TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Detektor přítomnosti pomocí Bluetooth Low Energy

Bakalářská práce

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

Autor práce: **David Tampier**
Vedoucí práce: Ing. Miroslav Holada, Ph.D.
Konzultant: Ing. Jiří Mareš, Ph.D.

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Detektor přítomnosti pomoci Blueto-oth Low Energy

Jméno a příjmení:	**David Tampier**
Osobní číslo:	M16000191
Studijní program:	B2612 Elektrotechnika a informatika
Studijní obor:	Elektronické informační a řídicí systémy
Zadávající katedra:	Ústav informačních technologií a elektroniky
Akademický rok:	**2019/2020**

Zásady pro vypracování:

1. Popište mechanizmus vyhledávaní bluetooth zařízení a porovnejte rozdíly mezi Bluetooth verzemi (2.x versus LE).
2. Vytvořte Qt aplikaci pro vyhledávání uživatelských bluetooth zařízení v dosahu přijímače. Vytvořte simulátor strany uživatelského zařízení (například ESP32).
3. Změřte a statisticky vyhodnoďte rychlosti detekce uživatelského zařízení v aplikaci v závislosti na okolí (vzdálenost, počet viditelných bluetooth zařízení, technologie bluetooth).
4. Diskutujte robustnost a bezpečnost detekce vybraných BT zařízení.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	cca 30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] TKÁČ, Josef. *Jak na Bluetooth v rekordním čase*. Praha: Grada, 2005. V rekordním čase. ISBN 80-247-1081-1.

[2] PUŽMANOVÁ, Rita. *Bezpečnost bezdrátové komunikace: jak zabezpečit Wi-Fi, Bluetooth, GPRS či 3G*. Brno: CP Books, 2005. ISBN 80-251-0791-4.

Vedoucí práce:	Ing. Miroslav Holada, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	9. října 2019
Předpokládaný termín odevzdání:	18. května 2020

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 18. října 2019

Prohlášení

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

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

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

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

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

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

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

31. 5. 2020 David Tampier

Detektor přítomnosti pomocí Bluetooth Low Energy

Abstrakt

Tato práce se zabývá testováním protokolu Bluetooth Low Energy. Předmětem testování jsou parametry ovlivňující latenci detekce. Nejdříve se práce věnuje popisu protokolu, ve větší míře jsou rozebrány části, které se týkají testovaných parametrů a bezpečnosti. Dále jsou popsány prostředky využité k testování a architektura aplikace. Statistické vyhodnocení výsledků je provedeno pomocí aplikace MATLAB a protokol je testován na vývojových deskách od společnosti Nordic Semiconductor. V závěru jsou rozebrána možná bezpečností rizika a prostředky, které protokol poskytuje pro jejich řešení.

Klíčová slova: BLE, MATLAB, latence, bezpečnost

Presence detector using Bluetooth Low Energy

Abstract

The thesis aims to test the Bluetooth Low Energy protocol. The main object of testing is the parameters influencing discovery latency. Firstly a broad overview of the protocol is described with emphasis on areas that deal with tested parameters and security. After that, the technologies used for testing are listed. Statistical analysis is performed with MATLAB application, and the protocol is tested on development kits from the Nordic Semiconductor company. Finally, the thesis deals with potential security risks and ways to solve them.

Keywords: BLE, MATLAB, latency, security

Obsah

Seznam zkratek

BLE	Bluetooth Low Energy
BR/EDR	Bluetooth Basic Rate/Enhanced Data Rate
SIG	Bluetooth Special Interest Group
IoT	Internet of things
PHY	Physical Layer
ISM	Industrial, scientific and medical
GFSK	Gaussian Frequency Shift Keying
CRC	Cyclic redundancy check
FHSS	Frequency Hopping Spread Spectrum
IEEE	Institute of Electrical and Electronics Engineers
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
SDIO	Secure Digital Input Output
L2CAP	Logical Link Control and Adaptation Protocol
GAP	Generic Access Profile
GATT	Generic Attribute Profile
ATT	Attribute Protocol
UUID	Universally Unique Identifier
API	Application Programming Interface
GUI	Graphical User Interface
SoC	System on a Chip
RAM	Random Access Memory
I/O	Input/Output
JTAG	Joint Test Action Group
SDK	Software Development Kit
IDE	Integrated Development Environment
LESC	Low Energy Secure Connections
ECDH	Elliptic-Curve Diffie–Hellman
DH	Diffie–Hellman
AES	Advanced Encryption Standard
OOB	Out Of Band
NFC	Near Field Communication
MITM	Man-In-The-Middle
PKK	Power Profiler Kit

Seznam obrázků

Úvod

Rozvoj oblasti Internet of things (IoT) roste v posledních letech s množstvím přibývajících zařízení téměř exponenciálně. Bezdrátový protokol BLE se na tomto poli stal velmi silným hráčem. Jeho nespornou výhodou je implementace v téměř každém chytrém telefonu. Tohoto faktu využila například aplikace eRouška, která má pomoci řešit současnou pandemii koronaviru. Aplikace dokáže sbírat informace o styku s potencionálně nakaženými osobami a zpětně uživatele varovat. Telefony využívají pro vysílání a přijímání rádiového signálu právě technologii BLE. Na základě síly přijímaného signálu lze vyhodnotit vzdálenost od možné infikované osoby. Aplikace si nepotřebuje vyměňovat velké objemy dat a zároveň potřebuje neustále běžet v pozadí a proto je technologie BLE vhodným řešením.

Společnost Jablotron, která se zabývá vývojem zabezpečovacích a ovládacích systémů by ráda protokol BLE využila pro nová inovativní řešení. Jedním z možných uplatnění je automatické odjištění a zajištění zabezpečovacího systému na základě BLE komunikace. Pokud by se osoba přiblížila k objektu na určitou vzdálenost, došlo by k ověření identity zařízení pomocí BLE a odjištění objektu. V momentě, kdy všechny ověřené zařízení opustí vymezený prostor, by nastalo automatické zajištění. Pro tuto aplikaci jsou klíčové určité parametry, které by protokol měl být schopný splnit.

Tato práce si proto klade za cíl klíčové parametry otestovat. Nejprve je vytvořena aplikační vrstva k protokolu na vývojových deskách, která splňuje funkcionalitu reálných zařízení. Pomocí aplikace je poté nasimulována možná komunikace mezi objektem a zařízením uživatele. Důležité informace o průběhu komunikace jsou zaznamenávány na PC a vyhodnocovány v prostředí MATLAB. Otestování bezpečnosti je provedeno pomocí třetího zařízení a aplikace Wireshark, kde je možné sledovat pakety, které si vývojové desky mezi sebou vyměňují.

1 Komunikační protokol BLE

1.1 Historie Bluetooth

První verzi Bluetooth vyvinula v roce 1994 firma Ericsson.[1] Původně byl Bluetooth zamýšlen bezdrátová komunikace na krátkou vzdálenost, která měla nahradit kabely počítačových periferií jako myš nebo klávesnice. Dnes již Bluetooth obsahuje téměř každé chytré zařízení od bezdrátových sluchátek po různá fitness zařízení. Podle zamýšlené aplikace může být zařízení schopné komunikovat pomocí BLE či klasického Bluetooth, jejichž rozdíly budou popsány v následující kapitole.

Společnosti jako Nokia nebo Intel během počátků vývoje Bluetooth pracovaly na velmi podobné technologii. V roce 1996 si naštěstí uvědomily, že pokud má daný protokol fungovat napříč spektrem různých zařízení, bude potřeba sjednocení. Proto v roce 1998 vznikla organizace Bluetooth Special Interest Group (SIG), která sjednotilo všechny technologie pod Bluetooth standart. Ještě na konci roku 1998 měla organizace víc jak čtyři sta členů. Dnes má již více než třicet tisíc členů, což vysvětluje výskyt technologie v téměř každém zařízení.[2]

Během následujících let poté vznikali vylepšené verze, které zlepšovali například bezpečnost, datový přenos a spotřebu energie. V roce 2010 vyšla Bluetooth specifikace 4.0, která poprvé obsahovala dva různé protokoly BLE a BR/EDR. Oba protokoly jsou vždy bohužel součástí jedné specifikace. BLE vznikl jako reakce na rozvoj IoT aplikací a jejich požadavky na nízkou spotřebu. Prvním telefonem implementujícím protokol BLE se stal iPhone4S v roce 2011.[1] Důležitou specifikací byla verze 4.2 uvedena v roce 2014, která přinesla podporu internetového protokolu IPv6 a zvýšenou bezpečnost. Stala se tak zásadní verzí pro oblast IoT a tuto verzi podporuje dnes již téměř každé zařízení. Nejnovější specifikace 5.1 a 5.2 se zaměřily téměř výhradně na protokol BLE, což signalizuje jeho budoucí důležitost.

1.2 Rozdíly BLE a Bluetooth Classic

Jak již bylo zmíněno, Bluetooth specifikace obsahuje dva rozdílné protokoly, které jsou vzájemně nekompatibilní. Nicméně většina mobilních zařízení podporuje oba protokoly. Největším rozdílem je datový přenos a spotřeba energie, což se odráží na využití v odlišných typech aplikací a systémů. Klasický Bluetooth (BR/EDR) je určen především pro streamovácí aplikace, přenos zvuku nebo souborů. Toho využívají zařízení jako myš, bezdrátová sluchátka, tiskárny nebo herní ovladače. Aplikace, které vyžadují nízký přenos dat při malé frekvenci, využívají výhradně BLE. Jedná

se o dálkové ovládaní v domácnostech, přenos údajů z fitness zařízení nebo vysílání reklamních zpráv do okolí. S příchodem verze 5, která přinesla zvýšený dosah až do vzdálenosti osmi set metrů, se BLE začalo využívat v průmyslové automatizaci. Zařízení jsou schopna vyměňovat si zprávy napříč celou továrnou.

1.3 Architekruta BLE

Dva hlavní pojmy v protokolu BLE jsou centrální zařízení a periferie. Typický příkladem centrálního zařízení je počítač, chytrý telefon nebo tablet. Škála periferií je velmi široká. Do této kategorie se může řadit snímač srdeční frekvence, fitness náramek, inteligentní sensor pohybu nebo chytré hodinky.

Periferie vysílá svoji přítomnost do okolí pomocí zpráv zvaných advertising packets a může přijmout výzvu k připojení od centrálního zařízení. Periferie má možnost fungovat pouze jako vysílač zpráv, což umožňuje zařízení obsahovat zredukovaný hardware a využívat jen část BLE protokolu. Typickým příkladem využití jsou majákové technologie, které mohou umožnit například lokaci v nákupních domech nebo vysílání reklamních sdělení v obchodech. Vysílač se od typické periferie dá odlišit pomocí obsahu paketů, které vysílá. Centrální zařízení může poté rozlišit, zda je možné s danou periferií navázat spojení. Periferie se v momentě navázání spojení stává slave a centrální zařízení master.

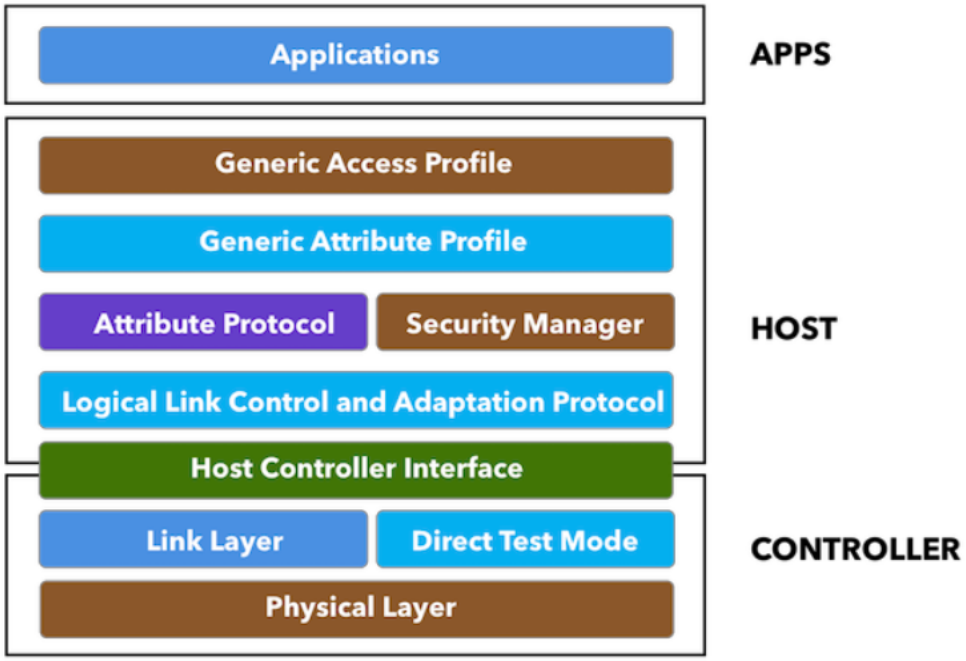
Centrální zařízení poslouchá přicházející zprávy a podle typu zprávy se může rozhodnout navázat spojení. Podle výpočetního výkonu může provozovat několik spojení současně. Zredukovaná verze centrálního zařízení se nazývá pozorovatel, který pouze přijímá zprávy a není schopen navázat spojení. BLE komunikace je asymetrická a většina řízení spadá do režie centrálního zařízení. To umožňuje periferiím významně uspořit energii a operovat na baterii v řádu několika měsíců. Některé zařízení mohou pracovat v obou rolích současně. Chytrý telefon může například sbírat data od senzorů v domácnosti a zároveň přeposílat data centrálnímu řídicímu systému.

