ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Meteostanice s využitím Bluetooth Low Energy
Student:	David Šafrata
Vedoucí:	Ing. Lukáš Ručkay, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačové inženýrství
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Navrhněte a implementujte meteostanici, která bude měřit a zaznamenávat údaje o teplotě, tlaku a vlhkosti vzduchu. Naměřená data budou přenášena pomocí bezdrátové technologie Bluetooth Low Energy (BLE) do chytrého telefonu nebo PC.

Meteostanici implementujte na čipu EM9304 s využitím vhodných periferií pro připojení senzorů. Během implementace klaďte důraz na nízkou spotřebu a proveďte analýzu spotřeby energie meteostanice v různých fázích aktivity. Pro zobrazení údajů z meteostanice v chytrém telefonu/PC je možné využít nebo upravit již existující aplikace.

Meteostanice je určena pro externího zadavatele a proto jsou některé prvky systému (např. Bluetooth čip) specifikovány zadavatelem.

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 2. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra číslicového návrhu

Bakalářská práce

Meteostanice s využitím Bluetooth Low Energy

David Šafrata

Vedoucí práce: Ing. Lukáš Ručkay, Ph.D.

15. května 2017

Poděkování

Děkuji zejména mému vedoucímu za rady a praktickou pomoc v průběhu tvorby této práce. Děkuji také své rodině za vytvoření zázemí po celou dobu mého bakalářského studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Šafrata, David. *Meteostanice s využitím Bluetooth Low Energy*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce popisuje implementaci meteostanice, která pomocí Bluetooth Low Energy (BLE) umožňuje zasílání historie naměřených dat. Aplikace je implementována na čipu EM9304 s připojenou EEPROM pamětí a senzorem teploty, tlaku a vlhkosti pomocí sériového rozhraní. Výsledkem práce je funkční zařízení, se kterým je možné komunikovat pomocí speciální klientské aplikace pro smartphone, kde uživatel zasílá požadavky a získává vizualizovaná data. Práce obsahuje také výsledky měření spotřeby elektrické energie v různých fázích aktivity čipu.

Klíčová slova Bluetooth Low Energy, BLE, meteostanice, EM9304, SPI, Android, měření spotřeby

Abstract

This bachelor's thesis describes implementation of weather monitoring station, which is able to send historical measurements via Bluetooh Low Energy (BLE). Application is implemented for SoC EM9304 with EEPROM memory and temperature, pressure and humidity sensor connected over serial interface. Result of this thesis is working device, which communicates with smartphone using dedicated client application, where user sends commands and receive visualized data. The thesis also contains results of power consumption measurements in various phases of activity.

Keywords Bluetooth Low Energy, BLE, weather monitoring, EM9304, SPI, Android, consumption measurement

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Oblast vestavných systémů v posledních letech zažívá velký rozmach. Roste trh s takzvanými nositelnostmi a se zařízeními z oblasti internetu věcí [4], které jsou často napájeny z baterií a spotřeba čipů přímo ovlivňuje jejich výdrž. Díky těmto zařízením roste poptávka po čipech umožňujících bezdrátově komunikovat se světem, které si současně udržují velmi nízkou spotřebu.

Pražská návrhová divize společnosti EM Microelectronic se na oblast nízkopříkonových čipů ve velké míře zaměřuje. V době psaní této práce byl dokončen vývoj čipu EM9304, který vývojářům umožňuje vytvářet aplikace využívající pro komunikaci bezdrátovou technologii Bluetooth Low Energy (dále jen jako BLE). Tento čip je nástupcem čipu EM9301, který pracuje pouze jako kontrolér implementující nižší vrstvy BLE konektivity, vyžadující spolupráci s externím mikrokontrolérem. EM9304 přidává kompletní podporu všech vrstev – od fyzické až po aplikační. [5, 6]

Požadavkem firmy je připravit pro prezentaci potenciálním zákazníkům ukázkové aplikace, které předvedou různé možnosti tohoto čipu. Z tohoto důvodu je jako téma této bakalářské práce zvolena meteostanice. Ta představuje komplexnější aplikaci, která pracuje s periferiemi, využívá mnoha možností čipu a demonstruje energetickou úspornost aplikace postavené na této platformě.

Kapitola 1

Seznámení s technologiemi

1.1 Bluetooth Low Energy[1]

Bluetooth Low Energy (zkráceně BLE), známý též pod obchodním označením Bluetooth Smart je standard bezdrátové komunikace, který byl vydán v rámci *Core Specification* verze 4.0 pod záštitou organizace Bluetooth SIG. [1, 7] Historie BLE sahá dále do minulosti k technologii nazvané Wibree, která byla vydána skupinou okolo společnosti Nokia v roce 2006 [8] a po následných jednáních bylo dohodnuto, že se tato technologie integruje do standardu Bluetooth. [9] To se stalo skutečností v roce 2010, kdy byla vydána výše zmíněná specifikace verze 4.0. Prvním smartphonem s podporou Bluetooth Low Energy byl iPhone 4S, který byl představen koncem roku 2011. [7]

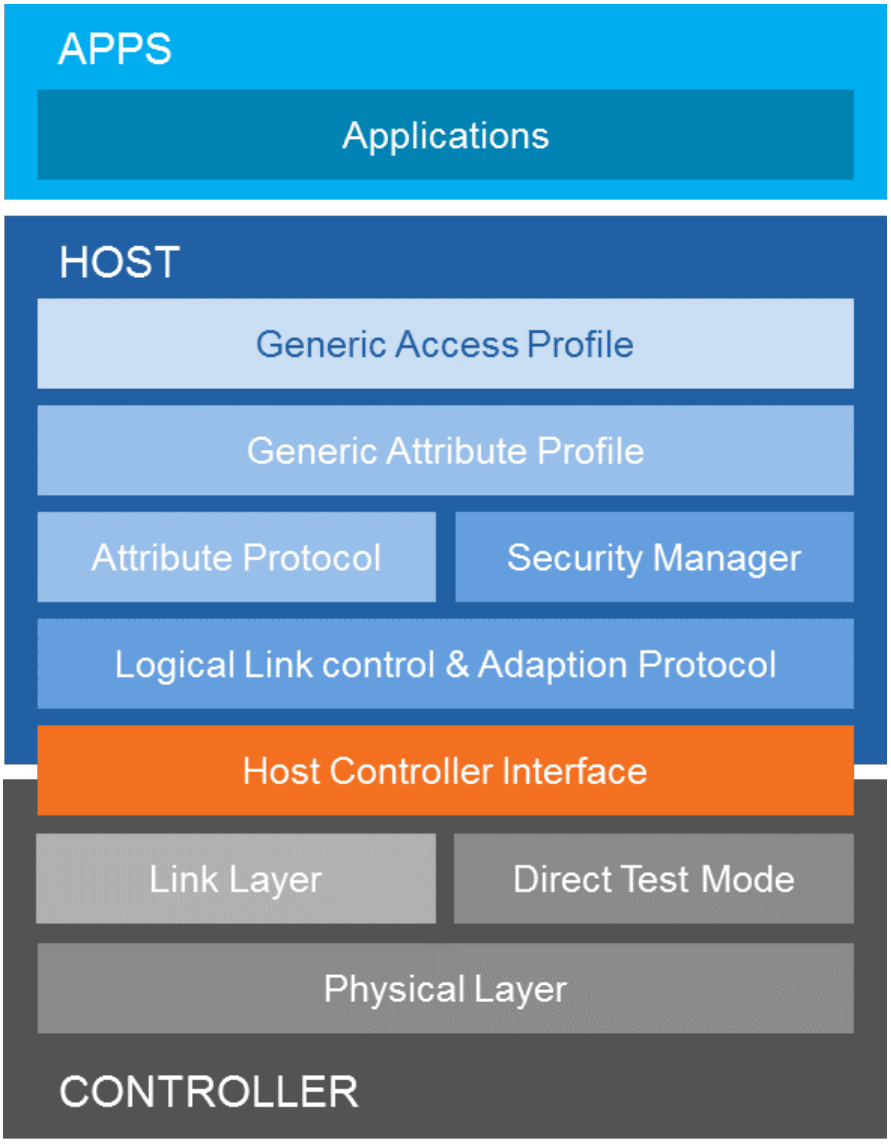
Vývoj specifikace pokračoval dále a na konci roku 2013 byla uvolněna specifikace verze 4.1, následovaná verzí 4.2 z konce roku 2014 a verzí 5.0 z konce roku 2016. [10]

BLE vznikl na základě potřeby vytvořit bezdrátový komunikační protokol, který by byl energeticky úspornější, než byla v té době běžně využívaná řešení. Významného snížení spotřeby se dosahuje zejména díky omezení doby aktivního využití rádia. [9] Podobně jako u síťové komunikace a modelů jako ISO/OSI nebo TCP/IP se BLE logicky rozděluje na jednotlivé vrstvy, které obdobně jako u síťové komunikace popisují fyzickou až aplikační vrstvu. Souhrnně se referenční model běžně označuje jako *stack*, jehož struktura je graficky znázorněna na obrázku 1.1. Dále jsou rozebrány jednotlivé vrstvy a jejich klíčové vlastnosti.

1.1.1 Fyzická vrstva

Komunikace na fyzické vrstvě probíhá bezdrátově v nelicencovaném frekvenčním pásmu 2.4 GHz, které využívá i technologie Wi-Fi. [9] Pro rádiový pře-

[1] Informace v této sekci čerpány z [2], pokud není uvedeno jinak. Jedná se tedy o informace vycházející ze specifikace standardu BLE verze 4.2, kterou implementuje čip EM9304.



Obrázek 1.1: Struktura BLE stacku s jednotlivými vrstvami, převzato z [1]

nos dat se využívá jednoduché frekvenční modulace, známé jako jako GFSK (Gaussian frequency-shift keying). Toto klíčování využívá pro modulaci jen 2 frekvence – jedna z nich odpovídá logické 1, druhá z nich odpovídá logické 0. V případě BLE se konkrétně jedná o změnu od střední frekvence. Logická 1 má frekvenci typicky o 185 kHz vyšší, zatímco logická nula má frekvenci typicky o 185 kHz nižší. Frekvenční pásmo se dělí celkem na 40 kanálů s rozestupy 2 MHz. Přenosová rychlost fyzické vrstvy[2] je 1 Mbit/s.

[2]Jako u ostatních tvrzení se i zde jedná o informaci dle BLE standardu verze 4.2. Následující verze 5.0 definuje i jiné přenosové rychlosti. [10]

1.1.2 Linková vrstva

Linková vrstva je jedna z nejkomplexnějších vrstev BLE stacku, protože musí jako jediná pracovat přesně v reálném čase. Výše zmíněných 40 kanálů je rozděleno na dvě části po 37 a po 3 kanálech, které se využívají pro specifické účely. Část se 3 kanály se používá při vysílání takzvaných advertising paketů, které se používají mimo spojení a pomocí těchto paketů je inzerováno, zda-li je zařízení zjistitelné a připojitelné a případně v reakci na tento paket na této vrstvě proběhne navázání spojení. Díky těmto paketům je také možné do okolí odesílat omezené množství dat. Zbylých 37 kanálů se používá během spojení a jsou stejně jako advertising pakety průběžně střídány pro zaručení dostatečné robustnosti.

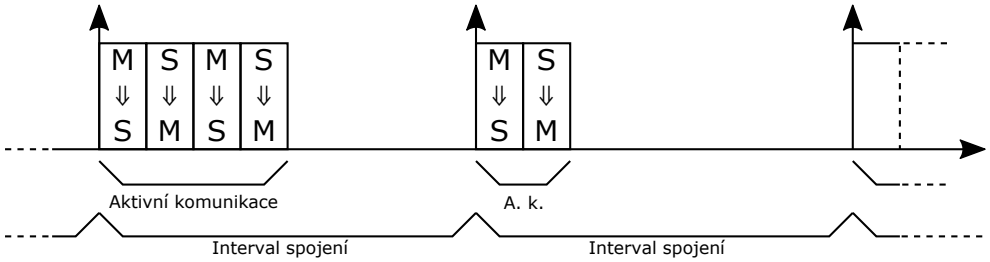
Data jsou zasílána v paketech, které umožňují odeslat 0-255 bajtů dat, jenž jsou využívána zejména vyššími vrstvami, doprovázené minimálně 10 bajty režijních informací, které jsou rozděleny na 1 B preambule, 4 B přístupové adresy, 2 B hlavičky (1 B určující typ dat a další informace a 1 B délky dat) a 3 B kontrolního součtu CRC. U advertising paketů, které jsou detailněji popsány v 1.1.6, je struktura podobná, pouze data mají rozsah 0-37 bajtů.

PA [1 B]	Přístupová adresa [4 B]	Hlavička [2 B]	Data [0-255 B]	CRC [3 B]

PA = preambule (0xAA)

Obrázek 1.2: Paket na linkové vrstvě (dle verze 4.2 standardu)

Jak již bylo dříve uvedeno, BLE může být energeticky efektivní zejména díky omezení doby, kdy je aktivně využíváno rádio. Toho je při spojení docíleno komunikací v periodicky se opakujících intervalech smluvené délky, jak je zobrazeno na obrázku 1.3. Tyto intervaly fungují tak, že na začátku intervalu proběhne vzájemná komunikace mezi zařízeními, tudíž rádio je v tu dobu plně aktivní. Po výměně dat může zařízení po zbytek intervalu mít rádio vypnuté a zapnout ho opět až na začátku dalšího intervalu. V drtivé většině případů je poměr aktivní komunikace vůči celkové délce intervalu spojení velmi malý, tudíž většinu času je rádio vypnuté a díky tomu zařízení šetří energii.



Obrázek 1.3: BLE komunikace v intervalech, kde M je zkratka pro zařízení role master, které iniciovalo spojení a S je zkratka pro zařízení role slave

1.1.3 HCI

HCI (Host controller interface) je standardizované rozhraní mezi kontrolérem (fyzická a linková vrstva) a vyššími vrstvami stacku, který je graficky znázorněn na obrázku 1.1. Z vyšších vrstev je možné do kontroléru odesílat příkazy a z opačného směru přijímat odpovědi a události. Formát dat je přesně daný standardem, ale konkrétní fyzická podoba rozhraní HCI je volitelná. Většinou se jedná o sériové protokoly UART, USB nebo SDIO; výrobce však může zvolit i proprietární řešení.

1.1.4 L2CAP

L2CAP (Logical link control and adaptation protocol) v BLE zodpovídá zejména za rozdělení velkých paketů z vyšších úrovní tak, aby se vešly do velikosti paketu v linkové vrstvě. Tato vrstva také definuje logické kanály, které umožňuje multiplexovat. Popsané funkce mají samozřejmě i inverzní charakter, tedy L2CAP umožňuje pakety skládat a logické kanály demultiplexovat.

