Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Vývoj Bluetooth Low Energy zařízení na bázi chipů NRF5x

Bakalářská práce

Autor: Stanislav Mádr
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Pavel Kříž, Ph.D.

Hradec Králové srpen 2019

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 16.8.2019 Stanislav Mádr

Poděkování:

Děkuji vedoucímu Ing. Pavlu Křížovi, Ph.D. za metodické vedení práce a pomoc při pájení a následném připojení čipu nRF52840 k vývojovému kitu.

Anotace

Bakalářská práce se věnuje seznámení s vývojovými kity od společnosti Nordic Semiconductor s čipy nRF51822 a nRF52840, a jejich následnému použití při vývoji aplikací využívajících technologii Bluetooth Low Energy (BLE). V práci se dále nachází seznámení s technologiemi Bluetooth Low Energy a Bluetooth Mesh a jejich implementace pomocí nRF5 Software Development Kitu (SDK), který poskytuje výrobce daných čipů. Technologie Bluetooth Mesh je blíže ukázána na příkladu nacházejícím se ve složce SDK.

Praktická část této bakalářské práce se zabývá vývojem aplikací s těmito případy užití. Prvním případem je měření dosahu čipů. Využity jsou 2 vývojové kity, kdy se při přerušení spojení rozsvítí LED dioda. Druhým případem je skenování okolních zařízení a vypsání jejich MAC adres a RSSI do BLE charakteristik.

Klíčová slova: Nordic Semiconductor, nRF51822, nRF52840, Bluetooth Low Energy, Bluetooth Mesh

Annotation

Title: Development of Bluetooth Low Energy Device Using NRF5x Chip

This bachelor thesis acquaints the reader with the development kits from the company Nordic Semiconductor with chips nRF51822 and nRF52840, and their subsequent use in the development of applications utilizing Bluetooth Low Energy (BLE) technology. Furthermore, the thesis introduces the Bluetooth Low Energy and Bluetooth Mesh technologies and their implementation using the Software Development Kit that is provided by the chip manufacturer. Bluetooth Mesh technology is shown in more detail in the example located in the SDK folder.

The practical part of this thesis is focused on the applications development using the aforementioned examples. The first case measures the range of chips - there are two development kits used. The LED diode lights up when the connection is interrupted. The second case is scanning the surrounding devices and listing their MAC addresses and RSSIs into BLE characteristics.

Keywords: Nordic Semiconductor, nRF51822, nRF52840, Bluetooth Low Energy, Bluetooth Mesh

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V současné době se rozmáhá vliv chytrých zařízení na každodenní úkony člověka. Ať už chceme znát aktuální čas, dopravní situaci, předpověď počasí, anebo si poslechnout naši oblíbenou hudbu, využijeme aplikaci v mobilním telefonu, či v jiném zařízení, ve které nalezneme vše, co potřebujeme.

Chytrá zařízení neslouží pouze pro zobrazení dat, ale také pro jejich sběr. Například sportovec si při své aktivitě nasadí chytré hodinky, které kromě standartního zobrazení času a data snímají pomocí přídavných senzorů jeho tep, aktuální polohu, nadmořskou výšku, či rychlost, kterou se právě pohybuje.

Zde však nastává problém, jak daná data shromažďovat na jednom místě, nebo jak zajistit komunikaci mezi těmito zařízeními. V minulosti veškerá komunikace probíhala přes datový kabel, to však bylo velice nepraktické. Z tohoto důvodu se v současné době rozmáhá bezdrátová komunikace a mezi nejpoužívanější standardy této komunikace patří technologie Bluetooth.

S růstem počtu zařízení s podporou bezdrátové komunikace samozřejmě vznikají i firmy, které se snaží vyvíjet a distribuovat čipy pro tuto komunikaci. Jednou z těchto firem je Nordic Semiconductor, jež se zabývá produkcí čipů s velmi nízkou spotřebou energie.

2 Cíl práce

Cílem této bakalářské práce je analyzovat možnosti vývoje Bluetooth Low Energy aplikace na čipech nRF51822 a nRF52840 od společnosti Nordic Semiconductor za pomoci jejich Software Development Kitu a seznámení s technologií Bluetooth Mesh. Pro tento cíl byly vybrány 2 typy zařízení nRF51822 Development Kit a nRF52840 Development Kit, obsahující právě tyto čipy.

Pro demonstrační ukázku byly zvoleny tyto případy užití. Prvním případem je měření dosahu čipů. Využity jsou 2 vývojové kity, které jsou k sobě připojeny. Ztráta připojení je reprezentována rozsvícením LED diody. Druhým případem je vytvoření zařízení, které po stisknutí tlačítka naskenuje okolní zařízení. Skenovaní je prováděno v určitém časovém intervalu a po jeho ukončení se do BLE charakteristik uloží MAC adresa a poslední naměřené RSSI prvních 10 různých zařízení.

3 Technologie Bluetooth

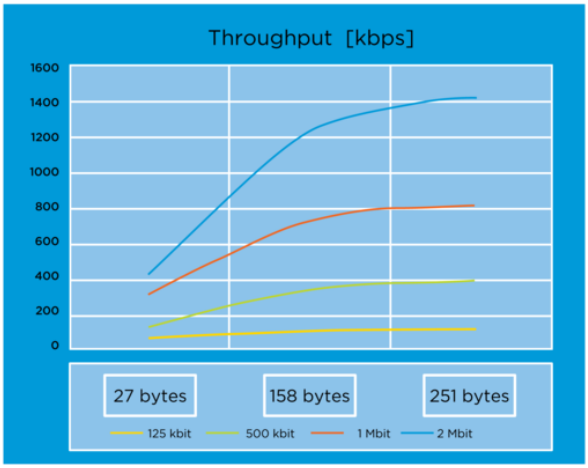
Bluetooth je standard pro bezdrátovou komunikaci na krátkou vzdálenost. Slouží pro propojení dvou a více zařízení a pracuje na frekvenci 2,4 GHz.
Tento standard získal své jméno po králi Heraldu Modrozubovi (Herald Blåtand), který byl takto přezdíván díky jeho oblibě v jezení borůvek. Logo dále představuje složení dvou run, které reprezentují iniciály tohoto krále. [1] [2]

3.1 Historie

V srpnu roku 1942 byl vytvořen patent na technologii Frequency Hopping Spread Spectrum (FHSS) pod názvem „Secret Communication System“, na kterém posléze Bluetooth založilo svůj komunikační protokol. Tato technologie sloužila pro radiově naváděná torpéda, a to z důvodu, že funguje na principu přeskakování mezi několika kanály, což znemožňovalo nepříteli rušit navádějící signál. [3]
Kořeny Bluetooth sahají do roku 1994, kdy se švédská telekomunikační společnost Ericsson rozhodla nahradit komunikační rozhraní RS-232 technologií založenou na rádiovém vysílání. Avšak tento nápad dostaly i ostatní společnosti jako Intel, či Nokie. Tyto společnosti se proto rozhodly, že daná technologie by měla být standardizována a řízena jedním sdružením. Sdružení Special Interest Group (SIG) bylo založeno v roce 1998 a první verze Bluetooth vyšla o rok později. [3]
V dnešní době nalezneme tento standard ve velkém množství aplikací, kde slouží, ať už jako bezdrátové propojení mezi mobilním telefonem a reproduktorem, anebo jako prostředník mezi zařízeními v hale automatizované výroby. [3]

3.2 Bluetooth 5

Bluetooth 5 je nejnovější verzi tohoto standardu, která přináší mnoho nových zdokonalení.
Mezi důležitá zdokonalení patří zvýšení přenosové rychlosti, která se od předchozí verze zdvojnásobila a Bluetooth nyní zvládá přenést až 2Mb/s. Z tohoto důvodu se zvyšuje rychlost konektivity mezi zařízeními, ale také i optimalizace odezvy. [4]



Obr. 1 Bluetooth 5 Propustnosti. Zdroj: [5]

Dalším zdokonalením je dosah, který se zvýšil na čtyřnásobek oproti minulé verzi, která dosahovala maximálně 50 metrů ve volném prostranství a 10 metrů v uzavřené místnosti. Dosah je přímo závislý na datových rychlostech, mezi nimiž je možné volit (2 Mb/s, 1Mb/s, 500 kb/s a 125 kb/s). Čím menší je přenosová rychlost tím delší je dosah. [4]

Nakonec je nutno poznamenat, že zařízení se standardem Bluetooth 5 nemusí splňovat vlastnosti uvedené v předchozím odstavci. Například většina zařízení nepodporuje nižší rychlosti přenosu.

3.3 Bluetooth Low Energy

Bluetooth Low Energy je technologie, která zajišťuje operace, které spotřebují jen velice málo energie. Využívá se například v zařízeních, které vyžadují dlouhou výdrž baterie. Mezi taková zařízení patří chytré hodinky, sportovní a fitness senzory, či dálková ovládaní. [6]

Úspora energie je zapříčiněna tím, že se výrazně snížila doba mezi „probuzením“ zařízení, komunikací a následném přechodu zařízení do úsporného režimu. Oproti klasickému Bluetooth je až 20krát nižší. Dále je výrazně snížena rychlost přenosu dat. [6]

3.4 Bluetooth Mesh

S nástupem Bluetooth 5 dostala tato technologie podporu protokolu mesh networking založeného právě na Bluetooth Low Energy. S tímto protokolem je možná M:N komunikace, která je zajištěna pomocí techniky řízené záplavy. Nyní se tedy mohou stavět topologie o stovkách až o tisíci zařízeních určených například pro automatizaci budov či výrobních linek. Mezi hlavní výhody technologie Bluetooth Mesh patří rozšířený dosah, kdy jednotlivé uzly prostřednictvím ostatních mohou přenášet data na vzdálené zařízení. Další výhodou je to, že pokud selže jeden z typů uzlů, na kterým nezávisejí ostatní, tak zbývající uzly mohou mezi sebou stále komunikovat. Tento protokol používá publish/subscribe systém zpráv.[7]

- Publish: Zařízení vyšlou zprávu na příslušnou adresu, kterou může mít jedno nebo více zařízení. [7]
- Subscribe: Zařízení přijme tuto zprávu, zkopíruje ji a pošle dál. Pokud je zpráva určena pro něj (má příslušnou adresu), tak na ni zareaguje. [7]

Jednou z klíčových vlastností mesh networkingu je „State“. Každé zařízení totiž reprezentuje svůj stav pomocí proměnných. Vysláním zprávy na daná zařízení vlastně měníme jejich proměnnou například z ON na OFF. [7]
Jednotlivé koncepty budou blíže vysvětleny v následujících podkapitolách.

3.4.1 Koncepty Bluetooth Mesh sítě

V následující kapitole budou popsány důležité koncepty Bluetooth Mesh sítě.

3.4.1.1 Nodes (Uzly)

Prvním konceptem Bluetooth Mesh je uzel. Uzlem je zařízení připojené do sítě. Zařízení, které není připojeno do této sítě se nazývá Unprovisioned device. Pokud se však z něho stane Provisioned device, připojí se do sítě jako uzel. Jednotlivé uzly mohou být rozděleny na několik jednotlivých částí označované jako elementy.

