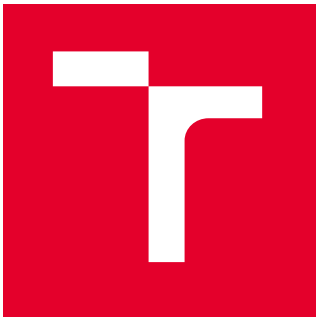


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

IOT LORAWAN SENSOR VLHKOSTI PŮDY A TEPLOTY

IOT LORAWAN SOIL MOISTURE AND TEMPERATURE SENSOR

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Peter Pánisz**

AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Vladislav Škorpil, CSc.**

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Peter Pánisz ***ID:*** 240964

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

IoT LoRaWAN sensor vlhkosti půdy a teploty

POKYNY PRO VYPRACOVÁNÍ:

Využijte modul HTTC-AB02A (Arduino kompatibilní) a navrhněte jednoduchý bateriově napájený sensor/měřič relativní vlhkosti půdy a teploty okolí pomocí tohoto modulu. Sensor bude napájen bateriově a připojen do LoraWAN sítě. Naměřená data bude odesílat do The Things Network nebo jiné vhodné LoRaWAN sítě, např. Helium a odtud pomocí MQTT protokolu se budou zapisovat naměřené hodnoty do vlastní cloudové služby sestávající z MQTT brokeru, NodeRed, InfluxDB a Grafana frameworku realizované na počítači Raspberry (popř. externí VPS) se zobrazením aktuálních hodnot a podrobným zpětným prohlížením hodnot a grafů.

DOPORUČENÁ LITERATURA:

[1] Baur, A.: Soil Moisture Tester for houseplants, Elektor magazine 4/2001. ISSN 1757-0875

[2] Claussen, Mathias: MQTT Sensor Hub, Elektor magazine 5/2019. ISSN 1757-0875

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: doc. Ing. Vladislav Škorpil, CSc.

Konzultant: Ing. Ondřej Pavelka

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

Abstrakt

Cieľom bakalárskej práce bolo vytvoriť softvér a navrhnúť ochranný kryt pre vývojovú dosku, ktorá meria vlhkosť pôdy, teplotu vzduchu a jej vlhkosť. Riešené bolo pripojenie do nízko energetickej bezdrôtovej siete pre zariadenia internetu vecí, cez ktorú boli namerané údaje odosielané. Prijaté dáta bolo potrebné spracovať a dekódovať cez rôzne aplikácie, ktoré si tiež vyžadujú konfiguráciu. Výsledky meraní je možné graficky vizualizovať v podobe grafov na verejnej webovej stránke, ktorá je prístupná cez internet. Pre administrátora je dostupná súkromná adresa s dodatočnými informáciami. V práci sú ešte popísané koncepty, ako sú kontajnerizácia aplikácií, výber vhodnej nízko energetickej siete alebo riešenie problémov spojených s dynamickou verejnou IPv4 adresou.

Kľúčové slová

IoT, LoRaWAN, Docker, MQTT, Node-RED, InfluxDB, Grafana, OpenHAB, Home Assistant

Abstract

The goal of this bachelor's thesis was to develop software and design a protective cover for a development board that measures soil moisture, air temperature, and humidity. The project addressed the connection to a low-power wireless network for Internet of Things devices, through which the measured data was transmitted. The received data needed to be processed and decoded through various applications, which also require configuration. The measurement results can be graphically visualized in the form of charts on a public website accessible via the internet. For the administrator, a private address with additional information is available. The thesis also describes concepts such as application containerization, the selection of a suitable low-power network, and solving issues related to a dynamic public IPv4 address.

Keywords

IoT, LoRaWAN, Docker, MQTT, Node-RED, InfluxDB, Grafana, OpenHAB, Home Assistant

Bibliografická citácia

PÁNISZ, Peter. IoT LoRaWAN sensor vlhkosti pôdy a teploty [online]. Brno, 2024 [cit. 2024-05-21]. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/159124. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Vladislav Škorpil.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Peter Pánisz
VUT ID autora:	240964
Typ práce:	Bakalárska práca
Akademický rok:	2023/2024
Téma bakalárskej práce:	IoT LoRaWAN sensor vlhkosti pôdy a teploty

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia S 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

V Brne dňa: 27. mája 2024 ----------

podpis autora

Poďakovanie

Chcel by som srdečne poďakovať pánovi doc. Ing. Vladislavovi Škorpilovi, CSc., vedúcemu mojej bakalárskej práce, a pánovi Ing. Ondřejovi Pavelkovi, konzultantovi z firmy Resideo Technologies, Inc., za ich odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

V Brne dňa: 27. mája 2024 ----------

podpis autora

Obsah

ZOZNAM OBRÁZKOV **9**

ZOZNAM TABULIEK **10**

ÚVOD **11**

1. TECHNOLÓGIE A HARDVÉR **12**

1.1 UNRAID 12
1.2 DOCKER 13
1.2.1 Docker Engine *13*
1.2.2 Docker Image *13*
1.2.3 Docker Container *13*
1.3 PROTOKOL MQTT 15
1.3.1 Mosquitto *16*
1.4 NODE-RED 16
1.5 INFLUXDB 17
1.6 GRAFANA 18
1.7 OPENHAB 19
1.8 HOME ASSISTANT 20
1.9 DUCKDNS A PORT FORWARDING 21
1.9.1 DuckDNS *21*
1.9.2 Port forwarding *21*
1.10 VÝVOJOVÁ DOSKA HELTEC CUBECELL AB02A 22
1.11 SENZORY 23
1.11.1 DHT-22 ..... *23*
1.11.2 Odporový senzor vlhkosti pôdy *24*
1.11.3 Kapacitný senzor vlhkosti pôdy *24*
1.12 ARDUINO IDE 25

2. BEZDRÔTOVÁ KOMUNIKÁCIA A JEJ NASADENIE **27**

2.1 LORA 27
2.2 LORAWAN 27
2.2.1 Klasifikácia LoRaWAN sietí *29*
2.2.2 Triedy uzlov *29*
2.3 VÝBER VHODNEJ LORAWAN SIETE 30

3. SOFTVÉROVÉ RIEŠENIE A TECHNICKÁ REALIZÁCIA **32**

3.1 INŠTALÁCIA A KONFIGURÁCIA ARDUINO IDE 32
3.2 OPERAČNÉ SYSTÉMY 33
3.2.1 Inštalácia OS Unraid *33*
3.2.2 Inštalácia IoT Stack *34*
3.3 KONFIGURÁCIA DOCKER KONTAJNEROV 34
3.3.1 Mosquitto MQTT *35*

3.3.2 InfluxDB 35
3.3.3 Node-RED 35
3.3.4 Grafana 37
3.3.5 OpenHAB 39
3.3.6 Home Assistant 41
3.3.7 DuckDNS 43
3.3.8 Port Forwarding 43
3.1 NÁVRH A REALIZÁCIA 44
3.1.1 Konštrukcia 44
3.1.2 Zapojenie 45
3.2 VÝVOJ A ANALÝZA SYSTÉMU 45
3.2.1 Zdrojový kód 45
3.2.2 Transformácia dát 46
3.2.3 Overenie správnej funkčnosti 48
3.2.4 Porovnanie platforiem Home Assistant, OpenHAB a Grafana 49
ZÁVER 51
LITERATÚRA 52
ZOZNAM SYMBOLOV A SKRATIEK 55
ZOZNAM PRÍLOH 57

ZOZNAM OBRÁZKOV

1.1 Porovnanie kontajnerov a virtuálnych strojov 14
1.2 Schéma princípu fungovania MQTT protokolu[6] 16
1.3 Pinout diagram použitej vývojovej dosky 23
2.1 Schéma princípu fungovania siete LoRaWAN 28
2.2 Pokrytie sieťou TTN v Brne[33] 30
2.3 Pokrytie sieťou Helium v Brne[34] 31
3.1 Schéma projektu 32
3.2 Nakonfigurovaný Flow v Node-RED 37
3.3 Blokový diagram Blockly 41
3.4 Výsledok presmerovania portov na TP-Link routeri 44
3.5 Výsledný Flow v Helium konzole 48
3.6 Zobrazenie verejných dát zo senzorov 48
3.7 Mapa pokrytia hotspotmi 49

ZOZNAM TABULIEK

Tabuľka 1.1 - Rozdiely medzi vývojovými doskami Heltec CubeCell 22
Tabuľka 1.2 - Špecifikácie senzoru DHT-22 24

Úvod

Moderné technológie sa neustále snažia ľudstvu zlepšiť a zjednodušiť život. Táto bakalárska práca sa zameriava na IoT LoRaWAN sensor vlhkosti pôdy, teploty a vlhkosti vzduchu, ktorý by sa v budúcnosti mohol stať bežnou súčasťou v poľnohospodárstve a napríklad aj v domácich záhradách.

Cieľom bakalárskej práce je konfigurácia vývojovej dosky Heltec CubeCell HTCC-AB02A, výber vhodnej IoT siete pre pripojenie, zaznamenanie nameraných dát zo senzorov a následná prezentácia užívateľom. Namerané údaje zo senzorov ohľadom vlhkosti pôdy, teploty vzduchu, vlhkosti vzduchu a napätia batérie budú posielané do vhodne vybranej LoRaWAN siete. Odtiaľ budú dáta pomocou MQTT protokolu zaznamenávané do vlastnej cloudovej služby vytvorenej z MQTT brokeru, NodeRed vizuálneho nástroja pre automatizáciu, InfluxDB databázy a Grafana frameworku pre zobrazenie nameraných dát.

Bežný používateľ bude schopný zobraziť základné namerané dáta na verejnej webovej adrese, administrátor po zadaní loginu uvidí aj podrobnejšie, obvykle skryté údaje. Aby sa predišlo vybitiu batérie vo vývojovej doske, v prípade, že napätie integrovanej 3 V batérie klesne pod stanovenú hranicu, bude automaticky odoslaná notifikácia na Telegram cez platformu OpenHAB. Taktiež bude možnosť nastavenia notifikácií v prípade prekročenia hranice určitých údajov. Nakoniec budú porovnané platformy domácej automatizácie Home Assistant a OpenHAB.

Dokument je rozdelený do troch hlavných kapitol. Prvá kapitola popisuje výber použitých technológií, podrobné informácie o fungovaní rozličných systémov a vysvetľuje niektoré kľúčové teoretické koncepty použité v tomto projekte ako aj princípy virtualizácie. Tiež je vysvetlená problematika použitých senzorov a opísané sú špecifikácie použitej vývojovej dosky. Druhá kapitola vysvetľuje aplikovaný bezdrôtový komunikačný systém a jeho nasadenie. Vysvetlená je modulačná technika LoRa, protokol LPWAN MAC a princíp fungovania LoRaWAN. Taktiež sú rozpísané ďalšie dôležité časti, ako klasifikácia sietí, triedy uzlov, porovnanie a výber medzi sieťami Helium a The Things Network. V poslednej kapitole je do podrobnosti rozpísaná konfigurácia všetkých softvérových technológií, vysvetlené sú hlavné časti zdrojového kódu a popis fungovania prepojení medzi jednotlivými softvérovými technológiami, bežiace vo virtualizovanom prostredí. Ďalej sú porovnané platformy Home Assistant, OpenHAB a Grana, každá s iným účelom v tejto práci. Na záver je popísaný návrh, realizácia zapojenia senzorov a konštrukcia krytu pre použitú vývojovú dosku.

1. TECHNOLÓGIE A HARDVÉR

Prvá kapitola sa zaoberá všetkými technológiami, ktoré boli použité v realizácii tohto projektu. Každá technológia je podrobne vysvetlená, popísaná jej implementácia a odôvodnený jej výber. Rovnako je opísaný výber vývojovej dosky, rozdiely oproti iným vývojovým doskám od rovnakého výrobcu a voľba senzorov.

1.1 Unraid

Unraid je proprietárny Linuxový operačný systém vyvinutý spoločnosťou Lime Technology, Inc. Je navrhnutý aby bežal na domácich serveroch ako sieťové úložisko (NAS), aplikačný server alebo virtualizačný hosť. Jeho výhodou je možnosť prispôsobenia webového používateľského rozhrania, inštalácie aplikácií a pluginov tretej strany a taktiež používanie ľubovoľných Docker kontajnerov alebo virtuálnych strojov (VM)[1].

Unraid podporuje širokú škálu typov úložísk, vrátane pevných diskov (HDD), SSD ale aj vymeniteľných médií ako sú USB kľúče. Bootovanie je vyriešené trochu neštandardným spôsobom, z USB kľúča. Je to kvôli tomu, že ak by bol USB kľúč nejakým spôsobom poškodený, dáta ktoré sa nachádzajú na diskoch by boli nedotknuté a ich obnova by bola jednoduchá.

Ako každé sieťové úložisko, aj tu sa nachádza podpora technológie paritných diskov RAID. Medzi štandardne podporované súborové systémy patria XFS, Btrfs a po novom aj ZFS. Oproti konkurenčnému softvéru, ako napríklad TrueNAS, je možnosť používania rôznych veľkostí diskov v prípade využitia parity so súborovými systémami XFS a Btrfs. Platí však, že paritný disk musí byť vždy rovnako veľký alebo väčší ako najväčší disk v poli.

Ďalšie konkurenčné softvéry vhodné spomenúť sú VMware ESXi alebo Synology DSM. Ako jednu nevýhodu môžem vytknúť, že štandardná licencia nie je zadarmo a na jedno zariadenie so 6 diskami stojí 59$. Je tu však možnosť vyskúšať skúšobnú verziu na 30 dní zadarmo.

Výber tohto OS bol ovplyvnený tým, že som s ním mal dostatok znalostí ešte pred použitím v tejto práci a spĺňal všetky požiadavky na realizáciu tohto projektu. Nie je to však jediné možné riešenie. Jedna z možností, ktorá je zadarmo, je použitie otvoreného projektu IoTstack, ktorý je perfektne robený na implementáciu IoT zariadení na vzdialených VPS alebo vývojových doskách ako Raspberry Pi[2].

1.2 Docker

Docker je open-source platforma napísaná v programovacom jazyku Go, ktorá umožňuje vývojárom vytvárať, nasadzovať a spúšťať aplikácie v izolovanom prostredí, nazývanom kontajner. Toto izolované prostredie zvyšuje bezpečnosť a umožňuje beh viacerých kontajnerov naraz na jednom zariadení. Kontajnery sú nenáročné na hardvér, obsahujú všetky závislosti potrebné na beh aplikácie, a umožňujú jednotlivým aplikáciám bežať konzistentne bez ohľadu na prostredie, v ktorom sú spustené.

Docker poskytuje jednotný spôsob zabalenia, distribúcie a spustenia aplikácií, čím zjednodušuje proces vývoja, testovania a nasadzovania softvéru. Je široko využívaný v oblasti kontajnerizácie a kontajnerovej virtualizácie. Výhodou je, že kontajner, ktorý je vytvorený na jednom zariadení, bude fungovať na iných zariadeniach rovnako[3].

1.2.1 Docker Engine

Docker Engine je jadrom celej Docker platformy. Je to open-source aplikácia, ktorá beží na rôznych Linuxových distribúciách (Ubuntu, Debian, Fedora, CentOS...) a operačnom systéme Windows Server. Hlavným procesom je kontajnerový daemon, nazývaný dockerd, ktorý spravuje vytváranie, spúšťanie a konfiguráciu kontajnerov.

1.2.2 Docker Image

Docker Image (obraz) používa Docker Engine na tvorbu kontajnerov. Obraz je šablóna, ktorá obsahuje kód aplikácie, závislosti potrebné na beh tejto aplikácie v izolovanom prostredí a jej konfiguráciu. Slúži na vytvorenie Docker kontajnera, ktorý je určený len na čítanie. Často vychádza z iného obrazu s nejakými dodatočnými úpravami. Napríklad môžete vytvoriť obraz založený na obraze Raspbian, kde sa nainštaluje Apache web server a Vašu aplikáciu, ako aj konfiguračné údaje potrebné na spustenie Vašej aplikácie.

Na využitie je možnosť tvorby vlastných obrazov alebo používať už vytvorené. Na vytvorenie vlastného obrazu je potrebné vytvoriť Dockerfile s jednoduchými krokmi na vytvorenie a spustenie obrazu. Každá inštrukcia v Dockerfile vytvára vrstvu v obraze. Keď upravíte Dockerfile a znovu vytvoríte obraz, len tie vrstvy, pri ktorých nastala zmena, sú zmenené. Toto je to, čo robí obrazy také nenáročné, malé a rýchle v porovnaní s inými technológiami virtualizácie.

