TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Inteligentní systém pro anonymní detekci počtu osob v místnosti

Bakalářská práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídící systémy

Autor práce: **Jan Tichý**
Vedoucí práce: Ing. Lenka Kosková Třísková

Liberec 2018

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Intelligent system for anonymous detection of the number of people in the room

Bachelor thesis

Study programme: N2612 – Electrotechnology and informatics
Study branch: 2612R011 – Electronic Information and Control Systems

Author: **Jan Tichý**
Supervisor: Ing. Lenka Kosková Třísková

Liberec 2018

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Tichý**
Osobní číslo:	**M15000120**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Inteligentní systém pro anonymní detekci počtu osob v místnosti**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Proveďte rešerši vhodného HW k sestavení systému, jenž měří CO2 v uzavřené místnosti a ukládá data do online úložiště.
2. Navržený systém kromě měření CO2 dále zaznamenává jiná data vhodná k určení počtu lidí v místnosti.
3. Systém doplňte o rozhraní, jež s malým zpožděním inofrmuje o naměřených hodnotách přímo v místnosti.
4. Systém zprovozněte a testujte v učebně.
5. K vyhodnocení měření a tvorbě záznamů využijte nástrojů zvoleného úložiště (např. AWS či Azure).

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **30 - 40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] Greengard, Samuel: The Internet of Things (The MIT Press Essential Knowledge series), MIT Press, 2015, ISBN: 978-0262527736
[2] Ruparelia Nayan B.: Cloud Computing (The MIT Press Essential Knowledge series), MIT Press 2016, ISBN: 978-0262529099
[3] Yiu, Joseph: The Definitive Guide to ARM? Cortex?-M3 and Cortex?-M4 Processors, Third Edition, 3th Edition, Newnes, 2013, ISBN: 978-0124080829

Vedoucí bakalářské práce: **Ing. Lenka Kosková - Třísková**
Ústav nových technologií a aplikované informatiky

Datum zadání bakalářské práce: **19. října 2017**

Termín odevzdání bakalářské práce: **14. května 2018**

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

Technická univerzita v Liberci - 14 -

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2017

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 14.5.2018

Podpis:

Poděkování

Rád bych poděkoval své vedoucí práce paní Ing. Lence Koskové Třískové, za vedení mé bakalářské práce, trpělivost, pevné nervy, užitečné rady a hlavně za náhled z jiné perspektivy na problémy, které jsem během práce potřeboval řešit. Také bych chtěl poděkovat Ing. Jiřímu Šindelářovi z firmy Jablotron za podporu, technickou podporu a prospěšné rady. Nakonec bych chtěl poděkovat firmě Jablotron, za poskytnutí přístupu ke službám AWS.

Abstrakt

Bakalářská práce se zabývá inteligentním systémem, který pomocí měření profilu koncentrace oxidu uhličitého umožňuje určit počet osob v místnosti. Systém je založený na vestavěném IoT zařízení s operačním systémem FreeRTOS a propojením s cloudovými službami. Projekt využívá NoSQL databázi pro uchování měřených, Lambda funkci pro výpočet na straně serveru a S3 bucket pro hostování webového rozhraní. Práce provádí rešerši dostupného HW/SW a vysvětluje, jak pracuje vyhodnocení počtu osob na základě měření koncentrace plynu CO_2. Data o množství osob může být využitelný pro řízení ventilačních systému, a tím pro snížení spotřeby elektrické energie nebo zlepšení životních podmínek v uzavřených prostorech. Celá práce je zakončena experimentální měřením v učebně A10 na Technické Univerzitě v Liberci.

Klíčová slova:

Oxid uhličitý, Internet věcí, NoSQL, Amazon Web Services, FreeRTOS

Abstract

The bachelor thesis present an inteligent system for detection of occupants indoor based on carbon dioxide concentration in school class. System is based on IoT device with FreeRTOS operating system and connected to Amazon cloud services. I use ecosystem Amazon Web services that include NoSQL database, Lambda function for calculation of occupants on server side and webhosting. I made complex research of available HW/SW. I explain how work dynamic algorith for detection of occupants indoor. Nubmer of occupants can be use for reducing electricity consumption, control of ventilation systems or improving living conditions indoor. The whole bachelor thesis ends with experimental measurment on actual data.

Key words:

Carbon dioxide, Internet of things, NoSQL, Amazon Web Services, FreeRTOS

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

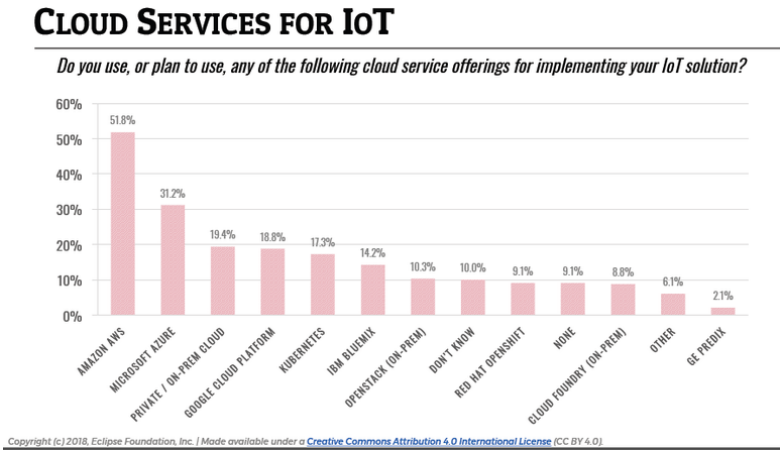
TUL	Technická univerzita v Liberci
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
IoT	Internet věcí
AWS	Amazon Web Services
MCU	Microcontroller unit
RTOS	Real-time operating system
OS	Operační systém
SDK	Software development kit
FIFO	Firts in First out
CCS	Code Composer Studio
LTS	Long Term Support
CO_2	Oxid uhličitý

Úvod

Pokud se ohlédneme stručně do historie, internet odstartoval svou etapu už koncem šedesátých let minulého století. Od počátku tisíciletí se začínají objevovat první zmínky o tzv. „*internetu věcí (IoT)*“. Hlavním cílem IoT je vytvoření sítě, ve které jsou připojená zařízení. Ty poskytují data, která budou schopna reagovat na povely nebo odesílat informace o svém stavu. Až okolo roku 2016 se koncepce IoT rozšířila masově do dalších odvětví, jako jsou například vestavěné systémy, bezdrátové senzory, chytré domy, automatizace atd. Tento segment trhu se viditelně rozvíjí a za posledních pár let se objevilo mnoho IoT řešení. Velmi diskutovanou formou IoT je propojení vestavěných aplikací na cloudové služby.

Čtvrtý ročník průzkumu vývojářů IoT od Eclipse Foundation zjišťoval hlavní IoT platfromu na trhu. Respondenti odpověděli takto, *51.8* procenta vývojářů uvedla *Amazon Web Services (AWS)* jako jejich hlavní IoT platformu, *31.22* procenta uvedla *Microsoft Azure*. *Google Cloud Platform* uvedlo *18.79* procent respondentů. Průzkum byl prováděn na 502 vývojářích v roce 2017. Pro srovnání nejoblíbenějších IoT platforem průzkum uvádí obrázek č.1 [18].

Tato práce se zaměřuje na vytvoření aplikace pro anonymní detekci osob v uzavřeném prostředí s použitím principu IoT. Informace o množství osob prostoru může pomoci řídit vzduchotechniku, automatizovat otevírání oken a omezit tak spotřebu energie.



Obrázek 1: Žebříček nejpoužívanějších cloudů v roce 2017 [18]

1 Analýza

V této kapitole jsem se nejprve zabýval metodikou měření oxidu uhličitého a důvody jeho detekce. Následně jsem se zaměřil na výběr hardwaru, který bude tvořit tuto aplikaci. Popsal jsem algoritmus na výpočet lidí v místnosti založený na hmotnostní rovnici. Rozebral jsem dostupné možnosti vývoje vestavěných systémů a použití operačních systémů. Vybral jsem jednu cloudovou platformu a popsal služby, které jsou k práci potřeba a v neposlední řadě jsem uvedl základní parametry místnosti, kde se úloha aplikovala.

1.1 Měření CO_2

Měření koncentrace oxidu uhličitého provádíme za účelem zjištění kvality ovzduší v měřeném prostředí. Vysoká koncentrace oxidu uhličitého může mít nežádoucí účinky na lidské zdraví. Při dlouhodobé expozici člověka vysokou hodnotou CO_2, řádově okolo 1500 ppm a více, může docházet k únavě a ztrátě pozornosti. Z tohoto důvodu je více než vhodné udržovat na pracovišti nebo v domácnostech pravidelnou výměnu čerstvého vzduchu [5].

Kvalita ovzduší je důležitá jak pro dodržení hygienických limitů, tak i pro produktivitu lidí. Touto problematikou se zabývá vyhláška stavebního zákona č. 268/2009 Sb. o technických požadavcích na stavby [8].

Pokud je hodnota koncentrace oxidu uhličitého v prostředí známá, tak lze automaticky řídit vzduchotechnické systémy nebo určovat aktuální množství lidí v místnosti. Takto se dá anonymně měřit aktuální počet lidí na pracovišti a na základě těchto údajů dodat více vzduchu do místnosti a pouštět odvětrávání. Dá se takto efektivně šetřit elektrickou energií. Anonymní sběr dat je výhodný, protože není potřeba mít souhlas lidí v objektu s jejich monitoringem pomocí kamerového systému, který by mohl množství lidí v prostředí také vyhodnocovat.

Při měření koncentrace CO_2 je nutné počítat se všemi producenty, které mohou CO_2 produkovat. Taková chyba měření může nastat při v případě otevřených dveří nebo oken, popřípadě velkého množství rostlin. Dále do měření zasahují netěsností oken, které trvale propouští atmosféru z vnějších prostor do sledovaného prostředí.

Základní požadavek na senzor oxidu uhličitého je měřený rozsah, který se pohybuje okolo 5% nebo-li 5000 ppm měřené atmosféry. V běžném prostředí, kde se lidé pohybují, by neměla hodnota tento rozsah překročit.

1.1.1 Charakteristika CO_2 a jeho působení na člověka

Oxid uhličitý je bezbarvý plyn, bez zápachu, 1,52x těžší než vzduch. Vzniká reakcí kyslíku s uhlíkem - oxidací organických látek, spalováním uhlovodíků, spalováním CO a je produktem látkové výměny většiny organismů. Tím, že je plyn těžší než vzduch, tak se drží při zemi. Proto jej ve vyšších koncentracích můžeme nalézt v jámách, kanálech nebo studních.

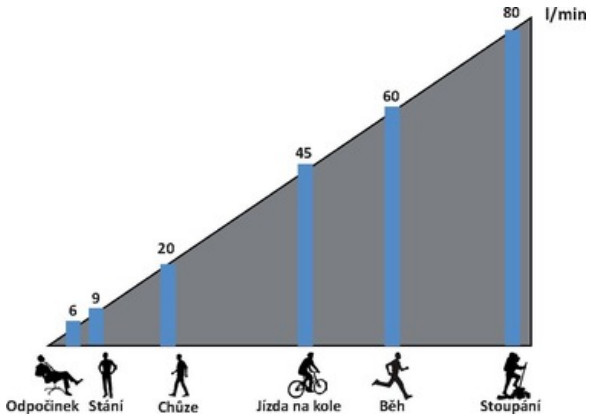
Jak jsem již v úvodu zmínil, základní hladina koncentrace CO_2 je do 1 500 ppm. Hodnoty nad touto hranicí už mohou mít viditelný vliv na lidský organismus. Běžná venkovní koncentrace se pohybuje okolo 350-450 ppm, záleží na teplotě, tlaku, ročním období, počasí, popřípadě dopravním provozu.

Vystavení člověka vyšším hodnotám CO_2, řádově nad 5 000 ppm, může vést k silným bolestem hlavy, nevolnosti, hučení v uších nebo zvýšení krevního tlaku. Více o vlivu oxidu uhličitého na lidi uvádím v tabulce 1.1.

Koncentrace CO_2	**Komentář, symptomy**
$<$ 400 ppm	koncentrace ve venkovním vzduchu
$<$ 1 000 ppm	doporučená úroveň CO_2 ve vnitřních prostředí
$<$ 1 500 ppm	maximální doporučená úroveň CO_2 ve vnitřním prostředí
$>$ 1 500 ppm	příznaky únavy, snižování koncentrace, ospalost, letargie
$<$ 5 000 ppm	maximální bezpečná koncentrace CO_2 bez zdravotních rizik
$>$ 5 000 ppm	příznaky nevolnosti, bolesti hlavy, zvýšený tep
$>$ 10 000 ppm	při dlouhodobém působení prokázány zdravotní problémy
$>$40 000 ppm	životu a zdraví nebezpečná koncentrace

Tabulka 1.1: Vliv oxidu uhličitého na lidský organismus [6]

Produkce oxidu uhličitého lidským organismem je ovlivněna jeho fyzickou aktivitou, věkem, váhou a pohlavím. Vydechovaný vzduch obsahuje cca 4 % obj. CO_2. Na obrázku 1.1 jsou znázorněny typické fyzické činnosti s uvedením potřeby vzduchu v litrech za minutu[6].



Obrázek 1.1: Orientační graf potřeby vzduchu v závislosti na fyzické činnosti [6]

1.1.2 Porovnání metod měření CO_2

Pro měření koncentrace oxidu uhličitého v atmosféře existují tři metody měření:

- Optická metoda
- Elektrochemická metoda
- Polovodičová metoda

V následujících kapitolách přiblížím princip, jakým senzory zmíněné metody používají a jaké jsou jejich výhody a úskalí.

Optický infračervený senzor

Tato metoda měření pracuje na základě absorpce částí infračerveného spektra v molekulách CO_2. Senzor obsahuje zdroj infračerveného světla, které prochází komůrkou nasměrováno na optický detektor, v části před optickým filtrem se zde dostávají molekuly CO_2, kterým se to komůrky dostávají molekuly CO_2. Detektor vyhodnocuje pokles infračerveného záření. Čím více je v komůrce molekul CO_2, tím méně dopadá infračerveného světla na detektor. Z tohoto poklesu jde spočítat koncentrace plynu v prostředí.

Jistým limitujícím faktorem měřících vlastností v infračerveném spektru je skutečnost, že vyšší koncentrace CO_2 vede k tzv. „oslepnutí“ senzoru. Více molekul je schopno pohltit prakticky veškeré IR záření sledovaných vlnových délek, což se projeví na úbytku kvality měřícího signálu v oblasti vyšších měřících rozsahů[6].

Pro vhodnou detekci koncentrace oxidu uhličitého se nejlépe dají využít vlnové délky 7.20 µm, 14.99 µm a 4.256 µm [9].

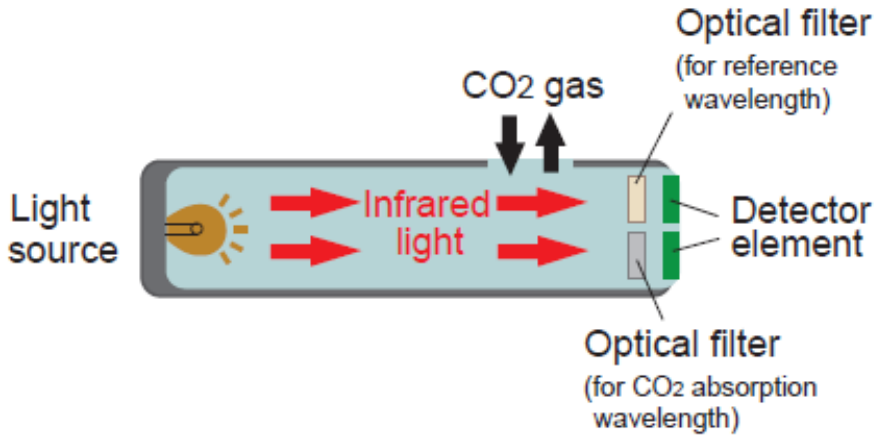


Fig. 1 - Basic structure of CDM7160 optics

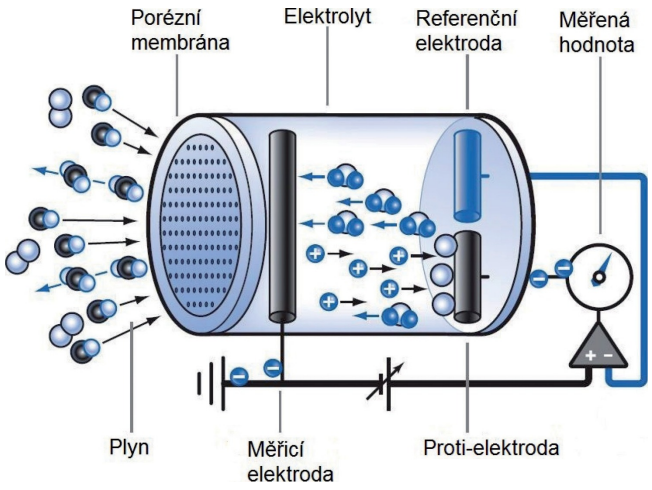
Obrázek 1.2: Princip měření optického senzoru CDM7160 [10]

Elektrochemický senzor

Základní princip funkce elektrochemického senzoru je vytváření přímoúměrného signálu na základě reakce molekul oxidu uhličitého a elektrolytu, který je uvnitř detektoru. Porézní membránou projdou jenom molekuly CO_2, které naráží do měřící elektrody. Tím na ní dojde k elektrochemické reakci, jejímž důsledkem je vznik volných elektronů. Ty putují na elektrolytem na referenční elektrodu. Je potřeba tento nízkoproudový signál změřit a zesílit. Velikost proudu odpovídá koncentraci plynu [6].

