TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Software pro chytrý senzor

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Jan Tichý**
Vedoucí práce: Ing. Lenka Kosková Třísková, Ph. D.

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Software pro chytrý senzor

Jméno a příjmení:	**Bc. Jan Tichý**
Osobní číslo:	M18000154
Studijní program:	N2612 Elektrotechnika a informatika
Studijní obor:	Informační technologie
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	**2020/2021**

Zásady pro vypracování:

1. Seznamte se se zapojením a funkcí prototypu chytrého senzoru teploty a CO2 od společnosti VisionQ.
2. Navrhněte architekturu software čidla, jež prostřednictvím protokolu LoRA vyčítá data ze senzoru a ukládá je do zvoleného cloudového úložiště.
3. Vyberte vhodný operační systém pro realizaci software čidla.
4. Při návrhu software zohledněte další vývoj senzorů – návrh není vázán jen na konkrétní čidlo.
5. Realizujte a popište ukázkový software pro čidlo.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	40 – 50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

1. Internet of things and Data Analytics Handbook, Hwaiyu Geng, John Wiley & Sons, Inc., 2018.
2. Dong B., Prakash V., Feng F, Neill Z. O.: Energy & Buildings A review of smart building sensing system for better indoor environment control,
   Energy & Buildings, 2019, 199, Pages: 29 – 46.
3. Lasla N., Doudou M., Djenouri D., Ouadjaout A., Zizoua C.: Wireless energy efficient occupancy-monitoring system for smart buildings, Pervasive and Mobile Computing, 2019, Vol. 59.

Vedoucí práce:	Ing. Lenka Kosková Třísková, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	19. října 2020
Předpokládaný termín odevzdání:	17. května 2021

Technická univerzita v Liberci
L.S.
-18-

V Liberci dne 19. října 2020

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

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

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

17. 5. 2021 Bc. Jan Tichý

Poděkování

Rád bych poděkoval jmenovitě Lence Koskové Třískové za dlouhodobou spolupráci napříč všemi projekty, které jsem na Technické Univerzitě v Liberci psal. Za vedení, věcné rady a nadchnutí pro vývoj vestavěných zařízení. Dále chci poděkovat firmě VisionQ, která zapůjčila svůj hardware pro tvorbu této práce, panu Jiřímu Merzovi za umožnění této spolupráce a Zdeňkovi Halbichovi za konzultace a pomoc při hledání chyb v realizaci této práce.

Software pro chytrý senzor

Abstrakt

Cílem této práce bylo realizovat software pro vestavěné čidlo na měření koncentrace oxidu uhličitého, teploty a vlhkost s konektivitou do sítě Narrow Band. Software by měl být platformově nezávislý. Naměřená data se vizualizují ve webové aplikaci. Při vývoji firmwaru na platformě STM32 bylo použito CMSIS metody přístupu k periferiím. Knihovny pro měření a zpracování dat se oddělily aplikační vrstvou pro možné použití na jiné architektuře. V práci se realizoval měřící systém pro sběr atmosférických dat v místnosti, poslaných přes NB síť a zobrazovaných prostřednictvím webové aplikace. Výsledky této práce mohou pomoci dalším studentům při vývoji vestavěného systému na míru. Práce obsahuje postupy pro vytváření zakázkového firmwaru, jak pro práci s registry procesoru, tak pro návrh aplikační nadstavby.

Klíčová slova: MCU, čidlo, Narrow Band, firmware, STM32, Koncentrace oxidu uhličitého

Software for smart sensor

Abstract

The main goal is to develop software for embedded board that measure carbon dioxide, temperature and humidity. The board is communicating and sending data measurement using NB-IoT network. Software had to be platform independent. Measured data are visualized on web application. I used STM32 platform for firmware development and CMSIS interface for use of features of microcontroller. A library for data measurement are separate from application layer for better platform compatibility on diferent MCU architecture. The result of this thesis could help another students in developing embedded device. The work contains methods for making embedded firmware, for use of register on MCU and to create an application.

Keywords: MCU, sensor, Narrow Band, firmware, STM32, Carbon dioxide

Obsah

Seznam obrázků

Seznam tabulek

Seznam rovnic

Seznam zkratek

API	Application Programming Interface
BCD	Binary Coded Decimal - dvojkově reprezentované dekadické číslo
CMSIS	Cortex Microcontroller Software Interface Standard
CO_2	Koncentrace oxidu uhličitého
CPU	Central processing unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
EWARN	IAR Embedded Workbench for Arm
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
FRAM	Ferroelectric Random Access Memory
FW	FirmWare
GPIO	General-purpose input/output
HAL	Hardware Abstraction Layer
HW	HardWare
I^2C	Inter-Integrated Circuit - sériová sběrnice
IDE	Integrated Development Environment
IOT	Internet of Things
IP	Internet protocol
IRQ	Interrupt ReQuest
KB	KiloByte
Kb	KiloBit
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
LL	Low Layer
LoRa	Long Range - řešení pro bezdrátový přenos dat
MCU	Microcontroller unit
NB	Narrow Band - řešení pro bezdrátový přenos dat
NDIR	nondispersive infrared sensor
OS	Operační Systém
RAM	Random Access Memory
RTC	Real Time Clock
SDA	Synchronous Data
SLC	Synchronous Clock
SPI	Serial Peripheral Interface
SW	SoftWare
TUL	Technická univerzita v Liberci
UART	Universal asynchronous receiver-transmitter

Úvod

Při výběru své diplomové práce jsem hledal zajímavou výzvu, co by mi pomohla zdokonalit své znalosti, které jsem získal v průběhu celého studia na fakultě mechatroniky. Chtěl jsem projekt řešit komplexně od návrhu požadavků na systém až po realizaci všech funkčních prvků.

Tyto předpoklady splnilo zadání od firmy VisionQ, která mě oslovila díky mým zkušenostem s měřením koncentrace oxidu uhličitého z mé bakalářské práce. Firma navrhla prototyp desky čidla a dodala základní představu, jak by systém měl fungovat. Na základě předběžných schůzek jsme navrhli požadavky na funkci a vlastnosti chytrého čidla.

Hlavním cílem bylo navrhnout funkční FW pro desku čidla. Propojit HW se sítí NB-IoT a z cloudového úložiště vyčítat naměřená data do webového rozhraní. Vice zadání rozebírám v kapitole 1. Čidlo by v budoucnu mohlo sloužit pro monitorování atmosférických dat klimatu ve třídách základních škol.

Obrázek 1: Chytré čidlo

1 Požadavky na senzor firmou VisionQ

S firmou VisionQ jsme se domluvili na spolupráci vývoje FW pro jejich chytrý senzor, který by monitoroval teplotu, vlhkost a koncentraci oxidu uhličitého v místnosti. Sběr těchto dat by měl trvat alespoň dva roky na dvou lithiových bateriích. Další fází projektu je vytvoření frontend webové aplikace, která naměřená data stáhne z cloudového úložiště a ta budou vizualizována formou tabulky a grafů. V této kapitole rozeberu zadání a popíši jednotlivé části systému. Stručný bodový popis požadavků:

1. Oživit všechny periferie
2. Navázat komunikaci přes bezdrátovou síť s Cloudem
3. Vyvinout FW, který bude energeticky úsporný
4. Vymyslet datový model
5. Navrhnout webovou aplikaci
6. Implementovat rozdělovač pro práci s datovým modelem

1.1 Firmware

Nároky na firmware byly vcelku jednoznačné, všechny komponenty na řídící desce musí komunikovat a fungovat. Čidlo má měřit data a posílat je do cloudového úložiště od společnosti Vodafone. Mým základním úkolem bylo senzor zprovoznit a postupně napsat firmware pro jednotlivé součásti hotového testovacího vzorku. Velký důraz byl kladen na spotřebu. Zařízení bude fungovat na dvou *Saft Lion* baterií, proto je žádoucí, aby procesorový čas trval, co nejkratší dobu.

Pro úsporu baterie se procesor uspí do hlubokého spánku a probudí se pouze a jenom tehdy, je-li vybuzen externím impulsem přerušení. Periodu měření řídí obvod reálného času, do kterého se pokaždé, než se uloží MCU do spánkového režimu, nahraje časový interval pro další probuzení.

1.2 Frontend Aplikace

Webová aplikace bude sloužit jako vizualizace naměřených dat čidlem. Cílem je stáhnout data z cloudu s použitím dostupného API a navrhnout dekodér pro zpracování binárního formátu dat do čitelné podoby pro uživatele. Mezi základní požadavky na webovou aplikaci patří:

- Využití API pro čtení hodnot uložených na cloudu
- Základní vizualizace dat (tabulka, graf)
- Využití frameworku **Vue.js**
- Implementace datového parseru

Hlavní důraz u této aplikace byl kladen na použití javascriptového frameworku **Vue.js**, který firma využívá i pro jiné své projekty tak, aby šla jako třída jednoduše implementovat do jejich prostředí.

Zadání této diplomové práce původně nepočítalo s vývojem webové aplikace. I přesto se v práci stručně zmíním o této nadstavbě, protože jsem ji pro zákazníka v průběhu vývoje vytvořil.

1.3 Popis zvoleného HW

Veškeré komponenty řídící desky od firmy VisionQ byly již dané, až na čidlo pro měření koncentrace oxidu uhličitého, to bylo zvoleno po mém doporučení. V této kapitole přiblížím základní specifikace zvoleného hardwaru a jejich výhody a nevýhody. Hlavní důraz u všech periferií byl kladen hlavně na jejich spotřebu. Každou periferii zařízení je možné prostřednictví microprocesoru vypnout pomocí GPIO výstupů.

Mikroprocesor STM32L051C8T6

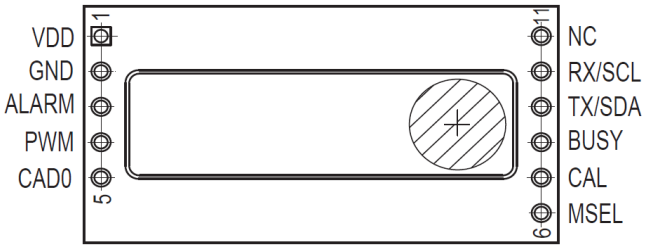
Mikroprocesor, dále jenom **MCU**, je hlavním prvkem celého čidla. V tomto případě se jedná konkrétně o MCU **STM32L051**, který předem určila firma VisionQ. S tímto MCU mají dobré zkušenosti u jiného výrobku a jeho použití bylo jedním z požadavků. Tento nízkospotřebový procesor architektury ARM Cortex-M0+ tiká na 32 MHz, obsahuje 32 KB Flash paměti a 8 KB paměti RAM. Nejdůležitější vlastností je spotřeba energie, která dosahuje podle dokumentace [1] v nejhlubším spánku pouze 0.27 µA. V tomto režimu, ale dochází k vymazání paměti RAM a lze MCU probrat z hlubokého spánku pouze dvěma *wokeup* vstupy. Vzhledem k povaze aplikace, jakožto bateriového zařízení, je spotřeba nejvíce rizikový faktor.

periferie	Flash	EEPROM	Timer	SPI	I^2C	UART	GPIO	Freq
Počet	32 KB	2 KB	5	2	2	2	37	32 MHz

Tabulka 1.1: Základní vlastnosti MCU [1]

Senzor CMD7160 na měření CO_2

Tento senzor byl do aplikace přidán na mé doporučení. S touto komponentou jsem již v minulosti pracoval na své bakalářské práci *Inteligentní systém pro anonymní detekci počtu osob v místnosti* [2]. Čidlo měří koncentraci oxidu uhličitého na optickém principu. Více o této měřící metodě se lze dočíst v mé předchozí práci zde [2]. U tohoto čidla je celkem nežádoucí vysoká spotřeba energie, a to řádově vyšší oproti ostatním komponentám, viz graf 3.5.



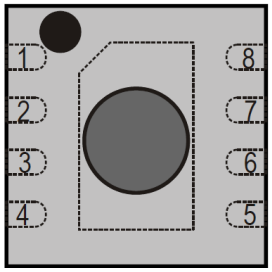
Obrázek 1.1: CMD7160 rozložení pinů [3]

Rozsah měření	300-5000 ppm
Princip měření	NDIR
Napájení	4,75-5,25 V
Přesnost	-+50 ppm+3 %
Sběrnice	I^2C/UART
Interval měření	2s
Spotřeba	60 mA špička, 10 mA avg.

Tabulka 1.2: Základní vlastnosti CMD7160 [3]

Čidlo SHT31-DIS-B pro měření teploty a vlhkosti

Jedná se o kalibrované rychlé čidlo komunikující na sběrnici I^2C, umožnující rychlost posílání dat až 1 Mhz. Jde o senzor od firmy Sensirion. Interval měření se liší podle zvoleného módu měření.



Obrázek 1.2: SHT31 rozložení pinů [4]

Pin	Název	Funkce
1	SDA	Sériová data
2	ADDR	Pin pro změnu adresy
3	ALERT	Indikátor stavu čidla
4	SLC	Sériové hodiny
5	VDD	Napájení
6	nRESET	Reset pin aktiní v nule
7	R	Bez funkce
8	GND	Zem

Tabulka 1.3: Popis pinů SHT31 [4]

Rozsah teploty	5-60 °C
Rozsah vlhkosti	20-80 %RH
Napájení	3,3-5,5 V
Přesnost teploty	+-0.1 °C
Přesnost vlhkosti	+-1.5 %RV
Sběrnice	I^2C
Interval měření	4/6/15 ms
Odběr	220 uA

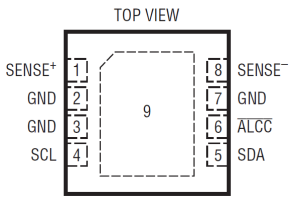
Tabulka 1.4: Základní vlastnosti SHT31 [4]

Culombmetr LTC2943IDD

Hlavní účel této periferie je měření stavu nabití, napětí a proudu baterie. Tyto informace budou sloužit pro upozornění uživatele na výměnu baterií v případě nízkého napětí na článcích.

Napájení	-0.3 do 24 V
Rozsah napětí	0-20 V
Přesnost napětí	+-50 mV
Sběrnice	I^2C
Odběr	650(idle)/15(off)/80(sleep) uA

Tabulka 1.5: Základní vlastnosti LTC2943IDD [5]



Obrázek 1.3: LTC2943 rozložení pinů [5]

Externí hodiny reálného času RV-3028-C7

Hlavním důvodem použití externího obvodu reálného času je generování pulzu přerušení pro probuzení MCU z hlubokého spánku. Pro nastavení časového intervalu se používá číselný zápis **BCD**. Tato periferie je jedna z mála, která poběží i v době spánku čidla. Proto byl do projektu vybrán takový modul, který za běhu spotřebovává pouze 60 nA.

Napájení	1,2-5,5 V
CLKOUT	32768 Hz
Sběrnice	I^2C
Odběr	60 nA

Tabulka 1.6: Základní vlastnosti RV-3028-C7 [6]

Dočasná paměť FRAM FM24CL16B-G

Tato paměť v projektu slouží pro dočasné ukládání naměřených dat. Vzhledem k nutnosti udržení baterie naživu co nejdéle se použila paměť, která uchovává data i po odpojení napájení. Jedná se o nevolatilní paměť, která využívá schopnosti feroelektrického jevu. Hlavní rozdíl, oproti EEPROM, je v rychlosti zápisu do paměti, který je téměř bez zpoždění.

Napájení	2,7-3,65 V
F-RAM	16 Kb
Sběrnice	I^2C
Odběr	3 uA

Tabulka 1.7: Základní vlastnosti FM24CL16B-G [7]

Lithiový článek LS14500

V projektu jsou použity dvě lithiové baterie **SAFT LS 14500 STD**. Jde o baterii standartu AA s napětí 3.6V. Složení baterie je $Li - SOCI_2$. Výrobce uvádí dlouhou životnost, nízké samovolného vybíjení (1% za rok při teplotě 20°C). Baterie splňuje bezpečnostní standart IEC 60086-4 a IEC 60079-11 [8].