Elementy si můžeme představit jako jednotlivé žárovky v lampě, které lze jednotlivě ovládat. [8]

3.4.1.1.1 Typy uzlů

Kromě posílání a přijímaní zpráv mohou mít uzly volitelné vlastnosti, které mohou být kdykoliv aktivovány, nebo deaktivovány. V závislosti na těchto vlastnostech se uzly dělí na Relay node, Proxy node, Friend node a Low Power node. Relay node umožňuje přeposílat zprávy, které jsou vysílány jiným uzlem. Díky tomu je možné prodloužit dosah zpráv i pro ty uzly, které nejsou v dosahu původce zprávy. Pro umožnění komunikace s BLE (Bluetooth Low Energy) zařízením, které nepodporuje Mesh se využívá Proxy node. Proxy node slouží jako prostředník a využívá GATT operace pro interakci s připojeným zařízením a mesh sítí. K tomuto účelu je určen proxy protocol, který je postaven nad GATT. Připojené zařízení čte a zapisuje PDU proxy protokolu z GATT charakteristik implementovaných daným uzlem, které je posléze transformuje na mesh PDU používané v mesh síti. Friend node a Low Power node (LPN) jsou spolu blízce spojeni. Friend nod ukládá do mezipaměti zprávy určené pro přiřazené LPN, které zůstává v režimu spánku a nepřijímá žádné zprávy. Po určitém čase se LPN probudí a od Friend nodu si přečte veškeré zprávy a následně je zpracuje. [9]

3.4.1.2 States

Druhým konceptem jsou States (Stavy). Jednotlivé elementy mohou nabývat různých stavů, které mění jeho chování. Pokud se stav daného elementu změní, nazývá se to state transition. Tato změna může být okamžitá, či se objevit v průběhu času. V některých případech jsou stavy mezi sebou svázány, což znamená, že například změněním jednoho stavu, změníme i ostatní stavy elementů v uzlu. Určitý kontext hodnoty stavu určují Properties (Vlastnosti), které lze rozdělit na vlastnosti určené výrobcem (umožněno pouze čtení) a správcem (čtení i zápis). [8]

3.4.1.3 Messages

Třetím konceptem jsou Messages (Zprávy). Veškerá komunikace v Bluetooth Mesh síti je message-oriented. Uzly používají zprávy, aby se navzájem kontrolovaly, či vyměňovali informace. Existují tři druhy zpráv: SET Message, GET Message a STATUS Message. Jsou definované jedinečným operačním kódem. GET Message je zprávou vyžadující stav od jednoho nebo více uzlů. SET Message slouží pro nastavení hodnoty daného stavu. Posledním typem je STATUS Message, která je využita v různých případech, a to buď jako odpověď s hodnotou stavu na GET Message, nebo jako potvrzení SET Message. Může sloužit i jako nezávislá zpráva ohlašující stav některého z elementů. [8]

3.4.1.4 Adresy

Čtvrtým konceptem jsou Adresy. Určují původce a cílové zařízení zprávy. Existují tři typy adres. Prvním typem je Unicast Address identifikující jednotlivý uzel. Druhým typem Group Address, která slouží k identifikaci skupiny uzlů a většinou reprezentuje jejich fyzické uspořádání (Veškeré uzly v jedné místnosti). Tato adresa je buď definována Bluetooth SIG pod označením SIG-Fixed Group Address, anebo uživatelem pomocí konfigurační aplikace (Dynamic Group Address). Posledním typem je Virtual Address identifikující jeden, nebo několik elementů jednoho, či více uzlů. Adresa je ve formě 128-bitového UUID. [8]

3.4.1.5 Publish / Subscribe

Pátým konceptem je vzor Publish / Subscribe. Publishing je aktem posílání zpráv a Subscribing je konfigurací, která povoluje vybraným zprávám (s cílovou Virtual nebo Group Address) přístup na specifickou adresu ke zpracování. Jednotlivý uzel může mít Subscribe na více adres. Například světlo v šatníku, které může reagovat jak na group adresu šatníku, tak i na group adresu pokoje. [8]

3.4.1.6 Managed Flooding

Šestým konceptem je Managed Flooding, kterou Bluetooth Mesh používá k přenosu zpráv. Jedná se o jakýsi kompromis mezi routovacím mechanismem a zaplavováním sítě bez ohledu na optimální trasy. Technika funguje na principu všesměrového

vysílání zpráv do všech uzlů v dosahu vysílače. Tyto zprávy jsou však posílány s několik optimalizacemi. [8]

První optimalizací je TTL (Time to Live), která limituje počet hopů přes uzly v dané síti. Například, když je hodnota nastavena na 0, zpráva může být poslána pouze do uzlu v dosahu vysílače. [8]

Druhou optimalizací je ukládání zpráv do mezipaměti ve všech uzlech. Pokud je zpráva uložena v mezipaměti a přijde znovu na uzel, bude zahozena. [8]

Třetí optimalizací jsou Heartbeat messages, které jsou rozesílány ostatním uzlům v síti, aby indikovali, že odesílatel je stále aktivní. [8]

Poslední optimalizací je Friendship týkající se vztahu mezi dvěma uzly. Tyto uzly mohou být typu Low Power (LPN), či Friend. [8]

3.4.1.7 Modely

Modely jsou jedním z nejdůležitějších prvků. Určují část, či celou funkcionalitu prvku. V Bluetooth Mesh se setkáme se třemi kategoriemi modelů, které se dají dále rozšiřovat. První kategorie je Server Model. Prvek obsahující tento model může přijímat, či odesílat jeho kolekce stavů, stavových vazeb a zpráv. Druhou kategorií je Client model, který nedefinuje žádné stavy, ale spíše zprávy (GET, SET, a STATUS), které jsou odesílané do Server model. Poslední kategorií je Control Model. Tento model zprostředkovává komunikaci mezi Server a Client modelem, ze kterých je sestaven. [9]

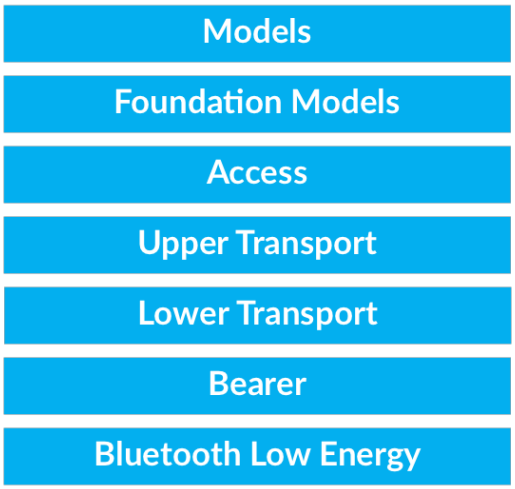
3.4.1.8 Scény

Scény jsou uloženou kolekcí stavů, které jsou identifikovatelné pomocí 16bitového čísla jedinečného v celé síti. Pomocí scén můžeme jednou akcí nastavit stavy na různých uzlech. Například iniciovat rozsvícení světel v obývacím pokoji a zároveň zvýšit teplotu v kuchyni v závislosti na hodnotě venkovního teploměru. [9]

3.4.1.9 Architektura Bluetooth Mesh

Bluetooth Mesh je postaveno nad Bluetooth Low Energy (BLE) a hojně využívá advertising stavu BLE zařízení. Pro vzájemnou komunikaci se zařízení k sobě

nepřipojují, ale využívají stavu scanning a advertising. Zvláštním případem je Uzel typu proxy uvedený v předchozí kapitole č 3.4.1.1.1. [9]



Obr. 2 Architektura Bluetooth Mesh. Zdroj: [9]

Bluetooth Mesh architektura se dělí na tyto vrstvy:

- Bluetooth Low Energy
- Bearer layer
    - Bearer layer definuje, jak se různé PDU zpracovávají. Existují dva typy Bluetooth mesh bearerů. A to Advertising bearer využívající stavů advertising a scanning BLE zařízení, a GATT bearer. GATT bearer využívá stavu connection a umožňuje interakci s mesh sítí pomocí GATT operací. [9]
- Lower transport layer
    - Lower transport layer se zabývá segmentací paketů z Upper transport layer a sestavením paketů z Bearer layer. [9]
- Upper transport layer
    - Upper transport layer se stará o šifrování, dešifrování, autentifikaci a Transport control zprávy (Heartbeat, Friendship, ...). [9]
- Access layer
    - Access layer definuje, jak aplikace využívá Upper transport layer a zpracovává formát dat aplikace, šifrování, dešifrování a ověření dat. [9]

- Foundation Models layer
    - Foundation Models layer se zabývá modely konfigurace a správy sítě. [9]
- Models layer
    - Models layer řeší implementaci modelů, jejich chování, zprávy, stavy a stavové vazby. [9]

3.4.1.10 The Provisioning Process

Provisioning process je jedním z nejdůležitějších konceptů Bluetooth Mesh sítě. Jedná se o proces, kdy se ze zařízení stává uzel. Pro přidání uzlu do sítě slouží zařízení nazývané provisioner a ve většině případech se jedná o smartphone, či osobní počítač. Provisioning proces se skládá z jednotlivých kroků. [10]

Vše začíná krokem Beaconing, kdy unprovisioned zařízení pomocí mesh beacon advertisement oznamuje svoji dostupnost. [10]

Zachycením tohoto oznámení, provisioner zareaguje posláním provisioning invite PDU. Tento krok se nazývá Invitation. Unprovisioned zařízení obdrží dané PDU a odpoví na něj pomocí provisioning capabilities PDU, které obsahuje tyto informace.

- Počet podporovaných elementů.
- Podporované bezpečnostní algoritmy.
- Dostupnost veřejného klíče přes technologii Out-of -Band (OOB).
- Schopnost čtení a zápisu hodnoty uživatelem.

[10]

Třetím krokem je Public Key Exchange. Bezpečnost v mesh síti vyžaduje kombinaci symetrických a nesymetrických klíčů, kterou například zaručuje Elliptic-curve Diffie-Hellman (ECDH) algoritmus. V tomto algoritmu se výměna veřejných klíčů uskutečňuje mezi provisionerem a unprovisioned zařízením. Tato výměna je provedena buď přes BLE (Bluetooth Low Energy), nebo přes kanál out-of-band. [10]

Čtvrtým krokem je Authentication. Autentifikace unprovisioned zařízení obvykle vyžaduje akci uživatele a je závislá na možnostech daného zařízení a provisioneru.

Například při způsobu output OOB, unprovisioned zařízení nějakým způsobem zobrazí číslo (počet bliknutí určité LED), které uživatel posléze zadá do provisioneru. Dalším způsobem může být input OOB, kdy je číslo vygenerované provisionerem a zadáno do unprovisioned zařízení. [10]

Posledním krokem je Provision Data Distribution, který nastává po dokončení autentifikace. Obě zařízení odvodí session key pomocí jejich soukromého a veřejného klíče. Ty následně slouží pro zabezpečení spojení a pro výměnu dalších provisioning dat. Po tomto kroku se ze zařízení stává uzel. [10]

4 Nordic Semiconductor

Nordic Semiconductor je Norská firma zaměřující se na vývoj a distribuci čipů s velmi malou spotřebou energie. Byla založena v roce 1983 a nyní nalezneme její výrobky v produktech jako jsou bezdrátové periferie pro počítače, multimediální ovládače, nebo různé bezdrátové senzory. [11]

Obr. 3 Nordic Semiconductor SoC. Zdroj: [12]

4.1 nRF51822 System on Chip

Multiprotokolový System on chip nRF51822 slouží především pro vývoj Bluetooth Low Energy aplikací, a to hlavně díky jeho nízké spotřebě a schopnosti komunikovat na frekvenci 2,4 GHz. Obsahuje 32-bitový procesor ARM Cortex M.0. a mnoho analogových, či digitálních periférií, které mohou vzájemně komunikovat pomocí PPI (Programmable Peripheral Interconnect) systému. [12]

Bez zásahu CPU dokáže zprostředkovat přímou a autonomní komunikaci mezi periferiemi přes 16-kanálovou sběrnici daného PPI systému. [12]

Zařízení má dále 2 hlavní módy ON/OFF, avšak jednotlivé bloky můžeme přepínat mezi stavy RUN/IDLE na základě našich potřeb.

Tento SoC se hojně využívá pro sportovní senzory a počítačové periferie.[12]

Další specifikace:

- 256 kB/128kB flash, 32kB/16kB RAM.
- Programovatelný výkon výstupu +4dBm až -20dBm.
- Externí krystal s frekvencí 16MHz a poměrně velkou odchylkou +- 40 ppm.

Pro naprogramování daného SoC slouží nRF51 Software Development Kit, ve kterém se nachází Příklady aplikací se zdrojovým kódem, Bluetooth LE a ANT+ profily, Knihovny pro veškeré periferie, Bootloadery a RTOS příklady. Poslední kompatibilní verze s předem vytvořenými šablonami, či projekty se nachází v SDK ve verzi 12.3.0. [12]

4.2 nRF52840 System on Chip

Multiprotokolový System on chip (SoC) nRF52840 je nejvýkonnější v řade nRF52. Podporuje například Bluetooth 5, Bluetooth mesh, Thread, Zigbee anebo ANT. Jeho jádrem je výkonný procesor ARM Cortex-M4F. Díky němu a rozsáhlé paměti FLASH i RAM (1 MB/256 kB) nabízí kompletní řešení pro single chip aplikace. [13]

S vysokou podporou vlastností Bluetooth 5 je SoC připraven využít veškeré výhody zlepšeného výkonu Bluetooth LE, jako je prodloužený dosah (až 4x větší jako ve verzi 4.0), či zvýšené přenosové rychlosti (2Mbs). [13]

Dále nabízí vysoké množství digitálních rozhraní, jako je například vysokorychlostní SPI (32MHz) a quad SPI (32 MHz), umožňující připojení displeje a dalších externích pamětí. [13]

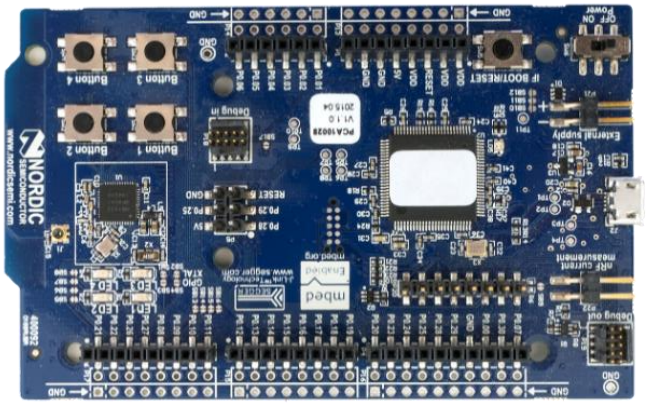
SoC lze napájet přímo z baterie, či z USB.