Takto lze velice přesně měřit vlastnosti plynu. Nevýhodou této metody je, že díky využití elektrolytu je životnost senzoru cca 1 až 2 roky. Ke stárnutím čidla dochází díky chemickým změnám vedoucím k postupnému vyčerpání elektrolytu. Aby nedocházelo k velkých výchylkám v měření, je zapotřebí čidlo pravidelně kalibrovat změnou citlivosti senzoru.

Zatímco se přesnost měření bezprostředně po kalibraci pohybuje v rozmezí ±5 %, mohou chyby měření již po 1 až 3 měsících přesahovat 20 %. Vzhledem k potřebě častější kalibrace se elektrochemické senzory pro měření CO_2 používají častěji v přenosných přístrojích, ve stacionárních systémech se téměř nepoužívají [6].

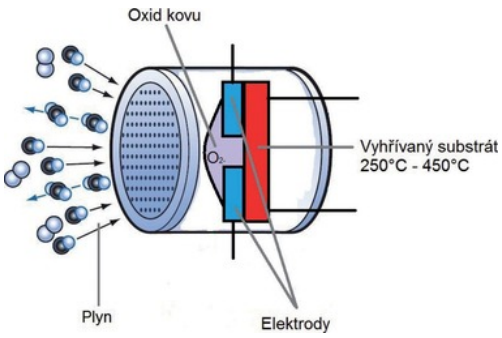


Obrázek 1.3: Princip měření elektrochemického senzoru [6]

Polovodičový senzor

Polovodičový senzor měří koncentraci oxidu uhličitého na základě změny vodivosti. Jde o nejlevnější řešení měření CO_2, ale jeho měřící vlastnosti jsou velice nepřesné. Pro úlohu čidla jsou použité oxidy kovů (např. oxidy zinku, cínu, wolframu, india) [6]. Na povrchu tohoto materiálu se vytvoří ve vzduchu rovnovážný stav s molekulami kyslíku, který se za přítomnosti jiného plynu poruší a způsobí změnu vodivosti [6].

Vzhledem k nepřesnosti této metody měření je využití polovodičového čidla hlavně jako signalizace překročení doporučených limitů. Hlavní výhodou je cena a dlouhodobá životnost senzorů v čistém prostředí.



Obrázek 1.4: Princip měření polovodičového senzoru [6]

1.1.3 Zdůvodnění zvolené měřící metody

Po provedení rešerše jsem došel k závěru, že nevhodnější je pořídit čidlo s použitím optické infračervené metody měření CO_2. Výhodou je dlouhá životnost (garance výrobcem 10 let), přesnost měření a stálost. Není žádoucí, aby se někdo musel pravidelně k čidlu dostávat, a provádět kalibraci, která by v případě zvolení elektrochemického principu byla nutná každé tři měsíce.

1.2 Rešerše dostupného hardwaru

Výběr hardwaru může být zásadní pro celou práci. Nejprve je potřeba zvolit vhodný vývojový kit, který bude tvořit základní zařízení, od něhož se budou určovat další komponenty. Vzhledem k rozmanitosti trhu jsem se v následující kapitole zaměřil na dostupné desky podle předem zadaných parametrů. Na těchto parametrech se spolupodílí i firma *Jablotron*, se kterou v rámci této práce spolupracuji.

Po výběru vývojového kitu je zapotřebí vybrat vhodný detektor CO_2. Kromě cenové dostupnosti byl kladen důraz i na životnost a přesnost měření.

1.2.1 Výběr vývojového kitu

Základní požadavky na výběr desky jsem volil na základě spolupráce s firmou Jablotron. Jejich hlavní požadavek byl využití operačního systému FreeRTOS. Výhodou tohoto operačního systému je v managmentu úloh, které se volají podle potřeby, deterministického chování. Dále je přesně určeno místo v paměti pro jednotlivé úlohy. Na základě tohoto požadavku se odvíjí i následující parametry:

- Podpora FreeRTOS
- Paměť RAM alespoň 100 kB
- Internetová konektivita (LAN nebo WiFi)
- Sběrnice UART a I^2C
- Cena do 2 500 Kč

Cenu do výše 2 500 Kč jsem zvolil jako hlavní kritérium pro snížení počtu desek. Na základě těchto parametrů jsem vybral následující desky viz. tabulka 1.3 na straně 21.

Z dostupných desek jsem vybíral bodovou metodou. Každému kandidátovi jsem dal bod za každou kategorii, kterou splňoval. Nejlépe z toho vyšla deska CC3220SF, která splnila všechna nutná kritéria, ale také ji lze jednoduše připojit ke cloudům pomocí software development kitu (SDK) (AWS, Azure, IBM).

1.2.2 Výběr senzoru na měření CO_2

Na trhu se objevuje hned několik vhodných čidel pro měření hodnoty CO_2. Pro výběr vhodného čidla pro tento projekt byla následující kritéria:

- Dostupnost
- Výstupy (analogový, digitální)
- Rozsah hodnot v ppm
- Napájení
- Životnost
- Cena

Dostupná čidla jsem shromáždil v tabulce 1.4 na straně 21. Chybějící údaje v tabulce jsou znakem toho, že čidlo není schopno komunikovat tímto způsobem nebo výrobce tento údaj neuvádí. Cena u čidla Comet nebyla dostupná, jedná se průmyslové řešení, pro které dodavatel stanovuje cenu dle zakázky. Z těchto dostupných údajů jsem jako nejlepší čidlo vybral **CDM7160**. Rozsah pro měření je dostatečný. S digitálními výstupy dokáže deska **CC3220SF** komunikovat. Výrobce garantuje dlouhodobou životnost a také certifikovanou kalibraci.

Čidlo CDM7160 má 11 pinů. V tabulce 1.2 je uvedena funkce jednotlivých pinů:

Pin	Název	Popis
1	V_{dd}	Vstupní napětí
2	GND	Zem
3	Alarm	Výstup alarmu na 1 000 ppm
4	PWM	PWM výstup
5	CAD0	Výběr I^2C adresy (interní zdvihací)
6	MSEL	Výběr komunikace I^2C/UART, log 0 je vybrán I^2C
7	CAL	Pokud je na tento vstup přivedena log 0, je aktivován mód kalibrace
8	BUSY	BUSY výstupní signál
9	Tx/SDA	UART Tx výstup/ I^2C SDA I/O
10	Rx/SLC	UART Rx vstup/ I^2C SCL vstup
11	NC	Nepřipojeno

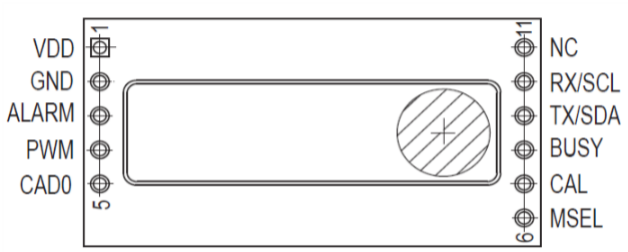
Tabulka 1.2: Popis pinů [10]

Název desky	Výrobce	Sběrnice	RAM [kB]	Konektivita	FreeRTOS	Cena
CC3220SF	Texas Instrument	SPI, UART, I^2C	256	WiFi	Ano	1000,-
FRDM-K64F	NXP	SPI, UART, I^2C, CAN	256	LAN	Ano	800,-
Nucleo-F746ZD	ST	I^2C,SPI,CAN,USB,UART	320	Ø	Ano	600,-
Seeedstudio-Arch-Pro	Seeed	SPI, UART, I^2C	64	LAN	Ano	900,-
TM44C1294	Texas Instrument	USB, I^2C, SPI, UART	256	LAN	Ano	460,-

Tabulka 1.3: Výběr desek na trhu podle zadaných kritérií

Název	Rozsah (ppm)	Způsob měření	Analog.	Digital.	Životnost	Cena
CDM7160	300-5 000	Optický	Ø	PWM, UART, I^2C	10 let	1 600,-
SE-0018	0-10 000	Optický	0-10V, 0-5V	OUT	15 let	2 000,-
MH-Z14A	0-5 000	Optický	0-2,5V, 0,4-2V	Ø	5 let	700,-
MG811	350-10 000	Polovodičový	0-2V, 0-4V, TTL	Ø	Ø	1 000,-
MH-Z16	400-10 000	Optický	Ø	UART, I^2C	Ø	1 600,-
Comet	0-2 000	Optický	Ø	Ø	RS232/485, ethernet	Ø

Tabulka 1.4: Výběr čidel na trhu podle zadaných kritérií



Obrázek 1.5: Rozmístění pinů čidla [10]

V tabulce jsou šedou barvou označeny ty vstupy, které musí pro chod čidla být zapojené. První dva vývody jsou jasné, jedná se o připojení napájení TTL logikou. **Alarm** je výstup čidla, lze nastavit při jaké hodnotě CO_2 se alarm spustí. V základním nastavení se alarm aktivuje při hodnotě *1000 ppm*. Lze takto například spustit jednoduchou signalizaci překročení určitého limitu přímo z čidla.

PWM výstup je nastavený na 1kHz v rozsahu měření od 0-5000 ppm. Vzhledem k tomu, že digitální výstup nabízí mnohem větší měřící rozsah, jsem tuto možnost měření vynechal.

CAD0 rozhoduje o adrese čidla. V tomto případě je možné mít na jedné sběrnici pouze dvě čidla tohoto typu, protože o adrese rozhoduje právě jeden bit. Tento limit v práci nepředstavoval žádný problém.

MSEL určuje jakou sběrnicí bude čidlo komunikovat. Lze zvolit sběrnice I^2C a UART. Pro výběr sběrnice I^2C je nutné na tento vstup přivést logickou nulu.

Vstup **CAL** je určený ke kalibraci, pokud je přivedena nula na tento vstup, tak se čidlo kalibruje na hodnotu, která je právě v jeho okolí jako nulová hodnota. Tato kalibrace byla již od výrobce vytvořena, proto jí již není třeba provádět.

BUSY bude na výstupu logická jednička v případě, kdy není možné číst data, protože ADC se připravuje na konverzaci nebo čtení dat už bylo započato. Procesor načte data a potom potvrdí, že tento bit se změní z logické jedničky na nulu. Tento výstup je indikace toho, zdali je možno čtení či nikoliv.

Výstupy/vstupy devět a deset jsou určeny pro komunikaci na sběrnici podle výběru na pinu MSEL.

Komunikace s čidlem CMD7160 na sběrnici I^2C

Pro tuto aplikaci jsem zvolil sběrnici I^2C, protože jsem se rozhodl UART sběrnici zachovat pro případné rozšíření aplikace o další přídavný modul, který by mohl zpřesnit moje řešení.

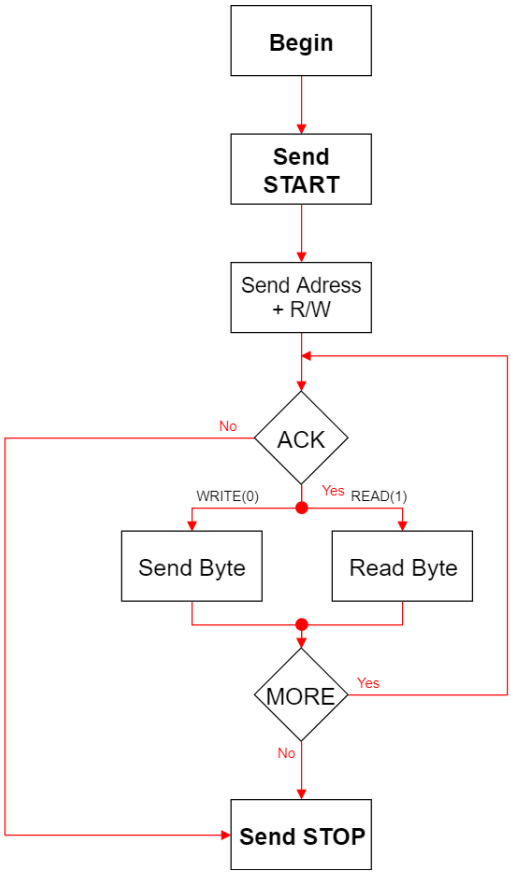
Aby komunikace s čidlem mohla proběhnout, je zapotřebí znát adresu čidla na sběrnici I^2C. U senzoru CMD7160 můžeme zvolit pouze dvě adresy. Adresa se nastavuje pomocí **MSEL** kontaktu. Ten po přivedení logické jedničky na poslední bit adresy přidá logickou jedničku. Tabulka níže ukazuje, jak taková adresa vypadá. Pokud chci ze senzoru hodnotu číst, použiji na LSB logickou jedničku a pokud chci zapisovat, tak doplním na nejnižší bit logickou nulu.

Hodnota CO_2 je v čidle zapsaná v dvou osmibitových registrech. Když chceme registry přečíst, pošleme na I^2C sběrnici adresu čidla s logickou jedničkou na konci

MSB							LSB	Adresa (hex)
1	1	0	1	0	0	CAD0	R/W	
1	1	0	1	0	0	1	1	0xD3
1	1	0	1	0	0	1	0	0xD2

Tabulka 1.5: Výběr adresy čidla CMD7160 [10]

adresy **R/W** bitu. Blokové schéma 1.6 ukazuje, jak komunikace po sběrnici I²C funguje.



Obrázek 1.6: Blokové schéma komunikace I²C

1.3 Algoritmus na výpočet lidí v místnosti

Detekci osob v uzavřených prostorách lze provést pomocí různých senzorů jako například čidla PIR, video-kamera, infračervená kamera, měření rádiových vln zařízení, počet přihlášených počítačů v učebně nebo pomocí koncentrace oxidu uhličitého.[4]

Vzhledem k citlivosti sběru dat na školní půdě jsem zvolil formu měření anonymním způsobem a to pomocí koncentrace CO_2.

1.3.1 Algoritmus založený na hmotnostní rovnici

Nejjednodušší algoritmus pro detekci přítomnosti lidí v uzavřeném prostředí lze založit čistě na analýze gradientu monitorovaného profilu koncentrace oxidu uhličitého. Tento způsob dokáže odhalit přítomnost lidí na základě jednoho vstupu, ale nedokáže rozlišit počet. Problémem této metody je, že porucha ve formě otevřeného okna znehodnotí výsledek měření.

Další způsob vyhodnocení množství lidí v místnosti je pomocí hmotností rovnice.

$$n_{occ}C_{prodpp} + \dot{m}_{vent}C_{vent} - \dot{m}_{vent}C_R = V\frac{\mathrm{d}C_R}{\mathrm{d}t} = 0 \tag{1.1}$$

1.1: Obecný tvar hmotnostní rovnice[4]

Z rovnice 1.2 lze získat dvě informace, když je známá hodnota CO_2, tak můžeme určit počet lidí nebo můžeme predikovat hodnotu CO_2 na základě počtu lidí v místnosti. Budeme-li předpokládat, že vzduch v místnosti je homogenní a za předpokladu ideálního směšování dodávky okolního vzduchu. V našem případě se bude měřit množství lidí v učebně, která nemá klimatizaci ani žádnou vzduchotechniku na výměnu atmosféry.

$$[C]_{i+1} = \left(1 - \frac{\frac{[\dot{m}_{airx}]}{\rho_{air}}[\Delta t]_i}{V_{office}}\right)[C]_i + \frac{\frac{[\dot{m}_{v,amb}]_i}{\rho_{air}}[\Delta t]_i}{V_{office}}[C_{amb}]_i + \frac{\frac{[\dot{m}_{v,in}]_i}{\rho_{air}}[\Delta t]_i}{V_{office}}[C_{adj}]_i + \frac{[n_{occ}]_i[\Delta t]_i}{V_{office}}[C_{PpP}]_i \tag{1.2}$$

1.2: Hmotnostní rovnice na výpočet množství CO_2[4]

$$[\dot{m}_{air}] = [\dot{m}_{v,amb}] + [\dot{m}_{v,in}] = [\dot{m}_{mv}] + [\dot{m}_{inf}] + [\dot{m}_w] + [\dot{m}_d] \tag{1.3}$$

1.3: Hmotnostní rovnice[4]

- $[\dot{m}_{airx}]$ Celkový součet hmotnosti vzduchu $(kg \cdot s^{-1})$
- $[C]$ Změřená hodnota koncentrace CO_2 v místnosti (ppm)
- $[\dot{m}_{v,amb}]$ Výměna vzduchu skrz ventilace, jedná se o hmotnost vzduchu

$$[n_{occ}]_i = \frac{\left[\left(1-\frac{\frac{[\dot{m}_{airx}]_i}{\rho_{air}}[\Delta t]_i}{V_{office}}\right)[C]_{i-1}+\frac{\frac{[\dot{m}_{v,amb}]_i}{\rho_{air}}[\Delta t]_i}{V_{office}}[C_{amb}]_i+\frac{\frac{[\dot{m}_{v,in}]_i}{\rho_{air}}[\Delta t]_i}{V_{office}}[C_{adj}]_i\right]-[C]_i}{\frac{[C_{PpP}]_i[\Delta t]_i}{V_{office}}} \quad (1.4)$$

1.4: Aproximace vzorce 1.2 na detekci množství osob[4]

přicházející z okolního

- $[C_{amb}]$ Koncentrace CO_2 v okolního
- $[\dot{m}_{v,in}]$ Hmotnost vzduchu v přilehlém okolí měřené místnosti
- $[C_{Adj}]$ Koncentrace CO_2 v přilehlém okolního
- $\Delta t * [n_{occ}]$ Počet lidí v místnosti za určitou dobu
- $\Delta t * [C_{PpP}]$ Množství CO_2 produkované člověkem za jednotku času (L/s)
- ρ_{air} Hustota vzduchu $(kg \cdot m^{-3})$
- V_{office} Objem měřené místnosti (L)
- $[\dot{m}_{mv}]$ Mechanická ventilace
- $[\dot{m}_{inf}]$ Vzduch z okolního
- $[\dot{m}_w]$ Vzduch vyměněný okny
- $[\dot{m}_d]$ Vzduch vyměněný dveřmi

Když první rovnici aproximujeme, dostaneme se k rovnici na výpočet osob v uzavřených prostorách, tak jak to prezentuje vzorec 1.4.

