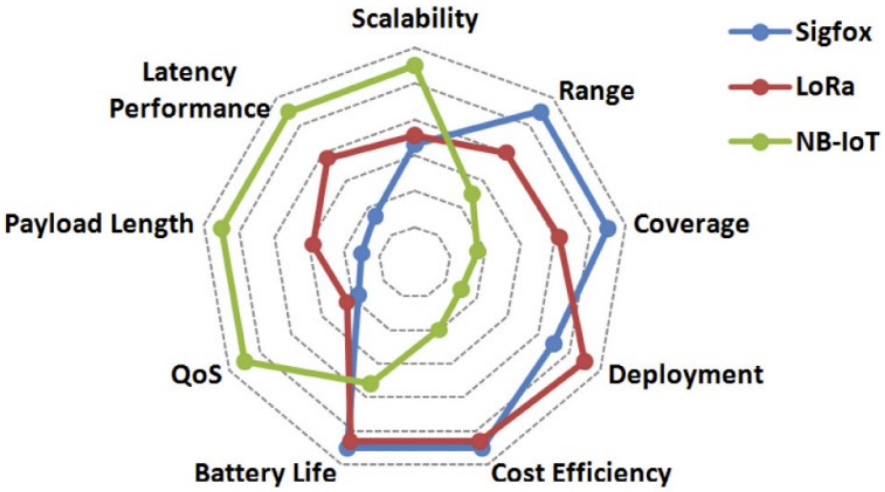
Samotnou komunikaci můžeme rozdělit, podle odesílatele a příjemce, na komunikaci mezi bránou a koncovými zařízeními nebo bránou a databází.

1.4.1 IoT brána – IoT zařízení

Protokol, který je možný využít pro komunikaci mezi IoT bránou a IoT zařízením, volíme podle kritérií komunikačního systému, tedy podle spotřeby, přenosové rychlosti, dosahu nebo frekvence.

Často se setkáváme s komunikací pomocí LoRa nebo Sigfox, kde je zajištěn důraz na životnost baterie a levné pořizovací náklady. Někdy je můžeme označit jako komunikace pro dlouhé vzdálenosti.

Mezi další komunikace lze zařadit tzv. komunikace na krátkou vzdálenost. Patří sem stále více populární ZigBee, Bluetooth/BLE nebo třeba standartní komunikační systém NFC/RFID, jež se používá pro vzdálenosti v několika málo centimetrech.



Obrázek 3 Porovnání systémů LoRa a Sigfox s NB-IoT [3]

LoRa

Je technologie fyzické vrstvy. Moduluje signály v pásmu sub-GHZ ISM pomocí patentové techniky rozprostřeného spektra. Používá nelicencovaná pásma ISM tj. 868 MHz, 915 MHz a 433MHz. Umí komunikovat obousměrně díky CSS (Chirp Spread Spectrum). Výsledný signál má proto nízkou hladinu šumu, díky tomu má vysokou odolnost proti šumu.

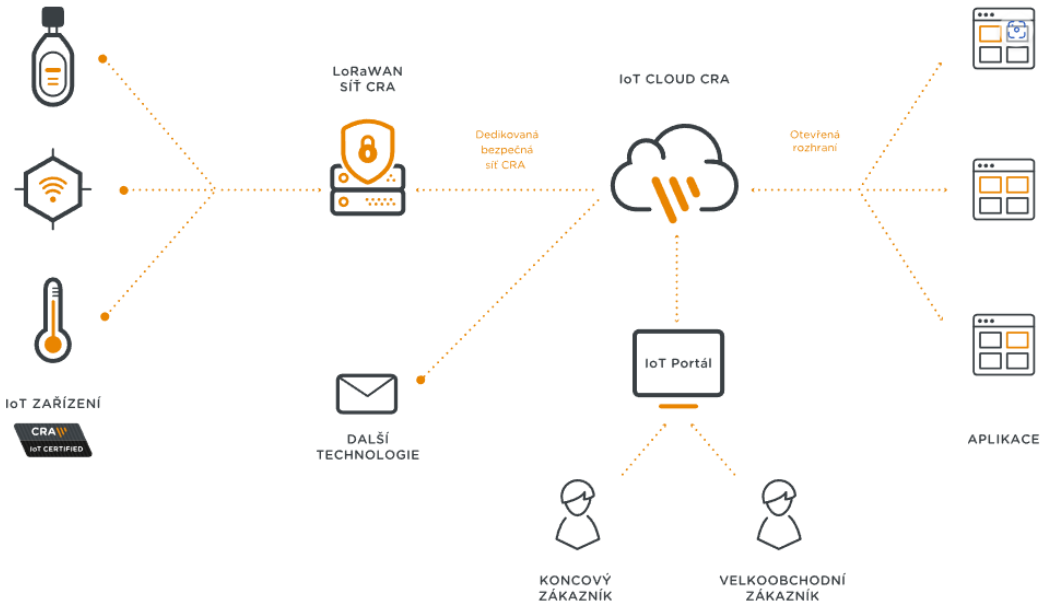
Využívá šest faktorů šíření, resp. SF7 – SF12. Pomocí nichž lze přizpůsobit přenosovou rychlost dat a kompromis rozsahu, kdy vyšší faktor šíření umožňuje větší dosah na úkor nižší přenosové rychlosti.

Hlavními výhodami LoRa jsou, jak je patrné z *Obrázku 3*, možnosti komunikace na velké vzdálenosti, nízká spotřeba provozu, snadná implementace a zabudování do systému, dostupná cena a široké pokrytí a možnost komunikace až do vzdálenosti 150 km.

Přenosová rychlost se nachází v rozmezí 300 bps až 50 kbps, v závislosti na faktoru šíření a šířce pásma. [3]

Celková délka datové části se může lišit dle komunikačního protokolu, avšak nejdelší možná délka datové části je 255 bajtů, kdy se LoRa, díky této velikosti, nehodí na rozsáhlou komunikaci. [4]

Je možné použít komunikační protokol založený na LoRa nazvaný LoRaWAN, jež byl standardizován společností LoRa-Alliance. Pomocí LoRaWAN je každá zpráva přenášena a přijata všemi základními stanicemi v dosahu. V České republice je dosaženo, díky společnosti CRA, téměř 98 %. [5]



Obrázek 4 IoT síť LoRaWAN od společnosti CRA [5]

Sigfox

Další velice populární je Sigfox, jež je provozovatelem sítě LPWAN, který nabízí end-to-end řešení IoT infrastruktury. Je založené na patentované technologii. Sigfox, stejně jako LoRa, využívá nelicencovaná pásma ISM. Díky použití ultra úzkého pásma, pracuje Sigfox efektivně s frekvenčním pásmem, avšak maximální propustnost je pouze 100 bps.

Zpočátku byla podpora omezena pouze na odesílání datových zpráv, ale později se vyvinula obousměrná technologie, stejně jako u LoRa.

Maximální počet odeslaných zpráv je omezen na 140 zpráv za 24 hodin a maximální datová část pro každou zprávu je omezena na pouhých 12 bajtů. Maximální počet přijatých zpráv je omezen na 4 zprávy za 24 hod s maximální datovou částí 8 bajtů.

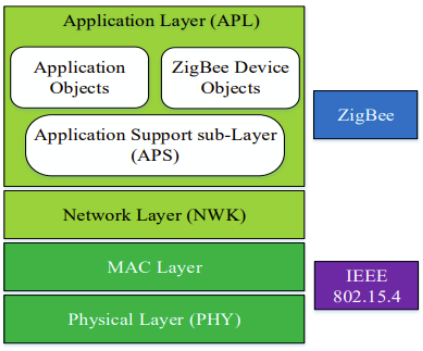
Jak je patrné není možné odeslat potvrzení na každou přijatou zprávu. Z tohoto důvodu je zajištění spolehlivosti odeslané zprávy pomocí časové a frekvenční rozdílnosti, kde je zahrnuta i duplikace přenosu, tzn. každá zpráva je odeslána vícekrát (často 3krát). Proto došlo k rozdělení např. pásma mezi 868,180 MHz a 868,220 MHz, na 400 ortogonálních 100 Hz kanálů, kdy je 40 kanálů vyhrazeno a nevyužito. [3]

Vzhledem k tomu, že stanice smí přijímat zprávy současně přes všechny kanály, mohou koncové zařízení náhodně vybírat frekvenční kanál, což zjednodušuje konstrukci koncových zařízení a snižuje náklady. Celkové pokrytí v České republice je 94 % [6]

ZigBee

Patří mezi další bezdrátové technologie. Je založená na průmyslových standardech, pro které byla primárně vyvinuta, aby umožňovala levné a nízkoenergetické sítě pro aplikace M2M (machine-to-machine) a IoT. [7]

Patří mezi otevřené standarty, což umožňuje mnoho kombinací implementace pro různé výrobce. ZigBee využívá mnohem nižších přenosových rychlostí ve srovnání s Wi-Fi. Dále ZigBee používá protokol topologie sítě, tím se vyhýbá centralizovaného vybavení a tvoří architekturu schopnou se sama opravovat. Fyzická vrstva pracuje ve standartu IEEE 802.15.4.



Obrázek 5 ZigBee topologie sítě [7]

1.4.2 IoT brána – databáze

Pro přístup k databázi je možné využít např. SQL nebo REST API. Je ale možné k databázi dat přistupovat i pomocí jiného systému IoT např. pomocí MQTT, kdy jsou data odesílána a přijímána pomocí tzv. MQTT brokeru, který lze zabezpečit pomocí šifrované komunikace.

SQL

Je databázovým počítačovým jazykem, jež je určený pro správu dat v systémech správy relačních databází. Pomocí něho lze snadno přistupovat do databáze, filtrovat data, mazat data, aktualizovat data. To vše s poměrně malou latencí. Data jsou strukturována pomocí tabulek. Lze nastavit bezpečností pravidla. Serverové databáze, které využívají SQL jazyka jsou např. Microsoft SQL Server (Azure), MySQL. [8]

REST API

Komunikační architektura REST API (Representational State Transfer), která byla vytvořena již v roce 2000, je konceptem, který je velice rozšířen při použití webových aplikací jako SaaS (Software as a Service). Pro datovou komunikaci využívá standartního protokolu HTTP. Díky využívání protokolu HTTP je možné použít TLS – šifrované spojení TCP/IP protokolu. [9]

MQTT

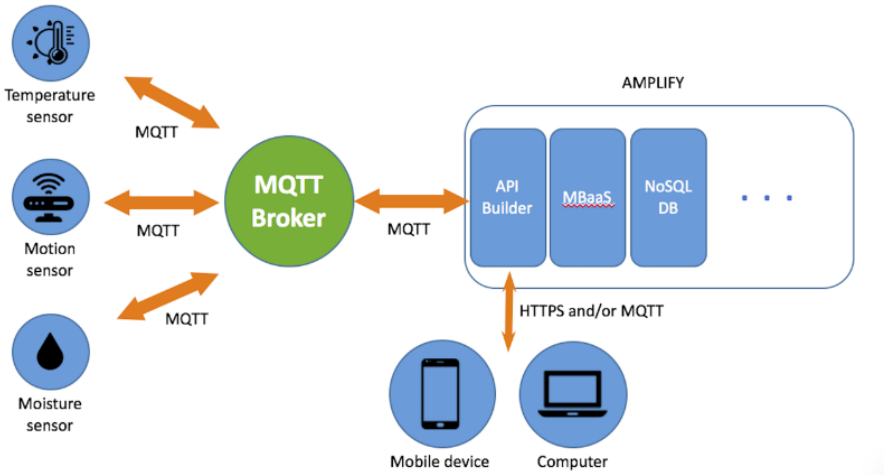
Patří mezi standartní protokol aplikační vrstvy pro IoT sítě. MQTT (Message Quering Telemetry Transport) organizuje a distribuuje zprávy mezi jednotlivá koncová zařízení.

Mezi hlavní komponenty patří MQTT klient. Označíme tak jakékoli zařízení, které může zprávy odesílat, říkáme že funguje jako vydavatel (publisher), tak i přijímat, poté funguje jako příjemce (subscriber). Z této podstaty vyplívá, že každé zařízení, které umí komunikovat pomocí MQTT přes síť, můžeme nazvat MQTT klientem. [10]

Další nedílnou součástí je MQTT broker, jenž je backendovým systémem, který slouží ke koordinaci zpráv mezi klienty. Hlavním účelem MQTT brokeru je příjem a filtrování zpráv, identifikace klientů, kteří jsou přihlášeni k odběru každé zprávy a samotné zasílání zpráv. Má za úkol autorizaci a ošetření MQTT klientů, předávání zpráv jiným systémům pro další analýzu a zpracování zmeškaných zpráv a relací klienta.

Samotná komunikace je zprostředkována pomocí protokolu TCP/IP. Klienti se nikdy nespojí napřímo mezi sebou, vždy musí využít MQTT brokera.

Díky komunikaci pomocí protokolu TCP/IP lze použít zabezpečeného spojení TLS, které využívá SSL k ochraně citlivých dat přenášených mezi zařízení IoT. Díky tomu může využít implementaci identity, ověřování a autorizaci mezi klienty a zprostředkovatelem pomocí SSL certifikátů a hesel.



