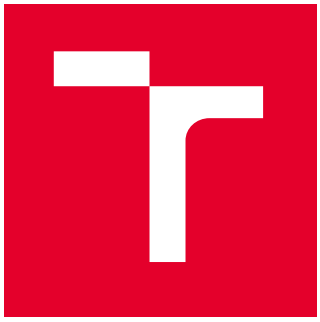


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

FIRMWARE MĚŘICÍ STANICE PRO BEZDRÁTOVÝ SBĚR DAT PŘES LORAWAN

MEASURING STATION FIRMWARE FOR WIRELESS DATA COLLECTION VIA LORAWAN

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Vít Jánoš**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radovan Juráň**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Vít Jánoš ***ID:*** 220892

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Firmware měřicí stanice pro bezdrátový sběr dat přes LoRaWAN

POKYNY PRO VYPRACOVÁNÍ:

Těžiště pozornosti internetu věcí a bezdrátových datových přenosů je kladeno především do oblasti chytrých měst a průmyslu, nicméně své uplatnění tyto systémy sběru dat nachazejí i v zemědělství, lesnictví, geologii a dalších oborech. Cílem práce je realizace firmware prototypu terénní měřicí stanice s využitím SoC (system-on-a-chip) STM32WL55 na příslušeném vývojovém kitu STM Nucleo a protokolu LoRaWAN.

1) Prostudujte LoRaWAN a vývojový kit s STM32WL55 [1].
2) S využitím uvedeného kitu implementujte firmware měřicí stanice, který bude umět a) odečítat data z čidel CO2 a teploty, b) řídit motor odvětrávání komory a ventilátor, c) připojit se k bráně (gateway) a odeslat testovací data, d) zálohovat data na SD kartu s časovou značkou. Podrobnosti viz [2].
3) Integrujte prototyp do existující sítě a vyzkoušejte datové přenosy měřených dat (868 MHz).
4) Optimalizujte návrh a proveďte optimalizaci spotřeby energie, případně navrhněte kroky k jejímu dosažení.
Výstupem je program, který běží na vývojovém kitu.

DOPORUČENÁ LITERATURA:

[1] STM32WL5x: Ultra-low power multi-modulation wireless STM32WL5x microcontrollers [online]. [cit. 2021-9-13]. Dostupné z: https://www.st.com/en/microcontrollers-microprocessors/stm32wl5x.html
[2] JURÁŇ, R.; POVALAČ, A. Field Sensor Network for Microclimatological Measurements. In 2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). International Congress on Ultra Modern Telecommunications and Control Systems and Workshops. Brno: IEEE, 2020. s. 149-153. ISBN: 978-1-7281-9281-9. ISSN: 2157-023X.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: Ing. Radovan Juráň

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

ABSTRAKT

Tato práce se zabývá vývojem software s mikrokontrolérem STM32WL55-JC1. Vývojový kit byl zvolen především kvůli jeho LPWAN možnostem. Výsledný prototyp komunikuje prostřednictvím protokolu LoRaWAN, na aplikační server odesílá přes veřejnou síť The Things Network měřená data, jimiž jsou koncentrace oxidu uhličitého, teplota a tok oxidu uhličitého z půdy. Data jsou také společně s časovou značkou zálohována lokálně na SD kartu. Software byl navržen s ohledem na použití komorové měřící metody, která je v textu také popsána.

KLÍČOVÁ SLOVA

LoRaWAN, LPWAN, tok plynu, oxid uhličitý, IoT, mikrokontrolér, jazyk C

ABSTRACT

This thesis deals with software development for STM32WL55-JC1 MCU. Development kit was chosen primarily bacause of its LPWAN possibilities. Final prototype communicates on LoRaWAN network. Data are sent through public network from The Things Network, these data are concentrations of carbon dioxide, temperatures and carbon dioxide flux. Data are also backed up locally on SD card along with timestamp. The software was designed with the use of the chamber measurement method in mind, which is also described in the text.

KEYWORDS

LoRaWAN, LPWAN, flux, carbon dioxide, IoT, microcontroller, C language

JÁNOŠ, Vít. *Firmware měřicí stanice pro bezdrátový sběr dat přes LoRaWAN*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 65 s. Bakalářská práce. Vedoucí práce: Ing. Radovan Juráň

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Vít Jánoš
VUT ID autora:	220892
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Firmware měřicí stanice pro bezdrátový sběr dat přes LoRaWAN

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu semestrální práce panu Ing. Radovanu Juráňovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato práce se zabývá vývojem firmwaru pro měřicí stanici, jejímž úkolem je měřit koncentraci oxidu uhličitého vyvěrajícího z půdy a následně bude naměřená data odesílat přes IoT LPWAN protokol LoRaWAN.

Ačkoliv je hlavním předmětem této práce software, tak u mikrokontrolérů nelze plně oddělit hardwarovou a softwarovou část. Je tedy důležité zvolit vhodné senzory a integrovat je do systému jako celku a přizpůsobit jim jeho návrh. Po zvolení periferií je nutné zprovoznit jejich komunikaci s MCU. Dalším cílem je z naměřených dat vypočítat tok plynu z půdy, který je hlavní sledovanou veličinou. Předpokládá se využití komorové metody měření. Z toho vyplývá nutnost simulace ideálních měřících podmínek, které jsou reprezentovány především přirozeným vánkem. Ten bude simulovat malý ventilátor běžící na nízké otáčky. Jelikož se jedná o uzavřený systém, je vyžadováno vyvětrání komory před měřením. To zajistí servomotor, který bude otvírat víko komory a zmíněný ventilátor, kterému budou pro tento účel zvýšeny otáčky. Po měření budou data odeslána přes LoRaWAN pro ukládání a zobrazení obsluze. Pozornost si také zaslouží dostupné možnosti vizualizace dat.

V následujících řádcích je představen především protokol LoRaWAN, ale také je prostor věnován alternativním technologiím pro srovnání. Poté je srovnána zamýšlená komorová měřící metoda s metodou Eddy covariance, která je mimo jiné používána ve vědecké komunitě k měření emisí skleníkových plynů. Druhá kapitola se zabává samotným řešením práce. Na začátku jsou popsány vybrané části zdrojového kódu a dále ze znalostí z první kapitoly určeny provozní parametry pro komunikaci přes LoRaWAN. V další sekci přichází na řadu zpracování přijatých dat a jejich přeposlání na aplikační server. V poslední části je popsán zvolený způsob prezentace dat a předtaven backend tohoto řešení.

1 Teoretická část studentské práce

1.1 LPWAN komunikace

Internet věcí zažívá v posledních letech obrovský rozmach. Řešení postavené na této myšlence se rozrůstají do velkého množství odvětví, od zabezpečení, přes zemědělství až po chytré domy či města. Tyto aplikace vyžadují specifické nároky jako dlouhý dosah a nízkou spotřebu. Existující technologie jako Bluetooth nebo Zigbee, které nabízí pouze krátký dosah, naopak mobilní sítě nabízející vysoký dosah a vysokou přenosovou rychlost, která není nezbytná pro IoT, spotřebovávají vysoké množství energie [1]. Proto po roce 2009, kdy byla založena firma Sigfox, začaly vznikat komunikační technologie na bázi Low Power Wide Area Network navrženy pro odesílání malého množství dat nízkou rychlostí na dlouhé vzdálenosti, a to především při chodu na baterii. Tento standard v dnešní době reprezentují především technologie LoRaWAN, NB-IoT a Sigfox, který je ovšem touto dobou v úpadku. Ačkoliv je obecný účel všech technologií stejný, existuje mezi nimi spoustu technických rozdílů.

1.1.1 LoRaWAN

LoRaWAN je LPWAN síťový protokol vyvinutý pro IoT. Podporuje obousměrnou half-duplex komunikaci, koncové (end-to-end) šifrování a lokalizaci. Počet zpráv za den je neomezený. Nabízí také možnost vytvoření privátní sítě. Veškeré verze protokolů LoRaWAN definuje LoRa Alliance v dokumentech LoRaWAN Specification, které jsou volně dostupné na jejích webových stránkách.

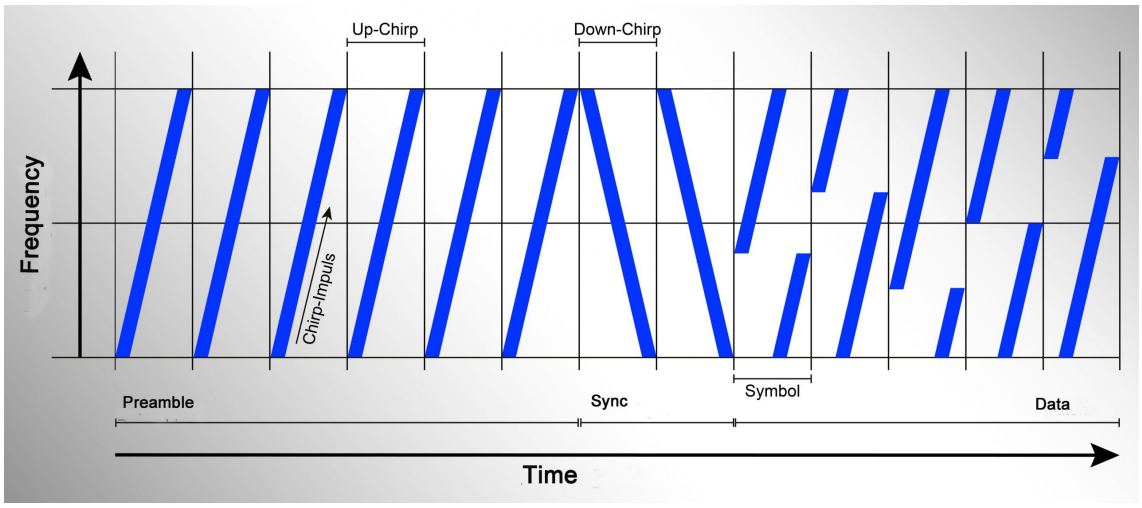
LoRa

LoRa je fyzická vrstva, nad kterou je LoRaWAN postaven. Jde o modulační techniku odvozenou z CSS. Komunikace je obousměrná. Zprávy se kódují do rádiových vln při užití tzv. chirp pulzů. Chirp je spojitý signál, kterému v čase buď roste (upchirp) nebo klesá (downchirp) frekvence. Samotný rámec je zakódován následovně: nejdříve je vysílána preambule o délce 8 up-chirp symbolů, poté 2 downchirp synchronizační symboly, datové symboly a následně volitelné CRC [2].

V souvislosti s technologií LoRa mluvíme o následujících parametrech:

Spreading Factor

Spreading Factor je veličina udávající počet bitů nesených každým symbolem. V pásmu 863-870 je daný Spreading Factor od 7 do 12. Volíme vyšší při větším rušení



Obr. 1.1: Časový průběh LoRa rámce [2]

nebo vzdálenosti od gateway. Vyšší SF prodlouží ToA (Time on Air), sníží bitovou rychlost, ale zvýší dosah.

Chip

Symbol má 2^{SF} hodnot, tj. při SF 7 128 chipů.

Coding rate

Udává poměr bitů nesoucích zprávu a celkového počtu bitů pro rekonstrukci chyb. Zbývající bity tvoří redundanci. Čím je vyšší počet redundantních bitů, tím je vyšší šance na rekonstrukci zprávy.

Coding rate	
1	4/5
2	4/6
3	4/7
4	4/8

Tab. 1.1: Převodní tabulka coding rate [5]

Dle tabulky 1.1 můžeme převést konvenční hodnotu v levém sloupci na vypovídající hodnotu. Např. pokud CR=1, potom počet bitů nesoucích informaci je $\frac{4}{5}SF$

Chip rate

Jednotkou šířky pásma BW je Hertz. Tato veličina je shodná s chip rate:

$$BW = Rc = chiprate[chips/s] \tag{1.1}$$

Symbol rate

$$R_s = BW/2^{SF} = Rc/2^{SF}[symbol/s] \tag{1.2}$$

Data rate (bit rate)

$$R_b = SF * \frac{BW}{2^{SF}} * \frac{4}{4+CR}[b/s] \tag{1.3}$$

Duty cycle

Duty cycle, doslova přeloženo jako střída, je veličina údávající poměr času, který zařízení stráví vysíláním, vůči celkové časové periodě.

$$DC = \frac{t_{TX}}{T} \tag{1.4}$$

Time on Air

Další veličinou je ToA (Time on Air), která udává čas, který zabere doručení zprávy z koncového zařízení na server.

$$ToA = t_{packet} = t_{preamble} + Tpayload \tag{1.5}$$

Kde t jsou příslušné časy preambule, hlavičky a payloadu, které jsou vypočteny následovně.

$$t_{preamble} = (n_{preamble} + 4,25) \cdot T_s \tag{1.6}$$

Kde n je délka preambule a T $_s$ doba trvání symbolu, což je převrácená hodnota symbol rate.

$$t_{payload} = T_s(8 + max(ceil(\frac{8PL - 4SF + 28 + 16CRC - 20H}{4(SF - 2DE)}))(CR + 4), 0) \tag{1.7}$$

Kde PL představuje payload v bajtech a CR coding rate. Následující hodnoty jsou booleovské, reprezentují tedy aktivaci či deaktivaci daného parametru - CRC je kontrolní součet, DE je Low Data Rate Optimize a H hlavička, u které je aplikována obrácená logika, kde 0 znamená povoleno.