Člověk, který nevykonává žádnou náročnou fyzickou aktivitu vyprodukuje za hodinu 19 litrů oxidu uhličitého. To dělá 0,0052 L/s. Do hmotnostní rovnice musíme hodnotu $[C_{PpP}] \cdot 10^6$, aby rovnice dávala správný výsledek [17]. Celkovou hodnotu vyměněného vzduchu nelze přesně měřit, proto se v algoritmu použijí jako konstanta v určitém rozmezí, podle parametrů místnosti. Jako konstanta se bere i parametr koncentrace přilehlých a venkovních prostor.

- $[\dot{m}_w]$: 1-10 l/h
- $[\dot{m}_{inf}]$: 0-0.4 l/h
- $[\dot{m}_d]$: 0-1 l/h
- $[C_{amb}]$: 340-460 ppm
- $[C_{Adj}]$: 400-1000 ppm

1.4 Operační systém RTOS

V této kapitole se budu zabývat tématem operačního systému reálného času. To znamená, že operační systém musí stihnout vykonat všechny úkoly tak, aby se na žádný nezapomnělo do konce časové uzávěrky. Každé jádro procesoru se může v jednom okamžiku vždy věnovat pouze jedné činnosti.

Úlohy jednotlivým procesům přiděluje tzv. plánovač úloh (scheduler), a to tak, aby byl optimálně využit procesor a další zdroje systému. Ten díky rychlosti pře-

pínání programů dokáže vytvořit iluzi, že je vykonáváno více činností najednou. Tomu se říká multitasking. Plánovač v operačním systému v reálném čase (RTOS) je navržen tak, aby se všechny úlohy stihly v termínu.

1.4.1 Výběr vhodného operačního systému pro aplikaci

FreeRTOS je otevřený operační systému, který je navržen tak, aby mohl běžet na mikrořadičích. Díky tomu je tento OS připravený na malé množství ROM i RAM paměti. Aplikace FreeRTOS je určena hlavně pro vestavěné (embeddované) aplikace.

Letos jsem měl možnost jet se podívat na veletrh Embedded World 2018 v Norimberku a zde jsem se setkal s velkým nasazením RTOS na různé IoT aplikace.

Pro vybranou základní desku CC3220SF mám na výběr tři možnosti implementace operačního systému a nebo vlastní řešení.

První možnost je použít TI-RTOS vyvíjený společností Texas Instrument přímo na tento hardware.

Druhá možností je použití FreeRTOS, který je opensource. Pro desku CC3220SF je veliké množství příkladů vytvořených pro FreeRTOS přímo na míru. Výhodou tohoto řešení je, že už je hotové SDK pro připojení na cloud AWS od Amazonu a Azure od Microsoftu.

Třetí variantu je použití přímo Amazon FreeRTOS, který je ze všech možností nejmladší. Zvolil jsem poslední možnost. Rozhodl jsem se pro volbu tohoto operačního systém, protože představuje rychlou a jednoduchou možnost integrace služeb AWS a FreeRTOS. Lze se připojit bezpečně a rychle do cloudu společnosti Amazon. Navíc Amazon FreeRTOS má vytvořené SDK pro nejnovější verzi firmwaru CC3220SF, tím se stává nejbezpečnější volbou.

Amazon FreeRTOS je operační systém pro mikrořadiče (MCU), který ulehčuje vývoj, nízko-energetickou náročnost, zabezpečení a konektivitu. Systém je založený na jádře FreeRTOS. Tento operační systém pro MCU je rozšířen o softwarové knihovny, které bezpečně připojuje mikrořadiče ke službě AWS IoT Core.

Vzhledem k limitaci výpočetního výkonu a paměti MCU je zapotřebí vytvořit program tak, aby i na tomto nízko výkonovém HW aplikace fungovala. Operační systém dokáže tyto aplikace rozdělené na úlohy (task), které se postupně volají podle toho, jak je nutně aplikace zrovna potřebuje. Vzhledem k tomu, že nepoužívám jenom operační systém, ale reálný operační systém, tak se úkoly budou volat v závislosti na čase expirace operace nebo na prioritě úkolu.

Výhodou systému FreeRTOS je přenositelnost. Tím pádem bude do budoucna možné tuto aplikaci přenést na jinou desku s jiným MCU a přitom zachovat funkčnost programu.

1.4.2 Amazon FreeRTOS

Pro použití Amazon FreeRTOS je zapotřebí být seznámen s následujícími pojmy. Ve zkratce zde uvedu, jejich funkci.

- Úloha (Task)
- Plánovač (Scheduler)
- Fronta (Queue)
- Semafor (Semaphore)
- Přepínače (Mutex)

Úloha

Úloha je druh funkce, kterou procesor vykonává podle toho, jak je plánovač přiděluje. Konkrétně se může jednat o čtení ze sběrnice, načítání hodnot z GPIO nebo třeba jenom čekání na časovač. Každá úloha vyžaduje určitě množství RAM, kam se ukládá stav úlohy, když je zrovna v blokovaném stavu. Když je task vytvořen pomocí funkce *xTaskCreate()*, tak se v paměti RAM alokuje část paměti z FreeRTOS heap.

Nově vytvořená úloha je inicializován v aktivním stavu. To znamená, že bude proveden ve chvíli, kdy nebude mít jiná úloha vyšší prioritu pro spuštění. Úlohu můžeme vytvořit před nebo po tom, co je plánovač spuštěn[11].

```
BaseType_t xTaskCreate( TaskFunction_t pvTaskCode,
const char * const pcName,unsigned short usStackDepth,
void *pvParameters, UBaseType_t uxPriority,
TaskHandle_t *pxCreatedTask );
```

pvTaskCode	Tento parametr je prostý ukazatel na funkci, ze které vytvoří úlohu
pcName	Popisuje, jak se úkol bude jmenovat. Vhodné hlavně při ladění aplikace
usStackDepth	Každá úloha má svůj vlastní zásobník, který je přidělen jádrem, když je úloha vytvářena. Tento parametr určuje počet slov, které se dají v zásobníku uchovat (nejedná se o počet bytů). Pro příklad do zásobníku, jehož délka jednoho zápisu jsou 4-byty a parametrem *usStackDepth = 100* bude možné v rámci jedné úlohy uložit 400 bytů informací.
pvParameters	Jedná se ukazatel. Hodnota, která je přidělena ukazatele bude přidána do úlohy a bude možno s ní dále pracovat.
uxPriority	Definuje prioritu díky, které bude plánovačem přidělovat úkoly na zpracovaní procesoru. Hodnoty se udávají od 0 do ($configMAX_PRIORITES-1$), což je nejvyšší parametr.
pxCreatedTask	Handler, který bude při vytváření s úlohou manipulovat

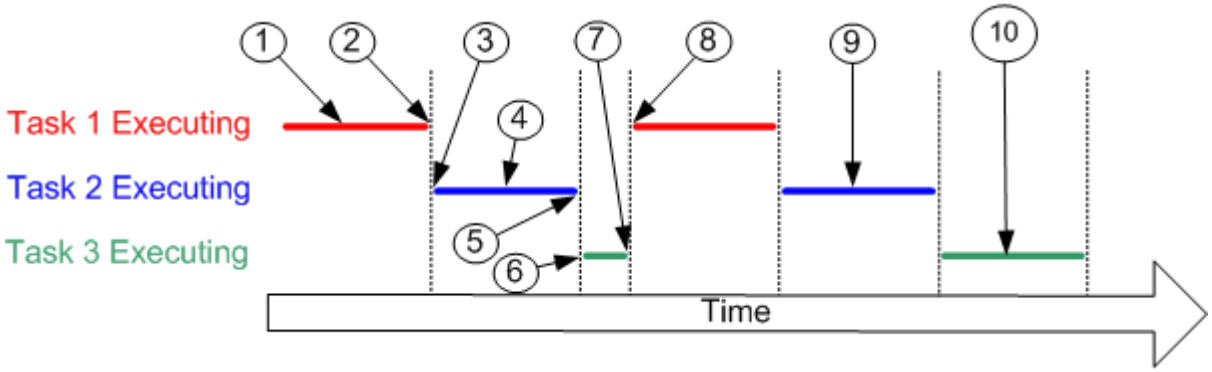
Tabulka 1.6: Parametry vytvoření úkolu[11].

Plánovač

Plánovač, jak už název napovídá, bude sloužit k tomu, aby se vykonávaly úkoly v takovém pořadí,v jakém bude zapotřebí. Hlavní úkol plánovače se dá analogicky představit jako studenta před zkouškovým obdobím. Student má před sebou hodně

úkolů, každý má jinou prioritu a jiný termín odevzdání. Aby student mohl projít do dalšího semestru bez potíží, musí si úkoly správně naplánovat, aby se všechny úkoly v termínu stihly. V jakém pořadí bude student úkoly řešit, už nezáleží, pokud neřeší úkol, který potřebuje data z úkolu jiného.

Pro správné plánování se používá algoritmus, který rozhoduje, jaký úkol se bude vykonávat v daném bodě v čase. Ve skutečnosti může vypadat takový plán tak, jak je vyobrazeno na obrázku 1.7.



Obrázek 1.7: Zobrazení vykování úkolů v čase (dostupné na www.freertos.com)

1. úloha 1 se vykonává
2. jádro pozastaví úlohu 1
3. obnoví se úloha 2
4. je vykonána úloha 2, dojde k uzamknutí procesorové periférie pro svůj exkluzivní přístup
5. jádro pozastaví úlohu 2
6. obnoví se úloha 3
7. úloha 3 se pokouší přistoupit do stejného procesorové periférie, zjistí že, je zablokovaný a nemůže dál pokračovat a je pozastaven sám sebou
8. jádro obnoví úlohu 1
9. úloha 2 je vykonávána, bude dokončena s procesovou periférií a odemkne sekci
10. úloha 3 je vykonávána, zjistí že má přístup k procesorové periférii a tentokrát dojde provedení, dokud nedojde k pozastavení

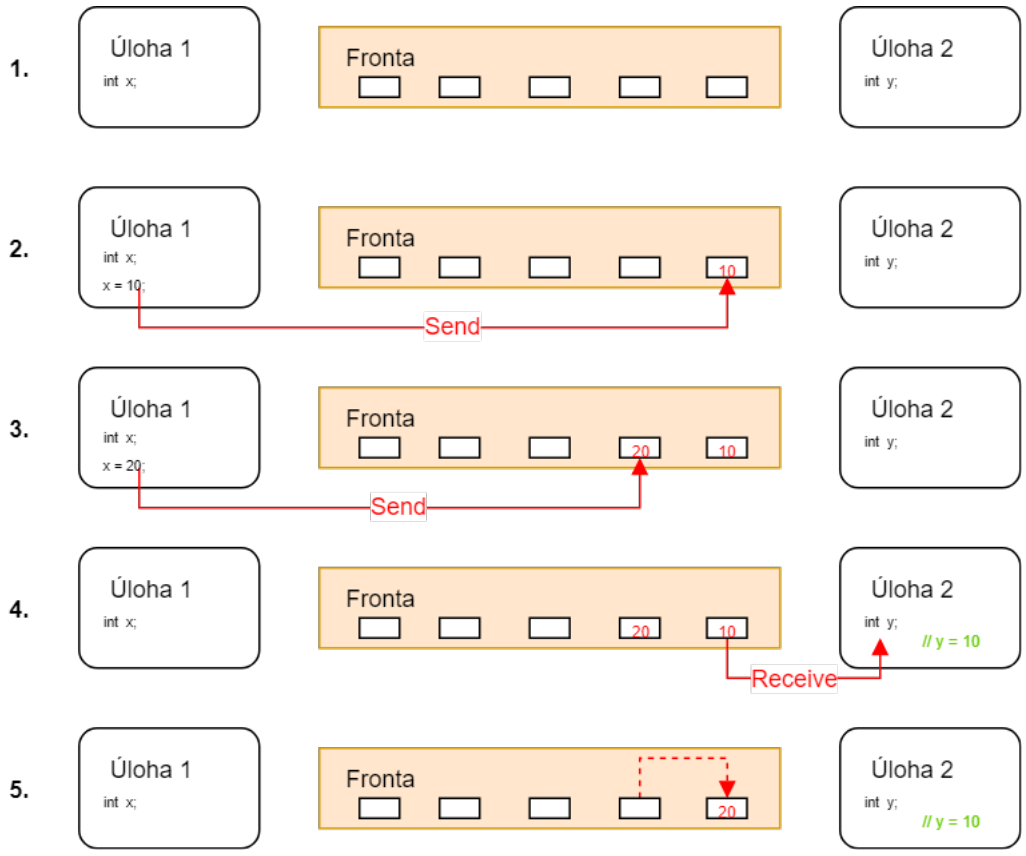
Fronta

Fronta se dá představit jako zásobník, kterým postupně protékají data k dalším úkolům, Používá se k předávání dat mezi úlohami a přerušením. Fronta může být vytvořena předtím nebo potom, co byl plánovač spuštěn. Pro lepší pochopení, jak fronta funguje, jsem uvedl následující příklad na obrázku 1.8.

```
QueueHandle_t xQueueCreate( UBaseType_t uxQueueLength, UBaseType_t
    uxItemSize );
```

uxQueueLength	Maximální počet bloků dat, které se do fronty můžou zapsat najednou
uxItemSize	Velikost jednoho bloku dat v bytech, který dokáže fronta udržet

Tabulka 1.7: Parametry pro vytvoření fronty[11]



Obrázek 1.8: Příklad funkce fronty[11]

1. Fronta je vytvořena tak, aby díky ni mohly komunikovat úlohy 1 a 2. Fronta dokáže najednou pojmout 5 hodnot.

2. Úloha 1 zapíše do fronty hodnotu 10

3. Úloha 1 změní hodnotu x na hodnotu 20, ta se zapíše do fronty za první hodnotu
4. Úloha 2 zažádá frontu o hodnotu. Napřed se z fronty odešle nejstarší hodnota
5. Hodnota 20 se ve frontě přesune na první místo

Z této ukázky je zřejmé, že fronta se chová jako paměť FIFO (First in First out).

Semafor

Semafor technicky dělá to, co bychom od semaforu čekali. Blokuje úkoly, které v tu danou chvíli nemají být na řadě. Semafor použitý na synchronizaci nevrací funkci *„odevzdat“* zpět, když je úspěšně přijata funkce *„vzít“*. Synchronizace je implementována jedním úkolem nebo přerušením, který semafor předá a další úkol si ho vezme. To znamená, že v jednu chvíli ho může mít pouze jeden úkol a ostatní čekají až na ně přijde řada.

Přepínač

Semafor a přepínač jsou velice podobné funkce, ale nejsou zcela totožné. Přepínač obsahuje prioritní systém. Priorita úlohy obsahujícího přepínač bude zvýšena, pokud se jiný úkol s vyšší prioritou pokusí získat stejný přepínač. Úloha, která již obsahuje přepínač, říká, že *”zdědí”* prioritu úlohy, která se pokouší *”vzít”* stejný přepínač. Zděděná priorita bude *”vyloučena”*, když bude vrácen přepínač (úloha, která zdědila vyšší prioritu, během toho co vlastní přepínač, se po návratu vrátí na původní prioritu)[11].