Napětí	3.6 V
Kapacita	2,6 Ah
Odběrová špička	250 mA/0.1 s
Doporučený odběr	50 mA
Provozní teploty	-60 °C - +85°C

Tabulka 1.8: Základní vlastnosti LS14500 [8]

Komunikační modul Narrow Band BC95-B20

Jedná se o bezdrátový komunikační modul, který má schopnost napojení do NB-IoT sítě od firmy Vodafone. Zařízení lze ovládat pomocí AT příkazů z UART sběrnice. Pro ladění připojení NB modulu šlo k čidlu připojit PC a připojení důkladně otestovat. Pro správné spojení se sítí je potřeba mít k NB modulu zapojenou ESIM. Hlavním využitím modulu je posílání naměřených dat do cloudu. Sekundární vlastností této komponenty je, že po připojení do sítě během bootovací sekvence se ze sítě přečte aktuální čas, který je následně uložen do RTC periferie.

Napájení	3.8 V
Vysílací frekvence	800 MHz
Sběrnice	UART
Odběr	6(idle),250 mA(trans), 5 uA(sleep)

Tabulka 1.9: Základní vlastnosti Narrow Band BC95-B20 [9]

2 Rešerše trhu s cílem mapovat a vzájemně porovnat existující řešení

V této kapitole se budu stručně zabývat teoretickou částí práce. Shrnuji komerční možné řešení při dodržení základních požadavků. Věnuji část textu vestavěným operačním systémům a v poslední řadě přibližuji srovnání mezi EEPROM a použitou FRAM pamětí.

2.1 Dostupná komerční řešení

Než jsem začal s realizací diplomové práce, udělal jsem si průzkum trhu, hledal jsem zařízení s podobnými rysy podle následujících porovnávacích kritérií.

- Čidlo měří CO_2
- Čidlo měří i jiné veličiny
- Jedná se o bateriové zařízení
- Výdrž na baterii
- Konektivita
- Indikace mařených dat

Výsledek mého průzkumu ukazuje tabulka 2.1. Zjistil jsem, že se řešení dost zásadně liší v garanci běhu na baterii a hlavně v ceně. Osobně považuji za nejlepší komerční volbu čidlo **Unipi**, které je cenově dostupné a měří množství veličin.

Název	CO_2 Guard 10 [10]	MAX02003 [11]	Unipi [12]	COMET W8810 [13]
Cena	5 950 Kč	10 700 Kč	4 199 Kč	10 909 Kč
Princip měření CO_2	NDIR	NDIR	NDIR	NDIR
Měřené veličiny	CO_2	CO_2, teplota, vlhkost	CO_2, intenzita osvětlení,teplota, tlak, vlhkost	CO_2, Teplota
Bateriové zařízení	ANO	ANO (Li-SOCl 2)	NE	ANO
Kapacita baterie	4,8 Ah	6 Ah	-	8,5 Ah
Zivotnost na baterii	24 měsíců	neuvádí	-	4 měsíce - 7 let
Konektivita	NE	LoRa IoT	RS-485, Wi-Fi, LoRa	Sigfox
Indikace měření	LED	LCD	Webové rozhraní, LED	Display, webová aplikace

Tabulka 2.1: Porovnání komerčních řešení

Název	FreeRTOS	Mbed OS	embOS	Contiki	RIOT	TinyOS	Zephyr
min RAM	<4 KB	<13 KB	70 B	<2 KB	<1,5 KB	<1 KB	<16 KB
min ROM	<9 KB	<38 KB	<1,7 KB	<30 KB	<30 KB	<4 KB	4-5 KB
podpora C	ANO	ANO	ANO	ANO	ANO	NE	ANO
Licence	MIT	Apache 2.0	Proprietary	BSD	LGPL	BSD	Apache 2.0
Multi-thread	ANO	ANO	ANO	ANO	ANO	NE	ANO
Real-time	ANO	ANO	ANO	NE	ANO	NE	ANO
Power saving mode	ANO	ANO	ANO	ANO	ANO	ANO	ANO

Tabulka 2.2: Porovnání vestavěných operačních systémů

2.2 Operační systémy pro vestavěné systémy

Pro vývoj zakázkové aplikace lze využít několik metod. Nejsnadnějším možným řešením je provádět vše v průběhu nekonečné smyčky, která se neustále opakuje. Tento postup se využívá u jednoduchých projektů. Další možností je využití z dostupných vestavěných operačních systémů. V této kapitole bych rád vybral takové operační systémy, které by se mohly hodit pro tento projekt.

Výhody operačního systému:

- Stabilita
- Rychlost
- Optimalizace vykonávání úloh
- Přenositelnost

Nevýhody operačního systému:

- Horší hledání chyb
- Hůře se udržuje
- Jsme limitovaný zdroji procesoru
- Vliv na vyšší spotřebu baterie

Využití operačního systému může být užitečné u projektů, které jsou komplexní, případně je nutné vykonávat úlohy v přesný čas. Více o funkcích a použití operačního systému u vestavěných projektů uvádím ve své bakalářské práci [2].

Pro výběr systému jsem sledoval tyto parametry:

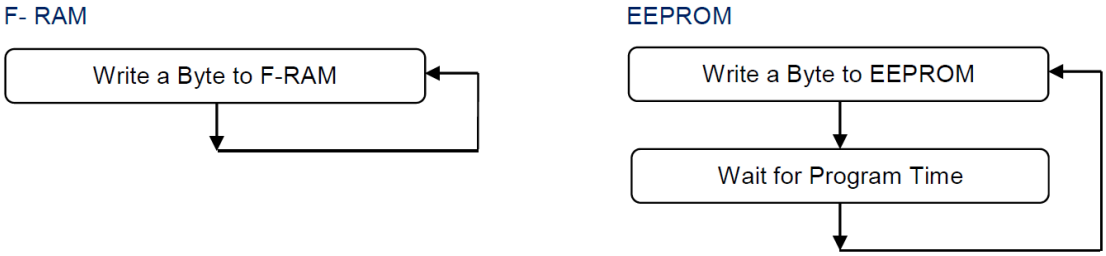
- Nároky na paměť
- Podpora jazyka C
- Podpora HW
- Nároky na spotřebu
- Podpora multi-threadingu
- Podpora RTOS
- Licence

V tabulce 2.2 jsem shrnul sedm běžně používaných operačních systémů, ze kterých bych vybíral. Vzhledem k situaci, že zákazník si nakonec nepřál použití operačního systému, tak jsem zde pouze shrnul dostupné platformy.

2.3 Porovnání FRAM s EEPROM pamětí

Tradiční nevolatilní paměť uchovává data na pole unipolárních tranzistorů pracujících s plovoucími hradly. Tento princip zápisu způsobuje značné zpoždění při zápisu dat. Pro porovnání FRAM a EEPROM pamětí o velikostech 64 Kb a frekvenci zápisu 20 MHz trvá zápis do EEPROM *256 b / 5 ms*. Kompletní zaplnění paměti trvá 1283,3 ms. Zápis pro FRAM paměť je *256 b/14 µs* a kompletní zaplnění trvá *3,23 ms* [14].

Z toho plyne, že ačkoliv má, podle tabulky 2.3, FRAM paměť větší spotřebu energie na zápis dat, tak ale řádově kratší doba zápisu má pozitivní vliv na celkovou spotřebu energie našeho bateriového zařízení.



Obrázek 2.1: Srovnání rychlosti zápisů mezi F-RAM a EEPROM [14]

Další rozdílem těchto technologií je počet přepsání pamětí. Zatím co EEPROM lze přepsat $10^6\times$, tak FRAM lze přepsat $10^{14}\times$, což je v porovnání s EEPROM pamětí značný rozdíl.

	Cypress 64 Kb		
Room/Date	F-RAM (FM25CL64B)	EEPROM (AT25640B)	Jednotky
Odběr zápisu	77.1	33	μA
Běh programu	-	2600 (1.5 ms)	μA
Odběr čtení	82	640	μA
Odběr standby	1.35	0.62	μA

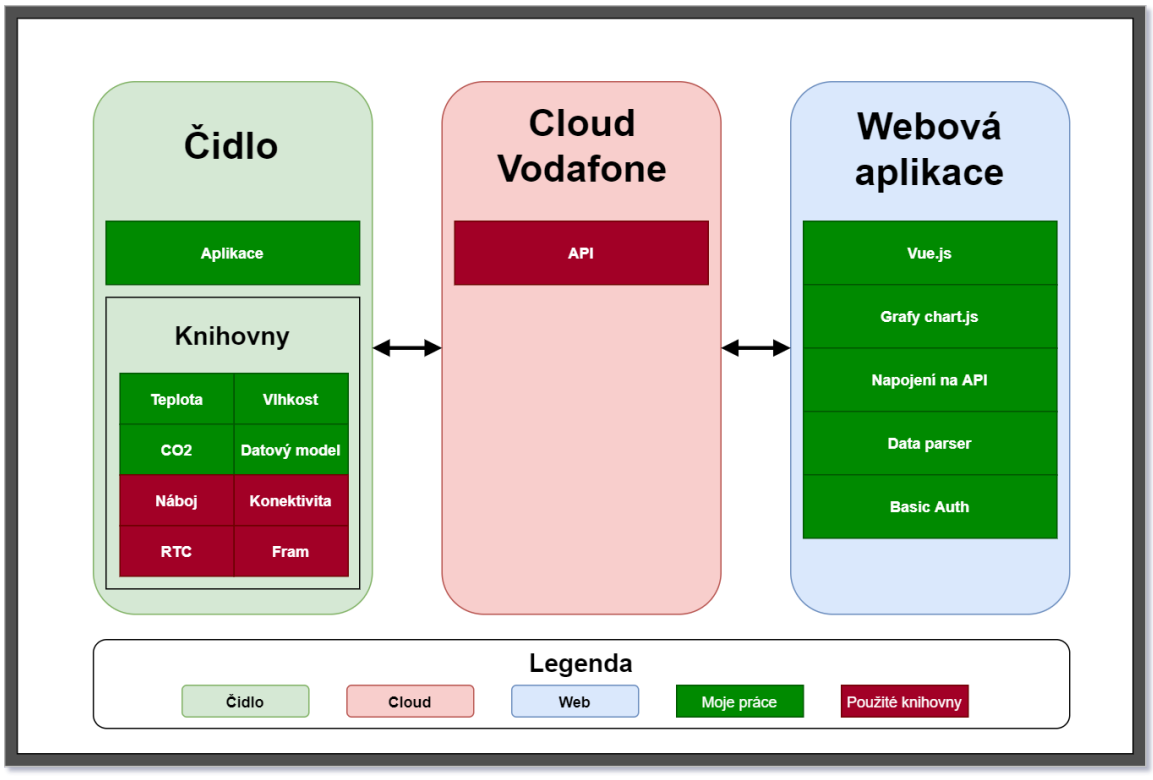
Tabulka 2.3: Porovnání FRAM a EEPROM [14]

3 Popis hledání řešení

V této kapitole chci shrnout funkci celého ekosystému aplikace. Jde o souhrn stavových, vývojových a funkčních diagramů. Nastíním zvolený postup pro řešení a rozdělím jej podle prvků, které jsem vytvářel a které jsem pouze použil.

3.1 Základní rozdělení projektu

Projekt chytrého čidla by šel rozdělit na tři základní stavební kameny. Prvním je čidlo samotné, které obstarává data. Druhým stavebním kamenem je infrastruktura NB-IoT sítě od firmy Vodafone. Třetím kamenem je webová aplikace pro vizualizaci a zpracování dat.



Obrázek 3.1: Základní schéma projektu

Jak už legenda na obrázku 3.1 napovídá, sytě zeleně jsou vyznačené části práce, které jsem vytvořil a červeně se znázorňují bloky, které jsem dostal k dispozici od zá-

kazníka. Ve schématu jsem vizualizoval pouze funkční prvky, které popisují základní vlastnosti projektu. Rozbor vyvinutých a použitých knihoven uvádím v realizační části textu viz 4.2.

Čidlo

Nejvíce práce bylo odvedeno v této části práce. V první fázi bylo zapotřebí desku oživit a potvrdit funkci jednotlivých komponent a periferií. Po potvrzení funkce bylo nutné pro každý funkcionální blok vytvořit nebo použít již existující knihovnu. U všeho bylo potřeba dbát na spotřebu energie. Proto každá funkční komponenta lze pomocí GPIO výstupu z MCU přímo odpojit od napájení. Pro odesílání dat na cloud se muselo počítat s tím, že se jedná o nejvíce energeticky nákladnou operaci. Proto se odesílání dat omezilo na periodu jedenkrát za hodinu. Interval měření se stanovil po pěti minutách.

Dalším úkolem bylo nutné sestavit základní datový model, který je posílán do úložiště. Pomocí tohoto modelu se budou data dočasně ukládat do FRAM paměti a po hodinové periodě se odešlou do vzdáleného úložiště.

V poslední fázi práce jsem vytvořil aplikaci, která spojila vše do jednoho funkčního celku.

Cloud Vodafone

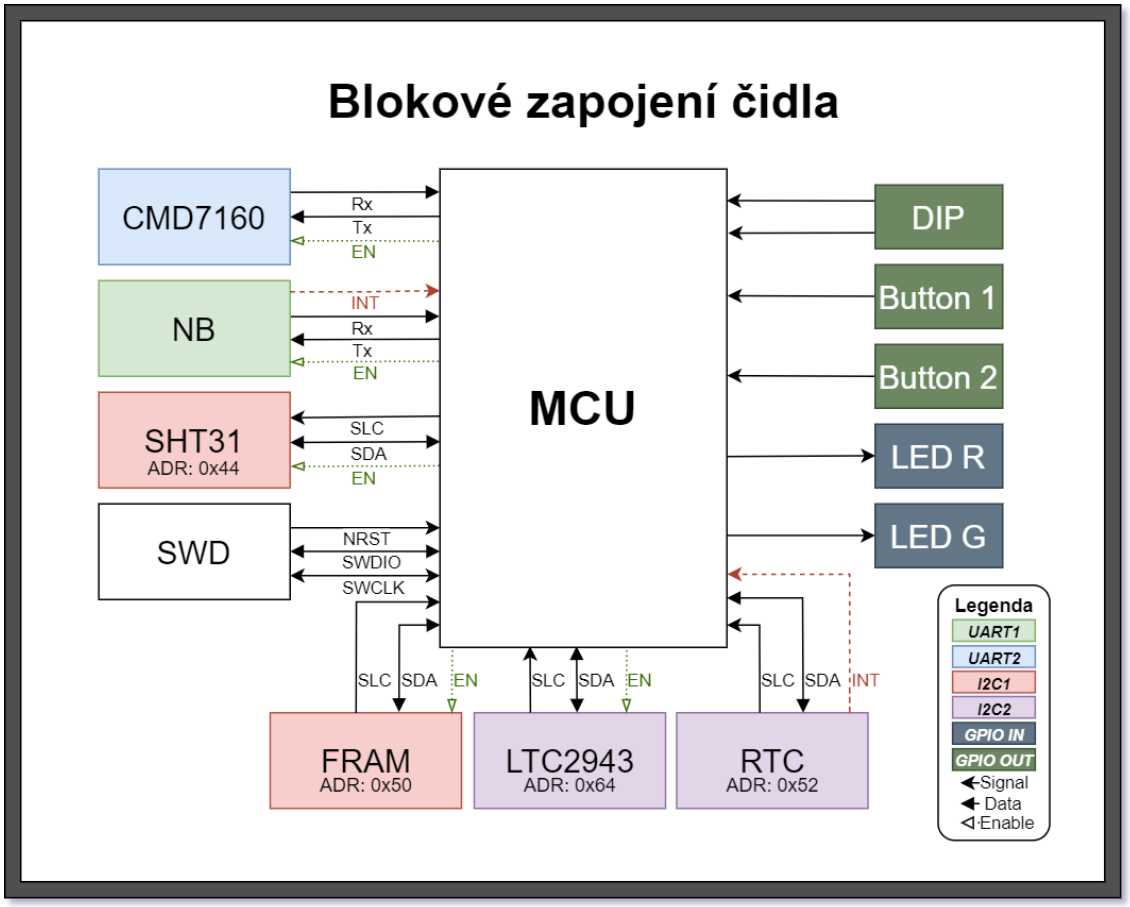
Samotné připojení do sítě NB-IoT nebylo obtížné, tuto část jsem dostal již připravenou od zákazníka. Stejně tak jsem dostal k dispozici hotové REST API pro napojení na tento cloud.