Obrázek 6 Princip fungování MQTT systému [11]

1.5 Implementace IoT

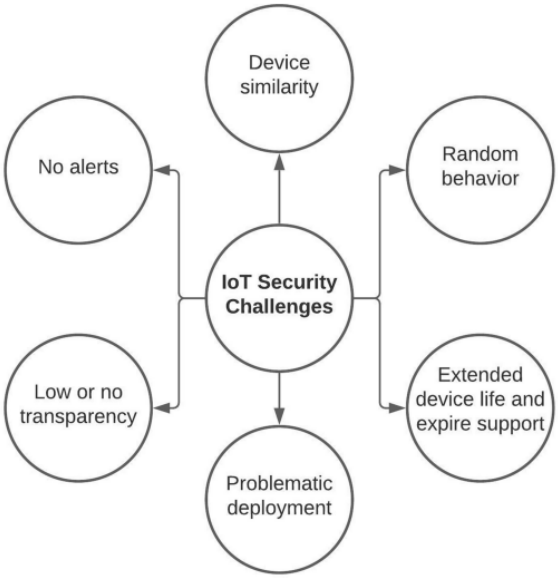
Výrobci často pro snadné použití a zprovoznění implementují jednotlivé části IoT přímo do specializovaných integrovaných obvodů nebo kombinovaných mikročipů, které podporují různé IoT komunikační systémové protokoly. Případně pro prototypování a testování je možné používat kity ve formě modemů či routerů. Typickým příkladem takové implementace je např. LoRaWAN, Sigfox či 6LoWPAN. To následně zajišťuje snadné použití v malosériové výrobě, kde by se vývoj na úrovni samotných součástek nevyplatil. Kity často obsahují rozhraní pro správu modemu např. sériová komunikace pomocí RS232/UART nebo pomocí USB, případně ethernetového rozhraní, kdy máme možnost přímého napojení na již existující jednotku. [1]

1.6 Bezpečnost

Vzhledem k pokročilému využívání IoT sítí je stále větší nutností zabezpečení jednotlivých částí. Problémy, kdy je nedostatečné zabezpečení, vznikají z důvodů, kdy např. nedostatečně zabezpečíme hardwarové nebo softwarové součásti infrastruktury. Takové části mohou generovat bezpečnostní problémy pro celou infrastrukturu. Nechráněná IoT síť může být napadena velice snadno útokům, zejména DoS útoku. Je proto důležité zabezpečit každý prvek infrastruktury. [1]

Možným zabezpečením je kompletní uzavřenost sítě v lokálním prostoru bez přístupu do veřejné sféry. Takové řešení je však vhodné pouze pro malé množství příkladů. Další možností by mohlo být šifrování komunikace, zde je pak kladen vyšší nárok na hardware a software, což ale často vyjde levněji, avšak je potřeba řešit pravidelnou správu a údržbu, kdy zajistíme bezpečný provoz i z veřejné části sítě.

Většina IoT sítí obsahuje osobní údaje, kde se můžou nacházet osobní informace. Z toho důvodu je zapotřebí anonymizovat a omezit správu soukromých údajů. Takovéto problematice se zabývá kryptografická technologie, která umožňuje ukládání a šíření zabezpečených dat. Případně je možné použít tzv. *měkký identifikátor*, kde jsou data uzpůsobena na konkrétní téma, aniž by došlo k zjištění zbytečných detailů, což by vedlo k porušení důvěrnosti. Můžeme také využít zabezpečené protokoly a systémy.



Obrázek 7 Výzva v oblasti bezpečnosti IoT [1]

1.7 Výhody a nevýhody

Jak již bylo řečeno, použití IoT sítí má nespočet výhod, ať už je to jeho snadná architektura, možnost dynamického výběru komunikačního systému, snadné implementace či velkému rozšíření mezi všechna zaměření. Avšak díky nestandardizovanému využívání různých komunikačních systému jsou kladeny vyšší nároky na připojování různých zařízení s IoT bránou, která musí disponovat všemi těmito systémy. Možným řešením by bylo vytvoření více podsítí, kde by každá síť obsahovala vlastní koncová zařízení a bránu, která by odesílala data do centrální databáze nebo na centrální IoT bránu, To ovšem zvyšuje cenu celé infrastruktury.

2 TVORBA IOT SÍTĚ

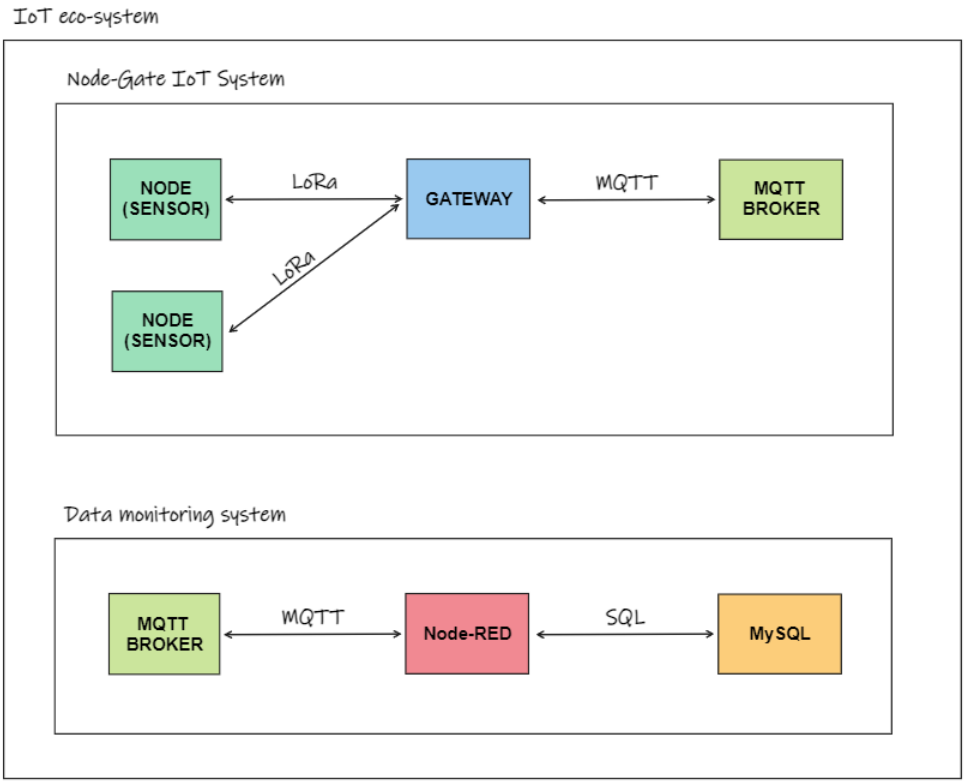
Pro vytvoření IoT sítě, bylo zapotřebí určení vlastností samotného systému a výběr jednotlivých částí architektury IoT.

2.1 Představa

Pro komunikaci mezi IoT zařízením a IoT bránou, jsem použil LoRa systém. LoRa mi umožní komunikaci na větší vzdálenost s možností přenášet až 255 bajtů v datové části. Nezarušuje okolní bezdrátové sítě, protože využívá dlouhé vlny o frekvenci 433 MHz. Nízké náklady za implementaci LoRa modulu do IoT koncových zařízení a IoT brány, díky tomu lze vytvářet levné IoT senzory, tlačítka, světla atd. Funkce v prostředí bez připojení k internetové síti.

Pro odesílání a příjem dat mezi IoT bránou a databází, jsem využil MQTT. Díky němu jsem schopný dostatečně zabezpečit a oddělit vnitřní IoT síť od veřejné sítě. MQTT broker mi poslouží jako manažer správy dat z IoT koncových zařízení. Díky MQTT je možné s daty pracovat poměrně snadno a rychle. Nutností je však připojení IoT brány pomocí Wi-Fi a protokolu TCP/IP, případně bezpečného spojení díky TLS s SSL zabezpečením.

Díky Node-RED aplikaci, jsem schopný s daty pracovat, kontrolovat, ukládat a monitorovat velice snadno. Node-RED poskytuje editor toku dat, který je založený na webovém prohlížeči, který lze využít pro vytváření funkcí pomocí JavaScriptu.



Obrázek 8 Návrh IoT sítě

2.2 Výběr komponentů

Pro správné fungování IoT sítě byl nutný výběr správně zvolených komponentů. Při výběru jsem cílil na nižší cenu a spotřebu energie.

2.2.1 LoRa

Na LoRa komunikaci, jsem použil modul RA-02 SX1278 na frekvenci 433 MHz od výrobce Ai-Thinker. Pracovní napětí samotného modulu je v rozmezí 1,8 – 3,7 V. Odebíraný proud v režimu spánku je uváděn 0,2 μA, v režimu příjmu dat okolo 10,8 mA a v režimu, kdy jsou data odesílána okolo 120 mA při +20 dBm. Možný dosah komunikace činí až 10 km, při použití dostatečně dimenzované antény. Výhodou je fungování v polo-duplexním SPI komunikačním rozhraní, díky kterému lze snadno nastavovat, číst a zapisovat data do modulu. Modul lze modifikovat pro jiné komunikační systémy např. OOK, FSK, MSK nebo GMSK.

Obrázek 9 LoRa modul RA-02 SX1278 [12]

2.2.2 ESP32

Jako hlavní řídící jednotku pro IoT bránu a IoT zařízení, zde využiji možností mikrokontroleru ESP32.

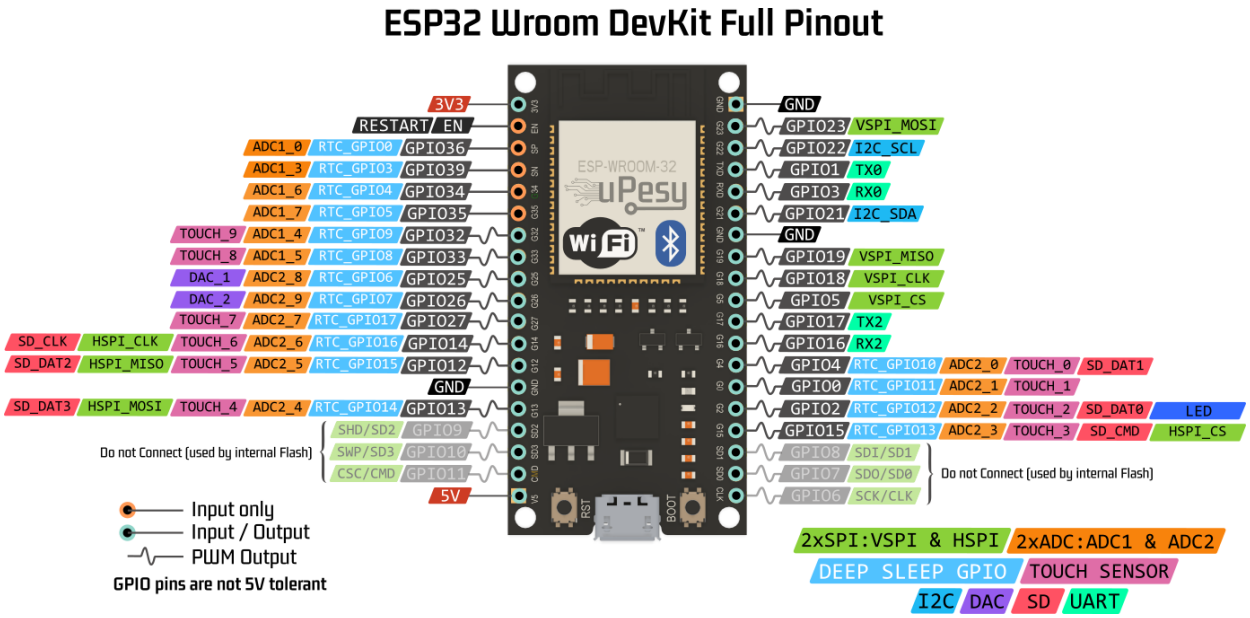
ESP32 patří k řadě levných mikrokontrolerů s nízkou spotřebou energie. Obsahuje integrovaný Wi-Fi modul s možností komunikační frekvence 2.4 GHz a duálním režimem Bluetooth/BLE ve verzi 4.2. Jako hlavní mikroprocesor je zde zabudován dvoujádrový Tensilica Xtensa LX6, jež funguje na architektuře RISC-V na frekvenci 80 až 240 MHz a zahrnuje veškeré anténní spínače. Dále obsahuje výkonný zesilovač, nízko-šumový přijímací zesilovač, filtry a důležitý modul řízení spotřeby energie.

Byl vyvinut čínskou společností, sídlící v Šanghaji, Espressif Systems. Je vyráběn pomocí 40nm procesu společností TSMC.

Mezi jeho hlavní funkce patří veliká 320 KiB RAM paměť, možnost bezdrátového připojení pomocí Wi-Fi a Bluetooth a velké množství vstupně-výstupních programovatelných GPIO pinů. Má možnost připojení pomocí rozhraní typu SPI, I2C, UART případně možnost připojení analogového signálu na A/D převodník nebo vytvoření analogového signálu pomocí D/A převodníku.

Nedílnou součástí je možnost připojení flash paměti přes dedikovaný SDIO/SPI řadič.

Pro snadnou implementaci přímo do projektu zde využívám vývojou desku s označením ESP-WROOM-32, která obsahuje již zabudovanou flash paměť o velikosti 4 MiB.



Obrázek 10 ESP-WROOM-32 vývojová deska s popisem rozložení GPIO pinů [13]

2.3 Výběr softwaru

Aby bylo možné zprovoznit IoT síť, je nutné vytvořit softwarový algoritmus, který nahraji do mikrokontroleru ESP32 a nastavit Node-RED včetně MQTT brokeru a MySQL databáze.

Pro programování mikrokontroleru ESP32 jsem použil oficiální Espressif IoT Development Framework (ESP-IDF), který je napsán v jazyce C a obsahuje plnou podporu real-time operačního systému FreeRTOS, kompatibilního s RISC-V mikroprocesory. Výhodou využívání ESP-IDF je možnost pokročilé konfigurace výsledného kódu s možností komplexního logování pomocí UART. Možnost přidávání a vytváření tzv. komponent, které zpřehledňují kód a lze díky nim přidávat programovou kompatibilitu přidaných modulů. Veškerá důležitá nastavení lze pro každou komponentu nastavovat pomocí příkazu *menuconfig*, díky přidanému souboru Kconfig, ve kterém se nachází definice konfiguračního menu dané komponenty. Obsahuje i novou verzi MQTTv5, díky čemuž mohu využít všech dostupných vlastností MQTT.

