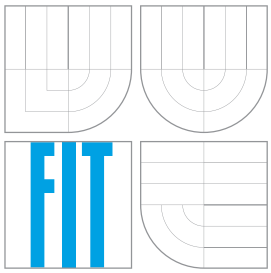
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

MONITOROVÁNÍ NEELEKTRICKÝCH VELIČIN S VYUŽITÍM BLUETOOTH LOW ENERGY

MEASUREMENT AND MONITORING OF NON-ELECTRICAL QUANTITIES USING BLUETOOTH LOW ENERGY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETER PRÍTEL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZDENĚK VAŠÍČEK, Ph.D.

BRNO 2016

Abstrakt

Táto práca je zameraná na technológiu Bluetooth Low Energy a jej využitie pre komunikáciu so snímačmi neelektrických veličín. Práca popisuje všeobecné vlastnosti technológie Bluetooth. V praktickej časti sa práca venuje aplikácii založenej na frameworku Cordova. Výsledná aplikácia demonštruje komunikáciu so snímačmi prostredníctvom technológie Bluetooth Low Energy.

Abstract

The aim of this report is Bluetooth Low Energy and communication with sensors of non-electrical quantities. The first part describes general overview of Bluetooth technology. In the implementation part, development of application based on Cordova framework is described. The application demonstrates communication with sensors using Bluetooth Low Energy.

Klíčová slova

bluetooth, bluetooth low energy, BLE, bluetooth 4.0, UI, Apache Cordova, multiplatformová aplikácia

Keywords

bluetooth, bluetooth low energy, BLE, bluetooth 4.0, UI, Apache Cordova, cross-platform application

Citace

Peter Prítel: Monitorování neelektrických veličin s využitím Bluetooth Low Energy, bakalářská práce, Brno, FIT VUT v Brně, 2016

Monitorování neelektrických veličin s využitím Bluetooth Low Energy

Prohlášení

Vyhlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Zdeňka Vašíčka, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Prítel
18.5.2016

Poděkování

Ďakujem svojmu vedúcemu Ing. Zdeňkovi Vašíčkovi, Ph.D. za odborné vedenie tejto práce.

© Peter Prítel, 2016.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

V tejto práci sa venujem technológii Bluetooth Low Energy a jej využitiu v oblasti snímačov. Dnešné trendy nositeľnej elektroniky a športového príslušenstva komunikujúceho s mobilným telefónom si vyžiadali bezdrôtový prenos dát. Špecifikom týchto aplikácií je nízka spotreba energie. Naopak prenosová rýchlosť nie je hlavným parametrom.

Technológia Bluetooth prešla od svojho vzniku v roku 1994 vývojom a je neustále vylepšovaná. Pojem Bluetooth Low Energy sa poprvýkrát objavil v špecifikácii Bluetooth verzie 4.0 v roku 2010. Hlavným prínosom Bluetooth Low Energy je výrazne znížená energetická náročnosť dátových prenosov. Technológia Bluetooth Low Energy sa začala rýchlo uplatňovať v pomerne novom odvetví nositeľnej elektroniky. Medzi najpopulárnejšiu nositeľnú elektroniku patria športové monitorovacie náramky, ktoré obsahujú snímače neelektrických veličín ako snímače tepu, gyroskop, akcelerometer, teplomer. Dáta z týchto snímačov generujú malé objemy dát, ktoré je potrebné čo najefektívnejšie bezdrôtovo preniesť do mobilného telefónu alebo iného zariadenia. Technológia Bluetooth Low Energy sa stala štandardom práve pre tento typ aplikácii.

Bluetooth Low Energy má z pohľadu vývojára širokú softvérovú aj hardvérovú podporu naprieč všetkými modernými operačnými systémami. Trend vo vývoji aplikácii smeruje k multiplatformovej podpore jednej aplikácie. Takýto model značne znižuje náklady na vývoj a náročnosť údržby aplikácie. V práci sa venujem možnostiam vývoja multiplatformových aplikácii s podporou Bluetooth Low Energy. V práci som sa primárne zameral na framework Cordova a jeho využitie s technológiou Bluetooth Low Energy.

Kapitola 2

Bluetooth

Bluetooth je štandard pre bezdrôtovú komunikáciu. Má za úlohu prepojiť na krátku vzdialenosť dve alebo viac elektronických zariadení. Bluetooth technológia využíva vlny s frekvenciou od 2.4 do 2.485 GHz.

V tejto kapitole bude popísaná história[3][2], princípy a fungovanie technológie Bluetooth.

2.1 História

- 1994 – Švédska firma Ericsson bezdrôtový štandard Bluetooth ako alternatívu k štandardu RS-232
- 1998 – Intel, Toshiba, Nokia, IBM a Ericsson Technology Licensing sformovali The Bluetooth Special Interest Group (SIG)
- 1999 – vydaná špecifikácia Bluetooth 1.0
- 2000 – na trh prichádza prvý mobilný telefón, prvá PC karta, a prvý headset s technológiou Bluetooth
- 2001 – na trh prichádza prvá tlačiareň, laptop, handsfree kit do auta s technológiou Bluetooth
- 2002 – Bluetooth v1.1, na trh prichádza prvá klávesnica a myš, GPS príjimač, digitálna kamera s technológiou Bluetooth, počet Bluetooth certifikovaných zariadení dosiahol počet 500. V tejto verzii boli odstránené mnohé chyby z verzie 1.0, bola pridaná možnosť využívať nešifrované kanály a bol pridaný indikátor sily signálu RSSI[1].
- 2003 – Bluetooth v1.2, prvý mp3 prehrávač s technológiou Bluetooth. V tejto verzii technológia Bluetooth poskytovala rýchlejšie pripojenie a odhaľovanie zariadení. Zvýšila sa rýchlosť prenosu dát až do 721 kb/s oproti predošlej verzii. Taktiež boli predstavené dva nové módy protokolu L2CAP[2] – Flow Control Mode a Retransmission Mode.

[1] Received Signal Strength Indicator
[2] Logical Link Control and Adaptation

- 2004 - Bluetooth v2.0 + EDR[3] je verzia ktorá podporuje EDR pre rýchlejší prenos dát. Udávaná prenosová rýchlosť bola 3Mbit/s pričom réalna prenosová rýchlosť bola 2.1Mbit/s.
- 2007 – Bluetooth v2.1 + EDR – v tejto špecifikácii bola zavedená vlastnosť Secure Simple Pairing (SSP), ktorá zlepšovala párovanie zariadení a zvyšovala bezpečnosť tohto procesu. Ďalším vylepšením bolo Extended Inquiry Response (EIR), ktoré poskytovalo viac informácií pri procese dotazovania popísaného v kapitole 2.3.1 a umožnilo lepšie filtrovanie zariadení pred pripojením
- 2009 – Bluetooth v3.0 + HS[4]. HS poskytuje teoretické prenosové rýchlosti až 24 Mbit/s hoci Bluetooth je využívaný na vyjednávanie a dohodnutie parametrov prenosu a dáta sú prenášané cez súbežné spojenie 802.11 známe aj ako Wi-Fi.
- 2010 – Bluetooth v4.0 – Tento štandard zahŕňa protokoly klasického Bluetooth, Bluetooth HS a nové protokoly BLuetooth Low Energy
- 2013 – Bluetooth v4.1 – oproti v4.0 sa jedná o softwarový update ktorý zvyšuje použiteľnosť zariadení. Update zahŕňal podporu pre LTE[5] a Bulk data exchange rates.
- 2014 – Bluetooth v4.2 – v tejto verzii bola uvedená niektorá funkcionalita potrebná pre IoT[6]. Niektoré nové vlastnosti vyžadujú aktualizáciu hardware, ako napríklad Data Length Extension, pričom pri niektorých by mala stačiť softwarová aktualizácia.

2.2 Technická špecifikácia

2.2.1 Frekvencia a sila signálu

Technológia Bluetooth je založená na bezdrôtovom prenose za pomoci využitia rádiových vĺn. Každé Bluetooth zariadenie musí byť schopné posielať a prijímať rádiové vlny v pásme 79 rôznych frekvencií od 2.4 GHz do 2.485 GHz. Šírka pásma sa v niektorých krajinách líši. V tabuľke 2.1 sú uvedené šírky pásma pre jednotlivé krajiny.

Krajina	Rozsah frekvencie	RF kanály	
Európa a USA	2400 - 2483.5 MhZ	f = 2402 + k MhZ	k = 0,...,78
Japonsko	2471 - 2497 MhZ	f = 2473 + k MhZ	k = 0,...,22
Španielsko	2445 - 2475 MhZ	f = 2449 + k MhZ	k = 0,...,22
Francúzsko	2446.5 - 2483.5 MhZ	f = 2454 + k MhZ	k = 0,...,22

Tabulka 2.1: Šírky pásiem v jednotlivých krajinách[4]

Nie každé zariadenie potrebuje vysielať na veľkú vzdialenosť, prípadne zariadenie potrebuje mať čo najnižšiu spotrebu energie. Podľa sily vysielaného signálu zariadenia delíme do troch skupín, ako môžme vidieť v tabuľke 2.2. Niektoré moduly dokážu operovať v jednej triede, kým ostatné môžu meniť svoj vysielací výkon.

[3] Enhanced data rate
[4] High-speed
[5] Long-Term Evolution
[6] Internet of Things

Trieda	Maximálny vysielací výkon mW(dBm)	Približná vzdialenosť
Trieda 1	100 mW (20 dBm)	100 metrov
Trieda 2	2.5 mW (4 dBm)	10 metrov
Trieda 3	1 mW (0dBm)	1 meter

Tabulka 2.2: Rozdelenie do tried podľa vysielacieho výkonu[5]

2.2.2 Bluetooth protocol stack

Medzi protokoly ktoré sa radia do jadra Bluetooth špecifikácie patria:

- Radio protokol (RF)
- Link controll protokol (LC)
- Link manager protokol (LM)
- Logical link control and adaptation protocol (L2CAP)

Protokolový zásobník technológie Bluetooth môžeme rozdeliť na dve hlavné časti:

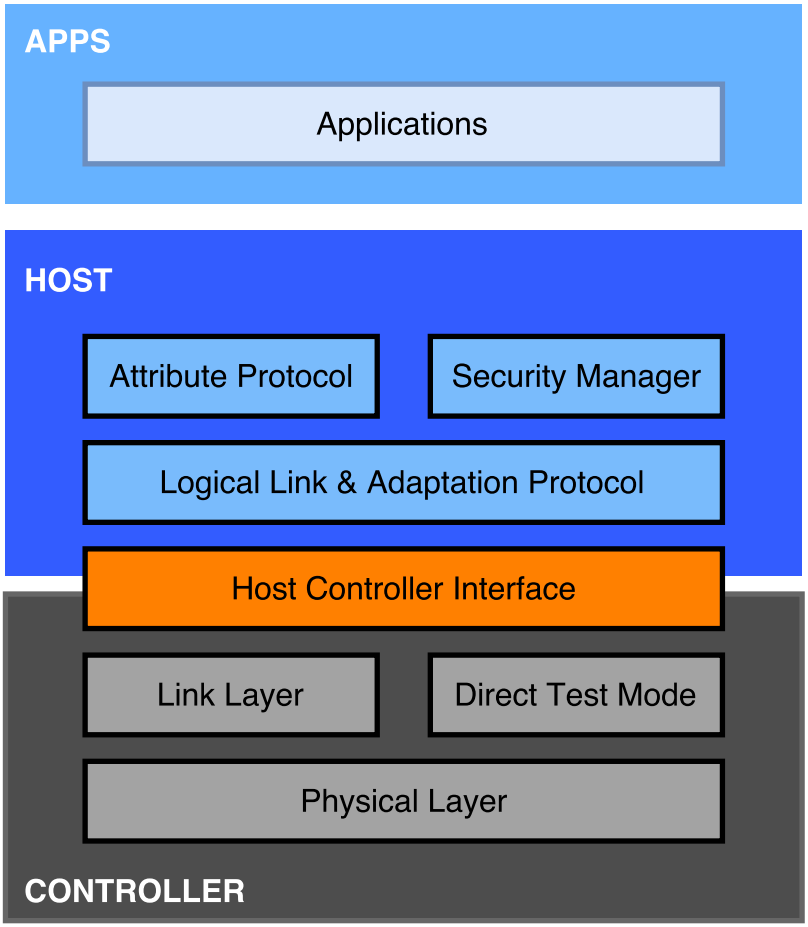
- Bluetooth controller stack – do tejto časti patria tri spodné protokoly (RF, LC, LM), ktoré ovládajú rádiový čip. Controller je väčšinou implementovaný v lacnom kremíkovom zariadení obsahujúcim Bluetooth rádiový čip a mikroprocesor.
- Bluetooth host stack – protokoly v tejto časti majú za úlohu pracovať s dátami na vyššej úrovni ako protokoly v controller stack. Tieto protokoly sú väčšinou implementované ako časť operačného systému. Pri integrovaných zariadeniach, ako napríklad Bluetooth slúchadlo, môžu oba tieto protokolové zásobníky byť implementované na jednom rovnakom mikroprocesore. Takáto implementácia vytvára takzvaný hostless systém.

Controller môže byť v jednej z nasledujúcich troch konfigurácií:

- BR[7]/EDR
- LE[8]
- kombinovaný BR/EDR controller a LE controller, s jednou Bluetooth adresou ktorú zdieľa kombinovaný controller

Bluetooth špecifikácia umožňuje spoluprácu medzi systémami, definovaním protokolových správ, ktoré sú zasielané medzi ekvivalentnými vrstvami. Taktiež umožňuje spoluprácu medzi nezávislými subsystémami za pomoci rozhrania medzi Bluetooth controller a Bluetooth host časťami. Bluetooth controller a Bluetooth host využívajú rozhranie – Host to Controller Interface (HCI) – cez ktoré medzi sebou obojsmerne komunikujú. Na obrázku 2.1 sú znázornené jednotlivé vrstvy Bluetooth protocol stack, ktoré budú popísané v nasledujúcich kapitolách.

[7] Basic Rate
[8] Low Energy



Obrázek 2.1: Bluetooth protocol stack

- Physical (PHY) Layer – vrstva, ktorá kontroluje prenos a prijímanie rádiových vĺn za pomoci komunikačných kanálov.
- Link Layer – definuje štruktúru a kanály paketov, proces odhaľovania a pripájania a proces odosielania a prijímania dát.
- Direct Test Mode – poskytuje možnosť testovania, umožňuje dať pokyn Physical vrstve preniesť alebo prijať danú sekvenciu paketov. Príkazy sa môžu odosielať napríklad cez HCI.
- Host to Controller Interface – štandardné rozhranie medzi Bluetooth controller a Bluetooth host subsystémami.
- Logical Link Control and Adaptation Protocol Layer – paketovo orientovaný protokol, ktorý prenáša pakety na HCl alebo priamo na Link Manager pri hostless systémoch. Podporuje protokolový multiplexing, segmentáciu paketov, reassembling paketov a dokáže doručiť informácie o QoS[9] vyšším vrstvám.
- Attribute Protocol – definuje klient-server protokol pre výmenu dát po úspešnom spojení zariadení. Atribúty sú zoskupené do služieb s využitím GATT[10]. ATT sa

[9] Quality of Service
[10] Generic Attribute Progfile

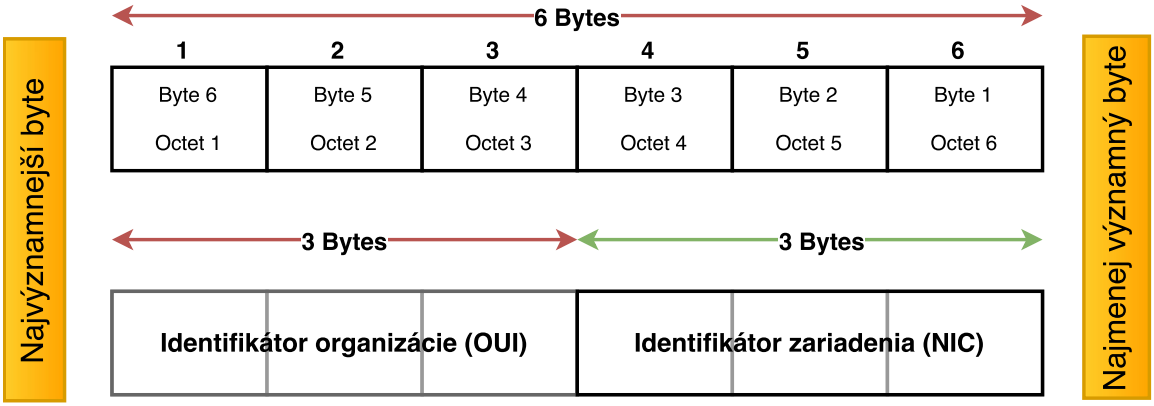
využíva hlavne v LE implementáciach a občas v implementáciach BR/EDR.

- Security Manager – definuje protokol a správanie ktoré spravuje párovanie zariadení, overovanie a šifrovanie medzi zariadeniami a poskytuje bezpečnostné funkcie, ktoré využívajú ostatné komponenty pre rôzne druhy použitia.

Vrstvy Generic Attribute Profile a Generic Access Profile sa využívajú v Bluetooth Low Energy a sú popísané v kapitolách 3.2.1 a 3.2.2.

2.2.3 Bluetooth identifikátor

Každé jedno Bluetooth zariadenie má unikátny 48 bitový identifikátor - adresu, zvyčajne skrátenú ako BD_ADDR. Prvých 24 bitov adresy je unikátny identifikátor organizácie, ktorý identifikuje výrobcu. Zvyšných 24 bitov je konkrétna adresa. Rozdelenie adresy je možné vidieť na obrázku 2.2. Zariadeniu môže byť priradený aj užívateľsky prívetivý názov, podľa ktorého je jednoduchšie zariadenie identifikovať.



Obrázek 2.2: Formát Bluetooth adresy

2.3 Vytvorenie spojenia, viazanie a párovanie

2.3.1 Proces spojenia

Proces spojenia dvoch Bluetooth zariadení (zariadenie A sa pripája na zariadenie B) môžeme rozdeliť do troch stavov:

1. **Dotazovanie (Inquiry)** - Ak obe zo zariadení nemajú o sebe žiadnu informáciu, zariadenie A odošle Inquiry požiadavku, teda snaží sa odhaliť zariadenie B. Ak zariadenie B takúto požiadavku zaregistruje, odošle správu zariadeniu A so svojou adresou, prípadne svojim menom a inými informáciami.

2. **Pripájanie (Pairing)** - Pripájanie je proces vytvárania spojenia, kedy zariadenie A aj zariadenie B už poznajú navzájom svoje adresy.

3. **Spojenie (Connection)** - Po dokončení procesu pripájania sú zariadenia v stave pripojené a nachádza sa v jednom zo štyroch módov:

    - **Aktívny mód** - V tomto móde sa nachádza zariadenie, keď aktívne odosiela alebo prijíma dáta.

- **Sniff mód** - Tento mód šetrí energiu, keď je zariadenie menej aktívne. Zariadenie spí a čaká na prenos dát v danom intervale (napríklad každých 100ms)
- **Hold mód** - Je dočasný šetrič energie, kedy zariadenie spí zadaný časový interval a po uplynutí sa vráti do aktívneho módu. Master zariadenie môže prikázať zariadeniu slave tento mód.
- **Park mód** - Park mód je mód hlbokého spánku. Master môže prikázať tento mód zariadeniu slave, ktoré sa stane neaktívne, až kým mu master neprikáže aby sa znova zobudilo.

2.3.2 Viazanie a párovanie

Ak sú dve Bluetooth zariadenia viazané a sú od seba vzdialené dostatočne blízko, automaticky vytvoria spojenie. Aby mohli byť zariadenie viazané, musia raz prejsť procesom párovania.

Keď sa dve zariadenia spárujú, zdieľajú medzi sebou svoje adresy, mená a profily, a zvyčajne ich uložia do pamäte. Pri párovaní si zariadenia uložia aj spoločný tajný kľúč, aby sa mohli kedykoľvek v budúcnosti automaticky spojiť.

Párovanie zvyčajne vyžaduje aj proces overenia, v ktorom musí užívateľ potvrdiť spojenie medzi zariadeniami. Postup overovania sa môže líšiť a závisí od užívateľského rozhrania zariadenia. Môže ísť od stlačenia tlačidla až po zadávanie PIN kódu na každom zo zariadení. PIN kód môže byť rôzny čo sa týka dĺžky a zložitosti, teda môže obsahovať numerické aj alfanumerické znaky a môže byť dlhý štyri až šestnásť znakov.

Kapitola 3

Bluetooth Low Energy

Bluetooth Low Energy je bezdrôtová technológia zameraná na použitie v nových druhoch aplikácií v zdravotníctve, fitnes alebo bezpečnosti. BLE poskytuje značne zníženú spotrebu energie a nákladov, kým zabezpečuje podobný komunikačný rozsah ako klasická technológia Bluetooth. V tejto kapitole budem popisovať technológiu Bluetooth Low Energy. Budem sa venovať rozdielom medzi Bluetooth a Bluetooth Low Energy, protokolovému zásobníku – z ktorého podrobnejšie popíšem dve vrstvy a podporou na rôznych operačných systémoch.

3.1 Hlavné rozdiely oproti technológii Bluetooth

Na rozdiel od 79 kanálov technológie Bluetooth, Bluetooth Low Energy má 40 kanálov indexovaných 0 až 39, pričom kanály 37, 38 a 39 sú takzvané advertising kanály, na ktorých sa vysielajú informácie o zariadení.

Bluetooth low energy je vyvíjaný s dôrazom na spotrebu energie, niektoré parametre sa od klasickej technológie Bluetooth líšia. V tabuľke 3.1 je základné porovnanie Bluetooth a Bluetooth Low Energy.