Protokol je rozdělen do tří hlavní částí. Jsou jimi aplikační vrstva, hostitel a kontrolér zobrazené na obrázku.

Rozhraní hostitel-kontrolér umožnuje častí protokolu implementovat na samostatné čipy.

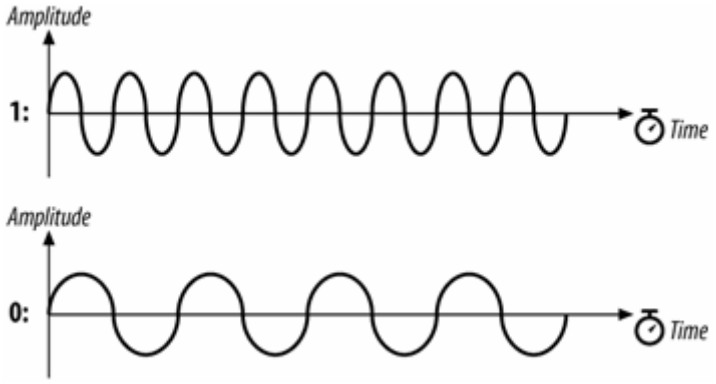
1.3.1 Fyzická vrsta

Fyzická vrstva se označuje jako PHY, jejím hlavním úkolem převod dat na rádiový signál a rekonstrukce dat z přijatého signálu. Rádio operuje v ISM pásmu 2.4GHz a má rozsah 2402MHz - 2480MHz. Jednotlivý kanál má šířku 2MHz, tudíž celé pásmo obsahuje čtyřicet kanálů. Pro advertisment paketů, navazování spojení jsou vyhrazeny pouze tři kanály, které jsou rozmístěny co nejdále od sebe, aby se předešlo rušení. Tato skutečnost výrazně snižuje dobu nalezení vysílajícího zařízení. Zbylé kanály jsou využívány pro komunikaci po navázání spojení. K omezení možnosti přetížení datových kanálů využívá BLE metodu FHSS. Během komunikace



Obrázek 1.1: Architektura BLE (převzato z [1])

jsou s vysokou frekvencí náhodně měněny komunikační kanály a posílaná zpráva se tak rozprostře po celém pásmu. Pokud dojde k chybnému přenosu, stačí opětovně poslat jen velmi malý datový úsek. Pro reprezentaci dat se využívá Gaussovo klíčování frekvenčním posuvem (GFSK). Výhodou je daleko nižší špičkový výkon oproti amplitudové modulaci a odolnost vůči šumu, který může změnit amplitudu signálu. GFSK kóduje data pomocí změny frekvence v nosném signálu. BLE používá dvoustupňové GFSK.[3] Při přenosu logické jedničky se zvýší frekvence o stanovenou odchylku a logická nula se vysílá bez odchylky.



Obrázek 1.2: Frekvenční modulace (převzato z [4])

Rádiu trvá několik set period, aby vyhodnotilo, zda se jedná o logickou jedničku či nulu. Frekvence signálu je 2.4GHz a přenosová rychlost má dvě modulační sché-

mata 1Mbps nebo 2Mbps. Zvýšení přenosové rychlosti lze dosáhnout čtyřstupňovou modulací nebo zvýšeným vzorkování, což by ovšem vyžadovalo daleko komplexnější vysílač a přijímač.

Hardware nemusí obsahovat vysílač nebo přijímač podle zamýšleného využití. Zmíněným majákovým technologiím stačí pro funkcionalitu pouze vysílač. Rozsah síly signálu se u vysílače může pohybovat mezi setinou až stovkou miliwattů. Pro verze 4.x je maximum deset miliwattů.[3] Při překročení maximálního přenosového výkonu může dojít nenávratnému poškození přijímače. Síla signálu přímo ovlivňuje rozsah.

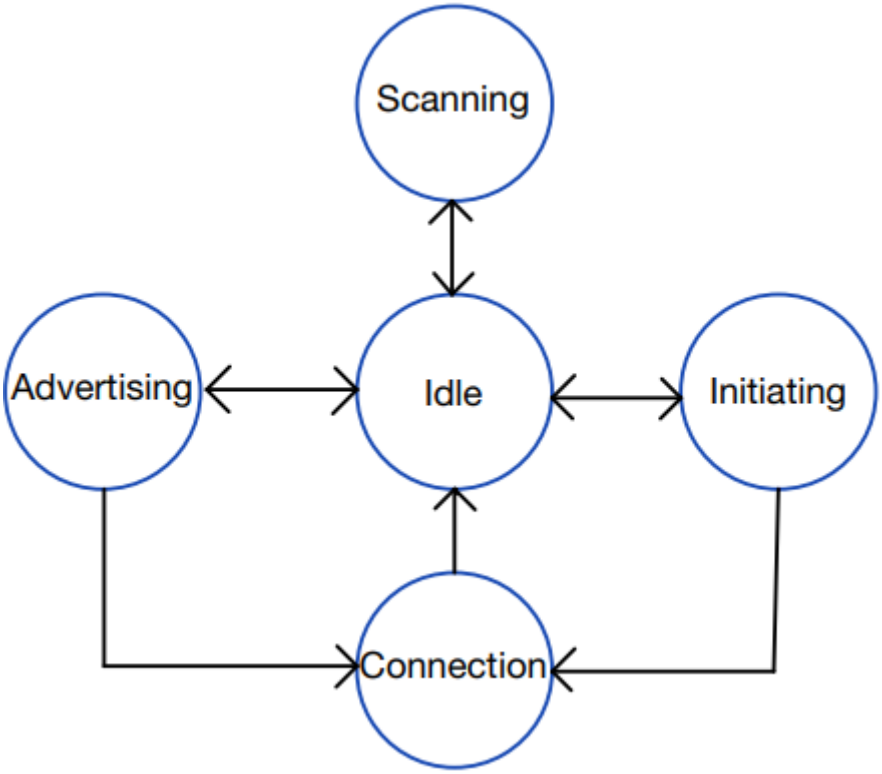
Minimální síla signálu, kterou dokáže přijímač dekódovat je -70dBm. Velmi nízká síla signálu ovšem způsobuje daleko vyšší bitovou chybovost a tudíž zpomaluje rychlost komunikace. Konzistentnost dat kontroluje cyklický redundantní součet, který funguje na principu kontroly zbytku modulárního dělení zprávy s předem určeným polynomem. Vhodně zvolený polynom dokáže ve zprávě odhalit jakoukoliv chybu určitého typu. BLE využívá standardizovaný polynom CRC-16-CCITT, který objeví jakoukoliv chybu ve zprávě menší než 32751 bitů. Citlivost přijímače může být dále ovlivněna okolním šumem a stavem počasí.

1.3.2 Spojová vrstva

Jedná se o první vrstvu, která je softwarově řízena. Poskytuje abstrakci nad fyzickou vrstvou a vytváří první datové struktury paketů. Paket není nic jiného než kolekce jednotlivých bitů o určité velikosti. Dále vrstva ovládá hardware, který výrazně urychluje operace jako CRC, datovou enkrypci a generování náhodných čísel. Spojová vrstva je implementována jako stavový automat, který určuje v jakých stavech se rádio může nacházet.

V záložním stavu (Idle) rádio nepřenáší ani nevysílá data a lze do něj vstoupit z jakéhokoliv jiného stavu. Záložní stav je velmi podobný spánkovému módu a spotřebovává velmi málo energie. Ve skenujícím stavu (Scanning) hledá rádio vysílající zařízení na třech stanovených kanálech. Zařízení má také možnost přijímat pouze specifické přicházející pakety a synchronizovat poslech s jejich frekvencí. Vysílající stav (Advertising) vysílá pakety do okolí, po každém poslání paketu čeká krátkou chvíli na odpověď od centrálního zařízení. Do zahajovacího stavu (Initiating) může přejít skenující zařízení, které se rozhodlo navázat spojení. V připojeném stavu (Connection) vzniklo spojení mezi periferií a centrálním zařízením a dochází k pravidelné výměně dat. Jako slave vstupuje zařízení do spojení z vysílajícího stavu. Slave je omezen na komunikaci s jediným masterem.

Pro navázání spojení se využívají adresy obou zařízení. Adresy mají délku šesti bajtů a mohou být dvojího typu veřejné a náhodné. Veřejná adresa je pevně daná a musí být zaregistrována pod organizací IEEE. Proto většina výrobců preferuje adresy náhodné, které nevyžadují žádnou registraci. Náhodná adresa se dělí na statickou a privátní. Statická adresa je ekvivalent veřejné, generuje se během startu zařízení a zůstává stejná po celou dobu jeho běhu. Privátní adresa řeší zabezpečení, k její opakované generaci dochází za běhu pomocí speciálního klíče a náhodného



Obrázek 1.3: Stavový automat spojové vrstvy

čísla. Ostatní zařízení se k ní mohou připojit, pokud si v předchozí komunikaci vyměnily speciální klíče.

1.3.3 Rozhraní hostitel-kontroler

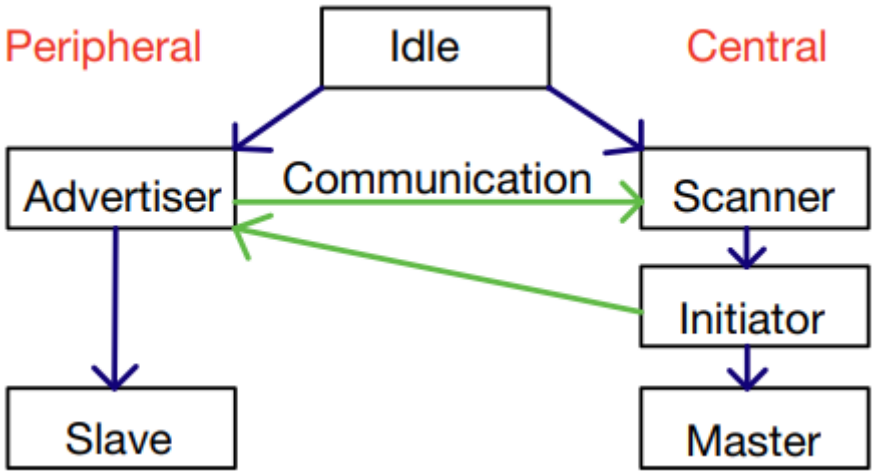
Požadavky na funkčnost rozhraní jsou popsány v Bluetooth specifikaci. Rozhraní definuje způsoby, kterými spolu hostitel a kontrolér mohou komunikovat. Hlavní důvod pro jeho existenci je možnost rozdělení těchto dvou vrstev do samostatných integrovaných obvodů. Vývojář proto může například pro osazení vyráběného plošného spoje využít integrované obvody od dvou různých výrobců a díky specifikaci má zajištěnou jejich interoperabilitu. Hardwarové rozhraní spojující integrované obvody mohou být USB, UART nebo SDIO. V případě SDIO je umístěn hostitel a kontrolér ve stejném obvodu. Rozhraní přenáší požadavky hostitele kontroléru a předává zprávy o důležitých událostech od kontroléru k hostiteli.

1.3.4 Logical link control and Adaptation protocol

L2CAP je přímo propojen se spojovou vrstvou a jeho jediný úkol je hladké přenášení zpráv z vyšších vrstev. Funguje jako multiplexor více protokolů z vyšších vrstev. Stará se zejména o fragmentaci do paketů a jejich opětovné sestavení v opačném směru. Řídí správné časovaní přenosů, a pokud se vyskytne problém, postará se o opětovné odeslání zprávy. V poslední řadě zajišťuje tvorbu bezpečného připojení.

1.3.5 Generic access profile

Tento protokol je pro práci nejdůležitější, a proto bude popsán nejpodrobněji. GAP definuje role a stavy ve kterých se zařízení mohou nacházet a parametry, které příslušné roli náleží. Implementace této vrstvy je dle specifikace[3] povinná pro všechna zařízení, co BLE protokol využívají. Čtyři role ve kterých se může zařízení nacházet jsou vysílač, pozorovatel, periferie a centrální zařízení. Jejich chování bylo již popsáno na začátku kapitoly1.3. Zařízení může mít více než jednu roli, ale může se nacházet pouze v jediné roli v určitém čase. Možný průběh rolí je zobrazen na obrázku 1.4.