Díky jeho vysokému výkonu se zařízení využívá pro IoT (Smart Homes, Průmyslová mesh síť), či herní kontroléry.[13]

4.3 Development Kits

Pro implementaci praktické ukázky bakalářské práce byly vybrány tyto dva Development Kity. V následujících kapitolách budou stručně popsány s popisem nasazení.

4.3.1 nRF51 Development Kit



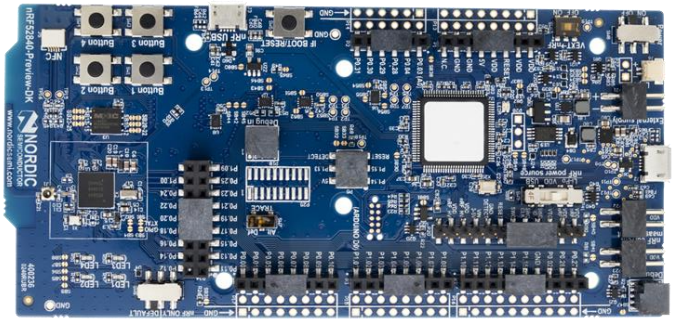
Obr. 4 nRF51 Development Kit. Zdroj: [14]

Development Kit nRF51 je kompatibilní s Arduino Revision 3, díky tomu jsou programátoři schopni vyvíjet veškerou funkcionalitu SoC řady nRF51. Na desce se nachází J-Link SEGGER debugger, který dokáže debugovat jak integrovaný, tak externě připojený čip. Pro snadný přístup jsou veškeré GPIO dostupné pomocí konektorů a headerů umístěných po stranách desky. Dále se zde nacházejí tlačítka a 4 LED diody, díky kterým lze snadno odlaďovat vstup, či výstup SoC. [14]

Celý kit může být napájen pomocí USB, či baterie, jejíž slot je umístěn ve spodu desky. [14]

Tento Development Kit bude využit pro porovnání datových rychlostí technologie Bluetooth Low Energy. Pro tento účel bude k tomuto DK připojen externí nRF52840 SoC.

4.3.2 nRF52840 Development Kit

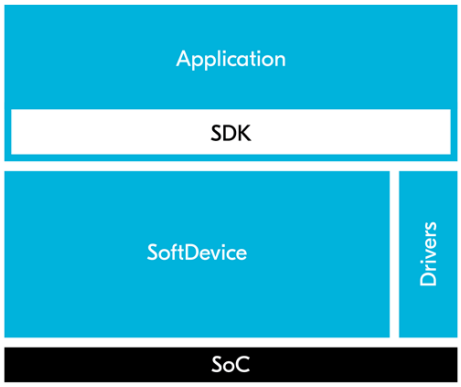


Obr. 5 nRF52840 Development Kit. Zdroj: [15]

Development Kit nRF52840 je využíván na vývoj aplikací pro nRF52840 SoC. Oproti předchozímu DK se zde například nachází podpora pro Bluetooth mesh a NFC anténa. [15]

Tento Development Kit bude sloužit pro vytvoření aplikace na skenování okolních zařízení a na ukázce technologie Bluetooth Mesh.

5 NRF5 Software Development Kit



Obr. 6 Overview architektury. Zdroj: [16]

NRF5 Software Development Kit (12.3.0 a 15.3.0) poskytuje vývojové prostředí pro SoC řady nRF5.

Nachází se zde tutoriály, ovladače, knihovny, ukázky řešení vybraných problémů, SoftDevices a proprietární protokoly.

Pro další kapitoly je nutné si vysvětlit tyto pojmy:

- Advertising Event je používán pro komunikaci na fyzické vrstvě. Při startu každého eventu vysílač vyšle advertising paket. Podle typu tohoto eventu, může skenovací zařízení vyslat scan request paket, na který vysílač odpoví pomocí stejného eventu. [17]
- Observer je role zařízení, které pasivně naslouchá ostatním BLE zařízením, a zpracovává data přijatých pomocí advertising paketu. [18]
- Broadcaster je role, kdy zařízení pouze vysílá advertising pakety s některými důležitými informacemi, a není se k němu potřeba připojit. [18]
- Periferie je zařízení, které dává o sobě vědět pomocí advertisingu ostatním zařízením. Většinou obsahuje nějaké informace, které jsou dostupné po připojení. Příkladem může být nějaký senzor, ke kterému se připojíme pomocí smartphonu a on nám předá svoje data. [18]
- Central je aktivní zařízení obklopené periferiemi, ke kterým se může připojit. [18]

5.1 SoftDevice

Jedná se o bezdrátovou protokolovou knihovnu, část firmwaru od společnosti Nordic Semiconductor, která je nahraná do určité oblasti mikrokontroleru. Tyto knihovny jsou předkompilované do binárního obrazu. Dále zajišťují ochranu paměti při běhu, bezpečnost vláken a deterministické chování v reálném čase. Při vytváření aplikace musí být API deklarováno v hlavičkových souborech pro programovací jazyk C. Veškeré služby zařízení jsou poskytovány aplikaci díky rozhraní, podobnému abstrakci hardwarového ovladače. [19]

Software Development Kit poskytuje spoustu SoftDevice pro různé technologie a SoC. Například S332, který kombinuje BLE a ANT protocol stack, nebo S132 určený pro nRF52810 a nRF52832. SoftDevice využité v praktické části bakalářské práce budou o něco blíže popsány v následujících podkapitolách. [19]

5.1.1 S130

SoftDevice S130 je Bluetooth 4.2 Low Energy protocol stack určený pro nRF51 SoC. Podporuje až 8 spojení společně s Observerem, který pasivně naslouchá ostatní zařízení, a Broadcasterem, který pouze vysílá data do okolí. [20]

5.1.2 S140

SoftDevice S140 je Bluetooth 5 Low Energy protocol stack pro SoC nRF52811 a nRF52840. Podporuje až 20 připojení společně s rolemi Observer a Broadcaster. Dále lze díky tomuto SoftDevice využívat v aplikaci datové rychlosti 2Mbps a Long Range. [21]

5.2 Knihovny

V této části bakalářské práce se nachází seznámení s některými základními procedurami a funkcemi, které již jsou implementovány v těchto vybraných knihovnách.

5.2.1 BLE Libraries

BLE Libraries poskytují funkce pro implementování Bluetooth Low Energy komunikaci.

5.2.1.1Database Discovery Module

Database Discovery Modul slouží k vyhledávání služeb na GATT serveru. Pokud se chce klient připojit k zařízení, které má požadovaný GATT server, je nutné zajistit, že budou pro komunikaci s charakteristikami použity správné handlery. Může se totiž stát, že zařízení pomocí advertisingu nevysílá veškeré svoje služby. [22]

5.2.1.2 Connection Parameters Negotiation

Tento modul je určen především pro periferie, které mají nastaveny již nějaké connection parametry. Může však nastat případ, že zařízení, které se k této periferii připojí nevyžaduje tak častou výměnu dat, a tak z důvodu snížení energetické náročnosti toto zařízení iniciuje změnu parametrů připojení. [23]

```
#define MIN_CONN_INTERVAL   MSEC_TO_UNITS(100, UNIT_1_25_MS)
/**< Minimum acceptable connection interval (0.5 seconds). */
#define MAX_CONN_INTERVAL   MSEC_TO_UNITS(200, UNIT_1_25_MS)
/**< Maximum acceptable connection interval (1 second). */
#define SLAVE_LATENCY       0
/**< Slave latency. */
#define CONN_SUP_TIMEOUT    MSEC_TO_UNITS(4000, UNIT_10_MS)
/**< Connection supervisory time-out (4 seconds). */

#define FIRST_CONN_PARAMS_UPDATE_DELAY  APP_TIMER_TICKS(20000)
/**< Time from initiating event (connect or start of notification)
 to first time sd_ble_gap_conn_param_update is called (15 seconds). */
#define NEXT_CONN_PARAMS_UPDATE_DELAY   APP_TIMER_TICKS(5000)
/**< Time between each call to sd_ble_gap_conn_param_update after the first call (5 seconds). */
#define MAX_CONN_PARAMS_UPDATE_COUNT    3
/**< Number of attempts before giving up the connection parameter negotiation. */

/**@brief Function for initializing the Connection Parameters module.
 */
static void conn_params_init(void)
{
    ret_code_t             err_code;
    ble_conn_params_init_t cp_init;
    memset(&cp_init, 0, sizeof(cp_init));
    cp_init.p_conn_params                  = NULL;
    cp_init.first_conn_params_update_delay = FIRST_CONN_PARAMS_UPDATE_DELAY;
    cp_init.next_conn_params_update_delay  = NEXT_CONN_PARAMS_UPDATE_DELAY;
    cp_init.max_conn_params_update_count   = MAX_CONN_PARAMS_UPDATE_COUNT;
    cp_init.start_on_notify_cccd_handle    = BLE_GATT_HANDLE_INVALID;
    cp_init.disconnect_on_fail             = false;
    cp_init.evt_handler                    = on_conn_params_evt;
    cp_init.error_handler                  = conn_params_error_handler;

    err_code = ble_conn_params_init(&cp_init);
    APP_ERROR_CHECK(err_code);
}
```

Obr. 7 Ukázka kódu nastavení connection parametrů. Zdroj: Vlastní tvorba

Na ukázce kódu výše je vidět možné nastavení connection parametrů.

5.2.1.3 Peer manager

Bluetooth Low Energy Peer Manager může být využit pro správu zabezpečení BLE, jako je šifrování, párování, či spojování. Využívá bezpečnostní postupy podle specifikace Bluetooth. Veškeré informace o připojených zařízeních ukládá na FLASH paměť. [24]

Po aktivaci Peer Manager nahrazuje aktuálního správce zařízení a přináší s sebou spoustu výhod. Hlavní funkce zůstávají stejné. [24]

- Podpora více rolí
    - Správce zařízení podporuje centrální i periferní roli, ale jakou roli bude využívat se musí rozhodnout už v době kompilace. [24]
- Autonomie
    - Peer Manager je více autonomní, například automaticky reaguje na požadavky na zabezpečení a má zabudovaný proces obnovení z chyb.
- Snadné použití
    - API tohoto manažeru je snadno použitelné, a díky vysoké úrovni autonomie je méně citlivý na uživatelské chyby. [24]
- Modularita
    - Je navržen tak, že je možné snadná údržba, testování, či naprogramování nových funkcí. [24]

- Zachytávání GATT atributů
    - Uchovává informace o GATT atributech vzdálených serverů pro veškeré peer zařízení a díky tomu redukuje požadovanou výměnu paketů, a tím se šetří energie. [24]
- Distribuce změněných indikátorů služby
    - Pokud aplikace informuje Peer manager o změně GATT databáze, on tuto informaci distribuuje na všechna připojena peer zařízení. [24]

5.2.1.4 Scanning Module

Zpracovává skenování Bluetooth Low Energy. Slouží k vyhledání zařízení, které právě vysílá a následně se s ním pokusí spojit. Toto vyhledávání lze specifikovat pomocí filtrů a seznamem povolených zařízení. [25]
Skenování může pracovat v těchto dvou módech:

- Simple mode
  - Využívá filtry a whitelist.
- Advanced mode
  - Využívá pokročilé filtry a whitelist.

```
static char const m_target_periph_name[] = "Nordic_Blinky";
/**< Name of the device we try to connect to. This name is searched in the scan report data*/
                .
                .
                .
ret_code_t          err_code;
// Connect if names match
init_scan.connect_if_match = true;

// Setting filters for scanning.
err_code = nrf_ble_scan_filters_enable(&m_scan, NRF_BLE_SCAN_NAME_FILTER, false);
APP_ERROR_CHECK(err_code);

err_code = nrf_ble_scan_filter_set(&m_scan, SCAN_NAME_FILTER, m_target_periph_name);
APP_ERROR_CHECK(err_code);
```

Obr. 8 Ukázka kódu Scan filter. Zdroj: Vlastní tvorba

Obrázek výše slouží jako ukázka kódu, jak by mohla vypadat implementace filtru. Pokud to je takto nastaveno, a sken najde zařízení se jménem „Nordic_Blinky“, pokusí se k němu připojit.