Vysílací interval

Po definování duty cycle a ToA můžeme zavést interval odesílání zprávy,

$$T_{TX} = \frac{ToA}{DC} - ToA \tag{1.8}$$

který reprezentuje frekvenci odesílání zpráv, což využijeme kvůli místním regulacím.

LoRa v ČR, resp. v EU, pracuje ve frekvenčním pásmu 433MHz nebo častěji 868MHz. Zařízení ve frekvenčním pásmu EU863-870 ISM musí implementovat tři základní kanály: 868,1; 868,3; 868,5MHz o šířce pásma 125kHz a 5 dalších volitelných kanálů, např. dle implementace operátora. Koncová zařízení mění vysílací kanály v pseudonáhodném pořadí pro každý přenos (proto Spread Spectrum v CSS). Tyto změny frekvence činí systém odolnější vůči rušení. Čas, po který není možné vysílat z důvodu změny frekvence se nazývá Hop Time. Přenosové rychlosti jsou v rozmezí 250b/s až 11kb/s a maximální užitečný payload 51B-242B dle konfifurace SF a šířky pásma. Typické hodnoty SNR jsou okolo -120dBm a SNR mezi -20 a +10dB [3].

Prostředí	**Dosah [km]**
Města	2-5
Venkov	5-15
Viditelný bod	>15

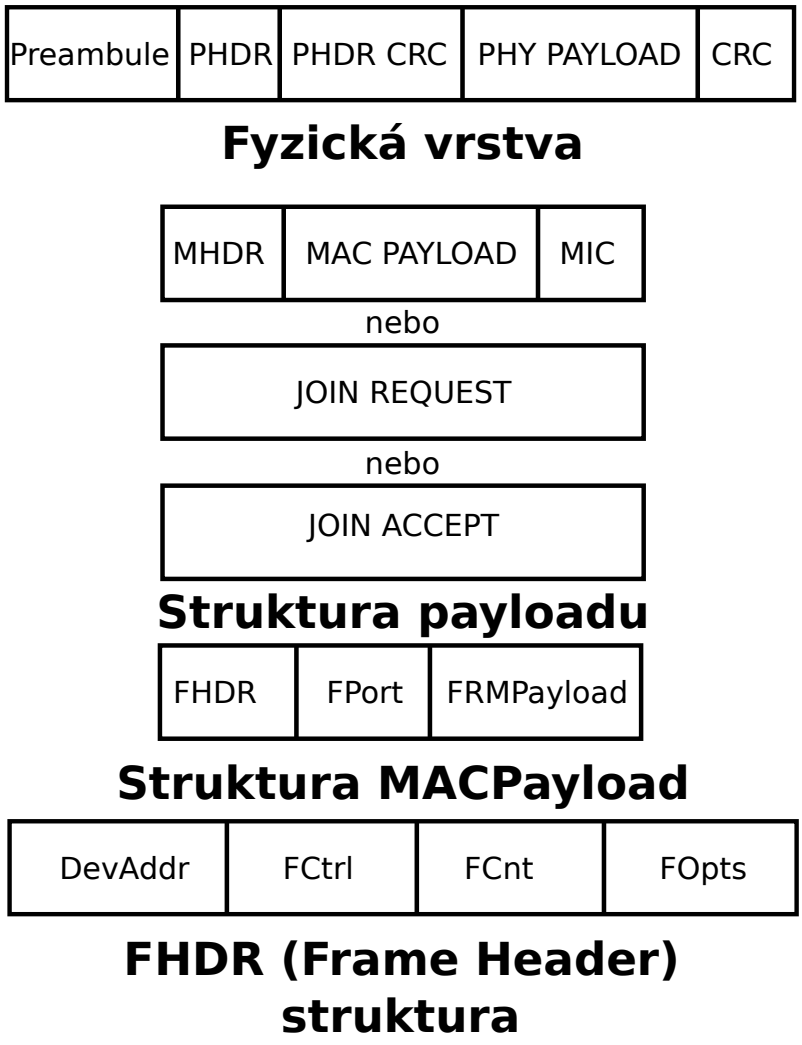
Tab. 1.2: Orientační dosahy v různých prostředích [5]

Architektura

LoRaWAN náleží druhé a třetí vrstvě modelu ISO/OSI. Koncová zařízení komunikují s bránami právě pomocí LoRaWAN protokolu. Každá brána je připojena do sítě internet, kde se nachází síťový server, který směruje zprávy na požadované místo, např. na aplikační server, kde dochází k dalšímu zpracování dat, jejich vizualizaci atd.. Sítě LoRaWAN jsou založeny na ALOHA protokolu, takže koncová zařízení nemusí být v kontaktu se specifickou bránou. Zprávu přijmou všechny v dosahu a duplicitní zprávy jsou ošetřeny serverem [4].

Druhy zpráv

Jak již bylo zmíněno, základní dělení zpráv je downlink a uplink, dle směru komunikace. Uplink zprávy vysílá koncové zařízení jedné nebo více bránám. Ať zpráva náleží join serveru či aplikačnímu serveru, síťový server zprávu přepošle správnému příjemci. Downlink zprávy posílá síťový server vždy pouze jednomu konkrétnímu



Obr. 1.2: Formát LoRaWAN paketu, inspirováno [8]

koncovému uzlu skrze jednu bránu. Zprávy mohou pocházet jak od join serveru, tak od aplikačního serveru.

Dále se zprávy dělí dle účelu a potvrzování. Tato informace o druhu zprávy je také přenášena v hlavičce. Jedná se o následující druhy.

Join Request (žádost o připojení) vždy odesílá zařízení síťovému serveru. Od verze 1.0.4 se tyto zprávy přeposílají join serveru. Dříve se posílaly jako ostatní zprávy aplikačnímu serveru. Tyto zprávy nejsou šifrovány.

Join Accept (potvrzení připojení) je odpovědí na žádost o připojení, takže opět od verze 1.0.4 ji odesílá join server (dříve síťový server) konkrétnímu zařízení.

Rejoin Request (žádost o znovupřipojení) odesílá koncové zařízení síťovému serveru. Existují tři druhy rejoin zpráv: 0, 1, 2. Tyto zprávy se používají k ustanovení nové relace pro koncové zařízení. Odpovědí je Join Accept.

Mimo výše uvedené zprávy, které se používají při OTAA (Over The Air Acti-

vation), existují čtyři druhy datových zpráv. Jedná se o kombinaci dvou vlastností - potvrzování a směr. Zprávy jsou tedy buďto Unconfirmed Data Up, Unconfirmed Data Down, Confirmed Data Up nebo Confirmed Data Down.

Posledním druhem zprávy je proprietární zpráva, která je určena k implementaci nestandardních formátů zpráv.

Datová zpráva může obsahovat libovolný počet MAC příkazů. Zprávy mohou nést jak aplikační data, tak MAC příkazy, obojí v oddělených polích. MAC příkazy mohou být odesílány buď ve Frame options field (FOpts) nebo ve frame payload field (FRMPayload), nikoliv však v obou současně. Aplikační data se nachází pouze v buňce FRMPayload a pokud se zde nachází, tak zde již nemohou být žádné MAC příkazy. Při posílání MAC příkazů v FOpts je jedinou podmínkou, aby jejich délka nebyla větší než 15B. Při použití pole FRMPayload se specifikuje parametr FPort, 0 značí MAC příkazy a nenulová hodnota aplikační data [8].

Třídy

LoRaWAN definuje tři třídy. Základní je A, kterou zbylé dvě rozšiřují.

Třída A

Třída A podporuje half-duplex komunikaci mezi zařízením a gateway. Uplink zprávy mohou být odeslány kdykoliv. Koncové zařízení poté otevře dvě časová okna ve specifikovaných chvílích (v praxi RX1 - sekunda po TX, RX2 dvě sekundy po TX). Server může v jednom z nich odpovědět, ale neměl by využívat obě, protože pokud koncové zařízení obdrží zprávu v prvním okně, druhé již neotevře. Koncové uzly implementující tuto třídu se vyznačují nejnižší spotřebou. Většinu času se nacházejí ve spánku, a proto je často nalezneme v zařízeních napájených z baterie.

Třída B

Tato třída rozšiřuje třídu A o plánované otvírání přijímacích oken. Gateway vysílá beacon rámce v pravidelných intervalech nazývaných beacon period, které slouží k synchronizaci. Čas, po který je zařízení schopno přijímat downlink zprávy se nazývá ping slot. Životnost baterie je zde typicky nižší než u třídy A, kvůli aktivitě během beacon a ping slotů.

Třída C

Třída C rozšiřuje třídu A ponecháním neustále otevřeného okna pro příjem, pokud zrovna neprobíhá uplink. Tato konfigurace nabízí nulové downlink zpoždění. Tato

výhoda je vyvážena vyšší spotřebou energie, protože jsou prakticky neustále aktivní. Jsou tedy nejčastěji napájeny ze sítě [6].

1.1.2 Sigfox

Sigfox je LPWAN technologická společnost nabízející služby založené na jejích patentovaných technologiích. Vytváří síť vlastních základen vybavených softwarově definovanými rádii, které jsou připojeny k serverům pomocí IP sítě. Komunikace mezi koncovým zařízením a základnou je zprostředkována binárním klíčováním fázovým posuvem (BPSK) za použití ultra úzkých kanálů (100Hz). Podobně jako LoRaWAN využívá nelicencované ISM pásmo (868MHz v Evropě, 915MHz v Severní Americe, 433MHz v Asii). Díky ultra úzkým kanálům je využití šířky pásma vysoce efektivní a má velmi nízkou hladinu šumu. Značné omezení je v počtu zpráv. Za den je možné odeslat 144 uplink zpráv (z toho jsou 4 rezervovány protokolem) o maximální velikosti 12 bajtů a pouze 4 downlink zprávy o maximální velikosti 8 bajtů. Z toho vyplývá, že potvrzování zpráv není umožněno. Spolehlivost je tedy zajištěna rozložením ve frekvencích a čase. Každé zařízení vysílá signál několikrát, ve výchozím nastavení třikrát, a skrze různé kanály. Například pásmo v Evropě mezi 868,180MHz - 868,220MHz je rozděleno do 400 kanálů šířky 100Hz. Jelikož jsou základny schopné přijímat signál paralelně na všech kanálech, koncová zařízení si mohou vybrat náhodný kanál pro vysílání [1].

1.1.3 NB-IoT

Narrow Band IoT je technologie, určená k implementaci na existujících vysílačích mobilních operátorů při použití licencovaných frekvenčních pásem. Specifikuje použití Stand-Alone (např. v bloku GSM). Preferovaná je ovšem koexistence s LTE, kde jsou dvě možnosti. V operačním módu Guard-Band je NB-IoT nasazeno v ochranném pásmu LTE, jsou tedy využity jinak nevyužité zdrojové bloky. V módu In-Band jsou využity zdrojové bloky v rámci nosných LTE, NB-IoT je tedy nasazeno na úkor LTE. Šířka pásma je 180kHz, což odpovídá jednomu zdrojovému bloku LTE. Výhodou je technicky relativně jednoduchá implementace ze strany operátora. Stačí provést pouze softwarový upgrade existující infrastruktury. Ačkoliv je komunikační protokol založen na LTE, většina funkcionalit je omezena na minimum s ohledem na potřeby IoT. Síť je škálovatelná a kapacita může být rozšířena alokováním dalšího bloku pro NB-IoT. Pro uplink slouží FDMA metoda přístupu k médiu a pro downlink ortogonální FDMA (OFDMA). Signál je klíčován QPSK modulací. Maximální velikost zprávy je 1600 bajtů, což je několikanásobně více než u ostaních technologií. NB-IoT nabízí nízkou latenci srovnatelenou s LoRaWAN třídy C. Tato technologie vyniká kvalitou služeb (QoS), protože využívá licencované pásmo LTE [7].