Webová aplikace

U vývoje webové aplikace jsem dbal na přání zákazníka. Využil jsem dostupný framework **vue.js**. Velký důraz při vývoji jsem kladl na snadnou integraci pro budoucí řešení zákazníka. Použil jsem pro vizualizaci dat JS knihovnu **char.js**. Dalším úkolem bylo vyčtení dat pomocí API na základě datového modelu.

3.2 Blokové zapojení čidla

Na blokovém schématu jsou vidět všechny důležité vazby mezi MCU a periferiemi připojenými k zařízení. Směr toku digitálních signálů určují šipky, které se dělí na tři typy. Ostrá šipka označuje elektrický signál, vyplněná tupá šipka značí datový vodič a nevyplněná tupá šipka zastupuje vodič pro zapnutí napájení dané komponenty. Ostrá červená šipka vizualizuje periferie, které mají možnost probudit čidlo z hlubokého spánku pomocí přerušení.



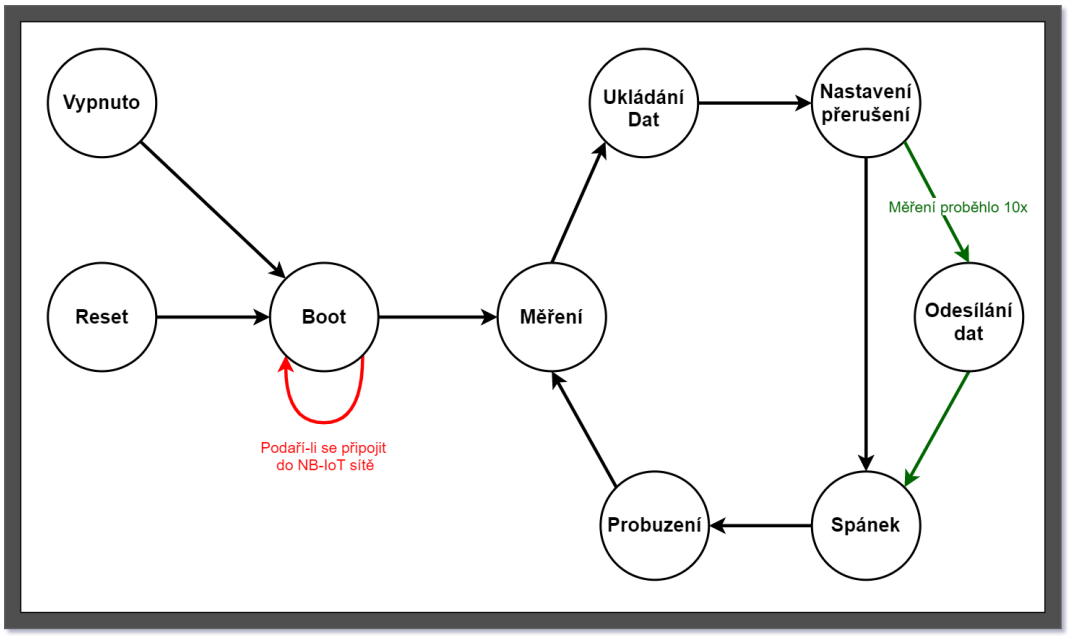
Obrázek 3.2: Blokové schéma zapojení čidla

Modré a světle zelené bloky společně s vodiči Rx a Tx jsou používány sběrnicemi UART. Červené a fialové obdélníky společně s vodiči SLC a SDA indikují sběrnice I^2C. Pro lepší přehlednost jsem k I^2C prvkům doplnil i jejich adresy. Procesor disponuje oběma typy sběrnic v páru a proto jsou barevně odděleny podle toho, jaké periferie sdílí společnou sběrnici.

Bloky tmavě zelené a šedé s bílým písmem naznačují vstupně výstupní periferie. Pro indikaci stavu čidla slouží dvě LED diody. Pro manipulaci s čidlem a vývojem FW je k dispozici DIP přepínač umožnující nastavit čtyři různé stavy a dvě tlačítka. Jedno tlačítko ovšem slouží pro tvrdý restart procesoru.

3.3 Stavový diagram měření

Stavový diagram popisuje základní chování čidla. Zapnutí čidla je podmíněno vložením alespoň jedné baterie nebo připojením externího napájení. Je-li tomu tak, následuje bootovací proces, který obsahuje kompletní oživení MCU, zpřístupnění GPIO vstupů i všech čtyř sběrnic. Spustí se testovací připojení do sítě Narrow Band. Dojde-li k pozitivní reakci, může čidlo začít s prvním měřením. Pokud se ale připojení do sítě z nějakého důvodu nepodaří, čidlo se restartuje a celá startovací smyčka se zopakuje.



Obrázek 3.3: Stavový diagram měření

Měřící cyklus zapne napájení všem čidlům a čeká se, dokud nebude dostupná informace o stavu oxidu uhličitého v prostředí. Tento interval trvá cca 2 vteřiny. Po přečtení dat z dostupných čidel se data uloží do FRAM paměti a nastaví se do obvodu reálného času doba, za jak dlouho potřebuji čidlo opět pomocí přerušení vzbudit. Po uložení intervalu se MCU přepne do hlubokého spánku, odpojí všechny nepotřebné periferie a vypne vnitřní krystal GPIO vstupy. V tomto stavu lze MCU probudit pomocí přerušení z RTC nebo pokud by Narrow Band modul přijal zprávu zvenčí.

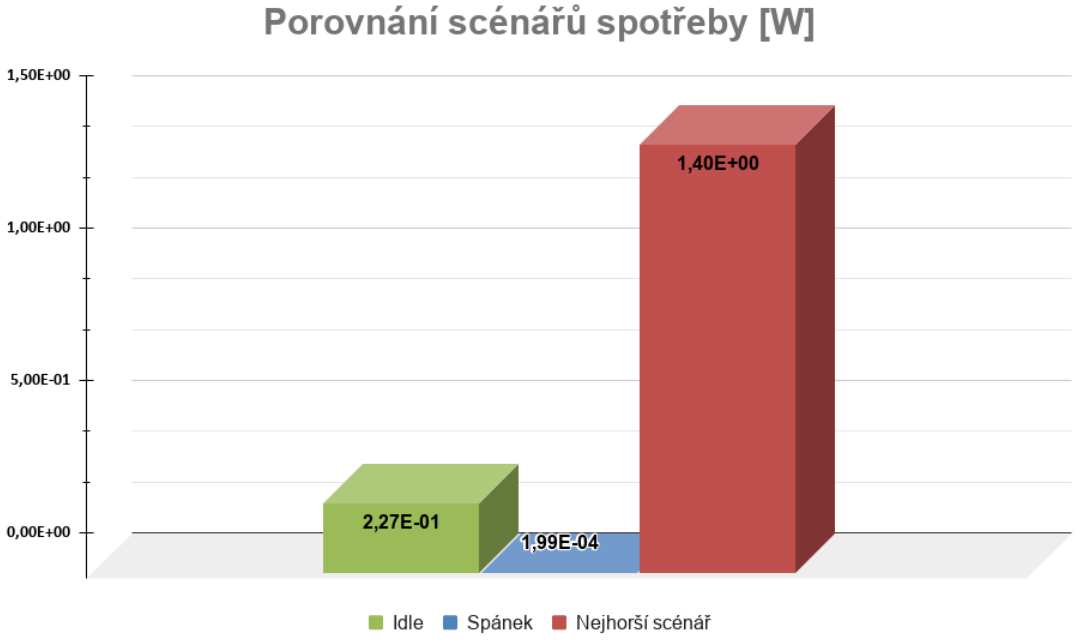
Dojde-li k přerušení, probere se čidlo z hlubokého spánku. Zapne se krystal a nastaví se potřebné GPIO a sběrnice, podobně jako je tomu u bootovací smyčky. Po zapnutí periferií se přepne čidlo opět do stavu měření. Tato smyčka se opakuje desetkrát. Při desátém měření se všechna naměřená data uložená do FRAM paměti odešlou do cloudového úložiště a z paměti FRAM se vymažou. Počítadlo se vynuluje a měřící smyčka opět pokračuje.

3.4 Spotřeba jednotlivých komponent

Nejhlídanějším parametrem celé práce byla spotřeba energie. Jedním z hlavních požadavků zákazníka byla co nejdelší životnost na jedno nabití. Rozdělil jsem chování chytrého čidla na tři základní scénáře. Idle označuje základní běh programu. Spánek vizualizuje minimální spotřebu v době, kdy jsou odpojené téměř všechny periferie a MCU je udržováno v hlubokém spánku. Nejhorší scénář popisuje maximální možnou spotřebu jednotlivých použitých komponent. Všechny údaje vychází z dostupných dokumentací, které cituji v 3.1.

			Spotřeba		
Typ HW	**Odběr**	**Napětí [V]**	**Idle [W]**	**Spánek [W]**	**Nejhorší scénář [W]**
Narrow Band [9]	5 uA(sleep), 6 mA(idle), 250 mA(transmit)	3,60	2,28E-02	1,90E-05	9,50E-01
CO_2 [3]	10 mA, 60 mA(peak)	5,00	5,00E-02	0,00E+00	3,00E-01
LED1-R [15]	30 mA	2,50	7,50E-02	0,00E+00	7,50E-02
LED2-G [15]	30 mA	2,50	7,50E-02	0,00E+00	7,50E-02
Dual Supply [16]	1 uA	3,60	1,30E-05	1,30E-05	1,30E-05
Multicell Battery [5]	650 uA(idle), 15 uA(off), 80 uA(sleep)	5,00	3,25E-03	7,50E-05	3,25E-03
TMP/HUM [4]	220 uA	3,30	7,26E-04	0,00E+00	7,26E-04
FRAM [7]	100 uA(active), 3 uA(standby)	3,60	1,08E-05	0,00E+00	3,60E-04
MCU [1]	88 uA(run), 0,27 µA (Standby mode)	3,60	3,17E-04	9,72E-05	3,17E-04
OPZ [17]	0.45 uA	5,00	2,25E-06	2,25E-06	2,25E-06
SD convertor [18]	400 nA	5,00	2,00E-06	2,00E-06	2,00E-06
RTC [6]	60 nA	3,60	1,80E-07	1,80E-07	1,80E-07
		Celkový výkon	2,27E-01	1,99E-04	1,40E+00

Tabulka 3.1: Energetická náročnost jednotlivých komponent seřazena podle spotřeby (tabulka ve větším formátu v přílohách A.4)



Obrázek 3.4: Porovnání spotřeby podle scénářů

Dále jsem porovnal v grafu 3.4 všechny hlavní scénáře. Z grafu je patrné, že kdyby byly aktivované všechny periferie na svůj maximální potenciál, tak by výkon čidla dosahoval cca **1,45 W**. To odpovídá odběru necelých **380 mA**, viz. tabulka v příloze A.3. Při výpočtu, jak dlouho by vydrželo fungovat čidlo na obě baterie při nepřetržitém provozu, vyšla životnost baterie na necelých 26 hodin. Při zadaných hodnotách napětí **7,2 V**, výkonu **1,45 W** a kapacity **5,2 Ah**.

$$I_{ns} = \frac{P}{U} = \frac{1,4}{7,2} = 0,19A \tag{3.1}$$

3.1: Proud pro režim nejhorší možné spotřeby

$$t_{ns} = \frac{C}{I} = \frac{5,2}{0,194} = 26,80h \tag{3.2}$$

3.2: Čas pro režim nejhorší možné spotřeby

Proto jsem se rozhodl uspořit baterie zkrácením režimu Idle cca na dvě vteřiny a zbytek času stráví čidlo v hlubokém spánku a všechny periferie se odpojí od napájení. Měřící cyklus se opakuje pětkrát za hodinu, to tedy 480 vteřin strojového času za den. Chytré čidlo by mohlo fungovat pouze touto úsporou na obě baterie 201 dní, za předpokladu maximální možné spotřeby v době, kdy je čidlo vzbuzené.

Pro lepší přiblížení, jak dlouho bude čidlo fungovat na obě baterie budu předpokládat už využití režimu Idle a spánku. V Idle stavu bude čidlo denně **480 s**. Zbytek dne (**85920 s**) bude spát. Výkon v režimu Idle je **0,23 W** a ve spánku **1.99E-04 W**.

$$I_{Idle} = \frac{P}{U} = \frac{0,227}{7,2} = 31,53 \times 10^{-3}A \tag{3.3}$$

3.3: Proud režimu idle

$$I_{spanek} = \frac{P}{U} = \frac{1,99 \times 10^{-4}}{7,2} = 2,76 \times 10^{-5}A \tag{3.4}$$

3.4: Proud režimu spánek

Když vím, jak dlouho čidlo funguje v konkrétních režimech, tak mohu odvodit spotřebu za jeden den.

Z výpočtu 3.8 je patrné, že úsporná opatření by mohla prodloužit výdrž na baterie téměř na 3 roky.