5.2.2 Timer Library

Tato knihovna označována též jako app_timer umožňuje aplikaci vytvořit několik instancí časovače založené na periferii RTC1. Kontrola časových limitů se provádí v obsluze přerušení RTC1. Prioritu daného přerušení lze měnit v sdk_config.h. [26]

```
#define OUR_CHAR_TIMER_INTERVAL     APP_TIMER_TICKS(1000) // 1000 ms intervals
//Define my timer
APP_TIMER_DEF(my_timer);
static void timers_init(void)
{
    // Initialize timer module.
    ret_code_t err_code = app_timer_init();
    APP_ERROR_CHECK(err_code);

    app_timer_create(&my_timer, APP_TIMER_MODE_REPEATED, timer_timeout_handler);
}
static void timer_timeout_handler(void * p_context)
{
      //What to do
}
```

Obr. 9 Ukázka kódu Implementace časovače. Zdroj: Vlastní tvorba

Na výše uvedeném obrázku je vidět možná implementace vlastního časovače. Akce, které chceme provádět budou zahrnuty ve funkci timer_timeout_handler.

5.3 Důležité koncepty BLE a jejich implementace

V této kapitole budou blíže popsány některé důležité aspekty technologie Bluetooth Low Energy společně s jejich implementací pomocí nRF5 Software Development Kit.

5.3.1 Advertising

Advertising souvisí s vysíláním dat, při kterém můžeme vyhledávat zařízení, či publikovat vybraná data. Vysíláme 2 typy až 31 bajtových paketů, a to Advertising paket anebo Scan response paket. Tyto dva typy paketů sdílejí on-air přístupovou adresu 0x8E89BED6, která umožňuje kterémukoliv zařízení skenovat a přijímat advertising nebo scan response data. [27]

5.3.1.1 Advertising paket

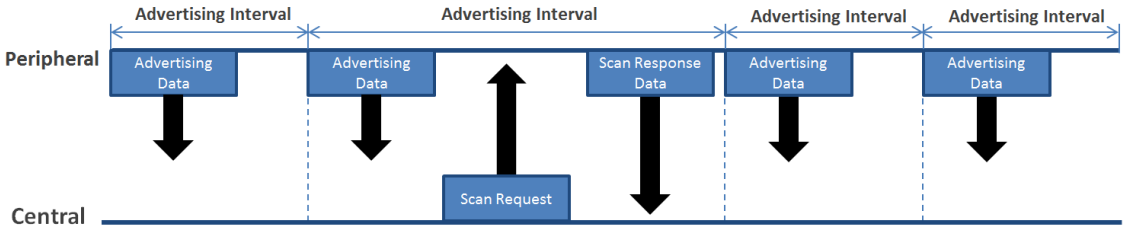
Inzerent v pravidelných intervalech vysílá Advertising paket spolu se svojí adresou. Tento úsek můžeme nastavovat podle svého uvážení v intervalu od 20 milisekund do 10,24 sekund. [27]

5.3.1.2 Scan response paket

Inzerent musí mít ve většině případů nastavenou periodu RX. Tato perioda se využívá buď při Connect requestu, nebo Scan requestu od Scanneru.

Pokud Inzerent přijme Scan request ze skenovacího zařízení žádajícího více informací, tak na něj odpoví pomocí Scan response paketu. Je nutno poznamenat,

že můžeme RX periodu vynechat z důvodu ušetření energie. Jedná se převážně o případ, pokud aplikace vysílá v non-connectable módu, kdy se nepotřebuje připojit k zařízení, nebo nepotřebuje posílat nadbytečná data v Scan response paketu. Obrázek níže popisuje, jak jsou výše zmíněné pakety vysílány. [27]



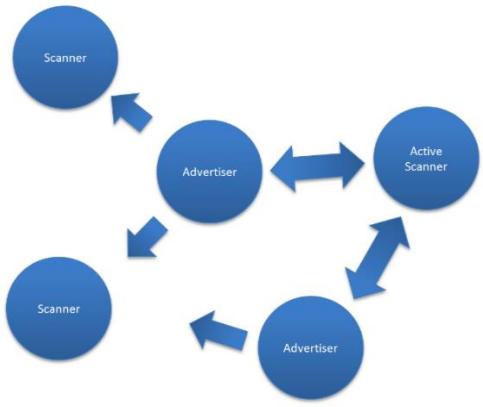
Obr. 10 Vysílání paketů. Zdroj: [27]

5.3.1.3 Kanály vysílání

Pro potřeby vysílání Advertising paketů jsou vyhrazeny 3 kanály z celkových 40 RF kanálů se středovým kmitočtem 2402+k*2MHz (číslo kanálu k je v rozmezí 0-39), které Bluetooth Smart využívá. Tyto kanály, 37 (2402MHz), 38 (2426MHz) a 39 (2480MHz), jsou vybrány tak, aby nekolidovaly s vytíženými Wifi kanály. [27]

5.3.1.4 Topologie vysílání

Pro potřeby advertisingu využíváme Broadcastovou topologii. Jedná se o connection-less síť, kde může být současně, jak více Inzerentů, tak Scannerů. [27]



Obr. 11 Advertising Topologie. Zdroj: [27]

Na obrázku lze vidět, že jedno zařízení může současně jak provádět advertising, tak i skenovat. Dále může být jedno zařízení připojeno k centrálnímu zařízení, či periferii, a přitom zároveň provádět advertising. Active Scanner dokáže pouze

vysílat Scan request, ve kterém je obsažena pouze jeho adresa. Passive Scanner zařízení neprovádějí Scan Request. [27]

5.3.1.5 Advertising typy

Existují 4 typy advertisingu.
Typ ADV_IND (connectable undirected advertising), který je velmi často využívaný. Zařízení zde může vysílat Scan response pakety a Connect requesty inzerentovi. [27]
Dále typ ADV_DIRECT_IND (connectable directed advertising), kde advertising paket je nasměrován na centrální zařízení, požádá o připojení, a pokud je odmítnut, tak jej veškeří inzerenti ho ignorují. Jedná se o broadcast paket, ale ostatní skenery ho ignorují, když se peer adresa s nimi neshoduje. [27]
Třetí typ ADV_SCAN_IND (scannable undirected advertising) pouze přijímá Scan request.
A poslední typ ADV_NONCONN_IND (non-connectable undirected advertising) pracuje bez RX periody, čili inzerent nepřijímá Connect ani Scan request. [27]

5.3.1.6 Typy adres

- Public address: Globální fixní adresa, která je registrována IEEE Registration autority a nikdy by neměla být měněna v životním cyklu zařízení. [28]
- Random static address: Náhodné číslo, které se může generovat při každém startu zařízení. [28]
- Private Resolvable address: Je generována z IRK (Identity resolving key) a náhodného čísla. Tato adresa se mění velice často, a to z důvodu bezpečnosti. [28]
- Private Non-Resolvable address: Náhodně generovaná adresa, která se může kdykoliv změnit. [28]

5.3.1.7 Implementace Advertising v SDK

Pro nastavení advertising a scan response dat slouží funkce sd_ble_gap_adv_set_configure (uint8_t *p_adv_handle, ble_gap_adv_data_t const *p_adv_data, ble_gap_adv_params_t const *p_adv_params), kde

Tabulka 1 Parametry funkce sd_ble_gap_adv_set_configure

p_adv_handle	Handler pro identifikaci advertising setu.
p_adv_data	Advertising data, může být nastaven na NULL
p_adv_params	Parametry

Zdroj: [29]

Start advertising funkcí sd_ble_gap_adv_start (uint8_t adv_handle, uint8_t conn_cfg_tag), kde

Tabulka 2 Parametry funkce sd_ble_gap_adv_start

adv_handle	Handler přijatý z funkce sd_ble_gap_adv_set_configure
conn_cfg_tag	Tag identifikující BLE konfiguraci od SoftDevice

Zdroj: [29]

Bližší konfiguraci je možné vidět ve zdrojových kódech praktické části bakalářské práce.

5.3.1.8 Eventy

Tato kapitola popisuje události, které mohou nastat

- BLE_GAP_EVT_TIMEOUT: Nastává, když uplyne Timeout advertisingu. Aplikace buď začne propagovat v jiném režimu, nebo přechází do režimu spánku. [27]
- BLE_GAP_EVT_SCAN_REQ_REPORT: Pokud aplikace dostane tento event, tak to znamená, že inzerent obdržel Scan request. [27]
- BLE_GAP_EVT_CONNECTED: Nastává, pokud se centrální zařízení připojí k periferii. [27]

5.3.2 Connection

Základní topologie Bluetooth Smart je hvězda, kde se centrální zařízení může připojit k několika periféríím, a zároveň periferie se mohou připojit pouze k jednomu centrálnímu zařízení. [30]

5.3.2.1 Navázání a ukončení spojení

Pokud centrální zařízení chce navázat připojení, tak zašle Connect request na vybraného inzerenta, který vysílá Advertising paket a zároveň je v periodě RX. Po obdržení tohoto paketu by měl inzerent přestat vysílat Advertising paket a zahájit připojení, které je definováno pomocí parametrů v Connect request paketu. [30]

Inzerent přijímá připojení, pokud je advertising v connectable módu a zároveň nastává jeden z těchto případů:

- Advertising je bez whitelistu.
- Adresa iniciátora je ve whitelistu.
- Jedná se o Directed advertising s platnou adresou iniciátora.

[30]

První connection event paket by měl být v předem definovaném vysílacím okně pomocí parametru transmit window size a transmit window offset. Po této události se z inzerenta stává Slave a iniciátor spojení se stává Masterem, který stále posílá v určitém intervalu connection event pakety, aby Slave vytvářel RX okna na každém intervalu připojení. Je nutné poznamenat, že Master nemá žádnou možnost, jak zjistit, zda je připojení na straně inzerenta schváleno, a tak předpokládá, že tomu tak bylo. [30]

5.3.2.1.1 Parametry připojení

V této podkapitole budou definovány parametry připojení.

Tabulka 3 LLData struktura pole v CONNECT_REQ PDU

LLData struktura pole v CONNECT_REQ PDU									
AA (4 oktety)	CRCInit (3 oktety)	WinSize (1 oktet)	WinOffset (2 oktety)	Interval (2 oktety)	Latency (2 oktety)	Timeout (2 oktety)	ChM (5 oktetů)	Hop (5 bitů)	SCA (3 bity)

Zdroj: [30]

- Connection Interval udává, jak často bude zasílán connection event paket od Master (Connection interval = interval*1,25ms). [30]
- Latency je počtem connect eventů, které může Slave přeskočit. [30]
- Timeout určuje, jak dlouho Master bude posílat connection eventy, bez odezvy Slave. [30]
- ChM říká, které kanály budou využívány a které nebudou. [30]
- Hop definuje, jak mnoho má být kanál přeskočen v ChM. [30]
- SCA (Sleep clock accuracy): Čím je větší, tím stoupá spotřeba energie na Slave. [30]
- AA: Unikátní přístupová adresa připojení, kterou budou používat obě zařízení. [30]
- WinSize: Výsledný TransmitWindowSize =WinSize*1,25ms. [30]
- WinOffset: Výsledný TransmitWindowOffset=WinOffset*1,25 ms. [30]

5.3.2.1.2 Whitelist

Pro připojení existují tyto dva whitelisty:

- Whitelist na iniciátorovi omezuje počet inzerentů. [30]
- Whitelist na inzerentovi, který obsahuje seznam iniciátorů připojení. [30]

5.3.2.1.3 Ukončení spojení

Ukončení spojení může nastat za těchto podmínek:

Za prvé jeden z účastníků spojení zašle LL_TERMINATE_IND PDU. Za druhé, když spojení vypršelo. Za třetí, když se zařízení ze začátku nedokázalo připojit, například, když jej inzerent odmítl. [30]

Pro LL_TERMINATE_IND může nastat několik chybových kódů:

- REMOTE USER TERMINATED CONNECTION (0x13), když vzdálené zařízení, či uživatel ukončí spojení. [30]
- BLE_HCI_CONNECTION_TIMEOUT (0x08), pokud nastal Timeout. [30]
- CONNECTION TERMINATED BY LOCAL HOST (0x16). Tento kód nastal, pokud zařízení ukončilo spojení. [30]
- CONNECTION FAILED TO BE ESTABLISHED (0x3E) nastává, když Slave neobdržel, nebo odmítl Connect request. [30]
- UNACCEPTABLE CONNECTION PARAMETERS (0x3B), pokud Slave se nechce připojit s connection parametry od Master. [30]

5.3.2.1.4 Implementace Connection v SDK

Pro ukončení skenování (ukončeno i po navrácení erroru) a připojení slouží funkce sd_ble_gap_connect(ble_gap_addr_t const *p_peer_addr, ble_gap_scan_params_t const *p_scan_params, ble_gap_conn_params_t const *p_conn_params)), kde

Tabulka 4 Parametry funkce sd_ble_gap_connect

p_peer_addr	Pointer na adresu peeru
p_scan_params	Pointer na strukturu sken parametrů
p_conn_params	Pointer na požadované parametry připojení

Zdroj: [29]

Pro ukončení připojení slouží v SoftDevice funkce

sd_ble_gap_disconnect(uint16_t conn_handle, uint8_t hci_status_code))

Pro zastavení skenu:

sd_ble_gap_scan_stop()

5.3.3 BLE Služby a Charakteristiky

5.3.3.1 Attribute Protocol

Attribute Protocol je založen na modelu komunikace Klient Server, kde Server uchovává informace, například jaké jsou hodnoty senzorů, a posléze je uspořádává do tabulky. Každý atribut v této tabulce je hodnota, nebo část informace s některými přidruženými vlastnostmi.