1.5 Cloudové služby AWS

Komplex cloudových služeb Amazonu by se dal definovat jako jeden veliký navzájem propojený ekosystém. To znamená, že jednotlivé segmenty spolu můžou interagovat, posílat si navzájem data nebo třeba notifikovat o stavu.

Jako příklad bych mohl uvést jednoduchou aplikaci. Mám pomocí javascriptového SDK připojený raspberry py, které posílá hodnotu teploty, vlhkosti a tlaku na IoT služby zabezpečeně do cloudu. Po přijetí dat se spustí Lambda funkce, která provede výpočet rosného bodu. Hodnota rosného bodu se zapíše spolu se základními hodnotami do NoSQL databáze DynamoDB.

Součástí aplikace může být i vizualizace dat. Proto na webu můžu použít PHP SDK, které se do databáze napojí a vyčte potřebná data pro vizualizaci v grafu pro uživatele. Pokud dojde k neobvyklé situaci, tak může Lambda funkce zavolat službu SNS, která pošle notifikaci ve formě Emailu, SMS nebo upozornění v mobilu.

Dále jsou služby propojené s Alexou, inteligentní asistentkou, se kterou můžeme služby taktéž propojit. Tohle je jenom zlomek služeb, které celkový cloud Amazonu disponuje.

Vzhledem k velikosti služby je zapotřebí hodně sledovat aktuality a změny, které se dějí prakticky denně. Momentálně na AWS můžete najít tyto základní služby:

- Výpočetní výkon
- Úložiště
- **Databáze**
- Migrace
- Síťové služby
- Vývojářské nástroje
- Managerské nástroje
- Mediální služby
- Zabezpečení
- Analytické nástroje
- Strojové učení
- Mobilní služby
- Augmentovaná a virtuální realita
- Zákaznické služby
- Desktopové a aplikační stream
- **Internet věcí**
- Herní vývoj

Zvýrazněné služby jsou ty, se kterými budu pracovat. Pro základní manipulaci s cloudem je zapotřebí účet. Při zakládání účtu je vyžadováno číslo kreditní karty, ale Amazon nabízí rok využívání služby naprosto bezplatně, pro snadnější vývoj aplikace bez ztrát způsobených chybami.

Po vytvoření účtu se lze přihlásit do tzv. „AWS konzole“, kde je přístup k celé paletě služeb. Doporučuji pro lepší manipulaci nainstalovat konzoly do počítače. Po instalaci bude konzole dostupná z příkazového řádku použitého operační systému. Důvodem, proč tuto instalaci doporučuji, je rychlejší úprava jednotlivých služeb, rychlé generování certifikátů a uživatelů nebo třeba odesílání testovacích dat.

1.5.1 Databáze DynamoDB

Jedná se o plně spravovanou databázovou službu NoSQL, která poskytuje rychlý a předvídatelný výkon s bezproblémovou škálovatelností. Touto službou na sebe Amazon přebírá administrativní zátěž z provozu, poskytuje hardware, nastavuje konfigurace, replikace, záplatování softwaru a škálování clusterů. Také nabízí šifrování, čímž eliminuje provozní zátěž a složitost při ochraně citlivých dat.

DynamoDB automaticky rozšiřuje provoz tabulek na dostatečný počet serverů, aby zvládly veškeré požadavky zákazníka na propustnost a ukládání dat. Všechna data jsou uložena na pevných discích (SSD).

V následující tabulce porovnávám základní terminologii SQL a NoSQL databáze.

1.5.2 Internet věcí

Služba Internet věcí od Amazonu poskytuje zabezpečenou obousměrnou komunikaci mezi zařízením připojeným k internetu, jako jsou senzory, ovládače, vestavěné

SQL	DynamoDB (NoSQL)
Tabulka	Tabulka
Řádek	Položka
Sloupec	Atribut
Primární klíč	Primární klíč
Pohled	Globální sekundární klíč
Vnořená tabulka nebo objekt	Mapa
Pole	List

Tabulka 1.8: Porovnání SQL a NoSQL terminologie

mikrořadiče či jiná chytrá zařízení a cloudu AWS. To umožňuje shromažďovat data z více zařízení, ukládat je a analyzovat[15].

Pro bezproblémový připojení zařízení je zapotřebí několika stěžejních systémových částí.

Brána zařízení (device gateway) Umožňuje zařízením se zabezpečeně a efektivně komunikovat s cloudem.

Zprostředkovatel zpráv (message broker) Jedná se o mechanizmus, kterým komunikace probíhá. Tato část se stará o odesílání a příjímání zpráv mezi všemi zařízeními. Ke komunikaci je použit MQTT protokol. Je také možné využít přímý protokol MQTT přes WebSocket.

Rozhodovací systém (Rules engine) Integruje AWS IoT s dalšíma službami od Amazonu, lze takto vybraná data ve formátu JSON dál delegovat do služeb typu Amazon DynamoDB, Amazon S3 nebo AWS Lambda. Lze také přijatou zprávu z jednoho zařízení přeposlat dalším odběratelům.

Zabezpečení a práva identit Poskytuje sdílenou zodpovědnost za bezpečnost na cloudu. Zařízení musí držet svoje přihlašovací certifikáty v bezpečí. Tato část systému se stará o generování certifikátů a jejich správu.

Registry Organizují zdroje propojující zařízení v rámci AWS cloudu.

Skupinové registry Skupiny umožňují spravovat několik zařízení najednou. Lze takto vybudovat hierarchie různých skupin a podskupin, protože i skupina se může skládat ze skupin.

Stín zařízení (device shadow) Jedná se o JSON dokument, do kterého se ukládá aktuální stav zařízení jako informace o stavu.

1.5.3 Lambda funkce

Výpočetní služba Lambda funkce umožňuje spouštět kód bez poskytnutí nebo správy serverů. Dokáže vykonat uživatelský kód jenom tehdy, je-li to zapotřebí a automaticky se dokáže spouštět. Platí se pouze za výpočetní čas, když je výpočet rychlý, tak tato služba nebude stát mnoho prostředků[16].

Lambda funkce lze psát v jazycích *Node.js, Java, C#, Go a Python*. Lze také přistupovat ke službám v rámci ekosystému. Například je možné vyčítat nebo zapisovat do databáze DynamoDB, lze reagovat na HTTP požadavky použitím Amazon API Gateway.

1.6 Místnost A10

Počítačová učebna A10 se nachází ve druhém patře budovy **A** Technické Univerzity v Liberci *(ozn. A03023)*. Místnost neobsahuje žádnou vzduchotechniku, větrání je řešeno pomocí otevřených oken.

Místo zvolené pro umístění měřící jednotky je na půdorysu vidět dole uprostřed. Měl by být ve výšce 1,5-3 m a naproti oknům.

Rozměry místnosti:

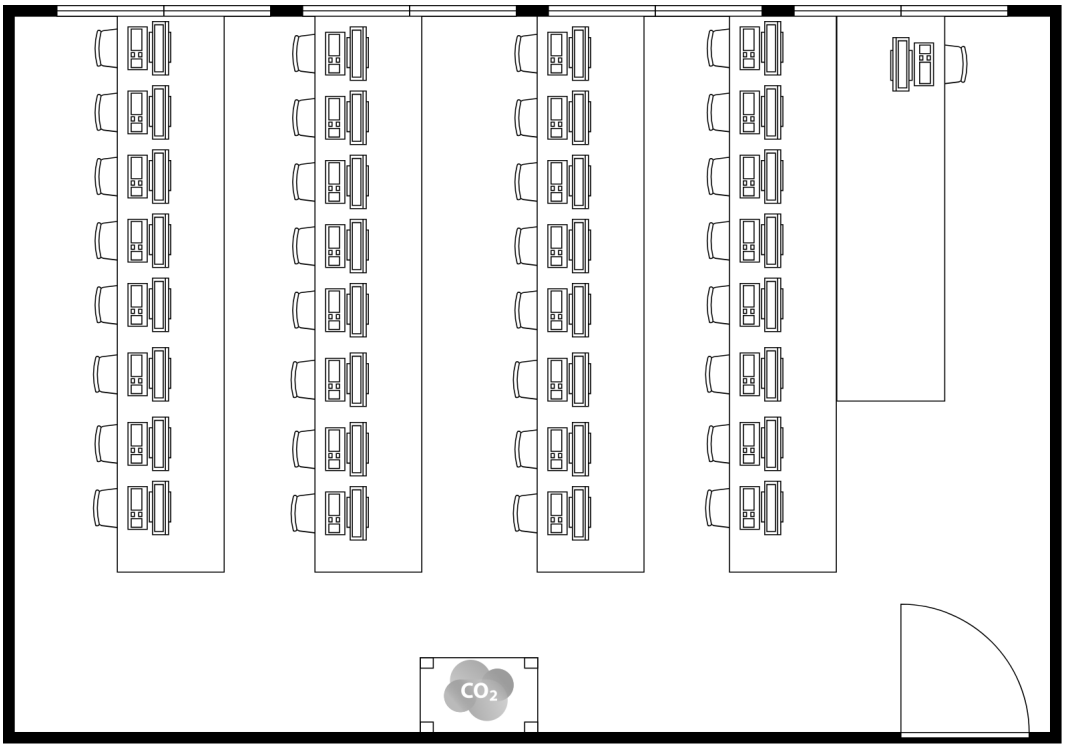
- *Šířka: 9 m*
- *Délka: 13 m*
- *Výška: 4,5 m*
- *Objem: 526,5 m^3*

Rozměry oken:

- *Šířka: 2 m*
- *Výška: 3 m*
- *Obsah: 6 m*
- *Počet: 4x*

Rozměry dveří:

- *Šířka: 1,5 m*
- *Výška: 2,10 m*
- *Obsah: 3,15 m*
- *Počet: 1x*



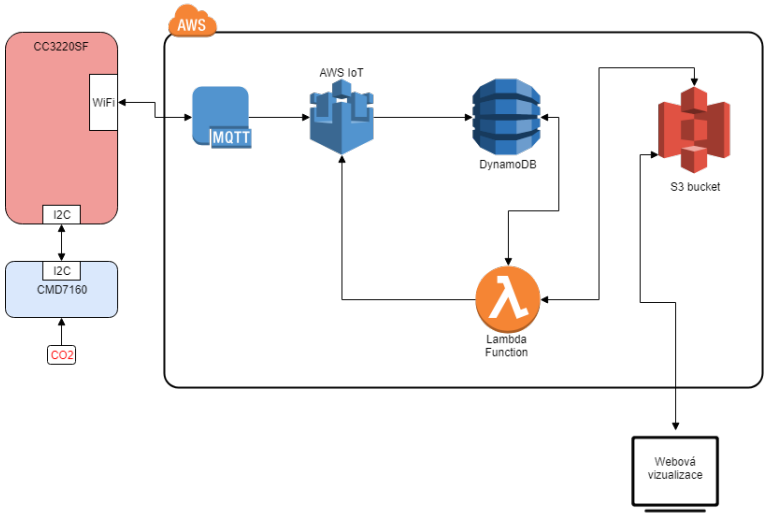
Obrázek 1.9: Půdorys učebny A10

2 Návrh

V této kapitole se budu zabývat postupným návrhem celé aplikace. Jako prvotní krok byl seznámit se s vybraným hardwarem. Vyhledal jsem si dostupnou literaturu k tomu, abych pochopil, jak funguje FreeRTOS, jak přistupovat k jednotlivým periferiím základní desky, jak komunikuje čidlo na měření hodnoty CO_2 a v neposlední řadě napojení celého ekosystému na cloudové služby od Amazonu.

Jako základ celé aplikace je tvořen upravenou verzí FreeRTOS, kterou lze stáhnout se stránek Amazonu. Pro úpravu kódu jsem použil dostupné IDE od firmy Texas Instrument Code Composer Studio (CCS). V následující kapitole jsem popsal, jak CCS nastavit pro danou desku tak, aby vše fungovalo a napojení SDK s Amazon FreeRTOS.

Toto blokové schéma vyobrazuje základní zapojení všech komponent a využitých cloudových služeb.



Obrázek 2.1: Blokové schéma projektu

2.1 Základní konfigurace SW/HW

Pro vývoj této aplikace jsem využíval aktuální verzi CCS 8.0.0 na platformě Windows 10, existuje i verze pro Linux a MAC. Toto vývojové prostředí si můžete zdarma stáhnout na stránkách výrobce Texas Instrument (TI).

1. Instalace Code Composer Studia 8.0.0.
2. Během instalace jsem vybral doinstalování knihoven k vývojové desce **CC3220SF** a ovladače **TI XDS110 USB** pro ladění na desce.
3. Po instalaci jsem připojil desku na správný COM port.
4. Instalace terminálu, např.: **TerraTerm** nebo **PuTTY** (verze **CCS 8.0.0** obsahuje vestavěný terminál, občas s ním byly problémy, proto se i výrobce odkazuje radši na terminál třetích stran).
5. Instalace **CCS UniFlash v.4.2.2.1692** nebo vyšší.

Pro vývoj aplikace jsem používal **SimpleLink CC3220 SDK v.2.10.0.04** a **XDCtools v.3.50.5.12_core**.

2.1.1 Nastavení terminálu

Protože hodně aplikací používá UART komunikaci pro zobrazování základních stavových informací během ladění, tak je potřeba mít terminál správně konfigurován. V následující tabulce 2.1 je vidět potřebné nastavení.

Parametr	Hodnota
Port	COM11
Řádek	Položka
Baud rate	115 200
Data	8 bit
Parita	none
Stop	1 bit
Flow control	none

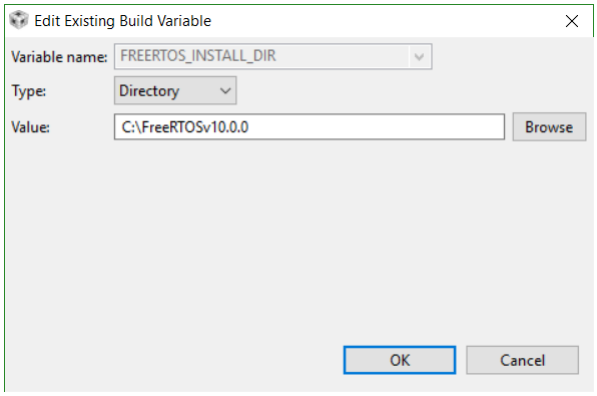
Tabulka 2.1: Nastavení terminálu

2.1.2 Nastavení podpory FreeRTOSv10 v CCS

Pro vývoj aplikace pomocí operačního systému FreeRTOS je zapotřebí mít v CCS zajištěnou podporu.

1. Stáhnu FreeRTOS verze 10 z oficiálního webu freertors.com
2. Nainstaluji FreeRTOS na C: disk
3. Zkopíruji obsah složky *<cesta k instalaci SDK>/tools/cc32xx_tools /FreeRTOS_patch/CCS* a vložím je do *<cesta k instalaci FreeRTOS>/FreeRTOS/Source/portable/CCS*
4. Spustím CCS

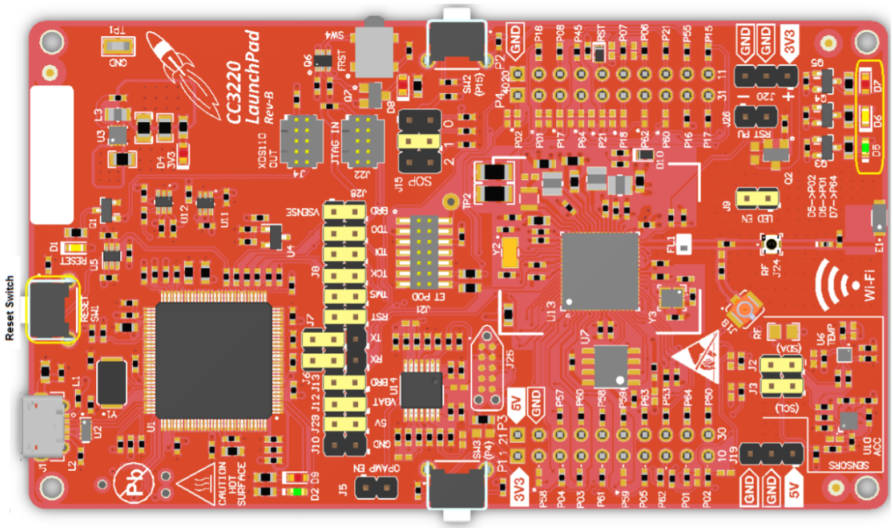
5. Vyberu ***Windows*** → ***Preferences*** → ***Code Composer Studio*** → ***Build*** → ***Variables*** → ***Add***
6. Pole proměnné (*Variable*) vyplním **FREERTOS_INSTALL_DIR**
7. Změním *Type* na **directory**
8. Vyplním pole *Value* cestou k instalaci FreeRTOS: ***C://FreeRTOSv10.0.0***



Obrázek 2.2: Nastavení podpory FreeRTOS v CCS

2.1.3 HW konfigurace desky

Na obrázku 2.3 je vidět, jak mají být zapojeny jumpery na základní desce CC3220SF. Jumpery jsou zvýrazněny žlutou barvou. Vlevo je vidět tlačítko reset a zprava jsou oznamovací LED diody.