Tabulka 3.2 zobrazuje procentuální zastoupení spotřeby energie podle řádů. Je patrné, že prvky z grafu 3.6 jsou největším problémem. V režimu Idle mají procentuálně největší zastoupení.

$$C_{Idle} = I * t = 31,53 \times 10^{-3} * 0,133 = 4,19 \times 10^{-3} Ah \quad (3.5)$$

3.5: Spotřeba režimu idle za jeden den

$$C_{spanek} = I * t = 2,76 \times 10^{-5} * 23,86 = 6,59 \times 10^{-4} Ah \quad (3.6)$$

3.6: Spotřeba pro režim spánek za jeden den

$$C_den = C_{Idle} + C_{spanek} = 4,19 \times 10^{-3} + 6,59 \times 10^{-4} = 4,85 \times 10^{-3} Ah/den \quad (3.7)$$

3.7: Celková spotřeba za jeden den

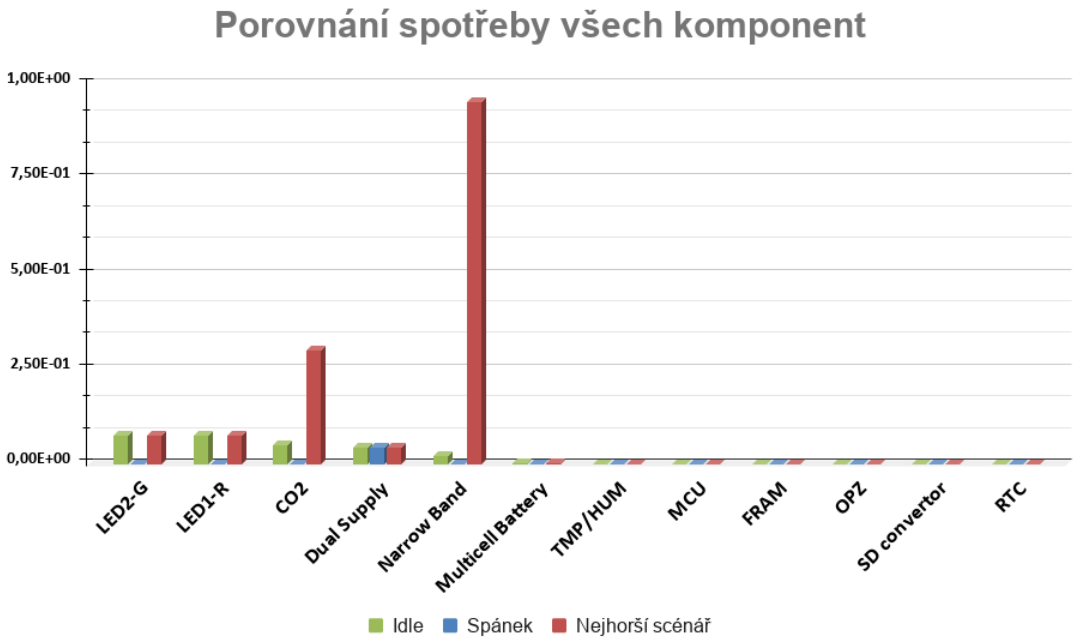
$$t_{celkem} = \frac{C_{baterie}}{C_{den}} = \frac{5,2}{4,85 \times 10^{-3}} = 1072 dni = 2,94 roku \quad (3.8)$$

3.8: Výsledná doba výdrže baterie

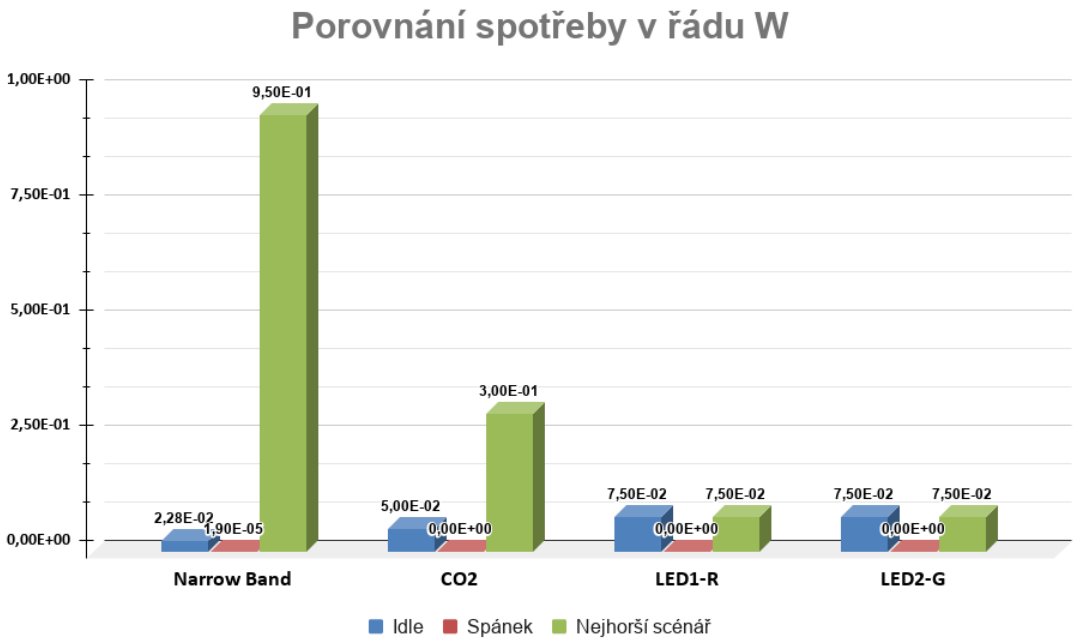
Řády	**Idle**	**Spánek**	**Nejhorší scénář**
W	98,10%	9,54%	99,67%
mW	1,90%	86,43%	0,33%
uW	0,00354%	4,03052%	0,00057%
Suma	100,00%	100,00%	100,00%

Tabulka 3.2: Srovnání spotřeby podle řádů

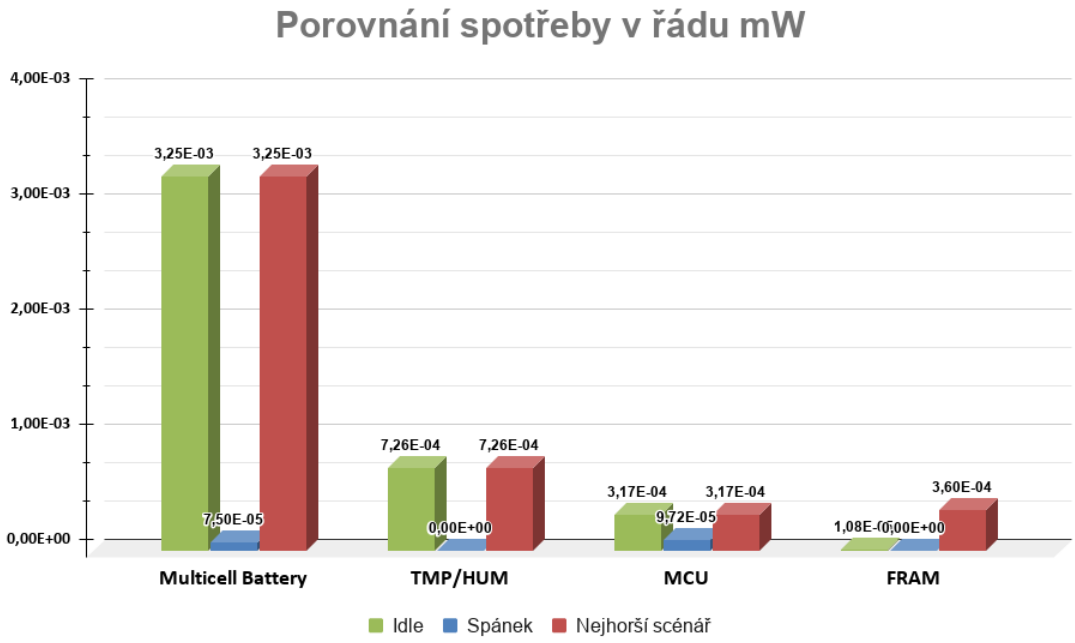
Na grafu 3.5 je vidět porovnání všech aktivním prvků. Toto zobrazení ale není úplně přesné, protože se zde porovnávají různé řády spotřeby. Pro lepší vizualizaci jednotlivých členů jsem v nadcházejících grafech 3.6, 3.7 a 3.8 porovnal prvky se spotřebou ve stejném řádu.



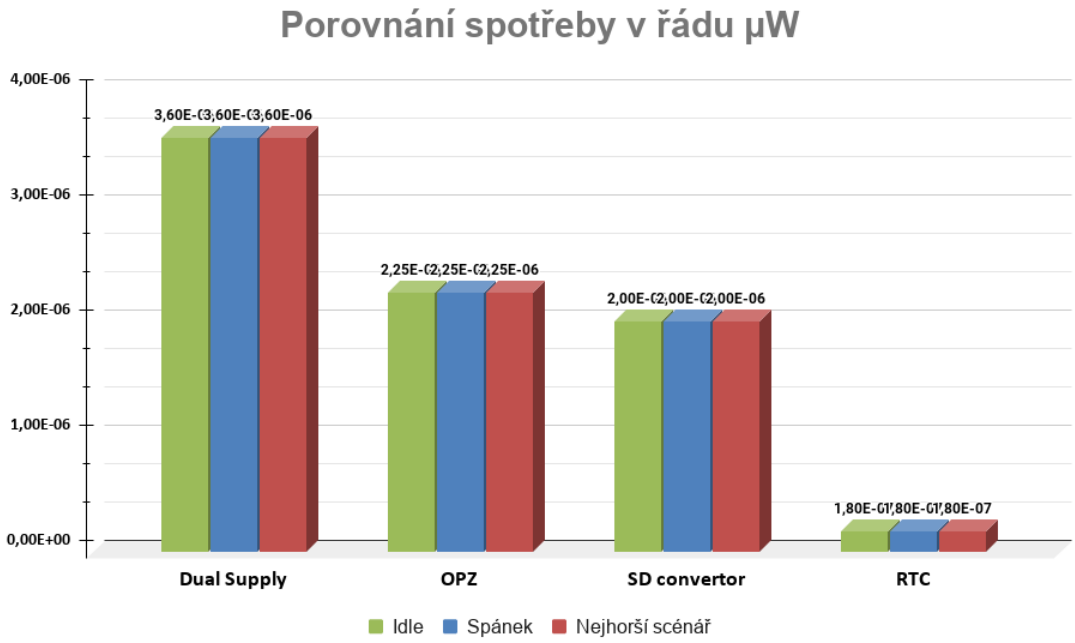
Obrázek 3.5: Porovnání spotřeby všech komponent najednou



Obrázek 3.6: Porovnání spotřeby v řádu W



Obrázek 3.7: Porovnání spotřeby v řádu mW



Obrázek 3.8: Porovnání spotřeby v řádu μW

4 Implementační detaily projektu

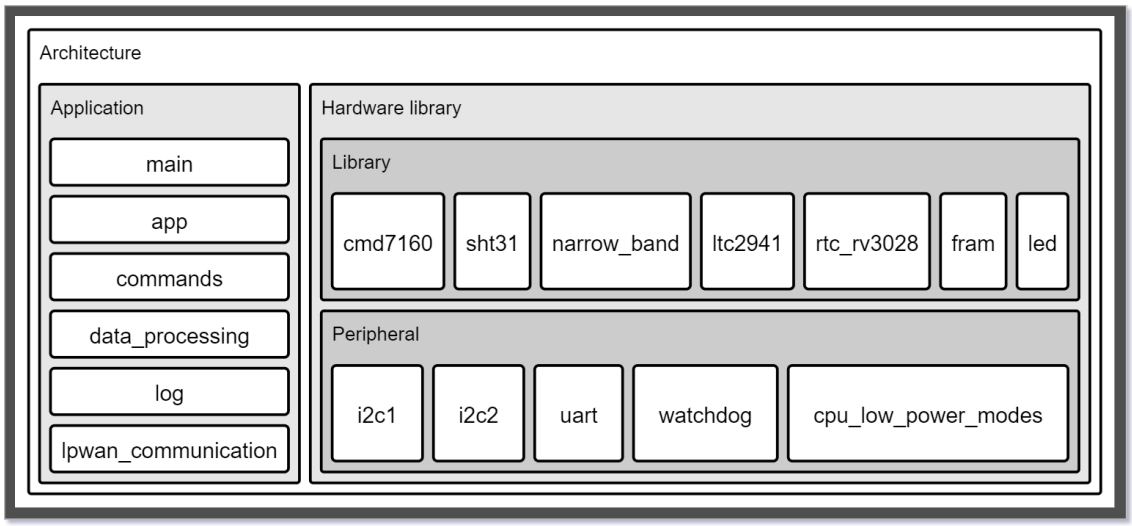
V této kapitole přiblížím postupy, které jsem pro dokončení práce využil. Zmíním problémy, jaké v průběhu vytváření softwaru pro chytré čidlo nastaly. Věnuji se zvlášť části, ve které pracuji na vývoji firmwaru pro chytré čidlo, a části, ve které vytvářím webové rozhraní pro práci s naměřenými daty.

4.1 Tvorba firmwaru

V první fázi vývoje, když jsem studoval vlastnosti dodané desky, jsem používal standardní knihovny HAL od výrobce MCU. Využil jsem i dostupných nástrojů viz seznam:

- Atollic True Studio
- STM32CubeMX

Vytvořil jsem základní projekt pro procesor *STM32L051* v programu *STM32CubeMX*. Tento nástroj obsahuje vše potřebné pro první nastavení vývojového prostředí včetně linkeru a kompilátoru. Více se tomuto nástroji věnuji v podkapitole 4.1.2.



Obrázek 4.1: Diagram firmwaru

Výrobce chytrého čidla měl hardwarově oddělené všechny komponenty od napájení, takže při oživování jsem se vždy mohl zaměřit pouze na jednu věc a ta, co začala fungovat, tak jsem mohl postoupit k další části. Tímto postupem jsem se vyvaroval zbytečných chyb a mohl tak vyloučit nefunkčnost periferie z důvodu kombinace více chyb naráz.

Aby byl firmware platformově nezávislý, musel jsem oddělit kód aplikační od kódu, který pracuje s periferiemi. Na obrázku 4.1 je znázorněno, jak jsem oddělil aplikační část kódu od knihoven. Hardwarové knihovny jsem ještě rozdělil podle stupně abstrakce.

Aplikační část obsahuje **main**, který se spustí a provede čidlo bootovací sekvencí. Pokud vše proběhne v pořádku, spustí se aplikace v části **app**. Když ale dojde při bootovaní k nějaké chybě, bude čidlo čekat v režimu na zásah uživatele. Ostatní části v aplikaci souvisí se zpracováním naměřených dat v části **data processing**, zobrazováním stavu čidla pomocí logů přes sběrnici UART, příkazy pro komunikaci čidla s cloudem v **lpwan communication** a jiné.

Význam knihoven:

- **cmd7160** - čte koncentrace oxidu uhličitého
- **sht31** - vyčítá teploty a vlhkosti, pracuje se všemi registry senzoru
- **narrow band** - komunikuje po síti
- **ltc2941** - měří stav baterie zařízení
- **rtc rv3028** - pracuje s obvodem reálného času, nastaví přerušení
- **fram** - ukládá dočasných naměřených dat
- **led** - pracuje se základními funkcemi LED diod

Funkční knihovny, které pracují s protokoly pro komunikaci s čidly, jsou separovány od knihoven, které pracují s registry procesoru.

V další fázi vývoje jsem musel opustit použití knihoven HAL a použít CMSIS metodu. To znamenalo, že všechny knihovny napsané pro HAL, přepsat na práci přímo s registry procesoru. Znamenalo to především změnit části, které jsou zobrazené na obrázku 4.1 v části *Periheral*.

4.1.1 Atollic True Studio

Atollic True Studio je IDE, které je založené na Eclipse. Mezi důležité vlastnosti tohoto IDE patří:

- Analyzér pádu CPU
- Živý vhled do globálních proměnných
- Pokročilé procházení operací v čase
- Podpora ladění RTOS operačních systému
- Více-vláknové a více-deskové ladění

Velice užitečná je také možnost ladění programu přímo za běhu na MCU. V mém případě jsem pro ladění a nahrávání firmwaru používal *ST-Link V2*. Lze ovšem využít pro ladění i *J-link*. Toto IDE jsem použil, protože s tímto prostředím mám zkušenosti a používám ho i pro jiné projekty.