1.1.5 SM

Security manager se stará o párování zařízení, které je následováno výměnou klíčů, kterými se poté šifruje komunikace. Od standardu verze 4.2 BLE podporuje také výměnu klíčů pomocí eliptických křivek, kde nejsilnější podporovaný algoritmus je P-256. Pro šifrování, které probíhá v linkové vrstvě, se využívá blokové šifry AES-128 v režimu CCM.

1.1.6 GAP

GAP je jedna z nejdůležitějších vrstev BLE stacku. Stará se například o vysílání a skenování takzvaných advertising paketů. Tyto pakety určují, jak se může přistupovat k danému zařízení. Zároveň se v těchto paktech mohou přenášet další užitečná data, která jsou dostupná všem skenujícím zařízením v dosahu.

GAP také rozlišuje 4 role zařízení, podle chování:

Broadcaster
: Zařízení, které nepodporuje spojení, vysílá jen advertising pakety typu ADV_NONCONN_IND, volitelně odpovídá pomocí SCAN_RSP. Tato role zařízení se hojně využívá v takzvaných majících (beacon), které se mohou využít například pro navigaci uvnitř budovy nebo dopravní stavby, obecně pro předání nějaké informace do okolí.

Observer
: Zařízení, které nepodporuje spojení a pouze skenuje data z okolí, pasivně nebo volitelně aktivně za využití SCAN_REQ paketů.

Peripheral
: Zařízení, které podporuje spojení iniciované Central zařízením. Typicky se jedná o nějaké zařízení, které poskytuje nějaká užitečná data, jako například meteostanice, měřič tepu a podobné zařízení.

Central
: Zařízení, které nevysílá advertising pakety, pouze skenuje advertising data z okolí a v případě že najde peripheral zařízení, umožňuje s ním navázat spojení. Typicky se jedná o smartphone nebo podobné zařízení.

Podobně jako při komunikaci ve spojení se i advertising pakety odesílají v určitých intervalech. Interval může mít délku od 20 ms do 10.24 s, ke kterému se přidá náhodná hodnota zpoždění z intervalu 0 až 10 ms. Advertising se s každou periodou intervalu střídá postupně na všech třech vyhrazených kanálech, které jsou popsány v 1.1.2.

Na advertising paket může reagovat druhá strana takzvaným scan request požadavkem, na který může dostat odpověď pomocí takzvaného scan response, což je paket stejného formátu jako advertising, ale s jinou hlavičkou a obsahuje dodatečná data k advertising paketu.

Advertising pakety mají podobnou strukturu jako pakety používané při spojení, pouze maximální délka vlastních dat je 37 bajtů. Jsou definovány tyto typy advertising paketů [11]:

ADV_IND
: Paket který určuje, že zařízení je připojitelné pro všechna zařízení. Data obsahují 6 bajtů vlatní adresy a až 31 bajtů vlastních dat.

ADV_DIRECT_IND
: Paket který určuje, že zařízení je připojitelné pro zařízení se specifikovanou adresou. Data obsahují 6 bajtů vlasní adresy a 6 bajtů adresy druhého zařízení.

ADV_NONCONN_IND
: Paket který určuje, že zařízení není připojitelné pro žádné zařízení. Data obsahují 6 bajtů vlatní adresy a až 31 bajtů vlastních dat.

ADV_SCAN_IND
: Paket se stejnou strukturou jako ADV_NONCONN_IND, ale navíc indikuje, že může poskytnout dodatečná data ve scan response paketu.

SCAN_REQ
: Požadavek o scan response paket. Data obsahují 6 bajtů vlastní adresy a 6 bajtů adresy zařízení, po kterém požaduje dodatečná data.

SCAN_RSP
: Dodatečná data (scan response), kde data paketu obsahují 6 bajtů vlastní adresy a až 31 bajtů vlastních dat.

CONNECT_REQ
: Požadavek o vytvoření spojení.

Nejčastěji se jako vlastní data advertising paketu používá celé, nebo částečné jméno zařízení a celý nebo částečný seznam služeb, které jsou poskytovány na vrstvě GATT.

Přes GAP se také inicializuje párování, které je zpracováváno v kooperaci s vrstvou Security Manager. Kromě párování je v BLE umožněno používat různé typy adres, které jsou buď fixní, nebo náhodné, což umožňuje zařízení zůstat v relativní anonymitě. Z takzvané náhodné resolvable adresy se dá při znalosti určitého tajného klíče zjistit reálná identita zařízení.

1.1.7 ATT

Atributový protokol (ATT) specifikuje rozhraní pro přístup k datům v zařízení na vyšší úrovni. Je postaven na architektuře klient - server. Každá logická jednotka dat (atribut) má lokální handle, unikátní identifikátor (UUID), vlastní data a práva, která určují, jak se k nim může přistupovat. Atributový protokol také umožňuje klientskému zařízení tyto atributy objevovat a prohledávat. Tento protokol specifikuje 6 typů zpráv, které se vážou k atributům:

Request (požadavek)
: Například požadavek na čtení atributu (read), nebo potvrzovaný zápis (write)

Response (odpověď)
: Odpověď na požadavek klienta

Command (příkaz)
: Například nepotvrzovaný zápis (write no response)

Notification (notifikace)
: Nepotvrzovaná data iniciovaná serverem pro klienta

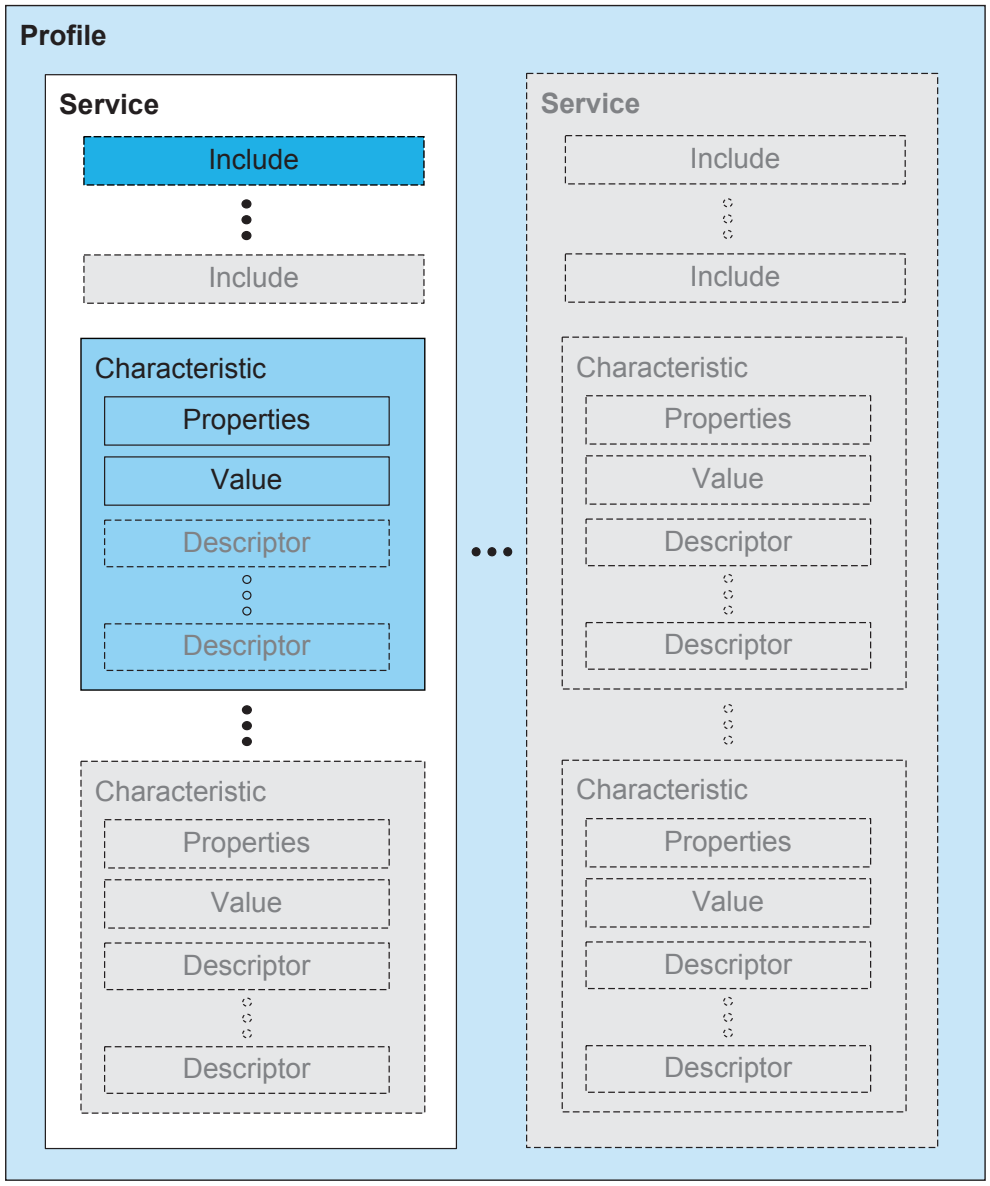
Indication (indikace)
: Potvrzovaná data iniciovaná serverem pro klienta

Confirmation (potvrzení)
: Potvrzení klienta o příjmu indikace

Práva přístupu jsou: čtení, zápis, případně kombinace obojího a volitelné omezení přístupu, kdy atribut může požadovat šifrování, autorizaci nebo autentizaci.

1.1.8 GATT

GATT je framework, který je postavený na ATT a obohacuje ho o dodatečná rozšíření. Hlavní logická jednotka protokolu GATT je takzvaná služba (service). Služba je určitá množina dat, která se váže k určité funkcionalitě. Rozlišují se primární služby, které může klient objevit v rámci procedury *Primary Service Discovery* a sekundární služby, které mohou být odkazovány z jiné služby nebo být použity vyšší vrstvou. Služba se může skládat z ostatních vložených (include) služeb a jednotlivých charakteristik. Charakteristika



Obrázek 1.4: Hierarchie GATT, převzato z [2]

reprezentuje určitou hodnotu. Je složená z více atributů, přičemž první atribut je deklarace charakteristiky, kde se určuje jak se může používat a handle atributu s vlastní hodnotou charakteristiky, která je obvykle uložena v následujícím handle za handle deklarace. K charakteristice se mohou vázat i dodatečné informace v následných atributech, které se označují jako deskriptory. Jedná se například o textový popis hodnoty, formát hodnoty a specifická nastavení od klienta nebo serveru. Celková hierarchie GATT je na obrázku 1.4.

Mnoho běžně využívaných služeb má své rozhraní pevně definované ve

standardu, celkově se jedná o několik desítek služeb. Při využití těchto standardizovaných služeb nebývá nutné vytvářet vlastní klientské aplikace, neboť je možné využít těch, které standard dodržují a implementují. Mezi služby definované ve standardu například patří:

- Continuous Glucose Monitoring
- Current Time Service
- Human Interface Device
- Indoor Positioning

V klientské aplikaci lze využít logického seskupení několika GATT služeb do takzvaného profilu. Příkladem může být například profil Proximity, který se skládá ze služeb Tx Power, Immediate Alert a Link Loss. Podrobné informace k problematice služeb a profilů vrstvy GATT se může čtenář dozvědět v [10, 12].

1.2 Čip EM9304[3]

Jak již bylo zmíněno v Úvodu, čip EM9304 byl v nedávné době vyvinut jako vylepšení předchozího čipu EM9301, který funguje pouze jako BLE kontrolér. To znamená, že čip EM9301 poskytuje fyzickou vrstvu, linkovou vrstvu a komunikuje přes standardizované rozhraní HCI s externím mikrokontrolérem, který implementuje vyšší vrstvy BLE stacku, včetně aplikační vrstvy. Hlavní myšlenkou čipu EM9304 je rozšíření na úroveň, kdy bude samostatně poskytovat všechny vrstvy BLE stacku a také prostor pro uživatelské aplikace.

Díky podpoře celého BLE stacku může čip operovat ve 3 režimech:

Režim HCI
: Funguje obdobně jako EM9301, tedy poskytuje rozhraní přes HCI (na obrázku 1.1 nad částí označené jako CONTROLLER). V tomto režimu se nachází čip po startu.

Režim ACI
: Poskytuje aplikační rozhraní (na obrázku 1.1 nad částmi označenými jako CONTROLLER a HOST) pro externí mikrokontrolér, kde běží aplikace.

Režim SoC
: Aplikace sdílí stejné prostředky, kterými se řídí BLE konektivita. V režimu SoC se tedy implementují všechny vrstvy BLE stacku, včetně aplikační vrstvy.

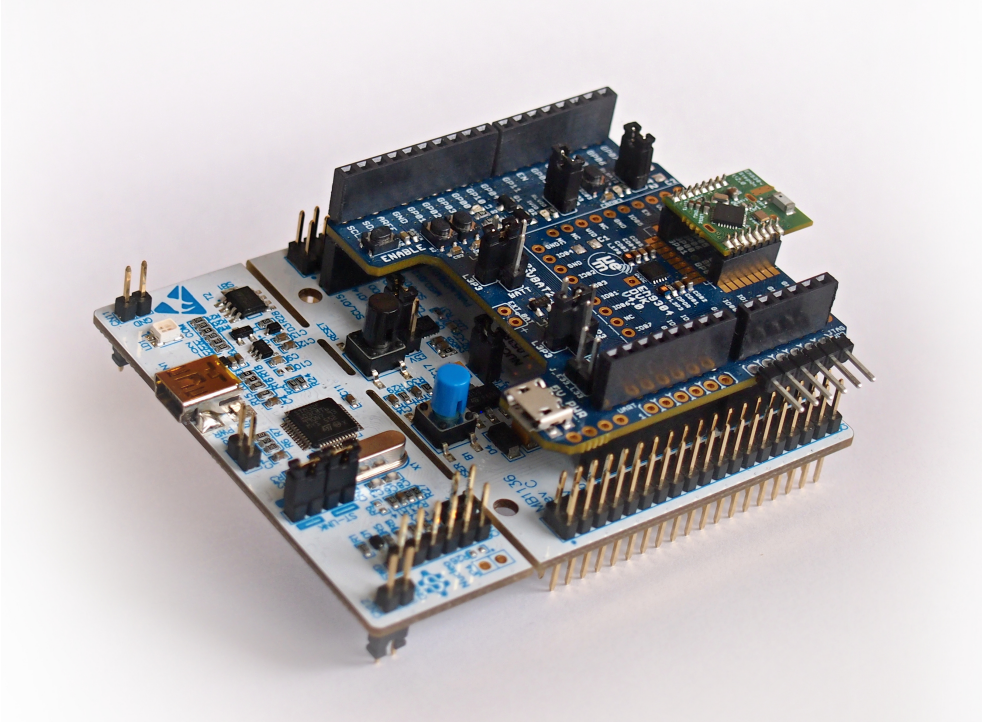
Dále následuje přehled nejzajímavějších hardwarových charakteristik čipu EM9304:

- 32-bitový RISC procesor ARC EM4, běžící na 24 MHz

[3] Informace v této sekci čerpány z [6], pokud není uvedeno jinak.