Když klient požaduje data nějakého senzoru, odkazuje se například na 10 řádek této tabulky atributů. [31]

Heart Rate Profile	Handle	Type of attribute (UUID)	Attribute permission	Attribute value
Service Declaration	0x000E	**Service declaration** Standard UUIDservice 0x2800	Read Only, No Authentication, No Authorization	Heart Rate Service 0x180D
Characteristic Declaration	0x000F	**Characteristic declaration** Standard UUIDcharacteristic 0x2803	Read Only, No Authentication, No Authorization	Properties (Notify) Value Handle (0x0010) UUID for Heart Rate Measurement characteristic (0x2A37)
Characteristic Value Declaration	0x0010	**Heart Rate Measurement Characteristic** UUID found in the Characteristic declaration value 0x2A37	Higher layer profile or implementation specific.	Beats Per Minute E.g "167"
Descriptor Declaration	0x0011	**Client Characteristic Configuration Descriptor (CCCD)** Standard UUIDservice 0x2800	Readable with no authentication or authorization. Writable with authentication and authorization defined by a higher layer specification or is implementation specific.	Notification enabled 0x000X
Characteristic Declaration	0x0012	**Characteristic declaration** Standard UUIDcharacteristic 0x2803	Read Only, No Authentication, No Authorization.	Properties (READ), Value Handle (0x0011), UUID for Body Sensor Location (0x2A38)
Characteristic Value Declaration	0x0013	**Body Sensor Location** UUID found in the Characteristic declaration value 0x2A38	Higher layer profile or implementation specific	Sensor Location (8-bit integer) E.g. 3 equals "Finger"

Obr. 12 Příklad tabulky atributů. Zdroj: [31]

- Attribute Handle jednoznačně identifikuje atribut na serveru. Toto 16bitové číslo nemusí být sekvenční, avšak můžeme o něm uvažovat jako o čísle řádku tabulky. [31]
- Attribute type je hodnota UUID, která bude blíže vysvětlena v samostatné podkapitole. [31]

- Attribute Permission definuje pravidla, jak zacházet s daným atributem, tedy s jeho hodnotou. Zda se do něj může zapisovat, nebo jen z něho číst a jaká je požadovaná autentifikace pro tyto operace. [31]
- Attribute value je jakákoliv hodnota, jako je číslo a textový řetězec, avšak to může být i informace, kde lze najít další atributy a jejich vlastnosti [31]

Service Declaration attribute je na začátku každé skupiny atributů a má vždy typ 0x2800. Jeho hodnota je vždy UUID, které definuje, o jakou službu se jedná. [31]

Characteristic declaration je podobná Service Declaration a má vždy typ 0x2803. Práva jsou nastavena na Read Only s možností zapnutí autentifikace. Ve své hodnotě uchovává

- Číslo Handleru své Characteristic Value Declaration.
- UUID, což udává, jakou hodnotu, nebo typ informací můžeme očekávat v Characteristic Value Declaration.
- Popis vlastností, jak se má s hodnotou této charakteristiky zacházet. Například, když je nastavena hodnota 0x02, tak můžeme danou hodnotu číst. [31]

Characteristic value declaration obsahuje již požadovanou hodnotu. [31]

Descriptor Declaration je poslední řádek v řetězci obsahuje:

- Novou Characteristic Declaration
- Novou Service Declaration
- Descriptor Declaration, například CCCD, které umožní posílat notifikace při každé změně teploty

[31]

5.3.3.2 The Generic Attribute Profile

"The GATT Profile specifies the structure in which profile data is exchanged. This structure defines basic elements such as services and characteristics, used in a profile." [1]

Jednoduše řečeno jedná se o pravidla, jak svazovat, prezentovat a přenášet data pomocí technologie Bluetooth Low Energy. [32]

5.3.3.3 Služby

"A service is a collection of data and associated behaviors to accomplish a particular function or feature. [...] A service definition may contain […] mandatory characteristics and optional characteristics." [1]

Služby jsou souborem informací, jako jsou například hodnoty některých senzorů. Každá aplikace může definovat své vlastní služby, avšak Bluetooth SIG chtěla usnadnit a standardizovat vývoj, tak předdefinovala své vlastní služby jako například Heart Rate. [32]

5.3.3.4 BLE Charakteristiky

"A characteristic is a value used in a service along with properties and configuration information about how the value is accessed and information about how the value is displayed or represented." [1]

Charakteristiky reprezentují skutečné hodnoty a informace, které potřebujeme vysílat, jako například jednotky, bezpečnostní parametry a další metadata, která se starají a vymezují, jak se tyto data zapouzdřují do charakteristik. [32]

Charakteristiky se využívají převážně z důvodu flexibility, efektivity a kompatibility mezi zařízeními s různými platformami. Jejich používáním aplikace dociluje toho, že zařízení rychle zjišťují, jaké informace jsou dostupné, a následně jim sdělí jen nezbytně nutné informace, čímž dosahuje vysoké časové efektivity a úspory energie. [32]

5.3.3.5 Universally Unique ID (UUID)

Universally Unique ID je unikátní číslo používané k identifikaci atributů, služeb, a charakteristik. Existují dva druhy UUID.

Prvním je energeticky nenáročný a paměťově efektivní UUID o velikosti 16 bitů. Poskytuje omezený počet unikátních ID. Z tohoto důvodu existuje pravidlo, které říká, že by se měly o této velikosti pouze přenášet předem definované UUID, a to od Bluetooth SIG. Například Heart rate service má UUID 0x180D a její uzavřená charakteristika Heart Rate Measurement characteristic má UUID 0x2A37. [32]

Pro účely tvoření vlastních služeb a charakteristik tedy existuje UUID o velikosti 128 bitů. Skládá se z vygenerovaného base UUID, která vypadá například takto 4A98????-1CC4-E7C1-C757-F1267DD021E8 a námi určeného 16bitového UUID, které reprezentují otazníky (např. A001). [32]

5.3.3.6 Implementace Služeb a Charakteristik v SDK

V této kapitole bude popsána práce se službami a charakteristikami pomocí Software Development Kitu ve verzi 15.3.0 a SoftDevice S140, která je implementována v příloze bakalářské práce, a to konkrétně v příkladu 03_APP.

5.3.3.6.1 Implementace služby

Pro deklaraci nové služby musíme v programu vytvořit novou proměnnou například typu ble_os_t. Tento datový typ je strukturou, která bude později definována. Pro větší přehlednost je dobré si pro službu vytvořit dva nové soubory pojmenované například my_service.h a my_service.c. V header souboru si definujeme BASE UUID a UUID nové služby společně s předem zmíněnou strukturou, která ponese její informace, jako je service handler, či jednotlivé handlery charakteristik. Do souboru my_service.c implementuje funkci pro inicializaci nové služby, ve které zaregistrujeme BASE UUID a 16bitové UUID dané služby do BLE stack tabulky. Pokud proběhne vše v pořádku zavoláme funkci sd_ble_gatts_service_add (uint8_t type, ble_uuid_t const *p_uuid, uint16_t *p_handle), která přidá deklaraci služby do Attribute Table. [32]

```
//------------------ HEADER FILE
#define BLE_UUID_OUR_BASE_UUID {0x23,0xD1,0x13,0xEF,0xAA,0x12,0xDC,0xAB,0xDE,0xEF,0x12,0x07,0x00,0x00,0x00,0x00}
#define BLE_UUID_OUR_SERVICE   0xABCD
typedef struct
{
    uint16_t    service_handle;
    uint16_t                    conn_handle;
    // Characteristics handlers
    ble_gatts_char_handles_t    scan1b_handles;

}ble_os_t;
//------------------
void new_service_init(ble_os_t * p_our_service)
{

    uint32_t   err_code;
    ble_uuid_t         service_uuid;
    ble_uuid128_t      base_uuid = BLE_UUID_OUR_BASE_UUID;
    service_uuid.uuid = BLE_UUID_OUR_SERVICE;
    err_code = sd_ble_uuid_vs_add(&base_uuid, &service_uuid.type);
    APP_ERROR_CHECK(err_code);
    p_our_service->conn_handle = BLE_CONN_HANDLE_INVALID;

    err_code = sd_ble_gatts_service_add(BLE_GATTS_SRVC_TYPE_PRIMARY,
                                        &service_uuid,
                                        &p_our_service->service_handle);
    APP_ERROR_CHECK(err_code);
    // ADD Characteristics
    scan1a_char_add(p_our_service);

}
```

Obr. 13 Implementace služby. Zdroj: Vlastní tvorba

Pro distribuci služby je nutné ji přidat do scan response dat v inicializační funkci pro advertising.

5.3.3.6.2 Implementace Charakteristiky

Přidání charakteristiky do služby bude sloužit uživatelem definovaná funkce s parametrem odkazujícím se na strukturu informací o předem vytvořené službě. Prvním krokem je přidat nově definované UUID, které využívá stejné BASE UUID jako služba do BLE stack tabulky. Dalším krokem je konfigurace metadata atributů, které ho popisují. Úplné minimum této konfigurace je určení, kde budou tyto atributy uloženy. Existují dvě možnosti, a to BLE_GATTS_VLOC_STACK pro uložení do paměti kontrolované SoftDevice, anebo BLE_GATTS_VLOC_USER, který atributy uloží do části paměti kontrolované uživatelem. Dalšími kroky je umožnění zápisu, či čtení do charakteristiky a nastavení zabezpečení k jejich přístupu. Pro účely bakalářské práce, nebylo nastaveno žádné zabezpečení, což bylo docíleno použitím makra BLE_GAP_CONN_SEC_MODE_SET_OPEN. Předposledním krokem je nastavení hodnoty charakteristiky a určení její maximální délky. Posledním krokem je přidání nově vytvořené charakteristiky do služby. [31]

Na níže uvedeném obrázku je uvedena implementace charakteristiky do služby vytvořené v minulé kapitole.

```
//------------------HEADER
#define BLE_UUID_SCAN1_RSSI_CHARACTERISTC_UUID     0xB001
//------------------
static uint32_t scan1b_char_add(ble_os_t * p_our_service)
{
    // ADD UUID
      uint32_t            err_code;
      ble_uuid_t          char_uuid;
      ble_uuid128_t       base_uuid = BASE_UUID;
      char_uuid.uuid      = BLE_UUID_SCAN1_RSSI_CHARACTERISTC_UUID;
      err_code = sd_ble_uuid_vs_add(&base_uuid, &char_uuid.type);
      APP_ERROR_CHECK(err_code);
    // Add read property
    ble_gatts_char_md_t char_md;
    memset(&char_md, 0, sizeof(char_md));
    char_md.char_props.read = 1;
    // Attribute Metadata
    ble_gatts_attr_md_t attr_md;
    memset(&attr_md, 0, sizeof(attr_md));
    attr_md.vloc        = BLE_GATTS_VLOC_STACK;
    //  read security level
    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&attr_md.read_perm);
    // Configure the characteristic value attribute
    ble_gatts_attr_t    attr_char_value;
    memset(&attr_char_value, 0, sizeof(attr_char_value));
    attr_char_value.p_uuid      = &char_uuid;
    attr_char_value.p_attr_md   = &attr_md;
    // Set characteristic length in number of bytes
    attr_char_value.max_len     = 1;
    attr_char_value.init_len    = 1;
    uint8_t value[1]            = {0x00};
    attr_char_value.p_value     = value;
    // Add Characteristic to the service
    err_code = sd_ble_gatts_characteristic_add(p_our_service->service_handle,
                                               &char_md,
                                               &attr_char_value,
                                               &p_our_service->scan1b_handles);
     APP_ERROR_CHECK(err_code);

    return NRF_SUCCESS;
}
```

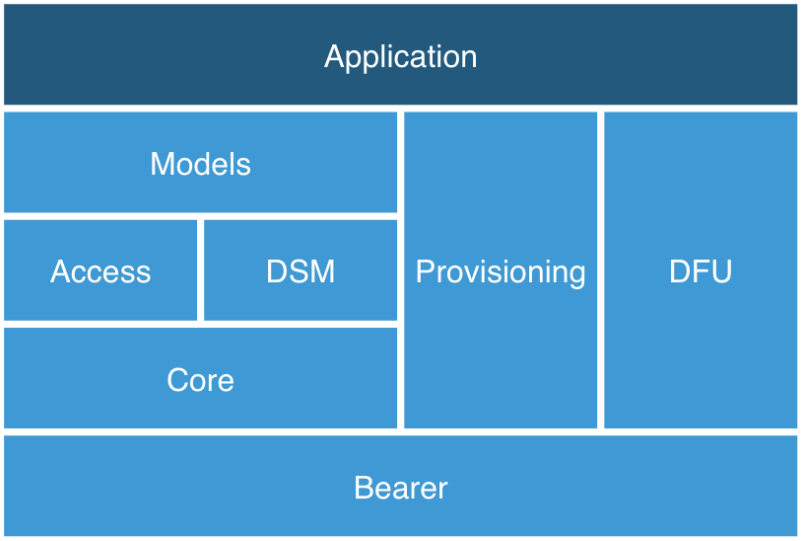
Obr. 14 Implementace charakteristik. Zdroj: Vlastní tvorba

Pro aktualizaci hodnoty charakteristiky slouží funkce sd_ble_gatts_value_set (uint16_t conn_handle, uint16_t handle, ble_gatts_value_t *p_value).