1.2 Měřicí stanice

Klíčovým cílem této práce je prototyp měřící stanice CO_2 pro měření nárůstu koncentrace plynu v čase. V této kapitole se zaměřím na hlavní způsoby měření tohoto nárůstu. Při návrhu zařízení bude nutné zvažovat homogenitu prostředí, následně i jeho stálost či odchylky měření. V nálsedujících řádcích představím některé dosud známé metody.

1.2.1 Oxid uhličitý v půdě

Půdní CO_2 je tvořen při biologických procesech, dekompozicí odumřelé organické hmoty a respirací živých organismů. Z geologických procesů jsou stěžejní především sopečné erupce a vyvěrání minerálních vod. Koncentrace také dynamicky závisí na úniku z podloží. Dokonce mohou někdy být koncentrace CO_2 tak vysoké, že nahradí O_2 a vytvoří hypoxické podmínky (nízký obsah kyslíku) [9].

Činitelé ovlivňující koncentraci CO_2 v půdě

Hladinu oxidu uhličitého v půdě ovlivňuje spousta faktorů, např. teplota, vlhkost, biologická aktivita. Každý z těchto faktorů přispívá k výsledné koncentraci různou mírou.

Jedním z hlavních faktorů je teplota. Zdroj tepla je sluneční záření, které prohřívá půdu a teplo prostupje do hloubky. V závislosti na tepelné vodivosti půdy se mění rychlost změny teplotního spádu mezi povrchem a sledovanou hloubkou. Experimenty několika výzkumných týmů zjistily, že při růstu teploty v rozmezí 20 až 40°C je růst půdního CO_2 exponencionální.

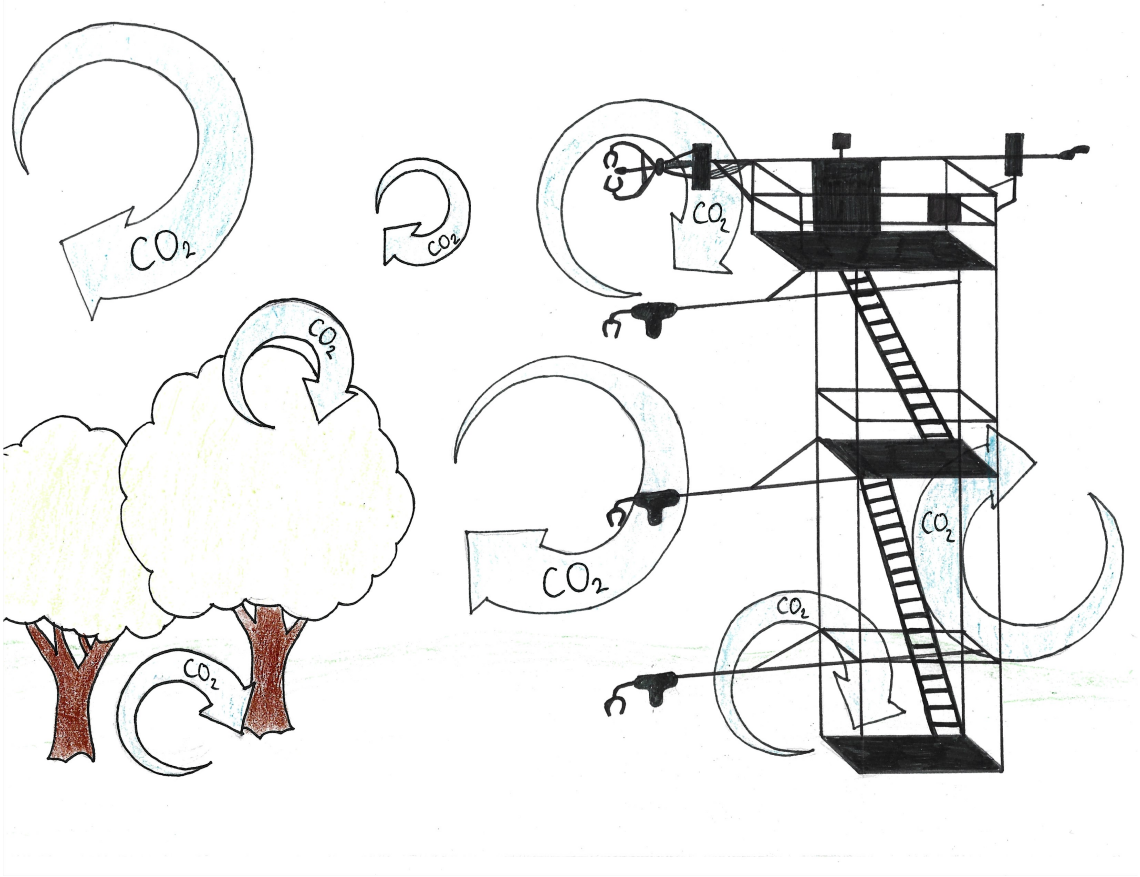
Dalším důležitým faktorem je vlhkost půdy. Nárůst obsahu vody v půdě způsobuje vyšší produkci plynu a následně jeho vyšší koncentraci. Vlivy vlhkosi se nejlépe sledují v oblasti tropického pásu, kde se střídají období sucha a dešťů.

Dalšími vlivy jsou půdní respirace, která se nejvíce projevuje v tropech, hloubka a charakter profilu půdy a v neposlední řadě zemědělské zásahy do půdy. Bylo zjišěno, že orba přispívá zvýšenému toku CO_2, protože díky nakypřené půdě je usnadněn prostup do atmosféry [10].

1.2.2 Metoda Eddy covarience

Tento způsob měření se řadí mezi pokročilé. Měří přechody plynů mezi půdou, vegetací a okolním vzduchem. Slovo eddy je v anglickém jazyce definováno jako cirkulace vzduchu způsobena fluktuací teploty, ta představuje různé teploty v různých částech dne. Konstantní fluktuace mezi dnem a nocí vytváří vánek, který způsobuje

cirkulaci vzduchu. Covarience na druhou stranu představuje společné měření změny koncentrace plynu a směru vířivého větru.



Obr. 1.3: Princip metody eddy covariance (autor)

Měřící stanice je vysoká základna, na které jsou v různých výškách umístěna měřící zařízení. Skládá se z anemometru, který měří rychlost a směr větru a z infračerveného senzoru pro měření koncentrace plynu, data jsou vzorkována řádově tisíckrát za minutu. Díky datům z několika různých výšek je možné analyzovat pohyb plynu v prostředí.

Zjednodušeně můžeme průměrný tok specifické složky vzduchu vyjádřit jako

$$\overline{N_s} = \overline{\omega' c_s'} + \int \frac{\delta \overline{c_s}}{\delta t} dz \tag{1.9}$$

kde N_s je výměna ekosystému (mol m^{-2} s^{-1}), ω je vertikální rychlost větru, c_s je molární koncentrace (molarita, látková koncentrace), čas t, vertikální vzdálenost od půdy z. Horní čára značí aritmetický průměr. První člen vztahu představuje cirkulaci vzduchu na vrcholu sledované oblasti, tedy opouští ji a část se vrací zpět, druhý člen určuje trend měření koncentrace z dané oblasti. V čase potom můžeme sledovat emise CO_2 do atmosféry a zároveň také kolik jej je znovupoužito živými

organismy [12]. Zajímavostí je, že data z celého světa jsou sdružovány ve vědecké databázi FLUXNET [11].

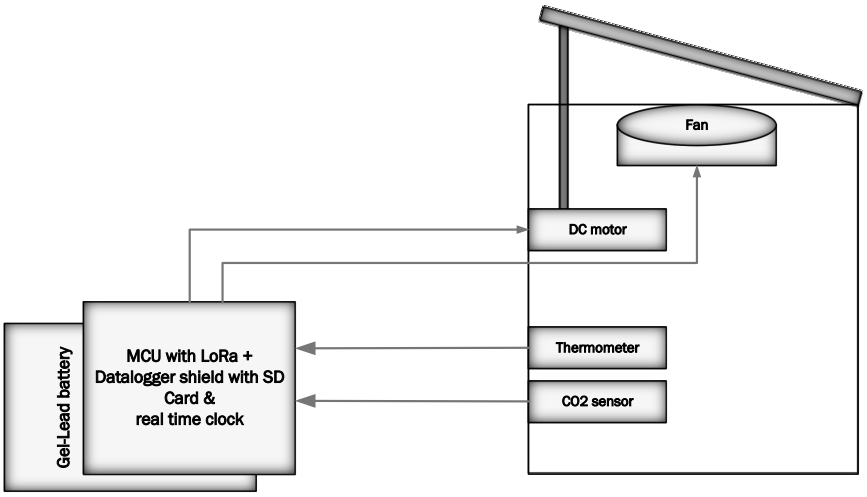
1.2.3 Uzavřená komora s ventilací

Předností tohoto řešení je především nízká cena či snadné použití a nasazení. Jedná se, narozdíl od předchozího řešení, o malý ekosystém, řádově jednotky až desítky litrů. Hlavní část tvoří jímací komora, kterou proudí vzduch a v níž vzorkujeme koncentraci po sledovaný čas. Podle [13], kde byla použita 25l komora, je doporučeno pokrýt komoru bílou látkou pro přiblížení se adiabatickému ději. Pokud chceme měřit vyvěrání plynu z půdy s minimalizací vlivu flóry, je vhodné vysekat veškerou zeleň v místě měření až k zemi. Na ideálně odnímatelný strop konstrukce je nutné nainstalovat větrák, který bude nízkou ale dostatečnou rychlostí promíchávat vzduch uvnitř. Měřící interval zvolili 10 minut se 4 opakováními, kdy za pomoci časově uzavíraných třícestných ventilů nashromáždili vzorky k pozdějšímu měření.

Produkce CO_2 za čas se vypočítá jako

$$F_{CO_2} = \frac{PV}{RTA}\frac{dC}{dt} \tag{1.10}$$

kde F_{CO_2} je množství CO_2 na jednotku plochy za čas [µmol m^{-2} s^{-1}], R je molární plynová konstatnta [J K^{-1} mol^{-1}], P je atmosferický tlak [Pa], A plocha pokryté půdy [m^2], V je objem komory [m^3], T teplota vzduchu [K] a $\frac{dC}{dt}$ změna koncentrace v komoře v čase [µmol/mol/s] [14]. Kvůli jednoduchosti a prostorovým možnostem bude předmětem této práce realizace podobného systému s uzavřenou komorou. Blokové schéma vyvíjeného systému je na obrázku 1.4.



Obr. 1.4: Blokové schéma měřící komory (autor)

1.3 Hardware

1.3.1 STM32WL55

STM32WL55 je procesor od firmy STMicroelectonics. Jedná se o první SoC řešení, které integruje mikrokontrolér a Sub-GHz vysílač v rámci jediného čipu. Mikrokontrolér je postavený na jádrech ARM Cortex-M4 a Cortex-M0, jsou dostupné jak jednojádrové, tak dvoujádrové architektury. Tyto mikrokontroléry podporují vícero modulací - LoRa, (G)FSK, (G)MSK, BPSK, umožňuje tedy vyvíjet aplikace pro různé IoT platformy jako LoRaWAN, Sigfox, W-MBUS, mioty či jiný vyhovující protokol [15].

Konektivita

Mimo zmíněné modulace a s nimi spojené IoT protokoly, čip nabízí podporuju následujících rozhraní.

USART, UART, LPUART Mikrokontrolér podporuje známý komunikační protokol USART (Universal Synchronous/Asynchronous Receiver and Transmitter), což je protokol pro plně duplexní sériovou komunikaci po dvou vodičích, které se kříží mezi piny Rx (přijaté) a Tx (odeslané). Asynchronní UART mód je vhodný pokud je přípustná menší rychlost přenosu a naopak je vyžadována nízká spotřeba energie. Synchronní mód USART nabízí pravý opak, kde je přenosová rychlost větší z důvodu externího hodinového signálu, který generuje vysílající zařízení. Přenosová linka je standardně na logicky vysoké úrovni napětí. Pokud chce zařízení vysílat, na jednu periodu sníží toto napětí na nízkou úroveň, tomuto se říká start bit. Po něm následují samotná data, která mohou mít minimálně 5 bitů a maximálně 8 bitů při použití parity a bez jejího použití 9 bitů. Další je již zmíněný volitelný paritní bit a nakonec jeden až dva stop bit(y) signalizující konec paketu. Obvykle se data posílají od LSB (nejméně významný bit) [16]. Mimo dvě USART rozhraní je zde ještě jedno LPUART (Low Power UART) rozhraní, jenž je zaměřeno na nízkou spotřebu. Pomocí tohoto rozhraní může být například zařízení probuzeno ze spánku, ať už přijetím stop bitu či libovolné vlastní zprávy. Toto rozhraní je aktivní ve všech běhových a spánkových režimech a může být probuzeno ze všech stop módů, pokud jsou LPUART hodiny nastaveny na LSE nebo HSI16 [15].