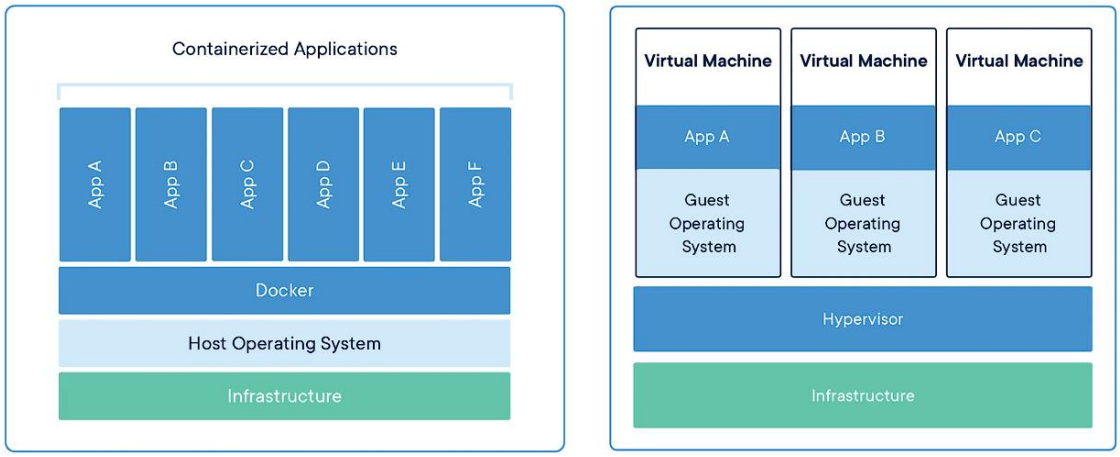
1.2.3 Docker Container

Docker kontajner je štandardizovaný kus softvéru, ktorý v sebe zahŕňa všetok kód, knižnice a závislosti na spustenie kontajnerizovanej aplikácie. V prípade Docker

kontajnerov sa obrazy stanú kontajnermi v prípade, že bežia na Docker Engine. Tým, že sú izolované od prostredia, v ktorom bežia, je zabezpečené to, že je jedno v akom prostredí je kontajner spustený, stále bude všade fungovať rovnako[4].

Porovnanie kontajnerov a virtuálnych strojov:

- Kontajnery a virtuálne stroje majú podobné benefity v izolácii a alokácii zdrojov, ale fungujú rozdielne kvôli tomu, že kontajnery virtualizujú operačný systém a nie hardvér. Vďaka týmto vlastnostiam sa kontajnery oveľa rýchlejšie zapnú z vypnutého stavu ako virtuálny stroj, a aktualizácia aplikácií prebieha tiež rýchlejšie.



Obrázok 1.1 Porovnanie kontajnerov a virtuálnych strojov

- Kontajnery sú abstrakcia aplikačnej vrstvy, ktorá zabaľuje spolu kód, knižnice a závislosti. Na tom istom stroji môže bežať viacero kontajnerov naraz a zdieľať jadro operačného systému s ostatnými kontajnermi. Napriek tomu každý funguje ako vlastný izolovaný proces v používateľskom prostredí. Veľkosť kontajnerov je menšia ako veľkosť virtuálnych strojov (pár desiatok MB oproti stovkám MB). Tým pádom je možné spúšťať naraz viacero kontajnerových aplikácií ako virtuálnych strojov s operačnými systémami.
- Virtuálne stroje (VMs) sú abstrakcia fyzického hardvéru, ktorá premieňa jeden server na viacero. Hypervízor dovoľuje beh viacerých virtuálnych strojov naraz na jednom fyzickom stroji. Každý virtuálny stroj obsahuje celý operačný systém, aplikácie, potrebné knižnice a závislosti, ktoré zaberajú stovky MB až GB.

Vyššie spomenuté vlastnosti Dockeru boli veľmi atraktívne, takže namiesto inštalácie každej technológie manuálne, napríklad v Linuxovom operačnom systéme, som sa rozhodol používať technológiu Docker kontajnerov, ktorá je už vstavaná v operačnom systéme Unraid.

1.3 Protokol MQTT

MQTT - Message Queuing Telemetry Transport je protokol určený na použitie pre IoT zariadenia v sieťach s veľmi nízkou prenosovou rýchlosťou a extrémne vysokou latenciou. Keďže je špecializovaný pre prostredia s nízkym prenosovým pásmom a vysokou latenciou, je ideálnym protokolom pre komunikáciu medzi machine-to-machine (M2M)[5].

MQTT funguje na princípe subscriber / publisher (odoberateľ / vydavateľ) a je ovládaný centrálnym sprostredkovateľom (broker). Toto znamená, že odosielateľ a prijímateľ nemajú priame spojenie, ale informácie medzi nimi im predáva broker. Odoberateľ sa prihlási na tú istú tému, nazývanú topic, kde vydavateľ odošle správu. Toto riešenie môže obsahovať viacero vydavateľov alebo odoberateľov, a môžu byť prihlásené na rôzne témy. Výhoda tejto technológie je, že vydavateľ a odoberateľ nemusia byť v tom istom čase naraz pripojený, čo drasticky znižuje energetické nároky.

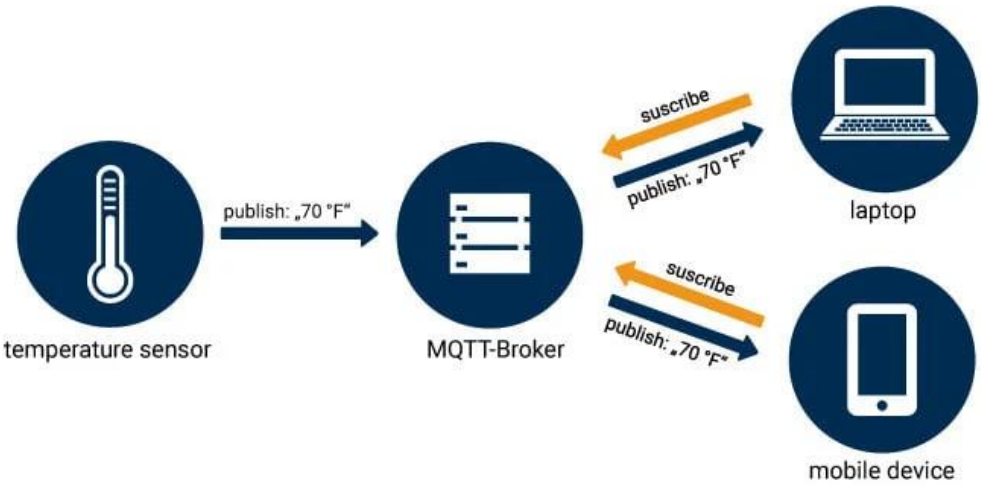
MQTT podporuje 3 typy kvality služieb (QOS) pre zaistenie doručenia správ:

- **QoS 0 (At most once) -** Nezabezpečuje žiadne doručenie správy. Ak je správa len jedenkrát poslaná a nedostane sa do destinácie, bude zahodená, a jej opätovné poslanie nenastane.
- **QOS 1 (At least once) -** Zabezpečuje doručenie správy tak, že správa bude doručená aspoň raz. Ak odberateľ ale nezíska potvrdenie doručenia, vydavateľ môže poslať správu znovu. Toto síce zabezpečuje doručenie správy, ale môže spôsobiť zdvojenie.
- **QOS 2 (Exactly once) -** Zabezpečuje doručenie správy presne raz vzájomným potvrdením. V prípade chyby je správa opakovane odoslaná.

Pre zabezpečenie komunikácie šifrovaním je možné použiť šifrovací protokol TLS alebo SSL. Taktiež sú podporované ďalšie bezpečnostné prvky ako autorizácia, autentifikácie a účtovníctvo.

Na prenos užitočných dát sa v protokole MQTT používa tzv. payload. Payload predstavuje obsah samotnej správy, ktorý môže byť v rôznych formách ako text, binárne dáta, JSON reťazec, alebo iné. Pred odoslaním payloadu je potrebné určiť jeho veľkosť a transformovať dáta do určitého formátu.

Tento transportný protokol bol ideálny na použitie pre tento projekt keďže je nízkoenergetický, odosielané množstvo dát je malé, nezáleží na latencii a je jednoduchý na implementáciu.



Obrázok 1.2 Schéma princípu fungovania MQTT protokolu[6]

1.3.1 Mosquitto

Mosquitto je open source MQTT broker implementujúci MQTT protokol od neziskovej organizácie Eclipse Foundation. Je populárny vďaka jednoduchej implementácii. Tým sa stal ideálnym kandidátom pre používateľov, ktorý chcú jednoducho, rýchlo a spoľahlivo spojazdniť MQTT IoT systém pre zbieranie a odosielanie dát. Napísaný je v jazyku C a podporuje širokú škálu operačných systémov od rôznych Linuxových distribúcií až po Windows. Určený je od nízkoenergetických IoT zariadení až po vysoko výkonné servery. Verzie podporovaných protokolov MQTT sú 3.1., 3.1.1 a 5.0. Mosquitto taktiež poskytuje populárne klientske služby mosquitto_pub a mosquitto_sub v príkazovom riadku, ktoré sú určené na rýchle otestovanie správnej funkčnosti a riešenie prípadných problémov[7].

1.4 Node-RED

Node-RED je open source vývojové prostredie založené na Node.js a JavaScript. Vytvorené bolo inžiniermi z IBM and je najviacej vhodné pre vývoj IoT systémov. Podrobnejšie povedané, je to procesovo založené virtuálne programovacie prostredie, ktoré vytvára dátové toky zo senzorov ku cloudu prepojením hardvérových

a softvérových komponentov. Je vhodné na záznam dát a tým pádom uľahčuje spracovanie dát. Je možné ho použiť na zostavenie spracovávania údajov a presun spracovaných dát na systémy vyššej vrstvy (SQL server, podnikový spravovací systém, ústredný dátový server alebo cloudová služba) už v priebehu pár minút alebo zobraziť ich v reálnom čase. Namiesto potreby naprogramovať webového rozhranie, ktoré by zobrazovalo údaje z rôznych zdrojov alebo senzorov, Node-RED poskytuje Dashboard používateľské rozhranie, umožňujúce vytvoriť či už jednoduché alebo komplexné rozhranie bez potreby programovacích znalostí v jazykoch ako HTML, CSS a JS.

Node-RED obsahuje rôzne uzly a série uzlov, ktoré umožňujú rôzne funkcie ako debugovanie a logovanie kódu, tvorba vlastných funkcií, TCP, MQTT, Websocket a iné sieťové služby, zápis a čítanie z rôznych zdrojov, formátovanie údajov a s pomocou inštalácie pluginov aj ďalšie možnosti[8].

Toto vývojové prostredie zjednodušuje transformáciu údajov v tomto projekte. Prijaté údaje z MQTT brokeru sú vyselektované, transformované do vhodného formátu pre uloženie a následne uložené databázy. Jednoduchým zobrazením logovania môžeme odhaliť prípadné chyby v našej implementácii.

1.5 InfluxDB

InfluxDB je open-source časovo závislá databáza (TSDB) vytvorená spoločnosťou InfluxData špeciálne navrhnutá na ukladanie, dotazovanie a správu časovo závislých dát ako sú monitoring, analýza dát v reálnom čase a IoT senzorové dáta. Väčšina kódu je napísaná v jazyku Go a dizajnovaná na vysoký výkon pri práci s dátami a efektivitu pri ukladaní dát. Každú sekundu dokáže uložiť tisíce dátových údajov, vďaka čomu je ideálna pre industriálne využitie[9].

Časovo závislá databáza je druh databázy, ktorá ukladá dáta vyvíjajúce sa postupom času. Typický príklad môže byť meranie teploty každých 5 minút, napríklad. Tento druh databázy zoberie presný čas, ktorý je synchronizovaný s NTP serverom, a priradí túto časovú pečiatku ku každej jednej nameranej hodnote.

Rozdiel medzi relačnými databázami, ako je MySQL, a časovo závislými databázami je v tom, že relačná databáza nevytvára vzťahy medzi dátami. Je perfektná na ukladanie údajov ako sú používateľské mená, ID a správy používateľov. V tomto type databázy sa vytvorí vzťah medzi používateľom a správou, aby používateľ mohol požiadať o určité dáta, povedzme.

Iný druh databázy, známy ako NoSQL, ako napríklad MongoDB alebo OpenSearch, uskladňuje dáta vo forme dokumentov, kde vzťah medzi informáciami zahŕňa rôzne údaje, ako sú používateľské mená, telefónne čísla, e-maily a ďalšie. Databázy typu

NoSQL sú konkurenciou s relačných databáz SQL a nie sú optimalizované pre ukladanie metrikových údajov, ako je to v prípade InfluxDB.

Z tohto textu jasne vyplýva, že časovo závislá databáza presne vyhovuje našim požiadavkám. Namerané dáta chceme uložiť do databázy, analyzovať ich a následne graficky vyobraziť.

1.6 Grafana

Grafana je open source interaktívna webová analytická platforma slúžiaca na vizualizáciu a analýzu dát. Je navrhnutá na tvorbu interaktívnych dashboardov, do ktorých je možné vložiť rôzne typy grafov a panelov. Vzhľad grafov, panelov a dashboardov sa dá upravovať viacerými spôsobmi a je možné ich zverejniť na verejnej webovej adrese, kde si ich môžu rôzny používatelia prezerať a analyzovať zverejnené dáta. Grafana podporuje široké množstvo zdrojov odkiaľ dokáže čerpať dáta. Najpoužívanejšie sú databázy MySQL, InfluxDB, Prometheus a ďalšie[10].

V súčasnosti je Grafana najpopulárnejší vizualizačný webový softvér, s viacej ako 1000 spoločností využívajúcich Grafanu. Jej obľúbenosti pomáha aj podpora rôznych pluginov.

Monitorovanie správneho chodu aplikácií je v dnešnej dobe veľmi dôležité, a preto ponúka Grafana možnosť nastaviť si rôzne upozornenia a notifikácie. Napríklad, ak sa na databázovom serveri prekročí využitie úložiska nad 80%, je možné administrátorom odoslať notifikáciu na telefón, email alebo ďalšími spôsobmi.

Na nastavenie notifikácií je potrebné vytvoriť si pravidlá. Pravidlá slúžia ako spúšťač pre vytvorené upozornenia. Využitie pravidiel má ale veľa účelov, a ďalšie z nich môže slúžiť napríklad na automatické vytvorenie nového dashboardu pri detekcii nového zariadenia v databáze.

Grafana má taktiež veľmi dobrú podporu filtrovania rôznych objektov. Je možné filtrovať minimálne, maximálne, priemerné a ďalšie hodnoty. Grafana Cloud je natívna cloudová služba, ktorá funguje ako Software-as-a-Service. Je určená pre tých, ktorí nechcú hostovať inštanciu Grafany na vlastnom hardvéri, ale prenechať túto starosť na iných. Je vhodné ešte spomenúť že je tu aj podpora oprávnení pre rôznych používateľov a skupiny. Napríklad administrátor môže vidieť všetky položky na dashboarde, ľudia z marketingového oddelenia vidia položky potrebné len pre ich prácu a HR oddelenie vidí zase iné. Možnosť zverejniť dashboardy na verejnej webovej adrese je tiež k dispozícii.

Na jednoduché a prehľadné zobrazenie nameraných dát zo senzorov v tomto projekte je táto platforma ideálna. Tiež bude využitá možnosť verejných dashboardov,

kde verejnosť uvidí základné dáta ako teplota a vlhkosť, administrátor uvidí viac podrobné informácie, ako napätie batérie, používaný kanál vývojovou doskou, silu signálu a ďalšie.

1.7 OpenHAB

OpenHAB (Open Home Automation Bus) je open-source platforma zameraná na automatizáciu domácnosti, napísaná v programovacom jazyku Java. Vznikla v roku 2010 pod záštitou neziskovej organizácie Eclipse Foundation. Táto platforma sa opiera o framework Eclipse SmartHome a je zároveň aktívnym projektom pre vývoj bindings (väzieb).
Vytvorená je na monitorovanie, správu a riadenie inteligentných zariadení ako sú svetlá, senzory, domáce spotrebiče a iné. Cieľom je prepojiť rôzne druhy inteligentných zariadení od rôznych výrobcov na jednej platforme a zužitkovať ich informácie na automatizáciu. Podporované sú rôzne protokoly ako Bluetooth, WiFi, Zigbee, Z-Wave, MQTT a ďalšie.

Tým, že je platforma napísaná v programovacom jazyku Java, je jednoduché ju spustiť na širokej škále operačných systémov, vrátane Linux, Windows a MacOS. Na výber je tu aj verzia pre Docker kontajnery alebo upravený Linuxový klon s názvom OpenHABian, ktorý je kompletným obrazom disku.