Obrázek 2.3: CC3220SF základní rozmístění jumperů

2.2 Amazon cloud services

V této kapitole jsem rozebral, co je všechno potřeba k práci v cloudové konzoly od Amazonu. Ukázal jsem, co je nutné k navázání komunikace mezi základní deskou a cloudem. Sepsal jsem jednoduchý postup, jak založit IoT zařízení, databázi a Lambda funkci.

2.2.1 Identity and Access Managment (IAM)

Pro základní přístup ke službám Amazonu je nutné založit účet, který bude mít přístup k potřebným částem AWS konzole. Tento uživatel je určen k přímému přístupu do administrace přes webové prostředí nebo z konzole osobního počítače.

Velice důležité bylo vytvoření dvou pravidel, které využívají služby pro komunikaci mezi sebou. Je to logické, aby se služby navzájem neovlivňovaly, proto je nutné mít služby oddělené. Tato pravidla je navzájem propojují a určité, k jakým částem služeb mají oprávnění.

Pro vytvoření pravidel najdu v kontextovém menu položku ***Roles***. Zvolím *create rules* → *AWS services* → *DynamoDB - Global Tables* → *Next permission* → *Next: Review* → *create role*. Stejný postup provedu i u vytvoření pravidla pro Lambda funkci. Jenom místo *DynamoDB* zvolím *Lambda*.

2.2.2 IoT zařízení

Sekce IoT core slouží jako základní rozhraní pro správu všech IoT zařízení, pravidel a MQTT brokeru. První, bylo třeba stáhnout Amazon FreeRTOS. Ten se nachází vlevo dole v položce *Software*. Zobrazila se nabídka několika možností, důležitá je hned ta první (*Amazon FreeRTOS Device Software*). Je důležité podotknout, že se velice často celé prostředí mění jak vizuálně, tak obsahem. Zvolil jsem již předdefinovanou položku pro desku CC3220SF.

Obrázek 2.4: Výběr položky Amazon FreeRTOS

Třemi tečkami, které jsou na pravé straně, se dá dostat ještě do kontextové nabídky, kde jde modulárně vybrat, jaké části SDK má balíček Amazon FreeRTOS obsahovat. Pro moji aplikaci nebyly ostatní části SDK potřeba.

Dále je potřeba připravit a nakonfigurovat zařízení CC3220SF v AWS IoT core. Nové zařízení se vytváří v položce *Manage* → *Things*. Zvolím *Create* → *Create a single thing*. Zvolím název zařízení, typ zařízení, to jenom v případě, mám-li více

stejných zařízení. Lze vytvořit i skupinu zařízení. Dále jsou velice důležité certifikáty. Veškerá komunikace mezi IoT zařízeními a cloudem probíhá zašifrovaně. Zvolil jsem první možnost *One-click certificate creation*, kterou AWS doporučuje. Vytvoří se 3 certifikáty, všechny jsem si je stáhl pro další použité, které budu popisovat v další kapitole.

2.2.3 Založení databáze DynamoDB

Pro vytvoření NoSQL databáze se formulář dotazuje na dva základní parametry. Název tabulky je naprosto individuální. Druhá položka je primární klíč. To je základní parametr, pomocí kterého se dají v databázi třídit data. V mém případě jsem tento parametr nazval **device** a datový typ **String**. Vzhledem k tomu, že naměřená data jsou časově závislá na čase, tak jsem k tabulce přidal ještě sekundární klíč. Ten se přidá zaškrtnutím políčka „*Add sort key*“. Název jsem zvolil **timestamp** a jako datový typ jsem určil **Number**.

2.2.4 Propojení IoT zařízení a databáze

Vzhledem v velice rozsáhlé integraci služeb Amazonu není toto nastavení nic těžkého. V sekci IoT core zvolím v kontextovém menu položku **Act**. Zde se dají vytvářet různá pravidla, která se budou vykonávat v podmínkách určených uživatelem. Dále je potřeba určit, pro jaká data bude funkce aktivní, na kterém topiku bude pravidlo naslouchat a dokonce jde i určit podmínku, pro jaké hodnoty bude pravidlo funkční.

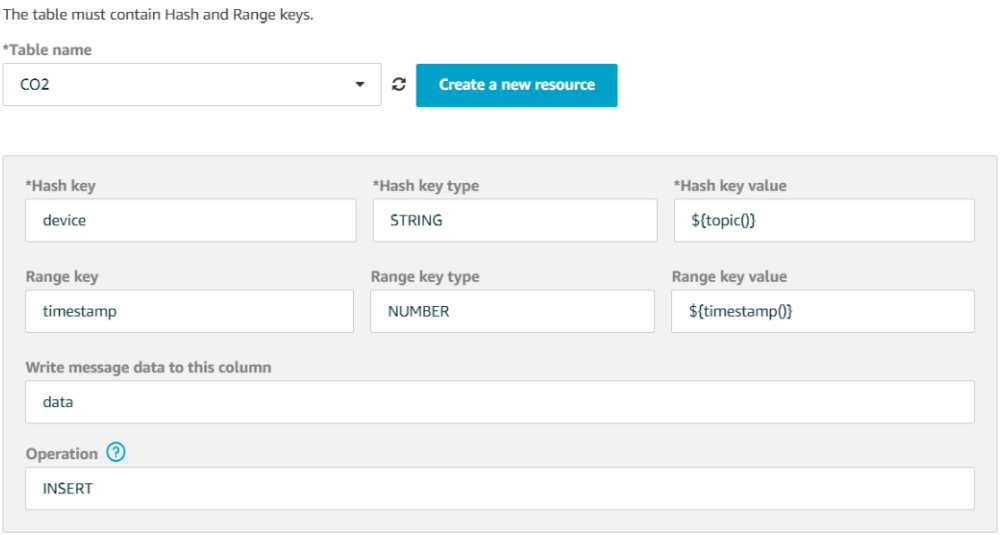
Základním nastavením Amazon FreeRTOS je zvolen topik ***freertos/demos/echo***. Jako atribut stačí zvolit *******. Výsledný SQL dotaz vypadá následovně:

```
SELECT * FROM 'freertos/demos/echo'
```

Poslední důležité nastavení je, co samotná funkce bude dělat, když se na topiku objeví zpráva ze zařízení. Zvolím *Add action* a vyberu ***Insert a message into a DynamoDB table***. Nabídka funkcí je velice pestrá, dá se zde nastavit například posílání notifikací pomocí služby SNS, zápis do databází Amazon S3 bucket nebo Elasticsearch. Zvolená funkce se musí potvrdit tlačítkem *Configure action*.

Nyní je nutné vyplnit formulář a zvolit tabulku, do které se budou data zapisovat. Potom vyplnit primární a sekundární klíče pro správné zařazení dat. Dále se zvolí název, pod kterým budou data uložena v databázi a také název operace, která funkci vykoná. V mém případě bude data jenom vkládat.

Na konci formuláře je ještě zvolení práv k přistupováním k různým službám. Vytvoření pravidel je vysvětleno v kapitole 2.2.1 Zvolím vytvořené pravidlo, které má práva přistupovat k databázi.



Obrázek 2.5: Nastavení pravidla pro zápis do DynamoDB

2.2.5 Vytvoření Lambda funkce

Lambda funkce je výpočetní služba, která dokáže spouštět kód aplikací bez poskytování nebo správy serverů. Výhodou je absence administrace na straně serveru, jednoduše se napíše kód a výpočetní funkce mohou bez problému běžet opakovaně celé dny tisíckrát za sekundu. Momentálně jsou podporované tyto programovací jazyky: ***Node.js, Java, C#, Go a Python***.

Funkce má možnosti přístupu do AWS služeb a může tak upravovat data v databázových systémech Amazon S3 bucket nebo Amazon DynamoDB. Je také možné zpřístupnit aplikaci promocí HTTP žádostí za pomoci služby Amazon API Gateway.

Vytvoření aplikace je velice jednoduché. Vzhledem k tomu, že veškerá konfigurace je již přednastavená, tak na vývojáře zbývá pouze vybrat si programovací jazyk a napsat aplikaci.

Já jsem pro čtení z databáze a výpočty na straně serveru zvolil jazyk **Node.js**. Momentálně je možné pracovat na *Runtime* verze Node.js 8.10, která je v době psaní práce aktuální a disponuje dlouhodobou podporou verzí s dlouhodobou podporou (LTS).

Při vytvoření lambda funkce lze začít aplikaci několika způsoby. První je vytvoření aplikace úplně od nuly. Druhý způsob je použití předpřipravených plánů, jedná se o různé napojení na ekosystém Amazonu. Poslední možností je použití aplikačních repozitářů třetích stran.

Já jsem zvolil přístup vytvořit si vlastní aplikaci. Pojmenoval jsem funkci a vybral aktuální runtime. Pro přístup k databázi je opět zapotřebí použití pravidla, využil jsem již dříve vytvořené pravidlo pro přístup do databáze popsané v kapitole 2.2.1. Zvolil jsem pravidlo a dal vytvořit funkci. Po vytvoření funkce se objeví návrhář, kde se jednoduše nastaví, k jakým službám má funkce přistupovat. Dále je vymezen prostor pro webový IDE *Cloud9*, ve kterém jsem aplikaci naprogramoval.

3 Postup a realizace

Předchozí kapitola přiblížila, jak správně postupovat při konfiguraci veškerých komponent použitých v této bakalářské práci. V následující kapitole popisuji, jak jsem jednotlivé části programoval a uvedl do provozu. Kapitola je členěná chronologicky podle toho, jak jsem v práci skutečně postupoval.

3.1 Programování desky CC3220SF

Pro realizaci bakalářské práce jsem napřed musel pochopit, jak freeRTOS funguje. Proto jsem začal naprogramováním komunikace mezi základní deskou *CC3220SF* a měřícím čidlem *CMD7160*. K tomu jsem využil dostupné knihovny od TI z balíčku *driverlib* (tento balíček je součástí SDK k základní desce).

Dalším krokem bylo navázání komunikace se službami od Amazonu. K tomu je potřeba využít již stažený SDK od Amazonu. Postup ke stažení SDK popisuji v kapitole 2.2.2 na straně 37.

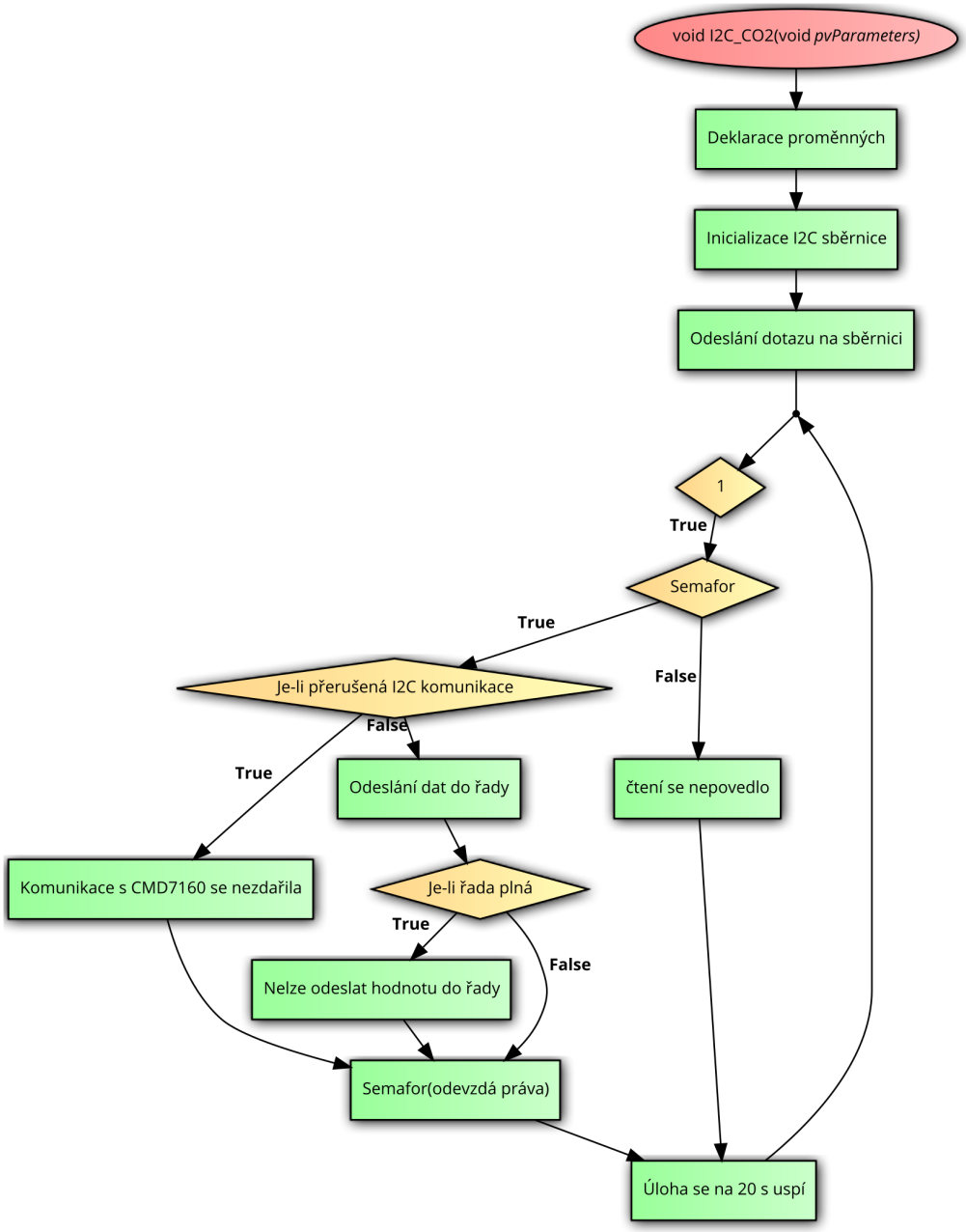
Balík je potřeba prvotně nakonfigurovat. Nastavení se skládá ze tří částí:

1. WiFi připojení
2. Import AWS certifikátů pro IoT zařízení
3. Určení topiku pro MQTT broker

Poslední krokem byla implementace předchozího kódu komunikace s měřícím čidlem a data přeposlat MQTT zprávou do cloudu.

3.2 Komunikace mezi CC3220SF a CMD7160

Vzhledem k absenci konkrétní knihovny pro čidlo CMD7160 jsem musel celou komunikaci vytvořit na základě prostudování dokumentace a knihovny pro I^2C z SDK. Základní postup komunikace přes sběrnici I^2C je vyobrazen na obrázku 1.6 na straně 23. Celá funkce, která zprostředkovává komunikaci, je tvořená jako úkol pro freeRTOS. Blokové schéma 3.1 zobrazuje, jak principiálně funguje celá úloha. Kódová forma je k dispozici v příloze na CD v souboru *aws_hello_wolrd.c*.



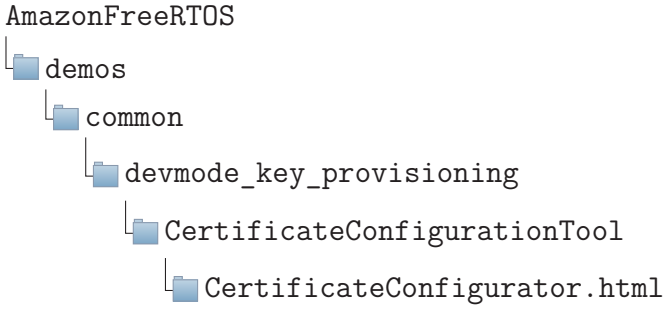
Obrázek 3.1: Algoritmus čtení dat

3.3 Konfigurace AWS SDK

Základní nastavení pro komunikaci s cloudem se nachází v souboru *aws_clientcredential.h*. Je potřeba změnit nastavení v těchto parametrech:

```
//Nastavení endpointu
static const char clientcredentialMQTT_BROKER_ENDPOINT[] = "
    **************.iot.eu-central-1.amazonaws.com";
//Nastaveni nazvu zarizeni
#define clientcredentialIOT_THING_NAME "CC3220SF"
//Port brokeru
#define clientcredentialMQTT_BROKER_PORT 8883
//Udaje o WiFi síti
#define clientcredentialWIFI_SSID       "mereniCO2"
#define clientcredentialWIFI_PASSWORD   "xxxx"
//Typ čzabezpeení
#define clientcredentialWIFI_SECURITY   eWiFiSecurityWPA2
```

Poslední část konfigurace AWS SDK je vložení klíčů pro IoT zařízení. Klíče se ukládají v textové podobě do souboru *aws_clientcredential_keys.h*. Klíče je potřeba vložit ve speciálním formátu. Amazon FreeRTOS SKD obsahuje nástroj, který klíče naformátuje. Cesta k nástroji je:



Do nástroje se vloží ob potřebné klíče a ono to automaticky vygeneruje soubor *aws_clientcredential_keys.h*.

Pokud je vše správně nastaveno, tak v tento moment by měla být deska CC3220SF schopná navázat spojení s cloudem od Amazonu.