SPI (Serial Pheripheral Interface) je sériová sběrnice sloužící k propojení dvou a více zařízení v architektuře, jeden master (nadřízený) a vícero slaves (podřízených). Master generuje hodinový signál a distribuuje jej do ostatních zařízení. Toto umožňuje synchronní a plně duplexní přenos dat. Pin pro hodinový signál se nejčastěji

označuje SCLK. Dalšími používanými vodiči jsou MISO (Master In Slave Out) a MOSI (Master Out Slave In), které zprostředkovávají zmíněnou plně duplexní komunikaci. Posledním vodičem který tato sběrnice používá je SSEL (Slave Select), který slouží k výběru podřízeného uzlu, se kterým bude probíhat komunikace. U většiny zařízení je možné nakonfigurovat polaritu hodin i hranu, dle které se synchronizuje. Každé zařízení při čistém použití SPI bez protokolů vyšších vrstev vyžaduje připojení SSEL vodiče. Z toho vyplývá nevýhoda lineárního vzrůstu vyhrazených pinů pro zařízení na této sběrnici. Podobně je to s omezením na jednoho mastera. Díky nutnosti synchronizace a neexistence potvrzování je také přenosová vzdálenost značně omezena. Naopak výhody jsou v jednoduchosti. Jedná se v podstatě o spojené posuvné registry, kde k posunu dochází dle hodinového signálu. Pokud master vysílá data, posune bity s hodinovou hranou, krajní bit se přesune na výstup (pro mastera MISO) a z jeho registru se dostane přes MOSI do registru u podřízeného uzlu. Toto se děje při generování hodinového signálu master uzlem. Analogicky přenos probíhá opačným směrem [17].

I²C (Inter Integrated Circuit, /i-squared-c/) je sběrnice vybavena pouze jedním datovým vodičem. Z toho vyplývá, že je možný maximálně poloduplexní přenos. Namísto fyzického spojení u SPI I²C adresuje softwarově dvoumístnou hexadecimální adresou. Je možné adresovat až 128 zařízení současně. Vodiče, respektive piny, které toto rozhraní potřebuje ke své funkci jsou datový pin SDA a hodinový SCL a signálovou zem, na které se připojí všechna zařízení. Tato sběrnice je také vhodnější pro komunikaci na delší vzdálenosti (řádově metry) než SPI. Také je důležité zmínit, že pro správnou funkci je nutné oba signálové vodiče připojit přes 1,5kΩ pull-up rezistor k napájecímu napětí, které plní svou funkci mimo dobu přenosu, kdy zvedají napětí na úroveň logické jedničky, což je definovaný klidový stav. Přenos je synchronizován s náběžnou hranou hodinového signálu. Komunikace probíhá přerušením klidového stavu, kdy master uveden SDA do logické nuly a o dobu určenou přenosovou rychlostí později také SCL. Toto se nazývá start bit. V zápětí začne vysílat adresu uzlu, se kterým si přeje komunikovat, s bitem, který určuje, zda cílový uzel data bude i vysílat, či pouze přijímat. Následují samotná data a stop bit, který je opačný než start bit. Nejdříve je do logické jedničky uveden SCL a poté SDA. Slave potvrzuje přijetí každého rámce a pokud je žádán o data, odesílá je [18].

Nízkovýkonové režimy

MCU mimo standardní režim běhu nabízí hned několik úsporných módů pro nejlepší vyvážení doby běhu a úspory energie.

Režim spánku (Sleep mode) Hodiny CPU jsou v tomto stavu vypnuty. Všechny periferie včetně periferií procesoru mohou být aktivní a probudit CPU, pokud nastane definovaná událost nebo hardwarové přerušení.

Úsporný běhový režim (LPRun) pokud frekvence systémových hodin klesne pod 2MHz, je program vykonáván buď z SRAM nebo flash paměti za účelem minimalizace provozního proudu.

Úsporný režim spánku (LPSleep) spánek, do kterého MCU vstupuje z LPRun režimu.

Stop 0 a 1 Obsah pamětí SRAM1 a SRAM2 zůstává zachován, všechny hodiny napájené hlavním napěťovým zdrojem jsou zastaveny, mohou běžet pouze LSE (Low-speed-external) a LSI (Low-speed-internal). RTC (Reset Clock Control) může také běžet a naopak sub-GHz vysílač by měl zůstat aktivní nezávisle na CPU.

Ve Stop módu 0 zůstává hlavní napěťový regulátor zapnutý, v důsledku toho je zvýšená spotřeba el. energie oproti následujícím stop módům. Kladně je zase ovlivněná doba probuzení.

Stop mód 1 se vyznačuje vypnutým napěťovým regulátorem. Jde o střední cestu mezi módem 1 a 2.

Stop 2 Tento mód je ze všech stop módů nejúspornější. Část periferií napájených hlavním zdrojem je úplně vypnuta, pouze SRAM1, SRAM2, CPU a některé periferie uchovávají svůj obsah.

Standby mód V tomto režimu je interní zdroj zcela vypnutý. Navzdory tomu může být zachován obsah paměti SRAM nastavením RSS bitu v PWR control registru. To zapříčiní napájení této paměti z nízkovýkonového zdroje.

1.3.2 MH-Z16

Tento senzor využívá NDIR (non-dispersive infrared) technologii, která se zakládá na silné absorpci infračerveného světla oxidem uhličitým (CO_2) ke zjištění koncentrace CO_2 ve vzduchu. Čidlo je vybaveno vestavěnou teplotní kompenzací, digitálním a PWM výstupem a UART rozhraním o rychlosti 9600Bd. Nabízí přesnost ±(100ppm + 6% z odečtené hodnoty).

Měřící rozsahy jsou 2000 a 5000 ppm, což je vzhledem k tabulce 1.3 pro běžné použití plně dostačující. UART rozhraní implementuje následující příkazy: čtení koncentrace CO_2, kalibrace nulového bodu, nastavení kalibrační hodnoty od nuly, vypnutí a zapnutí samokalibrace a nastavení měřícího rozsahu [19].

Koncentrace [ppm]	**Prostředí**
∼40000	Množství v lidském výdechu (20l CO2/h)
=5000	Limit koncentrace CO2 na pracovišti
>1000	Únava a snížená koncentrace
=1000	Doporučená hladina CO2 ve vnitřních prostorách
=400	Čerstvý přírodní vzduch

Tab. 1.3: Tabulka běžných koncentrací CO_2 v různých prostředích [20]

1.3.3 DS18B20

DS18B20 je digitální teploměr nabízející rozlišení od 9 do 12 bitů. Disponuje taktéž alarmem s programovatelnými spouštěmi. Teploměr komunikuje po One-Wire sběrnici, která pro komunikaci vyžaduje pouze jeden vodič a zemní vodič za použití 4,7kΩ pull-up rezistoru mezi napájecí a datový vodič. Tento rezistor má, podobně jako u I^2C, uplatnění v klidovém stavu, kdy udržuje napětí na sběrnici na úrovni logické 1. Čidlo dokonce umožňuje odebírat napájecí napětí přímo z datového vodiče (parazitní napájení). Zařízení má jedinečný 64 bitový identifikátor, díky kterému se může na sběrnici nacházet více DS18B20.

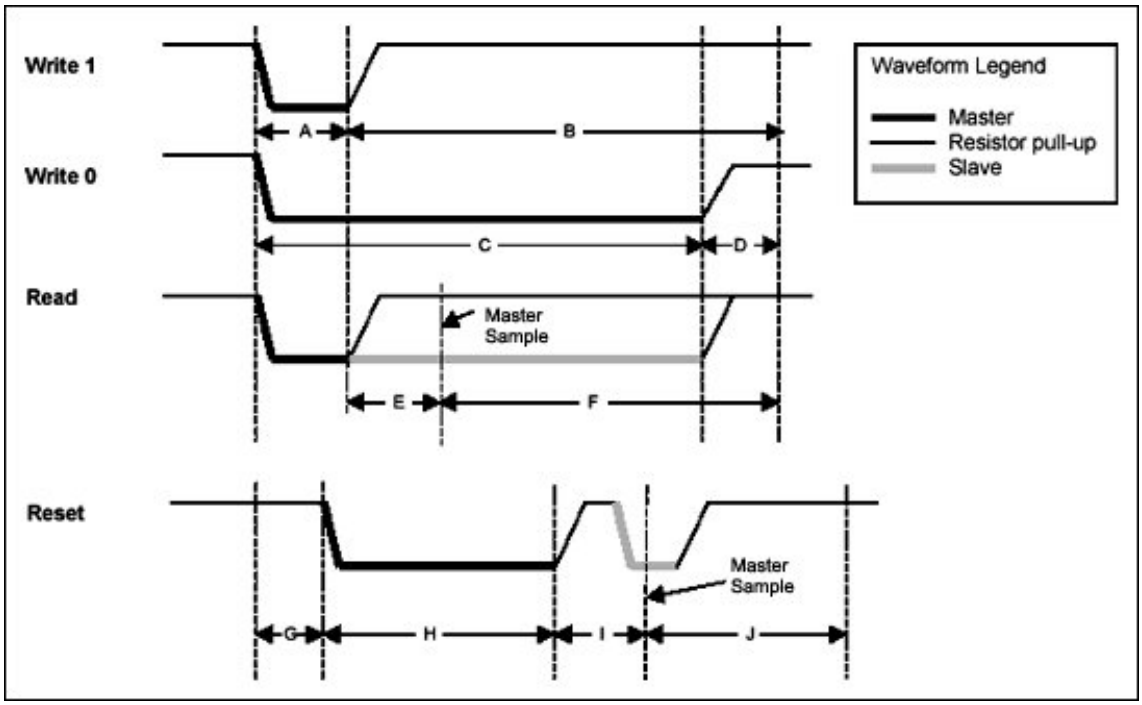
Zakladní operace na One-Wire jsou:

- Reset, který resetuje podřízené uzly a připraví je na příjem příkazu.
- Zápis 1, odesílá bit reprezentující 1 podřízeným uzlům.
- Zápis 0, odesílá bit reprezentující 0 podřízeným uzlům.
- Čtení, čte bit přicházející od podřízeného uzlu.

Parametr	**Doporučená doba [µs]**
A	6
B	64
C	60
D	10
E	9
F	55
G	0
H	480
I	70
J	410

Tab. 1.4: Časové intervaly One-Wire komunikace při standardní rychlosti [21]

Průběh komunikace je zřejmý z obrázku 1.5, který znázorňuje časové průběhy



Obr. 1.5: Časové průběhy One-Wire komunikace [21]

elementárních operací a doplňující tabulky 1.4 s délkami časových intervalů. Jednotlivé průběhy se nazývají Time Sloty (např. Read Time Slot).

Pomocí těchto operací můžeme provádět příkazy již specifické pro DS18B20. Například vyčtení hodnoty z teploměru probíhá následovně.

Ať se jedná o jakoukoliv komunikaci, master vždy na začátku resetuje sběrnici. Poté buď probíhá adresace zařízení nebo oznámení všesměrového vysílání, následně je inicializována konverze teploty, po definovaném čase je konverze dokončena a teplota je uložena do paměti. Master nyní může inicializovat novou komunikaci, která bude probíhat stejně s tím rozdílem, že vyšle příkaz na čtení teploty z paměti a nakonec přečte přijaté bajty nesoucí hodnotu teploty.

1.3.4 Lokální úložiště dat a reálný čas

Pro lokální zaznamenávání dat s časovou značkou byl zvolen Adafruit Data Logger Shield pro Arduino UNO, i díky kompatibilitě vývojového kitu s touto platformou. Na shieldu jsou osazeny čipy 74HC125 a DS1307. 74HC125 slouží jako buffer pro SPI, přes které probíhá čtení a zápis na kartu se souborovým systémem FAT16 nebo FAT32. Čip DS1307 obsluhuje funkcionalitu reálného času. Je vybaven rozhraním I^2C, přes které je možné nastavit či číst strukturu s aktuálním datem a časem. Shield je osazen paticí pro baterii CR1220, která udržuje hodiny v chodu při výpadku napájení. Nechybí ani prototypovací část, kam lze pájet další součástky [22].