Rozhranie je možné zobraziť vo webovom prehliadači, ale pre niektoré operačné systémy je možné použiť aj samostatnú aplikáciu. Po úvodnom nastavení sa Vám zobrazí domovská obrazovka s rôznymi položkami. Prvou z nich je **Things** (veci). Veci sú entity, ktoré môžu byť pridané do systému fyzicky, ale taktiež môžu reprezentovať nejakú webovú službu alebo iný zdroj informácií. Veci nemusia poskytovať iba jednu jedinú službu, môžu aj viacero. Napríklad jeden senzor môže merať teplotu a vlhkosť vzduchu zároveň. **Channels** (kanály) používajú schopnosti vecí. Ak máte inteligentnú zástrčku, ktorá meria frekvenciu, napätie, prúd a výkon, môžete si vytvoriť kanál iba pre výkon a napätie. Nemusíte použiť všetky schopnosti Vášho zariadenia, ktoré poskytuje vec. **Bindings** (prepojenia) sú softvérové prepojenia, ktoré umožňujú pripojiť zariadenia do Vášho domáceho automatizačného systému. Niektoré sú už vstavané do OpenHAB, iné je možné nainštalovať dodatočne. Jeden druh údajov môžete dostávať cez MQTT prepojenie a ďalší druh údajov napríklad z databázového InfluxDB prepojenia. **Items** (položky) reprezentujú schopnosti, ktoré môžu byť používané aplikáciami buď pre vykreslenie v používateľskom rozhraní alebo aj pre pravidlá automatizácie. Majú pridelený State (stav) a sú schopné prijímať príkazy. To, čo spája veci a položky, sa nazýva **Link** (väzba). Väzba je prepojenie medzi jedným

kanálom a jednou vecou. V prípade, že je kanál prepojený s položkou, je schopnosť položky, ktorú reprezentuje, prístupná cez kanál. Kanály môžu byť prepojené s viacerými položkami a položky môžu byť prepojené s viacerými kanálmi[11].

1.8 Home Assistant

Home Assistant (HA) je softvér na automatizáciu domácnosti, ktorý umožňuje vytvoriť jednotný riadiaci systém pre širokú škálu inteligentných domácich zariadení od rôznych výrobcov. Softvér je bezplatný na používanie, je open-source a neustále sa vyvíja, a to nielen hlavným vývojárskym tímom, ale aj komunitou Home Assistant.

Rieši jeden z hlavných problémov inteligentných zariadení – fragmentáciu. Predstavte si situáciu, kde máte robotický vysávač od jedného výrobcu, inteligentné LED svetlá od iného výrobcu a inteligentný termostat od ďalšieho výrobcu a chcete vytvoriť scénu na odchod z domu. Chcete, aby sa vypli svetlá, zapol sa robotický vysávač a teplota na termostate sa nastavila na 20,5 °C. Každý z týchto výrobcov má svoju vlastnú aplikáciu, ktorá podporuje rôzne funkcie. Bez Home Assistantu by bolo veľmi obtiažne zosúladiť tieto zariadenia a vytvoriť jednotnú scénu v jednej domácnosti. Ďalšia výhoda je podpora pre funkcie, ktoré výrobca v ich originálnom softvéri nepodporuje, alebo sú skryté len pre vyššie modely.

V momente, keď sa na trh uvedie nové inteligentné zariadenie, niektorí skúsenejší členovia komunity, ktorí si zakúpili toto zariadenie, napíšu kód pre jeho integráciu do Home Assistantu. Postupne sa takto schvaľujú a pridávajú nové zariadenia a tým sa rozširuje ich podpora.

Home Assistant nie je jediná možnosť. Existuje vyššie spomínaný OpenHAB, ktorý podporuje veľa rovnakých funkcií ako HA. Rozdiel je v tom, že zatiaľ čo komunita OpenHAB má len 6 000 členov, komunita HA má viac ako 240 000 členov. Takže v prípade, že máte nejaký problém s integráciou alebo otázku na nejaké inteligentné zariadenie, je väčšia šanca, že komunita HA Vám bude vedieť pomôcť[12; 13].

Spustiť Home Assistant je možné na veľkom množstve zariadení, či už na Raspberry Pi, x86 PC alebo na zariadení Home Assistant Green priamo od výrobcu, ktoré má HA priamo predinštalovaný a stačí ho len zapojiť. Podporované sú všetky bežne používané operačné systémy ako Windows, MacOS a rôzne Linuxové distribúcie. Ďalej sa Home Assistant rozdeľuje do štyroch rôznych inštalačných metód:

- **Home Assistant OS (HA OS)** – Plnohodnotný OS založený na Linuxe, špeciálne navrhnutý pre používanie HA, ideálny pre menej skúsených používateľov
- **Container** – Pokročilejší spôsob inštalácie pre používateľov, ktorí majú skúsenosti s kontajnerizáciou, umožňuje väčšiu možnosť konfigurácie

- **Core** – Najzákladnejšia forma HA, obsahuje iba softvér HA bez ďalších prídavných pluginov
- **Supervised** – Podobná ako HA Container ale nevyužíva kontajnerizáciu

Využitie softvéru Home Assistant je ideálne pre tento projekt, keďže podporuje jednoduchú integráciu a zobrazenie nameraných dát zo senzorov. Taktiež nám umožňuje porovnať rôzny prístup k jednotlivým funkciám medzi Home Assistant a OpenHAB.

1.9 DuckDNS a Port forwarding

1.9.1 DuckDNS

DuckDNS je bezplatná služba, ktorá umožňuje automatickú aktualizáciu dynamickej doménovej adresy (DDNS). Služba beží na virtuálnom privátnom cloude od Amazonu a podporuje len IPv4 adresy. Spojazdnenie je veľmi jednoduché a ideálna na domáce používanie, keď sa chcete z cudzej siete pripojiť do Vašej domácej siete bez znalosti Vašej verejnej domácej IPv4 adresy. Stačí si vytvoriť účet, vygenerovať doménu a uložiť si token, ktorý potrebujete na aktualizáciu IP adresy. Službu je možné spustiť na obrovskom množstve zariadení. Podporované sú všetky bežné operačné systémy ako Linux, Windows, MacOS ale aj operačné systémy určené pre routery ako sú OpenWrt, DD-WRT, pfSense a ďalšie. IPv4 adresu je možné aktualizovať použitím pár riadkového skriptu, použitím Docker kontajnera alebo načítaním špecifického URL odkazu obsahujúci Váš vygenerovaný token.

V tomto projekte som sa rozhodol použiť DuckDNS z dôvodu, že server obsahujúci všetky nevyhnutné služby pre funkčnosť celého projektu je pripojený k internetu prostredníctvom poskytovateľa, ktorý nám dynamicky prideľuje IPv4 adresu pri každom reštarte modemu. Táto nestála adresa je dôvodom, prečo je služba DuckDNS ideálnym riešením pre udržanie dostupnosti servera cez internet, a namiesto potreby si pamätať zložité čísla IPv4 adresy nám stačí použiť jednoducho zapamätateľnú adresu. Ďalšie platené a bezplatné DDNS služby zahrňujú DynDNS, No-IP, FreeDNS a iné. Niektoré z nich je možné nakonfigurovať vo webovom rozhraní routerov.

1.9.2 Port forwarding

Port forwarding (presmerovanie portov) je spôsob, ktorý umožňuje sprístupniť určité sieťové porty z lokálnej siete na internet. Najčastejšie sa nastavuje na domácom routeri pomocou webového rozhrania. Ak máte vo Vašej sieti double NAT (situácia, kedy sú dva routery za sebou pripojené a obidva majú zapnutý preklad sieťových adries), je potrebné vykonať presmerovanie portov na oboch. Treba mať na pamäti, že

otváranie portov do internetu predstavuje určité bezpečnostné riziko, takže sa odporúča otvárať čo najmenší počet.

V rámci tohto projektu bolo nevyhnutné implementovať presmerovanie portov a dynamickú aktualizáciu doménovej adresy, aby bol umožnený prístup k verejnému Grafana dashboardu z akéhokoľvek miesta.

1.10 Vývojová doska Heltec CubeCell AB02A

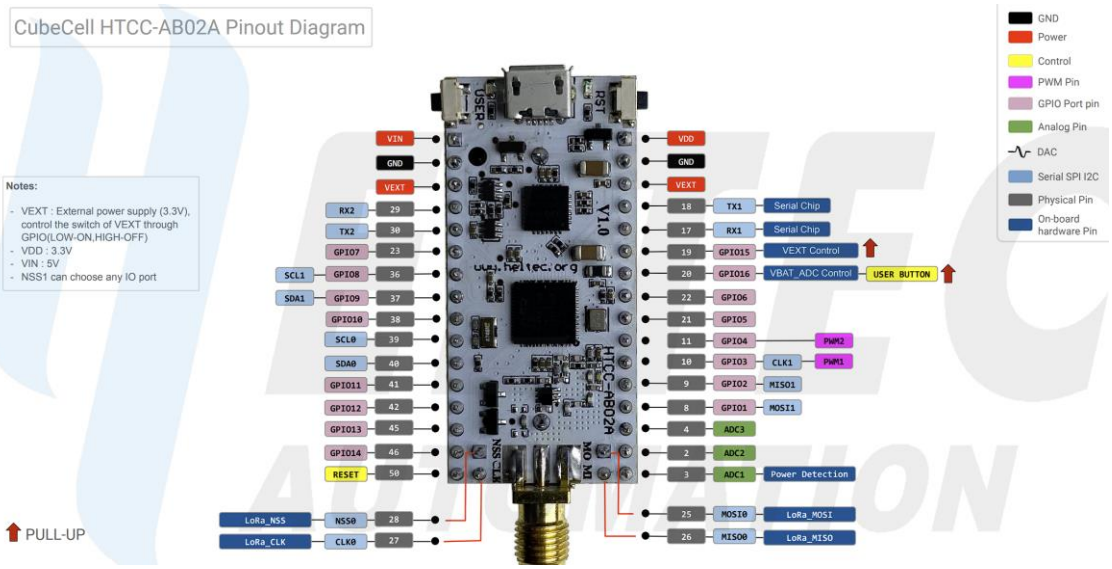
Heltec CubeCell 1/2AA Node, tiež známa ako HTCC-AB02A, je nízko nákladová vývojová doska vyvinutá spoločnosťou Heltec Automation. Jej primárne využitie je pre IoT LoraWAN aplikácie a pýši sa malými rozmermi a vysokou energetickou efektivitou. V čase písania tejto práce sa jej cena pohybuje okolo 15 €.

Doska je založená na ASR Microelectronics ASR6052 integrovanom, ultra nízko odberovom systéme na čipe (SoC) založenom na sérii PSoC 4000 mikrokontrolérov (MCU) a Semtech SX1262 LoRa rádiovom transceiveri (vysielač / prijímač)[14; 15]. Srdcom dosky je jedno jadro ARM® Cortex® M0+ bežiace na takte 48 MHz, ktoré je optimalizované na ultra nízku spotrebu energie. Uvedenie na trh bolo v roku 2012, použitá je ARMv6-M architektúra. Ďalej sa na doske nachádza 128KB flash pamäte a 16 KB RAM. Výrobca udáva spotrebu 3,5 µA v hlbokom spánku. Doska je kompatibilná s Arduino platformou, nachádza sa tu podpora pre protokol LoRaWAN 1.0.2, integrovaný CP2102 USB čip na sériovú komunikáciu, napájanie cez micro USB konektor a LoRa Antenna interface (SMA) na pripojenie antény. Pri kúpe je dôležité si vybrať správnu frekvenciu pre región, v ktorom sa bude doska používať. Podporované frekvencie sú 433 MHz, 470 ~ 510 MHz, 863 ~ 870 MHz a 902 ~ 928 MHz[16; 17].

Na výber sú aj iné dosky s podobným názvom na rôzne účely od spoločnosti CubeCell, konkrétne HTCC-AB02 a HTCC-AB02S[18; 19]. Dosky sa líšia len vo veciach, ktoré sú zobrazené v tabuľke a majú mierne odlišné rozloženie pinov. Ostatné parametre, ako veľkosť flash pamäte, SoC a ďalšie sú rovnaké:

Tabuľka 1.1 - Rozdiely medzi vývojovými doskami Heltec CubeCell

	HTCC-AB02A	**HTCC-AB02**	**HTCC-AB02S**
Napájanie	Puzdro na batériu veľkosti 1/2 AA	Systém pre nabíjanie lítiovej batérie	Systém pre nabíjanie lítiovej batérie
Solárne nabíjanie	☒	☑	☑
OLED displej	☒	☑	☑
GPS modul	☒	☒	☑



Obrázok 1.3 Pinout diagram použitej vývojovej dosky

Schematický diagram vývojovej dosky Heltec CubeCell AB02A sa v originálnej kvalite, pre lepšiu čitateľnosť, nachádza v elektronickej prílohe.

1.11 Senzory

1.11.1 DHT-22

Senzor DHT-22, vyvinutý spoločnosťou Adafruit, je nízko nákladový digitálny senzor určený na meranie teploty a vlhkosti vzduchu. Na meranie teploty používa NTC termistor a na meranie vlhkosti slúži kapacitný senzor s polymérovým kondenzátorom. Najčastejšie sa používa s jednodoskovými počítačmi ako sú Arduino a Raspberry Pi, ale je možné ho nájsť aj v iných produktoch. DHT-22 je nástupca senzoru DHT-11 a oproti nemu vykonáva oveľa presnejšie a rýchlejšie merania. Keďže relatívna vlhkosť závisí od teploty, pre čo najpresnejšie meranie vlhkosti je tento senzor teplotne kompenzovaný a kalibrovaný v presných kalibračných komorách. Následne je kalibračný koeficient uložený do OTP pamäte senzoru. Senzor má 4-pinový vývod. Jeden pin slúži na napájanie v rozsahu 3,3 – 6 V, ďalší je uzemňovací pin, tretí je dátový pin a posledný, štvrtý pin sa nezapája. Perióda ktorou môže tento senzor merať, je 2 sekundy. V čase písania tejto práce sa cena tohto senzoru pohybuje okolo 5 €. Tento senzor vyhovuje svojou presnosťou účelu, na ktorý bude v tejto práci používaný. Špecifikácie sú zobrazené v tabuľke nižšie[20; 21].

Tabuľka 1.2 - Špecifikácie senzoru DHT-22

	Rozlíšenie	Presnosť	Merací rozsah
Teplota [°C]	0,1	±0,5	- 40 ~ 80
Relatívna vlhkosť [%]	0,1	±2 ~ ±5	0 ~ 99,9

1.11.2 Odporový senzor vlhkosti pôdy

Odporový senzor vlhkosti pôdy s dvojitým komparátorom napätia LM393 je jednoduchý a cenovo dostupný senzor. Jeho fungovanie spočíva v meraní odlišného odporu pôdy pri rôznych úrovniach vlhkosti. Celý senzor je zložený z dvoch hlavných častí: Dvoch elektród, ktoré sa vkladajú do pôdy, a senzoru obsahujúceho LM393 komparátor a ďalšie súčiastky, ako sú dve LED a 10kΩ potenciometer. Na jednej strane dosky sa nachádzajú piny na pripojenie k elektródam. Tieto elektródy sa zapichujú do pôdy a merajú odpor pôdy v závislosti od jej vlhkosti. Senzor potom generuje digitálny a analógový výstup. Potenciometer slúži na nastavenie citlivosti digitálneho výstupu podľa nášho želania. Druhá strana obsahujúca ďalšie štyri piny sa najčastejšie pripája k jednodoskovým počítačom, ako sú Arduino alebo Raspberry Pi. Prvý pin slúži na pripojenie napájania v rozsahu 3,3 – 5 V, druhý pin je uzemnenie, tretí pin poskytuje digitálny výstup a štvrtý pin poskytuje analógový výstup. V čase písania tohto dokumentu sa cena takéhoto senzoru pohybovala okolo 1,50 €[22].

Po dlhšom používaní konkrétneho senzoru som dospel k záveru, že senzor je možné používať na meranie vlhkosti pôdy a jeho presnosť je dostatočná, ak potrebujete vedieť, či je vlhkosť pôdy nízka, optimálna alebo vysoká, no nie úplne vyhovuje mojim požiadavkám. Namerané hodnoty majú medzi sebou niekedy väčšie odchýlky, ako by som si želal, a musel som túto nepresnosť vyriešiť priemerovaním troch meraní. Ďalšia vec je, že v budúcnosti môže dôjsť k zhoršeniu presnosti senzoru v dôsledku prúdu, ktorý preteká cez elektródy. Tento prúd môže spôsobiť poškodenie elektród, čo negatívne ovplyvní spoľahlivosť a presnosť meraní. Kvôli týmto nevýhodám bol tento senzor eventuálne vymenený za kvalitnejší a spoľahlivejší.