6 NRF5 Software Development Kit for Mesh

NRF5 Software Development Kit for Mesh implementuje technologii Bluetooth Mesh pro SoC řady nRF5. V tomto SDK jsou obsaženy tutoriály, knihovny a ukázky kódu, které mají za úkol umožnit a zejména usnadnit vývoj aplikací.

6.1 Architektura nRF5 SDK for Mesh



Obr. 15 Základní architektura Mesh v nRF5 SDK for Mesh. Zdroj: [33]

Architektura nRF5 SDK for Mesh koresponduje se strukturou Bluetooth Mesh sítě uvedenou v kapitole č. 3.4.1.9.

- Models
  - Modely definují chování a zprávy, které jsou specifické pro danou aplikaci. Standartní modely jsou definované Bluetooth SIG, ale vývojářům je umožněno vytvářet své vlastní. [33]
- Access
  - Vrstva Access je zodpovědná za určení, jaké mesh zprávy jsou určené pro specifický model. [33]
- DSM
  - Vrstva DSM zpracovává uložení šifrovacích klíčů a adres používané v mesh stacku, které ukládá do trvalého uložiště a poskytuje k nim handlery. [33]

Core

- Vrstva Core se skládá z transportní a síťové vrstvy. Transportní vrstva je zodpovědná za šifrování odesílaných mesh paketů a jejich rozdělení do segmentů. Zpracovává i sestavení přijatých mesh paketů, které posléze naformátuje do formy pro Access vrstvu. Síťová vrstva šifruje veškeré segmenty paketů transportní vrstvy a přidá zdrojovou a cílovou adresu. Přijaté segmenty dekóduje a pomocí zdrojové adresy určí, zda mají být dále zpracované. [33]

- Provisioning
  - Každé zařízení musí projít provisioning procesem, pokud se chce stát uzlem mesh sítě. Zařízení do sítě je možné připojit dvěma způsoby, a to přímo prostřednictvím PB-ADV (advertising provisioning bearer) anebo vzdáleně (remote provisioning). PB-ADV provisioning probíhá pouze mezi dvěma zařízeními, které jsou v přímém dosahu. Remote provisioning probíhá pomocí ostatních uzlů a je podporováno pouze zařízeními od Nordic Semiconductor. [33]
- DFU
  - Vrstva DFU zpracovává aktualizace firmwaru zařízení od Nordic Semiconductor v dané mesh síti bez omezení chodu aplikace. [33]
- Bearer
  - Vrstva Bearer zpracovává low-level radio controller operace a není přístupná z aplikační vrstvy. [33]

6.2 Popis ukázkového příkladu Light Switch

Tento ukázkový příklad od Nordic Semiconductor se nachází ve složce „nrf5SDKforMeshv320src/examples" přílohy bakalářské práce. Příklad implementuje mesh modely Generic OnOFF Client a Generic OnOFF Server. Pro ukázku je potřeba nejméně dvou nRF52840 DK. Jedno sloužící jako Light Switch Server a druhé jako Light Switch Client. Server uchovává stav LED1 a jedná na základě jeho hodnoty. Client na základě stisknutí libovolného tlačítka posílá

SET message na předem definovanou adresu. Provisioner je reprezentován pomocí aplikace nRF Mesh[1] na smartphonu a nastavuje celou síť.

Pro správnou funkci ukázky je potřebné v aplikaci nRF Mesh provést tyto kroky. Přiřadit každému zařízení unikátní adresu, distribuovat potřebné aplikační a síťové klíče a nakonfigurovat mesh modely všech zařízení. [34] [35]

Bližší popis všech potřebných kroků se bude nalézat v příloze bakalářské práce.

6.2.1 Implementace Klienta

V této kapitole bude popsána implementace Light Switch Client.

Po implementování potřebných nastavení zařízení zavolá funkci start. Ve funkci start započne nastavení a následné spuštění Beaconing, za podmínky, že zařízení není již uzlem. Pro Beaconing je využit jak Advertising, tak GATT bearer jelikož je potřeba komunikovat s mobilní aplikací. Dále zde funkcí mesh_stack_start proběhne konfigurace mesh stacku, přiřazení UUID zařízení a přiřazení callback funkcí pro jednotlivé mesh eventy. Po dokončení této funkce zařízení vstoupí do klidového stavu a vyčkává na vyvolání jednotlivých eventů. [35]

```
static void start(void)
{
rtt_input_enable(rtt_input_handler, RTT_INPUT_POLL_PERIOD_MS);

if (!m_device_provisioned)
{
    static const uint8_t static_auth_data[NRF_MESH_KEY_SIZE] = STATIC_AUTH_DATA;
    mesh_provisionee_start_params_t prov_start_params =
    {
        .p_static_data    = static_auth_data,
        .prov_complete_cb = provisioning_complete_cb,
        .prov_device_identification_start_cb = device_identification_start_cb,
        .prov_device_identification_stop_cb = NULL,
        .prov_abort_cb = provisioning_aborted_cb,
        .p_device_uri = EX_URI_LS_CLIENT
    };
    ERROR_CHECK(mesh_provisionee_prov_start(&prov_start_params));
}

mesh_app_uuid_print(nrf_mesh_configure_device_uuid_get());

ERROR_CHECK(mesh_stack_start());

hal_led_mask_set(LEDS_MASK, LED_MASK_STATE_OFF);
hal_led_blink_ms(LEDS_MASK, LED_BLINK_INTERVAL_MS, LED_BLINK_CNT_START);
}
```

Obr. 16 Ukázka kódu mesh funkce start. Zdroj: Vlastní tvorba

[1]https://play.google.com/store/apps/details?id=no.nordicsemi.android.nrfmeshprovisioner&hl=en

Popis jednotlivých funkcí:

- models_init_cb
    - Funkce pro inicializaci potřebných modelů. Pro potřeby ukázky jsou zde iniciováni dva Generic OnOff Klienti.
- provisioning_complete_cb
    - Tato funkce je vyvolána, pokud je úspěšně dokončen provisioning proces.
- config_server_evt_cb
    - V této funkci se zpracovávají veškeré události, které má server zpracovat. Například CONFIG_SERVER_EVT_NODE_RESET nastává, pokud je zařízení odebráno ze sítě.
- button_event_handler
    - Funkce pro zpracování akcí jednotlivých tlačítek.

```
#define APP_STATE_OFF                (0)
#define APP_STATE_ON                 (1)
static void button_event_handler(uint32_t button_number)
{
uint32_t status = NRF_SUCCESS;
//Mandatory parameters for the Generic OnOff Set message
generic_onoff_set_params_t set_params;
//Generic Transition parameters for the model messages.
model_transition_t transition_params;
static uint8_t tid = 0;
switch(button_number)
{
    case 0:
    case 2:
        // State to set
        set_params.on_off = APP_STATE_ON;
        break;

    case 1:
    case 3:
        // State to set
        set_params.on_off = APP_STATE_OFF;
        break;
}
// Transaction ID.
set_params.tid = tid++;
//Transition time value in milliseconds.
transition_params.delay_ms = APP_CONFIG_ONOFF_DELAY_MS;
//Message execution delay in milliseconds.
transition_params.transition_time_ms = APP_CONFIG_ONOFF_TRANSITION_TIME_MS;

switch (button_number)
{
    case 0:
    case 1:
        /* Demonstrate acknowledged transaction, using 1st client model instance */
        /* In this examples, users will not be blocked if the model is busy */

        (void)access_model_reliable_cancel(m_clients[0].model_handle);
        //Sends a Set message to the server.
        status = generic_onoff_client_set(&m_clients[0], &set_params, &transition_params);

        hal_led_pin_set(BSP_LED_0, set_params.on_off);
        break;

    case 2:
    case 3:
        /* Demonstrate un-acknowledged transaction, using 2nd client model instance */
        //Sends a Set Unacknowledged message to the server.
        status = generic_onoff_client_set_unack(&m_clients[1], &set_params,
                                                &transition_params, APP_UNACK_MSG_REPEAT_COUNT);
        hal_led_pin_set(BSP_LED_1, set_params.on_off);
        break;
}
ERROR_CHECK(status);
}
```

Obr. 17 Ukázka kódu mesh pro zaslání SET message. Zdroj: Vlastní tvorba

Ve výše uvedené ukázce je zobrazena implementace posílání SET message, která je vyvolána stisknutím tlačítka. Tlačítka 1 (ON) a 2(OFF) zasílají message jako Klient 1, které vyžadují od příjemce potvrzení přijetí. Podobné chovaní majíc i tlačítka 3 a 4, které posílají message jako Klient 2, avšak zde není vyžadováno potvrzení přijetí.

6.2.2 Implementace Server

Implementace Light Switch Server je podobná jako pro Light Switch Client.

Změny nastávají ve funkci app_model_init, kde je iniciován model Generic OnOff Server na element s indexem 0. Pro nastavení statusu LED od Light Switch Client slouží funkce app_onoff_server_set_cb(const app_onoff_server_t * p_server, bool onoff).

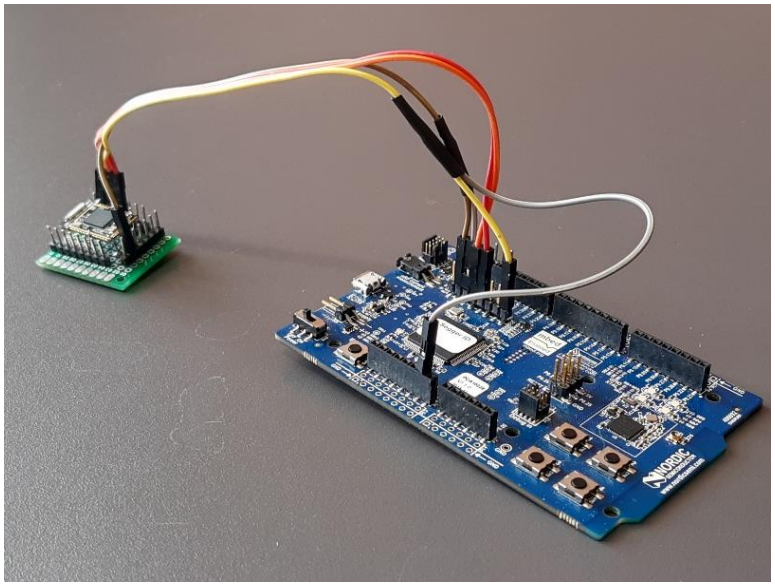
7 Demonstrační příklady

Pro implementaci demonstračních příkladů bylo použito vývojové prostředí SEGGER Embedded Studio for ARM ve verzi 4.12 a Eclipse Cpp IDE společně s GNU ARM Tool chain. Prostředí Eclipse sloužilo pro vývoj a debug aplikace staršího SoC nRF51822. Jako šablona byl použit ukázkový příklad ble_app_blinky. Z programu byly odstraněny nepodstatné funkce a doimplementována potřebná funkcionalita.
Zdrojové kódy je možné najít v podadresáři examples umístěný v adresáři nRF5_SDK_x.x.x pro daný čip (nRF51822 – v 12.3.0 a nRF52840 – v 15.3.0). Pro správnou funkci je dále nutné nahrát přes program nRFgo Studio SoftDevice S140 (nRF52840), nebo S130 (nRF51822), které je k dispozici jako součástí SDK ve složce components.