K MQTT brokeru lze díky kompatibilitě MQTTv3.3.1 a MQTTv5, připojit IoT bránu i s nižší verzí MQTT protokolu. Samotné spojení jsem zajistil pomocí bezpečného TLS spojení s autorizovaným přístupem pomocí uživatelského jména a hesla.

Přístup do databáze je zajištěna pomocí SQL jazyka.

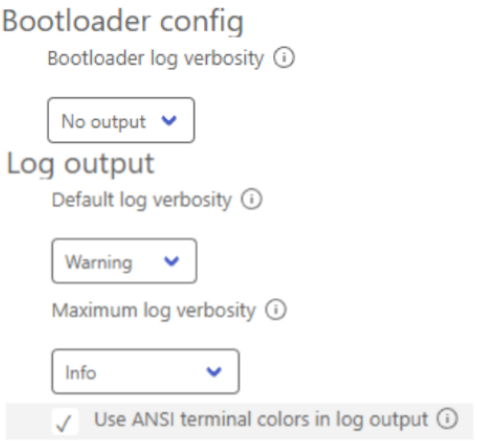
2.4 Příprava IDE prostředí

Abych mohl začít vytvářet algoritmus pro komunikaci mezi IoT bránou a IoT zařízením, byla nutná příprava a nastavení konfigurace IDE prostředí. Pro vývoj softwaru jsem použil IDE Visual Studio Code (VSCode) od Microsoft Corporation. Důvodem volby byla snadná orientace v prostředí a integrovaný IntelliSense, jež mi pomáhá při psaní, díky možnosti dokončování textu. Obsahuje i zvýrazňování syntaktických chyb a obarvování částí kódu, kdy se kód stává přehledným. Je zde možnost vyhledávání a nahrazování v různých částech kódu v celém projektu. Podstatnou je i přímá podpora rozšíření pro psaní kódu v ESP-IDF.

Po instalaci potřebných rozšíření pro psaní v jazyce C a v prostřední ESP-IDF, jsem mohl přejít k vytvoření samotného projektu pomocí šablony pro tvorbu projektu v ESP-IDF. Dále jsem provedl prvotní konfiguraci pro mnou vybranou vývojou desku s mikrokontrolerem ESP32. Nastavil jsem příslušný sériový port, na kterém byl mikrokontroler připojen, vybral jsem v nabídce kompatibilní firmware pro ESP32 a vybral protokol pro nahrávání a logování kódu. Vývojová deska využívá pro nahrávání a logování převodníku mezi rozhraní UART a USB.

Obrázek 11 Konfigurace připojeného mikrokontroleru ESP32

V menuconfig jsem vypnul možnost logování zavaděče a nastavil úroveň logování programu na varování, maximální možnou úroveň jsem ponechal na informativním, díky tomu mohu přímo v programu ovládat jednotlivé části logování a mít tak přehled o jednotlivých částech programu při budoucím testování a hledání chyb, pokud bych potřeboval logovat určitou část kódu více podrobněji, mohu v *menuconfig* nastavit nižší prioritu logování.



Obrázek 12 Nastavení úrovně logování programu

2.5 Tvorba programu

Vzhledem k použitému modulu pro komunikaci LoRa, jsem potřeboval přidat knihovnu do projektu. Při výběru komponentní knihovny jsem se snažil dodržet definované požadavky, tj. nízká spotřeba, možná konfigurace přímo v programu, kompatibilita s modulem SX1278 a snadná implementace. Vzhledem k nesplnění podmínek většiny řešení, která jsou dostupná, přistoupil jsem k vytvoření vlastní implementace modulu do programu.

2.5.1 LoRa modul

Abych mohl vytvořit implementační knihovnu pro modul LoRa SX1278, bylo zapotřebí prostudování příslušného datasheetu [14].

Nejprve jsem vytvořil kostru knihovny. Následně jsem vyřešil komunikaci s modulem pomocí SPI rozhraní. Slouží k tomu tyto dvě funkce pro čtení a zápis.

```
void lora_write_reg(lora_reg_t reg, uint8_t val);
uint8_t lora_read_reg(lora_reg_t reg);
```

Jako další jsem přidal funkce pro snadné nastavení nejdůležitějších vlastností pro správné fungování LoRa modulu.

```
static void lora_clear_irq(void);
static void lora_reset(void);
static void lora_set_tx_power(uint8_t level);
static void lora_set_frequency(uint64_t frequency);
static void lora_set_preamble_length(uint16_t len);
static void lora_set_sync_word(uint8_t word);
static void lora_enable_crc(void);
```

Funkce jsou použity pouze v rámci interní implementace, nelze je tedy využívat dále v programu. Použití funkcí je vázáno s inicializací modulu, které probíhá pouze jednou, a to na začátku programu. Slouží k tomu tato funkce.

```
esp_err_t lora_init(void);
```

Návratovou hodnotou je chybový kód, kdy, jestliže proběhne inicializace v pořádku, chybový kód bude *ESP_OK*.

Aby bylo možné definovat vlastnosti v rámci celého projektu, využil jsem možnosti projektové implementace konfigurace pomocí přiloženého souboru Kconfig.

Obrázek 13 Možnost konfigurace LoRa modulu

Z konfigurace je patrné, že lze specifikovat vlastnosti chování modulu, výběr výstupních pinů na mikrokontroleru ESP32 a výběr SPI rozhraní. Volitelná možnost je i implementace DIO pinů, které se nachází na LoRa modulu a slouží např. pro detekci přerušení. Díky tomu je možné během čekání na výsledek, zpracovat jiné části programu.

Operating Mode	DIOx Mapping	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
ALL	00	ModeReady	CadDetected	CadDone	FhssChangeChannel	RxTimeout	RxDone
	01	ClkOut	PllLock	ValidHeader	FhssChangeChannel	FhssChangeChannel	TxDone
	10	ClkOut	PllLock	PayloadCrcError	FhssChangeChannel	CadDetected	CadDone
	11	-	-	-	-	-	-

Tabulka 1 Možná konfigurace DIO pinů [14]

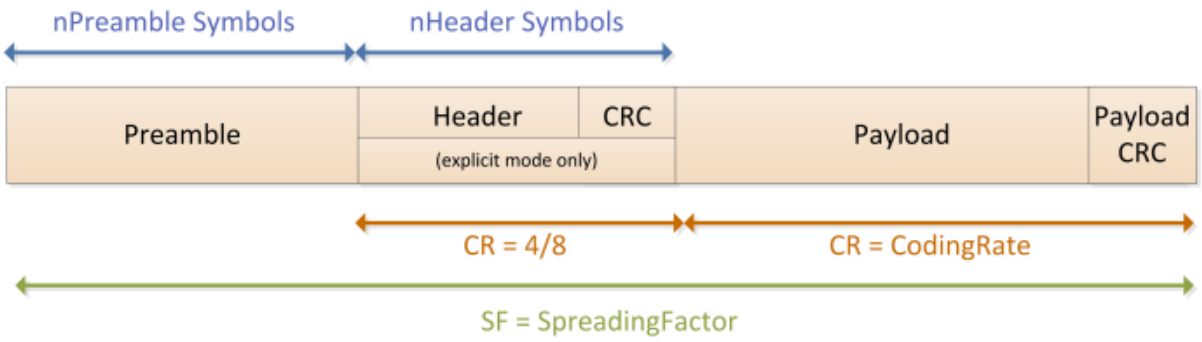
Přidal jsem možnost uspání modulu, kdy se spotřeba pohybuje okolo 0,2 µA. V tomto režimu není možné přistupovat k žádnému z vnitřních registrů, kromě změny režimů.

```
void lora_sleep(void)
```

Symbol	Description	Conditions	Min	Typ	Max	Unit
IDDSL	Supply current in Sleep mode		-	0.2	1	uA
IDDIDLE	Supply current in Idle mode	RC oscillator enabled	-	1.5	-	uA
IDDST	Supply current in Standby mode	Crystal oscillator enabled	-	1.6	1.8	mA
IDDFS	Supply current in Synthesizer mode	FSRx	-	5.8	-	mA
IDDR	Supply current in Receive mode	*LnaBoost* Off, band 1 *LnaBoost* On, band 1 Bands 2&3	- - -	10.8 11.5 12.0	- - -	mA
IDDT	Supply current in Transmit mode with impedance matching	RFOP = +20 dBm, on PA_BOOST RFOP = +17 dBm, on PA_BOOST RFOP = +13 dBm, on RFO_LF/HF pin RFOP = + 7 dBm, on RFO_LF/HF pin	- - - -	120 87 29 20	- - - -	mA mA mA mA

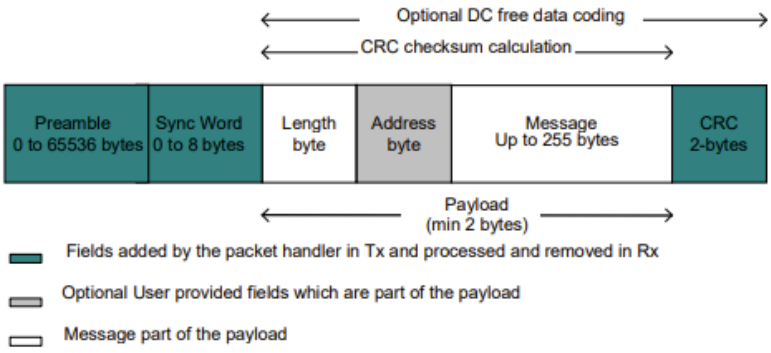
Tabulka 2 Spotřeba pro různé režimy modulu [14]

Data jsou odesílána a přijímána pomocí paketu. Obsahuje preambulační sekvenci (0101) o velikosti n, maximální velikost však může být až 2^{16}. Jako další následuje hlavička paketu. Hlavička je použita pouze, je-li nastaven explicitní režim datové struktury. Následuje samotná část datové zprávy o maximální velikosti 255 bajtů. Celý paket je zakončen kontrolním součtem CRC16, který je vypočítáván z datové části.



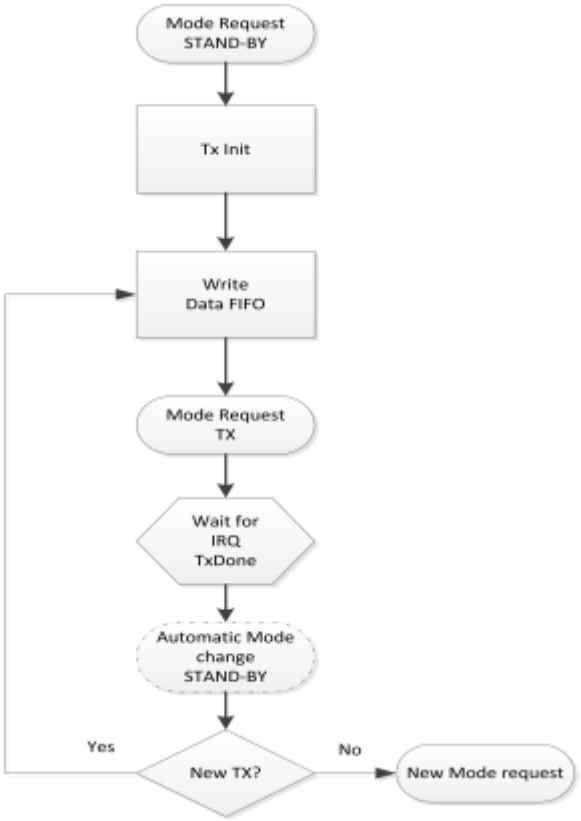
Obrázek 14 Struktura LoRa paketu [14]

V projektu využívám možnosti odesílání variabilně dlouhého paketu, kdy jsou krátké zprávy zpracovány rychleji a také díky tomu není zapotřebí alokovat vetší část mezi-paměti. Do hlavičky paketu je vloženo synchronizační slovo, což představuje virtuální oddělení sítí, kdy nejsou přijímány pakety s jiným synchronizačním slovem. Dále zde vkládám délku datové části, kdy mohu snadno docílit rozpoznání celkové velikosti paketu.



Obrázek 15 Formát variabilně dlouhého paketu [14]

Odesílání dat probíhá v sekvencích, kdy nejprve inicializuji modul k odesílání dat, tj. nastavím příslušný režim. Zapíši data do FIFO paměťového registru a přepnu vysílač do režimu odesílání. Modul vyčká na odeslání celého paketu a automaticky se přepne do režimu stand by a uspí vysílač.