- Aritmetická jednotka pro operace v plovoucí řádové čárce
- 136 kB ROM paměti, která obsahuje kompletní BLE stack
- 128 kB OTP paměť pro opravy, rozšíření, uživatelské aplikace a další data
- 48 kB RAM pro vývojářské účely
- Sériová rozhraní UART, I^2C a SPI
- Kryptografická jednotka s HW generátorem náhodných čísel, podporou AES-128 a kryptografie nad eliptickými křivkami P-256
- Volitelný 32 768 Hz hodinkový krystal pro přesné časování
- Rozsah napájecího napětí 1.05 V až 3.6 V
- 12 GPIO pinů

Zajímavé parametry se vztahují k nízké spotřebě energie, která byla jednou z hlavních priorit při vývoji tohoto čipu. Výrobce uvádí, že typická spotřeba při napětí 3 V a výkonu rádia 0 dBm je 3.0 mA při příjmu a od 2.2 mA při odesílání dat. Při režimu connected sleep mode, který udržuje navázaná spojení je jen 1 µA.



Obrázek 1.5: Hardwarový DVK pro EM9304

K čipu jsou dodávány podpůrné nástroje pro vývoj aplikací. Jedná se zejména o hardwarový DVK (development kit) pro snadné nahrávání aplikací přímo do paměti čipu a snadnější přístup k pinům a SDK (software development kit), který obsahuje rozsáhlé knihovny pro práci s čipem.

Kapitola 2

Analýza a návrh

2.1 Možnosti řešení

2.1.1 Hardware

V oblasti hardwaru, respektive podpůrných periferií, je možné vybírat z mnoha modulů. Aplikace pro splnění zadání potřebuje dostatečně velikou paměť pro ukládání naměřených hodnoty a odpovídající senzor/y pro měření teploty, tlaku a vlhkosti vzduchu. Vzhledem k tomu, že čip EM9304 neobsahuje integrované A/D převodníky, je při hledání senzorů na měření teploty, atmosférického tlaku a relativní vlhkosti vzduchu logické se omezit na senzory poskytující digitální přístup k datům. Volitelnost je v použití sériového rozhraní, pomocí kterého budou periferie komunikovat s čipem. EM9304 nabízí hardwarovou podporu pro rozhraní SPI a I^2C, která poskytují sériový přístup k datům a připadají v úvahu pro tento typ využití. Je také potřeba vybrat vhodné GPIO piny, ke kterým se přiřadí jednotlivé signály sériového rozhraní.

2.1.2 Systém a software

V oblasti embedded softwaru pro čip EM9304 jsou technologie pro řešení jasně dané. Aplikace se implementuje v jazyku C, ve kterém je vytvořeno SDK a další knihovny určené zejména k obsluze BLE konektivity, hardwarových prostředků čipu a podobně. Aplikace psané pro režim SoC, který je používán v této práci, kdy uživatelská aplikace využívá pro běh přímo procesor čipu EM9304, jsou doporučovány psát za využití odlehčeného systému reálného času QP-nano.

QP-nano je systém řízený událostmi a využívá asynchronní mechanismy. [13] Pro konkrétnější představu se jedná například o myšlenku návrhového vzoru známého zejména z vysokoúrovňového programování, který je často označovaný jako *Observer*. Tento vzor využívá takzvaného Hollywoodského principu „Nevolejte nám, my zavoláme vám“. [14] To se například projevuje u volání funkcí řídících hardwarové prostředky čipu, jako jsou třeba časo-

vače a sériová rozhraní, kdy se při volání předává ukazatel na funkci, která se vykoná při zpětném volání (callback), jakmile se požadovaná akce dokončí. Aplikace tedy nemusí vykonávat aktivní čekání nad daným prostředkem a může se věnovat jiné užitečné činnosti. Díky výše zmíněným obecným vlastnostem je QP-nano výhodné pro obsluhu BLE konektivity a umožňuje čipu strávit co nejvíce času v režimu sleep mode, kdy je spotřeba naprosto minimální. Zároveň je možné souběžně obsluhovat jak BLE, tak i kód vlastní aplikace. Konkrétní implementace přes výše uvedené omezení možností přesto poskytuje dostatečný prostor pro rozvinutí kreativity.

Volitelnost při implementaci komunikační části za využití BLE je v použití služeb na vrstvě GATT, která obsahují data v podobě charakteristik a definují přístup k nim; podrobněji je tato problematika rozebrána v 1.1.8. Je možné vybírat ze služeb definovaných ve standardu, kde se nabízí například služba *Environmental Sensing* [10], nebo využít proprietárních služeb.

2.1.3 Klientská aplikace

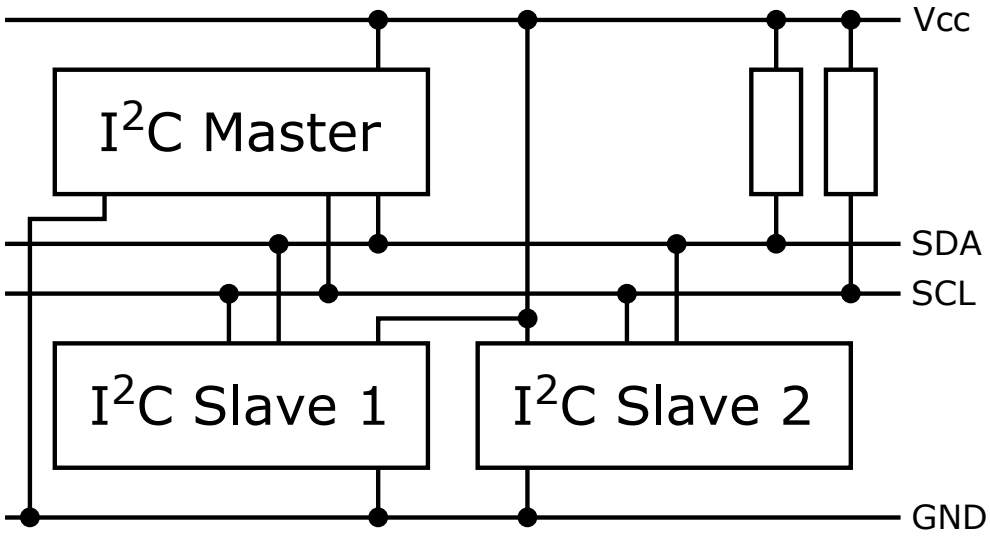
Zadání dává ve volbě klientské aplikace velkou volnost. Může se jednat o aplikaci pro PC nebo smartphone na libovolné platformě. Je také možné vlastní aplikaci neimplementovat a využít již existující univerzální aplikace. Konkrétní rozhodnutí závisí na volbě konkrétní služby z vrstvy GATT. V případě volby proprietární služby není možné počítat s podporou aplikací, které implementují jen služby definované ve standardu. Tvorba vlastní aplikace by znamenala rozšíření možností využití dle autorových představ, ale zároveň by vyžadovala vyhradit více času na samotnou implementaci.

2.2 Zvolené řešení

2.2.1 Hardware

2.2.1.1 Sériové rozhraní

Při volbě vhodných periferií, zejména tedy paměti pro ukládání výsledků a již zmíněných senzorů, bylo nejprve nutné zvolit vhodné digitální rozhraní. Rozhodování probíhalo zejména mezi sériovými rozhraními SPI a I^{2}C. Rozhraní UART nebylo při rozhodování uvažováno z důvodu vysoké energetické náročnosti, nicméně má stejně jako obě rozhraní zmíněná v předchozí větě v čipu hardwarovou podporu. [6]



Obrázek 2.1: Typické zapojení pro protokol I^{2}C

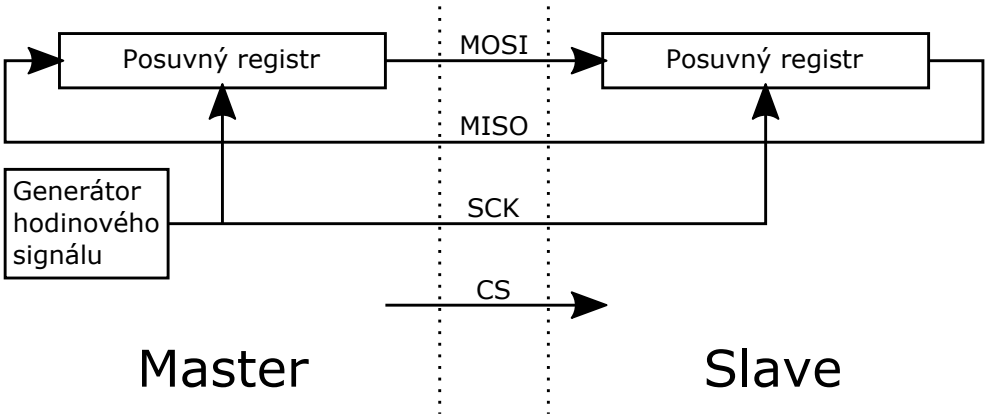
Jedním z faktorů při rozhodování byl počet pinů, které daný sériový protokol potřebuje, protože počet GPIO na čipu EM9304 je omezený. I^{2}C má v tomto ohledu jednoznačnou výhodu, protože ke své funkci potřebuje pouze 2 piny, nezávisle na počtu připojených zařízení na sběrnici. SPI v základu potřebuje 3 piny a pro každé připojené zařízení 1 pin navíc. Při malém počtu připojených zařízení a dostatku volných pinů nemá tento argument v celkovém důsledku zásadní roli.

Rozhraní	**Výhody**
I^{2}C	Menší počet využitých GPIO
SPI	Absence pull-up/down rezistorů Vyšší dosažitelná přenosová rychlost Plně duplexní komunikace

Tabulka 2.1: Porovnání výhod I^{2}C a SPI

Zásadním faktorem při rozhodování byla energetická úspornost daného rozhraní. Jedna ze zásadních nevýhod I^2C vychází ze samotného principu jeho fungování. Obě používané sběrnice SDA i SCL, jsou pomocí pull-up rezistorů nastaveny do klidové úrovně napájecího napětí. Data se přenášejí tak, že zařízení specifickým způsobem pomocí otevřeného kolektoru spojují sběrnici se zemí, a tak nastavují sběrnice do napěťové úrovně země. V tento okamžik se tak propojí napájecí napětí přes rezistor se zemí. Pokud je použito napájecí napětí ve výši 3 V a pull-up rezistor s odporem ve výši 1 kΩ je dle Ohmova zákona ($I = \frac{U}{R}$) proud protékající tímto jedním rezistorem $I_R = \frac{3}{1000}$ A = 0.003 A = 3 mA. Jedná se o relativně malé časové úseky, ale SPI díky absenci pull-up a pull-down rezistorů tímto specifikem netrpí.

Energetické úspornosti se dá dosáhnout také díky omezení času, kdy je nutné udržovat rozhraní aktivní. V tomto ohledu má opět výhodu rozhraní SPI, které umožňuje dosáhnout vyšších přenosových rychlostí, než rozhraní I^2C a navíc oproti tomuto rozhraní u SPI probíhá komunikace duplexně, neboli data jsou zasílána mezi zařízeními ve stejný okamžik a jednotlivé strany se nemusí v datovém využití sběrnice střídat. Díky rychlejší komunikaci se může omezit doba po kterou je čip aktivní, a tak může čip strávit více času v režimu sleep mode. Zásadní argumenty zvažované při volbě rozhraní jsou uvedeny v tabulce 2.1. Z výše uvedené úvahy vyšlo jako optimální využít rozhraní SPI.



Obrázek 2.2: Princip protokolu SPI

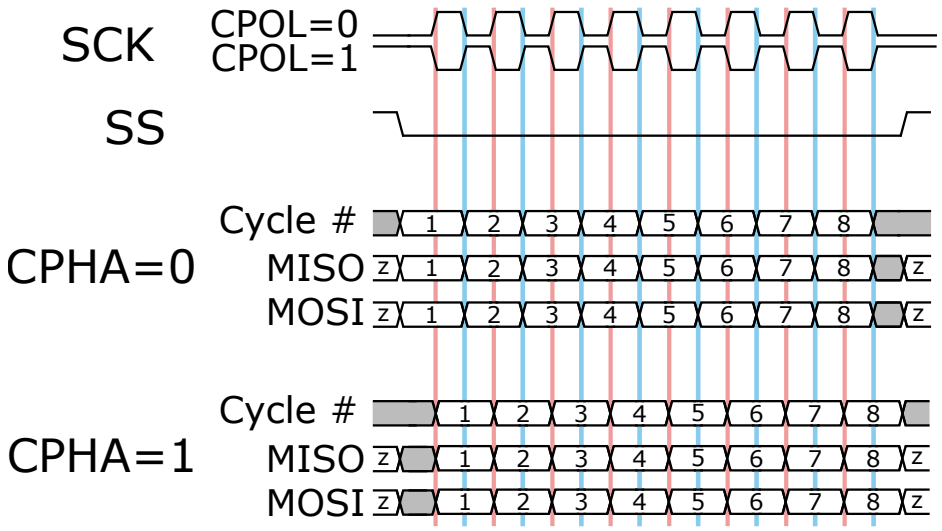
SPI je sériový synchronní komunikační protokol. Zařízení jsou připojena na sběrnice MISO (data ze slave do master zařízení) a MOSI (data z master do slave zařízení). Adresace konktrétního slave zařízení probíhá pomocí aktivace příslušného signálu CS, který je po skončení přenosu deaktivován. Přenos dat je synchronizován pomocí hodinového signálu, který se generuje z master zařízení a je běžně označován jako SCK. Běžné zjednodušení principu chování (graficky znázorněno pomocí 2.2) je, že se jedná o 2 posuvné registry – jeden v master zařízení a jeden v slave zařízení (obvykle jde o periferii). Pokud se například jedná na každé straně o registry délky 8 bitů, tak po 8 hodinových

taktech se vymění původní obsahy registrů, tedy původní data ze slave zařízení se přesunou do registru v master zařízení a naopak. Příslušné sběrnice a signály byly v implementovaném řešení po zvolení periferií[4] namapovány na GPIO piny čipu EM9304 tak, jak je popsáno v tabulce 2.2.