1.11.3 Kapacitný senzor vlhkosti pôdy

Väčšina lacných senzorov vlhkosti pôdy využíva odporový spôsob merania, kde sú dve elektródy zapichnuté do pôdy a senzor meria množstvo vody na základe odporu medzi nimi. Zo začiatku môže tento senzor vyzerať ako ideálny kandidát, ako som si aj ja myslel, ale nie je to tak. S postupom času začnú elektródy hrdzavieť, čím sa znižuje ich presnosť. Kvôli tomu je nutné ich často kalibrovať a v mojom prípade aj spôsobovali značnú chybu v meraní.

Kapacitné senzory sú našťastie ideálnou alternatívou. Cena je len mierne vyššia ako v prípade kapacitných senzorov, no na druhej strane sú oveľa spoľahlivejšie, odolné voči korózii a majú dlhšiu životnosť. Pracujú na princípe kapacitného merania, čo ponúka významné výhody oproti rezistívnemu meraniu. Tieto senzory majú len jednu sondu, žiadny vystavený kov, ktorý by mohol zhrdzavieť, a neškodia rastlinám zavádzaním elektrickej energie do pôdy.

Pracujú na báze 555 časového integrovaného obvodu (IC) a fungujú meraním, ako rýchlo alebo pomaly sa kondenzátor nabíja cez odpor. V tomto prípade však kondenzátor nie je doslova súčiastkou, ale je tvorený dvoma PCB stopami (PCB trace), ktoré sú blízko seba. Ich kapacita, a teda aj ich nabíjacia rýchlosť, sa mení v závislosti od množstva vody, ktorá sa nachádza okolo nich. Senzor je skonštruovaný z 3,3 V napäťového regulátora, čo ho robí vhodným pre 3,3 V a 5 V mikrokontroléry. Spotreba sa pohybuje na hodnote menej ako 5 mA. Pri použití 5V napájania sa výstup pohybuje v rozsahu od 1,5 V (pre mokrú pôdu) do 3 V (pre suchú pôdu). Výsledná hodnota je však závislá na viacerých faktoroch, vrátane hĺbky zastrčenia sondy a hustoty zeme[23].

Po používaní odporového senzoru po dobu pár mesiacov som dospel k záveru, že je potrebné ho vymeniť za kvalitnejší kapacitný senzor, keďže mi naozaj začal korodovať. Po výmene za kapacitný senzor bolo hneď jasné, že jeho presnosť je oveľa vyššia. Výstupné namerané hodnoty mali medzi sebou menšiu odchýlku.

1.12 Arduino IDE

Arduino IDE (Integrated Development Environment) je programátorské vývojové prostredie ktoré podporuje jazyky C / C++ určené na programovanie Arduina a Arduino kompatibilných zariadení. Je možné ho spustiť na rôznych operačných systémoch, ako je Windows, MacOS a Linux. Hneď ako ho prvýkrát otvoríte, uvidíte jeho minimalistický dizajn, kde textový editor zaberá väčšinu priestoru a na vrchu je panel nástrojov. Obsahuje prepracovanú správu knižníc s možnosťou automatickej inštalácie alebo manuálnej inštalácie zo .zip súboru. Ak Vaša doska nie je predvolene detegovaná, zíde sa Vám možnosť pridania jej podpory cez funkciu „Additional Board Managers“, ktorú nájdete v nastaveniach preferencií. K dispozícii je tiež funkcia aktualizácie firmvéru zariadenia alebo sériový monitor na debugovanie kódu v reálnom čase bežiaceho na Vašej vývojovej doske[24].

Rozhodnutie použiť Arduino IDE na programovanie vývojovej dosky od firmy Heltec bolo jednoznačné, pretože táto doska je plne kompatibilná s týmto vývojovým prostredím. Aby však doska správne fungovala, bolo nevyhnutné stiahnuť vývojový

framework prostredníctvom správcu dosiek (Boards manager). Okrem toho, bolo potrebné pridať vhodné knižnice cez správcu knižníc pre efektívnu prácu so senzormi.

2. BEZDRÔTOVÁ KOMUNIKÁCIA A JEJ NASADENIE

2.1 LoRa

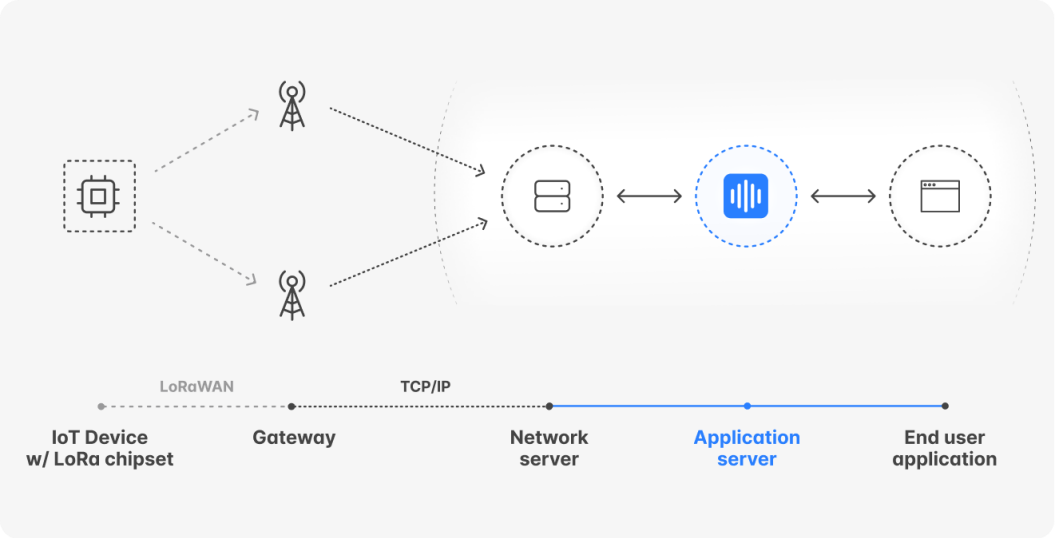
LoRa (Long Range) je nízkoenergetická modulačná technika založená na princípoch Chirp Spread Spectrum (CSS) technológie. Informácie sú zakódované v rádiových vlnách použitím chirp pulzov, kde sa frekvencia signálu mení nepretržite v priebehu času. Niektoré živočíchy, ako netopiere alebo delfíny, komunikujú podobným spôsobom. Táto modulačná technológia je odolná voči rôznym druhom rušenia, ako je Multipath fading, kde sa signál odráža od viacerých vecí naraz a každý sa dostaví k prijímačovi v mierne odlišnom čase, alebo Dopplerov efekt, kde nastane zmena frekvencie spôsobená pohybom pozorovateľa alebo zdroja. Vďaka týmto vlastnostiam je možné používať túto modulačnú techniku na veľké vzdialenosti, a dokonca aj pod povrchom zeme, a je ideálna na používanie v IoT aplikáciách na prenos malého množstva dát pri nízkych prenosových rýchlostiach. Typicky sa využívajú frekvencie v nelicencovanom frekvenčnom pásme 915, 868 a 433 MHz ale je možné používať aj 2,4 GHz pásmo. Vznik tejto technológie bol daný aj tým, že ostatné bezdrôtové technológie, ako WiFi, Bluetooth a mobilné siete (2G až 5G), nesplňovali požiadavky na dostatočne nízky odber elektrickej energie a nie sú vhodné na poháňanie IoT zariadení, ktoré majú vydržať na batériu niekoľko mesiacov až rokov. Na referenčnom OSI modeli (Open Systems Interconnection Reference model) sa LoRa nachádza na najnižšej, fyzickej vrstve a umožňuje hardvérovým zariadeniam využívať nelicencované frekvenčné pásma na nízko energetické WAN (LPWAN) účely [25; 26].

2.2 LoRaWAN

LoRaWAN (Long Range Wide Area Network) je proprietárny LPWAN Media Access Control (MAC) protokol založený na LoRa modulácii, vyvinutý spoločnosťou Semtech. Je to softvérová vrstva, ktorá definuje, ako budú zariadenia používať LoRa hardvér. V súčasnosti je spravovaný neziskovou organizáciou Lora Alliance a prvá verzia bola zverejnená v januári 2015. V čase písanie je najnovšia verzia 1.0.4, publikovaná v roku 2020. Na referenčnom OSI modeli LoRaWAN pokrýva sieťovú a linkovú vrstvu (L3 - network and L2 - data link layers). Sieť LoRaWAN pozostáva z dvoch základných rolí. Node (uzol) a Gateway (komunikačná brána), tiež nazývaná ako koncentrátor. Uzol je zariadenie, väčšinou nejaký senzor, ktorý je pripojený do LoRaWAN siete a odosiela údaje. Komunikačná brána je zariadenie, ktoré je najčastejšie pripojené k internetu.

Slúži na odosielanie a prijímanie dát medzi uzlom a serverom, ktorý prijaté dáta ďalej spracuje.

Napriek tomu, že LoRa funguje na princípe topológie peer-to-peer (rovný s rovným), LoRaWAN formuluje sieť do hviezdicovej (star) alebo mesh topológie. Hviezdicová topológia je najpoužívanejšia a funguje na princípe komunikácie uzlov priamo s komunikačnými bránami, ktoré sú potom pripojené k internetu. Mesh topológia umožňuje prepojenie viacerých uzlov naraz a zlepšiť tým pokrytie a spoľahlivosť siete.



Obrázok 2.1 Schéma princípu fungovania siete LoRaWAN

Okrem podpory šifrovania protokolom AES na rôznych úrovniach je k dispozícii aj technológia dynamickej zmeny frekvencie (frequency hopping) alebo nastavenie premenlivého dátového toku (variable data rate). Toto umožňuje kompromis medzi dĺžkou správy a dosahom. Vďaka adaptívnemu dátovému toku (adaptive data rate) môže sieťový server znížiť energetické nároky koncových uzlov zmenou vysielacieho výkonu. Vo všeobecnosti sa prenosová šírka LoRaWAN sietí pohybuje medzi 0,3 až 50 kbps. Dostatočné na jednoduchú komunikáciu, ale na aplikácie v reálnom čase, ako telefonovanie, by vysoká odozva spôsobovala problémy. V budúcnosti nie je vylúčené, že smartfóny dostanú podporu LoRaWAN sietí na núdzové účely, ktoré by mohli napríklad v horách alebo ťažko dostupných miestach bez signálu mobilnej siete zachrániť ľudské životy. V roku 2020 bol spravený svetový rekord, kedy vyslaný paket zo vzdušného balónu v Holandsku bol zachytený prijímačom na vrchu Radhošť v Českej republike. To je cez 800 km[27; 28; 29; 30]!

2.2.1 Klasifikácia LoRaWAN sietí

- **Verejné –** Verejné LoRaWAN siete sú typicky nasadené mobilnými operátormi alebo poskytovateľmi internetu. Tieto siete využívajú existujúcu infraštruktúru a pokrývajú konkrétne geografické oblasti . Fungujú na báze predplatného a môžu byť pribalené napríklad k internetovému balíčku.
- **Komunitné –** Komunitné LoRaWAN siete sú väčšinou podporované individuálnymi ľuďmi, malými organizáciami alebo neziskovými organizáciami. Typické príklady sú The Things Network (TTN) alebo sieť Helium. Obvykle sú otvorené pre všetkých ale pokrývajú menšie oblasti ako verejné siete.
- **Súkromné –** Súkromné LoRaWAN siete sú najčastejšie používané v podnikoch, ktoré potrebujú plnú kontrolu nad ich sieťou. Tieto siete je možné upraviť podľa vlastných potrieb a častokrát nemusia byť pripojené k internetu pre vyššiu bezpečnosť.

2.2.2 Triedy uzlov

LoraWAN uzly (Nodes) je možné klasifikovať do troch tried. Tieto triedy označujú spôsob komunikácie uzlov so sieťou a ich kompromisy medzi spotrebou elektrickej energie a odozvou. Typ triedy je možné zmeniť za behu.

Trieda A – Trieda A sa dá považovať za najviac energeticky efektívnu, ale aj najpomalšiu. Koncové zariadenie vie prijímať dáta v krátkom časovom intervale len ak predtým nejaké dáta odoslalo. Využitie je vhodné pre aplikácie, kde je najpodstatnejšia nízka spotreba energie a zariadenie neprijíma žiadne dáta alebo prijíma len minimálne množstvo. Táto trieda je ideálna pre tento projekt, lebo vývojová doska nebude prijímať žiadne dáta, iba odosielať.

Trieda B – Trieda B je založená na triede A pre prijímanie, ale pridáva možnosť naprogramovať periodické otvorenie okna na prijímanie dát s maximálnou odozvou 128 sekúnd. Toto umožňuje sieti odosielať dátovú komunikáciu s deterministickou odozvou. Spotreba energie je zanedbateľne vyššia, takže je výhodné používať túto triedu na aplikácie, kde je potrebné častejšie posielať dáta do zariadenia ako pri triede A.

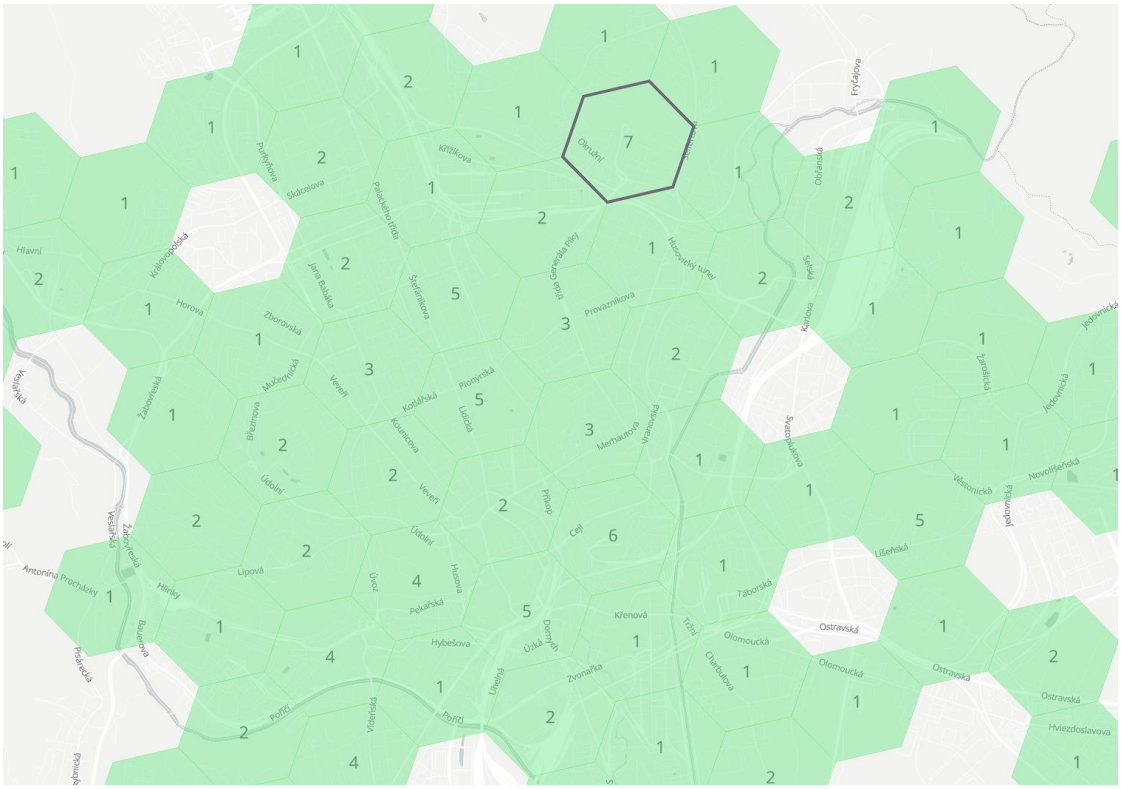
Trieda C - Trieda C umožňuje koncovému zariadeniu prijímať dáta v prípade, že žiadne dáta neodosiela (half duplex). Táto možnosť znižuje latenciu na minimum, ale drasticky zvyšuje spotrebuje energie, keďže zariadenie konštantne načúva pre príjem dát. V praxi sa táto trieda používa, ak zariadenie nie je napájané batériou a potrebuje rýchlo a často komunikovať s komunikačnou bránou.