7.1 Testování dosahu SoC nRF51822 a nRF52840

7.1.1 Analýza

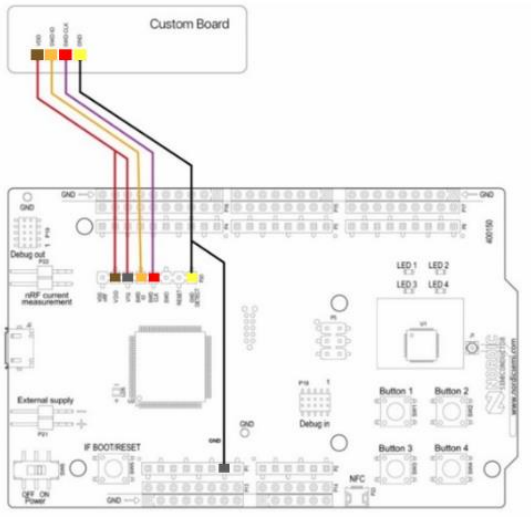
Zařízení podporující standard Bluetooth 5 dokáží komunikovat v několika režimech datových rychlostí, což má veliký vliv na jejich dosah. Úkolem této praktické části práce je porovnat klasickou rychlost 1 Mb/s a 125Kbps, označovanou též jako Bluetooth Long Range. Dalším úkolem bude porovnání dosahu jednotlivých SoC, a to zejména s ohledem na možnost nastavení většího výkonu vysílače na novějších čipech.
Testování bude prováděno pomocí dvou zařízení, a to nRF51 DK sloužící jako periferie a nRF52840 DK vystupující jako Central.



Obr. 18 Periferie s externě připojeným SoC. Zdroj: Vlastní tvorba

Na periferii bude umístěn BLE modul od společnosti Holyiot s integrovaným čipem nRF52840. Aby mohl tento modul sdílet debugger a zdroj napájení s periférií, je nutné ho připojit pomocí konektorů podle níže uvedeného schématu na obrázku č.19**.**

Modul bude napájen na plošný spoj a pomocí kabelů propojen jeho pin VDD s piny VDD a VTG periferie, GND s piny GND DETECT a GND, a piny SWD IO, SWD CLC mezi sebou.



Obr. 19 Schéma zapojení pro sdílení Debuggeru a zdroje napájení. Zdroj: Upraveno z [36]

7.1.1.1 Ztráty signálu

Při výběru lokalit pro porovnání měření bude brán ohled na útlum volného prostoru (Path Loss), ztrátu signálu způsobenou absorpcí signálu materiálem a dalších ztrát, které mohou být způsobeny odrazy ostatních signálů, srážkami, či vlhkostí.

„Path loss is the reduction in power density that occurs as a radio wave propagates over a distance. The primary factor in path loss is the decrease in signal strength over distance of the radio waves themselves. Radio waves follow an inverse square law for power density: the power density is proportional to the inverse square of the distance. Every time you double the distance, you receive only one-fourth the power. This means that every 6-dBm increase in output power doubles the possible distance that is achievable.“ [37]

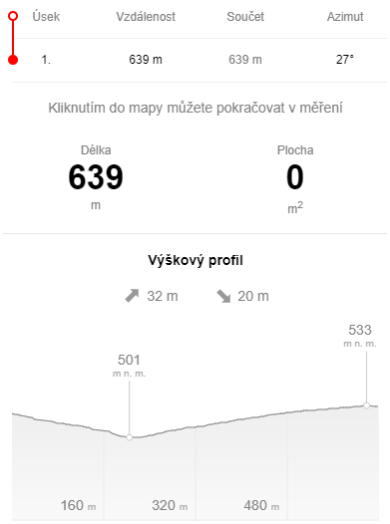
Díky tomuto útlumu a ztrátám signálu bude naměřen vyšší dosah například na rovném poli než v kopcovitém terénu. V interiéru budou tyto ztráty nejlépe patrné a dosah se výrazně sníží.

7.1.2 Návrh řešení

Návrh programu pro měření lze rozdělit na dvě části, a to na naprogramování periferie a centrálního zařízení.

Při měření bude centrální zařízení umístěno na pevném GPS bodě. Po spuštění a následném připojení periferie se bude postupně zvyšovat vzdálenost. Aby byl výsledek co nejpřesnější, je nutné, aby obě dvě zařízení byly nasměrovány svými anténami k sobě. Po odpojení zařízení, indikovaného LED1, bude zaznamenána GPS poloha. Aktuální poloha a vzdálenost mezi body bude stanovena pomocí aplikace Mapy.cz[2] a jejího nástroje Měření vzdáleností a plochy.

[2] https://www.mapy.cz



Obr. 20 Vzdálenost BLE Long Range 8dbm. Zdroj: Vlastní tvorba, Screenshot z aplikace

7.1.3 Princip činnosti

Periferie vysílá advertising pakety a bude se k ní možné připojit. Centrální zařízení slouží jako skener, a pokud nalezne periferii předem stanoveného jména, pokusí se s ní navázat spojení. LED na jednotlivých zařízení budou indikovat jejich aktuální stav. Pokud svítí LED 1 na periferii, je zařízení ve stavu Advertising. Na centrálním zařízení LED1 indukuje stav, kdy probíhá skenování. Rozsvícení LED2 indukuje stav Connected pro obě dvě zařízení.

7.1.4 Implementace

V této kapitole bude popsána implementace periferie a centrálního zařízení. Pro testování jsou v příloze ve složce HEX zkompilované hex soubory s různými režimy datových rychlostí a výkony antény pro jednotlivé SoC.

7.1.4.1 Nastavení Advertising parametrů periferie pro testování

Jednotlivé nastavení datových rychlostí se nalézá v inicializaci advertising. Na obrázku č. 21 můžeme vidět ukázku kódu, ve kterém je implementováno nastavení parametrů pro advertising. Pro uchování veškerých parametrů slouží proměnná typu ble_gap_adv_params_t. Po deklaraci této proměnné jsou k dispozici datová pole pro nastavování jednotlivých parametrů. Prvním z parametrů je primary_phy, které

definuje fyzickou vrstvu (PHY), kde jsou primárně vysílány pakety. Tento parametr určuje, jaké datové rychlosti budou použity. V SoftDevice S140 může pro náš případ nabývat hodnot BLE_GAP_PHY_1MBPS a BLE_GAP_PHY_CODED (Bluetooth Long Range). Druhým parametrem je duration, kterým je možné nastavit dobu trvání advertisingu. Třetím parametrem je properties, ve kterém se nachází vlastnosti advertising eventů. Nejdůležitější z nich je nastavení jejich typu, který bude potřeba změnit pro aktivování Long Range. Dalším parametrem je filter_policy, kde filtrujeme Scan a Connect request pakety. Hodnotou BLE_GAP_ADV_FP_ANY, je nastaveno, že budou přijaty requesty z jakéhokoliv zařízení. Posledním parametrem je interval, který udává, v jakém rozmezí budou advertising pakety vysílány. [38]

```
#define APP_ADV_INTERVAL                64
#define APP_ADV_DURATION                0  /**< The advertising time-out*/



ble_gap_adv_params_t adv_params;
// Set advertising parameters.
memset(&adv_params, 0, sizeof(adv_params));
adv_params.primary_phy     = BLE_GAP_PHY_1MBPS;
adv_params.duration        = APP_ADV_DURATION;
adv_params.properties.type = BLE_GAP_ADV_TYPE_CONNECTABLE_SCANNABLE_UNDIRECTED;
adv_params.p_peer_addr     = NULL;
adv_params.filter_policy   = BLE_GAP_ADV_FP_ANY;
adv_params.interval        = APP_ADV_INTERVAL;
```

Obr. 21 Nastavení advertising parametrů pro 1Mbps. Zdroj: Vlastní tvorba

Pro aktivování módu Long Range s možností připojení, je nutné nastavit v proměnné uchovávající parametry pro advertising tyto datová pole.

1. primary_phy na BLE_GAP_PHY_CODED, aby zařízení vysílalo rychlostí 128kbps. [38]
2. secondary_phy na BLE_GAP_PHY_CODED. Zařízení totiž bude vysílat eventy typu extended connectable, a právě toto PHY je použito pro navázání spojení. [38]

3. properties.type:
   BLE_GAP_ADV_TYPE_EXTENDED_CONNECTABLE_NONSCANNABLE_UNDIRECTED [38]

Ostatní datová pole mohou být ponechána stejně jako při módu 1Mbps. Kód pro inicializaci by měl tedy vypadat takto.

```
#define APP_ADV_INTERVAL                64
#define APP_ADV_DURATION                0  /**< The advertising time-out*/



adv_params.primary_phy = BLE_GAP_PHY_CODED;
adv_params.secondary_phy = BLE_GAP_PHY_CODED;
adv_params.properties.type = BLE_GAP_ADV_TYPE_EXTENDED_CONNECTABLE_NONSCANNABLE_UNDIRECTED;
adv_params.duration = APP_ADV_DURATION;
adv_params.p_peer_addr = NULL;
adv_params.filter_policy = BLE_GAP_ADV_FP_ANY;
adv_params.interval = APP_ADV_INTERVAL;
```

Obr. 22 Ukázka kódu inicializace Long Range. Zdroj: Vlastní tvorba

7.1.4.2 Nastavení Central zařízení pro Skenování

Pro skenování okolních zařízení je využit Scanning module, se kterým bylo seznámeno v kapitole 5.2.1.4.

```
NRF_BLE_SCAN_DEF(m_scan);


static void scan_init(void)
{
ret_code_t          err_code;
nrf_ble_scan_init_t init_scan;

memset(&init_scan, 0, sizeof(init_scan));

init_scan.connect_if_match = true;
init_scan.conn_cfg_tag     = APP_BLE_CONN_CFG_TAG;

err_code = nrf_ble_scan_init(&m_scan, &init_scan, scan_evt_handler);
APP_ERROR_CHECK(err_code);

// Setting filters for scanning.
err_code = nrf_ble_scan_filters_enable(&m_scan, NRF_BLE_SCAN_NAME_FILTER, false);
APP_ERROR_CHECK(err_code);

err_code = nrf_ble_scan_filter_set(&m_scan, SCAN_NAME_FILTER, m_target_periph_name);
APP_ERROR_CHECK(err_code);
}

.
.
.

err_code = nrf_ble_scan_start(&m_scan);
APP_ERROR_CHECK(err_code);
```

Obr. 23 Ukázka kódu inicializace Scanning modulu. Zdroj: Vlastní tvorba

V této ukázce kódu je předvedeno, jak je vyřešeno skenování. Prvním příkazem se vytvoří nová instance Scanning modulu. Ve funkci scan_init vytvoříme instanci

nrf_ble_scan_init_t. S pomocí jejího datového pole connect_if_match je nastaveno, že pokud je pomocí filtru (v tomto případě SCAN_NAME_FILTER) nalezeno požadované zařízení, pokusí se k němu centrální zařízení připojit.

Pro nastavení skenování módu Bluetooth Long Range a následného připojení byla použita reakce na event BLE_GAP_EVT_ADV_REPORT (Advertising report), který je vyvolán, pokud skener přijme advertising paket. [39]

```
#define SCAN_INTERVAL                   0x00A0
#define SCAN_WINDOW                     0x0050
static uint8_t m_enc_scandata[BLE_GAP_SCAN_BUFFER_EXTENDED_MIN];
static ble_gap_scan_params_t m_scan_param =
{
        .extended       = 1,
        .active        = 0x01,
        .interval      = SCAN_INTERVAL,
        .window        = SCAN_WINDOW,
        .timeout       = 0x0000, // No timeout.
        .scan_phys     = BLE_GAP_PHY_CODED,
        .filter_policy = BLE_GAP_SCAN_FP_ACCEPT_ALL,
};
static ble_data_t m_scan_buffer =
{
    m_scan_buffer_data,
    BLE_GAP_SCAN_BUFFER_EXTENDED_MIN
};

/**@brief Function to start scanning.
*/
static void scan_start(void)
{
ret_code_t err_code;
// If already scanning, stop
(void) sd_ble_gap_scan_stop();
// Signalize scanning
bsp_board_led_on(CENTRAL_SCANNING_LED);
// Start scanning
err_code = sd_ble_gap_scan_start(&m_scan_param, &m_scan_buffer);
APP_ERROR_CHECK(err_code);
}
```

Obr. 24 Ukázka kódu nastavení Central pro Long Range. Zdroj: Vlastní tvorba

Ve výše uvedené ukázce kódu je předvedeno, jak bylo nastaveno Centrální zařízení, aby mohlo skenovat Long Range advertising pakety. Prvním krokem je inicializace proměnné pro buffer, kde budou extended advertising reporty od SoftDevice po dobu skenování uloženy. Instance struktury ble_data_t uchovává velikost a pointer na tento buffer. Dalším krokem je vytvoření nové instance typu ble_gap_scan_params, sloužící pro nastavení parametrů skeneru. V parametrech je

nutné nastavit datové pole extended na hodnotu 1, pro přijímání extended advertising paketů, a datové pole scan_phys na BLE_GAP_PHY_CODED. [40]

7.1.5 Výsledky a testování

Z důvodu porovnání bylo měření prováděno v interiéru a volném prostranství. Jako volné prostranství byla vybrána louka s mírným kopcovitým terénem a jako interiér 3. patro budovy J Univerzity Hradec Králové.