SPI signál	SCK	MOSI	MISO	CS1 (senzor)	CS2 (paměť)
GPIO pin	7	8	9	10	11

Tabulka 2.2: Zvolené GPIO piny čipu EM9304 pro rozhraní SPI

Standard SPI komunikace od firmy Motorola [15] také specifikuje formát přenosu dat. Jedná se zejména o nastavení známá jako CPOL a CPHA. CPOL určuje polaritu hodinového signálu. Pokud je CPOL = 0, hodinový signál je aktivní v logické 1; u CPOL = 1 je aktivní v logické 0. CPHA určuje okamžik vzorkovaní a vystavování nové hodnoty. Pokud je CPHA = 0, vzorkování nastává při přechodu z klidové do aktivní úrovně hodin a vystavení nové hodnoty na sběrnici při přechodu z aktivní do klidové úrovně; u CPHA = 1 se data vzorkují při přechodu z aktivní do klidové úrovně a nová vystavují při přechodu z klidové do aktivní úrovně. V implementaci této práce se používá typické nastavení v podobě CPOL = 0 a CPHA = 0. Toto nastavení bylo zvoleno později, podle kompatibility s vybranými periferiemi[4].



Obrázek 2.3: Režimy SPI přenosu s různým nastavením CPOL a CPHA, převzato z [3]

[4]Výběr periferií je popsán v 2.2.1.2 a 2.2.1.3.

2.2.1.2 Senzory[5]

Na trhu existuje mnoho senzorů s podporou rozhraní SPI, které umějí měřit různé z požadovaných veličin – teplotu, atmosférický tlak a relativní vlhkost vzduchu. Jedním z mála, který integruje všechny tři veličiny při zachování cenové dostupnosti, je čip BME280 od firmy BOSCH.

Čip pracuje v napěťovém rozsahu od 1.71 V do 3.6 V a v teplotách od −40 °C do 85 °C.

Tento senzor podporuje obě sériová rozhraní, mezi kterými probíhalo rozhodování, tedy plně podporuje I^2C až do rychlosti 3.4 MHz i SPI. V případě SPI senzor umožňuje sériovou komunikaci až do přenosové rychlosti 10 MHz. Jakmile po startu čipu dojde k prvnímu přechodu signálu CSB z logické 1 do logické 0, deaktivuje se rozhraní I^2C a aktivuje se rozhraní SPI, které zůstane zvolené až do resetu čipu. Senzor podporuje 2 SPI režimy – CPOL = 0, CPHA = 0 a CPOL = 1, CPHA = 1. Režim je automaticky vybrán podle polarity hodinového signálu SCK po přechodu CS z logické 1 do 0.

Příznivě vychází i spotřeba[6] – proud ve fázi spánku dosahuje pouze 0.1 µA. Při doporučené konfiguraci pro monitorování počasí výrobce uvádí, že průměrná spotřeba dosahuje 0.16 µA.

Čip BME280 dokáže po startu operovat v těchto 3 režimech:

Sleep mode
: Režim spánku s minimální spotřebou, kdy senzor neprovádí žádnou aktivitu.

Normal mode
: Režim ve kterém se provádí cyklické měření v definovaném intervalu.

Forced mode
: Dočasný režim ve kterém čip provede jednorázové měření po obdržení požadavku.

Perioda měření byla v implementaci zvolena dle doporučeného nastavení na 1/60 Hz, tedy jedno měření za minutu. Pro toto využití je s důrazem na maximální úsporu energie jednoznačně nejvýhodnější využít forced mode. Z režimu sleep mode se čip dostane do režimu forced mode poté, co přijme specifický příkaz, následně provede měření dle parametrů v konfiguraci a po naměření a uložení výsledků se přepne zpět do režimu sleep mode.

Normal mode je nevýhodný z několika vážných důvodů. Maximální nastavitelná perioda měření je jen něco málo přes 1000 ms, což by znamenalo, že v každém měřícím cyklu by se až 59 měření bez užitku zmařilo, protože by během těchto měření čip spotřebovával znatelně více energie, než při spánku. I pokud by bylo možné nastavit periodu na 60 s, spotřeba čipu mezi měřeními

[5] Informace v této sekci čerpány z [16].

[6] Udávané hodnoty proudu u tohoto senzoru jsou typické hodnoty za pokojové teploty, platné pro celý napěťový rozsah čipu.

by v režimu normal mode dosahovala typicky 0.2 µA, zatímco při použití režimu forced mode je spotřeba mezi měřeními pouhých 0.1 µA. Využitím forced módu jsou měření synchronizována z čipu EM9304, který navíc disponuje přesným hodinovým zdrojem. Ten je využit pro generování periodických událostí pro vykonání měření.

Interakce s čipem probíhá přímým zápisem nebo čtením z registrového pole. Pro zápis do registrového pole pomocí SPI je nejprve nutné aktivovat signál $\overline{\text{CS}}$ (v [16] označen jako CSB). Následně je nutné odeslat bajt s adresou, kde nejvýznamnější bit má hodnotu 0, následovaný bajtem s hodnotou, která má být na místo určené adresou zapsána. Poté je možné kroky popsané v předchozí větě opakovat tolikrát, kolik je potřeba odeslat bajtů. Zápis nepodporuje samostatnou inkrementaci adresy a je potřeba ji vždy explicitně uvést. Po skončení zápisu se signál $\overline{\text{CS}}$ musí deaktivovat.

Pro čtení z registrového pole je nejprve nutné aktivovat signál $\overline{\text{CS}}$. Poté se odešle bajt s adresou, od které má čtení začít, kde nejvýznamnější bit je roven hodnotě 1 a následně je odesláno tolik bajtů, kolik je potřeba přečíst. Data se souběžně s těmito bajty, jejichž hodnota se ignoruje, zasílají na stranu master zařízení. Po skončení čtení se musí signál $\overline{\text{CS}}$ deaktivovat.

Data v registrovém poli, jejichž význam je rozebrán pod tabulkou, mají následující strukturu:

Adresa	**Obsah bajtu**
0xFE	hum_lsb<7:0>
0xFD	hum_msb<7:0>
0xFC	temp_xlsb<7:4>
0xFB	temp_lsb<7:0>
0xFA	temp_msb<7:0>
0xF9	press_xlsb<7:4>
0xF8	press_lsb<7:0>
0xF7	press_msb<7:0>
0xF5	t_sb<7:5>, filter<4:2>, spi3w_en[0]
0xF4	osrs_t<7:5>, osrs_p<4:2>, mode<1:0>
0xF3	measuring[3], im_update[0]
0xF2	osrs_h<2:0>
0xE1 ... 0xF0	kalibrační data
0xE0	reset<7:0>
0xD0	chip_id<7:0>
0x88 ... 0xA1	kalibrační data

Tabulka 2.3: Registrové pole senzoru BME280

temp_xlsb, temp_lsb, temp_msb

Naměřená nezpracovaná hodnota teploty. Celkem má 20 bitů, z nichž

nejnižší 4 jsou v registru temp_xlsb, pokud se používá vyšší přesnost měření, jinak je možné tyto 4 bity nahradit 0. Prostředních 8 bitů je v registru temp_lsb a 8 nejvyšších je v registru temp_msb.

press_xlsb, press_lsb, press_msb
Naměřená nezpracovaná hodnota atmosférického tlaku. Celkem má 20 bitů. Nejnižší 4 jsou v registru press_xlsb, pokud se používá vyšší přesnost měření; jinak je možné tyto 4 bity nahradit 0. Prostředních 8 bitů je v registru press_lsb a 8 nejvyšších je v registru press_msb.

hum_lsb, hum_msb
Naměřená nezpracovaná hodnota vlhkosti. Celkem má 16 bitů. hum_lsb obsahuje 8 nejnižších bitů a hum_msb 8 nejvyšších bitů.

osrs_t, osrs_p, osrs_h
Nastavení přesnosti pomocí oversamplingu – osrs_t patří k teplotě, osrs_p k tlaku a osrs_h k vlhkosti. Hodnoty odpovídají mocninám 2 zmenšené o 1, tedy 1 až 5 určuje poměr oversamplingu (1x až 16x). Hodnota 0 znamená, že se daná veličina neměří.

t_sb
Nastavení standby času pro periodické měření v režimu normal mode. Hodnoty jsou mezi 0.5 ms a 1000 ms.

filter
Nastavení IIR filtru pro zvýšení přesnosti měření teploty a tlaku.

spi3w_en
Nastavení SPI na režim, kdy se ke komunikaci využívají jen 3 signály namísto běžných 4.

mode
Nastavení režimu, ve kterém má pracovat senzor. Hodnota 00 odpovídá režimu sleep mode, hodnota 11 režimu normal mode a 10 i 01 odpovídá režimu forced mode.

measuring
Bit který určuje, zda probíhá měření.

im_update
Bit který určuje, zda probíhá kopírování dat do registrového pole z nevolatilní paměti senzoru.

reset
Pokud se do tohoto registru zapíše hodnota 0xB6, provede se kompletní restart čipu.

chip_id
Z tohoto registru se po dokončení resetovací procedury dá přečíst hodnota 0x06. Tímto registrem je možné kontrolovat, zda je čip aktivován.

kalibrační data
V těchto registrech jsou uložena kalibrační data pro jednotlivé měřené veličiny. S využitím těchto dat se upravují nezpracovaná data pomocí speciálních kompenzačních rovnic. Konkrétní podoba těchto rovnic je rozebrána v [16].

2.2.1.3 Paměť[7]

Na trhu existuje velké množství EEPROM pamětí, mezi kterými jsou díky jednostrannému využití jen velmi nepatrné rozdíly. Jako paměť pro ukládání naměřených hodnot byl zvolen model EEPROM paměti 25AA1024 od firmy Microchip. Tento konkrétní čip byl zvolen zejména kvůli podobnému napěťovému rozsahu, jako má hlavní čip EM9304.

Paměť musela být natolik velká, aby dokázala pojmout až několik jednotek dnů stará naměřená data. Tato paměť disponuje 128 kB paměti, což zajišťuje dostatek prostoru pro uložení výsledků – v implementovaném řešení se jedná konkrétně o necelých 6 dní historie dat. Paměť by za běžných podmínek měla vydržet 1 milión zápisových cyklů.

Výrobce uvádí, že při napětí 2.5 V dosahuje proud při čtení a zápisu 5 mA, v pohotovostním režimu 12 µA a v hlubokém spánku (deep power-down) 1 µA. Vzhledem k relativně vysoké spotřebě v pohotovostním režimu je potřeba, aby paměť byla co nejdelší dobu v režimu hlubokého spánku. Při očekávaném využití meteostanice je důležité, aby spotřeba byla co nejmenší zejména v době, kdy neprobíhá žádné čtení dat ze strany klientské aplikace. Po získání naměřených dat ze senzoru je nutné vzbudit paměť z hlubokého spánku, povolit zápis, zapsat upravené naměřené údaje do paměti a paměť opět převést do režimu hlubokého spánku.

Paměť umožňuje ochranu pro zápis různých částí pamětí pomocí ochrany sektorů, které paměť rovnoměrně dělí na 4 části. Nastavení ochrany je kombinované pomocí zápisu do STATUS registru a binárního vstupu $\overline{\text{WP}}$. V implementaci této práce se možnosti ochrany proti zápisu nevyužívá.

Komunikace s pamětí probíhá zasláním 1 bajtu instrukce, který je ve specifických případech doprovázen dodatečnými parametry.

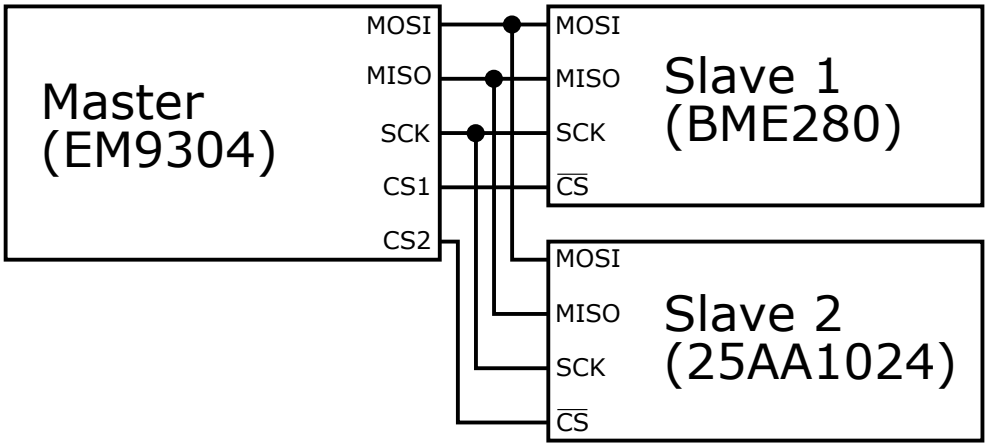
Instrukce	**Binární formát**	**Význam**
READ	0000 0011	Čtení z paměti od specifikované adresy
WRITE	0000 0010	Zápis do paměti od specifikované adresy
WREN	0000 0110	Povolení zápisu do paměti
WRDI	0000 0100	Zrušení povolení zápisu do paměti
RDSR	0000 0101	Čtení ze STATUS registru
WRSR	0000 0001	Zápis do STATUS registru
PE	0100 0010	Smazání specifikované stránky
SE	1101 1000	Smazání specifikovaného sektoru
CE	1100 0111	Smazání obsahu celého čipu
RDID	1010 1011	Návrat z režimu deep power-down
DPD	1011 1001	Přechod do režimu deep power-down

Tabulka 2.4: Podporované instrukce paměti 25AA1024

[7] Informace v této sekci čerpány z [17].

Pro čtení z paměti je po aktivaci signálu $\overline{\text{CS}}$ nutné poslat instrukci READ, následovanou 24bitovou adresou, která specifikuje, od které adresy se má začít číst a bajty s libovolnou hodnotou ve stejném počtu, jako je těch, které se mají z paměti přečíst. Hodnota těchto bajtů se ignoruje a souběžně se zasílají požadovaná data na stranu master zařízení. V posledním kroku se signál $\overline{\text{CS}}$ deaktivuje.

Zápis do paměti probíhá v podobném formátu. Po aktivaci signálu $\overline{\text{CS}}$ se pošle instrukce WRITE, následovaná 24bitovou adresou, odkud se má zapisovat a samotná data, která se mají do paměti zapsat. Nakonec se deaktivuje signál $\overline{\text{CS}}$. Důležité omezení, které je nutné mít na paměti je to, že celý zápis musí být proveden v rámci jedné fyzické stránky; data která by při zápisu překročila hranici stránek se nezapíšou. Velikost stránky je 256 B.