Obrázek 16 Diagram odesílání dat [14]

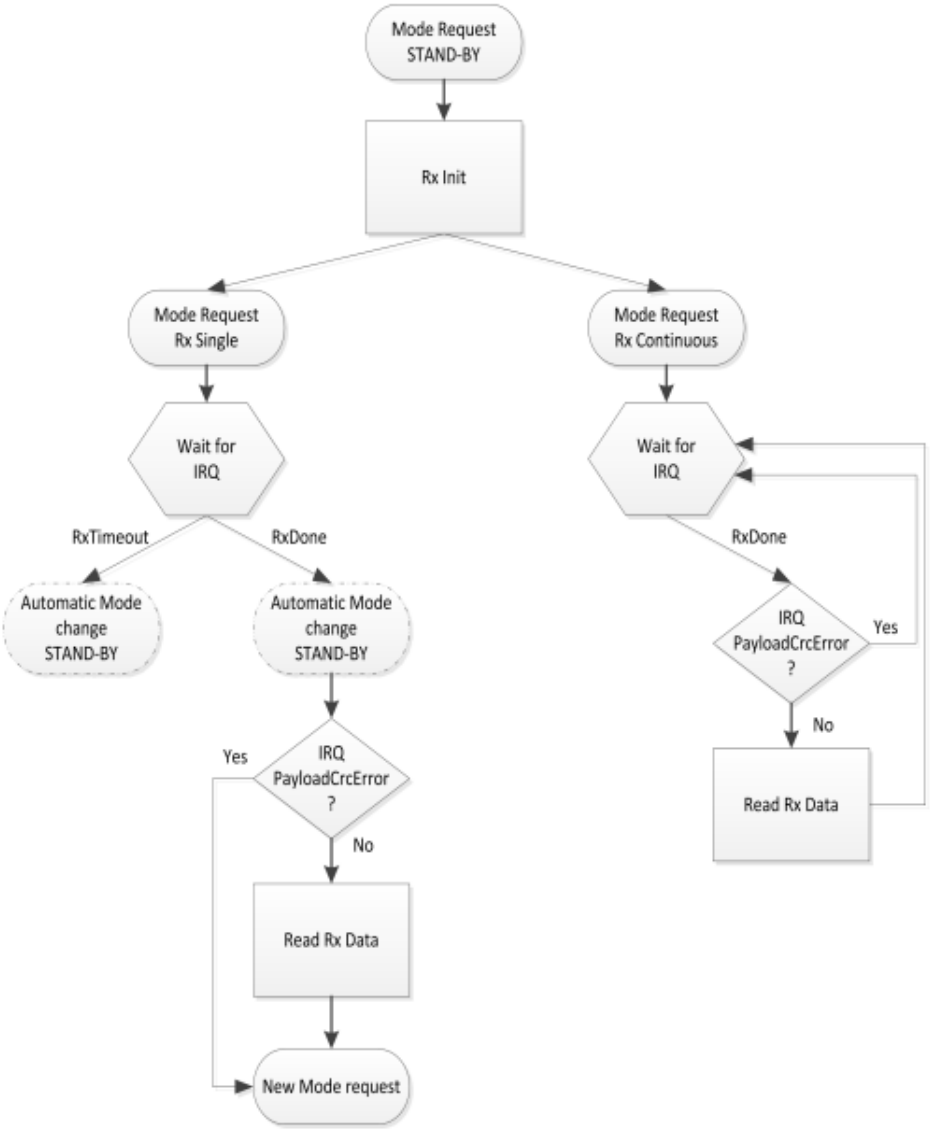
V programu pak stačí nejdříve inicializovat začátek odesílání pomocí *lora_begin_tx*, dále je možné zapisovat pomocí *lora_write_tx* libovolně dlouhé části dat, která jsou nahrávána za sebou do FIFO paměti. Při zapisování je možné ověřovat, zda lze pokračovat v zápisu, tj. kolik volného místa pro zápis je dostupné. K odeslání slouží poslední funkce *lora_end_tx*.

```
void lora_begin_tx(void);
esp_err_t lora_write_tx(uint8_t *buf, uint8_t size);
esp_err_t lora_end_tx(void);
```

Příjem dat lze provádět více způsoby, kdy jsem do kódu zahrnul pouze dvě možnosti:

- Standartní příjem dat, čekající na příchod jednoho příjmu
- Vyčkávání na CAD a následný pokus o příjem dat

První z možností, jak přijímat je standartní příjem, kdy je modem přepnut do režimu příjmu a zapnut přijímač. Následně modul vyčkává příchodu preambulace. Pokud nepřijde, během časového období, které je nastaveno, dojde k přechodu do režimu stand-by a vypnutí přijímače. Tento režim je označen jako tzv. režim jednoho příjmu (RX Single).



Obrázek 17 Diagram standartního příjmu dat [14]

V programu pak stačí nejdříve inicializovat začátek příjmu pomocí *lora_begin_rx*, v případě úspěšného přijetí, vrátí celkovou velikost dat. Čtení jednotlivých bajtů probíhá pomocí funkce *lora_read_rx*. Konec čtení dat již není potřeba uvádět, jelikož veškerá data, která byla přijata jsou již uložena v registru FIFO a modul se automaticky nachází v režimu stand-by.

```
esp_err_t lora_begin_rx(uint8_t *len);
esp_err_t lora_read_rx(uint8_t *buf, uint8_t size);
void lora_end_rx(void);
```

Pokročilou možností, jak přijímat data, je využití techniky CAD, technika používající modulace rozprostřeného spektra, která má význam při určování, zda je kanál již používán signálem, který může být pod šumovým prahem přijímače. [14] Použití RSSI by v této situaci bylo zjevně neproveditelné. Proto se využívá detekce aktivity kanálu (CAD), k určení přítomnosti dalších LoRa signálů. Výhodou je nižší spotřeba pro detekci příjmu dat.

Obrázek 18 Modifikace diagramu pro příjem dat s využitím CAD [14]

Provedl jsem tedy modifikaci v rámci kódu, kdy jsem implementoval do funkce *lora_waiting_cad_rx* CAD pro snadné použití, kdy funkce vyčkává v nekonečném cyklu na detekci CAD, následně spustí příjem dat.

```
void lora_waiting_cad(void);
bool lora_is_cad_detected(void);

esp_err_t lora_waiting_cad_rx(uint8_t *len);
esp_err_t lora_read_rx(uint8_t *buf, uint8_t size);
void lora_end_rx(void);
```

Funkce *lora_waiting_cad*, vyčkává v nekonečné smyčce na detekci CAD, oproti tomu funkce *lora_is_cad_detected*, otestuje pouze jednou, zda nedošlo k detekci CAD.

Poslední užitečnou součástí je možnost vyčtení vlastností posledního příchozího paketu, tedy jeho RSSI a SNR.

```
int16_t lora_packet_rssi(void);
float lora_packet_snr(void);
```

2.5.2 Wi-Fi manažer

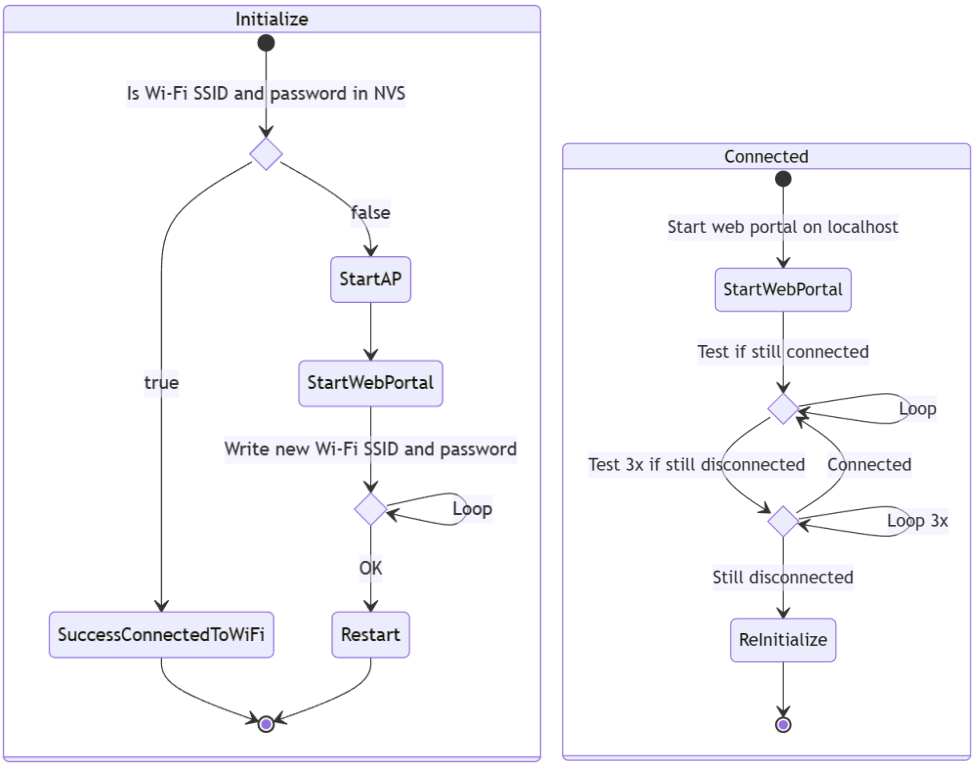
IoT systém, který využívám, potřebuje připojení k MQTT brokeru. Ten využívá pro svoji funkci protokol TCP/IP, resp. TLS. Aby bylo možné připojit IoT bránu k MQTT brokeru, je potřeba připojení, do již existující LAN sítě, ve které se nachází právě MQTT broker. Vzhledem k použitému mikrokontroleru ESP32, je jedním z řešení připojení pomocí Wi-Fi. Ta využívá připojení pomocí názvu sítě (SSID) a hesla. Problémem je, jak zajistit specifické připojení k síti.

Možným řešením by bylo zanést přímo do programu název sítě a patřičné heslo. Takové řešení, ale bude funkční pouze, pokud znám předem specifikované prostředí, ve kterém se bude IoT síť implementovat.

Lepším a spolehlivým řešením je zprovoznění Wi-Fi manažeru, který si vyžádá název sítě a heslo. Následně dojde k uložení do paměti zařízení pro budoucí snadné připojování do sítě Wi-Fi.

Jako Wi-Fi manažera, jsem vybral hotové řešení od Tonyho Pottiera [15], který vytvořil velice snadné řešení implementace přímo do projektu. V samotném programu pak stačí pouze na začátku vložit inicializační funkci, která se postará o vyžádání údajů.

Celé řešení spočívá v tom, že nejprve otestuje, zda se nenachází přístupové údaje k Wi-Fi síti. Pokud ano, pokusí se o připojení k síti, v opačném případě, kdy nebudou nalezeny údaje, nebo dojde při pokusu k chybě, vytvoří vlastní AP (access point), na kterém spustí jednoduché webové rozhraní pro výběr a zadání přístupových údajů. Po zadání dojde k restartování a opakování procesu.



Obrázek 19 Diagram funkce Wi-Fi manažeru

2.5.3 MQTT

V projektu používám vývojou sadu ESP-IDF. Díky tomu mohu využít již vestavěné implementace MQTT.

2.5.4 Protokol NGP

Aby byla komunikace pomocí LoRa snadno čitelná, vytvořil jsem nenáročný převod pomocí tzv. node-gate protokolu (NGP). Hlavním požadavkem při vývoji protokolu byla možnost autentizace, posílání aktuálního času v unixovém tvaru, tj. timestamp a nastavení hlavičky a dat zpráv včetně QoS pro odesílání přes MQTT.

Struktura dat

Struktura je organizována formou paketů. Samotný paket obsahuje hlavičku, datovou část zprávy a možnost přidání CRC.

HEADER															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RX token															
TX token															
QoS		PART		FNS				ID							
Property								Payload length							
Topic[0]								Topic[1]							
Topic[2]								Topic[3]							
PAYLOAD															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Data[0]								...							
...								...							
...								...							
...								...							
...								...							
...								Data [N]							
FOOTER															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CRC16															

Tabulka 3 NGP paket

FNS							
00h	01h	02h	03h	04h	05h	06h	07h
NONE	ECHO	MSG	OK	ERR	SNTP	NTOKEN	-

Tabulka 4 Možné hodnoty FNS

PART			
00h	01h	02h	03h
MAIN	1	2	3

Tabulka 5 Možné hodnoty čísla fragmentu

ID										
00h	FFh	48h	49h	126h	68h	83h	66h	76h	82h	87h
-	-	'0'	'1'	'~'	'D'	'S'	'B'	'L'	'R'	'W'
EMPTY	-	GATE	NODE	NOT SPECIFIED	IOT DEVICE	IOT SENSOR	IOT BUTTON	IOT LIGHT	IOT RELE	IOT SWITCH

Tabulka 6 Označení ID zařízení

V hlavičce paketu se nachází autorizační token odesílatele zprávy a token příjemce. Následně QoS zprávy sloužící pro určení priority pro MQTT. Jako další je číslo fragmentu zprávy, kdy hodnota znamená pozici offsetu zprávy. Označení příslušného paketu pomocí FNS (číslo stavové funkce). Dále ID, které slouží k označení zařízení, jež zprávu odeslalo a Property, označující volitelný byte, který může sloužit k hlubšímu označení zařízení. Jako další se zde nachází velikost datové části paketu. Konec hlavičky je vyhrazen 4 byty pro jméno hlavičky zprávy pro MQTT.

Za hlavičkou paketu následuje dynamicky alokovaná datová část zprávy, která může mít velikost od 1 bytu po 241 bytů. Délka datové části musí striktně korespondovat s délkou uvedenou v hlavičce.

Poslední částí paketu je možnost přidání kontrolního součtu CRC16, který je vypočítán ze všech dat celého paketu. V případě, kdy není použit, musí mít hodnotu 0.

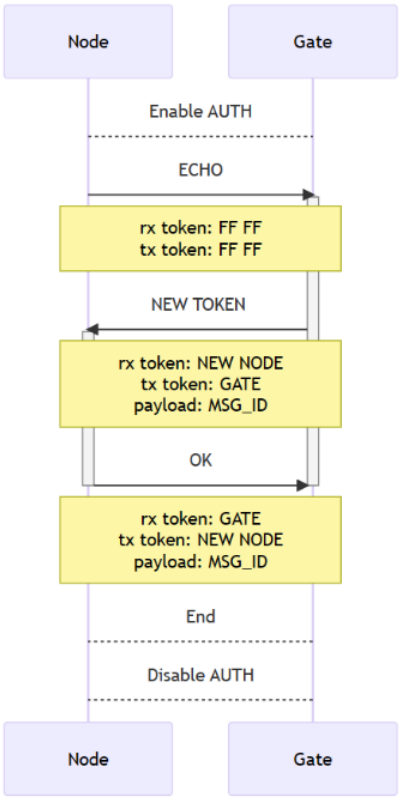
Token

Token je generován IoT bránou a slouží pro identifikaci jednotlivých zařízení v LoRa síti. Pro IoT bránu je generován vždy pokud již není uložen v NVS paměti nebo po vyžádaném restartu. Token pro IoT zařízení je generován na vyžádání, kdy zařízení, pokud nemá specifikován token, pošle autorizační požadavek na IoT bránu. Samotný požadavek je sekvencí několika zpráv.