4.1.2 STM32CubeMX

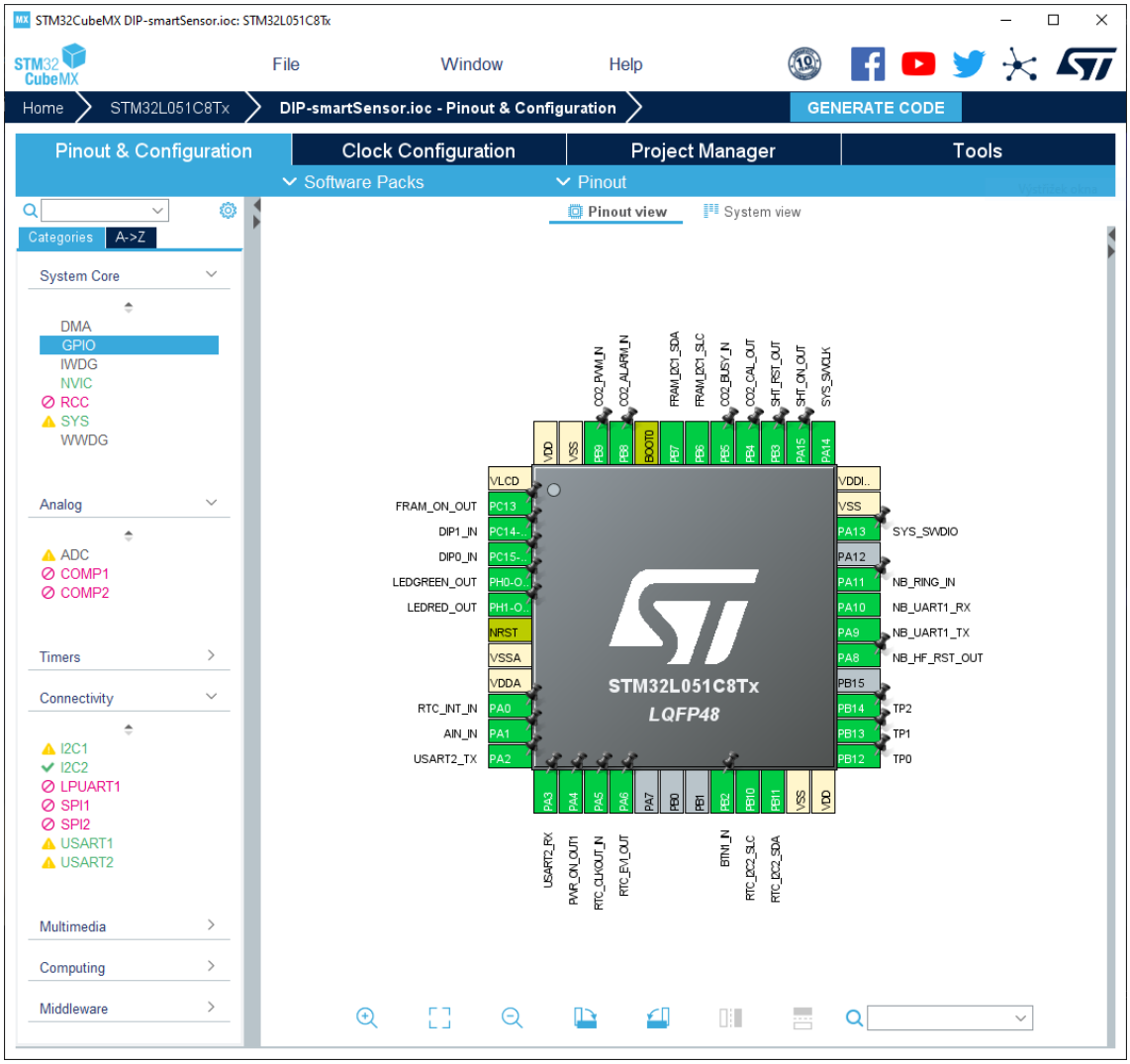
Nástroj STM32CubeMX je komplexní program, který obsahuje celou paletu procesorů od firmy ST. Lze si tu vybrat konkrétní procesor na konfiguraci projektu podle potřeby. Například jde zvolit vlastnosti jednotlivých pinů procesoru a určit tak jejich funkci.

Tento program mi velice pomohl při konfiguraci GPIO vstupů, nastavení parametrů sběrnic a interuptu. Program obsahuje veškeré dostupné materiály o daném typu procesoru od dokumentace, přes aplikační manuál, až po erratu. Jedna z důležitých vlastností programu je vizualizace zvolených pinů v projektu. To pomáhá lepší orientaci a ve finále program vygeneruje nastavený projekt přímo pro dané IDE, které si zvolíte.

Lze vytvořit projekt pro tato IDE:

- EWARN
- MDK-ARM
- SW4STM32
- TrueSTUDIO
- STM32CubeIDE

Na obrázku 4.2 je pohled na program STM32CubeMX. Konkrétně je vidět nastavení tohoto projektu. Velice užitečné je využití možnosti přejmenovat GPIO vstupy podle jejich skutečné funkce, je to lepší pro orientaci při práci v IDE.



Obrázek 4.2: Pohled z programu STM32CubeMX

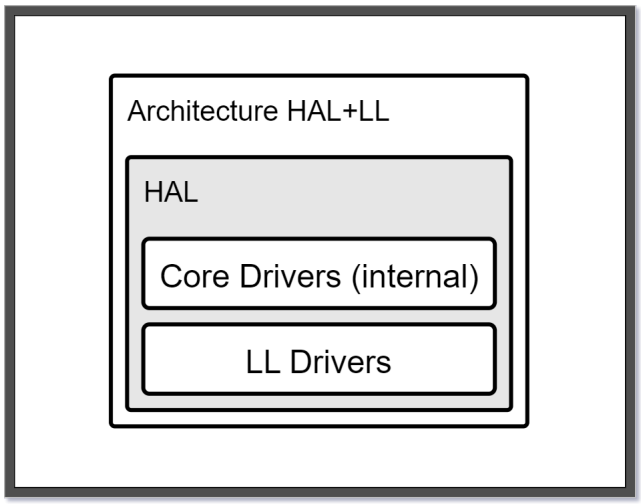
4.2 FW knihovny projektu

V této sekci se věnuji popisu pouze knihoven, které jsem pro tento projekt navrhl. Ostatní knihovny jsem, po vyloučení knihoven HAL, dostal od zákazníka k dispozici a na jejich vývoji jsem se nepodílel. Knihovny od zákazníka jsem pouze aplikoval v hlavní části programu. Lehce se zmíním o knihovnách HAL od firmy ST.

4.2.1 Knihovny HAL

HAL knihovna je další abstraktní vrstva nad LL knihovnami, které pracují s registry MCU. Jsou vhodné pro začátečníka, který má povědomí o základních funkcionalitách daných částí MCU, ale nepotřebuje znát všechny vazby na registry procesoru. Výhodou těchto knihoven je rychlost vývoje na dané platformě, dobře zpracovaná dokumentace a možnost automatického nastavení projektu, jak jsem popisoval v ka-

pitole 4.1.2. Nevýhodou použití těchto knihoven velikost binárních souborů po kompilaci.



Obrázek 4.3: Ukázka architektury aplikace s HAL knihovnou

4.2.2 Knihovna pro ladění na UART

Tato knihovna sloužila v prvotní fázi vývoje aplikace pro výpis do konzole přes sběrnici UART. Abych mohl lépe sledovat chování programu, navrhl jsem si univerzální funkce založené na knihovnách HAL. Tato knihovna ovšem v momentě odebrání knihoven HAL již postrádala hlubší smysl, proto jsem se rozhodl ji nepřepisovat.

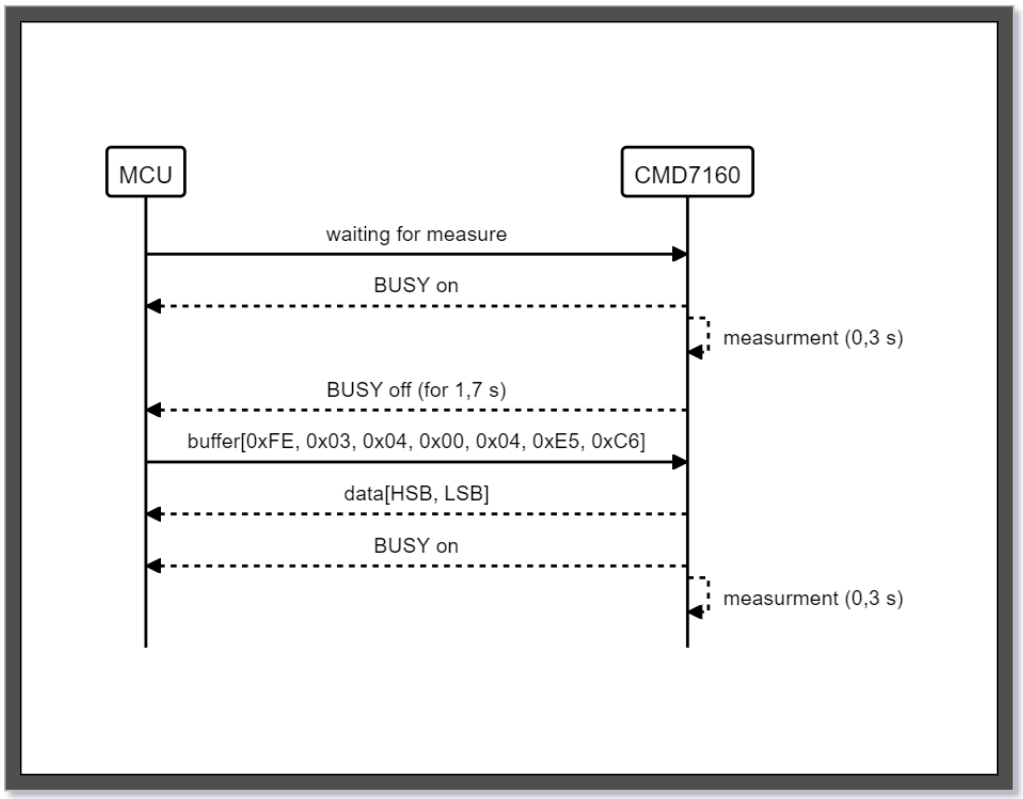
Seznam funkcí knihovny *log.h*:

- **log_print_string(UART_HandleTypeDef huart1, void *string)** - funkce vypíše na UART libovolní řetězec znaků
- **log_print_value_float(UART_HandleTypeDef huart1, float value, log_value_type type)** - funkce vypíše změřenou hodnotu s desetinou čárkou
- **log_print_value(UART_HandleTypeDef huart1, uint16_t value, log_value_type type)** - funkce vypíše změřenou hodnotu bez desetinou čárkou
- **log_print_i2c_device_addr (UART_HandleTypeDef huart1, I2C_HandleTypeDef hi2c)** - funkce vypíše všechny adresy periferií na sběrnici I^2C

Výpisové funkce využívaly jednoduchého enumu, kterým jsem přepínal, jaký text se má k vložené hodnotě doplnit. Musel jsem rozdělit výpis hodnot s a bez desetinné čárky. Poslední funkce sloužila pro výpis všech aktivních adres na sběrnici I^2C. Podle toho, jaký I^2C handler jsem funkci předal, se vypsaly adresy připojených periferií.

4.2.3 Knihovna CMD7160

Senzor na měření koncentrace oxidu uhličitého komunikuje po sběrnici UART. V této knihovně jsem se věnoval jenom čtení měřených hodnot, ostatní funkce nebyly potřebné. Alarm překročení CO_2 nad hodnotu 1000 ppm se snímá přímo na pinu 3 (*Alarm*). V aplikační vrstvě projektu se dále u tohoto čidla pracuje s pinem 8 (*Busy*), který určuje dobu, kdy je možné číst registry a kdy se měří. Knihovna cmd7160 pracuje s knihovnou *uart.h*.



Obrázek 4.4: Diagram komunikace s CMD7160

Seznam funkcí knihovny *cmd7160.h*:

- **cmd_init()** - spustí napájení a inicializuje uart sběrnici
- **cmd_get_concentration(uint16_t *value)** - přečte hodnotu o koncentraci CO_2

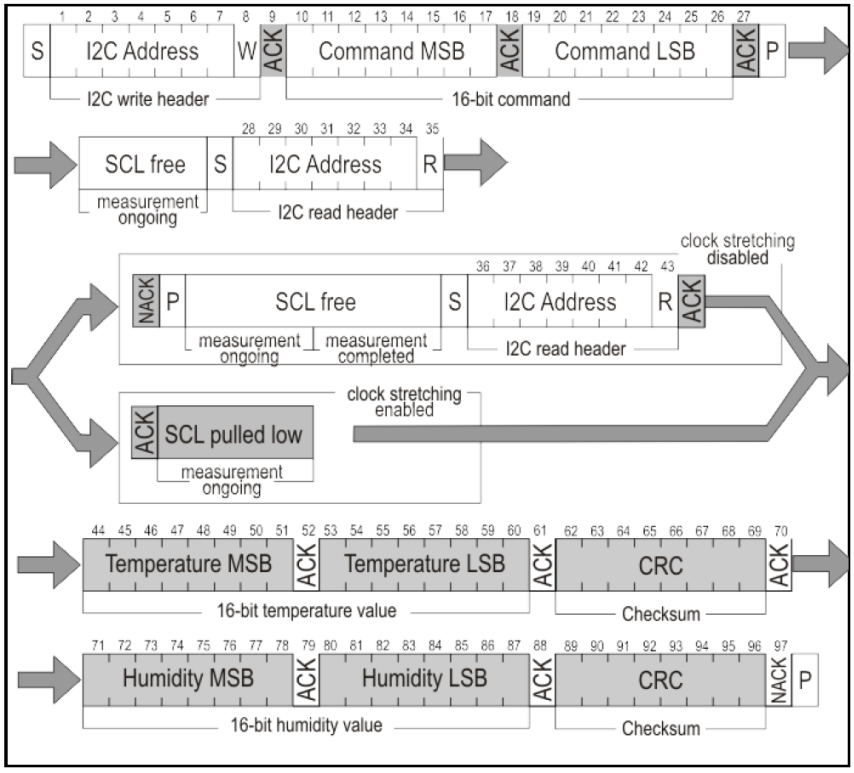
4.2.4 Knihovna SHT31

Čidlo na měření teploty a vlhkosti komunikuje po sběrnici I^2C. Knihovna pro měření teploty a vlhkosti obsahuje abstraktní vrstvu pro inicializaci čidla a seznam 6-ti funkcí, které čtou nebo pracují s datovými registry.

Seznam funkcí knihovny *sht31.h*:

- **sht31_init()** - spustí napájení a přepne reset pin do logické jedničky
- **sht31_periodic()** - spustí periodické měření
- **sht31_break()** - vypne periodické měření
- **sht31_soft_reset()** - resetuje bezpečně čidlo bez odpojení napájení
- **sht31_heater_on()** - zapne nahřívání
- **sht31_heater_off()** - vypne nahřívání
- **sht31_status_register()** - vyčte stav čidla
- **sht31_clear_status_register()** - vymaže status registr
- **sht31_get_measure_value(uint16_t* temperature, uint16_t humidity)** - vyčte teplotu a vlhkost

Původně napsaná knihovna pracovala s vrstvou HAL, kde se využívaly funkce čtení a zápisu po I²C sběrnici. Po odstranění knihoven HAL z projektu jsem musel pro komunikaci s I²C sběrnicí napsat knihovnu sám. Pro komunikaci jsem vytvořil knihovnu *i2c1.h*, ve které knihovně pracují přímo s registry MCU.



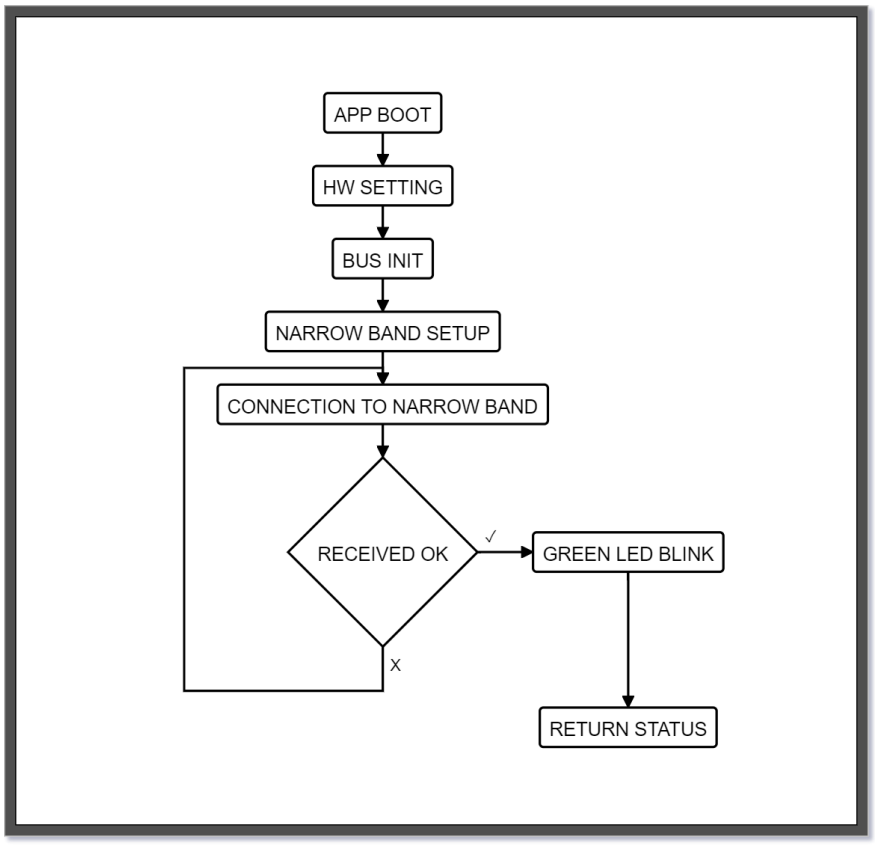
Obrázek 4.5: Diagram komunikace s SHT31 [4]

Seznam funkcí knihovny *I²C1.h*:

- **i2c1_start(int cnt)** - vytvoří start bit
- **i2c1_stop(void)** - vytvoří stop bit
- **i2c1_start_write(int cnt)** - předá čidlu informaci o zápisu
- **i2c1_start_read(int cnt)** - předá čidlu informaci o čtení
- **i2c1_read_sht(uint8_t *rx_data)** - čte naměřená data z registrů
- **i2c1_shift(uint8_t address, const void *tx_data, uint8_t tx_count, uint8_t *rx_data, uint8_t rx_count)** - čte data z I²C sběrnice

Pro běžnou komunikaci s periferiemi lze použít univerzální funkci *i2c1_shift(...)*. U čtení dat ze senzoru teploty a vlhkosti jsem ale narazil s touto univerzální funkcí na problémy. Proto jsem pomocí měření logickou sondou vyladil čtení na míru. Na obrázcích v příloze viz. A.6 a A.7 je zobrazena ukázka měření z logické sondy.