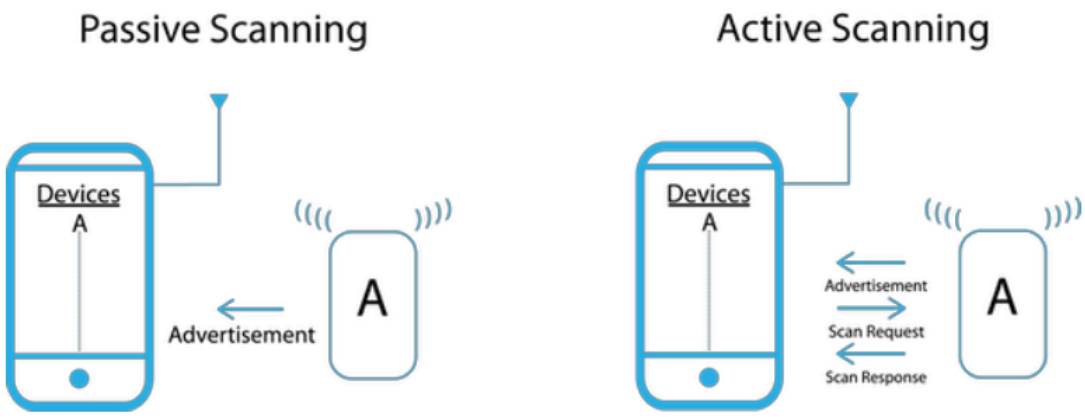


Obrázek 1.4: Možný průběh GAP rolí

Ve stavu inzerování vysílá zařízení pakety, které obsahují důležitá data pro možné připojení. Pakety jsou posílány pravidelně v rámci fixního intervalu, vliv tohoto intervalu na latenci detekce je později zkoumán. Pakety jsou postupně posílány na všech třech primárních inzerujících kanálech (37, 38 a 39) v rámci jednoho intervalu. Zařízení čeká po každém poslání krátkou dobu na stejném kanálu pro případ, že by došlo k odpovědi. Centrální zařízení analyzuje příchozí paket a podle jeho obsahu určí, zda je možné se k periferii připojit. Pokud se všechny důležité data nevejdou do primárních paketů, které mají omezenou velikost 31 bajtů, může přijít od centrální zařízení požadavek na poslání sekundárního paketu, který může mít až 254 bajtů. To umožňuje periferii zůstat v inzerujícím stavu bez nutnosti připojení a vysílat informace pro více zařízení najednou. Nevýhodou tohoto přístupu je nedostatek bezpečnosti.

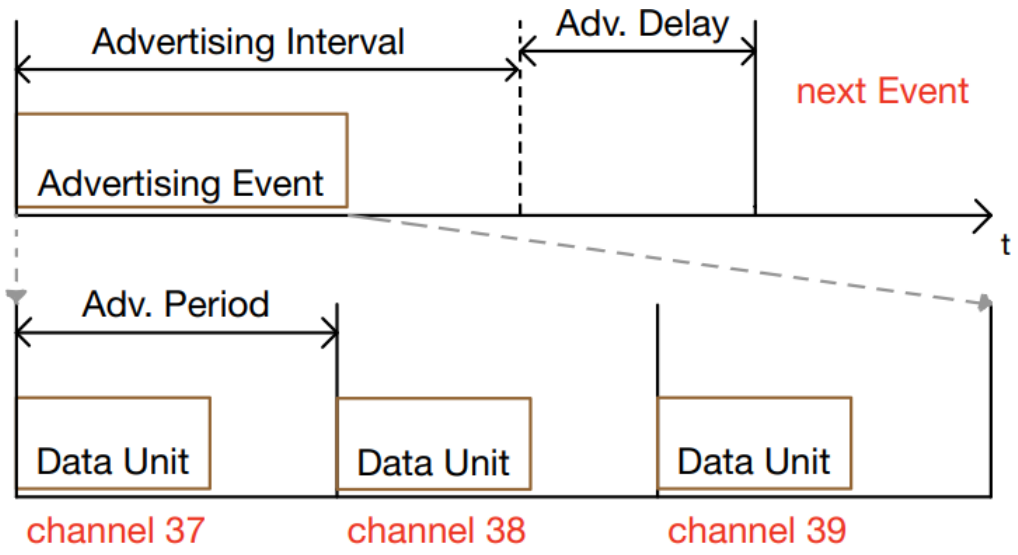
Centrální zařízení poslouchá po dobu fixního intervalu na jednom ze tří kanálů. Pro úspěšné objevení se musí periferie a centrální zařízení v jeden okamžik potkat na stejném kanálu. Pro zvýšení šance objevení mohou zařízení modifikovat několik parametrů, které jsou hlavním předmětem testovaní. Centrální zařízení, které vysílá požadavky na dodatečná data, se nachází v módu aktivního skenování, pokud nevyžaduje dodatečná data jedná se o skenování pasivní. 1.5

Inzerující událost proběhne v rámci jednoho vysílacího intervalu a během jedné



Obrázek 1.5: Pasivní a aktivní skenování (převzato z [1])

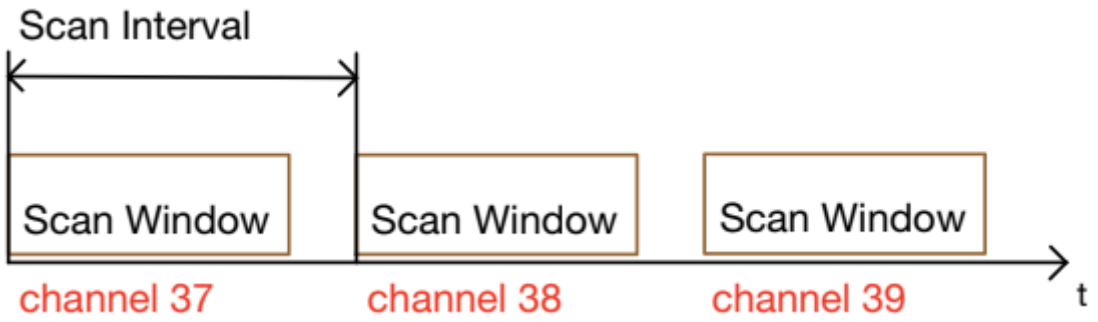
události se pošlou pakety na všech primárních inzerujících kanálech. Inzerující perioda je maximální doba, kterou může periferie čekat na odpověď po odeslání paketu. Zpoždění inzerce je náhodná doba, která je připojena na konec intervalu, aby se odlišila doba, ve které vysílají jednotlivá zařízení. Celý průběh inzerce je zobrazen na obrázku. (1.6) Události se mohou dále dělit podle toho, zda umožňují připojení,



Obrázek 1.6: Parametry inzerce

aktivní sken nebo cílí zprávu pouze určitým zařízení, které ji mohou pomocí klíče dekódovat. Nejdůležitějším parametrem je interval inzerce. Tento parametr zásadně ovlivňuje spotřebu energie a latenci detekce. Proto je jeho nastavení kompromis dobou běhu zařízení na jednu baterii a rychlostí detekce zařízení. Pro inzerované pakety existují v oficiální specifikaci šablony dat, které mohou pakety obsahovat.[5] Pakety mají standardně formát délka, typ a obsah a tyto tři hodnoty tvoří jednu datovou jednotku protokolu. Jednotlivé typy dat obsahují například lokální jméno zařízení, sílu přenosového signálu nebo příznakové bity, které specifikují možnosti zařízení.

Tři nejdůležitější parametry skenujícího zařízení jsou typ skenu 1.5, skenující okno a interval. Jejich význam je zobrazen na obrázku. 1.7 Pro navázání spojení se



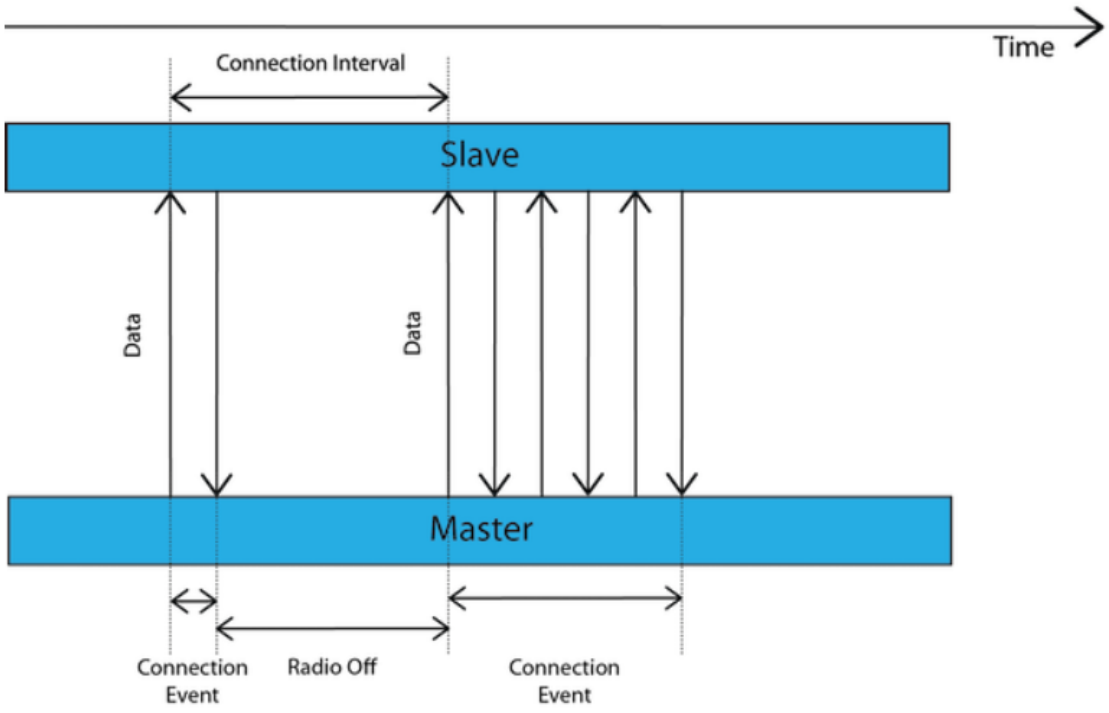
Obrázek 1.7: Skenovací parametry

musí obě zařízení nacházet ve stanovených rolích. Periferie musí inzerovat a v datech indikovat možnost připojení. Centrální zařízení skenuje okolí. Stanovené parametry musí umožnit, aby se s určitou pravděpodobností zařízení mohli potkat na jednom ze tří kanálů. Pro vytvoření připojení si zařízení musí vyměnit inzerční paket a paket se žádostí o spojení, v tento okamžik se začne formovat připojení. Veškeré další řízení jako parametry připojení a časování přebírá master, slave může pouze poslat preferované parametry připojení a master se může rozhodnout, zda je přijme či nastaví vlastní.

Během jedné komunikační události si master a slave posílají data, dokud obě strany nevyčerpají všechny zprávy, které chtějí poslat. Komunikační událost probíhá v pravidelných intervalech, dokud nedojde ke ztrátě spojení. Událost dále musí obsahovat vždy minimálně jednu datovou výměnu, jinak dojde k ukončení spojení. Opakování jednotlivých komunikačních událostí udává interval zobrazený na obrázku. (1.8)

Aby periferie ušetřila energii během komunikace, může využít parametru slave latence, který umožňuje periferii vynechat několik po sobě jdoucích komunikačních událostí bez toho, aby došlo ke ztrátě spojení. Tento parametr tak zvyšuje časový limit pro jedno úspěšné spojení, jehož rozsah se pohybuje mezi 0.1s - 32s. Dalším způsobem jak ušetřit energii může být rozšíření velikosti posílaných zpráv, menší část dat poté tvoří hlavičky jednotlivých vrstev, tím pádem stačí poslat menší celkový objem dat.

Velice důležitou částí protokolu je možnost filtrovat zařízení různými způsoby. Filtry se mohou použít jak pro inzerci tak skenování. Nejběžnějším způsobem filtrování je whitelist, který obsahuje seznam adres a jejich typů. Filtrování podle whitelistu se zpracovává ve spojové vrstvě a do vyšších vrstev přicházejí už jen zprávy od zařízeních ve whitelistu. Proto je tato možnost nejefektivnější způsob filtrování, protože minimalizuje operace ve vyšších vrstvách. Whitelist je nutné uchovávat v permanentní paměti, nejčastěji obsahuje čip přídavnou FLASH paměť, kde se zapisují potřebné údaje. Filtrování zařízení pomocí parametrů jako název zařízení nebo síla signálu se musí provádět v samotné aplikaci po přečtení a zparsování inzerujících dat.