Autorizační sekvence

Na začátku sekvence musí být na IoT bráně povolena možnost přijímat požadavek na autorizaci, v opačném případě jsou zprávy ignorovány.

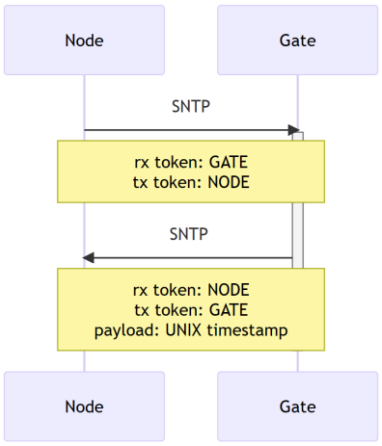
Jako první musí IoT zařízení vyslat tzv. ECHO paket, kde jsou tokeny nastaveny na broadcast adresu, tj. 0xFFFF a FNS zprávy je nastaven na ECHO. Následně IoT brána vytvoří nový token a odešle zprávu, kde nastaví vlastní token na tx a nový token na rx. Dále nastaví FNS na NEW_TOKEN a jako datovou část nastaví identifikátor zprávy. IoT zařízení přijme paket, uloží si tokeny do paměti a odešle OK, kdy do datové části nastaví stejný identifikátor zprávy pro potvrzení příjmu. Jestliže dorazil paket v pořádku na IoT bránu a identifikátor je shodný, uloží id do hashovací tabulky, pomocí níž lze identifikovat, zda byla zpráva od autorizovaného zařízení či nikoli.



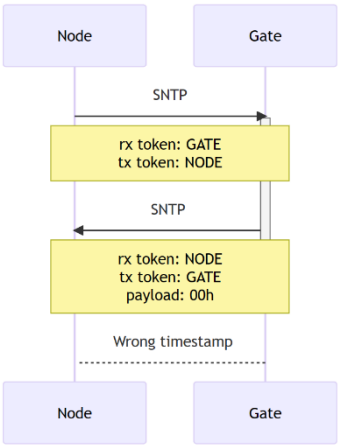
Obrázek 20 Autorizační sekvence

Časová synchronizace

Zařízení po autentizaci, mají možnost vyžádání si aktuálního času přímo v rámci paketu, kdy jako FNS nastaví SNTP. IoT brána následně odešle aktuální Unix timestamp, který zapíše do datové části. V případě, kdy je datová část nulová, se bude příchozí paket považovat za chybný. Důvodem může být např. nepřipravenost časových dat ze strany IoT brány.



Obrázek 21 Sekvence úspěšného získání času



Obrázek 22 Sekvence neúspěšného získání času

Odesílání zpráv na MQTT

Aby bylo možné odeslat zprávu na MQTT, musí být IoT brána připojena na MQTT brokera a IoT zařízení musí být autentizováno. Pro odeslání již pak stačí vyplnit datovou část zprávy a MQTT hlavičku a odeslat pomocí funkce *ngp_fns_send*, která přidá správné nastavení hlavičky paketu, tedy tokenů, ID, a FNS nastaví na MSG. Volitelnou funkcí je možnost nastavení QoS a fragmentu.

Pro kompresi dat jsem implementoval možnost komprimovat data pomocí msgpacku. Ten umí formátovat data ve tvaru JSON do binární zlehčené formy, kdy není na datovou část nahlíženo jako pole znaků, ale pole bytů. Po kompresi již nejsou data dobře čitelná pro člověka jako kdyby byly ve znakovém formátu JSON.

V kódu nejprve alokujeme buffery a následně zapíšeme data pomocí msgpack funkcí.

```
ngp_packet_t *packet = malloc(sizeof(ngp_packet_t));
memset(packet, 0x00, sizeof(ngp_packet_t));
ngp_msgpack_buffer_t msg_buffer;
ngp_packet_msgpack_init(&msg_buffer);
msgpack_packer *pk = &msg_buffer.pk;
```

Příklad převodu JSON dat na tvar msgpack:

```
{
    "data": 125,
    "data2": "hodnota",
}
```

```
msgpack_pack_map(pk, 2);

msgpack_pack_str_with_body(pk, "data", sizeof("data")-1);
msgpack_pack_uint8(pk, 125);

msgpack_pack_str_with_body(pk, "data2", sizeof("data2")-1);
msgpack_pack_str_with_body(pk, "hodnota", sizeof("hodnota")-1);
```

Po zadání dat, provedeme serializaci, která uloží data z bufferu do datové části paketu a nastaví správnou délku do hlavičky.

```
ngp_packet_serialize(packet, &msg_buffer);
```

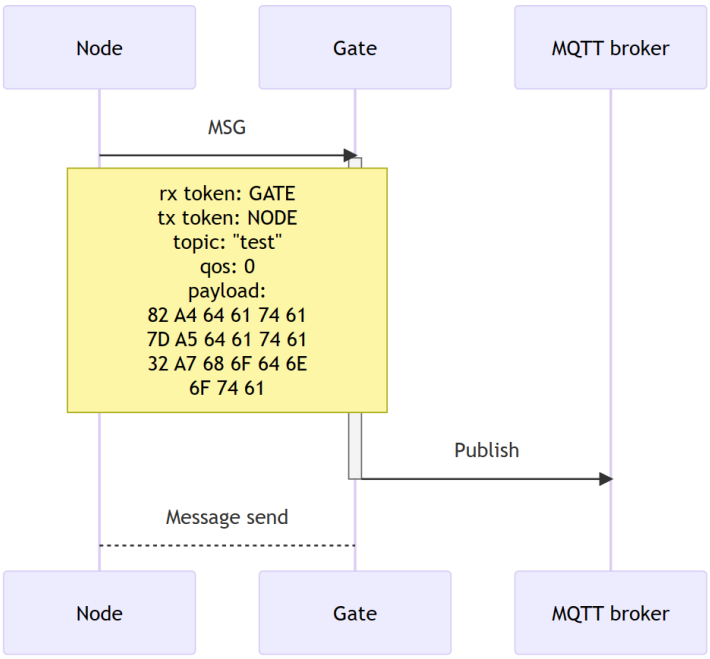
Poté přidáme hlavičku pro MQTT a QoS

```
memcpy(packet->header.fld.topic, "test", 4);
packet->header.fld.config.qos = 0;
```

Nakonec provedeme odeslání zprávy.

```
ngp_err_t ngp_fns_send_msg(ngp_packet_t *packet);
```

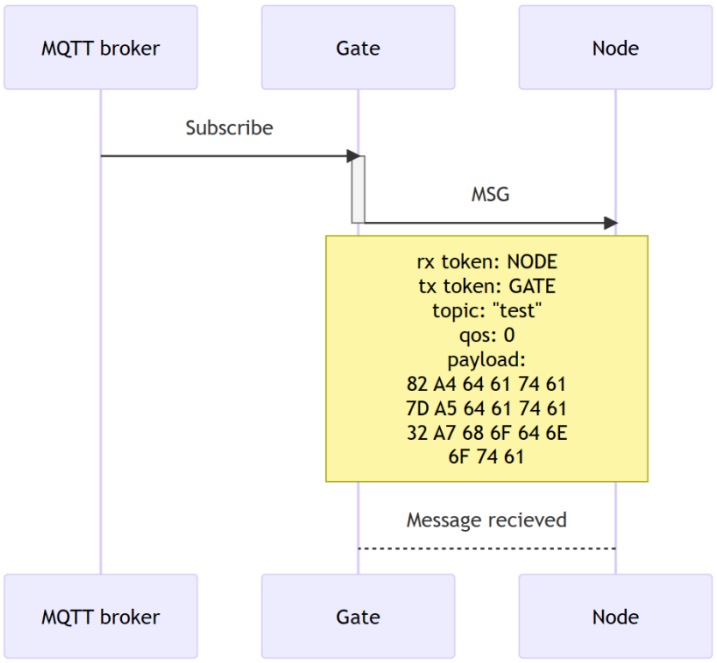
Ukázka sekvence odeslání na MQTT



Obrázek 23 Sekvence odesílání zprávy

Pro příjem dat z MQTT slouží msgpack objekt, který vznikne při deserializaci dat pomocí funkce.

```
ngp_err_t ngp_packet_deserialize(ngp_msgpack_object_t *obj, ngp_packet_t *packet);
```



Obrázek 24 Sekvence příjmu zprávy

Samotných možných funkcí je celá řada. Veškeré další možnosti převodu dat lze najít v dokumentaci msgpack [16].

2.5.5 GPIO rozhraní

Aby bylo možné identifikovat jednotlivá stádia průběhu programu na IoT bráně, přidal jsem pět stavových LED diod a jedno konfigurační tlačítko.

Tlačítko slouží pro zapnutí/vypnutí autorizačního stavu, případně pro reset uložených nastavení wifi připojení nebo uvedení zařízení do továrního nastavení, kdy dojde k odstranění všech uložených dat v NVS paměti a následnému restartu zařízení.

Po startu zařízení se otestuje funkce LED diod. Následně se již bude chování LED diod a tlačítka, nastavovat podle stavového automatu.

LED0 (RX_LED):

Svítí – příchod dat do bufferu, čekající na zpracování

Blikne – data byla zpracována programem

Nesvítí – žádná příchozí data

LED1 (TX_LED):

Svítí – předání dat do bufferu, čekající na odeslání

Blikne – data byla zpracována programem

Nesvítí – žádná odchozí data

LED2 (WIFI_LED):

Svítí – čekání na inicializaci

Pomalé blikání – odpojení od internetu

Rychlé blikání – zapnutí access pointu pro nastavení wifi ssid a hesla

Nesvítí – připojen k internetu

LED3 (MQTT_LED):

Svítí – čekání na inicializaci

Pomalé blikání – odpojení od MQTT brokeru

Nesvítí – připojen k MQTT brokeru

LED4 (AUTH_LED):

Svítí – je povolen privilegovaný režim

Pomalé blikání – po uvolnění tlačítka dojde k resetování wifi ssid a hesla

Rychlé blikání – po uvolnění tlačítka dojde k továrnímu resetu NVS paměti

Nesvítí – je vypnut privilegovaný režim

BTN0 (AUTH_BTN):

Po 2 sekundách stisku – po uvolnění tlačítka dojde k vypnutí/zapnutí autorizačního režimu

Po 6 sekundách stisku – po uvolnění tlačítka dojde k resetování wifi ssid a hesla

Po 10 sekundách stisku – po uvolnění tlačítka dojde k továrnímu resetu NVS paměti

2.5.6 Ukázka programu (IoT brána)

```
// Import iot_lora component
#include "iot_lora.h"

// Loggíng
#include "esp_log.h"
static const char TAG[] = "main";

// Wifi handler for log local time
void wifi_handler(iot_wifi_status_t s, void *args)
{
    if (s == IOT_WIFI_NTP_SYNC)
    {
        time_t now = 0;
        ESP_LOGI(TAG, "TIME: %lli", time(&now));
    }
}

void app_main(void)
{
    // Initialization leds and wait 2s for check if leds working
    iot_led_init();
    vTaskDelay(pdMS_TO_TICKS(2000));

    // Initialization NGP as gate id
    ngp_init((ngp_key_t){.id = NGP_ID_GATE, .property = 0x12}, NGP_ID_GATE, false);

    // Start program
    iot_init();

    //  Initialization wifi manager
    iot_wifi_cb_init(wifi_handler);
    iot_wifi_init();

    // Wait for wifi connected
    while (!iot_wifi_connected)
    {
        vTaskDelay(10);
    }

    // Start lora module
    iot_lora_init();

    // Start MQTT module
    iot_mqtt_init();

    // Infinite loop
    for (;;)
    {
        vTaskDelay(portMAX_DELAY);
    }
}
```

2.5.7 Ukázka programu (IoT zařízení)

```
// Import iot_lora component
#include "iot_lora.h"

// Logging
#include "esp_log.h"
static const char TAG[] = "main";

void app_main(void)
{
    // Initialization NGP as IoT device id (node)
    ngp_init((ngp_key_t){.id = NGP_ID_IOT_DEVICE, .property = 0×01}, NGP_ID_NODE, false);

    // Start program
    iot_init();

    // Get the latest time from gate

    // Pointer definition for a packet
    ngp_packet_t *packet;

    // Max wait
    uint8_t wait = 10;
    do
    {
        packet = malloc(sizeof(ngp_packet_t));
        ngp_err_t err = ngp_fns_send_sntp(packet);
        if (err ≠ NGP_ERR_OK)
        {
            ESP_LOGW(TAG, "ERR %i", err);
        }
        // Wait 5s
        vTaskDelay(pdMS_TO_TICKS(5000));
        wait--;
    } while (!ngp_global.sntp_ok && wait > 0);

    // If not set sntp_ok flag, start ECHO beacon, because gate not response
    if (!ngp_global.sntp_ok)
    {
        // Setting the NGP node to privileged mode
        ngp_global.privileged = true;

        do
        {
            packet = malloc(sizeof(ngp_packet_t));
            ngp_err_t err = ngp_fns_send_echo(packet);
            if (err ≠ NGP_ERR_OK)
            {
                ESP_LOGW(TAG, "ERR %i", err);
            }
            // Wait 5s
            vTaskDelay(pdMS_TO_TICKS(5000));
        } while (!ngp_global.echo_ok);

        // If set echo_ok flag, restart, gate is success create new token
        esp_restart();
    }

    // Log local time
    time_t now = 0;
    ESP_LOGI(TAG, "TIME: %lli", time(&now));
    struct tm timeinfo;
    iot_get_time(&timeinfo);

    // Infinite loop
    for (;;)
    {
        vTaskDelay(portMAX_DELAY);
    }
}
```

2.6 Zpracování dat

Pro ukázku zpracování dat, jsem využil bakalářské práce od spolužáka Lukáše Poklopa, který vytváří Agro senzor, jež slouží ke sledování teploty a vlhkosti půdy a okolí.