	Bluetooth Classic	**Bluetooth 4.0 Low Energy (BLE)**
IEEE štandard	802.15.1	802.15.1
Frekvencia (GHz)	2.4	2.4
Maximálna priepustnosť (Mbps)	1-3	1
Typická priepustnosť (Mbps)	0.7 - 2.1	0.27
Maximálny dosah (m) (vonkajšie prostredie)	1 - 100	50

Tabulka 3.1: Základné porovnanie Bluetooth a BLE[5]

3.2 Bluetooth Low Energy protocol stack

V kapitole 2.2.2 je popísaný protokolový zásobník technológie Bluetooth. V tejto kapitole budú konkrétnejšie rozobraté dva profily[1] Bluetooth host protokolového zásobníku, ktoré rozširujú klasickú Bluetooth špecifikáciu – Generic Access Profile a Generic Attribute Profile.

3.2.1 GAP - Generic Access Profile

Generic Access Profile je základným kameňom, ktorý umožňuje BLE zariadeniam medzi sebou na nižšej úrovni navzájom komunikovať. GAP môžeme definovať ako najvyššiu kontrolnú vrstvu, ktorá špecifikuje ako sa majú zariadenia správať pri odhaľovaní alebo spojení zariadení tak, aby umožnili výmenu dát medzi dvoma zariadeniami rozličných výrobcov.

Role zariadení

GAP definuje štyri rôzne role pre zariadenia:

- Boradcaster
- Observer
- Central
- Peripheral

Spomedzi týchto štyroch rolí, sú zaujímavé práve posledné dve

- Peripheral – sú definované ako slave zariadenia. Využívajú advertising pakety, aby ich mohli centrálne zariadenia odhaliť a prípadne s nimi nadviazať komunikáciu. Sú to väčšinou malé zariadenia s nízkou spotrebou energie a obmedzenými zdrojmi.
- Central – definované ako master zariadenia. Sú schopné naraz nadviazať viac spojení s peripheral zariadeniami. Sú iniciátorom spojenia. Pretože udržať viac spojení je náročnejšie na výpočtový výkon, zvyčajne to bývajú smartfóny, alebo tablety.

Advertising

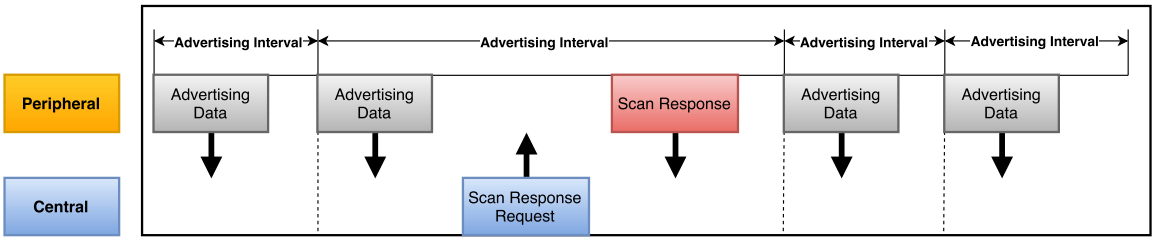
BLE má len jeden formát paketu a dva typy paketov – advertising a data pakety. Advertising pakety majú dva účely:

- Vysielať dáta pre aplikácie, ktoré nepotrebujú vzájomné pripojenie
- Odhaliť slave zariadenia a pripojiť sa na ne

Každý advertising paket môže obsahovať až 31 bajtov advertising dát plus dáta obsiahnuté v hlavičke, ktorá obsahuje aj adresu zariadnia z ktorého sa paket vysielal. Tieto pakety sa posielajú opakovane každý advertising interval, ktorý môže mať od 20 ms do 10.24 s. Čím kratší interval, tým väčšia frekvencia odosielania týchto advertising paketov, ktorá zvyšuje šancu, že tieto pakety budú zachytené cetral zariadením. Samozrejme platí, že čím väčšia frekvencia, tým je vyššia spotreba energie.

Central zariadenie môže dodatočne odoslať Scan Response Request, na ktorý mu peripheral zariadenie odpovie Scan Response Paketom.

Na obrázku 3.2 je možné vidieť priebeh odosielania Advertising a Scan Response dát.



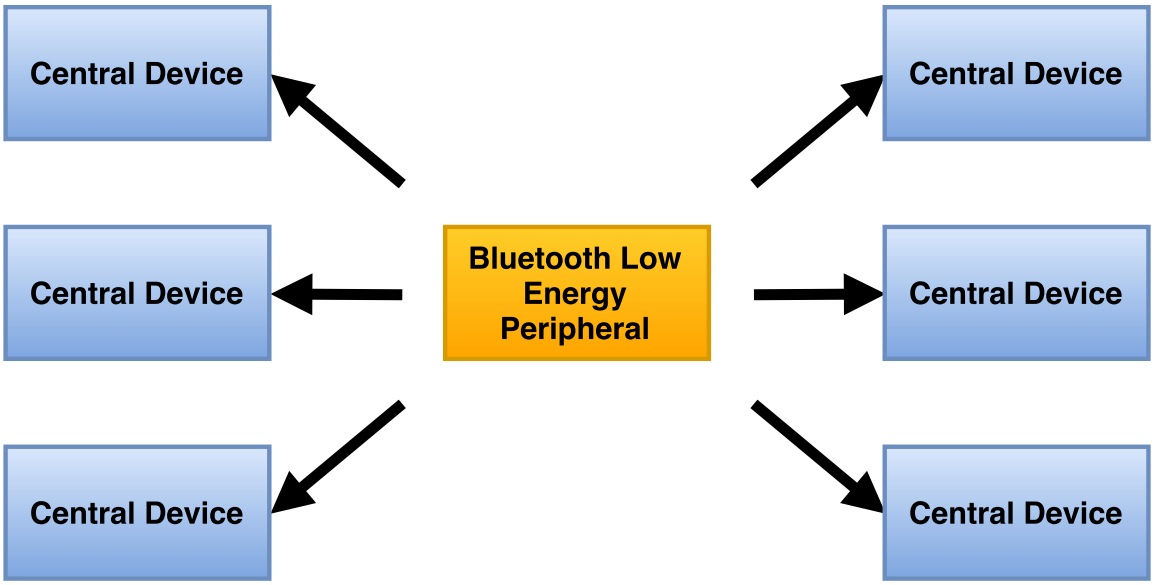
Obrázek 3.1: Priebeh vysielania advertising dát[6]

Broadcasting a Observing

Broadcaster je v tomto prípade peripheral, pretože vysiela požadované informácie a observer je aktívne central zariadenie, ktoré zachytáva tieto dáta.

Broadcast je v GAP definovaný ako všesmerové vysielanie, ktoré sa dá dosiahnuť výhradne len za pomoci Advertising dát. Observer dokáže pri skenovaní prijať tieto dáta, avšak ak by sa ktorýkoľvek observer chcel pripojiť na broadcast zariadenie, toto zariadenie zastaví odosielanie Advertising paketu a tým prestane broadcaster vysielať dáta. Táto situácia nastane z dôvodu výhradného spojenia slave zariadenia.

Veľkosť broadcastovaných dát sa dá zdvojnásobiť za pomoci Scan Response Data. Pretože ale Scan Response Data sú odoslané len pri requeste observera, kritické dáta by mali byť obsiahnuté v advertising pakete.



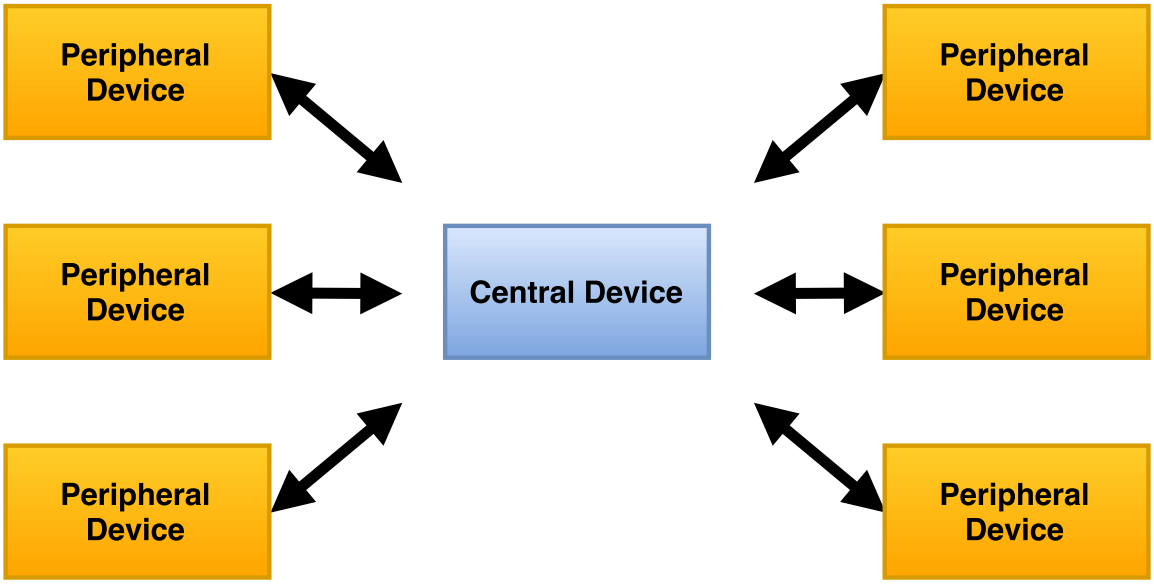
Obrázek 3.2: Broadcastová topológia Bluetooth Low Energy

3.2.2 GATT - Generic Attribute Profile

GATT definuje, ako budú dve Bluetooth Low Energy zariadenia odosielať a prijímať dáta za použitia služieb a charakteristík. Využíva pritom Attribute Protocol používaný na uloženie služieb, charakteristík a súvisiacich dát v jednoduchej vyhľadávacej tabuľke.

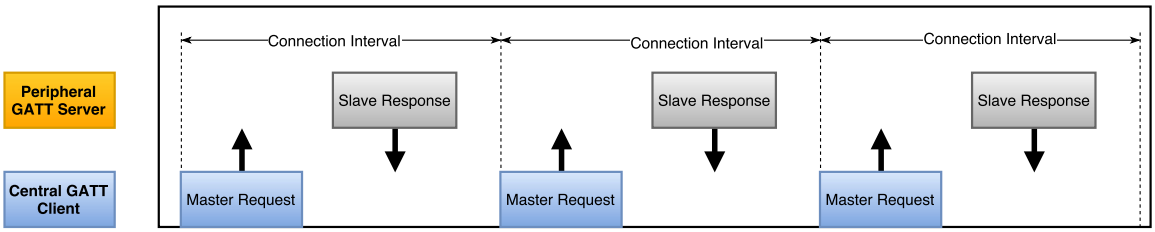
GATT definuje architektúru ako klient - server. Server - periférne zariadenie, je zariadenie ktoré obsahuje Attribue Protocol vyhľadávaciu tabuľku s dátami, službami a charak-

teristikami. Klient - centrálne zariadenie, je zariadenie ktoré serveru zasiela požiadavky na tieto dáta. Pripojenie klienta na server je definované ako výhradné, a preto server môže byť súčasne pripojený len k jednému klientovi. Pri pripojení server prestane vysielať na advertising kanáloch až kým nie je pripojenie prerušené. Na obrázku 3.3 je znázornená topológia siete pri nadviazanom spojení.