Obrázek 2.4: Schéma finálního zapojení přes SPI

2.2.2 Systém

Jedním ze zásadních rozhodnutí byla volba vhodné GATT služby. BLE standard nabízí službu *Environmental Sensing*, která obsahuje charakteristiky pro předávání teploty, tlaku i vlhkosti. Tato služba ale poskytuje jen jednu hodnotu, obvykle aktuální, respektive poslední naměřenou, a neumožňuje zasílat historii naměřených dat, jak je požadováno v zadání. Z výše uvedených důvodů a také proto, že služba zatím není součástí SDK pro čip EM9304, není služba *Environmental Sensing* vhodná.

Jinou možností je použití proprietárního řešení v podobě *Alpwise Data Exchange*. Tato služba umožňuje přijímat až 20 bajtů „surových dat“ pomocí zápisu (write / write no response) do charakteristiky a posílat také 20 bajtů pomocí notifikace nebo indikace (characteristic notification / indication), jejichž chování je detailněji popsáno v 1.1.7. Volba této proprietární služby poskytla dostatečnou volnost v nastavení průběhu komunikace, naproti tomu

tato skutečnost znamenala, že nebude možné využít aplikace třetí strany na klientské straně. Ty totiž většinou implementují jen služby definované standardem BLE. Bylo tedy nutné vytvořit vlastní klientskou aplikaci pro příjem a vizualizaci dat. Popis chování a implementace této klientské aplikace jsou detailně rozebrány v 3.2.

Komunikační protokol vypadá následovně. Při požadavku o získání dat je ze strany klienta posláno 10 bajtů, jak je zobrazeno v tabulce 2.5, z nichž prvních 6 má hodnoty (hexadecimálně) - 67 65 74 74 70 68, což v ASCII kódování odpovídá *gettph*, které jsou následovány dvěma 16 bitovými čísly. Tato čísla jsou odeslána ve formátu little endian. První číslo udává počet požadovaných naměřených hodnot (např. 5) a druhé jejich rozestup v minutách (např. 15 minut nebo 3 hodiny = 180 minut). Meteostanice následně vrátí až tolik paketů, kolik je požadováno. Pokud je naměřeno méně dat, než kolik je požadováno, meteostanice vrátí maximální možný počet validních dat.

6 B	2 B	2 B
gettph	Počet hodnot	Rozestup hodnot

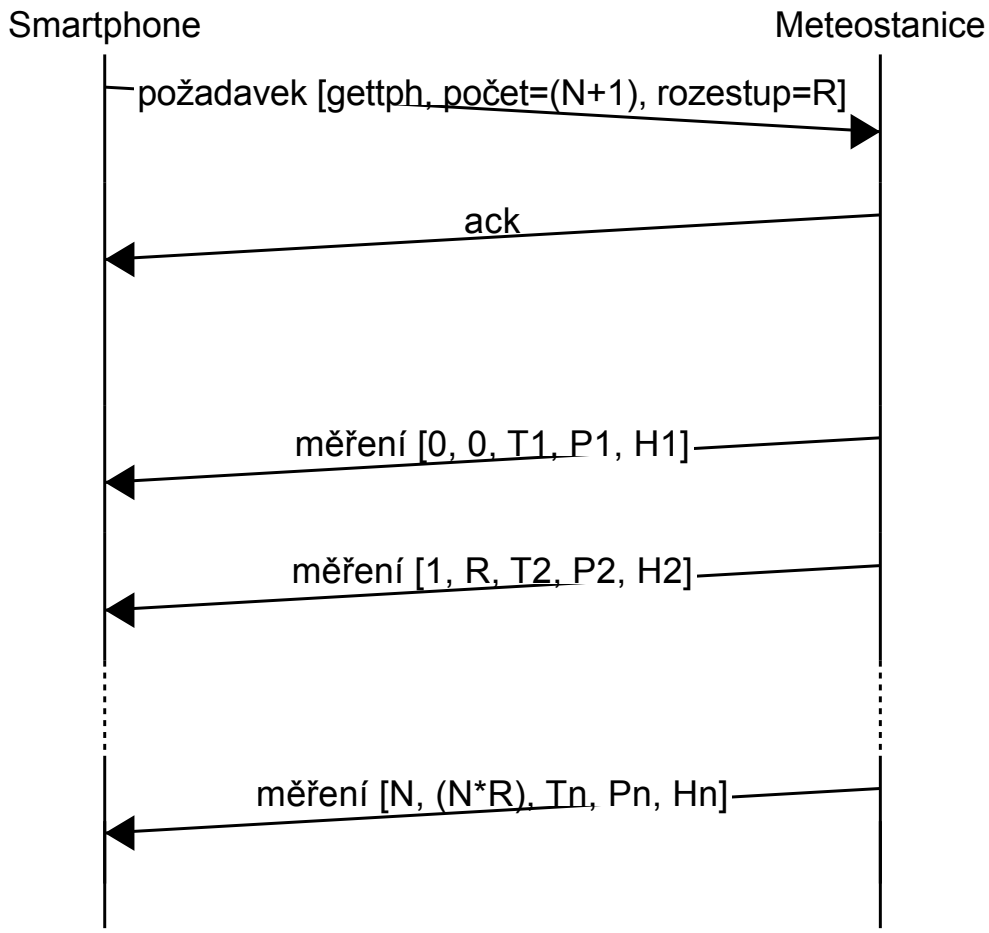
Tabulka 2.5: Požadavek - struktura přijímaných dat

Naměřené hodnoty se klientovi posílají v paketech, jejichž datová struktura je naznačena v tabulce 2.6. První 4 bajty dat z meteostanice udávají číslo paketu (0, 1, 2, ...). Další 4 bajty vyjadřují delta čas od měření zaslaného jako paket s číslem 0, například delta čas 15 znamená, že data z tohoto paketu byla zaznamenána o 15 minut dříve než paket s číslem 0. Zbylých 12 bajtů je po 4 bajtech rozděleno na teplotu, tlak a relativní vlhkost. Tyto 4bajtové hodnoty se odesílají ve formátu little endian. Hodnoty jsou vzhledem k použití celočíselné reprezentace u teploty 100x vyšší a u relativní vlhkosti 1024x vyšší než výsledné hodnoty. To znamená, že například hodnota 2278 u teploty odpovídá 22.78 °C, hodnota 101325 u tlaku odpovídá 101 325 Pa a u vlhkosti hodnota 78152 odpovídá 76.32 %. Výslednými hodnotami, vůči kterým jsou vztaženy násobky, jsou myšleny stupně Celsia, pascaly a procentuální vyjádření relativní vlhkosti.

4 B	4 B	4 B	4 B	4 B
Číslo paketu	Delta čas	Teplota	Tlak	Vlhkost

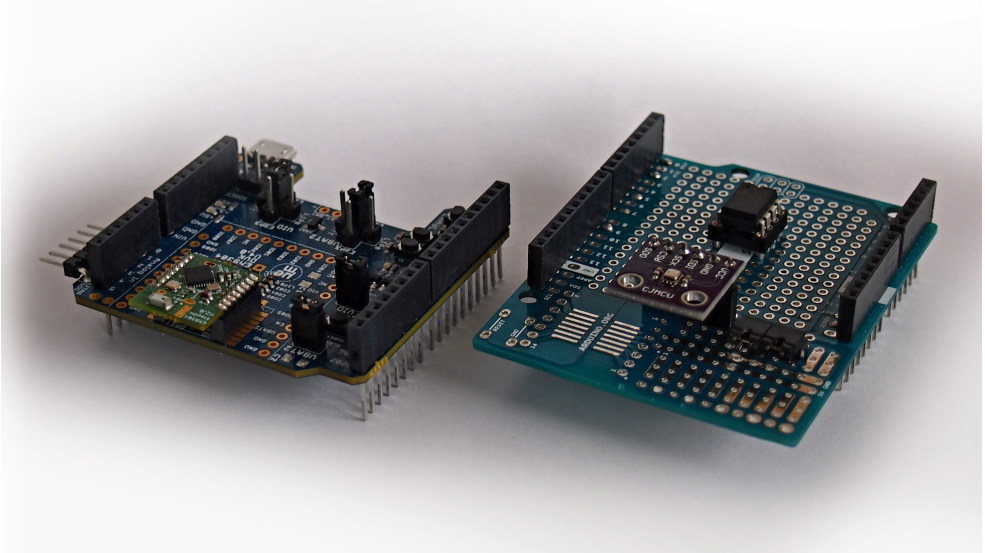
Tabulka 2.6: Odpověď - struktura odesílaných dat

V souvislosti se zachováním co nejnižší spotřeby bylo nutné upravit dva zásadní parametry BLE komunikace ovlivňující aktivitu čipu – interval vysílání advertising paketů a interval spojení. Vzhledem k tomu, že povaha aplikace předpokládá, že drtivou většinu času nebude k meteostanici připojený žádný klient, je zásadnější hodnotou interval odesílání advertising paketů. Pro zachování co nejnižší spotřeby je ideální dosáhnout co nejdelšího advertising



Obrázek 2.5: Schéma komunikace

intervalu, protože celá aplikace může díky tomu strávit delší dobu v režimu sleep mode. Na druhou stranu je z uživatelského hlediska potřeba, aby při skenování byly pakety zachyceny v přijatelně dlouhé době. Stejný požadavek je na dobu navazování spojení se zařízením, které je také ovlivněno délkou advertising intervalu. Po důkladném testování různých hodnot vyšel jako optimální interval délky 1025 ms, který představuje přijatelný kompromis mezi responzivitou aplikace a co nejdelším časem, kdy může aplikace být v režimu sleep mode. Ze stejných důvodů bylo u komunikačního intervalu rozhodnuto pro hodnotu mezi 100 ms a 125 ms; konkrétní hodnota je upřesněna klientským zařízením.



Obrázek 2.6: Hardwarový prototyp meteostanice, na kterém probíhal pozdější vývoj softwaru (vlevo deska DVK s čipem EM9304, vpravo deska se zapojenou pamětí a senzorem)

2.2.3 Klientská aplikace

Vzhledem k tomu, že v případě BLE služeb byla zvolena cesta proprietárního řešení, bylo nutné vytvořit vlastní aplikaci pro klientskou stranu. Zadání dávalo možnost volby mezi aplikací pro PC a smartphone. Z praktického pohledu využití vycházela jako nejlepší varianta aplikace pro smartphone. Smartphone je v dnešní době pro velkou část veřejnosti stále běžnějším zařízením a získání dat z meteostanice pomocí odladěné specializované aplikace je pohodlné a otázkou několika málo vteřin.

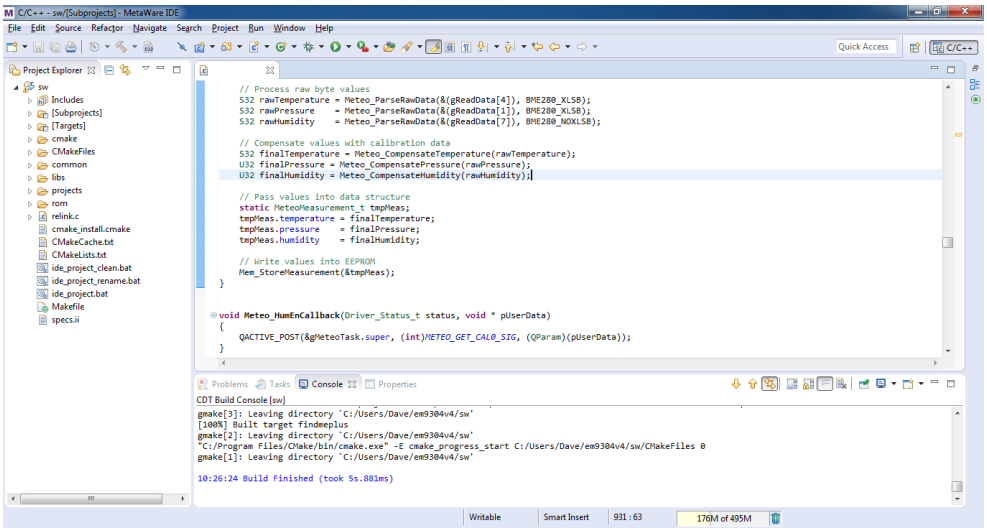
Jako platforma pro aplikaci byl zvolen operační systém Android, který autor využívá i pro osobní potřebu. Jako testovací zařízení v průběhu vývoje a testování aplikace posloužil smartphone LG G4 (H815). Společnost Google navíc nabízí ukázkovou aplikaci využívající BLE konektivity na veřejně přístupném repozitáři pod svobodnou licencí Apache 2.0, která umožňuje využití poskytovaného kódu pro odvozená díla. [18] Při vývoji se tedy vycházelo z této aplikace, tudíž nebylo potřeba vše vytvářet od úplného začátku.

Vývoj aplikací pro platformu Android probíhá zejména v jazyku Java a uživatelské prostředí se popisuje pomocí značkovacího jazyka XML. Vývojářům jsou k dispozici rozsáhlé knihovny, včetně těch, které jsou určené pro obsluhu BLE komunikace, zejména pro vrstvu GATT. Knihovní třídy a funkce jsou detailně popsány na stránkách pro Android vývojáře. [19]

Kapitola 3

Realizace

3.1 Software meteostanice



Obrázek 3.1: Vývojové prostředí MetaWare IDE, ve kterém probíhal vývoj softwaru pro meteostanici

Při vývoji bylo postupováno metodou shora dolů (top-down). To znamená, že vývoj začal implementací vyšších úrovních aplikace, a pak se postupně přesouval do nižších vrstev a směrem k hardwaru. Například hardwarové periferie byly ze začátku vývoje simulovány softwarově. V případě simulovaného senzoru byla aplikaci poskytována pseudonáhodná environmentální data, která byla ukládána a čtena ve vyhrazeném bloku operační paměti reprezentující EEPROM paměť. Jakmile byly vyladěné a otestované vyšší vrstvy – obslouužení komunikace a způsob uchování dat, vývoj se přesunul k implementaci reálných hardwarových periferií a řešení specifických hardwarových problémů

na ještě nižších úrovních.

Při přechodu k implementaci obsluhy hardwarových periferií se jako první řešil senzor BME280, který byl oproti EEPROM paměti znatelně složitější. U tohoto senzoru je nutné pro zpracování dat získat po startu čipu kalibrační data, která efektivně obsazují 32 bajtů dat v registrovém poli senzoru a využívají se pro přepočítání „surových dat“, poskytovaných senzorem jako výsledek měření, na reálné hodnoty. Implementace obsluhy EEPROM paměti se na první pohled zdála jednoduchá, ale zvláště kvůli vysoké spotřebě v pohotovostním režimu (popsáno v 2.2.1.3), bylo nutné přistoupit k probouzení před každým přístupem k paměti a následným uspáním po dokončení komunikace. Vzhledem k tomu, že do synchronních měření vstupují i asynchronní události v podobě požadavků na čtení dat z paměti, bylo nutné ošetřit aby nedocházelo k situacím, kdy by jedna z těchto událostí uspala čip během komunikace s pamětí iniciované druhou událostí. Před uspáním paměti do režimu deep power-down se kontroluje, zda souběžně neprobíhá jiná komunikace s pamětí, například probíhající čtení dat ze smartphonu po ukončení ukládání nových dat z periodického měření; v případě tohoto konfliktu se uspání neprovede.