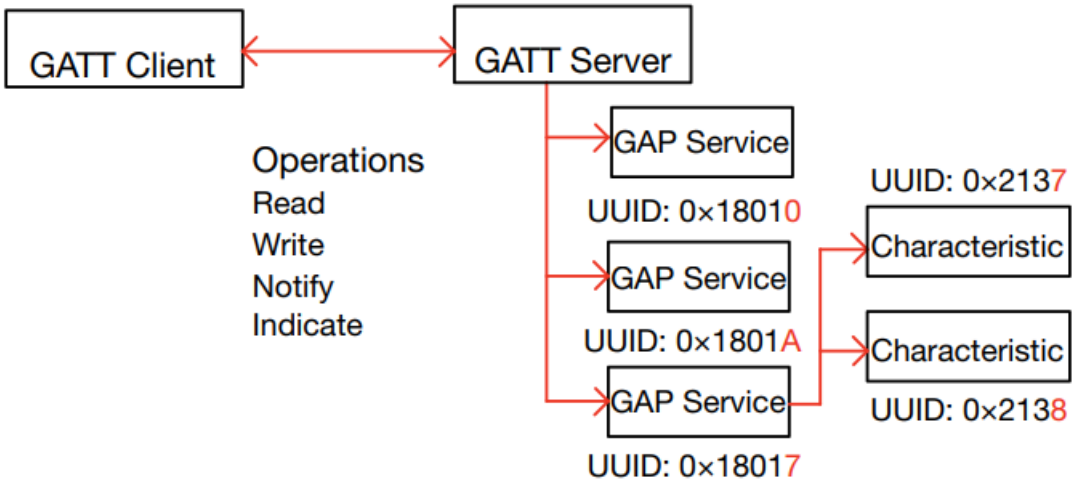


Obrázek 1.8: Průběh komunikace (převzato z [1])

1.3.6 Generic attribut profile and Attribute protocol

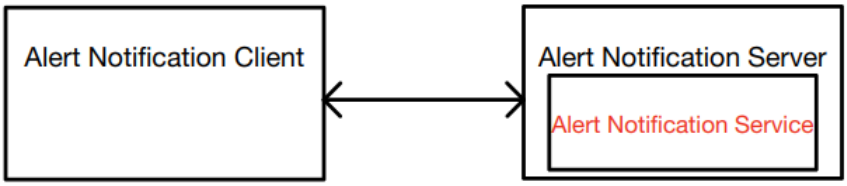
Generic attribut profile(GATT) a attribute protocol(ATT) jsou protokoly, které definují komunikační datové struktury. ATT slouží jako základní datový rámec pro GATT. Základní role, v kterých zařízení v těchto protokolech operují, jsou klient a server. Klient posílá serveru požadavky a ten posílá zpět odpovědi. Požadavky klienta mohou mít typ příkazů nebo žádostí. Zásadní rozdíl je, že žádost vyžaduje zpětnou odezvu, jedná se tedy o daleko bezpečnější řešení, klient si může být jistý, že se požadovaná operace vykonala. Komunikace ve směru server-klient může probíhat prostřednictvím notifikací nebo indikací. Indikace jsou bezpečnější variantou, protože vyžadují potvrzení převzetí klientem.

Základní datovou jednotkou, kterou server poskytuje klientovi, je atribut definovaný v ATT. Atribut se skládá z identifikátoru, hadlu a povolení. Identifikátor (UUID) je 16 nebo 128 bitové číslo, které určuje, co data v atributu reprezentují. Data mohou reprezentovat například napětí baterie nebo teplotu v místnosti. Specifikace definuje běžně používaná data jako 16 bitové UUID ve speciálním dokumentu. Pokud si chce programátor vytvořit novou reprezentaci dat, musí si nadefinovat vlastní UUID, to ovšem musí být již 128 bitové a zvyšuje se datový objem, který je nutné poslat. Handle je 16 bitová adresa, kterou server přiřazuje každému atributu, aby se na něj klient v případě potřeby mohl odkazovat. Povolení udává, zda je atribut určený pouze pro čtení nebo je do něj možné zapisovat. GATT skládá atributy do složitějších datových struktur, kterými jsou služby, charakteristiky a profily. Ukázka možné datové hierarchie na serveru je na obrázku. 1.9



Obrázek 1.9: Rozložení datové struktury na serveru

GATT dále definuje metody, které určují, jak s danými službami a jejich charakteristikami operovat. Služba se chová jako datový typ kontejner, seskupuje více charakteristik, které spolu logicky souvisí dohromady. Například servis teploty může poskytovat teplotní a napěťovou charakteristiku čidla. Charakteristiky jsou vždy součástí služeb a představují informaci, kterou chce server poslat klientovi. Charakteristiky mohou kromě nosné informace obsahovat pomocné atributy, které určují možné využití nebo pomocné informace dokumentující hodnotu. Bluetooth SIG opět vytvořila pro často využívané charakteristiky a služby předpřipravené standarty, které může vývojář využít bez nutnosti vytvářet vlastní datové typy. Profily se na rozdíl od služeb definují chování jak serveru, tak klienta. Definují komunikaci a bezpečnostní požadavky pro přenos datových typů. Služby jsou tedy podmnožina profilů. Poslední zásadní část BLE protokolu je manažer bezpečnosti, který je detailně popsán v poslední kapitole.



Obrázek 1.10: Ukázka profilu (převzato z [6])

2 Použité technologie

2.1 Vývojové prostředí Qt a deska ESP32

Původní vývoj testovací aplikace probíhal na platformě Qt pod operačním systémem Windows a na desce ESP32. Záměrem bylo, že PC bude simulovat stranu centrálního zařízení, zároveň na něm bude možné vypisovat výsledky testů a zobrazovat je pomocí Qt v GUI. Podle oficiální dokumentace poskytuje Qt API pro vývoj centrálních zařízení na platformě Windows.2.1

API Feature	Android	iOS	Linux (BlueZ 4.x/5.x)	macOS	UWP (Universal Windows Platform)	Win32
Classic Bluetooth	x		x	x	x	x
Bluetooth LE Central	x	x	x	x	x	x
Bluetooth LE Peripheral	x	x	x	x		
Bluetooth LE Advertisement & Scanning						

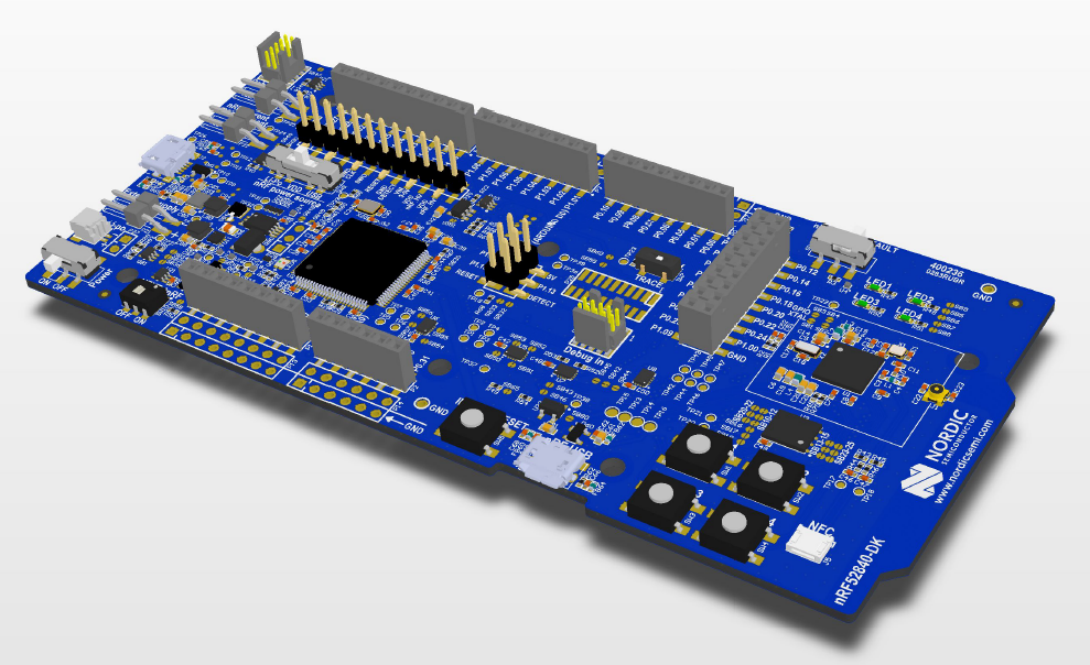
Obrázek 2.1: Podpora Bluetooth API na platformách (převzato z [7])

Pomocí API bylo možné skenovat okolí a navazovat spojení s inzerujícími zařízeními. Problém nastal při párovaní komunikující zařízení a ukládání informací do whitelistu. Bylo zjištěno, že Windows neposkytuje žádné API, které by umožňovalo programově párovat zařízení.[8] Tudíž není možné ovlivňovat filtrování zařízení pro testování. Prvotní způsob řešení problému byl přechod pod virtuální Linux. Nicméně konečným zvoleným prostředkem se staly vývojové desky od společnosti Nordic Semiconductor, které jsou speciálně vyráběny pro vývoj BLE prototypů. Na základě zvolených prostředků byl vývoj přesunut do prostředí MATLAB.

2.2 nRF Development Kits

K vývoji aplikace byly použity desky NRF52840. NRF52840 jsou multi protokolové SoC. Na jednom čipu integruje procesor, RAM a Flash paměť, Bluetooth 5, Zigbee, ANT a mnoho různých I/O ovladačů. Kromě NRF52840 se na desce nachází tlačítka, diody, konektory k univerzálním I/O pinům, NFC a BLE anténa. [9]

SoC obsahuje procesor od společnosti ARM, jedná se o typ AMR Cortex-M4. ARM procesory mají značně zjednodušenou instrukční sadu oproti počítačovým procesorům x86. To má za následek, že procesor spotřebovává daleko méně energie



Obrázek 2.2: Vývojová deska NRF52840 (převzato z [10])

a nepotřebuje žádné chlazení. Tohoto faktu začíná využívat značná část dnešních serverů. ARM Cortex-M4 je specificky navržen pro zpracovávání a kontrolu signálů a operuje na 64MHz, což dále snižuje jeho spotřebu. Pro matematické výpočty obsahuje doplňkový koprocesor.

SoC dále poskytuje 256kB RAM a 1MB Flash paměti, která je rozdělena mezi data programu a pomocné úložiště pro všechny protokoly. Uchovává se v ní například whitelist zařízení.(1.3.5) Pro komunikaci desek s počítačem využívám zabudovaný UART s podporou RS232 standartu. Na měření času jsem využil vestavěné hodiny reálného času. Pro debuggování má deska zabudovaný debugger od společnosti Segger, který komunikuje s procesorem pomocí speciálního rozhraní JTAG. Společnost Segger poskytuje také vývojové prostředí, které umožňuje s debuggerem lehce pracovat.

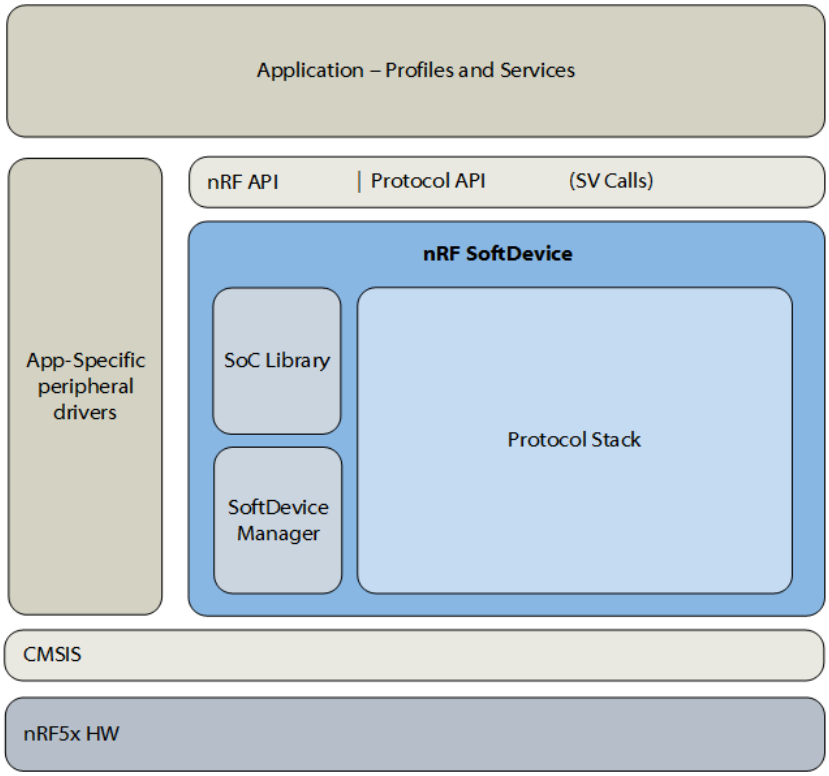
Pro možnost odposlouchávat komunikaci mezi deskami jsem využil nRF51 dongle, který neposkytuje takové možnosti jako vývojové desky. Je vhodný právě pro odposlech komunikace nebo jednoduchou emulaci periferie či centrálního zařízení.

2.3 nRF5 SDK a Segger Embedded studio

Nordic Semicondur zdarma poskytuje k vývoji aplikací pro své produkty sadu vývojových nástrojů (SDK). Tato sada obsahuje ovladače, knihovny a API pro ovládání protokolů. Celé SDK je napsáno v programovacím jazyku C, který je i v dnešní době převažující jazyk vestavěných systémů. Smyslem SDK je poskytnutí abstrakční vrstvy pro ovládání hardwaru. Programátor tak například nemusí zapisovat do speciálních funkčních regiristrů, pokud chce změnit frekvenci s jakou běží časovač,

ale stačí inicializovat modul, který s časovačem pracuje a poté zavolat funkci se správnými parametry. Tato funkce implicitně zajistí ovládání potřebných registrů. SDK dále obsahuje spoustu užitečných příkladů, jak pracovat s BLE protokolem. Příklady mohou posloužit jako šablona, která může být podle potřeb rozšířena.