3.4 Odeslání hodnoty CO_2 na cloud

Nyní už nic nebrání tomu odeslat hodnotu na MQTT broker a uložení dat do databáze. Celý hlavní kód se nachází v souboru *aws_hello_world.c*. Nebudu zde uvádět všechny funkce, které jsou ke komunikaci s MQTT brokerem nutné. Toto demo od Amazonu jsem obohatil o předchozí úlohu na čtení dat z CMD7160 přes sběrnici I^2C. Demo se skládá z dvou základních úloh. První vytvoří MQTT klienta *prvMQTTConnectAndPublishTask()* a druhá odesílá periodicky zprávu *prvMessageEchoingTask()*. Pro odesílání dat z fronty, do které ukládá hodnotu o změřené koncentraci CO_2 úloha *I2C_CO2()*, se stará úloha *prvPublishNextMessage()*. Tu jsem adaptoval následujícím způsobem.

```
short rx_short = 0;
    if (xQueueReceive(Global_Queue_Handle, &rx_short, 1000))
    {
        (void) snprintf(cDataBuffer, echoMAX_DATA_LENGTH, "{\"CO2\":
            \"%d\"}",
                        (short) rx_short);
        UART_PRINT("Received CO2 %i \n\r", rx_short);
    }
```

V základním nastavení se úloha *prvMQTTConnectAndPublishTask()* volá každých 5 sekund. Pro měření je toto moc vysoká frekvence, proto jsem nastavil hodnotu na dvacet sekund a to následujícím způsobem:

```
//deklarace hodnoty podle frekvence procesoru presne na 20 sekund
const TickType_t xTwentySeconds = pdMS_TO_TICKS(20000UL);

//cyklus se na dvacet sekund zastavi, žne se zavila úloha
   prvPublishNextMessage(x);
if (xReturned == pdPASS)
    {
        for (;;)
        {
            prvPublishNextMessage(x);
            //zde se nastavuje perioda opakovani
            vTaskDelay(xTwentySeconds);
        }
    }
```

3.5 Naprogramování Lambda funkce

Vyčítání dat z databáze pomocí Lambda funkce je realizováno pomocí jednoduché *javascriptové* funkce. Ta obsahuje *JSON* s parametry, z jaké tabulky má číst, jaká data a podmínku mezi kterými daty. Já určím dvě časové konstanty v *Unix formátu* se 13 znaky.

```
 var params = {
        TableName:"CO2",
        KeyConditionExpression:"#device = :deviceValue and #
            timestamp BETWEEN :from AND :to",
        ExpressionAttributeNames: {
            "#device":"device",
            "#timestamp":"timestamp"
        },
        ExpressionAttributeValues: {
            ":deviceValue":device,
            ":from": from,
            ":to":to
        }
    };
```

Zbytek kódu pro lambda funkci je obsaženo v přiloženém CD.

3.6 Zapojení CC3220SF a CDM7160

Napájení desky je realizováno µUSB konektorem za kterým je usměrňovač napětí na 5V a 3V. Pro zapojení čidla CMD7160 je potřeba mít přístup k sběrnici I^2C. Ta se nachází na portech *P*01 *(SLC)* a *P*02 *(SDA)*. Sběrnice potřebuje ke svému chodu zvedací odpory mezi vodiči a kladnou svorkou napájení. Podle hodnoty odporu lze určit rychlost komunikace, odpory můžou mít velikost 2kΩ (400 kbps) nebo 10kΩ (100 kbps). Správné zapojení ukazuje obrázek v příloze B.1.

3.7 Zobrazovací výstup

Pro vizualizaci naměřených dat a výpočtu dynamického algoritmu jsem použil webové prostředí. Jako úložiště jsem využil služby *Amazon S3 bucket*, kde jsem vytvořil hosting a uložil soubory k webu. Web je dostupný na adrese:
https://s3.eu-central-1.amazonaws.com/occupancycounter/index.html?
Webové rozhraní jsem zabezpečil přístupem pouze z IP adresy školní sítě.

Pro realizaci jsem použil šablonu *Angular Boostrap*. O zobrazení dat se stará knihovna *Chart.js*, který vytváří graf. O komunikaci s AWS službami jsem použil SDK pro *javascript*.

Obrázek 3.2: Ukázka vizualizace webu

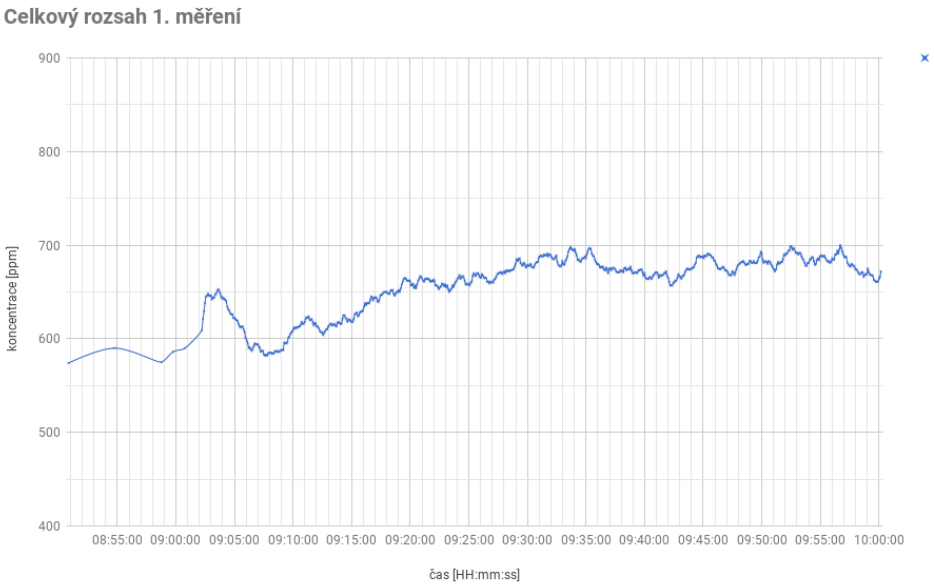
Do pole výběr času se zadává hodnota ve formátu *Unix*, který má 13 znaků. V tomto formátu jsou ukládané i data v DynamoDB. Výběr hodnot z databáze se stejný jako je popsán v kapitole o Lambda funkci viz. kapitola 3.5. Naměřená data jsem doplnil o výpočet klouzavého průměru, ten pouze vyhladí průběh o krátkodobé výkyvy, což vizualizuje modrá křivka.

4 Experimentální měření

Kapitola popisuje, jak jsem postupoval při ověřování funkce výpočtu pomocí dynamického algoritmu. Měření celkově proběhla čtyři. První bylo čistě jenom pro nastavení parametrů místnosti a ověření, že aplikace funguje jak má. Druhé měření proběhlo během normální výuky. Testoval jsem, jak doba je potřeba na ustálení dat i jak reaguje algoritmus na poruchu typu otevřeného okna. Třetí a čtvrté měření proběhla hned za sebou, jediná změna byla, že u třetího měření bylo neustále otevřené okno a u druhého byl větší počet lidí v místnosti. Všechny grafy jsou z důvodu přehlednosti v příloze bakalářské práce. Trend je vypočítaný na základě změřených dat v aplikaci *Google sheet*.

4.1 Měření č. 1

První testovací měření proběhlo 24.4.2018 mezi 9:00 - 10:00 hodinou. Cílem prvního měření byla kalibrace parametrů do algoritmu. Hodnota CO_2 byla snímána každých 5s. Celý průběh prvního měření je vyobrazeno na grafu 4.1. Během měření byla přítomna pouze jedna osoba.

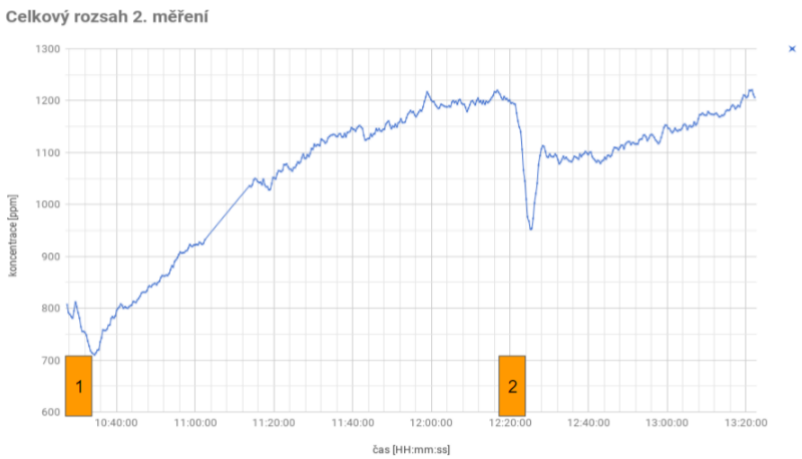


Obrázek 4.1: Celý rozsah 1. měření

Na grafu je vidět prvotní nárůstu koncentrace CO_2. To hlavně proto, že jsem byl blízko měřící stanici. Pak je vidět, že hodnota CO_2 klesá. To bylo způsobeno mým odchodem od čidla. Dále hodnota postupně rostla. Během jedné hodiny měření jsem dokázal hodnotu změnit jenom necelých 100 ppm, což je natolik zanedbatelná hodnota, že výsledek výpočtu z těchto naměřených dat nebudu uvádět. V 9:48 jsem otevřel okno, tato porucha se projevila na měření mírným poklesem koncentrace v 9:56. Výsledkem tohoto měření bylo zjištění, že pro otestování algoritmu je zapotřebí přítomnosti většího množství osob.

4.2 Měření č. 2

K druhému měření došlo dne 9.5.2018 v době od 10:30 do 13:20. Hodnota CO2 byla snímána s periodou 20 sekund, perioda byla změněna na základě prvního experimentálního měření. Ukázalo se že měření s periodou 5 sekund byla zbytečně podrobné. Docházelo k tomu, že se opakovali stejně nebo podobné hodnoty s rozdílem jednotek. Vzhledem k parametrům použitého čidla s chybou měření ±50 ppm nemá smysl koncentraci oxidu uhličitého měřit tak podrobně.



Obrázek 4.2: Celý rozsah 2. měření, *1* označuje přítomnost osoby v bezprostřední blízkosti čidla, *1* označuje dobu otevření okna

Na Grafu 4.2 je vidět průběh celého měření. Rozsah časové osy jsem nastavil na 20 minutové intervaly, ve kterých jednotlivě určuji počet lidí v místnosti pomocí dynamického algoritmu. První část měření je zkreslená, protože jsem byl přítomen blízko měřícího stanoviště, toto místo je označeno *1*. Proto je vidět potom pokles, protože se musela hodnota CO2 ustálit. Druhé označené místo *2* ukazuje otevření okna na dvě minuty. První naměřená data, která jsem zpracoval byla od 11:00 do 11:20. Průběh je vidět na grafu C.1. Přítomno bylo 7 lidí. Algoritmus vypočítal, že je přítomno 9 lidí. Během tohoto měření došlo k 12 minutovému výpadku.

Druhý úsek měření viz. C.2 ukazuje první ustálená data, protože dochází k trvalému nárůstu hodnoty koncentrace oxidu uhličitého. Během měření bylo přítomno 7

lidí a algoritmus vypočítal hodnotu zaokrouhlení 7 osob. Na grafu je názorně vidět, že občas došlo k drobné poruše měření způsobenou odchodem studenta na toaletu. Mírný pokles je vidět okolo 11:24 nebo 11:32.

Během třetího úseku viz. graf C.3 v průběhu měření přibyl v místnosti další člověk, čímž stoupl počet lidí na 8. Algoritmus spočítal na základě dostupných dat 8 osob. Dokázal tuto změnu zaznamenat.

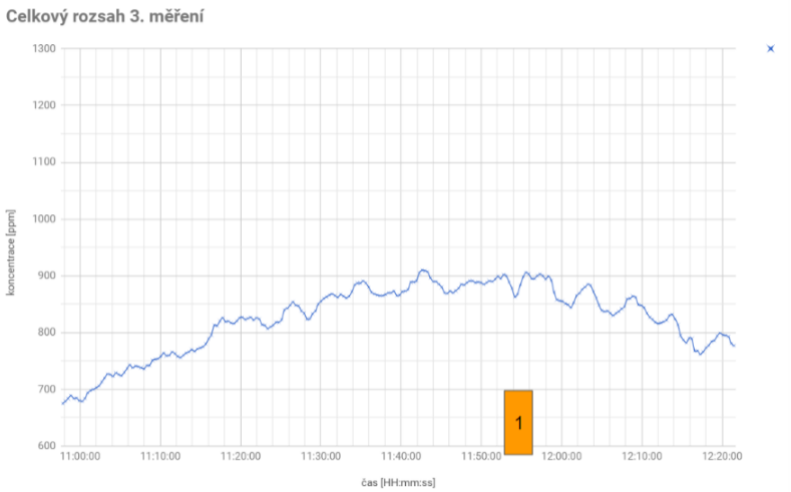
Na grafu C.5 je porovnání druhého a třetího úseku měření. Lze vidět, že trend byl prakticky totožný, i když bylo v místnosti přítomno 7 nebo 8 lidí.

Na grafu C.3 je vyobrazena porucha měření ve formě otevření okna po ukončení výuky. Okno bylo otevřeno ve 12:22 a zavřeno 12:24. Pouhé dvě minuty jednoho úplně otevřeného okna stačilo k tomu, aby hodnota koncentrace oxidu uhličitého klesla o 200 ppm.

Po poruše měření v podobě otevřeného okna už nedošlo k ustálení hodnoty CO_2. Byly přítomni pouze tři idé a měření bylo narušeno častými poruchami otevřených dveří nebo okna. Pro výpočet v těchto podmínkách by bylo potřeba poruchy přesně měřit a implementovat do algoritmu.

4.3 Měření č. 3

Měření č. 3 proběhlo dne 10.5.2018 v rozmezí 11:00-12:20. Měření probíhalo obdobným způsobem jako měření č. 2. Perioda čtení dat byla 20 sekund. Měření se zúčastnilo 8 osob. Během celého měření bylo otevřeno jedno okno u katedry. Celý rozsah měření je vidět na grafu 4.3. V grafu je označené místě *1*, které naznačuje otevření okna a dveří. Hodnota se díky otevřenému jedinému oknu nepřiblížila ani 1000 ppm. Měření je rozděleno na čtyři části po 20 minutách. V první části měře-



Obrázek 4.3: Celý rozsah 3. měření, *1* označuje odchod všech studentů

ní viz. C.6 je vidět stoupající trend koncentrace CO_2. V danou dobu měření bylo přítomno 8 lidí. Vzhledem k otevřenému oknu a nedostatečně ustáleným hodnotám algoritmus vyhodnotil 12 lidí v místnosti.

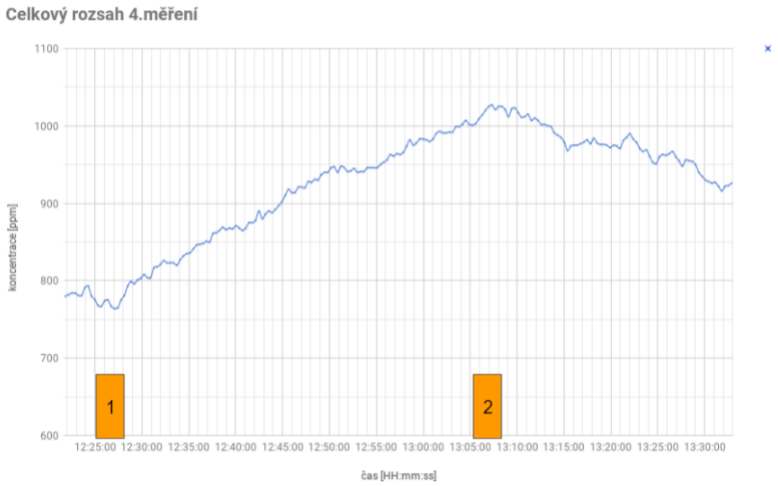
V druhé části měření viz. C.7 trend stoupání už není tak strmý. Důvodem je ustálení podmínek pro měření. Počet lidí v místnosti byl 6, protože jeden student odešel, algoritmus vyhodnotil v místnosti 6 lidí.

Ve třetí části měření viz. C.8 je trend koncentrace oxidu uhličitého ustálen. V 11:54 odcházejí studenti, což je na grafu vidět, že došlo k otevření dveří a okna došlo v tomto bodě měření k poklesu.

Ve čtvrté části měřeni viz. C.9 nejsou přítomni žádní studenti, proto hodnota CO_2 klesá.

4.4 Měření č. 4

Poslední čtvrté měření proběhlo hned po třetím měření. Toto měření bylo rozděleno pouze na tři části po 20 minutách, zúčastnilo se ho 9 osob a v průběhu měření se nijak neměnil. Byly zavřeny okna i dveře. Na grafu měření viz. 4.4 je znatelný rapidní nárůst koncentrace CO_2. Ve 13:10 bylo otevřeno okno a dveře, protože studenti ztráceli pozornost při výuce. Toto je také ověření tvrzení, že hodnota CO_2 nad 1000 ppm lidi už začíná ovlivňovat. V první části měření viz. C.10 algoritmus vypočítal 10 lidí, skutečná hodnota byla 9. V druhé části měření viz. C.11 algoritmus vypočítal 8 lidí. Ve třetí části měření viz. C.12 došlo k poruše ve formě otevření okna a dveří, tím byla tato část měření znehodnocena.