2.3 Výber vhodnej LoRaWAN siete

Na výber LoRaWAN siete som v tomto projekte mal medzi dvomi možnosťami. Sieť The Things Network (TTN) a Helium Network. Obidve siete sú otvorené, decentralizované a s veľkou komunitou, ale predsa medzi nimi existujú zásadné rozdiely. Sieť TTN sa spolieha na finančné dary od dobrovoľníkov a komunity, na rozdiel od siete Helium, ktorá je postavená na modeli blokchain, kde sú užívatelia odmeňovaní kryptomenou Helium Network Token (HNT) za prevádzkovanie hotspotov. Aby tento model odmien fungoval, Helium používa Proof-of-Coverage (PoC) model, kde sa overuje, či hotspoty naozaj poskytujú pokrytie a požadovanú kvalitu. Sieť TTN tento mechanizmus nepodporuje, takže kvalita pripojenia a pokrytie siete nemusí byť vždy ideálne[31; 32].

Vhodnú LoRaWAN sieť som si pre tento projekt vyberal podľa pokrytia. Na obrázku nižšie je vidieť, že sieť TTN má v Brne 6 komunikačných brán, z toho sú iba 4 funkčné. Ešte horšie je to v meste Nové Zámky na Slovensku, odkiaľ pochádzam. Napriek tomu, že je to okresné mesto s približne 40 000 obyvateľmi, sa tam nenachádza ani jedna TTN komunikačná brána. Toto by pre mňa predstavovalo problém, ak by som tam chcel na tomto projekte pracovať. Naopak, sieť Helium má v Brne viacej ako 40 komunikačných brán, takže by nebol problém sa k nej pripojiť. Aj Nové Zámky sú slušne pokryté s deviatimi bránami. Rozhodnutie teda padlo na LoRaWAN sieť Helium.

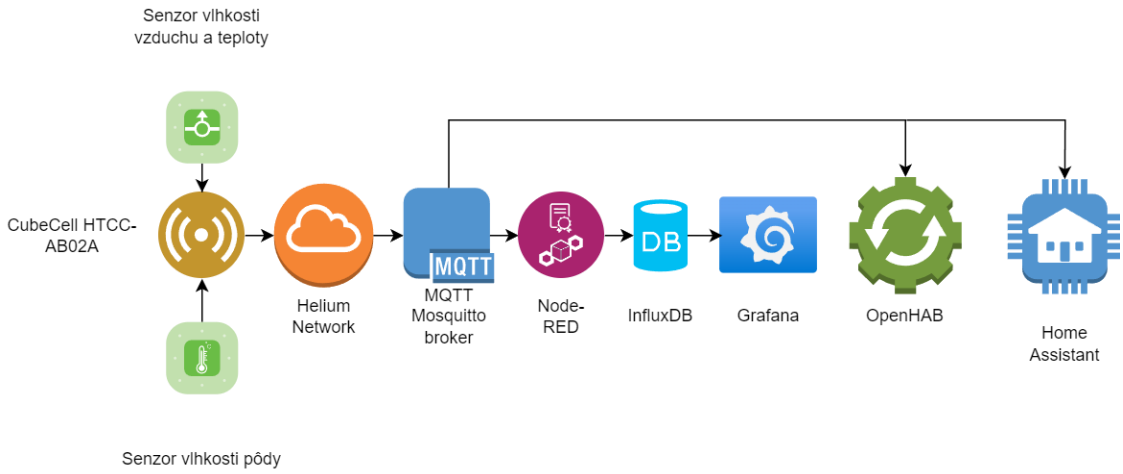
Obrázok 2.2 Pokrytie sieťou TTN v Brne[33]



Rád by som ešte pripomenul, že posielanie a príjem dát v Helium sieti nie je zadarmo. Cena sa vyvíja podľa veľkosti payloadu. Payload s veľkosťou 0 – 24 byteov stojí jeden Data Credit (DC). Dva DC stojí 25 – 48 bajtový payload, atď. Za 0.50 $ je možné zakúpiť 50 000 DC. Helium má vo veľmi limitovanom množstve aj 5G mobilné siete, za ktoré sa tiež platí pomocou DC, samozrejme, ale iným cenníkom. Rozdiel medzi DC a kryptomenou HNT je ten, že DC sa používa len na prenos dát v rámci siete. HNT je kryptomena, ktorú dostávajú vlastníci hotspotov ako odmenu a tiež sa používa na obchodovanie.

3. SOFTVÉROVÉ RIEŠENIE A TECHNICKÁ REALIZÁCIA

V poslednej kapitole je detailne popísaná integrácia vývojovej dosky Heltec CubeCell AB02A so všetkými použitými technológiami. Ďalej bude vysvetlená konfigurácia jednotlivých technológií, transformácia dát pre prácu s rôznymi protokolmi a zobrazený konečný výsledok celej práce a správnej funkčnosti. Nakoniec je popísaný technický návrh a realizácia.



Obrázok 3.1 Schéma projektu

3.1 Inštalácia a konfigurácia Arduino IDE

Ešte predtým, ako budeme používať vývojovú dosku Heltec CubeCell AB02A, bolo potrebné nainštalovať vývojové prostredie Arduino IDE. Po spustení programovacieho prostredia a pripojení micro USB kábla k vývojovej doske zariadenie nebude rozpoznané. Problém spočíva v tom, že Arduino IDE nedokáže rozpoznať túto dosku. V preferenciách Arduino IDE je možnosť pridania podpory pomocou „Additional Board Managers“. Ďalší problém je chýbajúci ovládač pre sériovú komunikáciu cez mikro USB port. Ten sa bohužiaľ automaticky nenainštaluje a je potrebné ho manuálne stiahnuť zo stránky výrobcu a nainštalovať v správcovi zariadení. Po vykonaní predošlých dvoch krokov je vývojová doska rozpoznaná programovacím prostredím Arduino IDE. Na správne fungovanie senzorov, ktoré budú neskôr pripojené k doske, je nevyhnutné pridať ich knižnice cez správcu knižníc. Konkrétne je to „Adafruit Unified Sensor“ a „DHT sensor library“, obidve od spoločnosti Adafruit. Následne je dobré začať s preddefinovanou LoRaWAN šablónou kódu, ktorú nájdete v záložke

„Examples“. Nezabudnite v záložke „Tools“ zmeniť región na „EU868“, debug level na „Freq & Dio“, poprípade ďalšie nastavenia podľa vlastného uváženia.

3.2 Operačné systémy

Existuje viacero operačných systémov, ktoré je možné použiť na tento projekt. Preto popíšem inštaláciu dvoch. Jeden platený, ktorý som používal ja, a druhý bezplatný ktorým je možné docieliť ten istý výsledok.

3.2.1 Inštalácia OS Unraid

V tomto projekte som používal operačný systém Unraid vďaka tomu, že som ho už mal nainštalovaný na mojom lokálnom serveri a umožňuje mi používať Docker kontajnery. Inštalácia je nasledovná[35]:

Najprv je potrebné si zakúpiť licenciu a nahrať operačný systém na USB kľúč použitím USB Flash Creator nástroja z Unraid web stránky. Potom stačí zapojiť USB kľúč do servera, ktorý budeme používať, a zmeniť boot order v BIOSe tak, aby sa bootovalo z USB kľúča. Po spustení OS si treba prejsť úvodným sprievodcom a zaregistrovať si licenciu. Následne je potrebné vytvoriť diskové pole. Odporúča sa mať aspoň 2 disky, aby mohla byť zaistená redundancia, ale nie je to nevyhnutné na fungovanie. Pred ďalším krokom odporúčam skontrolovať pripojenie servera k internetu. Ďalej je potrebné nainštalovať komunitné aplikácie, odkiaľ budú stiahnuté Docker kontajnery pre naše aplikácie, a v nastaveniach povoliť službu Docker. Na hornej lište je potrebné kliknúť na „plugins“ a vložiť nasledujúci odkaz a kliknúť na install:

```
https://raw.githubusercontent.com/Squidly271/community.applications/master/plugins/community.applications.plg
```

Po inštalácii pribudne v hornom paneli položka „apps“, klikneme na ňu a vyhľadáme požadované Docker kontajnery použitím vyhľadávacieho poľa. Konkrétne sú to Mosquitto, Node-RED, InfluxDB, Grafana, OpenHAB, Home Assistant a DuckDNS. Portainer-CE nie je potrebné nainštalovať, lebo UnraidOS obsahuje vlastné používateľské rozhranie na konfiguráciu a správu Docker kontajnerov. Zobrazenie Docker kontajnerov je možné kliknutím na hornej lište na položku „Docker“.

3.2.2 Inštalácia IoT Stack

Keďže som sa rozhodol používať Docker kontajnery pre všetky aplikácie, druhá z možností je použitím projektu IoTstack, napríklad na Raspberry Pi. Najprv je potrebné nainštalovať 64 bitový Raspberry Pi OS Lite a zapnúť SSH. Po inštalácii OS odporúčam skontrolovať pripojenie zariadenia k internetu a následne je potrebné vykonať aktualizáciu príkazmi:

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

Nasledujúcim curl príkazom sa stiahne IoTStack:

```
curl -fsSL https://raw.githubusercontent.com/SensorsIot/IOTstack/master/install.sh | bash
```

Ďalej, reštartujeme zariadenie. Po reštartovaní zariadenia sa znovu pripojíme cez SSH, otvoríme priečinok a spustíme skript:

```
cd IOTstack/
./menu.sh
```

V grafickom menu si vyberieme Docker kontajnerové aplikácie, ktoré chceme nainštalovať. Pre tento projekt odporúčam vybrať Mosquitto, Node-RED, InfluxDB, Grafana, OpenHAB, Home Assistant, Portainer-CE a DuckDNS. Oproti inštalácii Docker kontajnerov na Unraid OS tu musíme nainštalovať prídavný Portainer-CE kontajner, ktorý slúži na inštaláciu a správu kontajnerov pomocou webového rozhrania. V ďalšom kroku otvoríme webové rozhranie Portainer-CE zadaním URL do prehliadača na inom zariadení vo formáte raspberryIP:port (IP adresa RaspberryPi a predvolený port 9000). V mojom prípade adresa vyzerá nasledovne: http://192.168.0.220:9000. Po vytvorení administrátorského účtu kliknutím na „containers“, sa na ľavom paneli zobrazia všetky spustené a nainštalované kontajnery[36].

3.3 Konfigurácia Docker kontajnerov

Docker kontajnery sú ideálne na použitie v tomto projekte. Ich spustenie a konfigurácia je nenáročná a v prípade potreby je možné ich jednoducho premiestniť do iného zariadenia, na ktorom takisto beží služba Docker. Z tohto dôvodu sa tento postup líši iba minimálne pre oba vyššie spomínané operačné systémy.

3.3.1 Mosquitto MQTT

Pre správne fungovanie Mosquitto MQTT nie je potrebná žiadna konfigurácia. Po nainštalovaní Docker kontajnera sa spustí a hneď je možné ho používať bez akýchkoľvek ďalších zásahov. Potrebná je iba znalosť používanej IP adresy a portu kontajnera.

3.3.2 InfluxDB

Po inštalácii Docker kontajnera a pripojení cez SSH na server / Raspberry Pi je potrebné vytvoriť novú databázu na ukladanie dát nasledujúcimi príkazmi:

```
docker exec -it influxdb influx
CREATE DATABASE nazov_databazy
```

Podľa toho, ktorú verziu databázy používate, bude možno potrebné ešte pred predchádzajúcimi príkazmi napísať príkaz influx. Prvým ukázaným príkazom sa dostaneme do konzole konkrétneho kontajnera a druhým vytvoríme databázu s vlastným názvom. V tomto momente je vytvorená nová databáza, v ktorej sa žiadne údaje nenachádzajú.

V prípade, že ste do databázy nejaké dáta uložili, ich môžete zobraziť nasledujúcimi príkazmi:

```
influx
USE nazov_databazy
SELECT * FROM „nazov_merania“
```

Vymazanie všetkých údajov sa vykoná nasledujúcim príkazom:

```
DROP SERIES FROM "nazov_merania"
```

Vytvorenie nového administrátorského používateľa s heslom je možné vykonať takto:

```
CREATE USER "username" WITH PASSWORD 'password' WITH ALL PRIVILEGES;
```

3.3.3 Node-RED

V prípade, že inštalácia Docker kontajnera Node-RED prebehla úspešne, navštívením webovej adresy serverIP:port (IP adresa servera / RaspberryPi a predvolený port 1880) sa načíta webové rozhranie Node-RED. Ako prvý krok je potrebné nainštalovať InfluxDB prepojenia (Nodes). Na to musí byť Vaše zariadenie, na ktorom beží Docker, pripojené k internetu. „Nodes“ pridáte kliknutím na tri čiarky

v pravom hornom rohu a nasledovne na „Manage palette“. Do vyhľadávacieho poľa zadajte „node-red-contrib-influxdb“ a kliknite na install. Ak sa vrátite naspäť na Váš flow, na ľavej strane uvidíte panel so všetkými nodes, ktoré môžete používať. Prepojenia, ktoré budeme potrebovať, sú „mqtt in“, „change“ a „influxdb out“. Tieto nodes je potrebné prepojiť medzi sebou. Vtedy sa môžu zmeniť ich názvy. Kliknutím na „Change“ treba zmeniť názov na „set msg.payload“.

Na konfiguráciu MQTT vstupu klikneme „mqtt in“ prepojenie a následne na MQTT input tlačidlo, kde pridáme IP adresu servera, port a názov na ktorom beží MQTT Mosquitto broker. Ostatné nastavenia necháme na predvolených hodnotách. V ďalšom kroku klikneme na „Add“ hore vpravo, čím sa uloží tento MQTT server a otvorí sa nám druhé okno. V ňom zadáme topic ktorý sa musí vo všetkých aplikáciách a zariadeniach zhodovať. Políčko Output zmeníme na „a parsed JSON object“ a klikneme na Done.

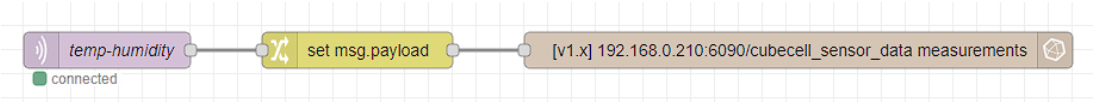
Ďalej otvoríme prepojenie „set msg.payload“, kde zmeníme vstupné pole „to the value“ na hodnotu „expression“, ktorá reprezentuje JSON expression. Táto funkcia nám zmení prijatú správu na JSON expression, s ktorou môžeme ďalej pracovať. Kliknutím na tri bodky napravo od vstupného poľa „to the value“ sa otvorí JSON Expression editor. V tomto editore zmeníme prijaté premenné na premenné, ktoré sú jednoduchšie na pochopenie (viď zdrojový kód nižšie). Po vykonaní zmien stačí kliknúť na Done dva krát, čím sa uložia zmeny.

```
{
  "temperature": msg.payload.decoded.payload.temperature,
  "humidity": msg.payload.decoded.payload.humidity,
  "soil_humidity": msg.payload.decoded.payload.soil_humidity,
  "battery_voltage": msg.payload.decoded.payload.battery_voltage,
  "channel": msg.payload.hotspots[0].channel,
  "frequency": msg.payload.hotspots[0].frequency,
  "lat": msg.payload.hotspots[0].lat,
  "long": msg.payload.hotspots[0].long,
  "name": msg.payload.hotspots[0].name,
  "rssi": msg.payload.hotspots[0].rssi,
  "snr": msg.payload.hotspots[0].snr
}
```

V poslednom kroku sa konfiguruje influxdb prepojenie. Kliknite na neho a v okne vedľa vstupného poľa „Server“ kliknite na ceruzku, ktorá otvorí Properties okno. V poliach „Host“ a „Port“ zadajte IP adresu a port InfluxDB databázy, ktorá je spustená v Docker kontajneri. Potom v poli „Database“ napíšte názov InfluxDB databázy, ktorú ste predtým vytvorili v kroku 3.3.2. Nasledovne kliknite na Done

a v ďalšom okne do poľa „Measurement“ zadajte názov merania, ktorý uznáte za vhodný. Kliknutím na „Done“ uložíte všetky zmeny.

V prípade, že sú všetky nodes medzi sebou prepojené a správne nakonfigurované, môžete kliknúť na červené tlačidlo v pravom hornom rohu Deploy, ktoré spustí nakonfigurovaný Flow.