Pro ovládání BLE slouží SoftDevice knihovna. Tato knihovna je již zkompilována do binárních souborů, tudíž funguje jako černá skříňka, kde má programátor přístup pouze k funkčním deklaracím a jejich dokumentaci. Jedná se o poskytnutí služeb pro ovládání BLE podobně jako tomu je u firmwaru, co ovládá zařízení. Ukázka knihovní struktury je uvedena na obrázku. 2.3 SoftDevice podporuje celou sadu BLE protokolů, umožňuje například zařízení udržovat až dvacet souběžným spojení. Komunikace s aplikační vrstvou probíhá jako reakce na události, které se v protokolu odehrají. Programátor může využít předdefinované výčtové typy a zaregistrovat je jako posluchače určitých událostí. I když C není objektově orientovaný jazyk, rozdě-



Obrázek 2.3: Architektura nRF5 SDK (převzato z [11])

lení knihovny se velice podobá objektovým principům. Základní knihovní jednotkou je hlavičkový soubor, který může reprezentovat například časovač, UART, prostředky pro logování informací nebo ovládání určité vrstvy protokolu. Hlavičkový soubor obsahuje datové typy a funkce, které s daným objektem souvisí. Využití jednotlivých modulů se dá nastavit ve speciálním hlavičkovém souboru sdk-config.h. Jako příklad použití modulu lze uvést logovací modulu. Logovací modul umožňuje posílat kontrolní výpisy za běhu aplikace. Každý modul se většinou inicializuje funkčním makrem. Logovacímu modulu můžeme předat jako parametr odkaz na časovač, aby

byl schopen k výpisům přikládat čas. Poté, co modul nainicializejeme mu přiřadíme backend, přes který vypisuje informace. Máme možnost využít RealTime Transfer technologie, což je speciální protokol vyvinutý firmou Segger, nebo UART rozhraní. Posílané zprávy můžeme členit do čtyř kategorií ERROR, WARNING, INFO, DEBUG. Klíčovými moduly, které byly využity pro vývoj testovací aplikace, jsou ovladače BLE protokolu, logovací modul, časovací modul, kryptografická knihovna pro šifrování dat a modul pro ovládání UART komunikace. Všechny operace, ve kterých hrozí riziko chybové stavu, mají speciální návratovou hodnotu. Jedná se o 32 bitový integer, kde každé číslo reprezentuje konkrétní chybu, která nastala. Toto číslo se pak předá makru, které zkontroluje, zda operace proběhla bez chyb, případně dokáže programátora informovat o konkrétním významu dané chyby.

Pro vývoj BLE aplikace jsem využil Embedded Studio od společnosti Segger. Embedded studio je IDE, které je speciálně navržené pro psaní vestavěných systémů v jazyku C a C++. IDE se zaměřuje na podporu všech ARM procesorů a obsahuje nezbytné prostředky, které jsou potřeba k vývoji vestavěných aplikací. Uživateli poskytuje různé typy kompilátorů, zabudovaný debugger a mnoho dalších výhod. Společnost Nordic Semiconductor uzavřela se společnostní Segger partnerskou smlouvu. Pokud uživatel vyvíjí software pro jakýkoliv produkt od Nordic Semiconductor, je IDE zdarma pro nekomerční i komerční účely. Další výhodou je přímá podpora nRF5 SDK. Uživateli pak stačí v IDE nahrát cestu k SDK a může jednoduše využívat všechny ovladače a knihovny, které sada vývojových nástrojů poskytuje.

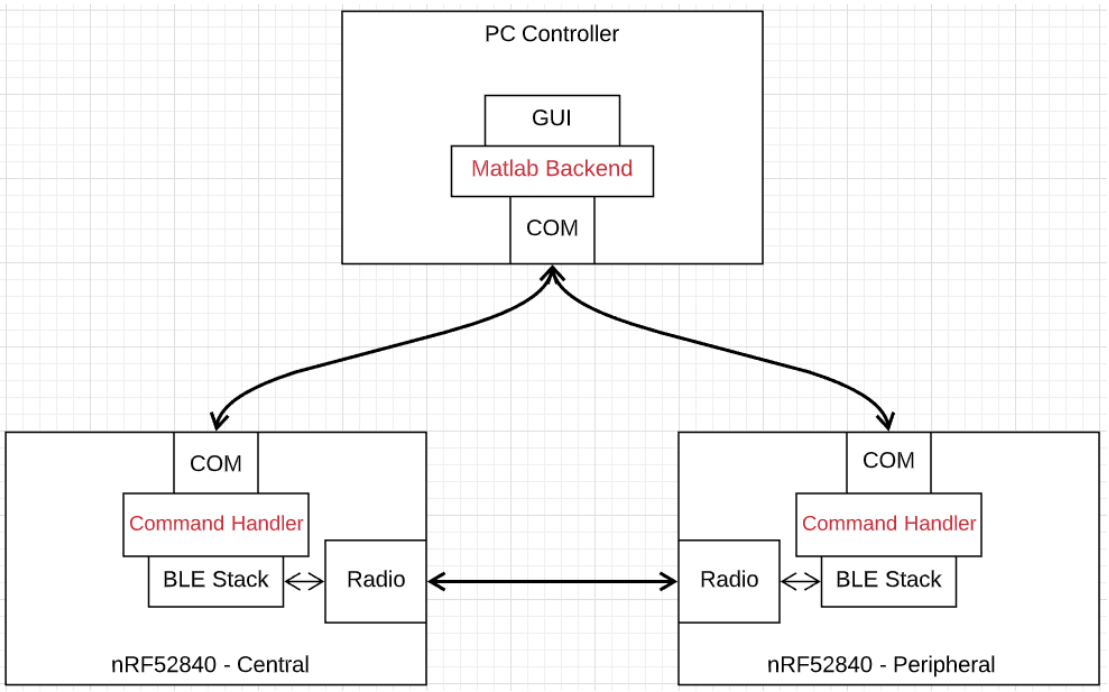
3 Testovací aplikace

3.1 Cíl aplikace

Aplikace si klade za cíl otestovat a statisticky vyhodnotit parametry, které ovlivňují latenci detekce a formování připojení mezi zařízeními. Další potřebnou funkcionalitou je možnost ovládání bezpečnostních parametrů BLE protokolu, aby bylo možné pozorovat změny v aplikaci Wireshark. Naprogramovanou funkcionalitu a spouštění testů je možné ovládat z jednoduchého GUI.

3.2 Celková anatomie aplikace

Celková aplikace se dá rozdělit na tři hlavní části, její architektura je zobrazena na obrázku. (Obr. 3.1) Spodní bloky reprezentují funkcionalitu na vývojových deskách.



Obrázek 3.1: Celková architektura aplikace

Jeden blok reprezentuje periferii a druhý centrální zařízení. Horní blok představuje

aplikaci, která běží na PC. Na deskách je naprogramována potřebná funkcionalita ovládající BLE protokol, která je přístupná pomocí jednoduchých příkazů. Ty může PC posílat přes virtuální UART, fyzická komunikace probíhá přes mikro USB kabel. Informace o důležitých událostech posílají desky zpět do PC, kde proběhne vyhodnocení.

3.3 Aplikace na deskách nRF52480

Pro aplikaci desek jsem využil jako šablonu příklad BLE Interactive Command Line nacházející se v SDK. [12] Příklad obsahuje strukturu pro datovou komunikaci a základy pro práci v roli periferie a centrálního zařízení. V aplikaci se nachází kód pro běh v obou módech. Jednotlivé logické celky jsou rozděleny do zdrojových souborů typu c. Jejich API je přístupné pomocí hlavičkových souborů.

3.3.1 Pomocné moduly

Hlavní funkce programu má klasickou strukturu vestavěných aplikací. V první části proběhne inicializace všech potřebných modulů a poté program vstoupí do nekonečné smyčky, kde program reaguje události. Pro omezení konsumpce proudu slouží modul řízení spotřeby. V případě, že není potřeba reagovat na nastalé události, umožní procesoru přejít do spánkového režimu a značně se tak ušetří spotřeba aplikace.

Pro zápis pomocných dat slouží logovací modul. Aby výpis obsahoval čas, je potřeba při inicializaci poskytnou pointer na funkci, které je schopná čas vyčíst. Logování je využito pro posílání informací řídícímu PC. Zprávy jsou rozlišovány do tří typů-*@message*, *@event* a *@data*. Na tyto typy poté řídící PC reaguje různými způsoby. *@message* má pouze informativní charakter. *@event* značí, že došlo k události, kterou je potřeba uložit pro pozdější vyhodnocení. *@data* signalizují, že je potřeba změnit stav do vnitřní proměnné v MATLAB aplikaci.

Pro záznam doby se vytvoří instance modulu časovač, který přebírá hodnoty od hodin reálného času. Aplikace inicializuje časovač a přiřadí ho k logovacímu modulu. Časovač se také využívá pro pravidelné obnovování informacích o zařízeních v paměti, které je možné posílat. Dále existuje možnost časovač resetovat, aby bylo možné časy na obou deskách synchronizovat pro měření.

Aby správně fungovala komunikace, je potřeba inicializovat korektně parametry modul reprezentující UART a určit ho jako defaultní komunikační kanál. Přenosová rychlost je nastavena na 115200 bit/s. Celkově má jedna zpráva 9 bitů, kde 8 bitů reprezentuje data. Poslední bit signalizuje ukončení zprávy. Zprávy jsou ve formátu malý endian, nejvíce významný bit je tedy na konci.

3.3.2 Ovládání BLE protokolu

Pro ukládání informací z BLE protokolu je nutné často dynamicky alokovat a uvolňovat paměť. Místo klasických funkcí malloc a free poskytuje SDK pro práci s pamětí

svůj vlastní modul. Tento modul pracuje na stejném principu. Pro zobrazování stavu, ve kterém se zařízení nachází, se využívají modul ovládající diody. Ty mohou signalizovat, zda zařízení inzeruje, skenuje nebo je připojené.

Při inicializaci BLE protokolu se musí určit pevně adresa v paměti. SoftDevice adresu využívá, jako orientační bod pro ukládaní informací, které souvisí s jednotlivými vrstvami protokolu. Základní konfigurace nutných parametrů pro běh protokolu je provedena pomocí statických hodnot z konfiguračního hlavičkového souboru *sdk-config.h*. Pro případ reagování na určité události je nutná incializace odpovídajících *handlerů*. Jednotlivým *handlerům* se dá přiřazovat různá priorita. Základním kritériem pro dělení událostí je, zda pocházejí od periferie nebo centrálního zařízení. Dělit události je také možné, podle jednotlivých funkčních modulů ze kterých pocházejí.

Mimo přímého volání funkcí ze *SoftDevicu* má programátor možnost využít moduly, které poskytují vrstvu abstrakce nad procedurálním programováním. Jednotlivé části BLE protokolu zapouzdžují do datového typu struktura. Tato struktura poskytuje pointery na funkce, které mění data, co se ve struktuře nachází. Jedná se tedy o analogii objektů v jazyce C. Instance modulů se dají vytvářet pomocí funkčních maker. Aplikace využívá skenující, inzerující a bezpečnostní modul pro dynamické měnění parametrů, které se týkají latence a bezpečnosti. Tyto funkce jsou poskytnuté jako API a využívá je modul, který reaguje na příkazy od řídícího PC. Pokud si BLE potřebuje uložit informace o partnerském zařízení jako adresu zařízení nebo kryptografický klíč využívá k tomu speciálně vyhrazenou část Flash paměti.

3.3.3 Řízení komunikace

Modul řízení komunikace definuje strukturu jednotlivých příkazů a přiřazuje jim *handlery*, které se vykonají, pokud řídící PC pošle odpovídající zprávu. SKD umožnuje příkazy dělit do úrovní. Proměnné parametry se registrují pomocí syntaxe *příkaz <parametr>*. Jednotlivé úrovně příkazů vytvářejí stromovou strukturu. V níže zobrazených seznamech jsou popsány jednotlivé příkazy pro periferii a centrální zařízení.

1. Příkazy pro periferii
    - adverise
        - on - Spustí inzerci.
        - off - Vypne inzerci.
    - display connected - Zobrazí připojená zařízení.
    - display bonded - Zobrazí spárovaná zařízení.
    - privacy
        - on - Inzerce probíhá pro vybráná zařízení z whitelistu.
        - off - Inzerce probíhá pro veškerá zařízení.

- advertising interval <value> - Změní hodnotu inzerujícího intervalu.
- transmit power <value> - Změní sílu signálu při inzerování.

2. Příkazy pro centrální zařízení

   - scan
     - on - Zapne sken.
     - off - Vypne sken.
   - connect - Zformuje připojení pomocí veřejné adresy.
   - private connect - Zformuje připojení pomocí uložených informací, vyžaduje spárování.
   - disconnect
   - pair - Pokusí se spárovat zařízení, musí být zformované připojení.
   - display advertisers - Zobrazí inzerující zařízení.
   - display connected - Zobrazí připojená zařízení.
   - display bonded - Zobrazí spárovaná zařízení.
   - scan window <value> - Změní hodnotu skenovacího okna.
   - whitelist
     - on - Zapne filtrování pro skenování.
     - off - Vypne filtrování pro skenování.