Obrázek 3.3: Topológia siete pri nadviazanom spojení[7]

Pri pokuse o nadviazanie spojenia, server navrhne časový interval spojenia klientovi. Klient sa snaží pripojiť každý časový interval aby zistil, či existujú nejaké nové dáta. Diagram na obrázku 3.4 zobrazuje proces výmeny dát medzi serverom a klientom.



Obrázek 3.4: Proces výmeny dát medzi serverom a klientom[7]

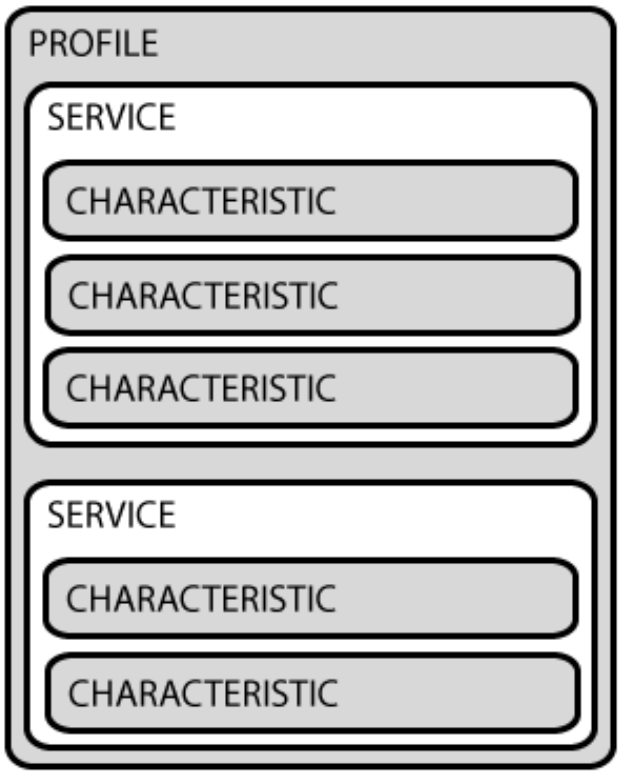
Profily, služby a charakteristiky

Profilom v technológii BLE je súbor preddefinovaných služieb vytvorených Bluetooth SIG[1] alebo vývojármi konkrétneho zariadenia. Profil je rozdelený službami na logické entity. Obsahujú špecifické dáta nazývané charakteristiky. Každá služba má unikátne číselné ID nazývané UUID, ktoré má dĺžku 16 bitov - pre oficiálne BLE služby, alebo 128 bitov pre vlastne vytvorené služby.

Najnižšou úrovňou GATT transakcií sú charakteristiky, ktoré zaobalujú dáta. Každá charakteristika obsahuje jeden údaj, pričom to môže byť aj pole dát, ktoré spolu súvi-

[1]Special interest group

sia (teplota a tlak z barometrického senzora). Podobne ako služby, aj charakteristiky sa identifikujú preddefinovaným 16 bitovým alebo 128 bitovým UUID



Obrázek 3.5: Zanorenie profilov, služieb a charakteristík[7]

3.3 Podpora v operačných systémoch

Aby operačný systém udržal krok s konkurenčnými operačnými systémami, vývojári musia reagovať na meniaci sa trh a prichádzajúce technologické novinky. Bluetooth Low Energy nie je výnimkou a preto túto technológiu podporuje každý moderný operačný systém. Medzi hlavné operačné systému s podporou BLE, ktorými sa táto práca zaoberá, patria:

- **Android**: od verzie 4.3 (API Level 18)
- **Windows Phone**: od verzie 8.1
- **Windows**: od verzie 8
- **iOS**: od verzie 5

3.4 Vývoj natívnych aplikácií

Pre každý z uvedených operačných systémov v kapitole 3.3 sa aplikácie vyvíjajú iným spôsobom. Hlavnou výhodou natívnych aplikácii je ich rýchlosť. Nevýhodou je ďalší vývoj pre

iný operačný systém. Či už ide o rozdielny programovací jazyk alebo rozdielne frameworky jednotlivých operačných systémov, vývoj BLE aplikácie môžeme zovšeobecniť do týchto piatich krokov:

- Získanie prístupu k Bluetooth hardware
- Vyhľadanie periférnych zariadení v dosahu
- Pripojenie sa na periférne zariadenie
- Čítanie a zapisovanie dát z a do zariadenia
- Ukončenie spojenia

Prístup ku konkrétnemu hardware rieši každý operačný systém rozdielne. Pre vyhľadávanie, pripojenie, prácu s dátami a ukončenie spojenia, existuje množina funkcií, ktorú implementuje každý z týchto operačných systémov. Hoci API[2] je pri rôznych operačných systémoch rozdielne, skúsim ho popísať nasledovnými univerzálnymi funkciami, ktoré implementuje aj plugin využitý pri vývoji aplikácie:

- **scan(services)** - funkcia, ktorá má za úlohu zapnúť skenovanie Bluetooth Low Energy zariadení v dosahu. Ako parameter sa jej predá zoznam služieb, ktoré ma periférne zariadenie obsahovať.
- **stopScan()** - funkcia má za úlohu zastaviť skenovanie
- **connect(UUID)** - funkcia, ktorá pripojí centrálne zariadenie ku periférnemu zariadeniu. Ako parameter sa jej predá UUID zariadenia, na ktoré sa chceme pripojiť.
- **disconnect(UUID)** - funkcia, ktorá odpojí centrálne zariadenie od periférneho zariadenia. Ako parameter sa jej predá UUID zariadenia, z ktorého sa chceme odpojiť.
- **read(UUID, service, characteristic)** - funkcia slúži na čítanie hodnoty charakteristiky. Ako parametre sa jej predajú UUID zariadenia, z ktorého chceme charakteristiku prečítať, službu ktorá obsahuje danú charakteristiku a konkrétnu charakteristiku, ktorej hodnotu chceme získať.
- **write(UUID, service, characteristic, value)** - funkcia je určená na zapísanie hodnoty do charakteristiky. Ako parametre sa jej predajú UUID zariadenia, ktorému chceme zapísať hodnotu charakteristiky, služba ktorá danú charakteristiku obsahuje, charakteristiku, do ktorej chceme zapísať hodnotu, a konkrétnu hodnotu.
- **writeWithoutResponse(UUID, service, characteristics, value)** - obdobne ako funkcia write, s tým rozdielom, že nám periférne zariadenie neodošle odpoveď, či sa zápis podaril alebo nie.
- **startNotification(UUID, service, characteristic)** - funkcia slúži na zaregistrovanie notifikácií, teda ak sa zmení hodnota charakteristiky, centrálne zariadenie bude informované o novej hodnote charakteristiky. Ako parametre sa jej predajú UUID zariadenia z ktorého chceme notifikácie dostávať, služba ktorá obsahuje požadovanú charakteristiku a konkrétna charakteristika.

[2] Application programming interface

- **stopNotification(UUID, service, characteristic)** - funkcia slúži na odregistrovanie notifikácií. Ako parametre sa jej predajú UUID zariadenia, z ktorého nechceme notifikácie dostávať, služba ktorá obsahuje požadovanú charakteristiku a konkrétna charakteristika, o ktorej dostávame notifikácie.

Kapitola 4

Vývoj multiplatformových aplikácií

S množstvom operačných systémov, ktoré sú na trhu, a so stúpajúcimi požiadavkami zákazníkov je čoraz náročnejšie a zložitejšie uspokojiť potreby čo najširšieho spektra užívateľov s rôznymi zariadeniami so systémami rozličných výrobcov. Vývoj moderných aplikácií sa uberá smerom, ktorý sa snaží eliminovať zložitosť a primárne znižovať cenu vývoja aplikácie.

Každé inteligentné zariadenie, či už mobilný telefón, tablet alebo osobný počítač disponuje webovým prehliadačom. Milióny užívateľov po celom svete denne využívajú webové služby a teda aj technológiu HTML. S novou špecifikáciou HTML5, ktorá vyšla v októbri 2014 potenciál webových aplikácií stúpol. Moderný webový prehliadač s podporou HTML5 teraz dokáže prehrávať video a audio, obsahuje canvas, na ktorý je možné renderovať 2D objekty a dokonca dokáže využívať potenciál grafickej karty pre svoju akceleráciu.

V tejto časti sa budem venovať webovým technológiam - jazyku javascript, štýlovaciemu jazyku CSS a značkovaciemu jazyku HTML5 ktoré sú v dnešnej dobe pre vývoj multiplatformových aplikácií na paltforme Apache Cordova kľúčové. Ďalej popíšem princípy a fungovanie platformy Apache Cordova, ktorú budem využívať pri vývoji multiplatformovej aplikácie.

4.1 HTML

HyperText Markup Language, je štandardný značkovací jazyk využívaný na tvorbu webových stránok. HTML sémanticky popisuje webovú stránku. Skladá sa z elementov, ktoré udávajú štruktúru dokumentu. Logicky rozdeľuje stránku na hlavičku a telo, následne na sekcie, nadpisy alebo odseky. HTML jazykom môžeme vkladať do dokumentu obrázky, zvuk alebo aj videá.

4.2 JavaScript

JavaScript je dynamický, netypovaný, prototypovo orientovaný programovací jazyk. Vo webových prehliadačoch slúži na dynamické zmeny elementov webovej stránky. JavaScriptové jadro interpretuje a spúšťa JavaScript. JavasScript môže existovať ako súčasť aplikácie ale aj ako samostatný interpret. Využíva sa aj v prostrediach ktoré nie sú webovo orientované, napríklad v PDF dokumentoch.

4.3 CSS

Cascading Style Sheet je štýlovací jazyk, ktorý upravuje zobrazenie dokumentu popísaného jazykom HTML. Najčastejšie sa využíva na vizualizáciu webových stránok a užívateľských rozhraní napísaných v HTML, ale môže byť použitý aj napríklad na XML dokument. Navrhnutý bol hlavne na oddelenie obsahu dokumentu od jeho vizuálnej reprezentácie. Obsahuje napríklad popis rozloženia elementov, farby alebo štýl písma.

4.4 Webový prehliadač

Webový prehliadač je aplikácia, ktorá v minulosti slúžila na zobrazovanie jednoduchého statického obsahu. Postupom času, s pribúdajúcim výpočtovým výkonom a s novými špecifikáciami, dokázal prehliadač prehrávať hudbu, videá alebo renderovať 2D grafiku. Moderný prehliadač sa skladá z niekoľkých hlavných častí:

- **User interface**: užívateľské rozhranie - obsahuje napríklad vstupné pole na zadávanie webovej adresy, tlačidlo späť a dopredu alebo záložky.
- **Browser engine**: jadro prehliadača - riadi akcie medzi užívateľským rozhraním a vykresovacím jadrom.
- **Rendering engine** - vykreslovacie jadro - je zodpovedné za spracovanie CSS a HTML a zobrazenie obsahu.
- **Networking** - sieťový modul - zodpovedá za sieťovú komunikáciu.
- **UI backend** - využíva sa na vykreslovanie základných ovládacích prvkov ako napríklad vstupné polia alebo tlačidlá.
- **Javascript interpreter** - prekladač javascriptu - využívaný na spúšťanie javascriptového kódu.
- **Data storage** - vrstva, ktorá zabezpečuje ukladanie interných dát, ako napríklad cookies. Moderné prehliadače taktiež podporujú úložné mechanizmy ako sú localStorage, IndexedDB alebo WebSQL.