Obrázok 3.2 Nakonfigurovaný Flow v Node-RED

3.3.4 Grafana

Na vykreslenie grafov, ktoré boli namerané senzormi, je použitá platforma Grafana. Po nainštalovaní a spustení Grafana Docker kontajnera prejdeme na serverIP:port (IP adresa servera / RaspberryPi a predvolený port 3000). V úvodnom sprievodcovi si vytvoríme administrátorský účet s heslom. Po prejdení úvodného sprievodcu sa Vám zobrazí domovská stránka. Klikneme na „Add your first data source“ a následne InfluxDB. V poli URL zadáme IP adresu Docker kontajnera, na ktorom beží InfluxDB, napríklad http://192.168.0.210:8086. Presunieme sa spodok stránky a v poli „Database“ zadáme názov databázy a zmeníme „http Method“ na GET. Potom klikneme na „Save & Test“. Ak je všetko správne nastavené, mala by sa Vám ukázať zelená fajka a popis „datasource is working“.

Ďalej potrebujeme vytvoriť dashboard, kde sa nám naše namerané údaje budú graficky zobrazovať. Na ľavom paneli klikneme na „Dashboards", následne na „ + New dashboard“ a „Add a new panel“. V spodnej časti, kde je napísané FROM, klikneme na „select measurement“ a vyberieme názov merania. Potom v „SELECT“ a „field(value)“ vyberieme hodnotu, ktorú chceme zobrazovať v grafe, napríklad teplota. Na pravej strane je možné v poli „Title“ zmeniť názov panelu. Taktiež môžete pridať vlastný popis, zmeniť farby, typ grafu a iné. Uloženie sa vykoná tlačidlom Apply v pravom hornom rohu. V prípade, že chceme zobraziť viac meraní, ako je teplota, vlhkosť a napätie batérie, je potrebné tento proces trikrát opakovať. Keď budete spokojný s výzorom Vášho dashboardu, je potrebné ho pomenovať a uložiť kliknutím na ikonku diskety „Save dashboard“ v hornej časti.

V prípade, že chcete mať aj verejný dashboard, ktorý bude prístupný cez internet, odporúčam vytvoriť ďalší samostatný dashboard. Aby sa mohli anonymný používatelia bez prihlasovacích údajov pozrieť na Váš verejný dashboard, je potrebné

povoliť anonymný prístup v konfiguračnom súbore grafana.ini. Najjednoduchší spôsob, ako upraviť tento súbor, je pripojením sa cez SSH na Váš server alebo Raspberry Pi a upravenie súboru pomocou Unixového editora „Vi“. Ovládanie je nasledovné. Šípkami sa pohybujete hore alebo dole v dokumente, ak chcete prejsť do režimu úprav, stlačte klávesu „i“. Režim úprav ukončíte stlačením ESC. Ukončenie editora Vi bez uloženia sa vykoná stlačením „:q!“. Ukončenie s uložením zmien je možné stlačením „:wq“, čo je skratka pre write and quit. Približne na riadku 554 sa nachádza sekcia „Anonymous Auth“. Tam je potrebné odkomentovať niektoré riadky a zmeniť premennú enable = true. Po úprave by mal kód vyzerať nasledovne:

```
############################# Anonymous Auth ############################
[auth.anonymous]
# enable anonymous access
enabled = true

# specify organization name that should be used for unauthenticated users
org_name = Main Org.

# specify role for unauthenticated users
org_role = Viewer

# mask the Grafana version number for unauthenticated users
hide_version = false

########################### GitHub Auth #################################
```

Dodatočne je nutné reštartovať Docker kontajner Grafany a otvoriť webové rozhranie. Tam je nutné prejsť na dashboard, ktorý chcete zverejniť, a na hornom paneli vedľa názvu dashboardu kliknúť na „Share“. Tým sa otvorí nové dialógové okno, kde prejdeme na sekciu „Public dashboard“. Potvrdíme všetky tri zaškrtávacie polia a klikneme na „Generate public URL“. Toto nám vygeneruje verejnú adresu. Môžete si všimnúť, že obsahuje Vašu lokálnu IP adresu kontajnera bez portu. Takže je potrebné ju upraviť na Vašu verejnú IP adresu a nezabudnite tiež na port. V prípade, že používate dynamické DNS ako ja, je treba zadať adresu dynamického DNS a nie verejnú IP adresu. Na upravenej URL adrese sa bude nachádzať Váš verejný dashboard. Úprava URL adresy by mohla vyzerať nasledujúcim spôsobom:

```
http://192.168.0.210/public-dashboards/abcd
```

URL upravíme na:

```
http://kapia.duckdns.org:6091/public-dashboards/abcd
```

Ako posledný krok odporúčam expandovať v dialógovom okne položku Settings a povoliť „Time range picker“, čím si budú návštevníci môcť zmeniť zobrazené časové obdobie v grafe. Uloženie prevedieme zatvorením dialógového okna a kliknutím na ikonku diskety.

3.3.5 OpenHAB

Pre posielanie notifikácií o nízkom napätí batérie bol použitý OpenHAB. Po stiahnutí a nasadení Docker kontajnera si treba prejsť úvodným sprievodcom vo webovom rozhraní, ktoré je na URL adrese serverIP:port (IP adresa servera / RaspberryPi a predvolený port 8080). Následne sa dostaneme na domovskú stránku. Na ľavom paneli klikneme na Settings a na spodku menu na Other Add-ons. Kliknutím na Search dole vpravo sa nám zobrazí vyhľadávacie pole, kam napíšeme „MQTT“. V zozname hľadaní nainštalujeme „MQTT Binding“ od autora openHAB. Týmto istým spôsobom nainštalujeme „JavaScript Scripting“, „Telegram Binding“ a „Telegram Actions“ prepojenia.

Po pridaní všetkých štyroch prepojení, na ľavom paneli klikneme na Things a dole vpravo na modré plus. Pridanie MQTT Binding a Telegram Binding vykonáme nasledujúcim spôsobom. Po kliknutí na modré plus klikneme na „MQTT Binding“, „MQTT Broker (Bridge)“, vyplníme Label vlastným názvom a klikneme na Show advanced. Potom do „Broker Hostname/IP“ zadáme IP adresu a pod tým port Docker kontajnera, kde beží MQTT Mosquitto broker. Ostatné veci necháme na predvolených hodnotách a uložíme tlačidlom Save vpravo hore. Vrátime sa znovu do Things, klikneme na modré plus a ďalej klikneme na „MQTT Binding“ a „Generic MQTT Thing“. Ako „bridge“ zvolíme MQTT broker, ktorý sme v predošlom kroku vytvorili. Ďalej sa v hornej lište prepneme do Channels, Add Channel a zmeníme Channel type na „Number Value“. Pridáme názov kanálu zmenou „Channel Identifier“ a popis zmenou „Label“. Zaškrtneme Show advanced a do „MQTT State Topic“ zadáme topic, ktorý sme si vytvorili a používame ho napríklad aj v Node-RED. Presunieme sa na spodok web stránky, do poľa „Incoming Value Transformations“ vložíme nasledujúci kód:

```
JSONPATH:$.decoded.payload.battery_voltage
```

Tento kód pomocou prepojenia JavaScript Scripting pretransformuje JSON objekt na použiteľné číslo. Uloženie vykonáme kliknutím na Done hore vpravo. Potom klikneme na zelené plus, následne klikneme na „Add Link to Item...“ a potom na „Create a new Item“. Týmto spôsobom vytvoríme nový Item. Ostatné položky necháme tak ako sú

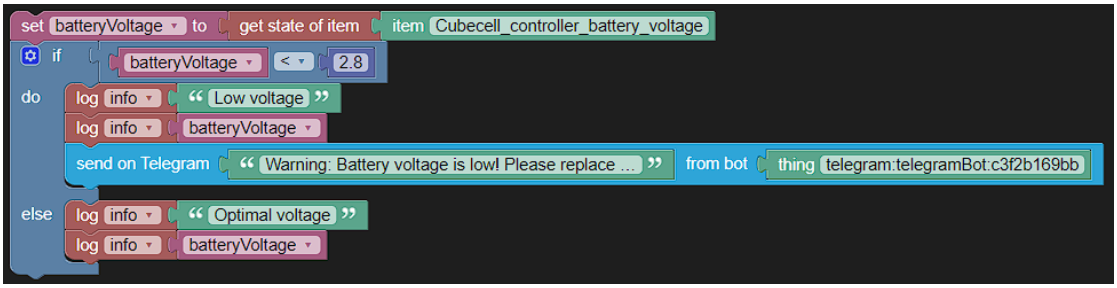
a prepojíme kliknutím na Link na spodku web stránky. Vďaka týmto nastaveniam sme prepojili Item a Channel.

Aby boli namerané hodnoty zobrazené v jednoduchej forme, vytvoríme si dashboard. Na ľavom paneli klikneme na Settings a ďalej na Pages. Vpravo dole označíme modré plus a klikneme na možnosť Create layout. V poli „label“ pomenujeme vytvorený panel, napríklad „Dashboard“. Odporúčam zaškrtnúť možnosť Show on sidebar, aby nám bola uľahčená navigácia v ľavom paneli. Uložíme tlačidlom vpravo hore a presunieme sa do vytvoreného dashboardu. Kliknutím na Edit, tiež vpravo hore, je možné upravovať a pridávať rôzne prvky. Môžete navrhnúť jednoduchý dizajn pridaním riadkov a stĺpcov. Kliknutím na biele plus v strede karty sa otvorí okno s výberom knižnice. Na vytvorenie dizajnu dashboardu som pridal Label Card a Gauge Card. Konfigurácia kariet je jednoduchá. Stačí kliknúť na Configure widget, kde sa otvorí nové modálne okno. V tomto okne vyberiete Item, ktorý sme si predtým vytvorili, napríklad „Teplota vzduchu“. V prípade použitia Gauge Card odporúčam nastaviť minimálne a maximálne hodnoty a pohrať sa s typom a farebnými hodnotami. Podobným spôsobom je možné nakonfigurovať aj Label Card. Tam odporúčam zmeniť „Chart Type“ na Month alebo Week a poprípade pridať ikonu do poľa „Icon“. Výsledný dashboard tohto projektu je priložený v elektronickej prílohe.

Pre príjem notifikácií cez Telegram na mobilnom zariadení je nutné nastaviť ďalší Thing. Takže sa presunieme do panelu Things, kliknutím na modré plus vpravo dole. Vyberieme Telegram Binding a následne Telegram Bot. Do poľa „Bot Token“ zadáme token, ktorý dostaneme od Telegram bota. Tento token je možné získať tak, že napíšete správu v aplikácii Telegram „BotFatherovi“, kde si príkazom /newbot vytvoríte nového bota. Následne z poskytnutej web stránky skopírujete token do OpenHAB. Kliknutím na „Create Thing“ sa Vám zmeny uložia.

Teraz máme vytvorené a nastavené všetky Things, ktoré potrebujeme, aby nám posielanie správ na Telegram fungovalo.

Ako posledný krok nám zostáva vytvoriť ešte skript. Na ľavom paneli klikneme na Scripts, následne na modré plus a ako skriptovaciu metódu vyberieme „Design with Blockly“. Blokový kód pre odosielanie notifikácie pri nízkom napätí batérie by mohol vyzerať nasledovne:



Obrázok 3.3 Blokový diagram Blockly

Správne fungovanie posielania správ je možné otestovať kliknutím na Run Now tlačidla dole vľavo. Blokový diagram pomenujeme a uložíme kliknutím na Save hore vpravo.

Na ľavom paneli sa presunieme na položku Rules, kde vytvoríme nové pravidlo kliknutím na modré plus. V sekcii When pridáme spúšťač (trigger) „If an item state is updated“. Potom v sekcii Then zvolíme „runs/executes a rule“ a vyberieme pravidlo, ktoré sme v predošlom kroku vytvorili. Pravidlo na vrchu pomenujeme a kliknutím na Save uložíme. Vykonaním týchto krokov bude na Vaše mobilné zariadenie s aplikáciou Telegram a pripojeniu k internetu odoslaná notifikácia, ak napätie batérie klesne pod 2,8 V. Podobným spôsobom je možné vytvoriť notifikácie aj pre ostatné namerané veličiny, ako teplota vzduchu, vlhkosť pôdy atď.

3.3.6 Home Assistant

Po inštalácii Docker kontajnera sa Home Assistant (HA) pár minút konfiguruje, čo môže trvať na slabšom hardvéri niekoľko minút. Na webovú stránku prejdeme zadaním URL adresy serverIP:port (IP adresa servera / RaspberryPi a predvolený port 8123). Uvíta Vás sprievodca prvým spustením, kde je potrebné nakonfigurovať používateľské meno a heslo pre prihlásenie. V kroku v ktorom sa Vás HA opýta, či chcete pridať nejaké integrácie, vyhľadáme „MQTT“. Nakonfigurujeme Broker options tak, že do poľa „Broker“ zadáme IP adresu servera / RaspberryPi, kde beží MQTT Mosquitto broker, a do poľa Port zadáme port brokera. V mojom prípade je adresa Brokera 192.168.0.210 a port 6088. Ostatné polia, ako používateľské meno a heslo, necháme prázdne, keďže sme žiadnu autentifikáciu nenastavovali. Dokončíme úvodného sprievodcu a dostaneme sa na domovskú plochu.

Pre konfiguráciu získavania meraných veličín z MQTT brokera je potrebné nájsť konfiguračný súbor HA s názvom „configuration.yaml“ na disku. Pre spustenie príkazového riadku vo vnútri Docker kontajnera je potrebné zadať nasledujúci príkaz:

```
docker exec -it <nazov_home_assistant_kontajnera> /bin/bash
```

Typicky sa konfiguračný súbor nachádza v priečinku /config. Otvoríme konfiguračný súbor napríklad s editorom vi alebo nano. Prejdeme pod posledný riadok „scene“ a pridáme nasledujúci kód:

```
mqtt:
  sensor:
    - name: "Air Temperature"
      state_topic: "temp-humidity"
      suggested_display_precision: 1
      unit_of_measurement: "°C"
      value_template: "{{ value_json.decoded.payload.temperature }}"
```

Týmto kódom sa pridá nameraná hodnota teploty ako MQTT Senzor do HA. Je dôležité podotknúť, že do „state_topic“ sa zadá topic, ktorý sme si vytvorili v MQTT Mosquitto brokeri. Rovnakým spôsobom bol pod teplotou vzduchu pridaný kód pre vlhkosť vzduchu, vlhkosť pôdy a stav napätia batérie. Po uložení zmien v konfiguračnom súbore je potrebné HA kontajner reštartovať. Po reštarte sa na domovskej obrazovke (Overview) zobrazia namerané hodnoty zo zadaných senzorov v konfiguračnom súbore. Kliknutím na ikonku oka je po čase možné sledovať zmeny v grafe.

Na zobrazenie nameraných hodnôt v prehľadnej forme sa odporúča tvorba dashboardu. Na ľavom paneli klikneme na Overview a ďalej na bielu ceruzku Edit, ktorá sa nachádza v pravom rohu. Po zapnutí editačného režimu pridáme kliknutím na biele plus nový View. Do políčka „Title“ vložíme názov a ostatné položky necháme prázdne. Podobne ako v prípade OpenHAB funguje editácia dashboardu pridávaním kariet. Kliknutím na ADD CARD sa zobrazí okno, v ktorom je na výber viacero kariet. V tomto projekte boli použité karty Gauge Card a Sensor Card. Pri upravovaní Gauge Card je nevyhnutné vybrať Entitu, ktorá bola vyššie konfigurovaná, ako napríklad Vlhkosť vzduchu. Ďalej odporúčam nastaviť minimálne a maximálne hodnoty a poprípade zmeniť farby pre rôzne namerané hodnoty. Sensor Card sa nastavuje podobne s pridanou možnosťou zmeny ikony. Týmto spôsobom je možné nadizajnovať dashboard podľa vlastných potrieb. Ja som ešte využil možnosti karty Conditional, kde som nastavil, že ak je napätie batérie nižšie ako 3,2 V, zobrazí sa graf s napätím batérie. Ak je napätie vyššie ako 3,2 V, zobrazí sa karta s nápisom „Napájanie cez USB“. Túto možnosť je možné využiť z toho dôvodu, že integrovaná batéria má maximálne napätie okolo 3,05 V a merané napätie v prípade napájania cez USB port je vždy 3,3 V.

3.3.7 DuckDNS

Služba DuckDNS je aplikovaná v tomto projekte z dôvodu dynamickej IPv4 adresy od poskytovateľa internetu. Jej nastavenie je jednoduché.

Najprv je potrebné sa prihlásiť na webovej adrese www.duckdns.org cez jednu z podporovaných prihlasovacích služieb. V našom prípade bolo použité prihlásenie cez Google účet. Po prihlásení si na domovskej stránke vytvoríme nový názov domény a skopírujeme token, ktorý budeme neskôr potrebovať. Už dlhé roky som využíval doménu kapia.duckdns.org a preto bola použitá aj v tomto projekte.