1.3.5 Cirkulace vzduchu a odvětrávání

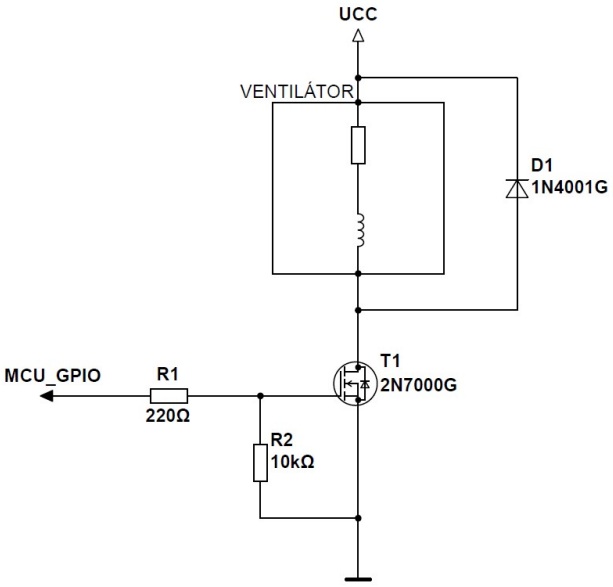
Pro nejlepší možnou simulaci ideálních podmínek je nutné, aby vzduch uvnitř komory cirkuloval jako v přirozeném prostředí. Toto zajišťuje malý větrák, který běží na velmi nízkých otáčkách. Ventilátor však nemůže poskytovat pouze tuto, nízkou, rychlost, protože před měřením je nutné komoru řádně vyvětrat. Proto budou otáčky ventilátoru řízeny PWM modulací, která je založena na periodickém spínaní napětí s danou střídou.

$$s = \frac{t_{zap}}{T} \tag{1.11}$$

kde t_{zap} je doba, po které je napětí sepnuto a T je perioda spínaní. Jelikož se jedná o poměrnou dobu, hodnota střídy s může nabývat hodnot od 0 do 1.

Poté můžeme vyjádřit střední napětí na zátěži U_z v závisloti na střídě jako součin přivedeného napájecího napětí U_{CC} a střídy s.

$$U_z = U_{CC} \cdot s \tag{1.12}$$



Obr. 1.6: Schéma zapojení ventilátoru s PWM regulací otáček (autor)

Spínací prvek nejčastěji tvoří unipolární MOSFET tranzistor s indukovaným kanálem, který zaručuje rozepnutý stav při nulovém napětí na hradle. Typ vodivosti volím N, především kvůli spojení zemí, což je standardní praxe, namísto spojování napájení. Další drobností, která nahrává tranzistoru typu N, jsou fyzikální zákony, dle kterých jsou elektrony lepšími nosiči náboje než díry, které se uplatňují u vodivosti typu P.

Předpokládané schéma zapojení je na obrázku 1.6. Za zmínku stojí rezistor R1 mezi hradlem a GPIO, ten chrání GPIO před příliš velkým proudovým odběrem při zapínaní tranzistoru, způsobeným nabíjením vstupní kapacity. Rezistor R2 zase zajišťuje, že se tranzistor nebude spínat ve chvíli, kdy GPIO není nastaven jako výstupní, například při inicializaci mikrokontroléru. Ventilátor představuje sériová RL zátěž připojená paralelně k diodě. Při běhu ventilátoru touto diodou neteče žádný proud. Ovšem po jeho vypnutí indukčnost stále dodává do obvodu proud ve stejném směru. Tento proces se nazývá demagnetizace indukčnosti. Dioda tedy zajišťuje bezpečnou demagnetizaci cívky. Bez jejího použití by hrozilo nevratné zničení tranzistoru proudovým průrazem.

Komora musí být z principu funkce uzavřená, je tedy třeba vyřešit mechanismus otevírání víka komory. Ten bude realizován pomocí servomotoru s odpovídajícím ramenem. Protože v této aplikaci nejsou nároky na rozměry, napájení celého systému bude zprostředkováno 12V akumulátorem gel-olovo o dostatečné kapacitě.

2 Výsledky studentské práce

2.1 Zdrojový kód

Tato část je zaměřena na představení a popis zdrojového kódu pro mikrokontrolér v jazyce C. Pro vývoj a ladění bylo použito vývojové prosředí přímo od výrobce vývojového kitu, resp. čipu STMicroelectronics - STM32CubeIDE, které je postavené na open-source řešení Eclipse, jenž je známé především jako vývojové prosředí pro jazyk Java. Jelikož má kit vestavěný debugger a podporu IntelliSense, je vývoj softwaru velmi efektivní v porovnání např. s platformou Arduino, jejíž IDE nabízí pokročilý debugging pouze v profesionální verzi. Díky debuggeru je možné nastavit tzv. breakpointy na konkrétních řádcích, na nichž se následně program zastaví a vývojář může sledovat aktuální hodnoty proměnných a případně krokově postupovat dále v programu.

Před použitím STM32CubeIDE je ještě vhodné použít generátor zdrojového kódu STM32CubeMX, který je v současné době již integrován přímo v IDE, takže se vlastně jedná o první krok při použití IDE. Ten po zvolení konkrétního čipu či rovnou vývojové desky umožňuje aktivovat a konfigurovat použitá rozhraní, nastavit funkci jednotlivých GPIO, registrovat použité middleware a v neposlední řadě nastavení zdroje a frekvence hodin. Po dokončení konfigurace je vygenerován inicializační kód. Specifikum tohoto řešení je, že vývojář svůj vlastní kód píše do bloků mezi komentáři /* USER CODE BEGIN */ a /* USER CODE END */. Pokud je vlastní kód jinde, dojde k jeho smazání při přístím generování kódu [23].

Každý program v jazyce C začíná spuštěním funkce *main* s návratovým typem int nebo void. V této funkci probíhá hlavní logika aplikace. Pro přenositelnost kódu a lepší přehlednost jsem pro jednotlivá čidla vytvořil samostatné zdrojové soubory s příslušnými hlavičkovými soubory, které jsou do hlavního zdrojového souboru main.c importovány direktivou #include.

2.1.1 Měření koncentrace CO2 po sběrnici UART

Z nabízených možností odečítání naměřené koncentrace je pro tuto aplikaci nejvhodnější UART rozhraní díky snadné implementaci a také nejlepším konfiguračním možnostem. Implementována byla funkce jak pro vyčtení hodnoty koncentrace oxidu uhličitého, tak výpočet kontrolního součtu ověřující validitu přijatých dat.

Výpis 2.1: Funkce čtení koncentrace CO_2

```
1
2  uint16_t MHZ16_Read(void)
3  {
4    uint8_t TxMessage[9] =
5      {0xFF, 0x01, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79};
6    uint8_t RxBuffer[9];
7
8    if(HAL_UART_Transmit
9      (&huart1, TxMessage, sizeof(TxMessage), 1000) != HAL_OK)
10     Error_Handler();
11
12   if(HAL_UART_Receive
13     (&huart1, RxBuffer, sizeof(RxBuffer), 1000) != HAL_OK)
14     Error_Handler();
15
16   if(getCheckSum(RxBuffer) == RxBuffer[sizeof(RxBuffer) - 1])
17   {
18     uint16_t ret = RxBuffer[2] * 256 + RxBuffer[3];
19     ret++;
20     return ret;
21   }
22   else return 0;
23 }
```

Čtení koncentrace CO_2 dle 2.1 je díky knihovně HAL (Hardware Abstract Layer) jednoduché. Stačí deklarovat dvě pole bajtů, jedno s příkazem ke čtení dle dokumentace [19] a druhé prázdné pro příjem dat. Následně jsou v podmínkách úspěšného provední volány funkce na vysílání a příjem signálu. Pokud úspěšně proběhnou, je přijatá zpráva předána funkci kontrolního součtu, jehož výpočet je taktéž k nalezení v dokumentaci. Jestliže tento součet odpovídá poslednímu přijatému bajtu reprezentujícímu právě kontrolní součet, je přistoupeno k převodu datových bajtů na číselnou hodnotu, která je funkcí navrácena.

2.1.2 Měření teploty po sběrnici One-Wire

STM32 nemá integrovanou podporu One-Wire, proto bylo nutné napsat komunikační protokol pro tuto sběrnici. Jak vyplývá z principu popsanému v úvodu, pro komunikaci je nutné generovat pulzy o hodnotě až 1µs. Knihovna HAL ovšem obsahuje vestavěnou funkci pro generování přerušení nejméně 1ms. Nejmenší dosažitelné

přerušení odpovídá periodě použitého časovače. Časovač TIM2 má v aktuální konfiguraci frekvenci 32MHz, pro periodu 1µs je třeba frekvence alespoň 1MHz. Časovači tedy nastavíme hodnotu děliče na 32 (32MHz/32 = 1MHz) a rozsah 16 bitového čítače na hodnotu 2^{16}, tedy 0xFFFF, pro možnost dosažení co největšího intervalu. Tato konfigurace zajistí inkrementaci čítače o 1 každých 32 period čítače, což je v tomto případě 1µs. Z toho vyplývá nejvyšší možný časový interval 0xFFFF = 65 536µs.

Výpis 2.2: Funkce pro mikrosekundové přerušení programu

```
1 void delayus(uint16_t delayus)
2 {
3     __HAL_TIM_SET_COUNTER(&htim2, 0);
4     while(__HAL_TIM_GET_COUNTER(&htim2) < delayus);
5 }
```

Následná implementace funkce pro generování mikrosekundového přerušení je již triviální. Vynuluje se čítač a následně běží cyklus dokud hodnota čítače nedosáhne požadované hodnoty.

Posledním krokem je vytvoření funkcí pro přepínání GPIO mezi vstupem a výstupem.

Výpis 2.3: Funkce přepínající mód GPIO

```
1 void Set_Pin_Output (GPIO_TypeDef *GPIOx,
2 uint16_t GPIO_Pin)
3 {
4     GPIO_InitTypeDef GPIO_InitStruct = {0};
5     GPIO_InitStruct.Pin = GPIO_Pin;
6     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
7     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
8     HAL_GPIO_Init(GPIOx, &GPIO_InitStruct);
9 }
```

Výpis 2.3 znázorňuje funkci pro nastavení GPIO jako výstupu. Obecně pro konfiguraci jednotlivých GPIO existuje inicializační struktura, která obsahuje všechny důležité hodnoty. V tomto případě nastavím upravovaný pin, nastavím požadovaný mód, tedy výstup push-pull, což je standardní mód Druhou variantou je open-drain. Rychlostí se rozumí rychlost naběžné a sestupné hrany. Je dobrým zvykem tuto rychlost udržovat co nejnižší a zvyšovat ji pouze v odůvodněných případech. V některých případech je také nutné nastavit integrovaný pull-up rezistor, to je sice třeba i zde, ovšem pullup rezistory v STM32 mají hodnotu přibližně 40kΩ, což je pro tuto aplikaci příliš. Katalogová hodnota je přibližně 5kΩ. Obdobně je realizována funkce pro nastavení vstupu.

Nyní již nic nebrání vytvoření samotného komunikačního protokolu. Implementoval jsem všechny základní operace dle obr. 1.5 a tab. 1.4.

Výpis 2.4: Inicializace sběrnice One-Wire

```
1  uint8_t DS18B20_Init (void)
2  {
3      uint8_t Response = 0;
4      Set_Pin_Output(DS18B20_PORT, TEMP_Pin);
5      HAL_GPIO_WritePin (DS18B20_PORT, TEMP_Pin, 0);
6      delayus (480);
7
8      Set_Pin_Input(DS18B20_PORT, TEMP_Pin);
9      delayus (60);
10
11     if (!HAL_GPIO_ReadPin (DS18B20_PORT, TEMP_Pin))
12         Response = 1;
13     else Response = 0;
14
15     delayus (420);
16
17     return Response;
18 }
```

V obr. 1.5 je důležité respektovat legendu čar, dle které je určeno, kdo daný stav inicializuje. Pokud jde o mastera, pak se nastaví GPIO jako výstup a po danou dobu udržuje na sběrnici požadovanou logickou úroveň. V případě že za daný interval zodpovídá pull-up rezistor, MCU uvolní sběrnici přepnutím do vstupního módu. Jakmile přijde na interval inicializovaný slave zařízením, probíhá čtení dat ze sběrnice.

V předchozím výpisu 2.4 se nachází zdrojový kód pro incializaci sběrnice, tato funkce se volá vždy při zahájení komunikace, je tedy pro funkci stěžejní. Po uvedení sběrnice na úroveň log. 0 po dobu 480µs a uvolnění, můžeme po dobu 60 až 240µs číst puls, který indikuje přítomnost zařízení. Nakonec program vyčká zbývající čas do 480µs pro dokončení inicializačního cyklu.