Aplikace po startu provádí nezbytnou inicializaci a následně většinu času vykonává dvě hlavní úlohy – každou minutu provádí měření nových dat a na požadavek klienta zasílá naměřená data. Níže jsou rozebrány jednotlivé procedury hlavního toku programu pro danou dílčí úlohu, které respektují pořadí vykonávání v kódu. Následně jsou popsány některé další funkce a struktury, které jsou podpůrně využívány během chodu programu. V místech, kde je použita výpustka (...), jsou většinou parametry, které souvisí se systémem QP-nano a pro pochopení významu funkcí nejsou podstatné.

3.1.1 Inicializace

3.1.1.1 Meteo_EntryFunction()

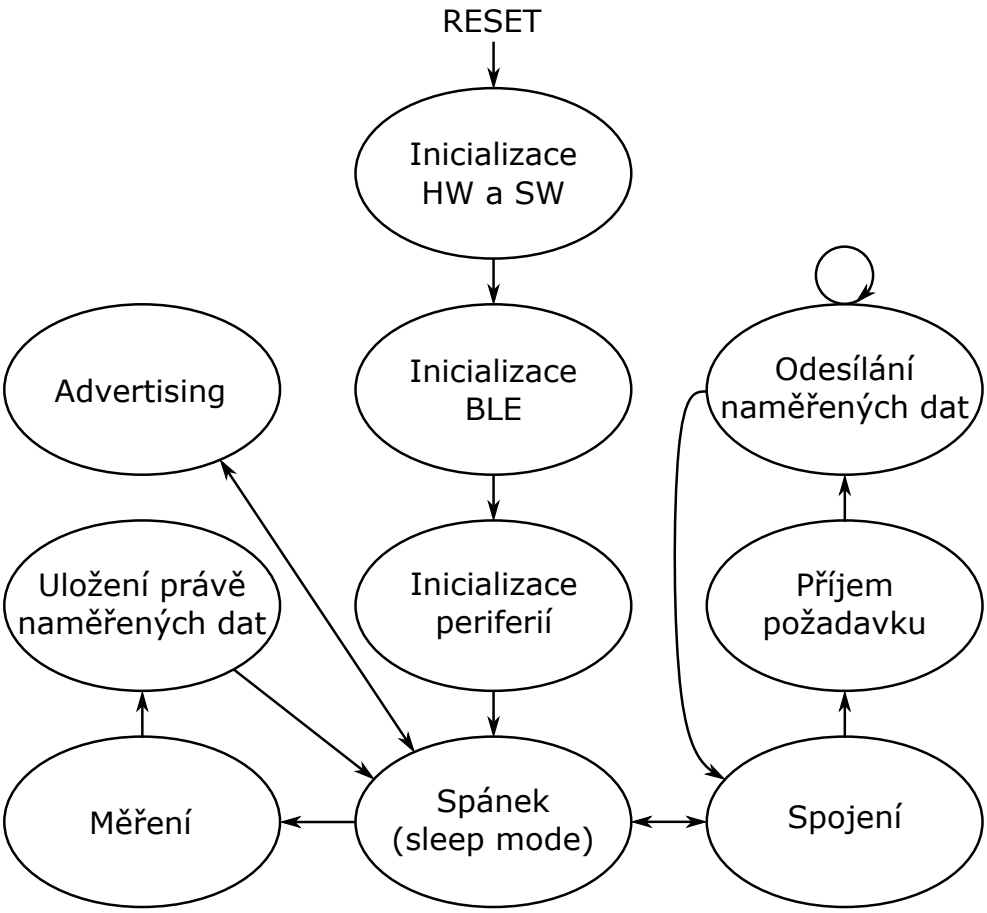
Vstupní funkce se vykonává jako první při nastartování čipu, buď po resetu, nebo opuštění režimu sleep mode. Po resetu se provádí inicializace systému QP-nano a inicializace hodinových zdrojů, časovače a modulu pro SPI komunikaci. Také se zde nainicializují důležité globální proměnné.

3.1.1.2 Meteo_Init(...)

Provede inicializaci GPIO 5 a 6, které se využívají pro debugovací účely a nastaví callback (zpětné volání) pro jádro BLE.

3.1.1.3 Meteo_WaitForStack(...)

Počká na dokončení inicializace BLE stacku, nastaví zobrazované jméno zařízení, zaregistruje se GATT služba *Alpwise Data Exchange* a nastaví další callbacky obsluhující BLE komunikaci.



Obrázek 3.2: Stavový diagram popsiující chování aplikace (detaily a přechody vysvětleny v 3.1.1, 3.1.2 a 3.1.3)

3.1.1.4 Meteo_WaitForRandomAddress(...)

Vygeneruje a nastaví náhodnou adresu zařízení.

3.1.1.5 Meteo_InitPeripherals(...)

V první řadě přes SPI nastaví parametry pro měření na senzoru, následně se přečtou a uloží jeho kalibrační údaje. Dále se inicializuje modul RTC, který obstarává periodické generování signálů pro měření každou minutu. Po zpracování nastaví v jakém intervalu se bude provádět takzvaný advertising. Nakonec se přejde do hlavního stavu aplikace, kde se obsluhují různé požadavky, vysílá advertising, nebo je zařízení v režimu sleep mode.

3.1.2 Měření

Měření se vyvolá událostí, kterou každých 60 s generuje RTC modul.

3.1.2.1 Meteo_PrepareMeasurement(...)

Přes SPI se do senzoru BME280 zašle příkaz, který čip přepne do režimu forced mode, ve kterém proběhne jednorázové měření.

3.1.2.2 Meteo_WaitForData(...) + Meteo_WaitForDataCallback(...)

Pomocí časovače zařídí, aby se počkalo určitý počet milisekund, než bude k dispozici výsledek měření vyvolaný v předchozí funkci.

3.1.2.3 Mem_WakeUp(U32 param)

Přes SPI se pošle do EEPROM paměti příkaz na probuzení z režimu deep power-down. Parametr určuje, kam bude běh programu pokračovat po dokončení procesu probouzení, to znamená pro zápis naměřených dat nebo čtení dat vyvolané klientským zařízením.

3.1.2.4 Mem_WakeUpDelay(...) + Mem_WakeUpDelayCallback(...)

Zařídí pomocí časovače, aby se počkalo 0.1 milisekundy na korektní dokončení procesu probouzení.

3.1.2.5 Mem_PrepareStore(...)

Pomocí příkazu zaslaného do EEPROM přes SPI se nastaví, aby se do paměti dalo zapisovat. Toto je potřeba provést před každým zápisem do paměti.

3.1.2.6 Mem_PrepareStoreCallback(...)

Přečte pomocí SPI naměřená data ze senzoru.

3.1.2.7 Meteo_GetDataCallback(...)

Zpracují se přečtená data a upraví se pomocí kalibračních parametrů, aby odpovídala reálným hodnotám.

3.1.2.8 Mem_StoreMeasurement(MeteoMeasurement_t * pMeas)

Zapíše upravená zpracovaná data na příslušnou adresu do paměti.

3.1.2.9 Mem_StoreMeasurementCallback(...)

Posune a nastaví novou adresu EEPROM paměti, do které se bude zapisovat následující měření a v případě, že neprobíhá vyčítání dat ze smartphonu, se paměť uspí zpět do režimu deep power-down.

3.1.3 Přenos dat

Přenos dat se vyvolá příjmem požadavku, při kterém je vyvolána událost indikující zápis do charakteristiky poskytované službou *Alpwise Data Exchange*, která slouží pro přenos dat ze strany klienta na stranu serveru.

3.1.3.1 Meteo_DataExchangeTx(...)

Zpracuje požadavek z klientského zařízení; pokud vyhovuje požadovanému formátu, nainicializují se parametry pro přenos.

3.1.3.2 Mem_WakeUp(U32 param) + Mem_WakeUpDelay(...) + Mem_WakeUpDelayCallback(...)

Chování funkcí je popsáno v 3.1.2.3 a 3.1.2.4.

3.1.3.3 Meteo_SendData(...)

Přečte data z aktuální požadované pozice v paměti. Pokud požadovaná data nejsou dosud dostupná nebo se odeslala všechna potřebná, ukončí se přenos.

3.1.3.4 Meteo_SendDataCallback(...)

Posune pozici v paměti na následující požadovanou pozici, to znamená že se posune o tolik pozic v paměti, jaký je požadovaný rozestup vyčítaných dat a dekrementuje počet zbývajících dat k přenesení. Přečtená data z předchozí funkce pošle přes *Alpwise Data Exchange* do klientské aplikace.

Po odeslání se pokračuje do funkce Meteo_SendData(...).

3.1.3.5 Meteo_TerminateTransport()

Ukončí aktuální probíhající přenos dat do klientského zařízení. K tomu dojde, pokud se odešlou všechna požadovaná, respektive maximum z dostupných a požadovaných dat, nebo když se přeruší BLE spojení se zařízením. Tato funkce uspí paměť zpět do režimu deep power-down, pokud právě neprobíhá ukládání dat z periodického měření.

3.1.4 Podpůrné funkce

3.1.4.1 S32 Meteo_CompensateTemperature(S32 v_uncomp_temperature_S32)

Pomocí získaných kalibračních parametrů upraví surová data ze senzoru teploty na reálnou hodnotu, kterou vrací v celočíselném formátu odpovídající stupňům Celsia, vynásobenou konstantou 100.

3.1.4.2 U32 Meteo_CompensatePressure(S32 v_uncomp_pressure_S32)

Pomocí získaných kalibračních parametrů a dílčího výsledku z úpravy teploty přepočítá surová data atmosférického tlaku na reálnou hodnotu, kterou vrací v celočíselném formátu v pascalech.

3.1.4.3 U32 Meteo_CompensateHumidity(S32 v_uncomp_humidity_S32)

Pomocí získaných kalibračních parametrů a dílčího výsledku z úpravy teploty přepočítá surová data vlhkosti vzduchu na reálnou hodnotu, kterou vrací v celočíselném formátu v procentech relativní vlhkosti, vynásobenou konstantou 1024.

3.1.4.4 Meteo_UpdateConnection(. . . , void *pUserData)

Upraví komunikační interval pro dané spojení.

3.1.4.5 S32 Meteo_ParseRawData(U8 * pData, int xlsb)

Převede surová data ze senzoru na 4 bajty dlouhé slovo pro následné zpracování.

3.1.4.6 S32 Mem_GetOlderIndex(MeteoTransport_t * pTrans)

Vrátí následující požadovanou pozici v paměti, kterou požaduje spojení předané v parametru funkce, odkud se bude číst následující zasílaná naměřená hodnota.

3.1.5 Datové struktury

3.1.5.1 Memory_t

Struktura, která udržuje pozici počáteční a koncové pozice v EEPROM paměti uvažovaná jako kruhový buffer. Také obsahuje flag, který udává, zda právě probíhá zápis do paměti.

3.1.5.2 MeteoTransport_t

Tato struktura udržuje informace o aktuálně probíhajícím spojení – celkový a zbývající počet požadovaných dat, aktuální pozice pro čtení z EEPROM paměti, rozestup (interval) dat a handle daného BLE spojení.

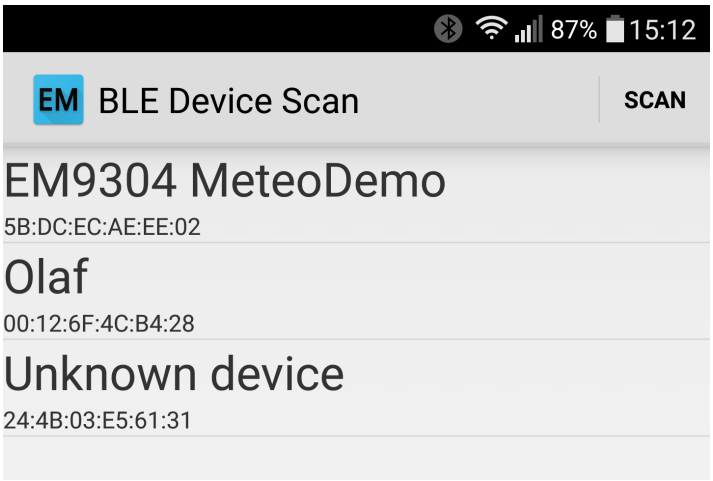
3.1.5.3 BMECalibration_t

V této struktuře jsou uložena kalibrační data ze senzoru, která se do této struktury načtou přes SPI během inicializace periferií po resetu zařízení.