Každý operačný systém, ktorým sa táto práca zaoberá, disponuje webovým prehliadačom a teda aj jadrom pre vykreslovanie webových stránok. Operačné systémy sú schopné poskytnúť toto jadro bez užívateľského rozhrania, teda bez polí pre zadávanie adresy alebo tlačidiel dopredu a späť. Toto zobrazenie sa nazýva WebView. Celý obsah displeja je len plocha na vykreslovanie komponentov webovej stránky pričom sú k dispozícii ostatné časti webového prehliadača. Týmto užívateľ nadobudne dojem natívnej aplikácie. Túto techniku využíva platforma Apache Cordova, ktorej sa budem venovať v nasledujúcich odstavcoch.

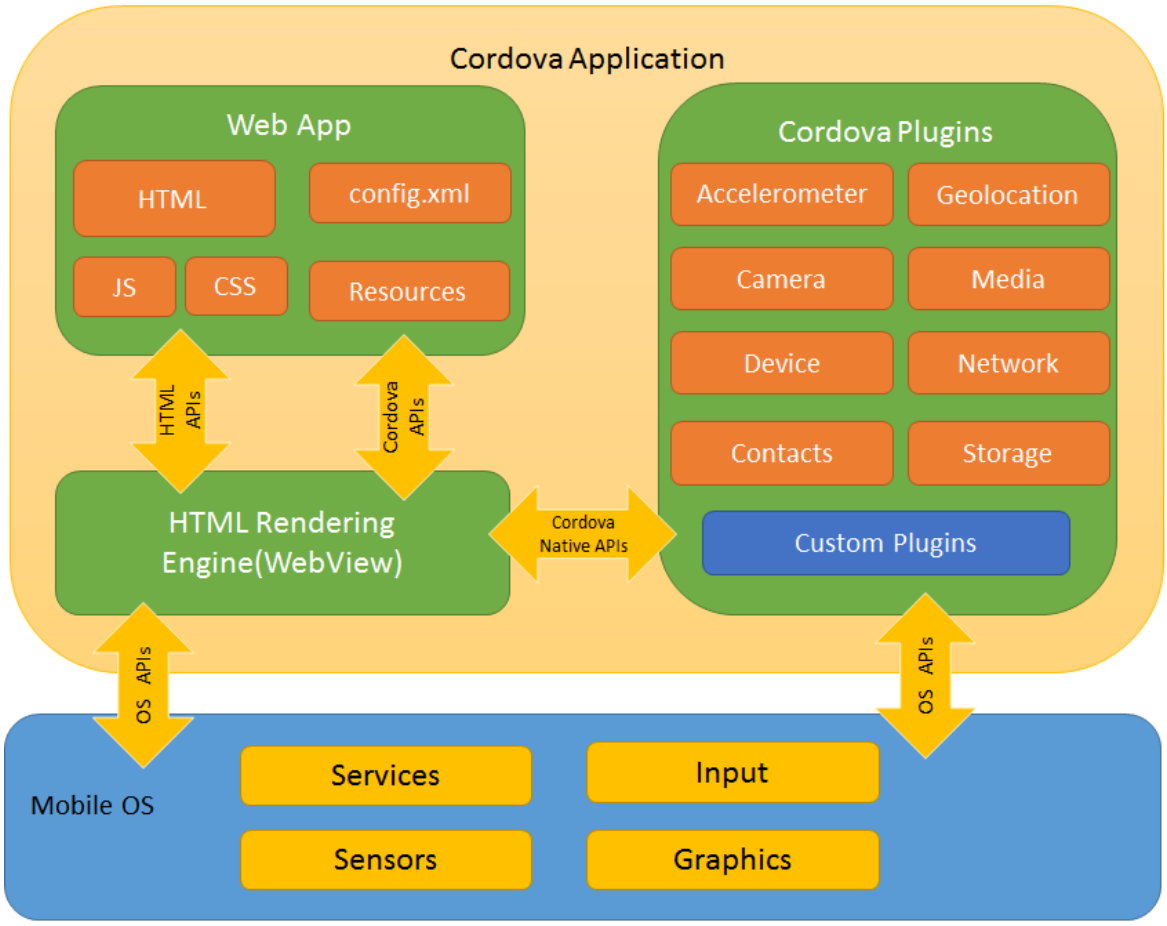
4.5 Apache Cordova

Apache Cordova[9] je open-source vývojový framework zameraný na mobilné zariadenia, ktorý využíva štandardné webové technológie - HTML5, CSS3 a JavaScript na vývoj multiplatformových aplikácii. Apache Cordova tvorí medzivrstvu medzi webovými technológiami a natívnymi funkciami, ktoré dokáže webovým technológiam sprostredkovať. V aktuálnej verzii Cordova podporuje nasledovné platformy:

- Android
- Blackberry od verzie 10
- iOS
- Ubuntu
- Windows Phone od verzie 8
- Windows od verzie 8.1

Vývoj pôvodného frameworku začal v roku 2009. V roku 2011 spoločnosť Adobe Systems kúpila spoločnosť Nitobi, pôvodného tvorcu frameworku, ktorý premenovala na PhoneGap a neskôr vydala jeho open-source verziu pod názvom Apache Cordova.

Existuje niekoľko komponent Cordova aplikácie. Diagram 4.1 zobrazuje architektúru Cordova aplikácie.



Obrázek 4.1: Architektúra Cordova aplikácie[9]

- **WebView** je špeciálne okno prehliadača, v ktorom sa vykresluje Cordova aplikácia. Základne podporuje klasickú funkcionalitu webového prehliadača. Framework využíva špeciálne pluginy, ktoré dokážu WebView rozšíriť o funkcionalitu, ktorú poskytujú natívne aplikácie.

- **Webová aplikácia** – Pri vytváraní aplikácie vývojár vytvára klasickú webovú stránku. Vstupným bodom aplikácie je súbor *index.html*, v ktorom sa definujú odkazy na ďalšie zdrojové súbory - CSS kód, JavaScrip kód, obrázky a ostatné súbory potrebné na chod aplikácie. Aplikácia sa spustí vo WebView obalenom v natívnej aplikácii. Táto aplikácia sa potom distribuuje na jednotlivé zariadenia.

  Pri vytvorení Apache Cordova projektu sa vytvorí aj súbor *config.xml*, ktorý obsahuje informácie a nastavenia aplikácie. Obsahuje zoznam pluginov, ale napríklad aj nastavenia zmeny orientácie displeja zariadenia.

- **Cordova pluginy** – Pluginy sú neoddeliteľnou súčasťou Cordovy. Poskytujú rozhranie, cez ktoré komunikuje Cordova a natívne časti systému. Toto umožňuje spúšťať natívny kód systému z JavaScriptu. Apache Cordova poskytuje základné pluginy, ktoré umožňujú pristupovať k zdrojom zariadenia, napríklad batéria, kontakty, kamera alebo Bluetooth.

Kapitola 5

Návrh a implementácia aplikácie

Vzhľadom na typ aplikácie a fakt že Apache Cordova predstavuje vrstvu medzi aplikáciou a natívnym volaním funkcií, vzniká podozrenie, že pri nesprávnom návrhu vznikne spomalenie, ktoré bude mať vplyv na plynulú prácu s aplikáciou. Preto sa budem snažiť optimalizovať kód, ktorý bude mať menšie nároky na hardware.

5.1 Použité technológie

Technológie použité pri vývoji projektu:

- **Apache Cordova**
- **Ionic**
- **AngularJS**

Aplikáciu som sa rozhodol vyvíjať za pomoci frameworku Ionic, ktorý je nadstavbou platformy Apache Cordova a JavaScriptového MVC[1] frameworku AngularJS. AngularJS umožňuje rozdeliť aplikáciu na tri nezávislé časti podľa architektúry Model View Controller a poskytuje automatickú synchronizáciu dát časťami model a view. Ionic ponúka taktiež CSS framework s preddefinovanými štýlmi optimalizovanými pre mobilné aplikácie. Výhodou Ionic frameworku je hardwarová akcelerácia za pomoci CSS prechodov a animácií, čo dáva procesoru viac času na dôležitejšie úlohy hlavne obsluhu hardware a keďže sa jedná o mobilné zariadenie, ktoré nie je pripojené do siete elektrickej energie a má obmedzenú kapacitu batérie, Ionic prispieva aj k jej výdrži.

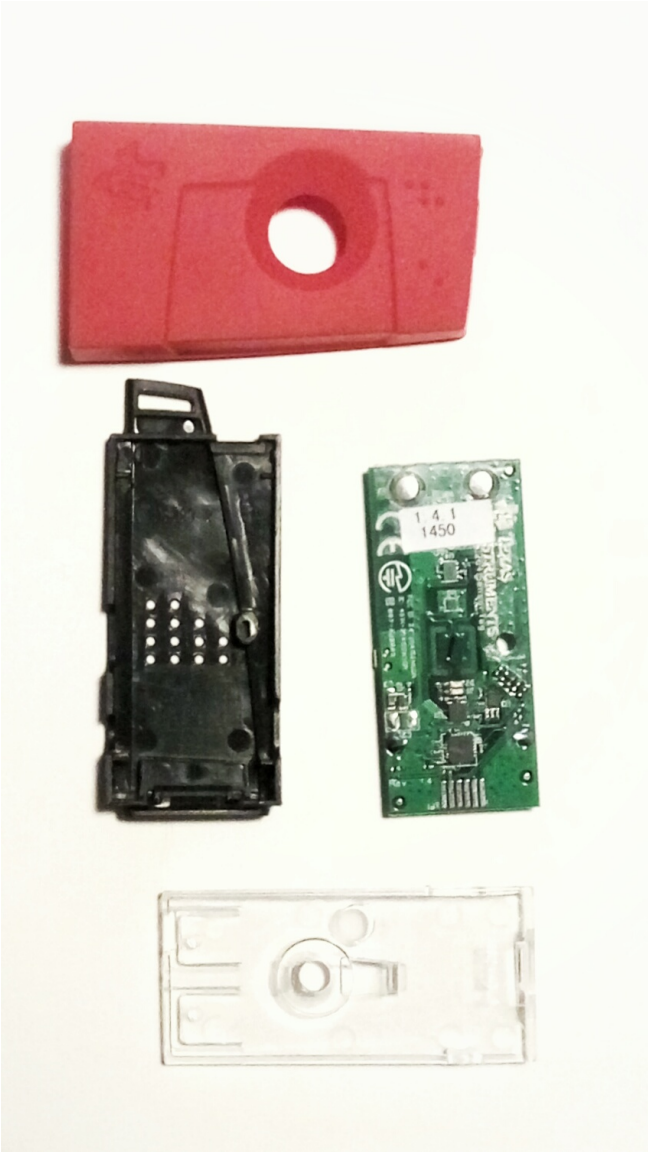
5.2 Použitý hardware

Vyvíjať a testovať aplikáciu budem na dvojici senzorov od spoločnosti Texas Instruments s označením TI CC2541 SensorTag. Ako centrálne zariadenie pre vývoj bude použitý mobilný telefón od spoločnosti LG s označením G2 na ktorom je verzia Android Marshmallow 6.0.1 (API level 23). SensorTag je Bluetooth Low Energy vývojový kit, ktorý funguje ako peripheral zariadenie. SensorTag je zobrazený na obrázku 5.1. Zameraný je na vývoj aplikácií pre mobilné zariadenia a poskytuje hneď niekoľko senzorov:

- Infračervený teplotný senzor

[1] Model-view-controller

- Senzor vlhkosti
- Tlakový senzor
- Akcelerometer
- Gyroskop
- Magnetometer



Obrázek 5.1: TI CC2541 SensorTag

5.3 Špecifikácie požiadaviek

Na základe osobných skúseností s vývojom aplikácií a s ohľadom na funkčnosť aplikácie som sa rozhodol definovať špecifikácie požiadavkov nasledovne: Aplikácia musí vedieť vyhľadať

dané senzory, ktoré sú v stave advertising. Centrálne zariadenie sa bude môcť pripojiť ku viacerým periférnym zariadeniam súčasne. Dáta, ktoré získa centrálne zariadenie zo senzoru, musia byť perzistentné. Po pripojení na periférne zariadenie, sa začne automaticky meranie zo senzoru.