2.6.1 Rozbor dat

Data jsou formátována ve tvaru JSON a následně zkomprimována a odeslána jako msgpack.

JSON

```
{
  "time": 1681659782,
  "battery": 100,
  "temp_amb": 254,
  "temp_soil": 223,
  "moisture": 100,
  "humid": 423
}
```

MSGPACK

```
86 A4 74 69 6D 65 CE 64
3C 17 86 A7 62 61 74 74
65 72 79 64 A8 74 65 6D
70 5F 61 6D 62 CC FE A9
74 65 6D 70 5F 73 6F 69
6C CC DF A8 6D 6F 69 73
74 75 72 65 64 A5 68 75
6D 69 64 CD 01 A7
```

Na první pozici se nachází hodnota „time“, která reprezentuje Unix timestamp. Jako další je „battery“, což jsou procenta nabití baterie v rozsahu od 0–100 %, následuje „temp_amb“, jež je teplota okolí ve °C. Hodnota je násobena 10 z důvodu vyšší komprese, kdy je hodnota reprezentována jako celé, nikoliv desetinné, číslo. Další v pořadí je „temp_soil“, teplota půdy ve °C. Hodnota je opět násobena 10. Dále je zde „moisture“, což označuje hodnotu vlhkosti půdy v rozsahu 0-100 %. Poslední hodnotou je „humid“, jež je hodnota vlhkosti okolí v rozsahu 0-100 %. Hodnota je také násobena 10.

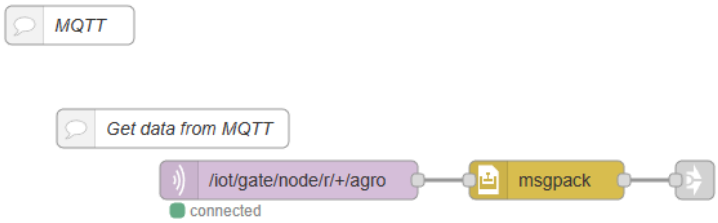
2.6.2 Node-RED

Data, která byla odeslána z IoT zařízení na IoT bránu a následně předána MQTT brokeru, je možné přijímat, pro možné zobrazování a ukládání do SQL databáze, přímo v rámci Node-RED.

V Node-RED jsem vytvořil několik flow, která se starají o správné předávání a formátování dat.

Flow – MQTT:

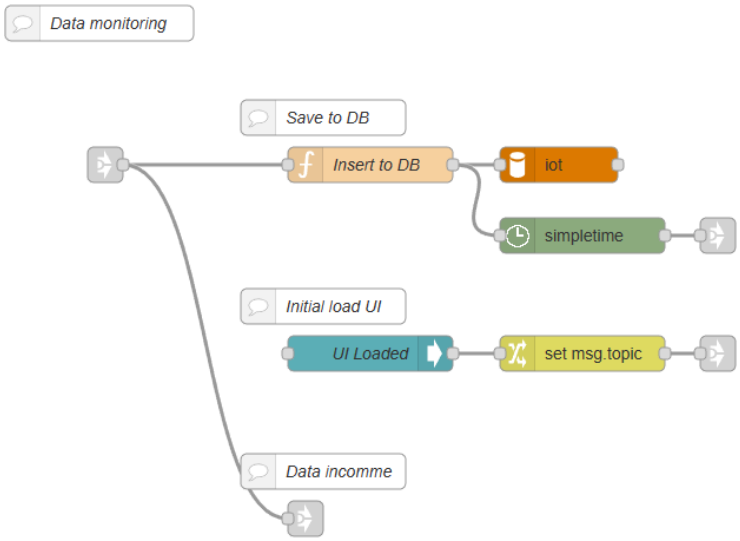
Slouží pro příjem dat z MQTT brokeru a dekompresi z msgpack do JSON formátu.



Obrázek 25 Node-RED – MQTT

Flow – Monitorování dat:

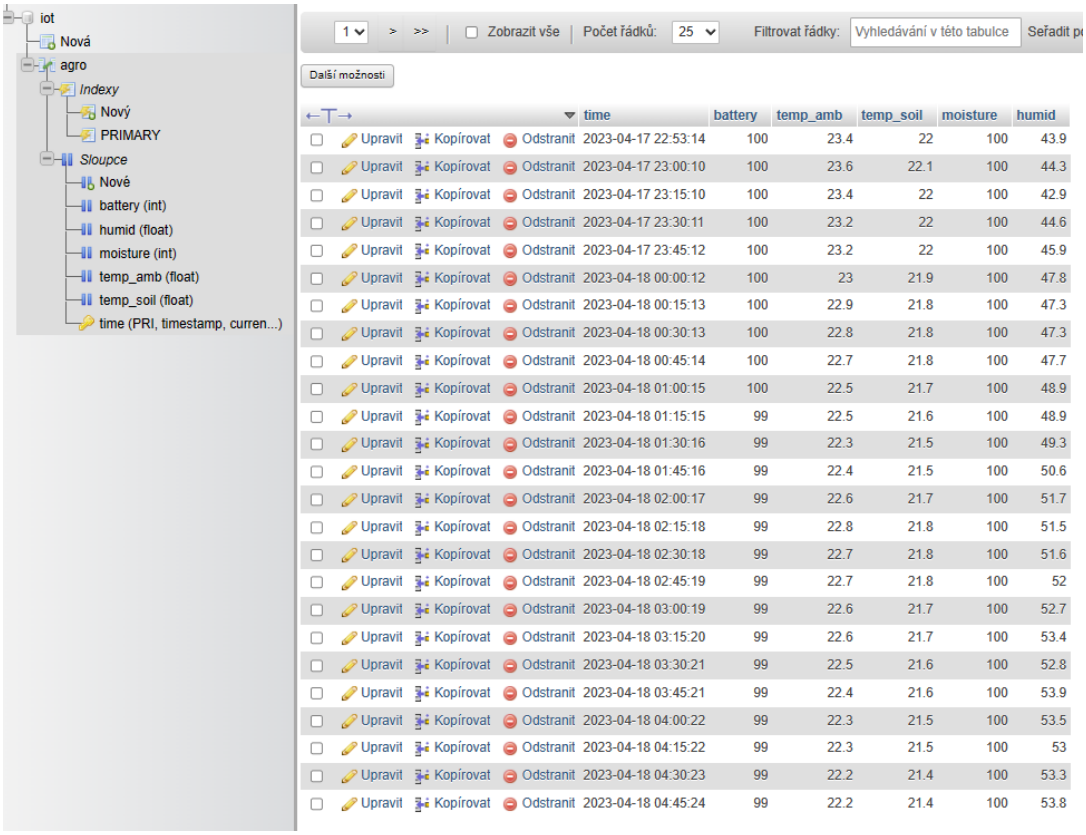
Účelem je data ukládat do SQL databáze a předávání dat k možnosti zobrazení v rámci grafů.



Obrázek 26 Node-RED – Monitorování dat

Ukázka uložení dat v SQL databázi:

Data jsou ukládána do databáze *iot*, ve které se nachází tabulka s názvem *agro*.



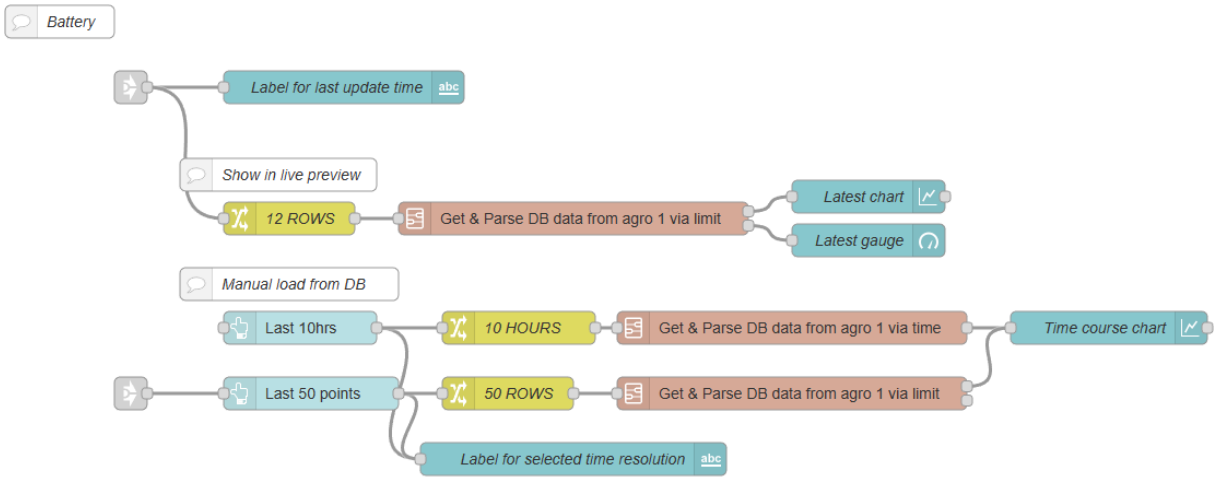
time	battery	temp_amb	temp_soil	moisture	humid
2023-04-17 22:53:14	100	23.4	22	100	43.9
2023-04-17 23:00:10	100	23.6	22.1	100	44.3
2023-04-17 23:15:10	100	23.4	22	100	42.9
2023-04-17 23:30:11	100	23.2	22	100	44.6
2023-04-17 23:45:12	100	23.2	22	100	45.9
2023-04-18 00:00:12	100	23	21.9	100	47.8
2023-04-18 00:15:13	100	22.9	21.8	100	47.3
2023-04-18 00:30:13	100	22.8	21.8	100	47.3
2023-04-18 00:45:14	100	22.7	21.8	100	47.7
2023-04-18 01:00:15	100	22.5	21.7	100	48.9
2023-04-18 01:15:15	99	22.5	21.6	100	48.9
2023-04-18 01:30:16	99	22.3	21.5	100	49.3
2023-04-18 01:45:16	99	22.4	21.5	100	50.6
2023-04-18 02:00:17	99	22.6	21.7	100	51.7
2023-04-18 02:15:18	99	22.8	21.8	100	51.5
2023-04-18 02:30:18	99	22.7	21.8	100	51.6
2023-04-18 02:45:19	99	22.7	21.8	100	52
2023-04-18 03:00:19	99	22.6	21.7	100	52.7
2023-04-18 03:15:20	99	22.6	21.7	100	53.4
2023-04-18 03:30:21	99	22.5	21.6	100	52.8
2023-04-18 03:45:21	99	22.4	21.6	100	53.9
2023-04-18 04:00:22	99	22.3	21.5	100	53.5
2023-04-18 04:15:22	99	22.3	21.5	100	53
2023-04-18 04:30:23	99	22.2	21.4	100	53.3
2023-04-18 04:45:24	99	22.2	21.4	100	53.8

Upravit Kopírovat Odstranit

Obrázek 27 Ukázka SQL databáze

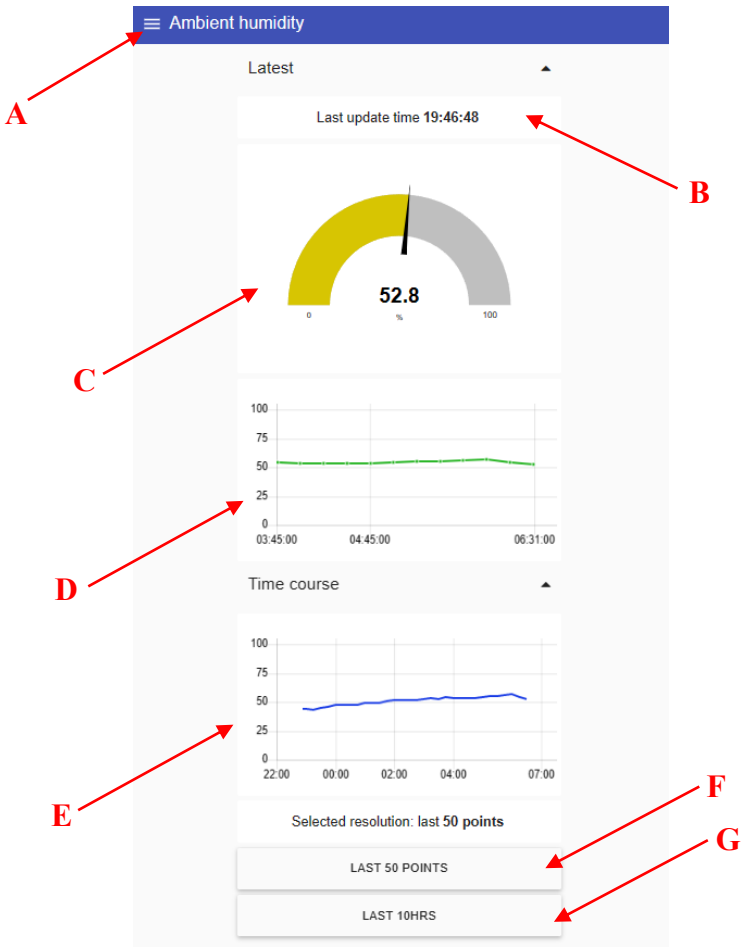
Flow – Nastavení pro grafické znázornění veličin:

Jako příklad nastavení, pro grafické znázornění, využiji zobrazení stavu baterie. Nastavení pro grafické znázornění ostatních veličin, je shodné.