3.2 Klientská aplikace

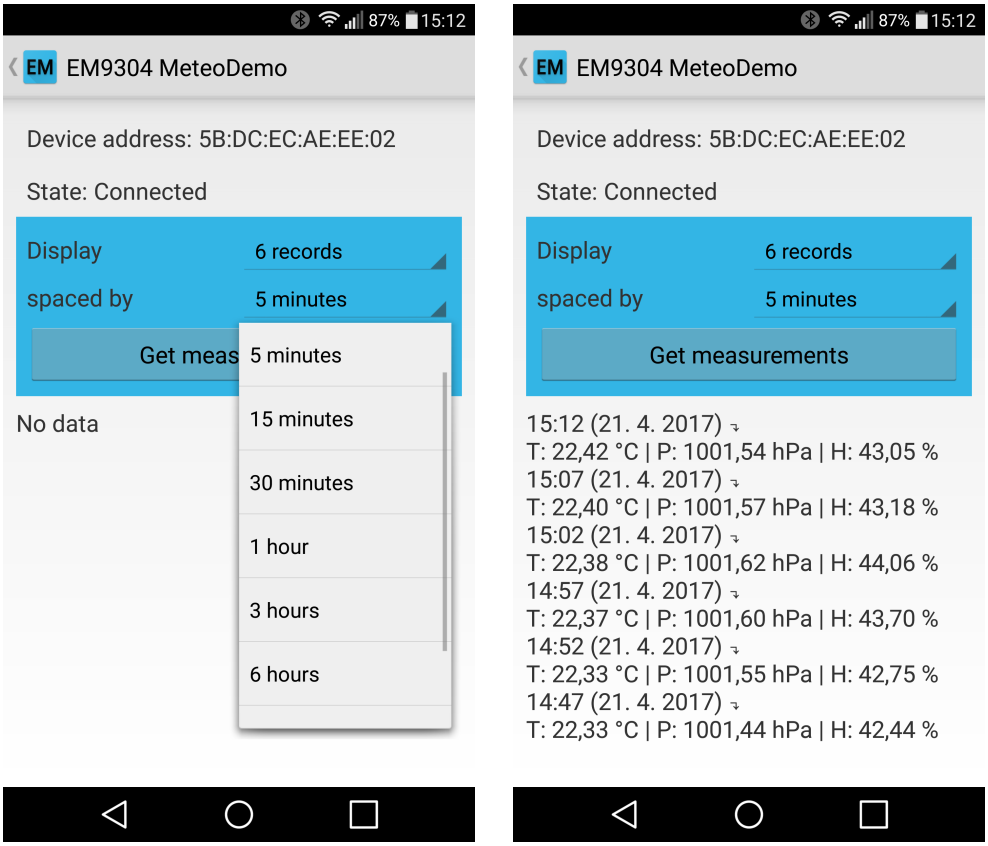
3.2.1 Popis chování

Uživatelské rozhraní aplikace se skládá ze dvou hlavních obrazovek, kterým se v systému Android říká aktivita. První aktivita, ve které se aplikace nachází po spuštění, zobrazuje viditelná zařízení v dosahu a v přehledném seznamu je zobrazuje s jejich jménem a adresou. Pomocí tlačítka *Scan* na horní liště se může opakovaně spustit prohledávání zařízení v okolí. Poté, co proběhne prohledávání zařízení, může uživatel po klepnutí na příslušné zařízení ze seznamu navázat spojení a následně ho obsloužit z druhé aktivity. Tato první aktivita byla vzhledem k univerzální funkčnosti (až na drobné změny) ponechána v původním stavu z ukázkové aplikace. [18]



Obrázek 3.3: Výřez z první aktivity aplikace

Hlavní díl práce souvisel z druhou aktivitou, kde se zařízení obsluhuje po vyhledání příslušného zařízení v první aktivitě a následném navázání spojení. V této aktivitě se v horní liště zobrazuje název zařízení a možnost odpojení s přechodem zpět na první aktivitu. Pod lištou je zobrazena adresa zařízení



Obrázek 3.4: Druhá aktivita – nastavení parametrů a čtení dat

v běžném hexadecimálním formátu a stav připojení. Ve zvýrazněném obdélníku má uživatel možnost nastavit, o jaká data má zájem. První pole určuje, kolik jednotlivých záznamů se má z meteostanice odeslat. Hodnota v druhém poli určuje časový rozestup mezi jednotlivými měřeními, která se budou zobrazovat. Po kliknutí na tlačítko *Get measurements* se do meteostanice odešle příkaz s příslušnými výše nastavenými parametry. Následně meteostanice začne odesílat požadovaná data, která se v průběhu přenosu začnou zobrazovat v textové formě ve spodní části obrazovky aktivity.

3.2.2 Implementace řešení

Podpora pro BLE službu *Alpwise Data Exchange* je součástí SDK pro čip EM9304, ale konkrétní detaily implementace v době vývoje v poskytované dokumentaci ani jinde na internetu nebyly vyhledatelné. Byla tedy využita jiná aplikace pro platformu Android, konkrétně *nRF connect* od firmy Nordic Semiconductor. Pomocí této aplikace bylo možné zjistit UUID služby *Alpwise Data Exchange* a obou jejích charakteristik včetně dalších detailů. Charakteristika pro zaslání dat do zařízení umožňuje potvrzovaný zápis (write) nebo nepotvr-

zovaný zápis (write no response). Charakteristika pro příjem dat ze zařízení umožňuje zasílání dat přes notifikaci nebo indikaci a má jeden deskriptor, konkrétně *Client Characteristic Configuration Descriptor*.

Pro běh aplikace byly jako nezbytné vytyčeny tři zásadní funkce aplikace: inicializace BLE komunikace, odeslání požadavku a příjem s následnou vizualizací dat. Následuje detailní popis těchto funkčních celků.

3.2.2.1 Inicializace

Při inicializaci se po vytvoření spojení vykoná metoda *initMeteo* ze třídy *DeviceControlActivity*. V této metodě se iteruje přes nalezené BLE služby, které jsou předány v jejím parametru *gattServices*. Pokud se nalezne služba *Alpwise Data Exchange*, proběhne nejdříve inicializace charakteristiky pro zápis, kde se lokálně nastaví typ zápisu na běžný write. Následně se nastaví charakteristika pro notifikace, kde se lokálně nastaví typ na notification a na stranu serveru se zapíše do *Client Characteristic Configuration Descriptor* a díky tomu se aktivuje zasílání notifikací na stranu klienta. [2]

3.2.2.2 Odeslání požadavku

Po vybrání parametrů požadovaných dat, která má meteostanice zapsat, se po klepnutí na tlačítko *Get measurements* vyvolá metoda *servicesGetButtonListener*, která je také součástí třídy *DeviceControlActivity*. V této metodě se načtou hodnoty počtu měření a jejich časový rozestup, které byly zadány v uživatelském rozhraní. Tyto hodnoty se připojí za 6 bajtů, které v ASCII kódování odpovídají *gettph*, v 16bitovém formátu s pořadím bajtů v little endian. Experimentálně bylo zjištěno, že procesor ARC EM4, na kterém běží software meteostanice, využívá ve výchozím nastavení pořadí bajtů v little endian, zatímco ARM procesor v čipsetu pro LG G4 – Qualcomm Snapdragon 808 používá pořadí bajtů v big endian, tudíž je nutná konverze vícebajtových typů.

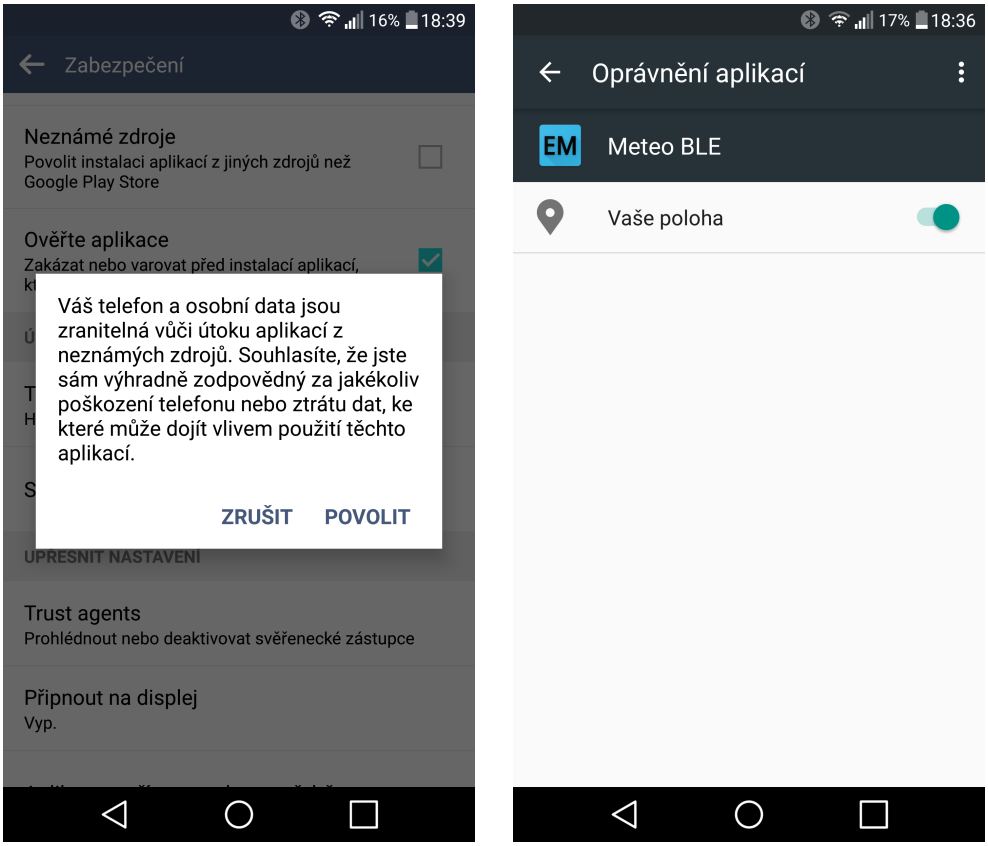
3.2.2.3 Příjem dat

Při přijetí notifikace se zavolá metoda *broadcastUpdate* ve třídě *BluetoothLeService*. V této metodě se validuje formát přijatých dat. Pokud mají přijatá data vyhovující formát, vytvoří se instance třídy *MeteoMeasurement*, které se v konstruktoru předává časová známka měření (časová známka přijetí nultého paketu - delta čas měření) a hodnoty teploty, tlaku a vlhkosti ve formátu, v jakém jsou přijaty z meteostanice. V konstruktoru se přijaté hodnoty převedou na čísla s pohyblivou řádovou čárkou, tedy v případě teploty a tlaku vydělí číslem 100 a v případě vlhkosti číslem 1024, aby uložená data byla v stupních Celsia, hektopascalech a procentech relativní vlhkosti. Každé přijaté měření se následně s využitím metody *toString* u třídy *MeteoMeasurement* vypíše

v textové formě do spodní části obrazovky aplikace, kde jsou naměřené hodnoty zobrazeny s přesností na 2 desetinná místa spolu s časem a kalendářním datem měření.

3.2.3 Kompatibilita a instalace

Aplikace byla testována na operačních systémech Android 4.4 a Android 6.0. Teoreticky by měla fungovat ve všech zařízeních s verzí systému od 4.4, která mají podporu Bluetooth dle specifikace od verze 4.0. Pro instalaci je nutné přenést APK soubor do paměti zařízení, v nastavení povolit instalaci aplikací ze zdrojů třetích stran a následně tento soubor otevřít. Pokud se aplikace úspěšně nainstaluje, je nutné si v nastavení aplikací ověřit, že aplikace má právo k přístupu k poloze zařízení. Bez tohoto oprávnění aplikace nepracuje korektně.

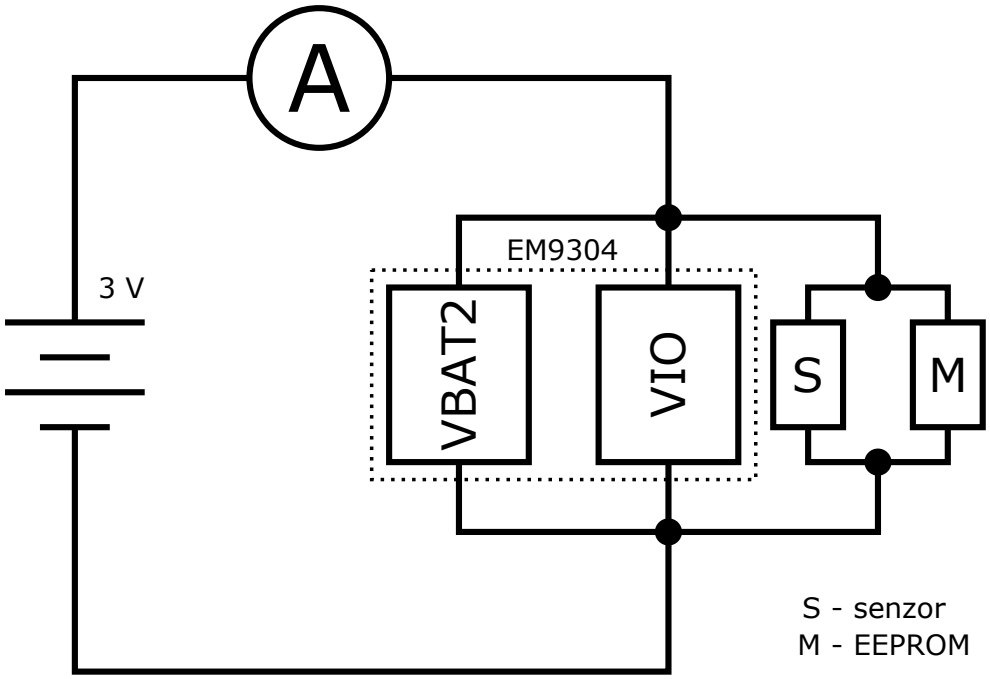


Obrázek 3.5: Nezbytná nastavení systému Android pro instalaci a správný běh klientské aplikace

Kapitola 4

Měření spotřeby

Ve chvíli, kdy byla vyvinuta meteostanice i klientská aplikace pro smartphone, bylo možné přejít k měření spotřeby. Software pro meteostanici, který se v průběhu vývoje a testování nahrával do části RAM paměti označené jako IRAM, bylo nutné nahrát do nevolatilní paměti OTP. Aktivovaná IRAM zvyšuje spotřebu čipu v řádu desítek mikroampérů a je tedy vhodná pouze pro účely vývoje. Deska, která je součástí hardwarového DVK, obsahuje několik indikačních LED. Měřicí obvod tedy bylo nutné zapojit tak, aby nebyl ovlivňován spotřebou součástek, které nejsou přímou funkční součástí aplikace.



Obrázek 4.1: Schéma zapojení pro měření proudu

U čipu EM9304 v režimu step-down, který se využívá v této práci, je samotný čip napájený z domény, která je označena jako VBAT2 a jeho vstupně-výstupní rozhraní je napájeno doménou VIO. [6] Periferie, tedy senzor BME280 a EEPROM paměť 25AA1024, jsou napájeny přímo z domény VIOP, ze které se následně odděluje již zmíněná doména VIO. Bylo proto nutné vytvořit takové zapojení, kde se společný zdroj přivede na VBAT2, VIO a samostatně k periferiím. Výsledné zapojení, v jakém probíhalo měření, je zobrazeno na schématu 4.1.

Měření probíhalo na analyzátoru Agilent N6705. Meteostanice byla napájena jedním z výstupů tohoto zařízení. Napájení bylo nastaveno tak, aby poskytovalo napětí 3 V, odpovídající běžně používané knoflíkové baterii. Protékající proud byl měřen na jednom ze vstupů analyzátoru, který byl nastaven do režimu měření proudu v čase.