História dát z jedného senzoru sa bude dať filtrovať podľa dátumu a času. Pri výpise histórie dát zo senzoru sa bude podľa zadaného časového filtra vykreslovať čiarový graf, ktorý sa bude aktualizovať novými dátami, pokiaľ je periférne zariadenie pripojené k centrálnemu zariadeniu a bude vyhovovať filtru.

Senzor pripojený na centrálne zariadenie sa bude dať odpojiť, čím sa preruší prenos dát.

5.4 Funkčné požiadavky

V rámci funkčnosti, musí aplikácia vykonávať nasledujúce úkony:

1. začať a ukončiť skenovanie periférnych zariadení
2. pripojiť a odpojiť sa z periférneho zariadenia
3. zobraziť a vyfiltrovať dáta z histórie meraní

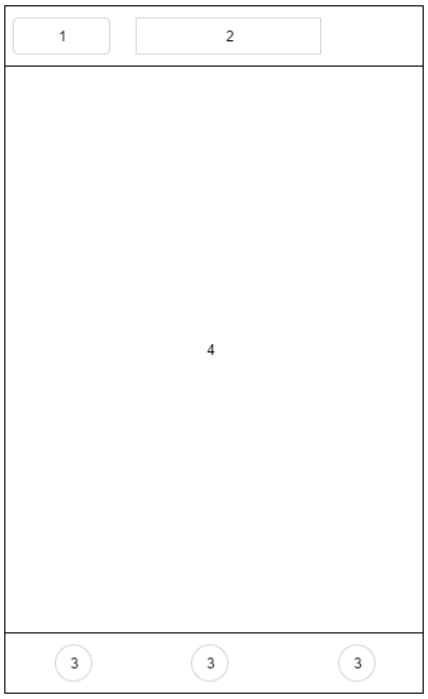
5.5 Uloženie dát

Na základe špecifikácie z kapitoly 5.3, musí byť aplikácia schopná perzistentne uložiť dáta prijaté z periférneho zariadenia. Webový prehliadač a teda aj WebView poskytuje rôzne možnosti ukladania dát. Po zvážení plusov a mínusov, vzhľadom na veľkosť a rozsiahlosť aplikácie, som zhodnotil, že najlepšou voľbou bude využiť mechanizmus localStorage, v ktorom sa ukladajú dáta do dátovej štruktúry asociatívne pole. Ako kľúč bude použitý unikátny identifikátor konkrétneho periférneho zariadenia a hodnota bude uložená ako JavaScriptový objekt vo formáte JSON. Pri veľkosti dát, ktoré sa budú do localStorage ukladať, by nemalo byť citeľné spomalenie aplikácie.

5.6 Užívateľské rozhranie

Užívateľské rozhranie som prispôsobil na menšie zariadenia, avšak pre lepšiu čitateľnosť by veľkosť uhlopriečky mala dosahovať aspoň štyri palce. Zameral som sa na prehľadnosť a jednoduchosť používania. Snažil som sa dosiahnuť, aby užívateľ musel pri čo najmenšom počte interakcií so zariadením dostať čo najviac informácií. Návrh UI je znázornený na obrázku 5.2, kde jednotlivé čísla zastupujú:

1. Tlačidlo späť, ktoré je v niektorých častiach aplikácie skryté.
2. Názov sekcie v ktorej sa užívateľ nachádza.
3. Menu, ktoré obsahuje tlačidlá na navigáciu v aplikácii.
4. Plocha do ktorej sa vykresľujú informácie o a z periférnych zariadení.



Obrázek 5.2: Návrh užívateľského rozhrania – rozloženie komponentov

5.7 Logika aplikácie

Aplikácia má za úlohu vyhľadávať zariadenia, pripájať a odpájať sa zo zariadení a ukladať a zobrazovať históriu dát. Z toho dôvodu som sa rozhodol aplikáciu rozdeliť na nasledovné views:

- **devices-scanned** view
- **devices-connected** view
- **devices-history** view

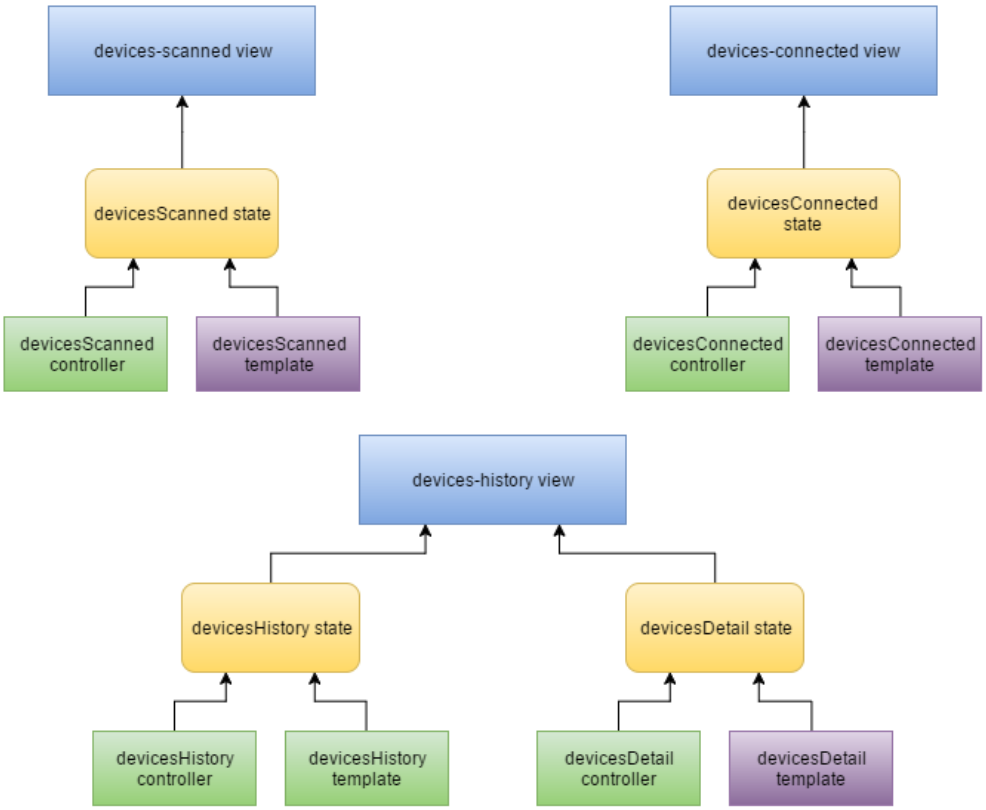
AngularJS využíva routovací systém, takže aplikácia môže byť podľa URL adresy rozdelená na rôzne stavy v ktorých sa aplikácia môže nachádzať. Tieto stavy sú nasledovné:

- **devicesScanned** je stav, v ktorom sa aplikácia nachádza ihneď po spustení. V tomto stave aplikácia zobrazuje zariadenia vysielajúce adveritsement paket, ktoré sú v dosahu centrálneho zariadenia.
- **devicesConnected** stav, v ktorom sa na displej zariadenia zobrazia peripheral zariadenia, ku ktorým je centrálne zariadenie aktuálne pripojené.
- **devicesHistory** je stav, v ktorom sa zobrazujú všetky pripojené zariadenia od prvého spustenia aplikácie.
- **devicesDetail** stav aplikácie, v ktorom užívateľ vidí konkrétne dáta z vybraného senzoru.

Pre jednotlivé stavy sa definuje view – v ktorom sa má vykresliť, šablóna – v ktorej je uložený design a controller – ktorý má daný konkrétny stav spravovať. Šablóna je HTML kód, ktorý sa vykreslí do view podľa stavu. Controller je kód napísaný v JavaScripte, ktorého úlohami je prepojiť view s dátami aplikácie a spracovať vstup od užívateľa. Zoznam controllerov a funkcionalita ktorú vykonávajú:

- **devicesScanned** controller – jeho náplňou je spúšťať a vypínať proces skenovania a plniť model dátami z advertising paketov. Po zvolení konkrétneho zariadenia prechádza aplikácia do stavu devices-history.
- **devicesConnected** controller – zabezpečuje odpájanie pripojených zariadení
- **devicesHistory** controller – plní model dátami a zabezpečuje prechod aplikácie do stavu devicesDetail, po zvolení zariadenia ktorého históriu chcem zobraziť.
- **devicesDetail** controller – sa stará o dáta pri zobrazovaní detailu zvoleného periférneho zariadenia.

Pre každý stav aplikácie existuje práve jeden controller a práve jedna šablóna. Vizualizáciu logického návrhu je možné vidieť na obrázku 5.3.



Obrázek 5.3: Návrh logiky aplikácie

5.8 Knižnice tretích strán

Ako bolo spomenuté v kapitole 4.5, framework Apache Cordova umožňuje pridať do aplikácie pluginy ktoré rozširujú základnú funkcionalitu Cordovy. Bluetooth Low Energy tech-

nológia nie je podporovaná základnými pluginmi a preto som musel plugin *Bluetooth Low Energy (BLE) Central plugin for Apache Cordova*[8] integrovať od vývojára tretej strany. Toto rozšírenie poskytuje rozhranie pre volanie natívnych funkcií systému ktoré ovládajú Bluetooth hardware.

Bluetooth Low Energy (BLE) Central plugin

Plugin je prevzatý od vývojára tretej strany. Verzovaný je pod licenciou Apache 2.0. Plugin poskytuje rovnakú funkcionalitu aká je dostupná v prípade volania natívnych funkcií. Dáta, ktoré plugin poskytuje pre advertisement a pakety odosielané po pripojení dvoch zariadení sú uložené v JavaScriptovom objekte a majú nasledovný formát:

```
{
    "name": "TI SensorTag",
    "id": "00:1A:7D:DA:71:13",
    "advertising": ArrayBuffer,
    "rssi": -37
}
```

Listing 5.1: Formát advertisement dát

Po úspešnom spojení dvoch zariadení, plugin ku dátam z advertisement paketu pridá dáta o službách a charakteristikách:

```
{
    "name": "TI SensorTag",
    "id": "20:FF:D0:FF:D1:C0",
    "advertising": [2,1,6,3,3,15,24,8,9,66,97,116,116,101,114,121],
    "rssi": -55,
    "services": [
        "1800",
        "1801",
        "180f"
    ],
    "characteristics": [
        {
            "service": "1800",
            "characteristic": "2a00",
            "properties": [
                "Read"
            ]
        },
        {
            "service": "1800",
            "characteristic": "2a01",
            "properties": [
                "Read"
            ]
        }
}
```

Listing 5.2: Formát dát po pripojení

Charts

Táto knižnica zabezpečuje vykresľovanie dát do rôznych typov grafov. Pre správnu funkčnosť aplikácie, je potrebné pridanie knižnice Angular Chart[10], ktorá zapúzdri funkcionalitu Charts.js[11] do frameworku AngularJS.