3. Příkazy pro obě zařízení

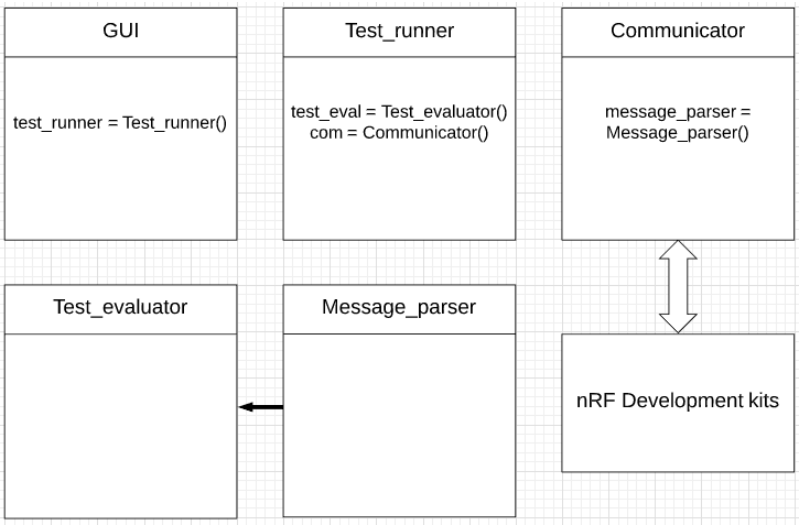
   - synchronize clock - Synchronizuje hodiny na obou deskách.
   - test clock - Provede kontrolní výpis časového *offsetu* mezi deskami.
   - remove bonds - Odstraní informace o zpárovaných zařízeních.

3.4 Aplikace MATLAB

Hlavní požadavky na funkcionalitu v MATLABu jsou komunikace s deskami, příjem a parsování zpráv podle určeného protokolu. (3.3.1) Funkcionalita byla rozdělena do čtyř tříd, které řeší logiku a GUI, které umožňuje s danými třídami pracovat. Hierarchie tříd je zachycena na obrázku. (3.2)

3.4.1 Komunikátor

Komunikátor zajišťuje čtení zpráv a posílání příkazů. Ve svém konstruktoru inicializuje sériový port se správnými parametry, aby mohla probíhat komunikace s deskami. Pro čtení příchozích zpráv registruje posluchače. Ty se spustí v momentu, kdy se v příchozích datech objeví znak, který signalizuje ukončení řádky. Pomocí parsovací třídy se vyhodnotí typ zprávy a předá se odpovídající parsovací rutině. Komunikátor dále poskytuje metody pro všechny potřebné příkazy.



Obrázek 3.2: Schéma tříd

3.4.2 Parsovač zpráv

Parsovač poskytuje metody pro identifikaci typů zpráv *@message, @event, @data*, přebírá příchozí zprávy a provádí jejich analýzu. Na zprávu typu *@message* nijak nereaguje, jelikož má pouze informativní charakter. Z typu @event získává informaci, o jakou událost se jedná, čas v který nastala a příznak, zda bylo objeveno testovací zařízení v roli periferie. Tyto informace ukládá do logu událostí, který může na vyžádání předat evaluátorů testů. Zpráva typu *@data* slouží pro nastavování vnitřních stavů, které korespondují se stavem desek. Tyto stavy slouží pro jako zábrana proti spuštění nelogické sekvence příkazů deskám. Není možné například vytvořit spojení, pokud periferie není v módu inzerce.

3.4.3 Evaluátor testů

Cílem evaluátoru je statisticky vyhodnotit naměřená data. Aby mělo měření statistický význam, je každý parametr mnohokrát testován ze stejných podmínek. Jelikož se nepodařilo vytvořit paralelní synchronizování časovačů desek pomocí konstrukce parfor, kterou MATLAB poskytuje ve svém paralelním toolboxu, vzniká mezi deskami kvůli sekvenční povaze běhu programu při synchronizaci určitý konstantní *offset*, který je většinou v řádu jednotek milisekund. Protože Windows není systém reálného času, nedá se zaručit, že časovače synchronizuje vždy se stejným *offsetem*. Proto evaluátor hlídá pomocí příchozích zpráv, že daný *offset* nikdy nepřesáhne určitou mez. S danou odchylkou se poté musí počítat jako nejistotou měření. Firmu Jablotron ovšem zajímají latence v řádech desetin sekund, na praktický dopad měření nemá tedy *offset* závažný vliv. Evaluátor má dále funkcionalitu na spočtení důležitých časů z naměřených dat. Například vyhodnocení doby od začátku inzerce a objevení zařízení. Tyto časy si podle předaných parametrů ukládá do datových struktur pro pozdější vyhodnocení. Na konci běhu specifického testu je evaluátor schopný data vyhodnotit a zobrazit výsledky do grafu. Z naměřených hodnot spe-

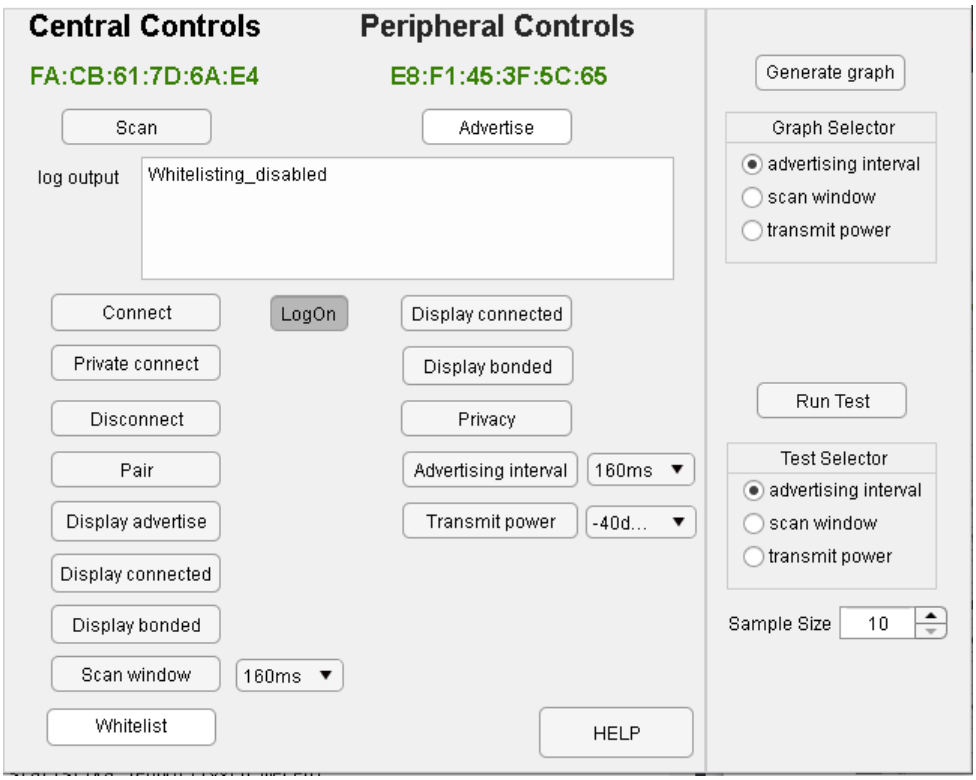
cifického parametru spočte střední hodnotu a vypočte, s jakou pravděpodobností spadá naměřená hodnota do určitého intervalu.

3.4.4 Spoušťeč testů

Testy se dají charakterizovat jako specifická sekvence po sobě jdoucích příkazů. Tematicky se rozdělují do třech logických celků. První celek by se dal nazvat navození potřebných předpokladů pro test. Velká část je pro většinu testů stejná. Jedná se o inicializaci datových záznamů a synchronizaci časovačů. Další předpoklady jako zapnutí skenování nebo nastavení whitelistu jsou už pro různé testy odlišné. V druhé části probíhá samotný test. Mění se zkoumaný parametr a výsledky měření se zapisují do příslušných datových struktur. Počet opakování měření se stejnými hodnotami se udává jako vstupní parametr. V poslední části proběhne deinicializace do původního stavu před začátkem měření. Probíhá například vypnutí skenu. Na konci testu se vykreslí graf s výsledky. Tím je test ukončen.

3.4.5 Graphical user interface

GUI slouží jako testovací platforma pro ověření funkcionality jednotlivých příkazů a zároveň je možné spouštět testy. Ukázka GUI je zobrazena na obrázku. (3.3) Na levé straně se nachází tlačítka pro jednotlivé příkazy. Na pravé straně je možné spouštět test s určitým počtem vzorků a generovat graf.



Obrázek 3.3: Ukázka GUI

4 Výsledky testování

Každý provedený test má stejný charakter. Pro testovaný parametr se provede opakované měření a vypočte se střední hodnota. Všechny ostatní parametry jsou během testu konstantní. Počet opakovaných měření byl zvolen tak, aby se chyba měření vždy pohybovala v řádu desetin sekundy při intervalu spolehlivosti 95%. Maximální rozsah chyby byl volen s ohledem na požadavky, které by měla splňovat zamýšlená aplikace automatického odjišťování. Výsledky konkrétního testu jsou vykresleny do grafu s pomocnou tabulkou, kde je uvedena statistika jednotlivých měření.

4.1 Testování latence detekce

Testování latence detekce měří dobu od vstupu uživatele do rozsahu, který umožňuje komunikaci mezi centrálním zařízením a periferií, po detekci periferie. Vstup uživatele simuluje testovací aplikace spouštěním skenování v náhodném intervalu. Čas měření je definován jako doba od zapnutí skenovaní a objevení periferie centrálním zařízením. Test předpokládá, že obě zařízení jsou již spárována, a proto využívá na straně centrálního zařízení whitelist (1.3.5) pro filtrování příchozích paketů, což může urychlit detekci.

4.1.1 Vzdálenost zařízení

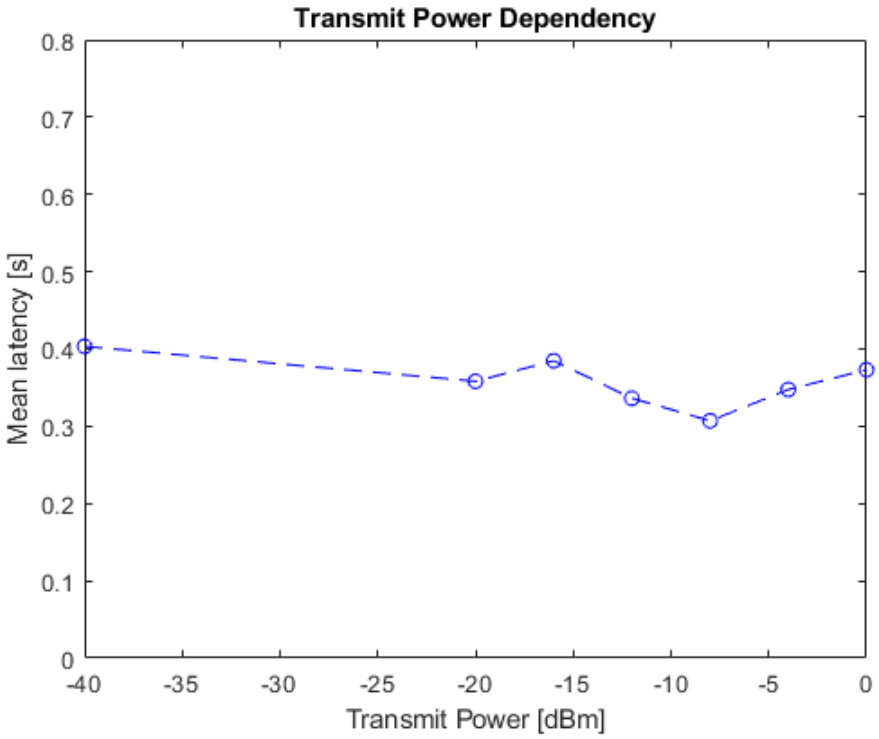
Vzdálenost zařízení je simulována měnící se silou inzerujícího signálu. Výsledek testu je zobrazen na obrázku (4.2) a statistika měření na obrázku. (4.1) Velikost vzorku pro

transmit power	mean values	margin of error
-40dBm	0.4033	0.1191
-20dBm	0.3582	0.0882
-16dBm	0.3847	0.0902
-12dBm	0.3362	0.0937
-8dBm	0.3070	0.0646
-4dBm	0.3474	0.0798
-0dBm	0.3731	0.0927

Obrázek 4.1: Statistika pro sílu signálu

uvedené měření byla třicet opakování. Výchozí hodnota inzerujícího intervalu byla

nastavena na 640ms a skenovací okno bylo nekonečné, což znamená, že centrální zařízení nikdy nepřestane skenovat. Z grafu (4.2) je vidět, že čas má pro měnící se sílu signálu konstantní charakter. To znamená, že síla signálu určuje pouze dosah připojení a nijak neovlivňuje rychlost detekce.