Obrázek 4.4: Celý rozsah 4. měření, *1* označuje příchod studentů, *2* označuje otevření okna a dveří

Závěr

Cílem bakalářské práce bylo vytvoření návrhu a realizace komplexního řešení pro detekci počtu osob v uzavřeném prostředí. Prvotním předpokladem pro tuto práci bylo seznámit se s dostupnou literaturou pro měření koncentrace oxidu uhličitého, dostupných IoT platforem a cloudových služeb. Hlavním parametrem pro vývojovou desku byl zabezpečený přístup k internetu. Po provedení rešerše dostupných materiálů bylo zapotřebí daný hardware naprogramovat tak, aby splňoval přísné podmínky pro komunikaci s cloudovými službami od Amazonu.

Pro měření hodnoty koncentrace CO_2 byl využit měřící princip NDIR, který využívá infračerveného svitu pro spočítání molekul CO_2 v atmosféře na základě odraženého světla o určité vlnové délce. Data z čidla jsou dále zpracována a poslána po sběrnici I^2C.

Dalším nutným krokem bylo pochopení fungování operačního systému FreeRTOS, aby bylo možné použít dostupné SDK od firmy Amazon a šlo systém propojit do jeho infrastruktury. Veškerá komunikace probíhá zašifrovaně, proto je nutné mít přehled o všech certifikátech a přístupové politice na straně serveru.

Pro hotovou měřící stanici bylo vytvořeno jednoduché webové rozhraní, ve kterém byla vizualizována naměřená dat a zobrazen počet osob v místnosti na základě předchozích dat. Finální verze webového rozhraní byla otestována na čtyřech proběhlých měření v prostorách učebny A10 na Technické Univerzitě v Liberci.

Výsledné měření ukázalo, že dynamický algoritmus dokáže celkem přesně detekovat počet osob v místnosti pouze z hodnoty CO_2, jsou-li podmínky měřeného prostředí stálé, bez častých poruch. Za poruchu lze považovat otevřené okno nebo dveře, popřípadě odchod většího počtu osob. Třetí měření testovalo, zdali dokáže algoritmus určit počet osob za otevření jednoho okna, zajišťující stálý přísun čerstvého vzduchu do místnosti. Výsledek měření ukázal, že pro detekci osob nemá tato porucha velký vliv, jenom je potřeba počítat s delším intervalem pro ustálení hodnot oxidu uhličitého. Prodloužením intervalu je možné eliminovat vliv stálé poruchy a počet lidí je následně detekován podle skutečnosti. Pro přesnější nastavení měřící stanice a by bylo zapotřebí dlouhodobějšího experimentálního měření s větším počtem osob v místnosti.

Literatura

[1] YIU, Joseph, 2014. The definitive guide to ARM® Cortex®-M3 and Cortex-M4 processors. Third edition. Amsterdam: Elsevier, Newnes. ISBN 978-0124080829.

[2] RUPARELIA, Nayan, 2016. Cloud computing. Cambridge, Massachusetts: The MIT Press. ISBN 978-0262529099.

[3] GREENGARD, Samuel., 2015. The internet of things. Cambridge, Massachusetts: MIT Press. ISBN 978-0262527736.

[4] CALÌ, Davide, Peter MATTHES, Kristian HUCHTEMANN, Rita STREBLOW a Dirk MÜLLER, 2015. CO2 based occupancy detection algorithm: Experimental analysis and validation for office and residential buildings. Building and Environment. 86, 39-49. DOI: https://doi.org/10.1016/j.buildenv.2014.12.011. ISSN 03601323.

[5] MSR Electronic: Technické materiály, MSR Electronic: Basics of gas measuring technique [online]. [cit. 2018-03-20]. Dostupné z: www.msr-electronic.de

[6] VAFEK, Ing. Zdeněk, 2016. Možnosti měření oxidu uhličitého: měřicí přístroje a čidla. In: Tzbinfo [online]. Časopis Vytápění, větrání, instalace: Dräger Safety, 14.3.2016 [cit. 2018-03-20]. Dostupné z: vetrani.tzb-info.cz

[7] AMAYRI, Manar, Abhay ARORA, Stephane PLOIX, Sanghamitra BANDHYOPADYAY, Quoc-Dung NGO a Venkata Ramana BADARLA, 2016. Estimating occupancy in heterogeneous sensor environment. Energy and Buildings [online]. 129, 46-58 [cit. 2018-03-28]. ISSN 03787788. Dostupné z: linkinghub.elsevier.com/retrieve/pii/S0378778816306223

[8] ČESKO, O technických požadavcích na stavby: § 11, 2009. In: Zákony pro lidi.cz [online]. AION CS, ročník 2009, částka 81, číslo 268. Dostupné také z: www.zakonyprolidi.cz

[9] WOLFGANG JESSEL. DRÄGER SAFETY AG & CO. KGAA a LÜBECK, 2001. Gase - Dämpfe - Gasmesstechnik ein Kompendium für die Praxis. Lübeck: Dräger Safety AG & Co. ISBN 3980807606.

[10] TECHNICAL INFORMATION FOR CDM7160: CO2 Module, 2016. Figaro Engineering Inc.1-5-11 Senba-nishiMino, Osaka 562-8505 JAPAN. Dostupné také z: www.figaro.co.jp/

[11] FreeRTOS Kernel: Reference Manual [online], 2018. 1. Amazon Web Services [cit. 2018-04-15]. Dostupné z:docs.aws.amazon.com/freertos-kernel

[12] FreeRTOS: About [online], [cit. 2018-04-15]. Dostupné z: www.freertos.org/

[13] S. TANENBAUM, Andrew a Herbert BOS, 2014. Modern Operating Systems. 4th edition. Harlow: Pearson Education Limited (Verlag). ISBN 978-1-292-06142-9.

[14] Amazon DynamoDB Developer Guide [online], 2018. 1. Amazon Web Services, Inc. [cit. 2018-04-22]. Dostupné z: docs.aws.amazon.com

[15] AWS IoT Developer Guide [PDF], 2018. 1. Amazon Web Services [cit. 2018-04-22]. Dostupné z: docs.aws.amazon.com

[16] AWS Lambda: Developer Guide [PDF], 2018. 1. Amazon Web Services [cit. 2018-04-22]. Dostupné z: docs.aws.amazon.com

[17] LACHAPELLE, Annie-Claude a James A. LOVE, 2012. Simulink® Model of Single CO2 Sensor Location Impact on CO2: Levels in Recirculating Multiple-Zone Systems [online]. University of Calgary, Alberta, Canada [cit. 2018-05-07]. Dostupné z: www.solarbuildings.ca. Faculty of Environmental Design.

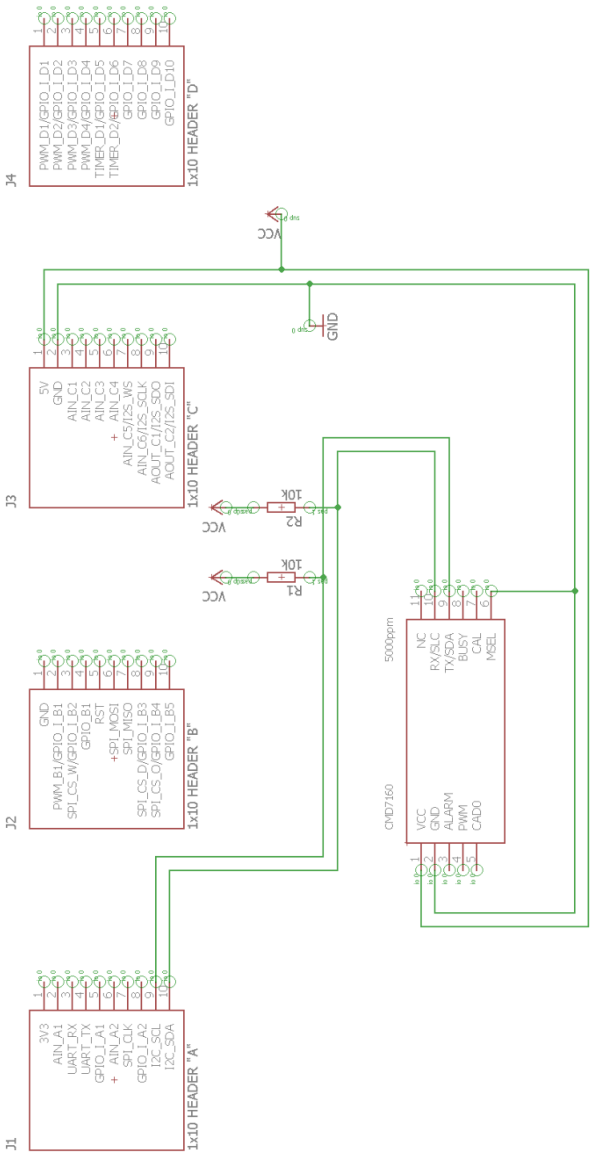
[18] DIGNAN, Larry, 2018. Developers favoring AWS, Microsoft Azure for cloud IoT platforms. In: ZDNet [online]. Between the Lines, 17.4.2018 [cit. 2018-05-08]. Dostupné z: www.zdnet.com

A Přílohy

A.1 Obsah na CD

AmazonFreeRTOS
Grafy měření
Matlab výpočty
Naměřená data
Půdorys A10
Web
Bakalarska_prace.pdf

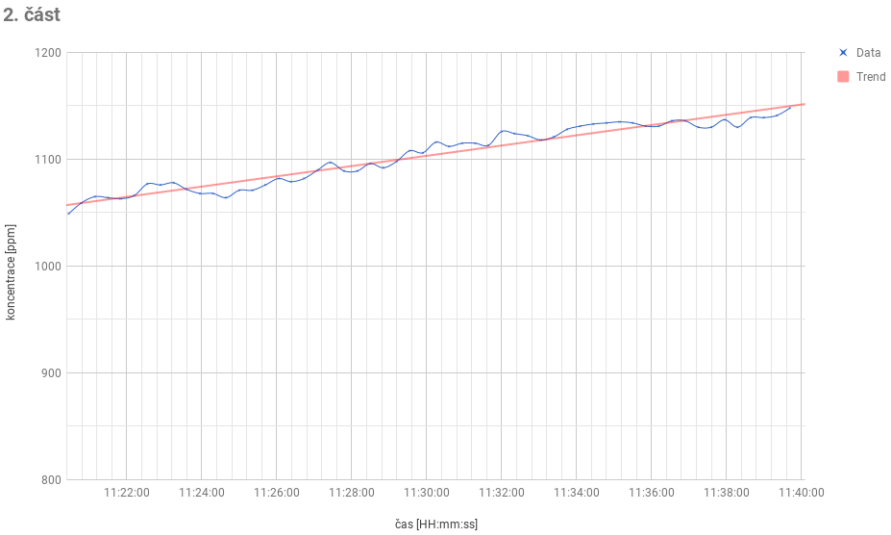
B Schéma zapojení



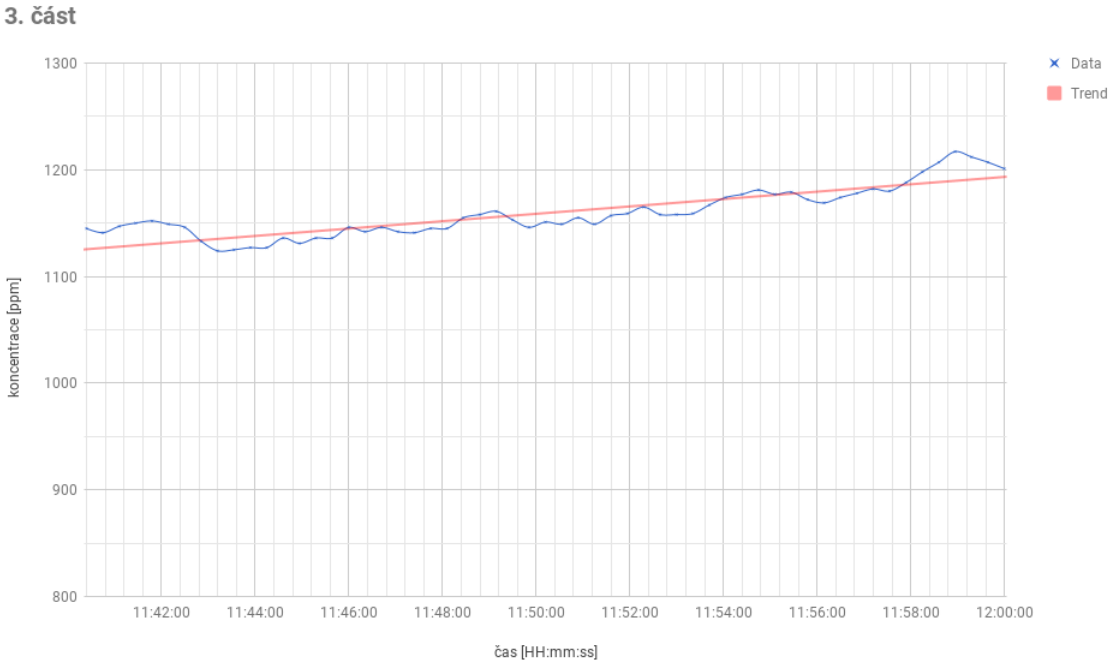
Obrázek B.1: Zapojení CC3220SF a CDM7160

C Data měření

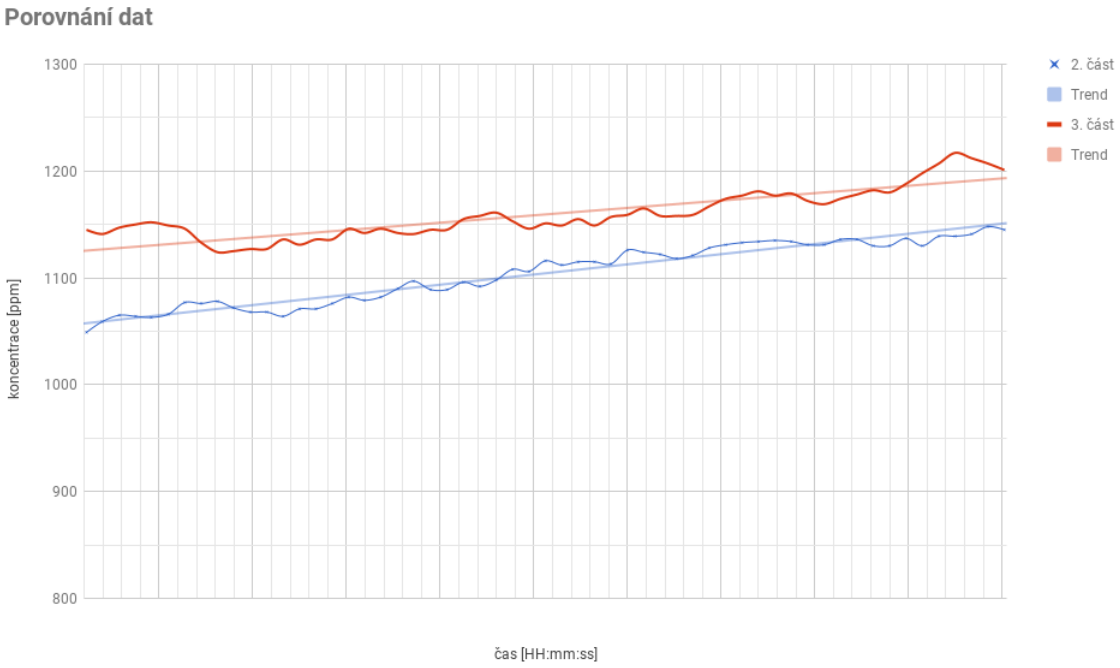
Obrázek C.1: První úsek 2. měření, přítomno 8 osob, algoritmus vyhodnotil 9 osob