Kapitola 6

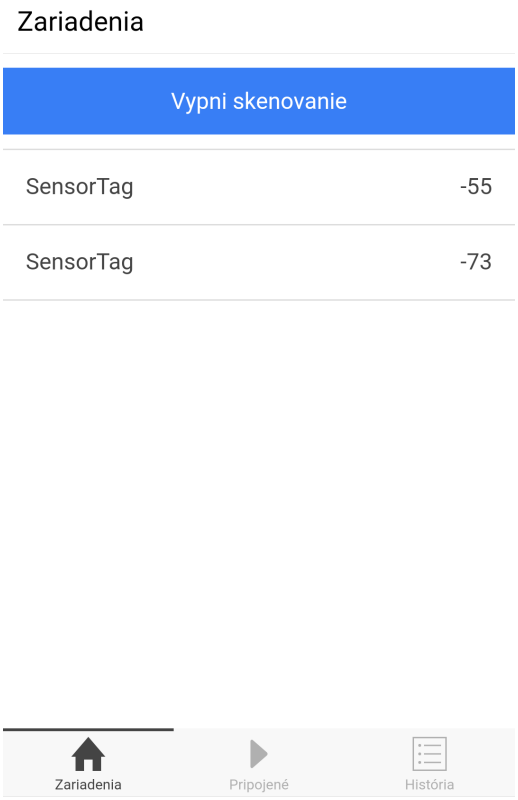
Implementácia

Ako vývojové prostredie pre implementáciu aplikácie som zvolil Visual Studio 2015 od spoločnosti Microsoft. Visual Studio obsahuje nástroje, ktoré uľahčujú integráciu frameworku Ionic s ostatnými technológiami použitými v projekte.

Vyhľadávanie zariadení – devicesScanned

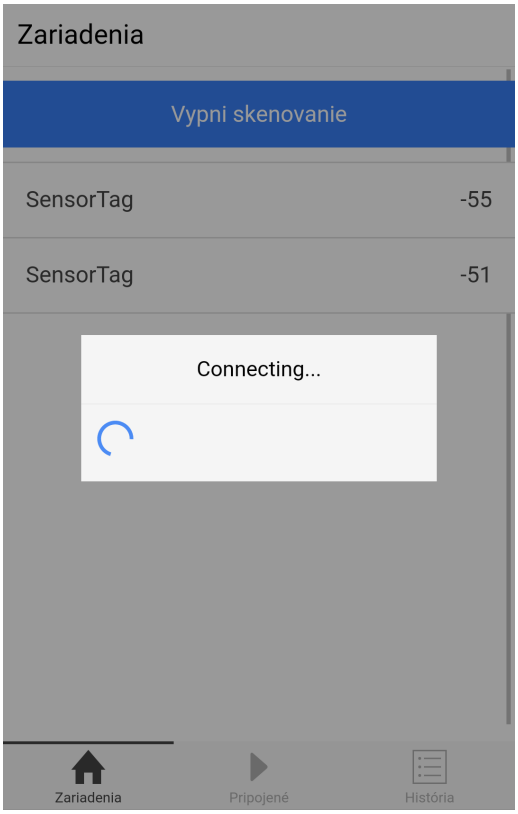
Vyhľadávanie zariadení má na starosti *devicesScanned* controller. Jeho úlohou je spustiť skenovanie, pričom musí zistiť stav Bluetooth hardware, teda či je Bluetooth spustený. K tomuto úkonu sa využíva dvojica funkcií z Apache Cordova API Bluetooth Low Energy pluginu – *enable()* a *startScan()*. Po kliknutí na tlačidlo Začať skenovanie sa zavolá v controlleri funkcia *scan()* ktorá skontroluje stav Bluetooth hardware a pokiaľ je Bluetooth zapnutý, spustí proces skenovania.

Proces skenovania sa automaticky spúšťa každý časový interval T, ktorý je v mojej aplikácii nastavený na hodnotu 750 ms. Ak Bluetooth počas skenovania nájde peripheral zariadenie v stave advertising, controller takéto zariadenie uloží do dátovej štruktúry typu object, kde kľúčom bude adresa objaveného zariadenia a hodnotou bude objekt ktorého atribúty budú dáta z advertisement paketu. AngularJS sa potom postará o vykreslenie týchto dát do šablóny, ktorá je spolu s controllerom definovaná pre konkrétny stav. Po stlačení tlačidla Vypnúť skenovanie sa skenovanie vypne a zmažú sa dáta ktoré obsahuje objekt s oskenovanými zariadeniami. Na obrázku 6.1 je aplikácia zobrazená v stave **devicesScanned**.



Obrázek 6.1: GUI v stave devicesScanned

Po kliknutí na konkrétne zariadenie v šablóne sa aplikácia pokúsi pripojiť na zariadenie – obrázok 6.2. Po úspešnom pripojení sa pokúsi centrálne zariadenie o zápis hodnoty do charakteristiky. Ak sa zápis dát podarí, snímač zapne teplotný senzor. Následne sa peripheral zariadeniu odošle požiadavka o zasielaní notifikácií, teda pri zmene dát zo senzora o tom bude naše zariadenie informované. Po úspešnom nastavení notifikácií pridá aplikácia informácie o pripojenom zariadení do objektu s aktuálne pripojenými zariadeniami a získa dáta z úložného priestoru WebView – localStorage. Dáta z localStorage sú vo formáte string, respektíve JSON, a je potrebné previesť do JavaScriptového objektu za pomoci funkcie *JSON.parse()* a uložiť do vnútorného objektu aplikácie uchovávajúceho históriu dát z jednotlivých zariadení. Po úspešnom pripojení prejde aplikácia do nasledovného stavu – **devicesConnected**.

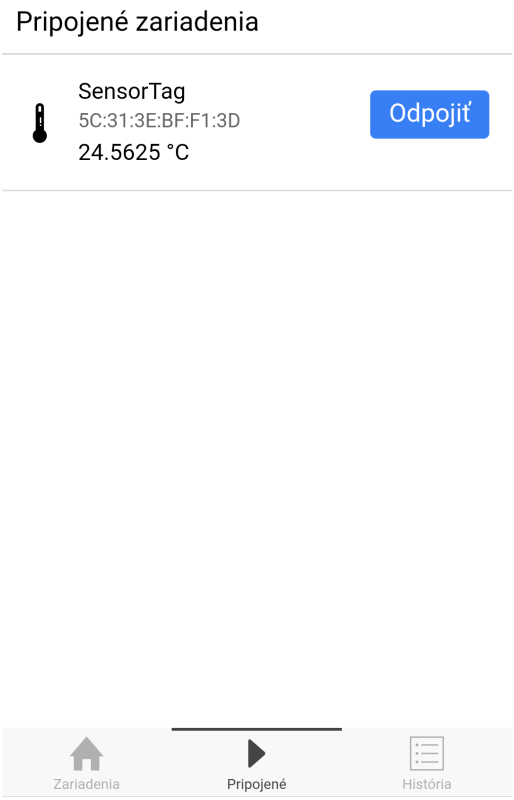


Obrázek 6.2: GUI v stave devicesScanned pri pokuse o pripojenie k peripheral zariadeniu

Pripojené zariadenia – devicesConnected

Spravovanie pripojených zariadení obstaráva *devicesConnected* controller. Zabezpečuje odpájanie pripojených zariadení a zobrazenie detailu zariadenia. Pri odpájaní zo zariadenia je potrebné získané dáta zo senzora opäť previesť do formátu JSON a uložiť do úložného priestoru localStorage. Prevod dát do formátu JSON zabezpečuje funkcia *JSON.stringify()*.

Úloha, ktorú zastáva tento controller je aj poskytovanie dát pre šablónu. V stave **devicesConnected** sa zobrazujú údaje o zariadení a to adresa zariadenia, názov zariadenia a teplota z teplotného čidla – zobrazené na obrázku 6.3. Tieto dáta controller získava z objektu s aktuálne pripojenými zariadeniami, do ktorého sa ukladá aj informácia z notifikácií o naposledy nameranej teplote.



Obrázek 6.3: Zobrayenie stavu devicesConnected

Pri kliknutí na konkrétne zariadenie sa aplikácia dostane do stavu **devicesDetail**.

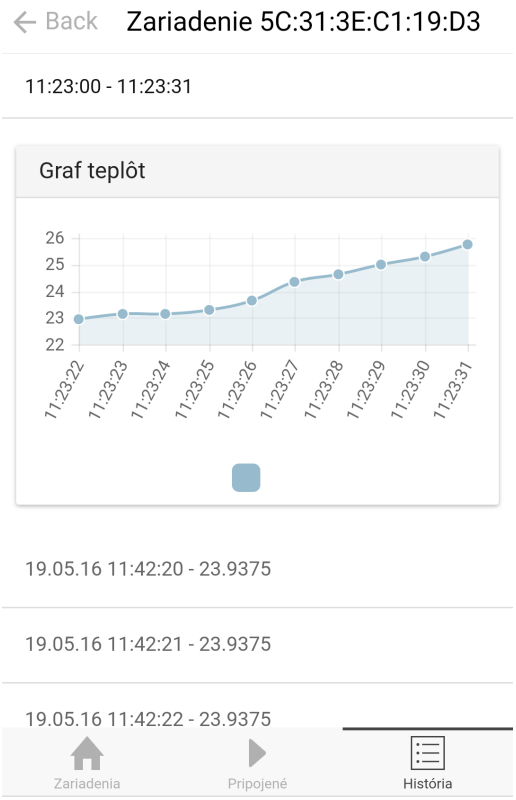
Detail zariadenia – devicesDetail

V stave **devicesDetail** sa aplikácia nachádza, pokiaľ chce užívateľ zobraziť detaily o aktuálne pripojenom zariadení alebo o zariadení na ktoré bol už niekedy v minulosti pripojený. O dáta tohto zobrazenia sa stará devicesDetail controller, ktorého úlohou je dáta zobraziť a pripraviť – pri nepripojených zariadeniach z localStorage a pri aktuálne pripojených zariadeniach z objektu aplikácie uchovávajúceho históriu nameraných dát.

Medzi ďalšiu funkcionalitu patrí príprava dát pre graf. Dáta sa získavajú z objektu obsahujúceho históriu dát pričom sa filtrujú podľa času kedy boli namerané. Čas sa zadáva do vstupného poľa odkiaľ sa získa, prekonvertuje sa na počet milisekúnd od 1. januára 1970[1]. S prekonvertovaným časom sa následne vyhľadáva v localStorage, prípadne v objekte uchovávajúcom históriu dát. O vykresľovanie grafu sa stará samotný AngularJS framework.