Získaný token je nevyhnutné zadať do premenných Docker kontajnera. Cez službu Portainer sa to robí nasledujúcim spôsobom. Prejdeme na URL adresu (a port) služby Portainer, prihlásime sa a naľavo klikneme na Containers, Duckdns. Na tlačidlá nachádzajúce sa na vrchu zvolíme Duplicate/Edit. V sekcii Advanced container settings klikneme na „Env“ a zmeníme tieto polia. Do poľa „SUBDOMAINS“ zadáme názov vytvorenej subdomény bez „ .duckdns.org “. V našom prípade bola použitá subdoména s názvom „kapia“ (kapia.duckdns.org). V prípade, že je pole „TZ“ prázdne, je nevyhnutné tam vložiť Vaše časové pásmo (Time zone), napríklad „Europe/Bratislava“. Do poľa „TOKEN“ prilepte Váš skopírovaný token a ostatné polia nechajte tak.

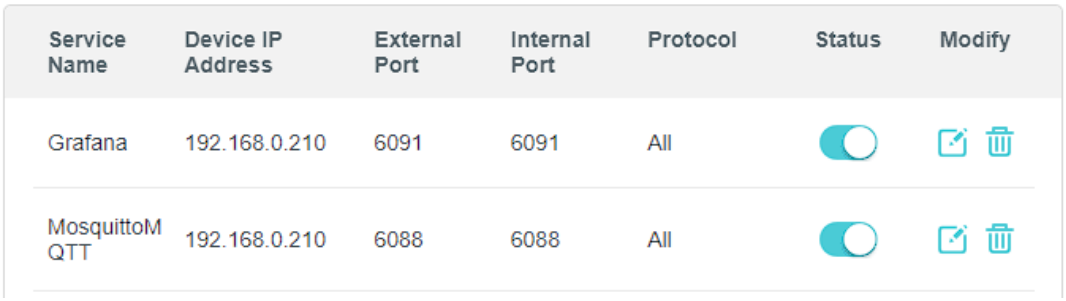
Vykonané zmeny sa uložia kliknutím na Deploy the container tlačidlo v sekcii Actions. Týmto spôsobom bol spustený Váš DuckDNS Docker kontajner, ktorý bude periodicky kontrolovať a aktualizovať Vašu dynamickú IP adresu.

3.3.8 Port Forwarding

Presmerovanie alebo otváranie portov (port forwarding) je nevyhnutné, aby bolo možné načítať cez Internet verejný dashboard Grafany a bol umožnený príjem a odosielanie údajov Mosquitto MQTT brokeru. Bez otvorenia portov by bolo možné zobraziť verejný dashboard iba na lokálnej sieti. MQTT broker Mosquitto by ale nefungoval vôbec, lebo by nevedel prijímať žiadne dáta z Helium konzoly. Je dôležité zdôrazniť, že otváranie portov so sebou nesie isté bezpečnostné riziko.

Postup, ako sa aplikuje presmerovanie portov, je mierne odlišný pre každého výrobcu routerov. V tejto kapitole demonštrujem, ako sa to nastavuje na domácom routeri TP-Link Archer AX20 v1 s firmvérom 1.3.8. V prvom kroku je potrebné otvoriť webové rozhranie routera zadaním jeho IP adresy, napríklad 192.168.0.1, a prihlásiť sa. Na hornej lište klikneme na tlačidlo Advanced, ktoré nám zobrazí pokročilé nastavenia. Na ľavom paneli klikneme na NAT Forwarding a podkategóriu Port Forwarding. Pridanie nových údajov vykonáme kliknutím na Add. Otvorí sa nám nové okno, v ktorom zadáme ľubovoľný názov do „Service Name“ poľa. Do „Device IP

Address“ zadáme IP adresu servera alebo RaspberryPi, na ktorom sú spustené naše Docker kontajnery. Poľe „External Port“ a „Internal Port“ sa musí zhodovať, a mal by tam byť zadaný port, na ktorom beží Grafana (predvolený port pre Grafanu je 3000). Protokol necháme na „All“ a uložíme kliknutím na Save. Ten istý postup, len s iným portom, zopakujeme pre Mosquitto MQTT broker. Po pridaní oboch služieb by výsledok mohol vyzerať nasledovne:



Service Name	Device IP Address	External Port	Internal Port	Protocol	Status	Modify
Grafana	192.168.0.210	6091	6091	All		
MosquittoMQTT	192.168.0.210	6088	6088	All		

Obrázok 3.4 Výsledok presmerovania portov na TP-Link routeri

3.1 Návrh a realizácia

3.1.1 Konštrukcia

Pre správne umiestnenie komponentov a ochranu vývojovej dosky pred nepriaznivým počasím bolo potrebné vybrať vhodný kryt. Vybraná bola plastová krabička s rozmermi 115x90x55 mm, triedou ochrany IP65 a priesvitným vrchným krytom.

Do krabičky boli dokopy vyvŕtané tri otvory. Jeden pre anténu, ktorá slúži na komunikáciu s LoRaWAN sieťou, ktorá je prepojená s vývojovou doskou cez 15 cm SMA male to SMA female kábel. Druhý otvor slúži ako vývod vodičov pre kapacitný senzor vlhkosti pôdy. Posledný otvor bol vyrezaný pre senzor DHT22 ktorý bol na neho priamo nalepený.

Pôvodne mal byť použitý iný, vode odolný senzor teploty a vlhkosti vzduchu ktorý mal mať vyvedené vodiče rovnakým otvorom ako kapacitný senzor vlhkosti pôdy. Žiaľ, komunikáciu medzi senzorom a vývojovou doskou cez I2C protokol sa nepodarilo spojazdniť. Pravdepodobne táto vývojová doska nie je kompatibilná s týmto konkrétnym mikrokontrolérom, keďže na Arduine Mega 2560 fungoval bez problémov.

Ďalej, na napájanie mala byť použitá väčšia 18650 lítiovo-iónová batéria, ktorá nakoniec nebola použitá kvôli svojej veľkosti. Držiak na batériu bol príliš veľký a nezmestil by sa so všetkými potrebnými vodičmi do krabičky. Vývojovú dosku je naďalej možné napájať cez Micro USB konektor alebo menšou 1/2AA batériou, ktorá má dostatočnú výdrž. Pri intervale merania každých 60 minút by mala vydržať pár mesiacov.

3.1.2 Zapojenie

Zapojene vodičov pre jednotlivé senzory je nasledovné (viď pinout diagram). Na doske sa nachádzajú 2x piny pre napájanie a zem (VEXT a GND). K obom senzorom (DHT22 a kapacitný senzor vlhkosti pôdy) je vedený jeden vodič z pinu VEXT pre 3,3 V napájanie a zem z pinu GND. Pre komunikáciu medzi senzorom DHT22 a vývojovou dosku je použitý GPIO7 pin. Analógový pin ADC2 je použitý pre komunikáciu so senzorom pôdy. Piny GPIO4 a GPIO5 sú použité pre určenie vysielacieho intervalu. Ak sú prepojené, vývojová doska vysiela každých 5 minút. Ak nie sú prepojené, interval vysielania je 60 minút.

3.2 Vývoj a analýza systému

Predtým ako bude vývojová doska pripojená do Helium LoRaWAN siete, odporúčam nastaviť všetky Docker kontajnery (kapitola 3.3), inak by sa zobrazovala chyba integrácie v Helium konzole. Aby sa vývojová doska vedela pripojiť a používať Helium LoRaWAN sieť, bolo potrebné si vytvoriť Helium účet. Po vytvorení účtu treba ísť do Helium konzole a tam pridať nové zariadenie (Device). Pri tvorbe nového zariadenia treba skopírovať Device EUI, App EUI a App Key do preddefinovanej šablóny. S týmito údajmi sa vývojová doska prihlasuje do Helium LoRaWAN siete. Ak je všetko dobre zadané, po nahraní kódu cez Arduino IDE by sa zariadenia malo pripojiť do Helium siete a vy by ste mali po rozkliknutí konkrétneho zariadenia vidieť v grafe Real time packets, request to join, join accept a poprípade aj uplink.

3.2.1 Zdrojový kód

V ďalšom kroku bolo potrebné napísať kód, ktorý bude bežať na doske. Kompletný kód je priložený v elektronickej prílohe, ale jeho podstatné časti by som chcel rozpísať.

Na začiatku celého programu sú definované piny, ktoré boli použité na pripojenie senzorov, a premenné použité neskôr v kóde.

Prvá funkcia prepareTxFrame slúži na pripravenie payloadu, ktorý bude odoslaný cez LoRaWAN Helium sieť. Treba si uvedomiť, že technológia LoRaWAN je low power network, takže každý jeden jediný byte sa počíta. Preto je appDataSize nastavený na

payload 8 byteov. Každá nameraná veličina, teda teplota vzduchu, vlhkosť vzduchu, napätie batéria a vlhkosť pôdy, je rozdelená na 2 byty. To je kvôli tomu, že jeden byte má 8 bitov a tým vieme reprezentovať maximálne 255 hodnôt. Predstavme si, že v tomto prípade reprezentujeme desatinné čísla, napríklad 23,65 °C. Aby toto číslo mohlo byť poslané, najprv je vynásobené 100 a tým získame číslo 2365. Toto číslo je následne poslané cez LoRaWAN v dvoch byteoch. High byte a low byte. High byte reprezentuje čísla 23, low byte 65. Táto hodnota poslaná cez sieť je následne dekódovaná transformáciou opačným procesom a ďalej uložená do databáze.

V setup funkcii sa nachádza inicializácia základných parametrov, ktoré bude vývojová doska neskôr v kóde používať. Táto funkcia je spustená hneď po zapnutí.

Vo funkcii loop sa nachádza hlavná logika celého programu. Najprv je zistený stav prepínača a podľa toho sa určí perióda merania (5 minút alebo 60 minút). Ďalej sa v kóde nachádza if cyklus, ktorý sa spustí, ak je čas od ktorého sa program spustil „millis()“ mínus čas posledného merania „lastMeasurementTime“ menší alebo rovný meraciemu intervalu „measurementInterval“. Merací interval je nastavený tak, aby bol 5 sekúnd pred odoslaním údajov do LoRaWAN siete. Týmto je možné docieliť čerstvo namerané údaje, ktoré sú hneď odoslané. Vďaka tomu vývojová doska nemusí na nič zbytočne čakať, takže sa aj ušetrí elektrická energia. Po splnení tejto podmienky sa vykoná meranie zo senzorov.

Na konci tejto loop funkcie sa nachádza kód na správu systému LoRaWAN pomocou rôznych stavov „deviceState“, ako INIT pre inicializáciu, JOIN pre prihlásenie sa do LoRaWAN siete, hlavný stav SEND, ktorý zavolá funkciu „prepareTxFrame“ a odošle obsah payloadu, CYCLE, ktorý spravuje, ako často posielať dáta pomocou premennej „txDutyCycleTime“, a nakoniec stav SLEEP, ktorý prepne vývojovú dosku do nízko odberového spánku.

Vykonaním krokov v kapitole 3.1 je možné kód nahrať prepojením PC a vývojovej dosky micro USB káblom, zvolením príslušného COM portu na vrchu vývojového prostredia a kliknutím na tlačidlo Upload. Výstup sériovej konzoly je možné zobraziť kliknutím úplne v pravom hornom rohu na tlačidlo Serial Monitor.

3.2.2 Transformácia dát

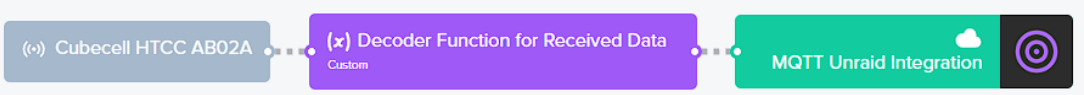
Aby bolo možné prijaté dáta z Helium siete poslať do Mosquitto MQTT brokeru, je potrebné ich transformovať. Na to bol vytvorená funkcia v MQTT konzole, ktorá dáta dekóduje. Postup na vytvorenie dekodéru je nasledovný. Navštívime webovú adresu console.helium.com. Na ľavom paneli klikneme na Functions, fialové plus Add New Function a zvolíme Custom. Funkciu pomenujeme a v textovom poli „CUSTOM SCRIPT“ prilepíme nasledujúci kód:

```
function Decoder(bytes, port) {
  // Combine the high and low bytes to reconstruct the integers
  var temperature = (bytes[0] << 8) | bytes[1];
  var humidity = (bytes[2] << 8) | bytes[3];
  var soil_humidity = (bytes[4] << 8) | bytes[5];
  var battery_voltage = (bytes[6] << 8) | bytes[7];

  // Scale back to floating-point values
  temperature /= 100.0;
  humidity /= 100.0;
  battery_voltage /= 1000.0; // Change the scaling factor to 1000

  return {
    temperature: temperature,
    humidity: humidity,
    soil_humidity: soil_humidity,
    battery_voltage: battery_voltage
  };
}
```

Ten kód z prijatých dát, ktoré sú rozdelené na low a high byty, rekonštruuje dáta naspäť na ľudsky čitateľný tvar čísel. Funkciu uložíme kliknutím na Save Function na pravom spodku stránky. Teraz sú dáta dekódované, ale ešte ich nevieme poslať do MQTT Mosquitto brokeru, lebo k nemu nie sme pripojený, tj. nie je vytvorená integrácia.

Aby bola vytvorená integrácia medzi dekódovanými dátami a MQTT Mosquitto brokerom, klikneme na ľavom paneli na Integrations, ďalej na zelené tlačidlo Add New Integration a v sekcii ADD A CORE INTEGRATION zvolíme MQTT. Pridelíme integrácii názov a v sekcii UPDATE YOUR CONNECTION DETAILS zadáme do „Endpoint“ IP adresu a port Docker kontajnera, v ktorom beží MQTT Mosquitto broker. Nezabudnite použiť prefix „mqtt://“. V tomto projekte bol použitý endpoint „mqtt://kapia.duckdns.org:6088“. Do polí „Uplink“ a „Downlink Topic“ je potrebné zadať topic, ktorý je použitý vo všetkých službách rovnako (napríklad v Node-RED Docker kontajneri). Uloženie sa vykoná kliknutím na tlačidlo Add Integration, ktoré je na spodku. V tomto momente je vytvorená funkcia na dekódovanie prijatých dát a je nastavená integrácia s MQTT Mosquitto brokerom. Už nám iba zostáva prepojiť všetky vytvorené Nodes v sekcii Flows.