4.2.5 Aplikační postupy



Obrázek 4.6: Diagram bootovací sekvence

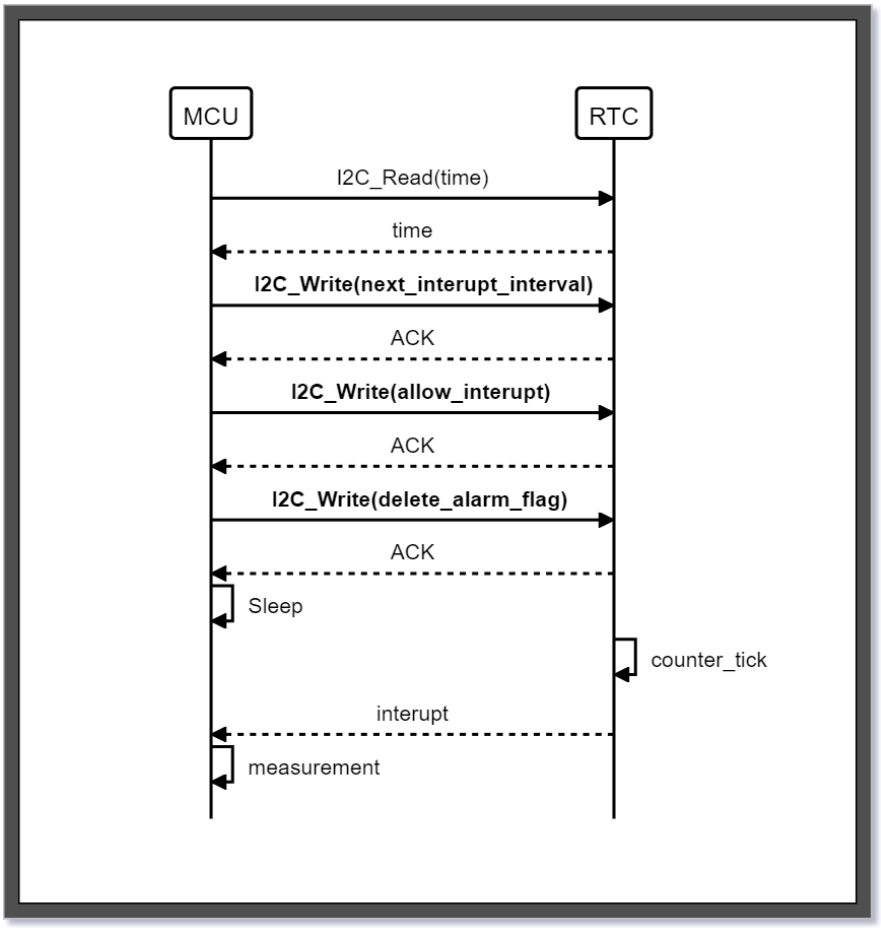
Bootovací sekvence

Před tím, než se čidlo přepne do aplikace, provádí se bootovací sekvence. V této části se spustí základní hardwarové nastavení **GPIO** podle jejich funkce, zapnou se jejich hodiny, inicializují se sběrnice pro komunikaci a provede se přihlášení do NB-IOT sítě. Na obrázku 4.6 popisuji startovací sekvenci, v průběhu které čidlo zapíná periférie.

Nastavení přerušení v RTC

V momentě, kdy je MCU uspané a všechny periferie jsou odpojené od napájení, zbývá zapnutý modul reálného času. Na diagramu 4.1 popisuji sled událostí, které provází právě nastavení dalšího probuzení čidla. Procesor si napřed vyčte aktuální hodnotu času, přičte k ní časovou konstantu, která nařídí RTC modulu vyslat přerušení, až interval pomine. Dále se povolí funkce přerušení. Je také nutné vynulovat příznaky, které si pamatují předchozí přerušení.

Pak je již vše nastaveno a MCU se probere hlubokého spánku po nastaveném intervalu.



Obrázek 4.7: Ukázka komunikace mezi MCU a RTC při nastavení IRQ

4.3 Tvorba webové aplikace

V této sekci se budu věnovat tvorbě webové aplikace pro vizualizaci dat. Tato část původně nebyla součástí zadání této diplomové práce, ale protože jsem pro zákazníka webovou aplikaci vytvořil, zmíním tu alespoň postup, který jsem pro tvorbu použil.

4.3.1 Framework vue.js

Jedná se o vývojářský nástroj pro snadné psaní webové aplikace. Použití tohoto prostředí bylo na přání zákazníka. Výhodou užití těchto frameworků spočívá v hodně snadném a rychlém nasazení. Po instalaci tohoto nástroje lze okamžitě spustit na lokální síti webovou aplikaci, která je dostupná všem uživatelům na specifickém portu IP adresy použitého PC. Každá další změna se po uložení okamžitě projeví v aplikace. Další výhodou je využití balíčkovacího systému **node.js**, ze kterého lze velice snadno stáhnout potřebné knihovny pro projekt. Každá nová instalace knihovny z třetí strany se zaznamenává do souboru *package.json*. Díky tomu není třeba na **GIT** ukládat celý projekt se všemi knihovnami ve složce *node modules*, ale po novém stažení projektu z verzovacího systému stačí spustit příkaz na spuštění webové aplikace. Při spouštění se všechny nutné závislosti v dané verzi z balíčkovacího systému automaticky stáhnou.

Čas	Teplota (°C)	Vlhkost (%)	CO2 (ppm)
2020-06-20 at 10:56	9472	22528	659
2020-06-20 at 10:59	9472	22528	652
2020-06-20 at 11:04	9472	22528	657
2020-06-20 at 11:09	9472	22528	624
2020-06-20 at 11:14	9472	22528	642
2020-06-20 at 11:19	9472	22528	687
2020-06-20 at 11:24	9472	22528	701
2020-06-20 at 11:29	9472	22528	724

Obrázek 4.8: Ukázka webové aplikace, v příloze A.8 najdete podrobnější zobrazení

4.3.2 Použité knihovny

Abych mohl pracovat s naměřenými daty, bylo potřeba se webovou aplikací napojit na server a konkrétní data stáhnout. Pro vytváření HTTP dotazů jsem použil dostupnou knihovnu **axios**. Abych docílil dynamické změny grafiky webu, použil jsem balíček **Vuetify**. Jedná se o jednoduché rozhraní, které za mě řeší kompatibilitu zobrazení pro webové prohlížeče, tablety a mobilní zařízení. Vzhled webové aplikace jsem navrhoval podle pouček **material design**. Pro vizualizaci dat v grafech jsem použil **vue chartjs**. Vytvořil jsem jednoduchý čárový graf pro zobrazení dat měřených veličin.

Data z cloudu se načítají do jednoho JSON souboru. Ten se pak využívá pro rozčlenění dat do datového modelu.

Účelem webové aplikace je především demonstrace funkce čidla. Nejdůležitější byla implementace ve vue.js rozhraní. Díky tomu může zákazník snadno implementovat datový rozdělovač podle navrženého datového modelu do svého budoucího webového projektu.

4.4 Využití operačního systému a proč ho zákazník nakonec nechtěl

Operačním systémům jsem se věnoval v kapitole 2.2. Využitím operačního systému by se dal efektivněji využít čas v měřící fázi. Prostoje procesoru v době, kdy čeká MCU na odpověď senzorů, by mohly být využity pro jiné dotazy. To by šlo realizovat rozčlenění měřící úlohy pro jednotlivé měřící fáze do dílčích úloh. Teoreticky by se tím dal ušetřit čas, který by se mohl projevit v dlouhodobé spotřebě elektrické energie.

Důvody, proč zákazník nechtěl operační systém:

- Jde pouze o datalogger
- Ostatní výrobky, které vyrábí, OS nemají
- Kapacita pro udržování OS
- Obavy ze spotřeby baterie

Po vzájemné dohodě se zákazníkem nebyl vývoj aplikace s operačním systémem nakonec realizován.

4.5 Datový model pro odesílání dat

Když jsem řešil způsob, jak budu data skladovat do paměti FRAM, tak bylo nejprve zapotřebí určit, jaká data je nezbytné ukládat. Proto jsem vytvořil jednoduchou tabulku 4.1 se všemi důležitými daty, která je potřeba uchovávat a odesílat.

Hodnota	Jednotka	Formát	Velikost
Čas	Timestamp	1583319920	4 B
Teplota	°C	12,52	2 B
Vlhkost	%	38,59	2 B
CO2	ppm	1024	2 B
Napětí	V	7,64	2 B
Proud	mA	337,3	2 B
Náboj	C	18720	2 B
Alarm	16 různých alarmů pomocí masky	0/1	2 B
		Suma	18 B

Tabulka 4.1: Datový model

V tabulce 4.1 jsou vidět základní veličiny, které čidlo normálně měří, časové razítko a v poslední řadě 2 B místa pro masku alarmů. Použil jsem pro jistotu na alarmy byty dva, aby byl v budoucnu prostor pro rozšiřování. Nultý bit obsahuje informaci o překročení koncentrace CO_2. Tento alarm kopíruje stav na GPIO vstupu, kam je připojen alarm přímo z CMD7160.

A[16]	...	A[6]	A[5]	A[4]
-	-	Current Alarm	Accumulated charge alarm	Temp Alarm
A[3]	**A[3]**	**A[2]**	**A[1]**	**A[0]**
Charge Alarm HSB	Charge Alarm LSB	Voltage Alarm	Under voltage alarm	CO_2 Alarm

Tabulka 4.2: Využití místa pro alarmy čidla

Ostatní alarmy jsou výstupy z culombmeteru LTC2943. V budoucnu lze doplnit do toho prostoru i další informace o stavu čidla.

Po analýze dat bylo potřeba zjistit, kolik dat je čidlo schopno uložit v případě, že dojde k výpadku sítě. Z jednoho cyklu měření nám vychází **18 B** dat. S frekvencí měření 5 krát za hodinu to činí **216 B** za hodinu, z toho plyne **5184 B** za den.

K dispozici máme 16 Kb v FRAM paměti. To činí 2 KB paměti. Za předpokladu, že by čidlo ztratilo konektivitu, tak by mohlo uchovat přes 9 hodin záznamů.

$$t_{16Kbit} = \frac{FRAM[B]}{B/hodinu} = \frac{2000}{216} = 9,26h \tag{4.1}$$

4.1: Výpočet možných dnů měření pouze v případě uchovávání dat na FRAM paměť

V případě potřeby delších záznamů lze FRAM vyměnit za větší model FM24CL64B-G s kapacitou 64 Kb paměti a množství záznamů prodloužit až na 37 hodin.

$$t_{64Kbit} = \frac{FRAM[B]}{B/hodinu} = \frac{8000}{216} = 37,04h \tag{4.2}$$

4.2: Výpočet možných dnů měření pouze v případě uchovávání dat na FRAM 16 Kb paměťi

Pro odeslání všech naměřených dat za jednu hodinu stačí jeden packet o velikosti 256 B, který se pošle do NB sítě. Díky tomu dojde i k cenové úspoře provozu a úspoře baterie, neboť se neposílá každé měření rovnou na cloud.

Závěr

V rámci této diplomové práce jsem se zabýval vývojem firmwaru pro procesor STM32L051 a návrhem webové aplikace pro vizualizaci dat. Pro realizaci projektu bylo zapotřebí se nejprve důkladně seznámit s dostupným hardwarem, který mi pro vývoj aplikace poskytla firma VisionQ.

Postupně jsem oživoval základní desku a navrhl knihovny pro výpis dat do konzole a měření veličin teploty, vlhkosti a koncentrace oxidu uhličitého. V prvotní fázi vývoje jsem využil nástrojů dostupných od výrobce procesoru.

Pomocí programu STM32CubeMX jsem nakonfiguroval veškeré vlastnosti potřebné pro komunikaci s periferiemi. Tento nástroj pak konfiguraci přegeneroval na projekt do IDE. Díky těmto nástrojům jsem byl schopen rychle spustit všechny důležité funkce čidla. Pro realizaci jsem od zákazníka dostal některé knihovny, které již měl vyvinuté. Jednalo se například o knihovny starající se o komunikaci se sítí Narrow Band, konfigurací obvodu reálného času, ukládání dat do FRAM paměti nebo spojení po sběrnici UART. Mým dílem ve vývoji byly knihovny pro čidla SHT31, CMD7160 a aplikační vrstva, která využívala všech dostupných knihoven.

Nakonec jsem prototyp aplikace založený na knihovnách HAL přepsal na CMSIS

Další částí mé práce byl vývoj webové aplikace. Od zákazníka jsem dostal zadáno, že mám využít **vue.js** frameworku, aby se v budoucnu mohla moje webová aplikace integrovat do jejich produkčního webu. Pro stažení dat z cloudového úložiště jsem dostal od zadavatele REST API, které jsem měl integrovat. Navrhl jsem jednoduché prostředí, kde se stažená data rozčlení z binárního formátu do datové struktury a vykreslí se měřené veličiny v grafech a v tabulce.

Funkci čidla jsem krátkodobě testoval v domácích podmínkách, abych ověřil opakovatelné buzení čidla ze spánku pomocí obvodu reálného času.

V budoucnu by šlo v práci pokračovat a přidat možnost nastavení čidla na základě požadavku z webové aplikace. Například nastavení frekvence měření podle potřeby uživatele. Tím by se dala regulovat spotřeba energie podle potřeby.

Když shrnu body zadání, tak se mi podařilo realizovat funkční firmware pro čidlo od firmy VisionQ. Funguje celý systém od naměření hodnot přes uložení do cloudového úložiště až po vizualizaci dat. Navrhl jsem aplikaci tak, aby byl, co nejvíce platformově nezávislá.