Výpis 2.5: Funkce pro čtení ze sběrnice One-Wire

```
1  uint8_t DS18B20_Read (void)
2  {
3      uint8_t value=0;
4
5      for(int i = 0; i < 8; i++)
6      {
7          Set_Pin_Output(DS18B20_PORT, TEMP_Pin);
8
9          HAL_GPIO_WritePin(DS18B20_PORT, TEMP_Pin, 0);
10         delayus(2);
11
12         Set_Pin_Input(DS18B20_PORT, TEMP_Pin);
13         if(HAL_GPIO_ReadPin (DS18B20_PORT, TEMP_Pin))
14         {
15             value |= 1<<i;
16         }
17         delayus(58);
18     }
19     return value;
20 }
```

Na 2.5 se nachází kód pro přečtení bajtu z DS18B20. To probíhá tak, že MCU, v cyklu s osmi opakováními pro každý bit, zahájí slot pro čtení jednoho bitu uvedením sběrnice na úroveň log. 0 po dobu alespoň 1µs. Po uplynutí této doby přečte přijatý bit na vstupu GPIO v podmínce na ř.15 v 2.5. Pokud je podmínka vyhodnocena kladně, je proveden bitový součet proměnné pro ukládání hodnoty a jedničky posunuté o počet bitů daný číslem iterace cyklu. V opačném případě zbývá jediná možnost 0, kterou je proměnná inicializována, takže není třeba ji znova zapisovat.

2.1.3 Měření

Nyní již můžeme využít zvolené senzory pro získání dat k analýze. Pro měření jsem navrhnul následující metodu.

Na začátku měřícího cyklu jsou vyčteny hodnoty z čidel a to v cyklu, jehož počet opakování závisí na konfigurační konstantě, jež je definovaná v programu. Poté se z naměřených hodnot vypočítá aritmetický průměr. Program vyčká definovanou dobu délky měření a opět opakuje předchozí krok. Obě měření probíhají několikanásobně z toho důvodu, že právě pouze z těchto dvou hodnot je pomocí lineární interpolace sestavena funkce růstu koncentrace CO_2. Je vhodné zmínit, že samotná funkce se v

programu nevyskytuje, je v něm použita její derivace, která je potřebná pro výpočet toku plynu prostředím dle 1.10.

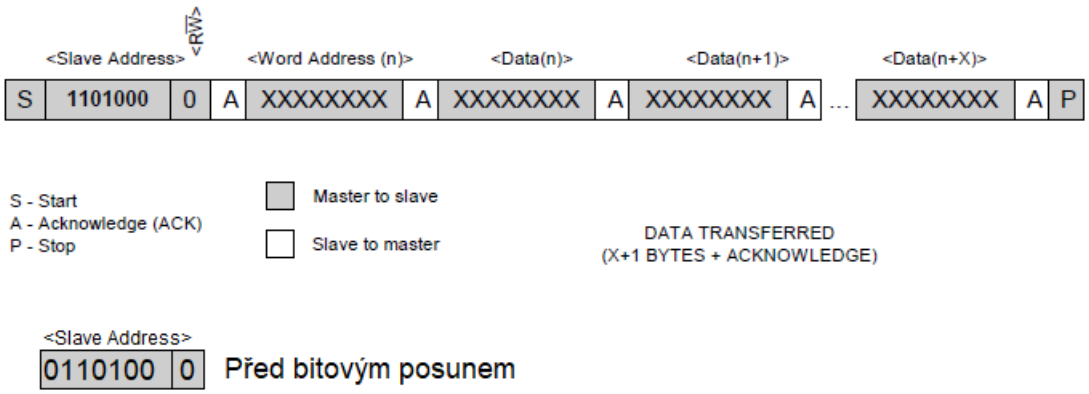
Výpis 2.6: Ukázka výpočtu toku plynu v jazyce C

```
1 float avgTemp = (saveData.avgTempStart
2     + saveData.avgTempEnd) / 2;
3 float deltaCdt = (saveData.avgPpmEnd
4     - saveData.avgPpmStart) / (DELAY_MIN * 60);
5 saveData.flux = ((10140 * CHAMBER_VOLUME) / (GAS_CONSTANT
6     * (273.15 + (avgTemp)) * COVERED_AREA)) * deltaCdt;
```

Na výpisu 2.6 je znázorněn výpočet toku plynu prostředím. Před samotným výpočtem se ještě musí vypočítat některé hodnoty. První je průměrná teplota v průběhu měření a druhá již zmiňovaná derivace koncentrace podle času. Po vypočtení těchto hodnot nic nebrání výpočtu samotného toku CO_2 komorou, který je na posledním řádku výpisu. Všimněme si proměnných psaných hůlkovým písmem. Ty jsou podobně jako počet opakování konfigurační konstanty reprezentující objem komory (CHAMBER_VOLUME) v m^3, jí zabíranou plochu (COVERED_AREA) v m^2 a jednu pravou konstantu - molární plynovou konstantu (GAS_CONSTANT) v J mol^{-1} K^{-1}.

2.1.4 Lokální uložení na SD kartu

Po skončení měřícího cyklu data ukládám na SD kartu. Pro lepší interpretaci výsledků používám v modulu vestavěné hodiny. Proto před zápisem dat ještě načtu aktuální čas. Pomocí funkce getTime() 2.7, v té se deklaruje pole bajtů a provolá se čtení I^2C, které má parametry handler I^2C sběrnice, adresa zařízení, počáteční adresa čtení, velikost jedné adresy, ukazatel na pole pro výsledek, počet čtených adres a timeout. U adresy stojí za zmínku, že v I^2C, je adresa 7 bitová a přenáší se na prvních 7 bitech prvního bajtu, viz 2.1. Proto je třeba provést bitový posun doleva tak, aby adresní bity byly na příslušné pozici. Po úspěšném čtení zbývá přijatá data dekódovat. Je použito BCD kódování, funkce pro dekódování se nachází taktéž ve výpisu 2.7.



Obr. 2.1: Struktura zapisovacího rámce s ukázkou bitového posunu [24], upraveno autorem

Výpis 2.7: Funkce pro čtení času z DS1307

```
1  #define DS1307_ADDRESS 0x68<<1
2  uint8_t bcdToDec(uint8_t val)
3  {
4    return (int)((val/16*10) + (val%16));
5  }
6  void getTime(void)
7  {
8    uint8_t get_time[7];
9    HAL_I2C_Mem_Read(&hi2c3, DS1307_ADDRESS, 0x00, 1,
10   get_time, 7, 1000);
11   s_time.seconds = bcdToDec(get_time[0]);
12   s_time.minutes = bcdToDec(get_time[1]);
13   s_time.hours = bcdToDec(get_time[2]);
14   s_time.dayofweak = bcdToDec(get_time[3]);
15   s_time.dayofmonth = bcdToDec(get_time[4]);
16   s_time.month = bcdToDec(get_time[5]);
17   s_time.year = bcdToDec(get_time[6]);
18 }
```

Výpis 2.8: Zápis dat na SD kartu

```
1  FIL fil;
2  FRESULT fresult;
3  fresult = f_open(&fil, "app_log.csv", FA_OPEN_APPEND
4              | FA_WRITE | FA_READ);
5  f_printf(&fil, "20%d-%02d-%02d␣%02d:%02d:%02d", s_time.year,
6  s_time.month, s_time.dayofmonth, s_time.hours,
7  s_time.minutes, s_time.seconds);
8  fresult = f_write(&fil, ";", 1, &bw);
9  f_printf(&fil, "%d", saveData.avgPpmStart);
10 fresult = f_write(&fil, ";", 1, &bw);
11 strcpy(buffer, temperatureStartString);
12 fresult = f_write(&fil, buffer, bufsize(buffer), &bw);
13 bufclear(buffer, sizeof(buffer)/sizeof(buffer[0]));
14 fresult = f_close(&fil);
```

Pro integraci SD karty STM nabízí middleware vrstvu FATFS, která podporuje různé souborové systémy rodiny FAT [25]. Pro implementaci je nutné napsat funkce pro fyzickou komunikaci. Ty se nachází v souboru fatfs_sd.c. Nakonec je nutné do souboru rozhraní FATFS user_diskio.c namapovat příslušné funkce pro základní operace jako read, write, init nebo status.

Na výpisu 2.8 se nachází útržek kódu zapisující data na SD kartu. Nejdříve je soubor otevřen v režimech připisování, nakonec čtení a zápis. Zvolil jsem soubor typu .csv, kvůli jednoduchému případnému SW zpracování. Do prvního sloupce je zapisován čas ve formátu YYYY-MM-DD HH:mm:ss, zápis celočíselné hodnoty je proveden funkcí f_prinf(), která má podobné chování jako známá funkce printf(), jen je nutné jako první parametr uvést ukazatele na soubor. Bohužel není možné stejným způsobem zapisovat desetinné hodnoty. Z toho důvodu se před zápisem teplota převádí na znakový řetězec, který se již zapisuje do souboru. Po dokončení zápisu se soubor zavírá.

2.1.5 Odeslání dat přes LoRaWAN

Nyní přichází chvíle získaná data odeslat na servery operátora. Pro toto řešení jsem zvolil třídu LoRaWAN A, především kvůli zamýšlenému chodu na baterii. Toto rozhodnutí podporuje fakt, že tato aplikace nevyžaduje přijímat downlink zprávy, jejichž zvýšená frekvence možnosti příjmu je přidaná hodnota zbylých rozšiřujících tříd. Frekvenční pásmo jsem zvolil ISM 868MHz z důvodu lepšího pokrytí na našem území než alternativní pásmo 433MHz. Pro ladící účely používám aktivační typ ABP, kde oproti OTAA není nutný handshaking se serverem. Jelikož má vývojový

kit relativně slabou anténu, tak je sice schopen odeslat uplink zprávu na dlouhou vzdálenost, ale již nedokáže přijmout potvrzovací downlink zprávu. Tím pádem se jeho užitím extrémně zvýší dosah. Navzdory tomu, že samotná specifikace nijak neomezuje počet zpráv za den, tak se na tuto aplikaci vztahují hned 2 regulace. První z nich je od ETSI, která říká, že maximální doba trvání uplink zpráv na zařízení je 1%. Druhá je od poskytovatele služeb The Things Industries, podle níž je maximální vysílací doba pro zařízení na den je 30 sekund, to je v porovnání s ETSI razantní zpřísnění. Pokud uvážíme, že den má 86 400 sekund, tak vychází, že maximální vysílací doba v procentech je 0,0347%.

Za předpokladu rovnoměrného odesílání dat v průběhu celého dne můžeme dle zavedených vztahů vypočítat čas zprávy na cestě a z něj pak minimální interval odesílání zpráv. Jak je zmíněno v popisu rovnice 1.6, za dobu trvání symbolu dosadím převrácenou hodnotu symbol rate. Spreading Factor = 12 a Coding rate pro větší robustnost systému, délka payloadu je 13B, protože odesílám dvě hodnoty datového typu int16_t (teplota), dvě hodnoty uint16_t (koncentrace), které v paměti zabírají po 2B. Vypočtený tok plynu nese proměnná int32_t zabírající 4B. Pro tuto hodnotu musela být zvolena větší proměnná kvůli skutečnosti, že pro zachování přesnosti je výsledek typu float násoben milionem. CRC je dle výchozího nastavení povoleno stejně jako hlavička a Low Data Rate Optimize, který je automaticky aktivní při šířce pásma 125kHz a SF větší než 11 včetně.

Po dosazení do vztahu 1.6 vychází čas vysílání preambule $t_{preambule}$ 401,4ms, doba trvání vysíláni payloadu je dle 1.7 1,049s. Nyní můžeme dosadit do 1.5 a získáme celkový čas trvání zprávy (ToA), což je 1,4504s.

Duty cycle dle 1.4, kde je za periodu dosazen počet sekund dne a za vysílací čas 30 sekund, což je povolená denní aktivita, vychází 0,0003472.

Nyní již můžeme dle 1.8 stanovit interval vysílání zpráv, při kterém nebudou porušovány zásady TTS. Výsledná hodnota je 4176 sekund, tedy 20 zpráv za den. Je vhodné poznamenat, že tato konfigurace je časově nejnáročnější, zvýšit ji může již jen větší délka payloadu [26].

Co se týče zdrojového kódu, podobně jako u FATFS i zde se nachází middleware LoRaWAN. Základní strukturu zdrojového kódu zprostředkoval STM32CubeMX. Programátorovi potom zbývá implementovat Rx, Tx a join funkce. Diskutovaná část se nachází v souboru lora_app.c. Pro tuto aplikaci jsou stěžejní následující.