Obrázek 28 Node-RED – Nastavení pro grafického znázornění stavu baterie

Popis grafického zobrazení:



Obrázek 29 Ukázka grafického zobrazení vlhkosti okolí

A – Možnost přepínání mezi jedntolivými okny s daty

B – Zobrazení posledního času aktualizace dat

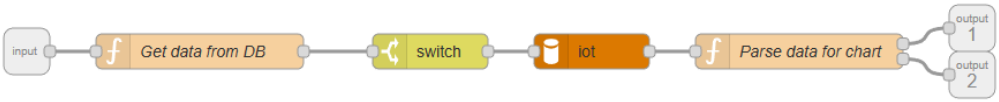
C – Zobrazení posledních dat v analogovém ukazateli

D – Zobrazení posledních dat v časovém grafu

E – Zobrazení posledních N bodů v závislosti na výběru tlačítek, při načtení je vybráno první

F a **G –** Tlačítka výběru zobrazení časových dat

Pro grafické znázornění se data nejprve získají z SQL databáze, dle nastavení. Následně jsou zformátována do kompatibilní syntaxe pro zobrazení v grafu. K tomu slouží vytvořený sub-flow „Get & Parse DB data from agro 1 via time“, případně „Get & Parse DB data from agro 1 via limit“. Výsledkem jsou zformátovaná data pro vykreslení v grafu a poslední časový naměřený bod, pro zobrazení v analogovém ukazateli



Obrázek 30 Node-RED – Formátování SQL dat k vykreslení grafem

Ukázka nastavení funkce „Parse data for chart“:

```
var payload = [];
var chart = [];
var label = msg.label == undefined ? msg.tag : msg.label;

if (msg.payload.length == 0) {
    chart = [{
        "series": [label],
        "data": [[{ "x": 0, "y": 0 }]],
        "labels": [label]
    }];
}
else
{
    for (let i = 0; i < msg.payload.length; i++) {
        let element = msg.payload[i];
        payload[i] = {
            "x": element.time,
            "y": element[msg.tag]
        };
    }
    msg.payload = payload;

    chart = [{
        "series": [label],
        "data": [msg.payload],
        "labels": [label]
    }];

}

msg.payload = chart;

return [msg, {payload:payload[0].y}];
```

3 KONSTRUKCE IOT BRÁNY

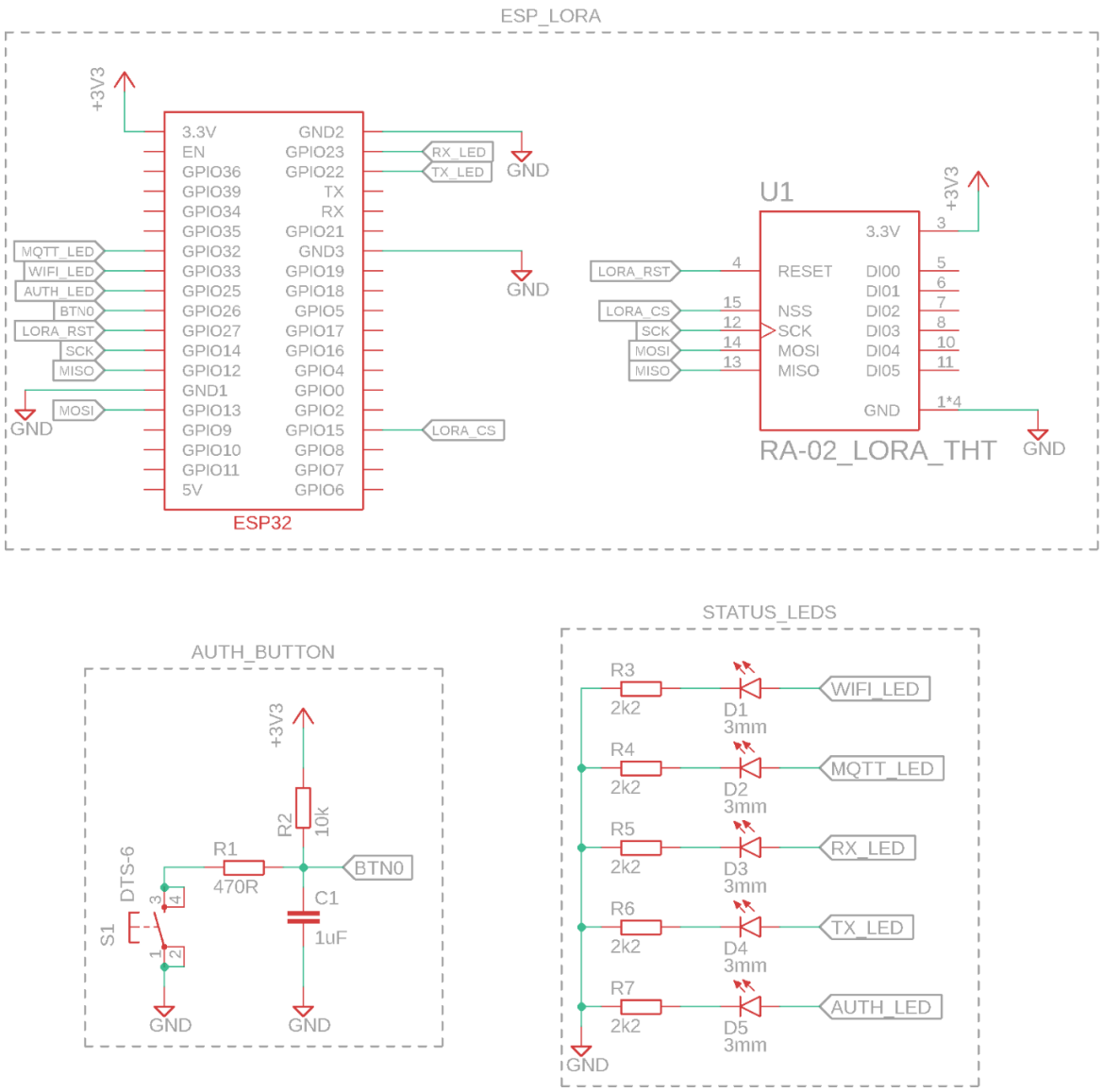
V této části proberu návrh a zhotovení konstrukční části IoT brány. Nejdříve se zaměřím na návrh PCB (desky plošných spojů) v návrhovém softwaru Eagle. Zhotovení a osazení desky plošných spojů vývodovými součástky a vybranými moduly, tj. LoRa a vývojový kit ESP32. Nakonec popíši tvorbu ochranného krytu.

3.1 Tvorba PCB

Aby bylo možné propojení jednotlivých součástí elektrického obvodu, je nutná tvorba PCB.

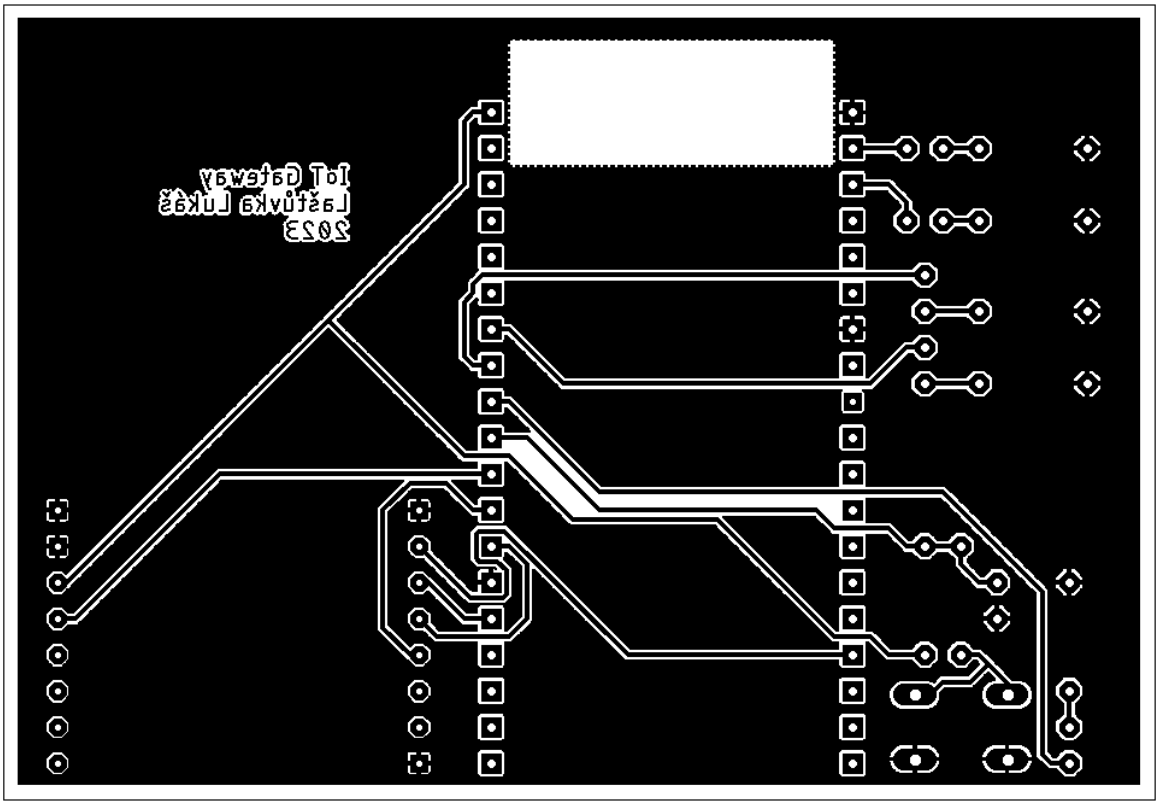
3.1.1 Návrh PCB

Pro návrh jsem použil software Eagle, do kterého jsem importoval knihovny pro LoRa modul a vývojový kit ESP32. Následně jsem provedl nákres schéma zapojení.

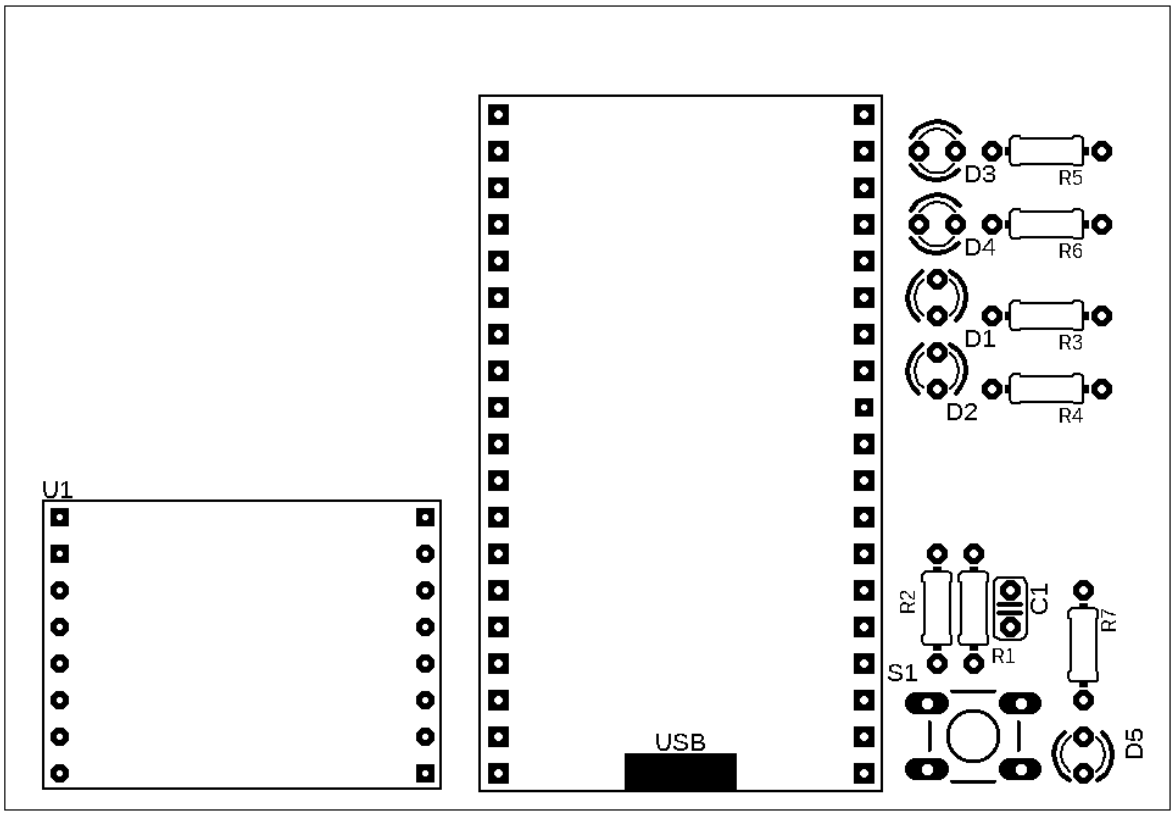


Obrázek 31 Schéma zapojení IoT brány

Jako další jsem rozmístil součástky, dle schéma zapojení, na návrhu desky plošných spojů.