Obrázek 4.2: Závislost latence na síle signálu

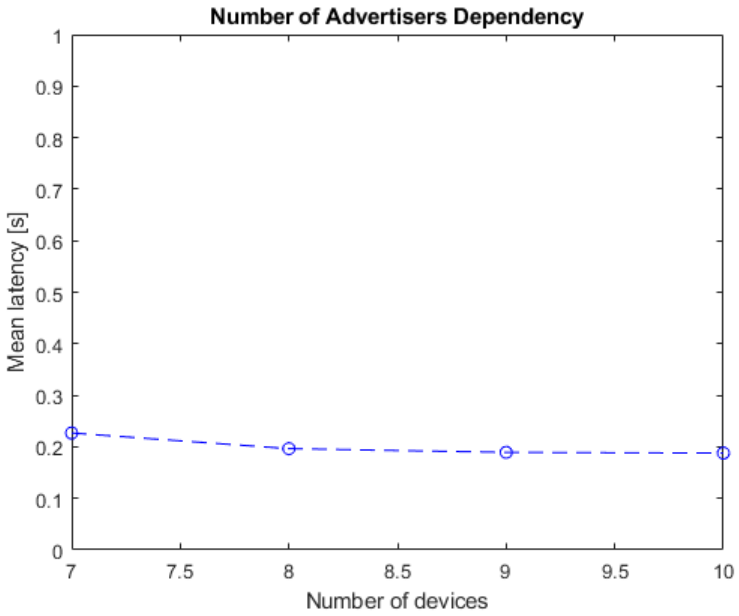
4.1.2 Počet inzerujících zařízení

Počet inzerujících zařízení pro test je vyhodnocen manuálně pomocí nRF51 posluchače a aplikace Wireshark. Zjištěná hodnota je poté zadána jako parametr testu v MATLABu. Výsledky testu jsou zobrazeny na obrázku (4.4) a (4.3). Měření bylo

number of devices	mean values	margin of error
7	0.2268	0.0402
8	0.1964	0.0319
9	0.1892	0.0267
10	0.1878	0.0284

Obrázek 4.3: Statistika pro počet zařízení

provedeno se stem opakování, inzerujícím intervalem 320ms a nekonečným skenovacím oknem. Ze statistik lze pozorovat, že počet inzerujících zařízení nemá žádný vliv na výslednou latenci. Test by bylo vhodné opakovat v prostředí s desítkami komunikujících zařízení. Během epidemie koronaviru bohužel nebyl přístup do nákupních



Obrázek 4.4: Závislost latence na počtu inzerujících zařízení

center. V případě zájmu firmy Jablotron může být později, test s velkým množstvím zařízení proveden.

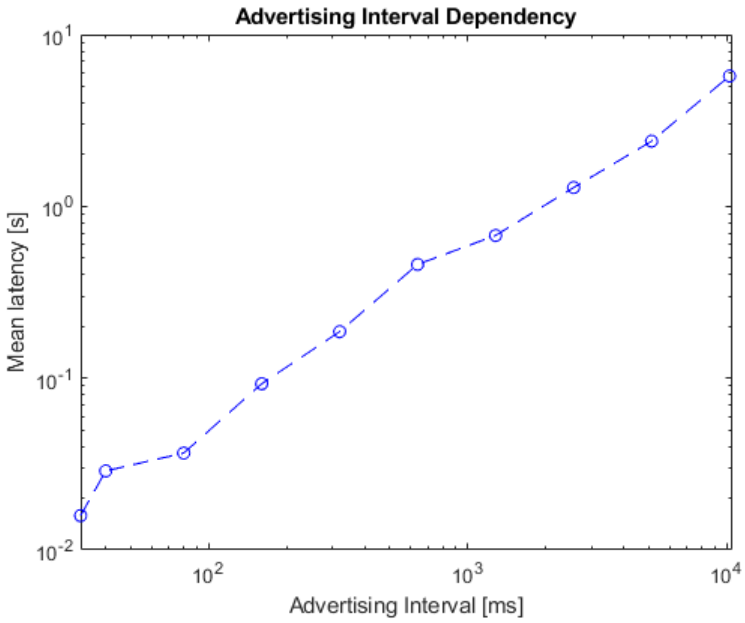
4.1.3 Inzerující interval

Inzerující interval určuje frekvenci, s jakou periferie vysílá datové pakety do okolí. V reálné situaci vstoupí uživatel do rozsahu skenujícího zařízení v náhodný čas během inzerujícího intervalu. Tento fakt je simulován pomocí časovače, který čeká po náhodnou dobu, která spadá do délky intervalu. Výsledky měření jsou na obrázku (4.6) a (4.5). Měření bylo provedeno s padesáti opakováními a nekonečným

advertising interval	mean values	margin of error
20ms	0.0158	0.0022
40ms	0.0288	0.0044
80ms	0.0366	0.0071
160ms	0.0928	0.0128
320ms	0.1865	0.0359
640ms	0.4592	0.0616
1280ms	0.6769	0.1373
2560ms	1.2865	0.2067
5120ms	2.3959	0.5230
10240ms	5.7368	0.9228

Obrázek 4.5: Statistika pro izerující interval

skenovacím oknem. Parametry pro inzerující interval byly zvoleny ve větší hustotě na začátku intervalu, proto je pro graf použitá logaritmická stupnice. Z výsledků je



Obrázek 4.6: Závislost latence na inzerujícím intervalu

zřejmé, že inzerující interval má na latenci detekce zásadní vliv. Je proto nutné ho zvolit tak, aby splňoval časové požadavky aplikace.

4.1.4 Skenující okno

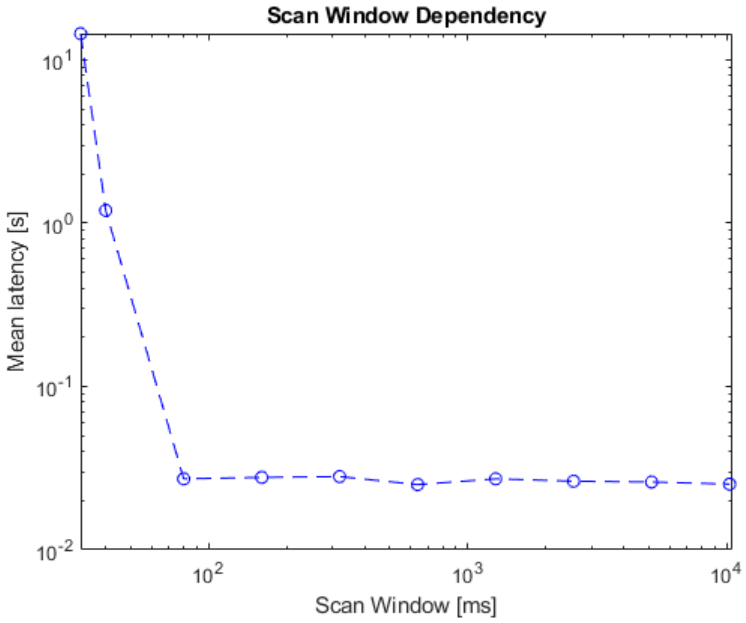
Skenující okno je část skenovací intervalu, kdy centrální zařízení přijímá zprávy na konkrétním kanálu. (1.3.5) Pro měření je důležitý inzerující interval, který byl pevně nastaven na 40ms a skenovací interval byl nastaven na 10240ms. Pokud je skenovací okno menší než inzerující interval, může se doba detekce extrémně protáhnout, protože existuje pravděpodobnost, že se zařízení během jednoho skenovací intervalu nepotkají na žádném kanálu. To je patrné z obrázků (4.8) a (4.7). Vždy je potřeba

scanning window	mean values	margin of error
20ms	14.4508	4.9081
40ms	1.1955	1.0576
80ms	0.0272	0.0042
160ms	0.0277	0.0039
320ms	0.0281	0.0042
640ms	0.0251	0.0038
1280ms	0.0272	0.0033
2560ms	0.0263	0.0040
5120ms	0.0260	0.0041
10240ms	0.0252	0.0037

Obrázek 4.7: Statistika pro skenující okno

volit skenovací okno větší než inzerující interval, pokud je okno jenom část skeno-

vacího intervalu. Většina centrálních zařízení má standardně vyšší kapacitu energie, a proto není problém nastavit neustálé skenování.



Obrázek 4.8: Závislost latence na skenujícím oknu

4.2 Možné rozšíření

Všechny parametry, kterými se dá ovlivnit latence detekce, mají vliv na spotřebu energie. Proto je nastavování těchto parametrů kompromisem mezi spotřebou a rychlostí detekce. Testovací aplikace by se dala rozšířit o testování spotřeby. Společnost Nordic Semiconductor poskytuje k vývojovým deskám specializovaný hardware power profiler kit (PKK)[13], který se dá k deskám připojit. PKK má možnost měřit špičkový a průměrný proud. Aplikace by tak pro vybraný parametr mohla zobrazovat i průměrnou spotřebu.

5 Bezpečnost BLE

Důležitost bezpečnosti se pro každou aplikaci liší. Pro autentifikaci uživatele na odjištění zabezpečovacího systému je ošem potřeba maximální bezpečnost. Následující kapitola si klade za cíl rozebrat možné bezpečnostní hrozby a popsat mechanismy, kterými BLE dané hrozby může řešit. Do roku 2014 využívalo BLE nepříliš robustní LE Legacy protokol, se specifikací Bluetooth 4.2 ovšem přišlo značné zlepšení v podobě protokolu LE Secure Connection (LESC). Dnes již většina zařízení obsahuje verzi Bluetooth 4.2 a vyšší, proto je detailně rozebrán pouze protokol LESC.

5.1 Bezpečnostní protokol

Dva důležité pojmy, které se týkají bezpečnosti, jsou párování a bonding. Párování může proběhnout až po zformování připojení a jedná se o výměnu nezbytných informací pro zřízení šifrovaného spojení. Bonding znamená trvalé uložení těchto informací do paměti, aby se párování nemuselo při dalším připojení opakovat. Nejprve jsou uvedeny obecné kryptografické metody, které párování využívá, poté je popsán samotný proces. Po proběhnutí párování je prolomení bezpečnosti velice obtížné. Největší riziko hrozí během samotného procesu.

5.1.1 Diffieho–Hellmanova výměna klíčů

Ve skutečnosti využívá BLE Diffieho–Hellmanův protokol s využitím eliptických křivek (ECDH), pro zachování srozumitelnosti je popsána pouze jeho základní verze s názvem Diffieho-Hellmanova výměna klíčů (DH). DH je metoda, která umožňuje bezpečnou výměnu symetrického klíče mezi dvěma zařízeními přes veřejný kanál. Klíč se nazývá symetrický, protože se využívá pro šifrování i dešifrování dat. Definice a název protokolu jsou lehce zavádějící, jelikož nejde o přímou výměnu klíče. V případě přímé výměny by mohl klíč kdokoliv zachytit a dešifrovat pozdější komunikaci. Zařízení si proto vymění určité typy veřejných proměnných, které jsou využity pro výpočet stejného symetrického klíče na obou stranách. Pro výpočet symetrického klíče jsou využity vlastnosti modulární aritmetiky.

Následuje popis principu výměny klíče mezi dvěma zařízeními. Na obrázku je zachycen počáteční stav potřebných parametrů. (5.1) Zařízení jsou podle kryptografické konvence pojmenována Alice a Bob.

V první fázi se určí čtyři hodnoty potřebné pro výpočet symetrického klíče. Za číslo n se volí prvočíslo, které bývá 4000 bitů velké. Číslo g je primitivní kořen n.

Alice	Public	Bob
private variable a	generator g	private variable b
	prime number n	

Obrázek 5.1: Počáteční stav DH výměny

([14]) Poté si Alice a Bob náhodně zvolí privátní hodnoty a a b, které jsou z intervalu $[1, n]$.

V druhé fázi spočtou Alice a Bob veřejné klíče p_a a p_b, které si poté vymění přes veřejný prostor.

$$p_a = g^a \ mod \ n \tag{5.1}$$
$$p_b = g^b \ mod \ n \tag{5.2}$$

Teoreticky je možné spočítat číslo a z rovnice 5.1. Ovšem jediná známá cesta je výpočet hrubou silou, což znamená dosazování za číslo a a kontrola zda výsledek odpovídá číslu p_a. Jelikož je hodnota čísla a obrovská, čas, který by byl potřeba pro získání hodnoty a běžným počítačem, je v řádu stovek tisíc let.

Ve třetí fázi spočítají Alice a Bob symetrický klíč s pomocí následujícího výpočtu.

$$s_a = (p_b)^a \ mod \ n \tag{5.3}$$
$$s_a = (g^b \ mod \ n)^a \ mod \ n \tag{5.4}$$
$$s_b = (p_a)^b \ mod \ n \tag{5.5}$$
$$s_b = (g^a \ mod \ n)^b \ mod \ n \tag{5.6}$$
$$s_a = s_b \tag{5.7}$$

Z rovnic (5.4) a (5.6) vyplývá (5.7), to znamená že Alice i Bob mají k dispozici ten samý symetrický klíč, který je poté využit na výpočet klíče pro šifrování dat.