Bibliografie

1. *STM32L051x8: Datasheet - production data* [online]. 3. vyd. STMicroelectronics, 2014 [cit. 2021-04-19]. Dostupné z: https://www.st.com/en/microcontrollers-microprocessors/stm32l051c6.html.
2. TICHÝ, Bc. Jan. *Inteligentní systém pro anonymní detekci počtu osob v místnosti*. Liberec, 2018. Bakalářská práce. Technická univerzita v Liberci.
3. *TECHNICAL INFORMATION FOR CDM7160* [online]. 1. vyd. JAPAN: Figaro Engineering Inc., 2016 [cit. 2021-04-20]. Dostupné z: https://cdn.sos.sk/productdata/52/08/dadc87c9/cdm7160.pdf.
4. *Datasheet SHT3x-DIS: Humidity and Temperature Sensor* [online]. 6. vyd. Switzerland: Sensirion, 2019 [cit. 2021-04-20]. Dostupné z: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/2_Humidity_Sensors/Datasheets/Sensirion_Humidity_Sensors_SHT3x_Datasheet_digital.pdf.
5. *LTC2943: Multicell Battery Gas Gauge with Temperature, Voltage and Current Measurement* [online]. A. Milpitas: Linear Technology Corporation, 2014 [cit. 2021-04-20]. Dostupné z: https://www.analog.com/en/products/ltc2943.html%5C#product-reference.
6. *RV-3028-C7: Extreme Low Power RTC Module* [online]. 1.1. vyd. Switzerland: Micro Crystal AG, 2019 [cit. 2021-04-20]. Dostupné z: https://www.microcrystal.com/en/products/real-time-clock-rtc-modules/rv-3028-c7/.
7. *FM24CL16B: 16-Kbit (2K × 8) Serial (I2C) F-RAM* [online]. 1. vyd. San Jose: Cypress Semiconductor Corporation, 2018 [cit. 2021-04-20]. Dostupné z: https://www.cypress.com/file/136491/download.
8. *LS 14500: Primary lithium battery* [online]. Bagnolet - France: Saft Specialty Battery Group, 2009 [cit. 2021-05-05]. Dostupné z: https://www.arrow.com/en/products/ls-14500/dantona-industries?utm_term=backorder%5C&utm_currency=%5C&utm_campaign=arrow_findchips_2021_Americas%5C&utm_medium=aggregator%5C&utm_source=findchips%5C&utm_content=inv_listing.

9. WANG, Hayden; WU, Evan. *Application Design Guide: BC95&BC95-G&BC68* [online]. 1.1. vyd. China: Quectel Wireless Solutions Co., Ltd., 2018 [cit. 2021-04-20]. Dostupné z: https://www.scribd.com/document/396537594/Quectel-BC95-BC95-G-BC68-Low-Power-Design-Guide-V1-1-1.
10. *Https://www.tcele.cz/* [online] [cit. 2021-05-07]. Dostupné z: https://www.tcele.cz/detektory-alarmy-a-spinace-co2/co2-guard-10-senzor-pro-mereni-co2-ve-skolach-kancelarich-a-domacnosti.
11. *Senzor kvality ovzduší pro LoRa IoT sítě - CO2, Teplota a vlhkost* [online] [cit. 2021-05-07]. Dostupné z: https://www.alternetivo.cz/senzor-kvality-ovzdusi-pro-lora-iot-site-co2-teplota-a-vlhkost_d57613.html.
12. *Kvalita vzduchu v interiéru* [online] [cit. 2021-05-07]. Dostupné z: https://www.unipi.technology/cs/produkty/kvalita-vzduchu-v-interieru-355?.
13. *IoT bezdrátový snímač teploty a CO2 s výstupem do sítě Sigfox* [online] [cit. 2021-05-07]. Dostupné z: https://www.cometsystem.cz/produkty/iot-bezdratovy-snimac-teploty-a-co2-s-vystupem-do-site-sigfox/reg-w8810.
14. MEDU, Harsha. *Energy Comparison of Cypress F-RAM and EEPROM* [online]. 001-94664 Rev. *A. San Jose: Cypress Semiconductor, 2017 [cit. 2021-05-02]. Dostupné z: https://www.cypress.com/file/46746/download.
15. *KP-2012SRC-PRV: Datasheet* [online]. 12. vyd. Kingbright, 2010 [cit. 2021-04-20]. Dostupné z: http://www.farnell.com/datasheets/622356.pdf.
16. *SN74AVC2T45 2-Bit, Dual Supply, Bus Transceiver With Configurable Level-Shifting and Translation* [online]. L. vyd. Texas: Texas Instruments, 2017 [cit. 2021-04-20]. Dostupné z: https://www.ti.com/lit/ds/symlink/sn74avc2t45.pdf?ts=1618852581380%5C&ref_url=https%5C%253A%5C%252F%5C%252Fwww.google.com%5C%252F.
17. *MCP6441: Datasheet* [online]. 1. vyd. U.S.A.: Microchip Technology Inc., 2010 [cit. 2021-04-20]. ISBN 978-1-60932-513-8. Dostupné z: https://www.tme.eu/Document/1dd9e095ebb0e71f1628687389ff5c40/mcp6441.pdf.
18. *TPS62745: TPS62745 Dual-cell Ultra Low IQ Step Down Converter for Low Power Wireless Applications* [online]. A. Texas: Texas Instruments, 2015 [cit. 2021-04-20]. Dostupné z: https://www.ti.com/lit/ds/symlink/tps62745.pdf.

A Přílohy

A.1 Tabulky

Typ HW	Odběr [A]	Napětí [V]	Spotřeba [W]
CO_2	1,00E-02	5,0	5,00E-02
Dual Supply, Bus Transceiver	3,60E-06	3,6	1,30E-05
FRAM	3,00E-06	3,6	1,08E-05
LED1-R	3,00E-02	2,5	7,50E-02
LED2-G	3,00E-02	2,5	7,50E-02
MCU	8,80E-05	3,6	3,17E-04
Multicell Battery Gas Gauge	6,50E-04	5,0	3,25E-03
Narrow Band	6,00E-03	3,8	2,28E-02
Operační zesilovač	4,50E-07	5,0	2,25E-06
RTC	6,00E-08	3,0	1,80E-07
Stepdown convertor	4,00E-07	5,0	2,00E-06
TMP/HUM	2,20E-04	3,3	7,26E-04
Celkem	**7,70E-02**		**2,27E-01**

Tabulka A.1: Spotřeba v režimu Idle

Typ HW	Odběr [A]	Napětí [V]	Spotřeba [W]
CO_2	0,00E+00	5,0	0,00E+00
Dual Supply, Bus Transceiver	3,60E-06	3,6	1,30E-05
FRAM	0,00E+00	3,6	0,00E+00
LED1-R	0,00E+00	2,5	0,00E+00
LED2-G	0,00E+00	2,5	0,00E+00
MCU	2,70E-05	3,6	9,72E-05
Multicell Battery Gas Gauge	1,50E-05	5,0	7,50E-05
Narrow Band	5,00E-06	3,8	1,90E-05
Operační zesilovač	4,50E-07	5,0	2,25E-06
RTC	6,00E-08	3,0	1,80E-07
Stepdown convertor	4,00E-07	5,0	2,00E-06
TMP/HUM	0,00E+00	3,3	0,00E+00
Celkem	**5,15E-05**		**2,09E-04**

Tabulka A.2: Spotřeba v režimu Spánek

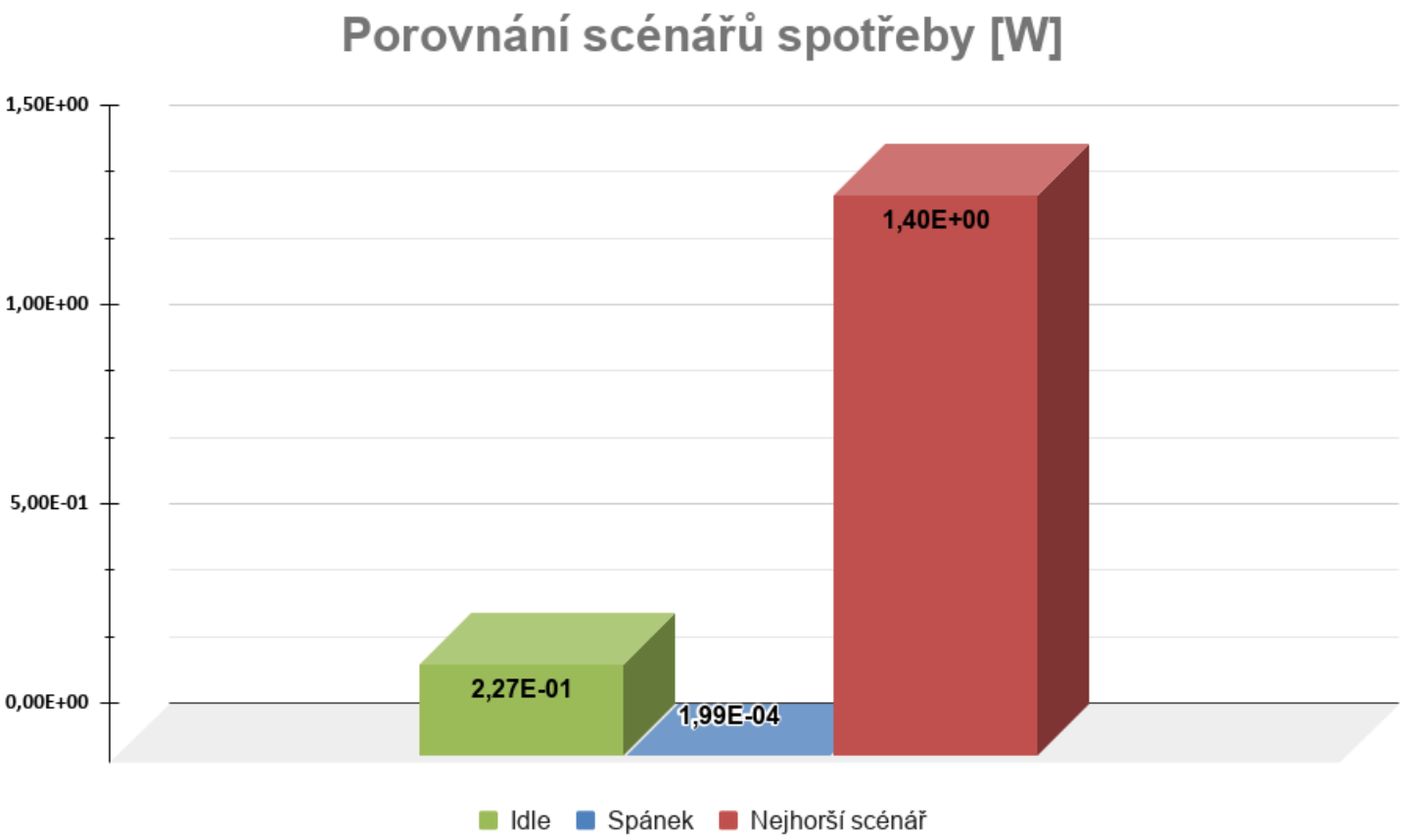
Typ HW	Odběr [A]	Napětí [V]	Spotřeba [W]
CO_2	6,00E-02	5,0	3,00E-01
Dual Supply, Bus Transceiver	3,60E-06	3,6	1,30E-05
FRAM	1,00E-04	3,6	3,60E-04
LED1-R	3,00E-02	2,5	7,50E-02
LED2-G	3,00E-02	2,5	7,50E-02
MCU	8,80E-05	3,6	3,17E-04
Multicell Battery Gas Gauge	6,50E-04	5,0	3,25E-03
Narrow Band	2,50E-01	3,8	9,50E-01
Operační zesilovač	4,50E-07	5,0	2,25E-06
RTC	6,00E-08	3,0	1,80E-07
Stepdown convertor	4,00E-07	5,0	2,00E-06
TMP/HUM	2,20E-04	3,3	7,26E-04
Celkem	**3,71E-01**		**1,40E+00**

Tabulka A.3: Spotřeba v nejhorším možném scénáři

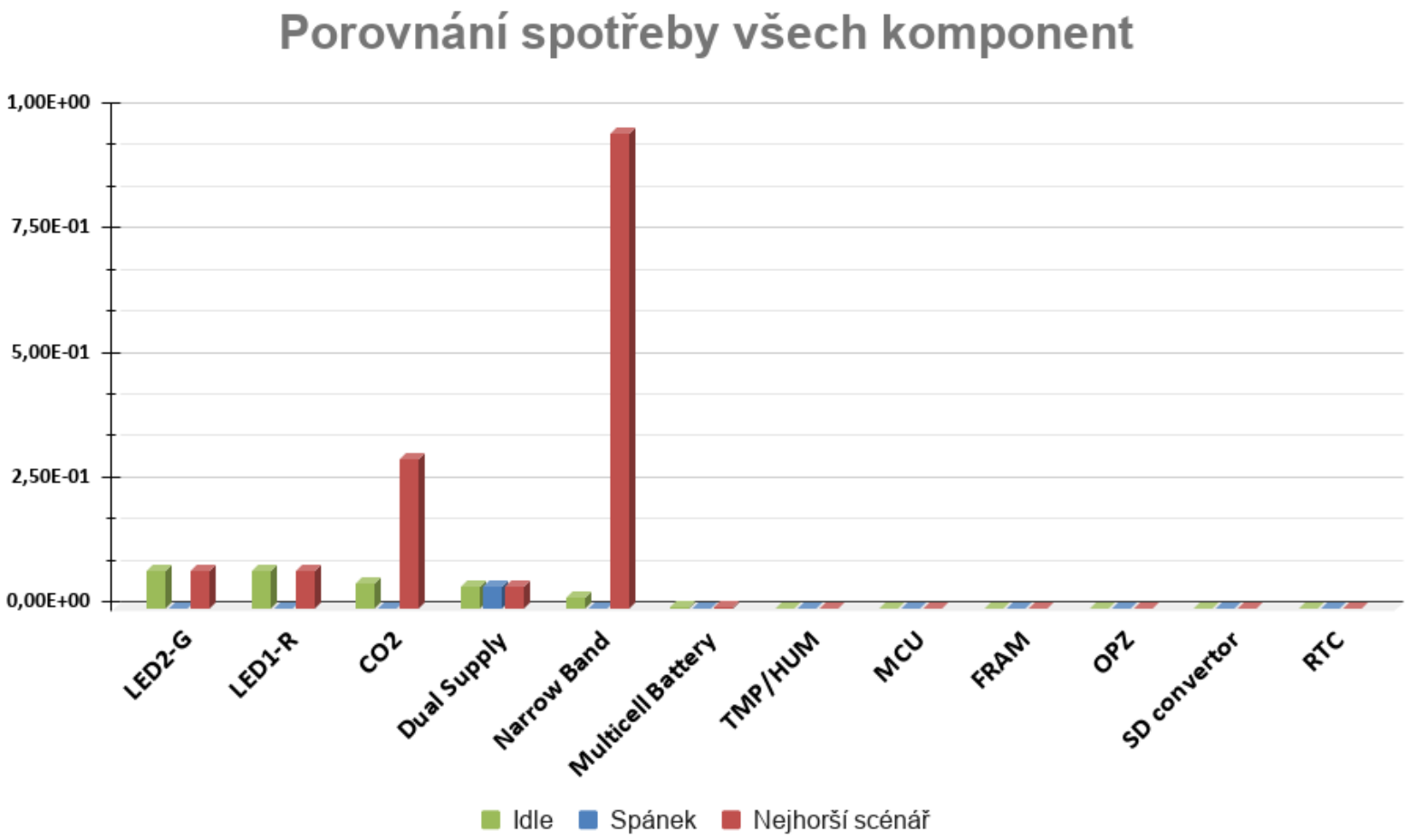
			Spotřeba		
Typ HW	**Odběr**	**Napětí [V]**	**Idle [W]**	**Spánek [W]**	**Nejhorší scénář [W]**
Narrow Band [9]	5uA(sleep), 6mA(idle), 250mA(transmit)	3,60	2,28E-02	1,90E-05	9,50E-01
CO_2 [3]	10mA, 60mA(peak)	5,00	5,00E-02	0,00E+00	3,00E-01
LED1-R [15]	30mA	2,50	7,50E-02	0,00E+00	7,50E-02
LED2-G [15]	30mA	2,50	7,50E-02	0,00E+00	7,50E-02
Dual Supply [16]	1uA	3,60	1,30E-05	1,30E-05	1,30E-05
Multicell Battery [5]	650uA(idle), 15uA(off), 80uA(sleep)	5,00	3,25E-03	7,50E-05	3,25E-03
TMP/HUM [4]	220uA	3,30	7,26E-04	0,00E+00	7,26E-04
FRAM [7]	100uA(active), 3uA(standby)	3,60	1,08E-05	0,00E+00	3,60E-04
MCU [1]	88uA(run), 0.27µA (Standby mode)	3,60	3,17E-04	9,72E-05	3,17E-04
OPZ [17]	0.45uA	5,00	2,25E-06	2,25E-06	2,25E-06
SD convertor [18]	400nA	5,00	2,00E-06	2,00E-06	2,00E-06
RTC [6]	60nA	3,60	1,80E-07	1,80E-07	1,80E-07
		Celkový výkon	2,27E-01	1,99E-04	1,40E+00