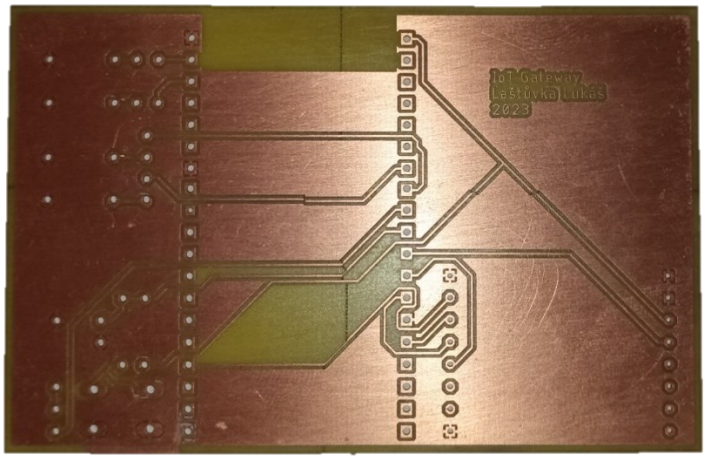
Obrázek 32 Návrh PCB IoT brány



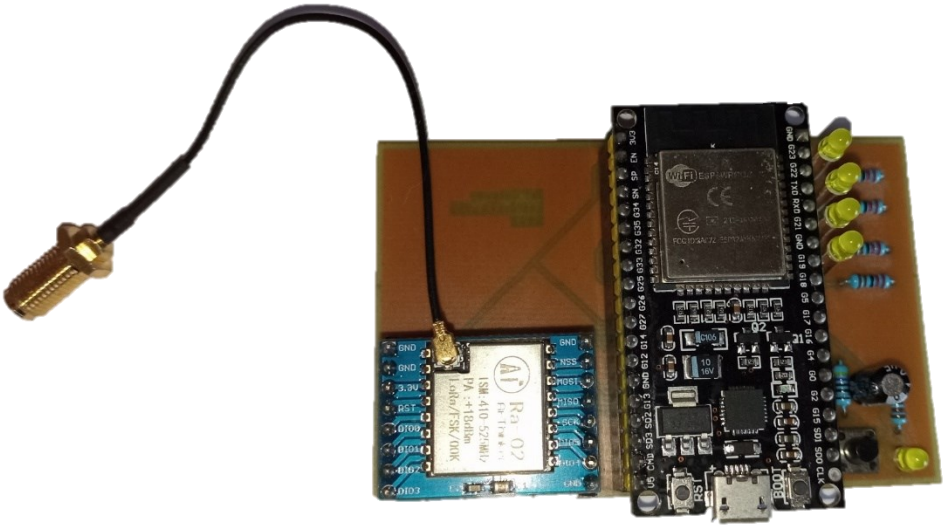
Obrázek 33 Rozmístění součástek na PCB Iot brány

3.1.2 Osazení PCB

Dle návrhu desky plošných spojů, jsem osadil příslušné součástky a moduly.



Obrázek 34 Neosazená PCB Iot brány



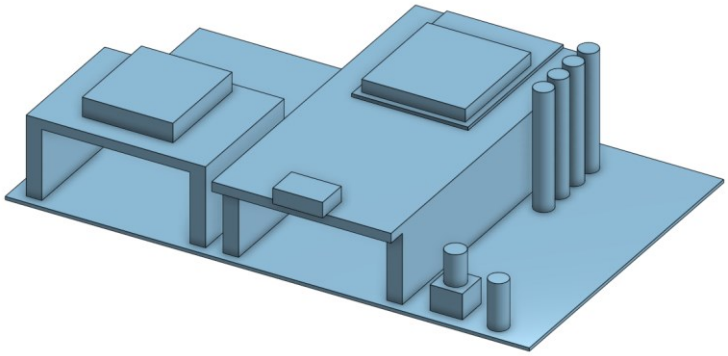
Obrázek 35 Osazená PCB IoT brány

3.2 Tvorba ochranného krytu

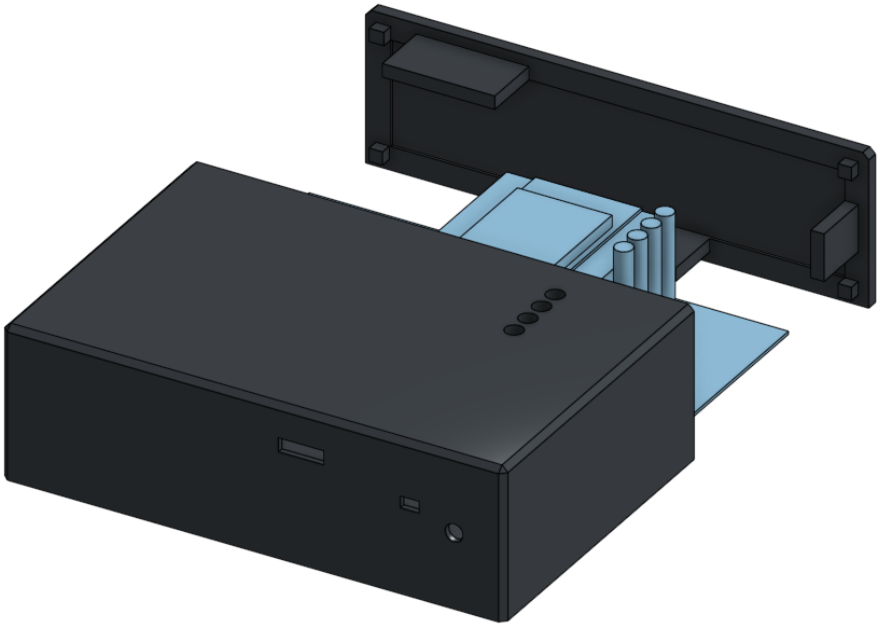
Posledním krokem konstrukce bylo vyhotovení ochranného krytu. K tomu jsem využil 3D modelovacího softwaru Onshape, ve kterém jsem vymodeloval pomocný model osazené desky v měřítku 1:1. Následně jsem vytvořil model krytu a vytisknul na 3D tiskárně.

3.2.1 Zakreslení 3D modelu krytu

Pro tvorbu modelu jsem použil software Onshape, který umožňuje tzv. parametrické modelování, což znamená, že je model zakreslen pomocí 2D náčrtů, kde je možné aplikovat omezení (vazby). Následně je možné vytvářet 3D prvky.



Obrázek 36 Pomocný 3D model osazené PCB IoT brány



Obrázek 37 3D model ochranného krytu IoT brány

3.2.2 **Výtisk 3D modelu**

Po 3D návrhu jsem připravil model k tisku. Použil jsem k tomu software Průša Slicer, který umí ze stereo-litografického modelu, vytvořit G-kód, který mohu použít pro 3D tiskárnu.

Obrázek 38 3D vytištěný model ochranného krytu IoT brány

ZÁVĚR

Hlavním cílem práce bylo navrhnout a sestrojit funkční IoT síť, kde bude využita jedna z technologií pro přenos dat (LoRa, ZigBee, Sigfox) s implementací IoT brány pro předávání dat mezi plnohodnotnou TCP/IP sítí.

Hlavním výběrem byla bezdrátová technologie LoRa spolu s TCP/IP protokolem MQTT. Celé řešení bylo vytvářeno na platformě ESP32, která má možnost využití bezdrátových sítí. Nabízí velice robustní a plnohodnotné vývojové prostředí ESP-IDF s implementací operačního systému FreeRTOS. Také nabízí možnost vytváření programu v jazyce C s možností tvorby vlastních knihoven. Dále je kompatibilní s vývojovým prostředím VSCode.

Při tvorbě byla zohledněna snadná implementace řešení do budoucích projektů využívajících IoT eko-systémů, proto byla práce psaná formou tutoriálu. Dále jsem vytvořil vlastní komunikační protokol NGP, využívající bezdrátové sítě LoRa pro přenos dat mezi IoT zařízením a IoT bránou. Samotný protokol jsem navrhl pro možnost autentizace zařízení v LoRa síti, snadného synchronizování času jednotlivých IoT zařízení a kompatibilním přenosem dat mezi IoT zařízením a MQTT brokerem díky vložení nastavení hlavičky (topics) a QoS dat.

Dále jsem přidal možnost dynamického nastavení přístupových údajů k Wi-Fi síti díky využívání Wi-Fi manažera. Také jsem přidal indikační LED diody a tlačítko, pro ovládání a monitorování stavů chování zařízení. Díky tlačítku lze zapínat/vypínat autorizační režim, resetovat nastavení Wi-Fi, případně provádět tovární obnovení. Díky indikačním LED diodám, lze vyhledávat případné chyby v chování IoT eko-systému.

V neposlední řadě jsem provedl možnost monitorování a ukládání dat díky Node-RED. V rámci něho lze graficky nastavit chování každého IoT zařízení. Zobrazovat data ve formě časových grafů a vytvářet interaktivní uživatelské aplikace.

Na konci práce jsem vytvořil ochranný kryt pro IoT bránu, který jsem následně vytiskl na 3D tiskárně.

V této práci je možné pokračovat dále, zejména pak v pokročilé implementaci NGP protokolu, kdy bude možné kontrolovat úspěšnost odeslání a příjmu dat. Rozšíření MQTT příkazů, kdy bude možné provádět kontrola stavu jednotlivých IoT zařízení a IoT brán. Díky snadnému a nenáročnému protokolu NGP, by zde byla možnost implementace do jiných bezdrátových IoT technologií a přidání pokročilé míry zabezpečení, např. šifrováním zpráv.

Tato práce splnila veškeré nastavené cíle a požadavky pro tvorbu IoT sítě.

POUŽITÁ LITERATURA

[1] RAMASAMY, Lakshmana a Seifedine KADRY. Internet of things (IoT). In: RAMASAMY, Lakshmana a Seifedine KADRY. Blockchain in *the Industrial Internet of Things [online]*. IOP Publishing, 2021 [cit. 2023-03-19]. ISBN 978-0-7503-3663-5. Dostupné z: doi:10.1088/978-0-7503-3663-5ch1

[2] VOJÁČEK, Antonín. Základní úvod do oblasti internetu věcí (IoT). In: *Automatizace.hw.cz* [online]. 2016 [cit. 2023-03-19]. Dostupné z: https://automatizace.hw.cz/zakladni-uvod-do-oblasti-internetu-veci-iot.html

[3] MEKKI, Kais, Eddy BAJIC, Frederic CHAXEL a Fernand MEYER. A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express* [online]. 2019, **5**(1), 1-7 [cit. 2023-03-19]. ISSN 24059595. Dostupné z: doi:10.1016/j.icte.2017.12.005

[4] Product Details SX1278 LoRa Connect. In: *Semtech* [online]. [cit. 2023-03-19]. Dostupné z: https://www.semtech.com/products/wireless-rf/lora-connect/sx1278

[5] Připojení k IoT síti LoRaWAN. In: *CRA* [online]. [cit. 2023-03-19]. Dostupné z: https://www.cra.cz/pripojeni-k-iot-siti-lorawan

[6] Connecting Things. In: SigFox [online]. [cit. 2023-03-19]. Dostupné z: https://sigfox.cz/cs

[7] TRAN, Hoang, Cuong NGUYEN, Nghia PHUNG a Minh NGUYEN. Mobile agents assisted data collection in wireless sensor networks utilizing ZigBee technology. Bulletin of Electrical Engineering and *Informatics [online]. 2023, 12(*2), 1127-1136 [cit**.** 2023-03-19]. ISSN 2302-9285. Dostupné z: doi:10.11591/eei.v12i2.4541

[8] SQL. In: Halvorsen [online]. [*cit*. 2023-03-19]. Dostupné z: https://www.halvorsen.blog/documents/tutorials/resources/Structured%20Query%20Language.pdf

[9] OO, Zaw, Theint LAI a Aung *MOE*. IoT Based Home Automation System using a REST API Architecture. European Journal of Technic [online]. [cit. 2023-03-19]*. ISSN 2536-5010. Dostupné z:* doi:10.36222/ejt**.**1018131

[10] AZZEDIN, Farag a Turki ALHAZMI. Secure *Data* Distribution Architecture in IoT Using MQTT. Applied Sciences [online]. 2023, 13(4) [cit. 2023-03-19]. ISSN 2076-3417. *Dostupné z: doi:10.3390*/app13042515

[11] MQTT. In: Axway [online]. [*cit*. 2023-03-19]. Dostupné z: https://blog.axway.com/wp-content/uploads/MQTT_1.png

[12] SX1278. In: LaskaKit [online]. [*cit*. 2023-03-24]. Dostupné z: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/orig/1754_ai-thinker-ra-02-sx1278-433mhz-lora-modul.jpg?61d95ca6

[13] ESP-WROOM-32 Full Pinout [online]. In: . [cit. 2023-03-24]. Dostupné z: https://cdn.shopify.com/s/files/1/0609/6011/2892/files/doc-esp32-pinout-reference-wroom-devkit.png?width=692

[14] SX1278 Datasheet. In: Semtech [online]. [cit. 2023-03-25]. Dostupné z: https://semtech.my.salesforce.com/sfc/p/E0000000JelG/a/2R0000001Rc1/QnUuV9TviODKUgt_rpBlPz.EZA_PNK7Rpi8HA5..Sbo

[15] Wi-Fi manager. In: GitHub [online]. [cit. 2023-03-26]. Dostupné z: https://github.com/tonyp7/esp32-wifi-manager.git

[16] Msgpack dokumentace. In: Msgpack-c C version *users* guide [online]. [cit. 2023-04-16]. Dostupné z: https://github.com/msgpack/msgpack-c/wiki/v2_0_c_overview

SEZNAM PŘÍLOH

PŘÍLOHA A

- Zdrojový kód projektu pro IoT bránu
- Zdrojový kód projektu pro IoT zařízení
- Soubor dat pro nastavení Node-RED
- Zdrojová data projektu tvorby DPS
- Data 3D modelu ochranného krytu