5.1.2 Standard pokročilého šifrování

Standard pokročilého šifrování (AES) je způsob, pomocí kterého probíhá veškeré šifrování dat. AES se zrodil ze soutěže o nejlepší šifrovací algoritmus v roce 1997. [15] Jedná se o mezinárodní standart, který je používán pro šifrování téměř každého internetového spojení. Funkčnost algoritmu byla od doby jeho vzniku nespočetněkrát testována.

Jednotka, s kterou algoritmus pracuje, je 128 bitový blok, uspořádaný do matice 4x4. Matice projde substitučně-permutační sítí, kde jsou jednotlivé operace definovány v rámci konečného tělesa. To znamená, že každá operace je definována tak, aby se její výsledek vždy nacházel v konečné množině čísel. Substituce jednotlivých bajtů je prováděna pomocí vyhledávacích tabulek a permutace je implementována

jako maticové násobení. Pro zajištění bezpečnosti šifry se po každém substitučně-permutačním bloku provádí exkluzivní disjunkce mezi výstupem a částí unikátního klíče. Tento klíč zařízení získají pomocí DH metody. (5.1.1) Dešifrování se provádí inverzí všech operací v opačném pořadí.

Jelikož se AES používá téměř všude, má značná část procesorů speciální instrukce pro jeho výpočet. Rychlost šifrování se tak pohybuje v Gbit/s.

5.1.3 Párování

Párování probíhá ve třech fázích. V první fázi si zařízení pošlou zprávu o svých hardwarových možnostech, které hrají důležitou roli při párovacím procesu, a podle předaných informací se domluví na způsobu zformování bezpečného připojení. Data nejsou v této fázi šifrována.

V druhé fázi proběhne výměna symetrických klíčů pomocí DH metody. (5.1.1) Výměna klíčů může probíhat pomocí čtyř různě bezpečných způsobů, které závisí na hardwarových možnostech obou zařízení. Nejméně bezpečný je způsob just works. Zařízení si vymění klíče a jejich prostřednictvím poté vygenerují kontrolní hodnoty, které si vzájemně pošlou a pokud se shodují, navážou spojení. Druhý způsob numerické srovnání je závislý na kooperaci uživatele a vyžaduje přítomnost displeje. Zařízení nezávisle spočtou šestimístné číslo pomocí vyměněného klíče a zobrazí ho uživateli. Ten musí poté vyhodnotit, zda se čísla shodují. Třetí způsob se nazývá párování pomocí přístupového klíče a vyžaduje klávesnici u jednoho ze zařízení. Způsob využívá identické šestimístné číslo. Toto číslo může být zadáno do obou zařízení nebo jedno zařízení číslo vygeneruje a uživatel ho poté zadá do druhého zařízení. Tato metoda vyžaduje aktivní kooperaci uživatele a dá se proto považovat za bezpečnější. Nejspolehlivější metodou je out of band (OOB) párování. Klíče jsou vyměněny pomocí jiného zabezpečeného kanálu. Nejčastěji se pro tento účel využívá technologie near field communication (NFC). NFC funguje, pouze pokud jsou zařízení přiblížena na velmi krátkou vzdálenost. Nehrozí proto narušení komunikace třetí stranou.

Třetí fáze je nepovinná a probíhá již se zabezpečenou komunikací. Zařízení si vymění informace pro automatické zabezpečení každé následující komunikace. Jedná se například o klíč pro identifikaci identity, který je používám na dekódování privátní adresy nebo klíč podpisu dat. Tento klíč potvrdí, že data pocházejí od důvěryhodného zdroje.

5.2 Možné typy útoků

Všechny uvedené útoky se týkají doby, kdy zařízení nemají zformované šifrované připojení nebo se pokouší o jeho vytvoření párovacím procesem.

5.2.1 Man-in-the-middle

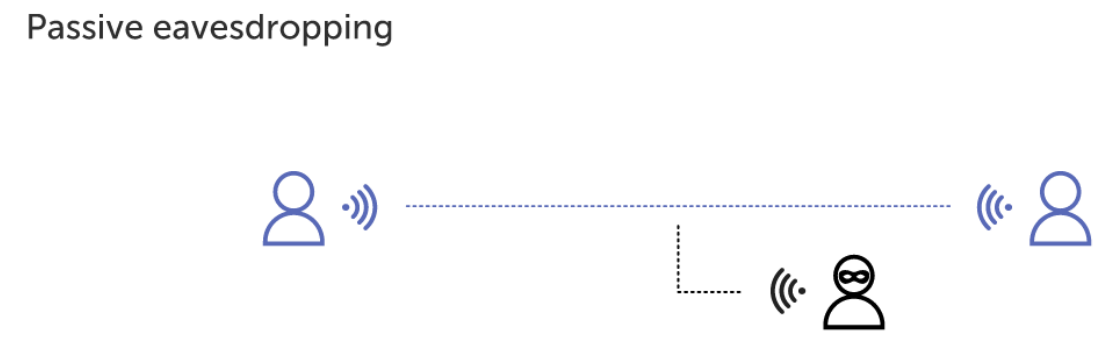
Při útoku man-in-the-middle se třetí škodlivé zařízení vydává za obě komunikující strany a vytvoří si tak nechtěné spojení s oběma zařízeními. Tento útok může být nejškodlivější, jelikož útočník může zachytit veškerá procházející data a vložit uměle vytvořená data a příkazy přímo do komunikace. Diagram MITM je na obrázku. (5.2) MITM útoku se dá předejít pouze pomocí vhodné metody během párovacího

Obrázek 5.2: Man-in-the-middle útok (převzato z [16])

procesu. Výměnu klíčů během DH (5.1.1) dokáže škodlivé zařízení zachytit a provede vlastní DH výměnu s oběma zařízeními. To ovšem znamená, že původní dvě zařízení nebudou mít stejný symetrický klíč. Tento fakt můžeme objevit pomocí metody numerického srovnání (5.1.3), kde se kontrolní vygenerovaná čísla budou lišit. V případě párovaní pomocí numerického klíče se bude kontrolní výpočet na obou zařízení lišit a k úspěšnému spárování tak vůbec nedojde. Nejbezpečnější řešení je párování pomocí NFC, kdy útočník nemá vůbec šanci výměnu klíčů zachytit.

5.2.2 Pasivní odposlech

Při pasivním odposlechu poslouchá škodlivé zařízení probíhající komunikaci a má možnost zachytit citlivá data. Schéma odposlechu je zobrazeno na obrázku. 5.3 Ten-



Obrázek 5.3: Pasivní odposlech (převzato z [16])

to problém odpadne po úspěšném párovacím procesu, když se začnou zprávy šifrovat. Průběh výměny klíčů a párování byl odposloucháván pomocí nRF51 dongle a aplikace Wireshark. Na obrázku (5.4) je možné vidět průběh komunikace. Poté, co proběhne párování, již Wireshark není schopný určit obsah paketů. (obr. 5.5) Díky

vlastnostem DH výměny není schopný útočník z odposlechnutých informací určit klíč, který je použit pro šifrování.

```
Sent Pairing Request: AuthReq: Bonding, SecureConnection | Initiator Key(s): LTK, IRK
Rcvd Pairing Response: AuthReq: Bonding, MITM, SecureConnection | Initiator Key(s): LTK, IRK
Sent Pairing Random
Rcvd Pairing Random
Sent Pairing DHKey Check
Rcvd Pairing DHKey Check
```

Obrázek 5.4: Pasivní odposlech DH výměny

```
Master_0x31e6e117    Encrypted packet decrypted incorrectly (bad MIC)
Slave_0x31e6e117     Encrypted packet decrypted incorrectly (bad MIC)
```

Obrázek 5.5: Pasivní odposlech šifrované komunikace

5.2.3 Sledování identity

Při sledování identity spojí útočník adresu specifické BLE zařízení s konkrétním člověkem a poté je schopný trasovat jeho pohyb. Tomuto problému dokáže BLE zabránit použitím privátní adresy, která se periodicky mění a jen pokud má jiné zařízení klíč pro identifikaci identity, dokáže tuto adresu přiřadit ke konkrétnímu zařízení.

Závěr

Cílem této práce bylo seznámit se s BLE protokolem, otestovat parametry ovlivňující latenci detekce a analyzovat bezpečnost. Vše bylo prováděno s ohledem na možné využití BLE v aplikaci automatického zajištění a odjištění objektů firmou Jablotron.

Výsledkem práce je aplikace na testování parametrů a podrobná analýza možných bezpečnostních rizik, které BLE hrozí. Dle provedených měření a analýzy lze usoudit, že BLE protokol je pro danou aplikaci vhodný, závisí pouze na jeho vhodné implementaci.

Parametry, které ovlivňují latenci detekce, musí být voleny tak, aby splňovaly časové požadavky a zároveň nezpůsobovaly zbytečnou spotřebu energie. Nejdůležitějším parametrem je v tomto ohledu inzerující interval, jelikož je periferie běžně napájena pomocí baterií.

V oblasti bezpečnosti je potřeba implementovat vhodnou párovací metodu na výměnu klíčů, aby se předešlo možným hackerským útokům. Nejbezpečnější metodou je použít párování pomocí NFC. V případě, že většina zařízení nebude NFC umožňovat, je druhá nejbezpečnější metoda párování pomocí přístupového klíče, jelikož vyžaduje aktivní účast uživatele. Pokud z hardwarových důvodu nebude ani jedna metoda možná, neví vhodné BLE protokol využít.

Použitá literatura

[1] AFANEH, Mohammad. *Bluetooth 5 & Bluetooth Low Energy: A Developer's Guide* [online]. 2nd ed. Novel Bits, 2019 [cit. 19. 03. 2020]. Dostupné z: https://www.novelbits.io/bluetooth-5-developers-e-book/.

[2] POTHITOS, Adam. *The History of Bluetooth* [online] [cit. 03. 04. 2020]. Dostupné z: http://www.mobileindustryreview.com/2017/08/the-history-of-bluetooth.html.

[3] *Bluetooth Core Specification* [online]. 2019 [cit. 03. 04. 2020]. Č. v5.2. Dostupné z: https://www.bluetooth.com/specifications/bluetooth-core-specification/.

[4] *Gaussian Frequency Shift Keying (GFSK)* [online] [cit. 03. 04. 2020]. Dostupné z: https://flylib.com/books/en/2.519.1/gaussian_frequency_shift_keying_gfsk_.html.

[5] Supplement to the Bluetooth Core Specification. In: [online], s. 9–18 [cit. 25. 05. 2020]. Dostupné z: https://www.bluetooth.com/specifications/bluetooth-core-specification/.

[6] *ALERT NOTIFICATION PROFILE* [online]. 2011 [cit. 11. 04. 2020]. Dostupné z: https://www.bluetooth.com/specifications/gatt/.

[7] *Qt Bluetooth* [online]. 2020 [cit. 11. 04. 2020]. Dostupné z: https://doc.qt.io/qt-5/qtbluetooth-index.html.

[8] *Windows 10 - Pairing a BLE device from code* [online] [cit. 25. 05. 2020]. Dostupné z: https://social.msdn.microsoft.com/Forums/de-DE/e321cb3c-462a-4b16-b7e4-febdb3d0c7d6/windows-10-pairing-a-ble-device-from-code?forum=wdk.

[9] *NRF52840 Product Specification* [online]. 2018 [cit. 12. 04. 2020]. Dostupné z: https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.0.pdf.

[10] RUBR. *PCA10056$_S$chematic$_A$nd$_P$CB* [online]. 2018 [cit. 12. 04. 2020]. Dostupné z: https://devzone.nordicsemi.com/cfs-file/___key/support-attachments/beef5d1b77644c448dabff31668f3a47 - b7b5701e8f7d44e3b82edb037eb59b9f / pca10056_5F00_schematic_5F00_and_5F00_pcb.pdf.

[11] *Nordic Semiconductor Inforcenter* [online] [cit. 12. 04. 2020]. Dostupné z: https://infocenter.nordicsemi.com/index.jsp.

[12] *NRF5 SDK* [online] [cit. 21. 04. 2020]. Dostupné z: https://www.nordicsemi.com/Software-and-Tools/Software/nRF5-SDK/Download%5C#infotabs.

[13] *Power Profiler Kit* [online] [cit. 01. 06. 2020]. Dostupné z: https://www.nordicsemi.com/Software-and-tools/Development-Kits/Power-Profiler-Kit.

[14] *Primitive Roots* [online] [cit. 29. 05. 2020]. Dostupné z: https://brilliant.org/wiki/primitive-roots/.

[15] *Advanced Encryption Standard* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 30. 05. 2020]. Dostupné z: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

[16] *How Secure Is the BLE Communication Standard?* [online]. 2019 [cit. 30. 05. 2020]. Dostupné z: https://lembergsolutions.com/blog/how-secure-ble-communication-standard%5C#ChoosingtherightBLEpairingmethod.

[17] CHO, Keuchul; PARK, Woojin; HONG, Moonki; PARK, Gisu; CHO, Wooseong; SEO, Jihoon; HAN, Kijun. Analysis of Latency Performance of Bluetooth Low Energy (BLE) Networks [online] [cit. 25. 05. 2020]. Dostupné z DOI: 10.3390/s150100059.

Obsah přiložené flash paměti

- Text bakalářské práce
  - TampierBP2020.pdf
- Vytvořený software
  - software.zip