Zobrazenie Flows sa vykoná kliknutím na sekciu Flows v ľavom paneli. Následne sa pridá kliknutím a potiahnutím bloku z Devices, Functions a Integrations. Prepojený výsledný Flow by mohol vyzerať takto:

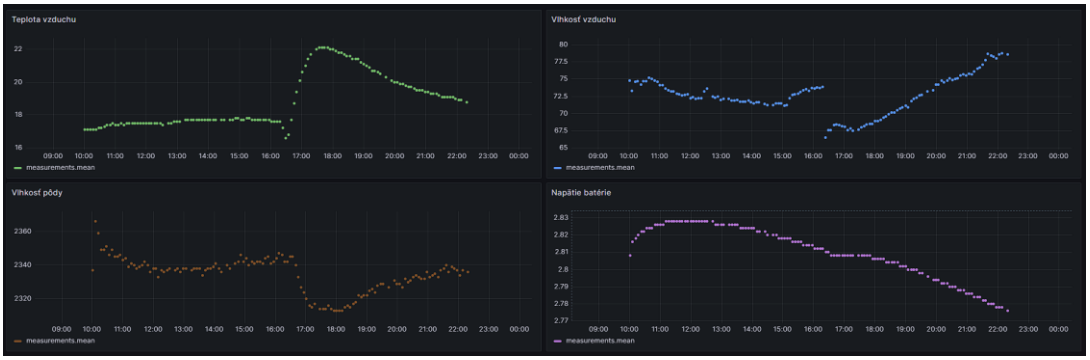


Obrázok 3.5 Výsledný Flow v Helium konzole

3.2.3 Overenie správnej funkčnosti

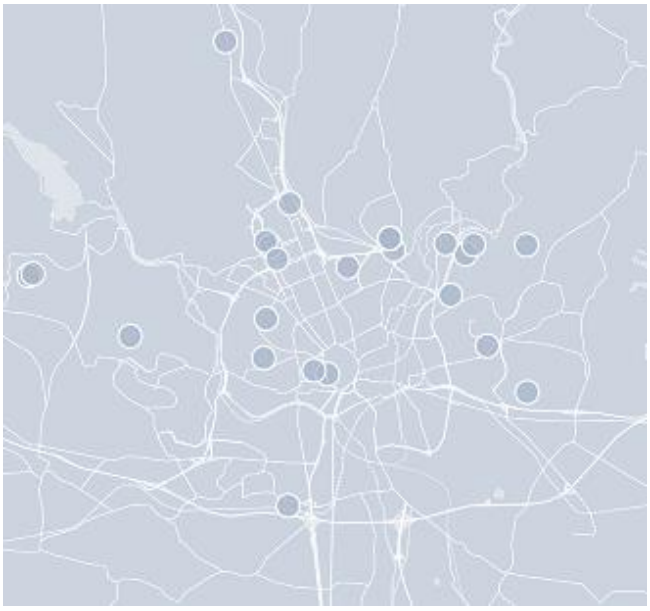
Po nahraní zdrojového kódu do vývojovej dosky Heltec CubeCell AB02A, pripojení všetkých periférií, napájania a vykonaním všetkých bodov z predošlých kapitol by sa vývojová doska mala prihlásiť do LoRaWAN siete Helium a odosielať namerané dáta zo senzorov, ktoré je možné zobraziť v Grafane. Odkaz na verejný dashboard pre túto bakalársku prácu sa nachádza na nasledujúcej adrese:

```
http://kapia.duckdns.org:6091/public-dashboards/0ec67c4c9a7d454599a27d137b33435c
```



Obrázok 3.6 Zobrazenie verejných dát zo senzorov

V Helium konzole v sekcii Devices je možné zistiť, či sa zariadenie pripojilo do IoT LoRaWAN siete, ktorý hotspot prijal aký packet, koľko DC stál prenos dát a ďalšie údaje. Zaujímavá je podľa mňa sekcia Coverage, ktorá zobrazuje pokrytie hotspotmi v blízkosti vybraného zariadenia. Je možné vidieť, že na ktorý hotspot sú najčastejšie posielané dáta podľa počtu packetov a intenzitu signálu. Na mape pokrytia je jednoduché vidieť, na akú vzdialenosť je možné z takéhoto maličkého zariadenia odosielať dáta. Hotspot s názvom „Atomic Hotpink Pelican“, ktorý je na mape úplne naľavo v časti Brno-Bystrc, je odo mňa vzdialený viacej ako 9 km. Sila signálu je -112.87 dBm a prijal 16 packetov zo 169. Veľmi pôsobivé, keď si zoberieme, že také WiFi pripojenie pod - 90 dBm vôbec nefunguje na mobilných zariadeniach.



Obrázok 3.7 Mapa pokrytia hotspotmi

3.2.4 Porovnanie platforiem Home Assistant, OpenHAB a Grafana

Platformy Home Assistant (HA) a OpenHAB sú najčastejšie používané v oblasti domácej automatizácie, zatiaľ čo Grafana sa používa na monitorovanie a vizualizáciu dát. V tejto práci bola Grafana použitá na monitorovanie a vizualizáciu nameraných dát zo senzorov. Dáta pre teplotu, vlhkosť vzduchu, napätie batérie a vlhkosť pôdy je možné zobraziť na verejnom Dashboarde, ktorý je prístupný cez verejnú webovú adresu odkiaľkoľvek. Pre administrátora je taktiež k dispozícii súkromný Dashboard, na ktorom je zobrazené väčšie množstvo informácií, ako intenzita signálu, približná poloha hotspotov a iné. Touto funkcionalitou nedisponuje HA ani OpenHAB. Ďalej je možné pomocou Grafany taktiež odosielať notifikácie cez e-mail, SMS alebo nejakú komunikačnú aplikáciu, ako aj pri HA alebo OpenHAB. V tomto projekte bol použitý iba OpenHAB na odosielanie notifikácií cez Telegram pri nízkom napätí batérie.

Na Dashboardoch HA a OpenHAB je podobne ako pri Grafane zobrazená hodnota nameraných veličín z vývojovej dosky, ale implementácia tejto funkcie sa medzi nimi značne líši. Kým pri HA stačilo pridať pár riadkov kódu do konfiguračného súboru a naklikať MQTT integráciu, v OpenHAB bola potrebná zložitejšia konfigurácia. Veci (Things), kanály (Channels), prepojenia (Bindings) a položky (Items) je potrebné medzi sebou správne nastaviť vo webovom rozhraní a taktiež ako v prípade HA pridať MQTT integráciu. Osobne mi pridanie pár riadkov v HA prišlo oveľa jednoduchšie. Myslím si, že menej skúsení používatelia budú rovnakého názoru.

Jedna vec, čo sa mi ale páčila na OpenHAB, je jednoduché nastavenie integrácie notifikácií cez Telegram. Stačilo len pridať Telegram integráciu, nastaviť token a pomocou blokového diagramu cez Blokly som jednoducho nastavil presne to, čo potrebujem. V prípade nízkeho napätia batérie je používateľovi odoslaná notifikácia, ktorá demonštruje funkčnosť Telegram integrácie. V niektorých aspektoch je možno OpenHAB zložitejší ako HA, ale ponúka hlbšiu možnosť prispôsobenia jednotlivých vecí. V prípade narazenia na problém ale môže byť nevýhoda menšia komunita OpenHAB oproti HA (6 000 členov oproti 240 000 členov).

Odporúčam teda vyskúšať obidve platformy na domácu automatizáciu a rozhodnúť sa, ktorá Vám viacej vyhovuje. Vizualizačný nástroj Grafana však nie je možné nahradiť iba jedným z nich a myslím si, že najlepšie riešenie pre podobný projekt, ako je tento, je využitie platformy Grafana spolu s jednou z dvoch spomínaných platforiem na domácu automatizáciu.

ZÁVER

Účelom tejto bakalárskej práce bolo naprogramovať vývojovú dosku Heltec CubeCell AB02A s rôznymi senzormi, realizovať konštrukciu a umiestnenie vývojovej dosky do vhodného krytu, zapojenie senzorov a antény, pripojenie do vhodnej LoRaWAN siete a prepojenie rôznych technológií medzi sebou. Výsledok nameraných dát je jednoducho prezentovaný používateľovi na verejnej webovej adrese.
Výber medzi LoRaWAN sieťou, ktorá bola použitá v tomto projekte, bol nakoniec ovplyvnený lepším pokrytím sieťou Helium oproti sieti The Things Network (TTN).

Demonštrovaná bola technológia Docker kontajnerov, ktorá uľahčila inštaláciu a konfiguráciu jednotlivých aplikácií, a tiež bola porovnaná jej implementácia v operačnom systéme Unraid (ktorý bol použitý v tomto projekte) a Raspberry Pi OS Lite.

Na prekonanie problémov so sprístupnením verejného dashboardu platformy Grafana na sieti s dynamickou verejnou IPv4 adresou boli v projekte demonštrované dve riešenia. Prvým z nich je využitie služby DuckDNS, ktorá slúži na aktualizáciu dynamickej doménovej adresy. Druhé riešenie sa zaoberá konfiguráciou presmerovania portov na domácom routeri značky TP-Link.

Taktiež bola zrealizovaná konštrukcia a umiestnenie vývojovej dosky do vhodného krytu, zapojenie a porovnanie rôznych senzorov vrátane ich komplikácií a porovnanie odlišných účelov platforiem Grafana, Home Assistant a OpenHab.

Výsledkom tejto práce je systém, ktorý periodicky monitoruje stav pôdy a okolitého prostredia. Výsledky sú prezentovaný formou grafov s možnosťou zobrazenia historického stavu a odoslanie notifikácie upozorňujúce na nízke napätie integrovanej batérie. Dá sa povedať, že všetky požiadavky pre splnenie bakalárskej práce boli splnené.

LITERATÚRA

[1] LIME TECHNOLOGY, INC. *Unraid Documentation* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://docs.unraid.net/

[2] *IOTstack* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://sensorsiot.github.io/IOTstack/Basic_setup/

[3] ORACLE CORPORATION. *What is Docker?* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://www.oracle.com/cz/cloud/cloud-native/container-registry/what-is-docker/

[4] DOCKER, INC. *What is a Container?* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://www.docker.com/resources/what-container

[5] *What is MQTT? A practical introduction* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://www.opc-router.com/what-is-mqtt/

[6] *IoT Energy Meter with Cayenne Dashboard using PZEM-004T v3 and Wemos D1 Mini* [online]. 2022 [cit. 2024-05-19]. Dostupné z: https://createlabz.store/blogs/createlabz-tutorials/iot-energy-meter-featuring-cayenne-dashboard-using-pzem-004t-v3-and-wemos-d1-mini

[7] *Mosquitto : MQTT - Overview* [online]. 2020 [cit. 2024-05-09]. Dostupné z: https://medium.com/@bhagvankommadi/mosquitto-mqtt-2a352bd8f179

[8] CHACZKO, Zenon a Robin BRAUN. Learning data engineering: Creating IoT apps using the node-RED and the RPI technologies. In: *2017 16th International Conference on Information Technology Based Higher Education and Training (ITHET)* [online]. IEEE, 2017, s. 1-8 [cit. 2024-05-08]. ISBN 978-1-5386-3968-9. Dostupné z: doi:10.1109/ITHET.2017.8067827

[9] *InfluxDB: Introduction* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://www.stackhero.io/en/services/InfluxDB/documentations/Introduction

[10] A MEDIUM CORPORATION. *What is Grafana?* [online]. 2023 [cit. 2024-05-08]. Dostupné z: https://medium.com/@MetricFire/what-is-grafana-8de44d241765

[11] ECLIPSE FOUNDATION AISBL. *Concepts | openHAB* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://www.openhab.org/docs/concepts/

[12] *What is Home Assistant, how does it work, and what do you need to get started?* [online]. 2023 [cit. 2024-05-12]. Dostupné z: https://www.pocket-lint.com/what-is-home-assistant-how-does-it-work/

[13] *Installation - Home Assistant* [online]. 2024 [cit. 2024-05-12]. Dostupné z: https://www.home-assistant.io/installation/

[14] SEMTECH CORPORATION. *LoRa Connect Transceiver, SX1262, +22dBm for Global* [online]. 2021 [cit. 2024-05-08]. Dostupné z: https://www.semtech.com/products/wireless-rf/lora-connect/sx1262

[15] CHENGDU HELTEC AUTOMATION TECHNOLOGY CO., LTD. *ASR Microelectronics ASR605x (ASR6501, ASR6502): development platform for PlatformIO* [online]. 2020, 2023 [cit. 2024-05-08]. Dostupné z: https://github.com/HelTecAutomation/platform-asrmicro650x

[16] CHENGDU HELTEC AUTOMATION TECHNOLOGY CO., LTD. *CubeCell 1/2AA Node (HTCC-AB02A)* [online]. 2020 [cit. 2024-05-08]. Dostupné z: https://heltec.org/project/htcc-ab02a/

[17] CHENGDU HELTEC AUTOMATION TECHNOLOGY CO., LTD. *Heltec CubeCell - PlatformIO v6.1 documentation* [online]. 2020 [cit. 2024-05-08]. Dostupné z: https://docs.platformio.org/en/stable/platforms/heltec-cubecell.html

[18] CHENGDU HELTEC AUTOMATION TECHNOLOGY CO., LTD. *CubeCell GPS-6502 (HTCC-AB02S)* [online]. 2020 [cit. 2024-05-08]. Dostupné z: https://heltec.org/project/htcc-ab02s/

[19] CHENGDU HELTEC AUTOMATION TECHNOLOGY CO., LTD. *CubeCell Dev-Board Plus (HTCC-AB02)* [online]. 2020 [cit. 2024-05-08]. Dostupné z: https://heltec.org/project/htcc-ab02/

[20] AOSONG ELECTRONICS CO.,LTD. *DHT22 (DHT22 also named as AM2302): Digital-output relative humidity & temperature sensor/module* [online - PDF]. 2009 [cit. 2024-05-08]. Dostupné z: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[21] WAVESHARE ELECTRONICS. *DHT22 Temperature-Humidity Sensor* [online]. 2021 [cit. 2024-05-08]. Dostupné z: https://www.waveshare.com/wiki/DHT22_Temperature-Humidity_Sensor

[22] *Soil Moisture Sensor Module* [online]. 2020 [cit. 2024-05-08]. Dostupné z: https://components101.com/modules/soil-moisture-sensor-module

[23] *Interfacing Capacitive Soil Moisture Sensor with Arduino* [online]. 2024 [cit. 2024-05-13]. Dostupné z: https://lastminuteengineers.com/capacitive-soil-moisture-sensor-arduino/

[24] ARDUINO. *Arduino Documentation* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://docs.arduino.cc/

[25] SATROM, Brandon. EETECH MEDIA, LLC. *Demystifying LoRa and LoRaWAN Wireless Network Protocols* [online]. 2022 [cit. 2024-05-08]. Dostupné z: https://www.allaboutcircuits.com/technical-articles/demystifying-lora-network-and-lorawan-network-wireless-network-protocols

[26] *What are LoRa and LoRaWAN?* [online]. 2015 [cit. 2024-05-08]. Dostupné z: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/

[27] SIGNALBOOSTERS. *LoRaWAN Explained* [online]. 2023 [cit. 2024-05-08]. Dostupné z: https://www.signalboosters.com/blog/lorawan-explained/

[28] LEIBBRANDT, Alexis. AKENZA AG. *Everything you need to know about LoRaWAN* [online]. 2023 [cit. 2024-05-08]. Dostupné z: https://akenza.io/blog/lorawan-explained

[29] LORA ALLIANCE. *What is LoRaWAN Specification* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://lora-alliance.org/about-lorawan/

[30] *LoRa World Record Broken: 832km/517mi using 25mW* [online]. 2020 [cit. 2024-05-08]. Dostupné z: https://www.thethingsnetwork.org/article/lorawan-world-record-broken-twice-in-single-experiment-1

[31] NOVA LABS INC. *Helium Documentation* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://docs.helium.com

[32] *Learn | The Things Network* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://www.thethingsnetwork.org/docs/

[33] SENSING IOT & TELEMETRY. *TTN Coverage* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://ttnmapper.org/heatmap

[34] NOVA LABS INC. *Helium Hotspots Map* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://explorer.helium.com

[35] LIME TECHNOLOGY, INC. *Quick install guide | Unraid Docs* [online]. 2024 [cit. 2024-05-08]. Dostupné z: https://docs.unraid.net/unraid-os/getting-started/quick-install-guide

[36] LEARN EMBEDDED SYSTEMS. *Easy Raspberry Pi IoT Server* [online]. 2022 [cit. 2024-05-08]. Dostupné z: https://learnembeddedsystems.co.uk/easy-raspberry-pi-iot-server

ZOZNAM SYMBOLOV A SKRATIEK

Skratky:

NAS	Network-attached storage
VM	Virtual machine
HDD	Hard Disk Drive
SSD	Solid-state drive
RAID	Redundant array of independent disks
Synology DSM	Synology DiskStation Manager
OS	Operačný systém
VPS	Virtuálny privátny server
MQTT	Message Queuing Telemetry Transport
M2M	Machine-to-machine
TLS	Transport Layer Security
SSL	Secure Sockets Layer
IoT	Internet of Things
WAN	Wide Area Network
SQL	Structured Query Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
SaaS	Software-as-a-Service
TSDB	Time Serie Database
NTP	Network Time Protocol
DDNS	Dynamic Domain Name System
NAT	Network Address Translation
SoC	System on a chip
MCU	Microcontroller unit
SMA	SubMiniature version A
NTC	Negative Temperature Coefficient
OTP	One time programmable memory
IDE	Integrated Development Environment
OSI	Open Systems Interconnection Reference model
P2P	Peer-to-peer
PoC	Proof-of-Coverage
JSON	JavaScript Object Notation

TZ	Time zone
IC	Integrated circuit
PCB	Printed circuit board
USB	Universal Serial Bus
MAC	Media Access Control
PoC	Proof-of-Coverage
BIOS	Basic Input Output System

Symboly:

U	napätie	(V)
I	prúd	(A)
f	frekvencia	(Hz)

ZOZNAM PRÍLOH

PŘÍLOHA A - OBSAH ELEKTRONICKEJ PRÍLOHY 58

Příloha A - Obsah elektronickej prílohy

Blokový diagram a pinout diagram vývojovej dosky použitej v tomto projekte je možné nájsť v elektronickej prílohe v súboroch s názvami „HTCC-AB02A_SchematicDiagram.pdf“ a „HTCC-AB02A_PinoutDiagram.pdf“. Zdrojový kód pre vývojovú dosku napísaný v Arduino IDE je uložený v súbore s názvom „IoT_LoRaWAN_vpt.ino“. V prílohe sa tiež nachádzajú obrázky ilustrujúce správnu funkčnosť celého systému.