Měření probíhalo ve 3 fázích aktivity:

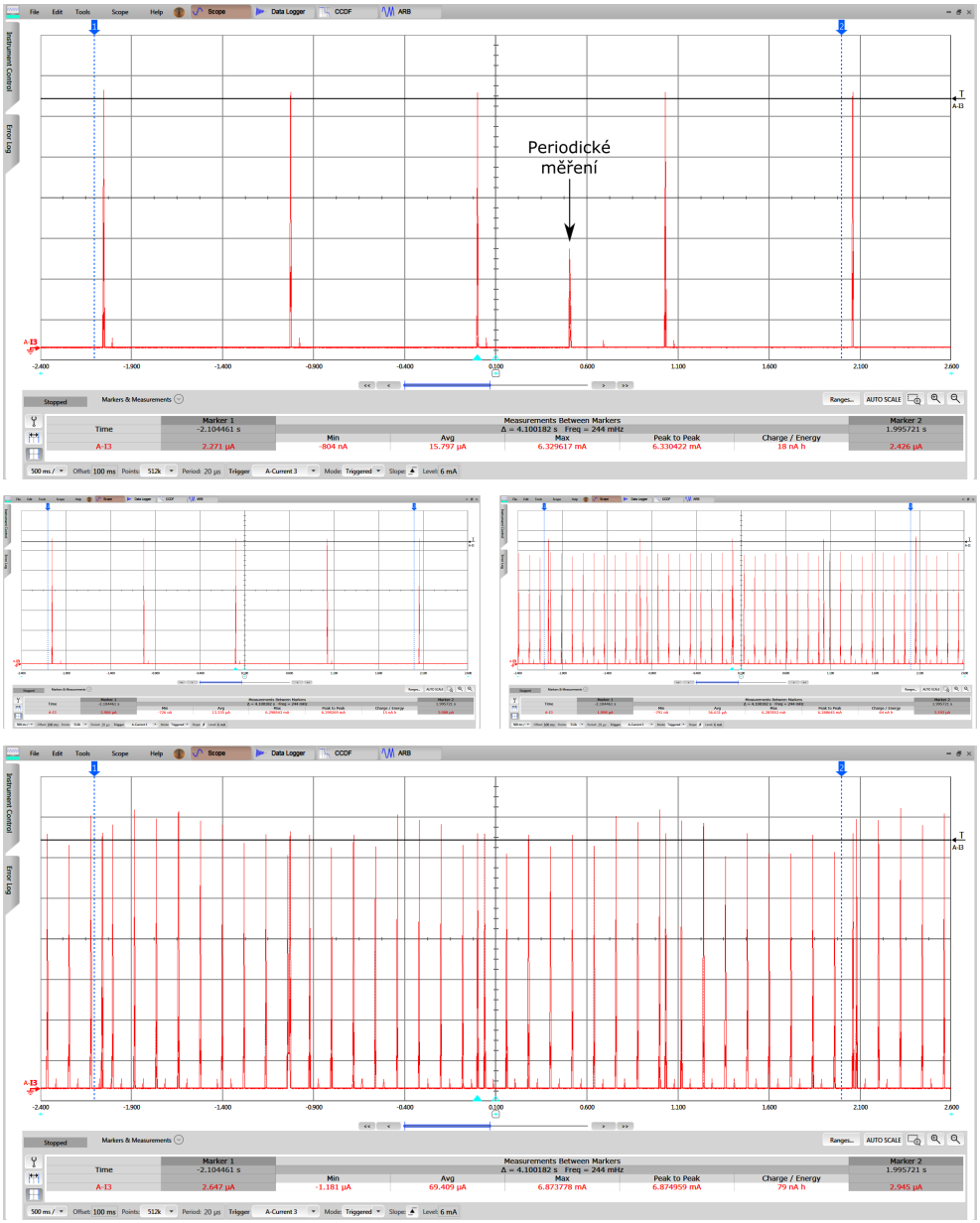
1. Žádné navázané spojení, pouze probíhá advertising a periodická měření
2. Pouze navázané spojení, bez aktivního zasílání požadavků
3. Navázané spojení s aktivním čtením naměřených hodnot z paměti na základě požadavku

Analyzátorem byla měřena průměrná spotřeba v časovém úseku dlouhém 4100 ms, což odpovídá přesně 4 periodám advertising intervalu. Všechna měření byla opakována 30krát, což představuje dostatečně velký statistický vzorek pro přesný odhad střední hodnoty. Vzhledem k tomu, že jednotlivá měření probíhala v časovém úseku dlouhém 4.1 s, bylo nutné výsledky upravit, aby reflektovaly proces měření hodnot (teploty, tlaku a vlhkosti), který nastává jen jednou za 60 s. V 1. fázi aktivity se provádělo měření jak na intervalu, ve kterém probíhal pouze advertising (I_a), tak na intervalu, kde do advertisingu vstupovalo i měření hodnot (I_{a+m}). Výsledná hodnota průměrně protékajícího proudu se vypočítala dle rovnice 4.1. V případě 2. a 3. fáze byl proud měřen s jedním připojeným klientským zařízením a pouze v časových úsecích, ve kterých neprobíhalo měření hodnot (I_c ve fázi 2 a I_{c+r} ve fázi 3). Spotřeba v okamžiku měření hodnot byla k naměřeným hodnotám proudu dodatečně připočtena, jak ukazují rovnice 4.2 a 4.3. Advertising během spojení stále probíhá. Meteostanice umožňuje připojení až 4 klientským zařízením najednou.

$$I_{f1} = \frac{4.1 \cdot I_{a+m} + 55.9 \cdot I_a}{60} \tag{4.1}$$

$$I_{f2} = I_c + (I_{f1} - I_a) \tag{4.2}$$

$$I_{f3} = I_{c+r} + (I_{f1} - I_a) \tag{4.3}$$



Obrázek 4.2: Ilustrace z měření časového průběhu spotřeby proudu: fáze 1 – I_{a+m} (nahoře), fáze 1 – I_a (uprostřed vlevo), fáze 2 – I_c (uprostřed vpravo), fáze 3 – I_{c+r} (dole)

$$I_{typ} = \frac{(1440 - (t_2 + t_3)) \cdot I_{f1} + t_2 \cdot I_{f2} + t_3 \cdot I_{f3}}{1440} \tag{4.4}$$

$$I_{typ} = \frac{1437 \cdot I_{f1} + (6 \cdot \frac{25}{60}) \cdot I_{f2} + (6 \cdot \frac{5}{60}) \cdot I_{f3}}{1440} \tag{4.5}$$

Označení měření / fáze	**Průměrný protékající proud**	
Advertising	I_a	13.355 µA
Advertising včetně měření	I_{a+m}	15.689 µA
Spojení	I_c	56.031 µA
Spojení včetně čtení dat	I_{c+r}	69.486 µA
Fáze 1	I_{f1}	13.514 µA
Fáze 2	I_{f2}	56.190 µA
Fáze 3	I_{f2}	69.645 µA

Tabulka 4.1: Naměřené a přepočtené hodnoty průměrně protékajícího proudu

Jako typický scénář využití aplikace je možné uvažovat 6 připojení k meteostanici za den (1440 minut), každé trvající 30 sekund, z nichž 5 sekund probíhá aktivní čtení naměřených dat. Do obecnější rovnice 4.4 je tedy dosazeno $t_2 = 6 \cdot (\frac{25}{60})$ a $t_3 = 6 \cdot (\frac{5}{60})$, jak ukazuje rovnice 4.5. V případě tohoto scénáře využití je průměrná spotřeba (I_{typ}), vypočtená pomocí rovnice 4.5, 13.608 µA. Na základě tohoto scénáře je v tabulce 4.2 uvedena teoretická výdrž meteostanice při různých variantách napájení z baterií.

Baterie	**Kapacita** při 3 V	**Výdrž meteostanice**
CR2032 (knoflíková)	220 mAh	~ 1 rok a 10 měsíců
2x AAA („mikrotužková“)	1000 mAh	~ 8 let a 4 měsíce
2x AA (tužková)	2500 mAh	~ 20 let a 11 měsíců

Tabulka 4.2: Výdrž meteostanice při napájení z baterií

Závěr

Cílem této bakalářské práce byla implementace funkční meteostanice postavené na čipu EM9304, která bude schopna pomocí BLE zasílat naměřená data připojenému klientskému zařízení. Výsledkem této práce je funkční prototyp meteostanice, ke které je možné se připojit smartphonem pomocí aplikace pro systém Android, která vznikla v rámci této práce. V této aplikaci je možné nastavit jaká data se mají z meteostanice získat a přijatá data následně přehledně zobrazit.

Při vývoji meteostanice byl kladen důraz na nízkou spotřebu energie. Výsledky měření spotřeby ukazují, že při napětí 3 V a reálném scénáři využití v průběhu dne dosahuje průměrná spotřeba meteostanice přibližně 13.6 µA. Tato hodnota představuje velmi dobrý výsledek – při napájení běžnou knoflíkovou baterií typu CR2032 (3 V, 220 mAh) by zařízení mělo teoreticky vydržet v provozu více než 1 rok a 10 měsíců. Nízké spotřeby bylo dosaženo zejména díky intenzivnímu využívání režimů nízké spotřeby (sleep mode) u čipu EM9304 a u připojených periferií, volbě vhodného sériového rozhraní a optimalizací délek intervalů BLE komunikace.

Výsledky měření také ukázaly, že zásadní dopad na spotřebu meteostanice má fáze, ve které není připojené žádné zařízení a meteostanice pouze vysílá tzv. advertising pakety a provádí periodická měření (teploty, tlaku a vlhkosti). Spotřebu energie by bylo možné dále snížit prodloužením advertising intervalu, avšak na úkor uživatelské přívětivosti (delší doba pro vyhledání meteostanice a navázání spojení). Jinou možností, díky které by se dal ušetřit až 1 µA, by bylo kompletní odpojování napájení EEPROM namísto režimu deep power-down. Toto řešení by však vyžadovalo další HW modifikace.

Klientskou aplikaci by bylo možné do budoucna dále vyvíjet a rozšířit například o zobrazování přijímaných hodnot do grafu s časovou osou.

Literatura

[1] Bluetooth SIG: Bluetooth Core Specification. [online], ©2017, [cit. 2017-05-13]. Dostupné z: https://www.bluetooth.com/specifications/bluetooth-core-specification

[2] Bluetooth SIG: *Core Version 4.2 [online]*. 2014, [cit. 2017-05-13]. Dostupné z: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439

[3] Burnett, C. M. L.: SPI bus timing diagram. [online ilustrace], 2010, [cit. 2017-05-13]. Dostupné z: https://commons.wikimedia.org/wiki/File:SPI_timing_diagram2.svg

[4] Swan, M.: Sensor Mania! The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0. *Journal of Sensor and Actuator Networks*, ročník 2012, č. 1, 2012-11-08: s. 217–253, ISSN 2224-2708, doi:10.3390/jsan1030217, [cit. 2017-05-13]. Dostupné z: http://www.mdpi.com/2224-2708/1/3/217

[5] EM Microelectronic-Marin: *EM9301 Data Sheet [online]*. 2016, [cit. 2017-05-13]. Dostupné z: http://www.emmicroelectronic.com/sites/default/files/public/products/datasheets/9301-ds.pdf

[6] EM Microelectronic-Marin: *EM9304 Data Sheet [online]*. 2017, [cit. 2017-05-13]. Dostupné z: http://www.emmicroelectronic.com/sites/default/files/public/products/datasheets/9304-ds_0.pdf

[7] O'Brien, T.: IPhone 4S claims title of first Bluetooth 4.0 smartphone, ready to stream data from your cat. [online], 2011, [cit. 2017-05-13]. Dostupné z: https://www.engadget.com/2011/10/12/iphone-4s-claims-title-of-first-bluetooth-4-0-smartphone-ready

[8] Grabianowski, E.: Is Wibree going to rival Bluetooth? [online], 2006, [cit. 2017-05-13]. Dostupné z: http://electronics.howstuffworks.com/wibree.htm

[9] Heydon, R.: *Bluetooth low energy*. Upper Saddle River (NJ): Prentice Hall, 2012, ISBN 9780132888363.

[10] Bluetooth SIG: Adopted Specifications. [online], ©2017, [cit. 2017-05-13]. Dostupné z: https://www.bluetooth.com/specifications/adopted-specifications

[11] Abraham, J.: Understanding Bluetooth Advertising Packets. [online], 2014, [cit. 2017-05-13]. Dostupné z: http://j2abro.blogspot.cz/2014/06/understanding-bluetooth-advertising.html

[12] Bluetooth SIG: Generic Attributes (GATT) and the Generic Attribute Profile. [online], ©2017, [cit. 2017-05-13]. Dostupné z: https://www.bluetooth.com/specifications/generic-attributes-overview

[13] Quantum Leaps: QP-nano. [online], ©2017, [cit. 2017-05-13]. Dostupné z: https://state-machine.com/qpn

[14] Mead, M.: Hollywood Principle – Don't Call Us, We'll Call You! [online], 2008, [cit. 2017-05-13]. Dostupné z: http://matthewtmead.com/blog/hollywood-principle-dont-call-us-well-call-you-4

[15] Motorola: *SPI Block Guide [online]*. 2003, [cit. 2017-05-13]. Dostupné z: https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf

[16] Bosch Sensortec: *BME280: Final data sheet [online]*. 2015, [cit. 2017-05-13]. Dostupné z: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280_DS001-11.pdf

[17] Microchip Technology: *25AA1024 Data Sheet [online]*. 2015, [cit. 2017-05-13]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/20001836J.pdf

[18] The Android Open Source Project: Android BluetoothLeGatt Sample. [software], 2016, [cit. 2017-05-13]. Dostupné z: https://github.com/googlesamples/android-BluetoothLeGatt

[19] The Android Open Source Project: Android Developers. [online], ©2017, [cit. 2017-05-13]. Dostupné z: https://developer.android.com

Příloha **A**

Seznam použitých zkratek

BLE Bluetooth Low Energy

SoC System on chip

SIG (Bluetooth) Special interest group

HCI Host controller interface

GFSK Gaussian frequency-shift keying

CRC Cyclic redundancy check

UART Universal asynchronous receiver/transmitter

USB Universal serial bus

SDIO Secure digital input output

L2CAP Logical link control and adaptation protocol

SM Security manager

GAP Generic access profile

ATT Attribute profile

GATT Generic attribute profile

HW Hardware

SW Software

AES Advanced encryption standard

CCM Counter with cipher block chaining message authentication code

UUID Universally unique identifier

ACI Application command interface

RISC Reduced instruction set computing

I^{2}C Inter-integrated circuit

SPI Serial Peripheral Interface

ROM Read-only memory

RAM Random-access memory

OTP One-time programmable (memory)

EEPROM Electrically erasable programmable read-only memory

GPIO General-purpose input/output

DVK Development kit

SDK Software development kit

A/D Analogově digitální

PC Personal computer

SDA Serial data line

SCL Serial clock line

MOSI Master output slave input

MISO Master input slave output

SCK Serial clock

CS Chip select

CPOL Clock polarity

CPHA Clock phase

ASCII American standard code for information interchange

ack Acknowledgement

XML Extensible markup language

IDE Integrated development environment

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `compiled` ..... adresář s přeloženými programy implementace
- `src`
  - `client-app` ..... zdrojové kódy implementace klientské aplikace
  - `latex` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP-SAFRATA-David-LS2017.pdf` ..... text práce ve formátu PDF