Jediná veřejná funkce tohoto zdrojového souboru je LoRaWANInit(). Ta zodpovídá, jak název napovídá, za inicializaci a nastavení zvolených komunikačních parametrů. V neposlední řadě zde také probíhá registrace funkce na odeslání dat SendTxData() jako proces sekvenceru. Sekvencer je softwarový nástroj, který poskytuje plánování spouštění funkcí na pozadí. Jedná se o bezpečnou metodu implementace Low Power módu v případě, kdy sekvencer nemá žádné čekající procesy k

vykonání. Základní funkce použité v tomto programu jsou UTIL_SEQ_RegTask(), která registruje proces. Dále se jedná o UTIL_SEQ_Run() vyvolávající všechny registrované funkce dle priority a poslední je UTIL_SEQ_SetTask(taskId), která vyvolá funkci s požadovaným ID [27]. V mezičase mezi voláními těchto funkcí je MCU v nejúspornějším módu - Stop 2.

Výpis 2.9: Uchovávané datové struktury

```
1  struct LogData
2  {
3    uint16_t avgPpmStart;
4    uint16_t avgPpmEnd;
5    float avgTempStart;
6    float avgTempEnd;
7    float flux;
8  };
9
10 struct LoRaWANData
11 {
12   uint16_t avgPpmStart;
13   uint16_t avgPpmEnd;
14   int16_t avgTempStart;
15   int16_t avgTempEnd;
16   int16_t flux;
17 };
```

Stěžejní funkcí programu je již zmíněná SendTxData(). Probíhá zde kompletní měřící proces. Data jsou ukládána do dvou struktur viz výpis 2.9. LogData slouží k zápisu na SD kartu a LoraWAN data obsahují pouze 16 bitové celočíselné proměnné. Kvůli jednoduchosti přenosu jsou desetinné hodnoty násobeny stem (milionem u toku plynu) pro zachování přesnosti na dvě desetinná místa. Na výpisu 2.10 je dále převod dvoubajtových proměnných na bajty, toho je docíleno bitovým součinem hodnoty s 0xFF, což zaručí, že všechny bity kromě 8 LSB budou nulové. V kombinaci s bitovým posunem o 8 můžeme stejnou proceduru opakovat i pro zbylých 8 MSB bitů, které se touto operací posunou právě na LSB pozici. Po zapsání dat do bufferu následuje odeslání, pokud bude úspěšné, je reportována zpráva na konzoli. Pokud bude odeslání neúspěšné, je vypsána informace o dalším vysílání, pokud je tedy v plánu.

Výpis 2.10: Převod datového typu float na byte a uložení do pole

```
struct LogData saveData;
struct LoRaWANData sendData;
int i = 0;
sendData.avgTempEnd = saveData.avgTempEnd * 100;
AppData.Buffer[i++] = (uint8_t)(sendData.avgTempEnd >> 8
& 0xFF);
AppData.Buffer[i++] = (uint8_t)(sendData.avgTempEnd & 0xFF);
sendData.avgPpmStart = saveData.avgPpmStart;
AppData.Buffer[i++] = (uint8_t)(sendData.avgPpmStart >> 8
& 0xFF);
AppData.Buffer[i++] = (uint8_t)(sendData.avgPpmStart & 0xFF);
AppData.BufferSize = i;

if (LORAMAC_HANDLER_SUCCESS == LmHandlerSend(&AppData,
LORAWAN_DEFAULT_CONFIRMED_MSG_STATE, &nextTxIn, false))
  {
    printf("SEND REQUEST\r\n");
  }
  else if (nextTxIn > 0)
  {
    printf("Next Tx in: ~%lu second(s)\r\n",
    (nextTxIn / 1000));
  }
```

2.2 Zpracování dat na síťovém serveru

Data jsou odesílána na síťové servery The Things Stack poskytovatele The Things Industries, kde dochází k jejich přeposílání na aplikační servery.

Něž k tomu dojde, musí být data doručena z koncového zažízení. Architektura sítě TTS rozděluje jednotlivé úlohy různým modulům, především kvůli decentralizaci sítě. V dalších řádcích nebudou uvažovány downlink zprávy, které nejsou předmětem této práce.

Prvním prostředníkem na cestě k serveru je již zmiňovaná gateway, která především přijatá data modulovaná LoRou zabalí do IP paketu. Zbylé chování je závislé na použitém gateway protokolu. Velká část bran ovšem používá stejný referenční protokol definovaný LoRa Alliance, ten ukládá povinnost k přijatým binárním datům přidat informace o hodnotách RSSI a SNR. Nezávisle na předchozím, gateway periodicky odesílá data o sobě samé jako například GPS souřadnice, počet přijatých a odeslaných paketů a ostatní metriky.

Vraťme se ještě k GPS souřadnicím, ty mohou být v některých aplikacích relevantní či přímo vyžadované, třeba pro určení polohy triangulací v případě, že je koncové zařízení v dosahu více bran. Z tohoto důvodu backend serveru uchovává poslední zprávu od každé brány a přidává informaci o GPS souřadnicích ke každé uplink zprávě.

Po přípravě paketu k odeslání přichází na směrování, které je rozděleno na několik kroků. Na straně gateway probíhá tzv. hrubé směrování. Spočívá v tom, že se ze zprávy vyextrahuje 32 bitová adresa, z níž 25 bitů přísluší operátorovi, ale není jedinečná. Každý broker (zprostředkovatel) oznámí počet adres, jež obsluhuje a začínají tímto prefixem. Princip je podobný jako u protokolu BGP. Zařízení v síti si mohou v pravidelném intervalu vyžádovat seznam brokerů v síti a jimi obsluhovaných prefixů.

První komponentou na straně serverů TTS je broker zmíněný v předchozím odstavci. Protože z principu fungování LoRaWAN sítě jsou zprávy serverem přijatý n-krát, kde n je počet bran v dosahu, musí být ošeřeny duplicitní zprávy. Mohlo by se zdát, že broker vybere jednu z nich a zbytek zahodí, to je ovšem pouze částečná pravda. Z každé zprávy jsou extrahovány specifické informace brány, které jsou přidány k výsledné zprávě přeposlané na server. Protože zprávy nepřijdou ve stejnou dobu je nutné je po jistou dobu udržovat v paměti. Tato doba je u TTS nastavena na 200ms.

Následuje finální směrování, kdy se rozhodne, které aplikaci a zařízení zpráva patří. To probíhá výpočtem Message Authentication Code (MAC) pro každého potencionálního příjemce. Aby toto mohlo proběhnout, broker si vyžádá seznam zařízení, která používají stejnou adresu ze síťového serveru a ověří zda MAC může být validována za užití relačního klíče. Pokud broker nenajde shodu, zpráva je zahozena.

V komponentě nazvané síťový server probíhá aktualizace stavu zařízení, přidání downlink šablony, kterou může využít handler pro komunikaci směřem k zařízení. Navíc v této chvíli je možnost přidat zprávě MAC příkazy. Například příkazy pro vysílání vyšší přenosovou rychlostí po analýze síly přijatého signálu.

Nyní je vše připraveno pro konečné formátování uplink zpráv, což zprostředkovává komponenta handler. V nabídce je několik možností, z nichž některé jsou předpřipravené šablony. Já jsem se rozhodl využít nejuniverzálnější z nich - formátování pomocí JavaScriptu.

Výpis 2.11: Převod bajtů na datový typ uint16 v jazyce JavaScript

```
function bytesToInt16(bytes, isSigned) {
  var bits = (bytes[0] << 8) | bytes[1];
  var sign = 1;
  if (isSigned && bits >>> 31 === 1) {
    sign = -1;
    bits = bits & 0x7FFFFFFF;
  }
  return bits * sign;
}
```

Pro úspěšné formátování je nutné implementovat funkci decodeUplink, která je implicitně volána při každém přijetí uplink zprávy [29]. Na výpisu 2.11 je ukázka funkce, která převádí bajty na uint16, funkce je ale univerzální a funguje i pro typ int16. Podobně jako při kódování i zde u dekódování je použito bitových posuvů. Vhodné dodat, že kódované desetinné hodnoty - v tomto případě teploty, je nutné aby volající funkce výsledek vydělila stem na konečnou hodnotu.

Na obrázku 2.2 jsou vidět přijatá data odesílaná v 10 sekundových intervalech.

Obr. 2.2: Snímek obrazovky s přijatými daty, pořízeno na webu The Things Stack

Ačkoliv se vysílač nacházel nedaleko, řádově vyšší desítky metrů od brány, tak se některé z paketů ztratily, což lze vyčíst z nepravidelného příjmu zpráv. Ve sloupci Data preview je možné vidět část přijatých dat v hexadecimální bajtové reprezentaci. Nachází se zde také dekódovaná data v poli measurements, která bohužel nejsou vidět z prostorových důvodů.

2.3 Vizualizace dat

Posledním krokem na straně TTS je nastavit integraci s konkrétní službou. Opět je v nabídce řada možností jako AWS IoT, Azure IoT Hub, MQQT API nebo vlastní WebHook, což je HTTP push API, jejíž funkce spočívá v okamžitém přeposílání dat na specifický koncový bod, tím je zajištěno přeposílání dat do aplikace v podstatě v reálném čase [30]. Protože jsem se při pokusech o použití některého z komerčních řešení setkal s nějasnými podmínkami freemium verzí, či tato možnost vůbec nebyla a také proto, že např. u Azure IoT Hub byla relativně pracná konfigurace s ohledem na to, že cílem je vykreslit jen několik grafů z jednoho zařízení, rozhodl jsem se využít soukromého serveru, právě prostřednictvím vlastního WebHooku na The Things Stack.

Na serveru běží upravený open source dohledový systém Zabbix, který je původně určen pro monitorování sítí, serverů apod. Protože REST API přijímá pouze požadavky typu GET s identifikátorem hosta, dvojicí klíč - hodnota a časovou známku, je nutné před konečným zobrazením provést jeden mezikrok, a to parsovat JSON a syntetizovat z něj URL požadavek typu GET.

Koncový bod WebHooku je tedy nastaven na spuštění skriptu lorajson.php, který je k nalezení v příloze A. Skript přečte celý vstup do stringu a následně z něj dekóduje JSON, ze kterého extrahuje relevantní informace zmíněné v předchozím odstavci. Poté ze získaných dat, v cyklu pro každý pár klíč - hodnota, sestaví URL adresu pro volání REST API, která je volána prostřednictvím funkce openURL($url).

Ukázka vykreslených grafů je dostupná v příloze B.

Závěr

Podařilo se číst naměřené hodnoty koncentrace oxidu uhličitého a teploty, které jsou použity k výpočtu sledovaného toku CO_2 komorou. Komunikace se senzorem CO_2 byla bezproblémová díky knihovnám od STMicroelectonics. Čtení hodnoty teploty bylo podstatně složitější. Po neúspěšné zkoušce několika nalezených řešení jsem se rozhodl napsat vlastní komunikaci pro One-Wire. Za tímto účelem bylo také nutné naprogramovat generování mikrosekundových pulzů. Také se podařilo implementovat generaci PWM pulzů, která bude použita pro regulaci otáček motoru a ventilátoru. Všechny naměřené a vypočtené hodnoty jsou odeslány na The Things Stack. Toto bylo otestováno v krátkých časových intervalech vzhledem k lokalitám, kde jsem zaznamenal dostatečný signál.

Pro případ výpadku komunikace se serverem jsou data ukládána lokálně na SD kartu. Funkcionalita byla implementována za pomoci middleware FATFS, který je oficiálně propagován výrobcem mikroprocesoru. Data jsou ukládána ve formátu csv. Data jsou opatřena časovou značkou pro možnost analýzy. Čip reálného času, který je osazen na modulu s čtečkou SD karet, komunikuje po sběrnici I^2C. Tato sběrnice má integrovanou podporu v knihovně Hardware Abstract Layer, a tak bylo čtení času poměrně jednoduché.

V době kdy zařízení nevyvíjí žádnou činnost se přepně do nízkovýkonového režimu - Stop 2. To zajišťuje velmi nízkou spotřebu v čase mimo měření. Pro dlouhodobé testování výdrže nebyl prostor, ale pro představu je udávaná spotřeba v módu Stop 2 řádově stovky nA oproti jednotkám mA v běžném Run módu.

Úspěšně se podařilo zpracovat přijatá data na serverech TTS, což obnáší dekódování z bajtů na číselnou hodnotu a přeposlání dat k prezentaci. Vizualizace je řešena open source platformou Zabbix. Vykreslování grafů bude dobře sloužit analýze dat. Rozhraní nabízí širokou paletu různých grafů a agregačních funkcí.