Na obrázku 6.4 je zobrazený stav **deviceDetail**. Obsahuje vstupné pole, do ktorého sa zadáva dátum, podľa ktorého sa má filtrovať, graf s nameranými hodnotami a ďalšie namerané hodnoty aj s presným časom.

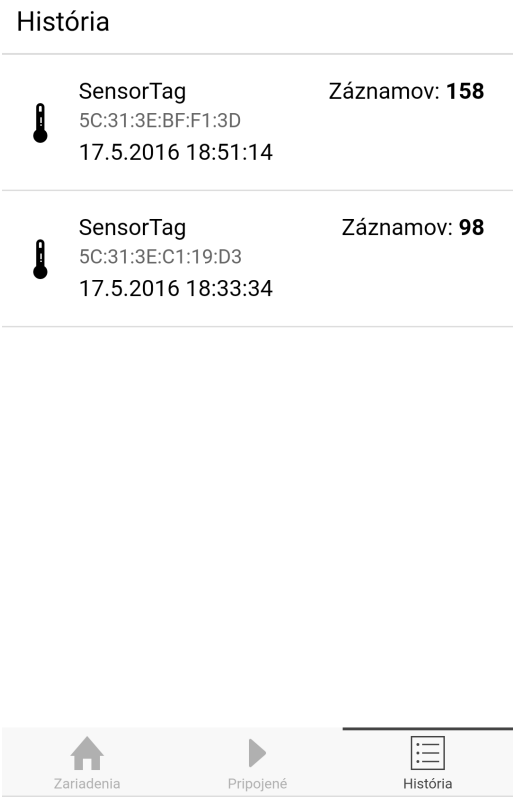
[1] Unix time format



Obrázek 6.4: Zobrazenie stavu deviceDetail

História nameraných hodnôt – devicesHistory

V stave **devicesHistory** aplikácia zobrazuje zoznam všetkých zariadení na ktoré sa užívateľ pripojil od začiatku jej používania. Dáta pre tento stav spravuje *devicesHistory* controller. Dáta získava z mechanizmu localStorage a v prípade, že je aplikácia pripojená k nejakému zariadeniu, dáta získa aj z objektu aplikácie uchovávajúceho históriu dát. V stave **devicesHistory** aplikácia zobrazuje informácie o názve zariadenia, adresu zariadenia, dátum posledného pripojenia k zariadeniu a počet údajov uložených v localStorage. Zobrazenie je možne vidieť na obrázku 6.5.



Obrázek 6.5: Zobrazenie stavu deviceHistory

6.1 Problémy pri vývoji

Ako bolo spomenuté v kapitole 4.5, Apache Cordova tvorí medzivrstvu medzi webovými technológiami a natívnymi funkciami operačného systému. Zároveň je to framework v ktorom sa dajú vyvíjať aplikácie pre viac platforiem. S rýchlym vývojom, či už malými alebo veľkými zmenami, v rámci operačných systémov sa musí Apache Cordova dynamicky prispôsobovať hneď niekoľkým systémom naraz. Prispôsobovať sa nemusí len samotný framework, ale aj pluginy, či už základné – obsiahnute priamo v Apache Cordova, alebo aj pluginy od vývojárov tretích strán.

Keďže som vyvýjal a testoval aplikáciu na zariadení s operačným systémom Android Marshmallow novšej verzie 6.0.1, plugin ktorý som využíval na komunikáciu s Bluetooth hardware nebol s touto verziou operačného systému kompatibilný. Autor tohto pluginu však vydal novú verziu, ktorá upravovala prístupové práva k hardware, čím sa problém vyriešil.

Na ďalší problém pri vývoji som narazil pri odpojení už pripojeného zariadenia. Peripheral zariadenie, ktoré sa dostalo zo stavu vysielania advertising paketov do stavu pripojené sa po odpojení nedostalo opäť do stavu vysielania advertising paketov. Po preštudovaní diskusného fóra[2] som našiel riešenie, ktoré zahrňovalo úpravu zdrojového kódu tohto pluginu zavolaním funkcie gatt.disconnect() pred uzatvorením spojenia.

Ďalší problém, na ktorý som narazil, a nebol som úspešný pri jeho vyriešení, je proces pripájania a čítania dát respektíve zápisu dát do senzora. Po procese skenovania a následného pokusu o pripojenie na dané zariadenie, centrálne zariadenie odmietne uskutočniť

[2]https://github.com/don/cordova-plugin-ble-central/issues/221

pripojenie z dôvodu, že peripheral zariadenie s danou adresou nebolo nájdené. Ďalej pri pokuse o zápis dát do senzoru (pokus o zapnutie tepelného čidla), zápis dát zlyhá s rovnakou chybou ako pri pripájaní na dané zariadenie. Toto chovanie aplikácie nie je pravidlom a po niekoľkých pokusoch sa úspešne podarí nadviazať spojenie aj čítať dáta.

Posledný problém, ktorý som nedokázal vyriešiť, je pripojiť sa na peripheral zariadenie po úspešnom čítaní dát z iného peripheral zariadenia. Po získavaní údajov z jedného zariadenia, následné skenovanie neodhalí žiadne peripheral zariadenia, ktoré sú v stave advertising a sú v dosahu centrálneho zariadenia.

Za zdroj týchto chýb považujem framework Apache Cordova, ktorý rozširuje funkcionalitu WebView o natívne funkcie operačného systému. Táto funkcionalita nie je pre technológiu Bluetooth Low Energy vyvíjaná v rámci frameworku Apache Cordova, ale vývojárom tretej strany. Preto považujem toto rozšírenie za nedostatočne kompatibilné s najnovšou verziou systému Android. Chyby tohto charakteru nemajú vplyv na funkčnosť, ale znižujú plynulosť a intuitívnosť výslednej aplikácie.

Kapitola 7

Záver

Cieľom tejto práce bolo zoznámiť sa s technológiou Bluetooth Low energy a implementovať aplikáciu demonštrujúcu komunikáciu so snímačmi neelektrických veličín.

Samotnej implementácii predchádzalo štúdium technológie Bluetooth, jej profilov, spôsobov nadväzovania komunikácie. Hlavné vlastnosti technológie som popísal v úvodných kapitolách.

Pre experimentovanie s technológiou Bluetooth Low Energy som si zvolil hotové snímače teploty, osvetlenia a vlhkosti. Tieto snímače dodáva spoločnosť Texas Instruments. Jedná sa o vývojové prostriedky, ktoré obsahujú napájacie obvody, samotné snímače a Bluetooth s anténou na jednej doske plošných spojov. Ako primárnu platformu pre vývoj som si zvolil operačný systém Android 6.0.1, ktorý bol nainštalovaný na mobilnom telefóne LG G2. Pre vývoj aplikácie som sa rozhodol využiť multiplatformový prístup. Naštudoval som možnosti technológie Cordova, ktorá umožňuje vývoj multiplatformových aplikácii pre tri dnes najvýznamnejšie mobilné operačné systémy. Pre vývoj multiplatformových aplikácií využíva Apache Cordova webové technológie – JavaScript, CSS3 a HTML5. Experimentami s frameworkom Cordova som zistil, že podpora Bluetooth Low Energy nie je zahrnutá v základných rozšíreniach. Podpora Bluetooth Low Energy vo frameworku Cordova je vyvíjaná open source komunitou.

Pre overenie funkčnosti bolo použité komunitné riešenie Bluetooth Low Energy. Výhodou tohto riešenia je otvorenosť. Na druhú stranu pri vývoji aplikácie a experimentoch s BLE a frameworkom Cordova som narazil na niekoľko problémov, ktoré určitý čas úplne bránili komunikácii BLE na použitej hardvérovej a softvérovej platforme. Počas vývoja aplikácie bola vydaná aktualizácia, ktorá vyriešila základné chyby v komunikácii.

Aplikáciu som vyvíjal primárne pre operačný systém Android a ako primárny snímač som si zvolil BLE teplomer od Texas Instruments. Výsledná aplikácia demoštruje komunikáciu s BLE senzormi. Počas vývoja som sa snažil zamerať aj na stabilitu komunikácie, ktorú znižovalo už spomenuté komunitné riešenie BLE podpory vo frameworku Cordova. Aplikácia vo výsledku dokáže skenovať dostupné zariadenia, pripojiť sa k zariadeniu a vyčítať dáta. V aplikácii som implementoval aj spracovanie dát, hlavne ich grafické zobrazovanie.

Práca sa v súčasnosti venuje získaniu dát zo snímačov BLE. Ďalšími krokmi vývoja by mohla byť analýza týchto dát z veľkého množstva snímačov. Integráciou snímačov do reálnej aplikácie by boli získané relevantné dáta. Analýzou týchto dát by bolo možné napríklad optimalizovať výdaje domácnosti na energie pri použití snímačov osvetlenia, teploty, vlhkosti.

Literatura

[1] Towsend, K.; Cufí, C.; Akiba; aj.: *Getting Started with Bluetooth Low Energy*. O'Reilly Media, April 2014, ISBN 978-1-4919-0057-4.

[2] WWW stránky: *Bluetooth specifications and features*. [cit. 2016-1-10], [Online; navštívené 10.1.2016].
URL
https://en.wikipedia.org/wiki/Bluetooth#Specifications_and_features

[3] WWW stránky: *History of the Bluetooth Special Interest Group*. [cit. 2016-1-10], [Online; navštívené 10.1.2016].
URL https://www.bluetooth.com/media/our-history

[4] WWW stránky: *Bluetooth Frequency Allocations/Frequency Bands*. [cit. 2016-2-20], [Online; navštívené 20.2.2016].
URL http://www.rfwireless-world.com/Tutorials/Bluetooth-frequency-allocations.html

[5] WWW stránky: *Bluetooth Basics*. [cit. 2016-2-23], [Online; navštívené 23.2.2016].
URL https://learn.sparkfun.com/tutorials/bluetooth-basics

[6] WWW stránky: *GAP*. [cit. 2016-4-13], [Online; navštívené 13.4.2016].
URL
https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap

[7] WWW stránky: *GATT*. [cit. 2016-4-13], [Online; navštívené 13.4.2016].
URL
https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt

[8] WWW stránky: *Bluetooth Low Energy (BLE) Central plugin for Apache Cordova*. [cit. 2016-4-16], [Online; navštívené 16.4.2016].
URL https://github.com/don/cordova-plugin-ble-central

[9] WWW stránky: *Cordova overview*. [cit. 2016-4-18], [Online; navštívené 18.4.2016].
URL https://cordova.apache.org/docs/en/latest/guide/overview/

[10] WWW stránky: *Angular Chart.js*. [cit. 2016-4-20], [Online; navštívené 20.4.2016].
URL http://jtblin.github.io/angular-chart.js/

[11] WWW stránky: *Chart.js*. [cit. 2016-4-20], [Online; navštívené 20.4.2016].
URL http://www.chartjs.org/

Prílohy

Seznam příloh

Príloha A

Obsah CD

- Zdrojové kódy aplikácie
- Elektronická verzia tejto práce vo formáte pdf
- Zdrojové súbory tejto elektronickej práce