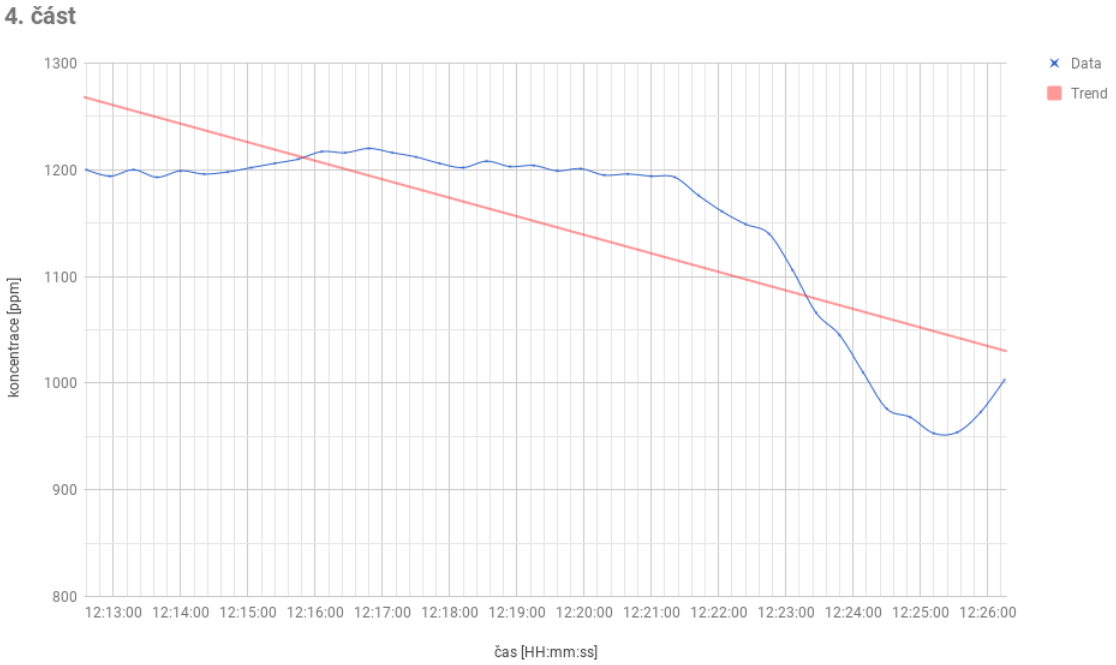
Obrázek C.2: Druhý úsek 2. měření, stoupající trend, přítomno 7 osob, algoritmus vyhodnotil 7 osob



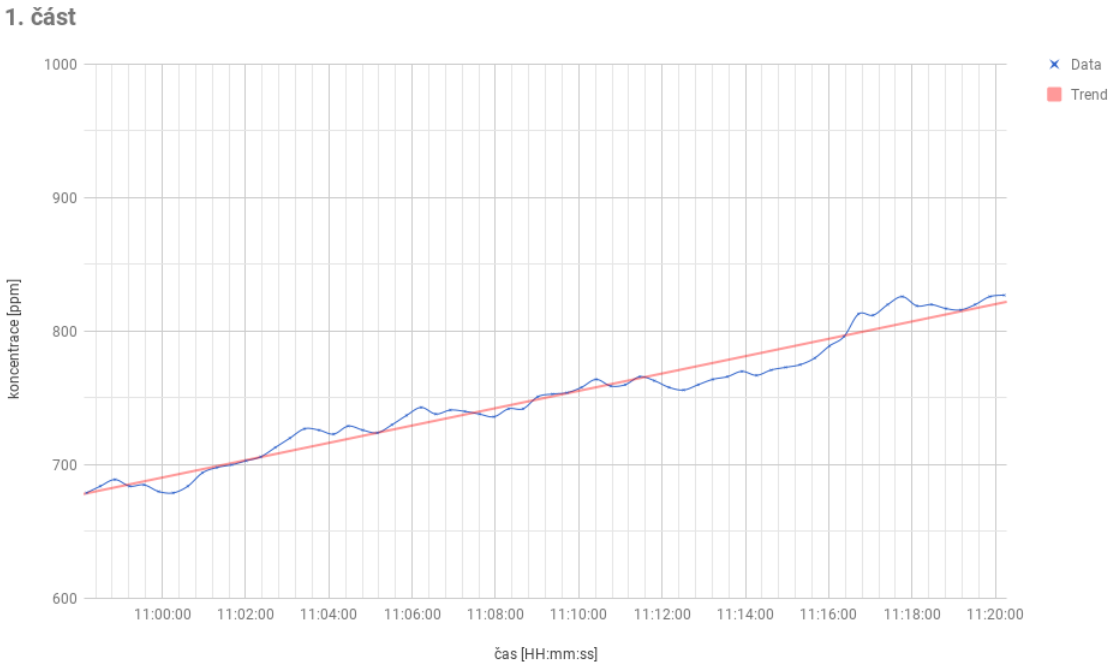
Obrázek C.3: Třetí úsek 2. měření, přítomno 8 lidí, algoritmus vyhodnotil 8 lidí



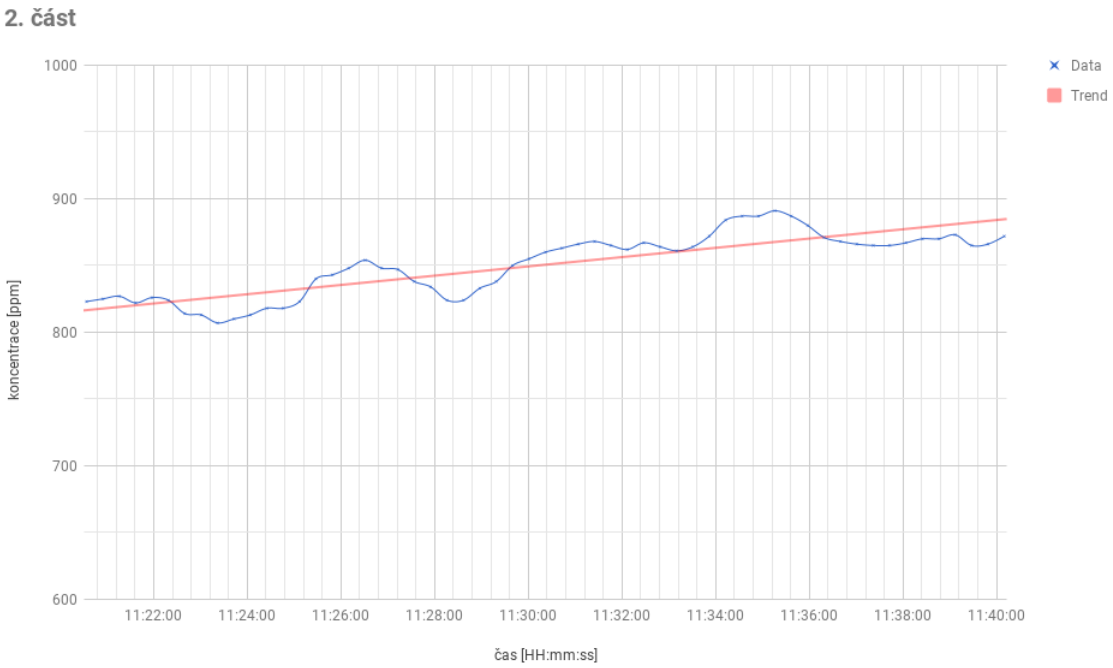
Obrázek C.4: Porovnání dat z druhé a třetí části 2. měření, přítomno 8 *(2)* a 7*(3)* osob



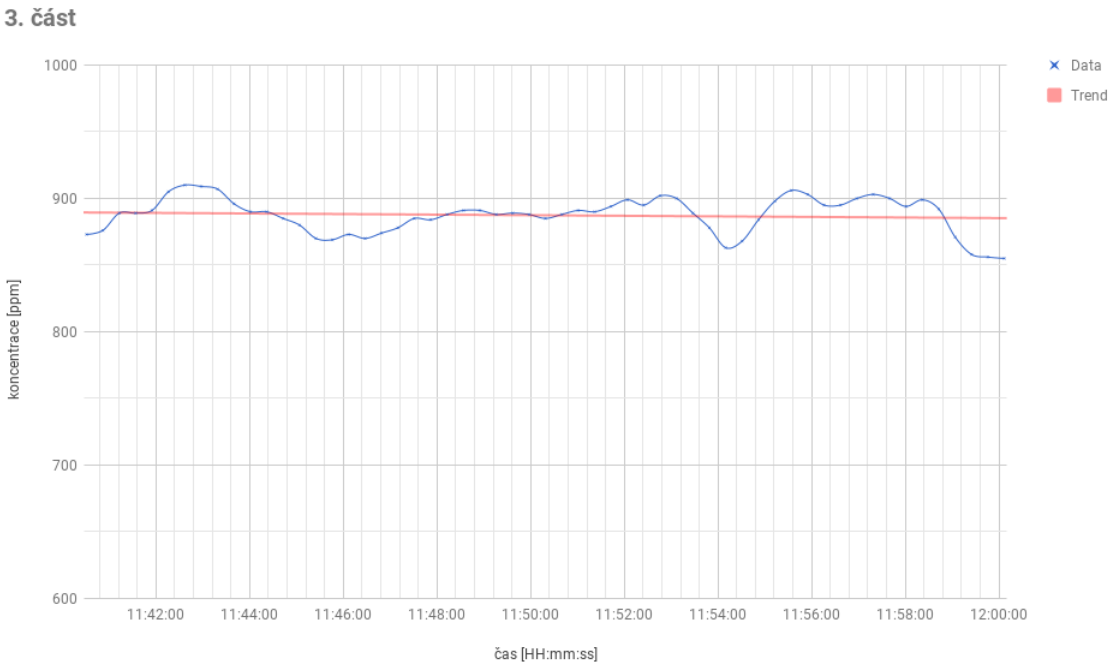
Obrázek C.5: Ukázka poruchy otevřeného okna 2.měření



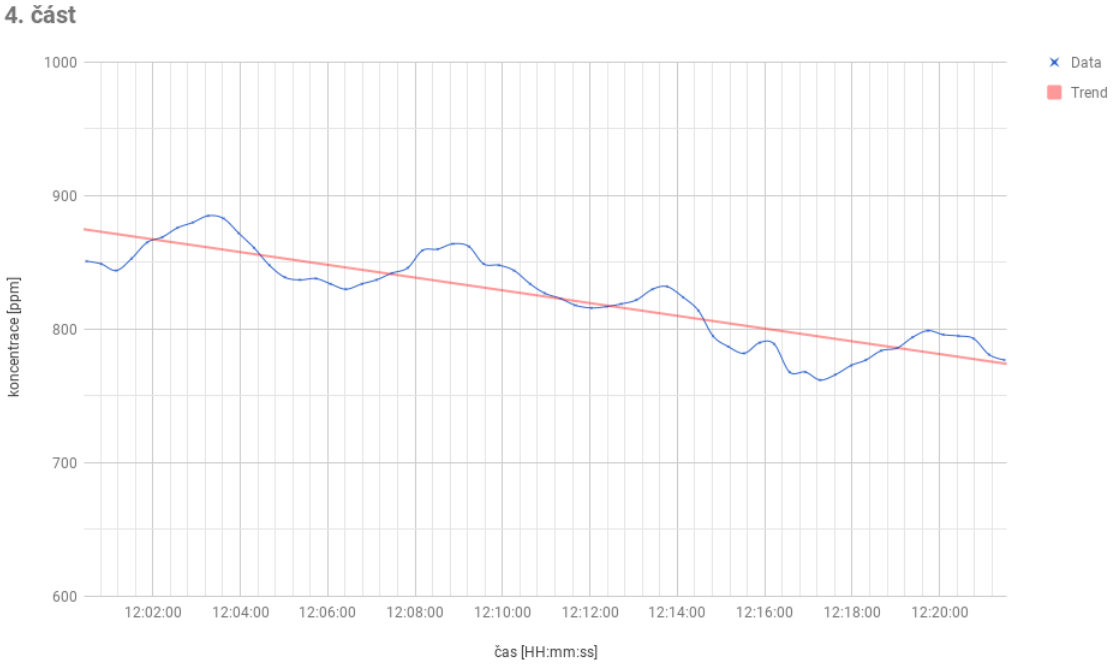
Obrázek C.6: První úsek 3. měření, stoupající trend, přítomno 8 osob, algoritmus vyhodnotil 12



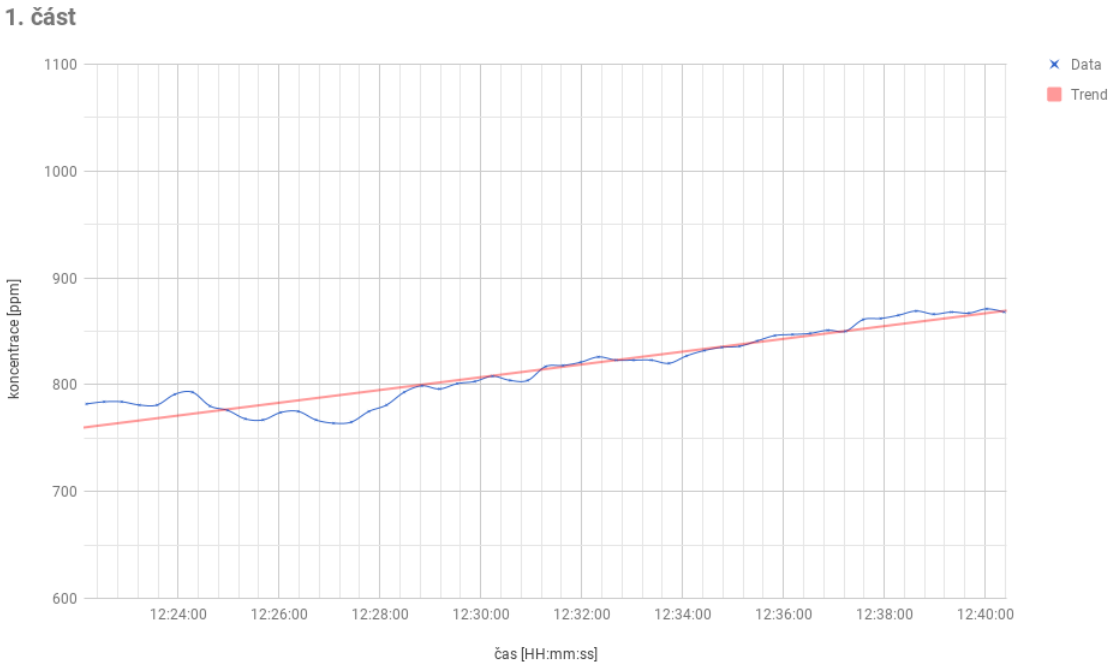
Obrázek C.7: Druhý úsek 3. měření, stoupající trend *(postupné ustálení)*, přítomno 6 osob, algoritmus vyhodnotil 6



Obrázek C.8: Třetí úsek 3. měření, ustálený trend, porucha ve formě otevřeného okna a dveří, během měření došlo k odchodů všech přítomných

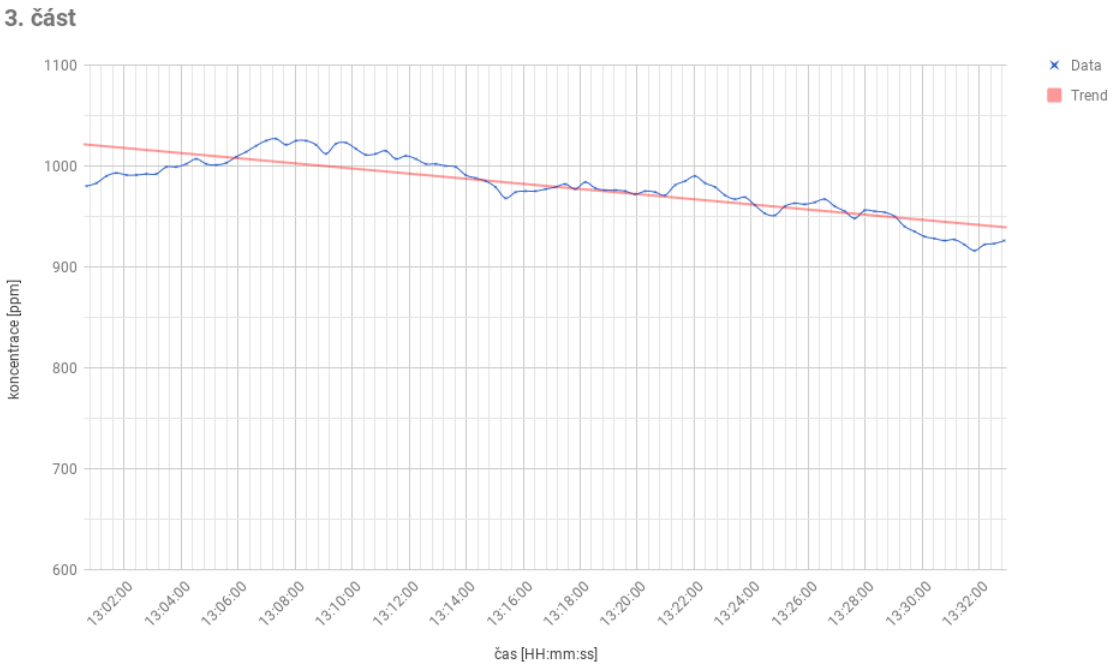


Obrázek C.9: Čtvrtý úsek 3. měření, klesající trend, odchod všech osob



Obrázek C.10: První úsek 4. měření, stoupající trend, přítomno 9 osob, zavřeny okna i dveře, algoritmus vyhodnotil 10 osob

Obrázek C.11: Druhý úsek 4. měření, stoupající trend, přítomno 9 osob, zavřeny okna i dveře, algoritmus vyhodnotil 8 osob



Obrázek C.12: Třetí úsek 4. měření, klesající trend, přítomno 9 osob, otevření okna i dveře, u klesajícího trendu algoritmus vyhodnotil zápornou hodnotu počtu osob