Literatura

[1] MEKKI, Kais, Eddy BAJIC, Frederic CHAXEL a Fernand MEYER. A comparative study of LPWAN technologies for large-scale IoT deployment. In: *ICT Express* [online]. Seoul: Elsevier B.V., 2019, s. 1-7 [cit. 2021-11-29]. ISSN 2405-9595. Dostupné z: doi:doi.org/10.1016/j.icte.2017.12.005

[2] What is the Technology Behind LoRa Frequency. *MOKO SMART* [online]. mokosmart.com: Moko Smart, 2020 [cit. 2021-11-23]. Dostupné z: https://www.mokosmart.com/lora-frequency/

[3] Regional Parameters. *The Things Network* [online]. Amsterdam, Netherlands: The Things Industries, 2021 [cit. 2021-11-12]. Dostupné z: https://www.thethingsnetwork.org/docs/lorawan/regional-parameters/

[4] LoRaWAN Architecture. *The Things Network* [online]. Amsterdam, Netherlands: The Things Industries, 2021 [cit. 2021-11-27]. Dostupné z: https://www.thethingsnetwork.org/docs/lorawan/architecture/

[5] LIE, Robert. LoRa/LoRaWAN tutorials. *Youtube* [online]. Netherlands: Mobilefish.com, 2021, [cit. 2021-11-14]. Dostupné z: https://www.mobilefish.com/developer/lorawan/lorawan_quickguide_tutorial.html

[6] Device Classes. *The Things Network* [online]. Amsterdam, Netherlands: The Things Industries, 2021 [cit. 2021-11-23]. Dostupné z: https://www.thethingsnetwork.org/docs/lorawan/classes/

[7] SINHA, Rashmi Sharan, Yiqiao WEI a Seung-Hoon HWANG. A survey on LPWA technology: LoRa and NB-IoT. *ICT Express*. 2017, **3**(1), 1-8. ISSN 2405-9595.

[8] *LoRaWAN 1.0.3 Specification* [online]. In: . Beaverton, OR USA: LoRa Alliance, 2018 [cit. 2021-11-27]. Dostupné z: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf

[9] RASTOGI, Monika a Shalini SINGH. Emission of carbon dioxide from soil. *Current Science* [online]. 2002, (82), 510-517 [cit. 2022-05-29]. ISSN 00113891. Dostupné z: http://www.jstor.org/stable/24105957

[10] SOBKOVÁ, Barbora. *Denní cyklus CO2 v půdách* [online]. In: . Brno: MUNI, 2010 [cit. 2021-12-02]. Dostupné z: https://is.muni.cz/th/blj2x/reserse_Sobkova.pdf

[11] BALDOCCHI, Dennis, Eva FALGE, Lianhong GU, Richard OLSON a David HOLLING. FLUXNET: A New Tool to Study the Temporal and Spatial Variability of Ecosystem—Scale Carbon Dioxide, Water Vapor, and Energy Flux Densities. *Bulletin of the American Meteorological Society* [online]. 2001, **82**(11), 2415—2434 [cit. 2022-04-15]. Dostupné z: https://www.researchgate.net/publication/228863190_FLUXNET_A_New_Tool_to_Study_the_Temporal_and_Spatial_Variability_of_Ecosystem-Scale_Carbon_Dioxide_Water_Vapor_and_Energy_Flux_Densities

[12] GU, Lianhong et al. The fundamental equation of eddy covariance and its application in fluxmeasurements. *Agricultural and Forest Meteorology* [online]. 2012, (152), 135-148 [cit. 2021-12-04]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0168192311002905

[13] WANG, Miao et al. Comparison of eddy covariance and chamber-based methods for measuring CO2 flux in a temperate mixed forest. *Tree Physiology* [online]. 2010, (30), 149-163 [cit. 2021-12-04]. Dostupné z: https://academic.oup.com/treephys/article/30/1/149/1648371?login=true

[14] JURÁŇ, Radovan. *Field sensor network for microclimatological measurements.* [online]. Brno, 2020 [cit. 2021-12-04]. Dostupné z: https://www.vut.cz/en/students/final-thesis/detail/121803. Diplomová práce. VUT v Brně. Vedoucí práce Ing. Aleš Povalač.

[15] *Multiprotocol LPWAN dual core 32-bit Arm® Cortex®-M4/M0+ LoRa®, (G)FSK, (G)MSK, BPSK, up to 256KB Flash, 64KB SRAM* [online]. In: . Geneva, Switzerland: STMicroelectronics, 2021 [cit. 2021-11-30]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32wl55cc.pdf

[16] PEŇA, Eric a Mary Grace LEGASPI. UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter. In: *Analog Devices* [online]. Wilmington, MA USA: Analog Devices, 2020 [cit. 2021-11-30]. Dostupné z: https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

[17] Externí sériové sběrnice SPI a I²C. *ROOT* [online]. online: Internet Info, 2008, [cit. 2021-12-01]. Dostupné z: https://www.root.cz/clanky/externi-seriove-sbernice-spi-a-i2c/

[18] Komunikace po sériové sběrnici I2C. *ROOT.cz* [online]. online: Internet Info, 2009 [cit. 2021-12-01]. Dostupné z: https://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/

[19] *MH Z16 Intelligent Infrared Gas Module* [online]. In: . [cit. 2021-11-23]. Dostupné z: https://www.winsen-sensor.com/d/files/MH-Z16.pdf

[20] CO2 — Measurement Theory — Basics. *JLC International* [online]. New Britain, PA USA: JLC International, 2005 [cit. 2021-12-06]. Dostupné z: https://jlcinternational.com/co2-measurement-theory-basics/

[21] 1-Wire Communication Through Software. *Maxim integrated* [online]. San Jose, CA USA: Maxim Integrated Products, 2002 [cit. 2021-12-06]. Dostupné z: https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/126.html

[22] EARL, Bill. *Adafruit Data Logger Shield* [online]. In: . 15.11. 2021 [cit. 2022-05-28]. Dostupné z: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-data-logger-shield.pdf

[23] STM32CubeIDE user guide. In: *STMicroelectronics* [online]. 2021 [cit. 2022-05-25]. Dostupné z: https://www.st.com/resource/en/user_manual/dm00629856-stm32cubeide-user-guide-stmicroelectronics.pdf

[24] *DS1307 64 x 8, Serial, I2C Real-Time Clock* [online]. In: . 2015 [cit. 2022-05-28]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/DS1307.pdf

[25] *FatFs - Generic FAT Filesystem Module* [online]. 2009 [cit. 2022-05-08]. Dostupné z: http://elm-chan.org/fsw/ff/00index_e.html

[26] LoRaWAN Airtime calculator, LoRa Packet duration calculation. *RF Wireless World* [online]. 2012 [cit. 2021-12-12]. Dostupné z: https://www.rfwireless-world.com/calculators/LoRaWAN-Airtime-calculator.html

[27] Building wireless applications with STM32WB Series microcontrollers. *STMicroelectronics* [online]. STMicroelectronics, 2021 [cit. 2021-12-13]. Dostupné z: https://www.st.com/resource/en/application_note/an5289-building-wireless-applications-with-stm32wb-series-microcontrollers-stmicroelectronics.pdf

[28] Network Architecture. *The Things Network* [online]. Amsterdam, Netherlands: The Things Industries, 2016 [cit. 2021-12-11]. Dostupné z: https://www.thethingsnetwork.org/docs/network/architecture/

[29] *Uplink Decoder* [online]. Amsterdam, Netherlands: The Things Industries, 2021 [cit. 2022-04-20]. Dostupné z: https://www.thethingsindustries.com/docs/integrations/payload-formatters/javascript/uplink-decoder/

[30] *Applications and integrations* [online]. Amsterdam, Netherlands: The Things Industries, 2021 [cit. 2022-03-08]. Dostupné z: https://www.thethingsnetwork.org/docs/applications-and-integrations/

Seznam symbolů a zkratek

ABP	Activation By Personalization
API	Application Programming Interface
CPU	Central Processing Unit
CSS	Chirp Spread Spectrum
CR	Coding rate
csv	Comma Separated Values
CRC	Control Redundancy Check
FDMA	Frequency-division multiple access
GPIO	General Purpose Input Output
GSM	Global System for Mobile communications
HAL	Hardware Abstract Layer
HTTP	Hypertext Transfer Protocol
ISM	Industrial, Scientific and Medical
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
LTE	Long Term Evolution
LPWAN	Low Power Wide Area Network
MAC	Media Access Control
MCU	Microcontroller unit
NB-IoT	Narrowband Internet of Things
OSI	Open Systems Interconnection
OFDMA	Orthogonal frequency-division multiple access
OTAA	Over The Air Activation

QPSK Quadrature Phase Shift Keying

PWM Pulse Width Modulation

QoS Quality of Service

REST Representation State Transfer

RSSI Received Signal Strength Indicator

SD Secure Digital

SNR Signal to Noise Ratio

SF Spreading Factor

SoC System on Chip

SPI Serial Pheripheral Interface

ToA Time on Air

TTI The Things Industries

TTS The Things Stack

URL Uniform Response Locator

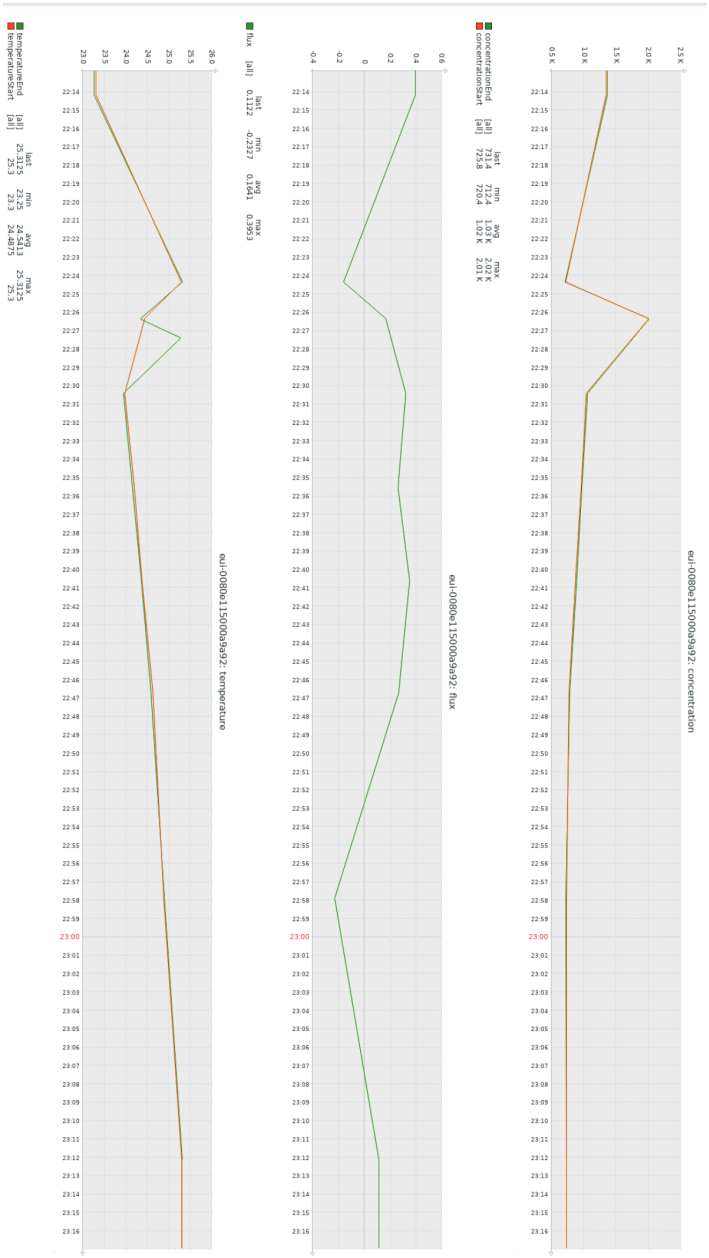
UART Universal Asynchronous Receiver and Transmitter

A Skript pro parsování JSON v jazyce PHP

Výpis A.1: Zdrojový kód pro zpracování JSON souboru

```
<?php
function openURL($url)
{
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    $data = curl_exec($ch);
    curl_close($ch);
    return $data;
}

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

$post = file_get_contents('php://input');
$json_a = json_decode($post,true);

$mydata = $json_a['uplink_message']
    ['decoded_payload']['measurements'];
$rxtime = $json_a['uplink_message']
    ['rx_metadata'][0]['time'];
$id = $json_a['end_device_ids']['device_id'];
$time_input = strtotime($rxtime);
$date_input = getDate($time_input);

foreach ($mydata as $key => $value)
{
    $mydateitme = $date_input['year']."-"
                .$date_input['mon']."-"
                .$date_input['mday']."-"
                .$date_input['hours']."-"
                .$date_input['minutes']
                ."-".$date_input['seconds'];
    $zabbixurl = "http://endpointurl.cz:8888/AutomerData"
."/?host={$id}&var={$value}&zkey={$key}&dt={$mydateitme}";
    echo openURL($zabbixurl);
}?>
```

B Zobrazení dat v prostředí Zabbix



Obr. B.1: Vykreslené grafy v prostředí Zabbix