Tabulka A.4: Energetická náročnost jednotlivých komponent seřazeno podle spotřeby (landscape)

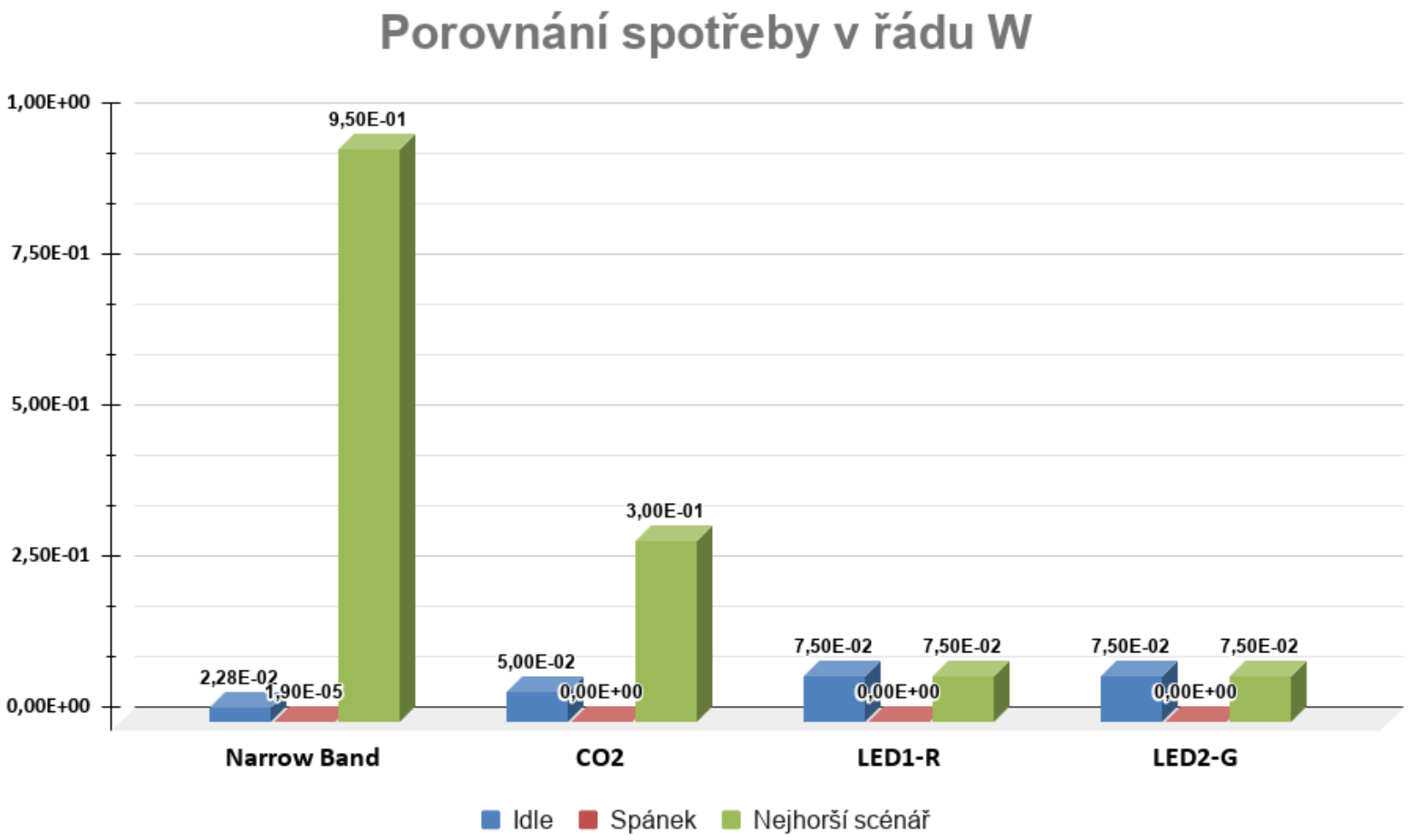
A.2 Grafy spotřeby



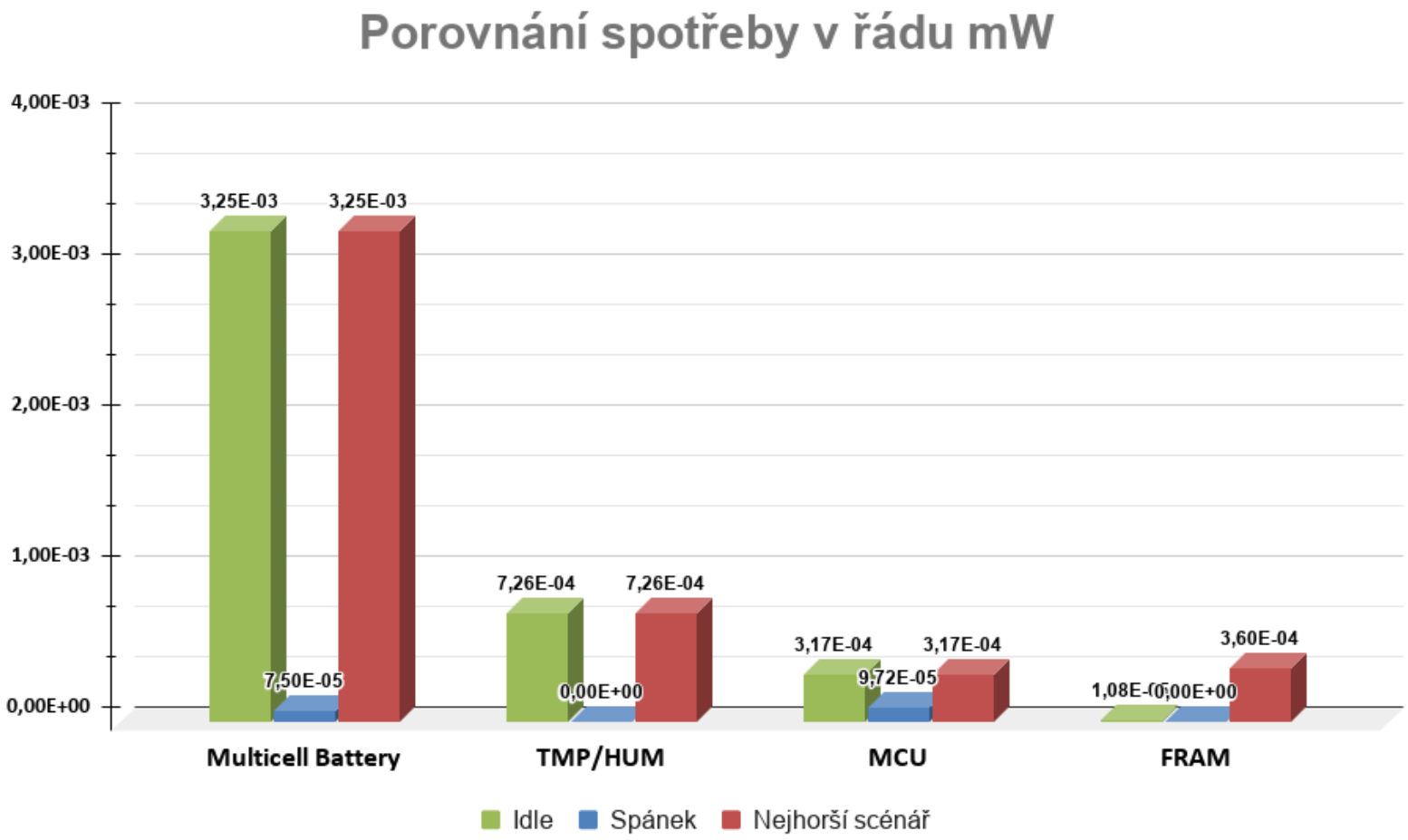
Obrázek A.1: Porovnání spotřeby podle scénářů



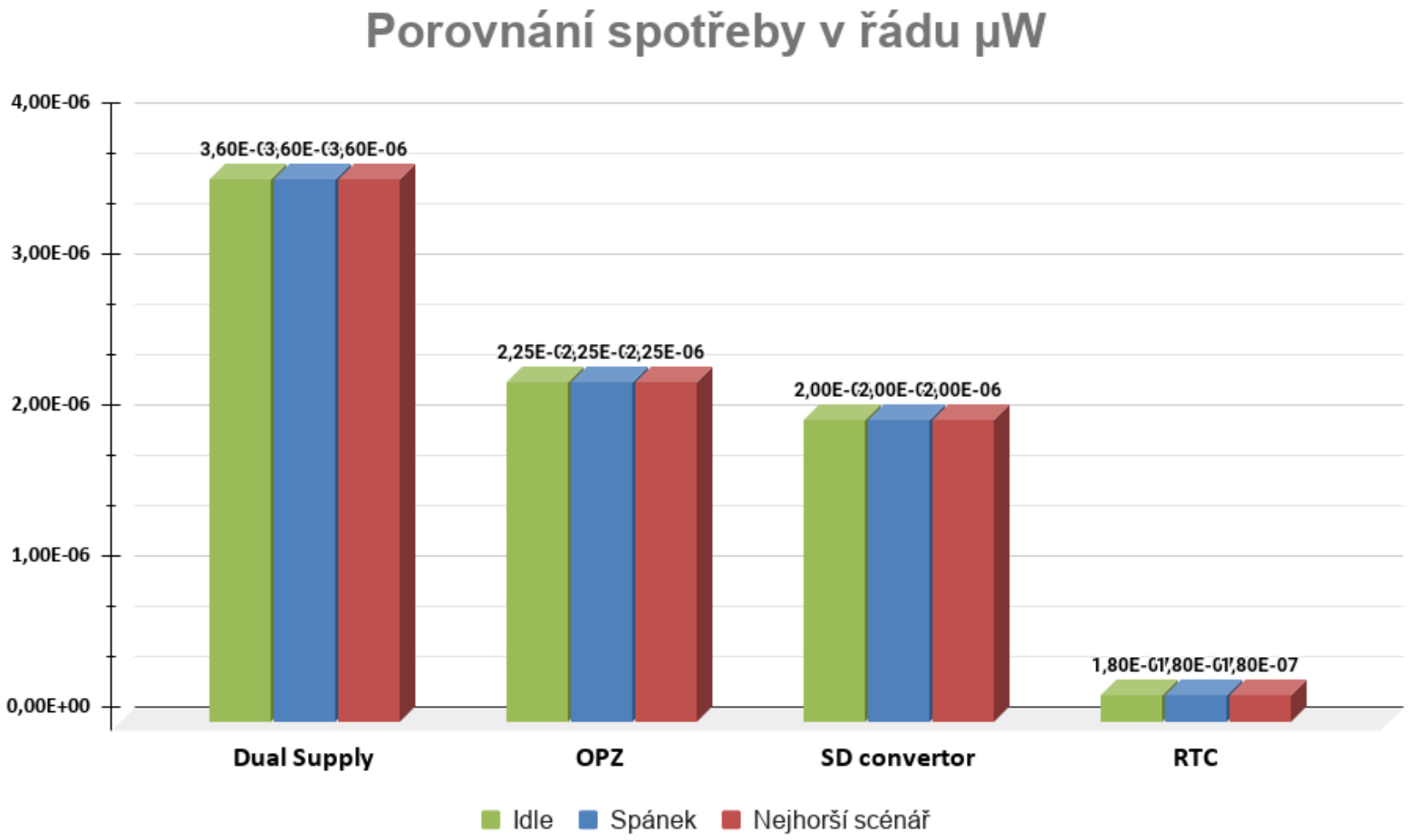
Obrázek A.2: Porovnání spotřeby všech komponent najednou



Obrázek A.3: Porovnání spotřeby v řádu *W*

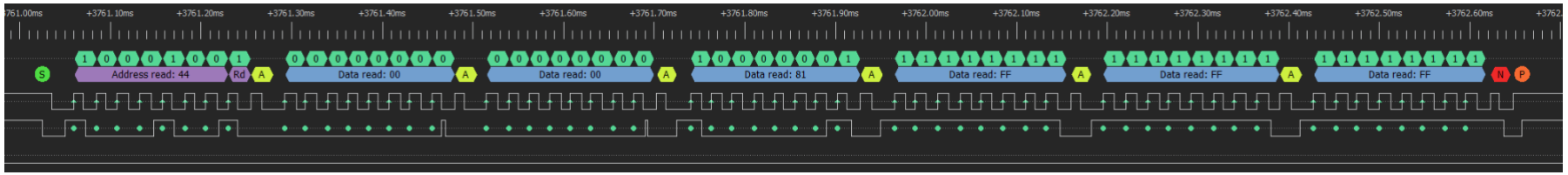


Obrázek A.4: Porovnání spotřeby v řádu mW

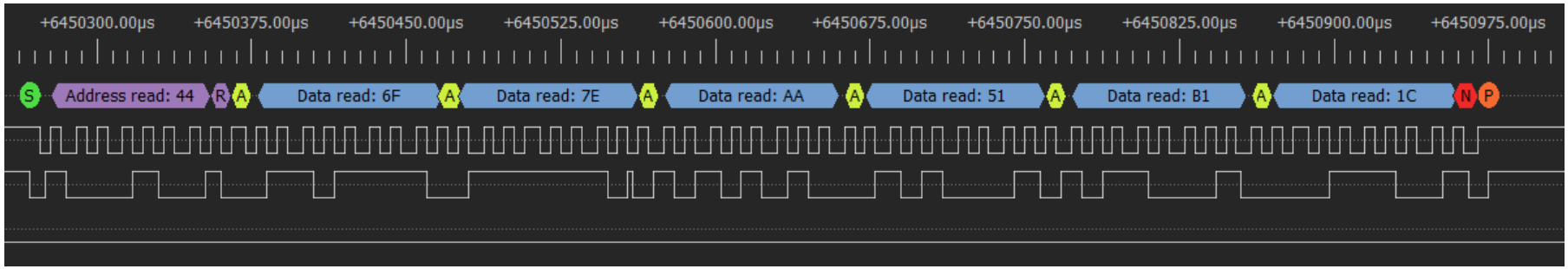


Obrázek A.5: Porovnání spotřeby v řádu μW

A.3 Vizualizace komunikace



Obrázek A.6: Měření komunikace SHT31 na I^{2}C sběrnici, ukázka chybného čtení



Obrázek A.7: Měření komunikace SHT31 na I^{2}C sběrnici, ukázka správného čtení

A.4 Webová aplikace

Čas	Teplota (°C)	Vlhkost (%)	CO2 (ppm)
2020-06-20 at 10:56	9472	22528	659
2020-06-20 at 10:59	9472	22528	652
2020-06-20 at 11:04	9472	22528	657
2020-06-20 at 11:09	9472	22528	624
2020-06-20 at 11:14	9472	22528	642
2020-06-20 at 11:19	9472	22528	687
2020-06-20 at 11:24	9472	22528	701
2020-06-20 at 11:29	9472	22528	724

Obrázek A.8: Ukázka webové aplikace (landscape)