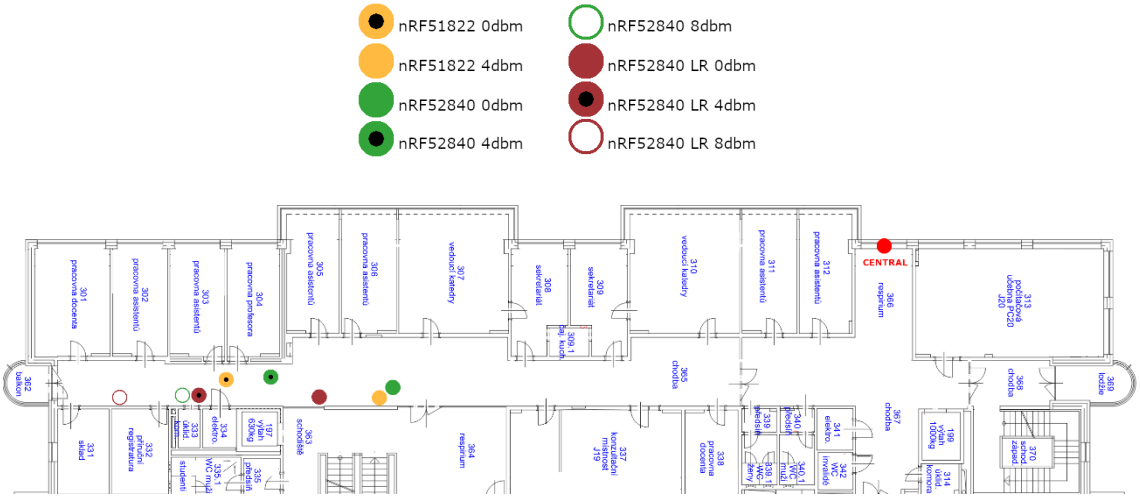
Ve volném prostranství byly naměřeny tyto hodnoty.

Tabulka 5 Naměřené hodnoty ve volném prostranství

Měření	SoC na Periferii	Použité PHY	Výkon antény	Vzdálenost
1.	nRF51822	BLE_GAP_PHY_1MBPS	0dbm	300 m
2.	nRF51822	BLE_GAP_PHY_1MBPS	4dbm	346 m
3.	nRF52840	BLE_GAP_PHY_1MBPS	0dbm	290 m
4.	nRF52840	BLE_GAP_PHY_1MBPS	8dbm	446 m
5.	nRF52840	BLE_GAP_PHY_CODED	0dbm	520 m
6.	nRF52840	BLE_GAP_PHY_CODED	8dbm	639 m

Zdroj: Vlastní zpracovaní

V interiéru nebyl dosah měřen v metrech, ale byl vyznačen na plánu budovy.



Obr. 25 Mapa měření interiér 1. Zdroj: Vlastní tvorba

Obr. 26 Mapa měření interiér 2. Zdroj: Vlastní tvorba

Z výsledků je možné odvodit, že SoC nRF51822 i nRF52840 mají při stejných výkonech antény podobné dosahy. Delšího dosahu bylo docíleno pouze navýšením výkonu antény na 8dbm, které starší nRF51822 nepodporuje. Po aktivování Bluetooth Long Range se dosah nRF52840 při 0dbm ve volném prostředí zvýšil o 79,3 %.

7.2 Aplikace pro skenování BLE zařízení

7.2.1 Analýza

Záměrem tohoto demonstračního příkladu je vytvořit funkční zařízení pro skenovaní okolních BLE zařízení. Prototyp bude zhotoven na vývojovém kitu nRF52840. Tento skener umožní, ve spolupráci k němu připojenému zařízení, zobrazení MAC adres a posledních naměřených RSSI hodnot prvních deseti naskenovaných zařízení.

7.2.2 Princip činnosti

Celá činnost začne spuštěním skenovacího zařízení, které zahájí sken okolních zařízení. Během této činnosti jsou jednotlivé advertising reporty zpracovávány a ukládány do pomocného datového pole. Po uplynutí předem definovaného časového úseku (10 vteřin) jsou jednotlivé záznamy z pole načteny do charakteristik a zařízení vstoupí do stavu advertising. V tomto stavu čeká, dokud se k němu

nepřipojí zařízení, které si dané charakteristiky přečte. Skenování lze znovu spustit pomocí tlačítka Button1.

7.2.3 Návrh řešení

O přenos dat mezi zařízení se bude starat služba, která v sobě bude obsahovat dvacet uspořádaných read only charakteristik pro uchování dat prvních deseti naskenovaných zařízení. Jednotlivé záznamy budou identifikovány pomocí UUID charakteristiky. Například hodnoty prvního naskenovaného zařízení budou uchovány v charakteristikách s UUID A001 (MAC adresa) a UUID B001(RSSI).
K zahájení skenování bude sloužit tlačítko Button 1 na desce skenovacího zařízení a tento stav bude reprezentován pomocí LED3. Po dokončení skenu se rozsvítí LED1 oznamující, že zařízení je připraveno ke spojení a poskytnutí dat.

Tabulka 6 Přehled služeb a charakteristik

	Charakteristika	Čtení/Zápis	Rozsah hodnoty	Příklad hodnoty(little-endien)	Význam hodnoty	Popis
Služba (UUID: ABCD)	A001	R	0-0xFFFFFFFFFF	E5DF50D758C9	E5:DF:50:D7:58:C9	MAC adresa zařízení 1
	B001	R	0-0xFF	3C	60 dbm	RSSI zařízení 1
	A002	R	0-0xFFFFFFFFFF	B7DF50D758C9	B7:DF:50:D7:58:C9	MAC adresa zařízení 2
	B002	R	0-0xFF	3A	58 dbm	RSSI zařízení 2
	A003	R	0-0xFFFFFFFFFF	C6DF50D758C9	C6:DF:50:D7:58:C9	MAC adresa zařízení 3
	B003	R	0-0xFF	30	48 dbm	RSSI zařízení 3
	A004	R	0-0xFFFFFFFFFF	D5DF50D758C9	D5:DF:50:D7:58:C9	MAC adresa zařízení 4
	B004	R	0-0xFF	3A	58 dbm	RSSI zařízení 4
	A005	R	0-0xFFFFFFFFFF	02DF50D758C9	02:DF:50:D7:58:C9	MAC adresa zařízení 5
	B005	R	0-0xFF	32	50 dbm	RSSI zařízení 5
	A006	R	0-0xFFFFFFFFFF	C7DF50D758C9	C7:DF:50:D7:58:C9	MAC adresa zařízení 6
	B006	R	0-0xFF	37	55 dbm	RSSI zařízení 6
	A007	R	0-0xFFFFFFFFFF	02DF50D758A7	02:DF:50:D7:58:A7	MAC adresa zařízení 7
	B007	R	0-0xFF	31	49 dbm	RSSI zařízení 7
	A008	R	0-0xFFFFFFFFFF	C8C950D758C9	C8:C9:50:D7:58:C9	MAC adresa zařízení 8
	B008	R	0-0xFF	32	50 dbm	RSSI zařízení 8
	A009	R	0-0xFFFFFFFFFF	E6DF50D758C9	E6:DF:50:D7:58:C9	MAC adresa zařízení 9
	B009	R	0-0xFF	3A	58 dbm	RSSI zařízení 9
	A010	R	0-0xFFFFFFFFFF	AADF50D758C9	AA:DF:50:D7:58:C9	MAC adresa zařízení 10
	B010	R	0-0xFF	3C	60 dbm	RSSI zařízení 10

Zdroj: Vlastní zpracování.

7.2.4 Implementace

Implementace služby a charakteristik je realizována v souborech new_service.h a new_service.c podle kapitoly č.5.3.3.

Hlavní funkcionalitou je sběr dat. V průběhu skenování se realizuje pomocí handleru pro BLE_GAP_EVT_ADV_REPORT. Pokud tento event nastane je zavolána funkce on_adv_report, jejíž parametrem je daný advertising report s potřebnými daty. Pro uchování zpracovaných dat bylo vytvořeno pole s instancemi struktury myreport.

```
struct myreport
{
uint8_t mac[6];
int8_t rssi;
};
// Array of structs
struct myreport arr_myreport[255];
// Number of scanned devices
int count=0;

static void on_adv_report(ble_gap_evt_adv_report_t const * p_adv_report)
{
ret_code_t err_code;
static int8_t   rssi_value = 0;
struct myreport temp_report;
//copy array with MAC addr
for(int j =0;j<6;j++)
{
temp_report.mac[j]=p_adv_report->peer_addr.addr[j];
}
// GET RSSI value
rssi_value = p_adv_report->rssi;
rssi_value=abs(rssi_value);
temp_report.rssi=rssi_value;

bool find = false;
for(int i=0; i <= count; i++ )
{
    if(count>0)
    {
        if(memcmp(temp_report.mac, arr_myreport[i].mac, sizeof(temp_report.mac)) == 0)
        {
        // Device is already in array of structs, only copy updated rssi
        arr_myreport[i].rssi=temp_report.rssi;
        find=true;
        break;
        }
    }
}
if(find==false)
{
arr_myreport[count]=temp_report;
// Increase number of Devices
count++;
}
// Continue in scanning
err_code = sd_ble_gap_scan_start(NULL, &m_scan_buffer);
APP_ERROR_CHECK(err_code);
}
```

Obr. 27 Ukázka kódu funkce on_adv_report: Zdroj: Vlastní tvorba

Po uplynutí předem definované doby (SCAN_TIMEOUT) je vyvolán BLE_GAP_EVT_TIMEOUT event, v jehož handleru se pomocí funkcí definovaných v souboru new_service.c překopírují hodnoty z pole struktur do charakteristik. Zařízení je nyní připraveno k připojení.

Pro usnadnění čtení z charakteristik je v příloze přítomen python script, který se připojí z RaspberryPi ke skenovacímu zařízení a přijaté hodnoty přehledně vypíše do příkazové řádky.

8 Závěr

Stanovené cíle práce se podařilo splnit. Práce nastínila možnosti implementace Bluetooth Low Energy aplikace na čipech od společnosti Nordic Semiconductor. Práce také okrajově přiblížila Bluetooth Mesh technologii, která může být použita v průmyslu 4.0.

Na vývojovém kitu nRF52840 byl úspěšně realizován demonstrační příklad pro skenovaní okolních zařízení. Příklad není příliš rozsáhlý, ale může sloužit jako základ budoucího vývoje aplikace založené na této platformě.

Při implementaci druhého demonstračního příkladu nastalo několik problémů. Prvním z problému byla absence šablony SEGGER Embedded Studio projektu pro starší čip nRF51822. Musela být tedy použita metoda programování čipu pomocí GNU ARM Toolchain a Eclipse IDE, při kterém je šablona vygenerovaná z MakeFile obsaženého v ukázkovém příkladu. Druhou překážkou byly prvotní problémy při aktivaci režimu Long Range v šabloně ble_app_blinky. Kromě těchto dvou problémů, implementace programu na měření vzdáleností dopadla úspěšně.

Měření bylo prováděno elementární cestou, pomocí indikace stavu LED. Pro větší přesnost, by měl být při budoucím testování použit dron a implementována komunikace s GPS zařízením, aby se co nejvíce zamezilo rušení signálu. Pro měření dosahu signálu uvnitř budovy je vhodnější použít indoor lokalizaci.

Programy lze zkompilovat a nahrát na zařízení pomocí volně dostupných programů, jako je nRFgo Studio a SEGGER Embedded Studio.

9 Seznam použité literatury

[1] Bluetooth SIG, *Bluetooth Core Specification* [online]. [cit. 2019-08-12]. Dostupné z: https://www.bluetooth.com/specifications/bluetooth-core-specification/

[2] Bluetooth SIG, *Origin of the Bluetooth Name* [online]. [cit. 2019-08-12]. Dostupné z: https://www.bluetooth.com/about-us/bluetooth-origin/

[3] Nordic Semiconductor, *A short history of Bluetooth* [online]. [cit. 2018-07-16]. Dostupné z: https://www.nordicsemi.com/eng/News/ULP-Wireless-Update/A-short-history-of-Bluetooth

[4] KILIÁN, Karel. *Bluetooth 5: jaké jsou největší výhody proti starší verzi 4.2?*. [online]. 20.12.2018 [cit. 2019-08-12]. Dostupné z: https://www.svetandroida.cz/bluetooth-5/

[5] Nordic Semiconductor, *Bluetooth 5* [online]. [cit. 2019-08-12]. Dostupné z: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/Bluetooth-5

[6] PALIVEC, Pavel a Marco WEIL. *Bluetooth Low Energy*. [online]. 03.2018 [cit. 2018-07-16]. Dostupné z: https://www.dps-az.cz/soucastky/id:9912/bluetooth-low-energy

[7] WOOLLEY, Martin. *An Intro to Bluetooth Mesh Part 1*. [online]. [cit. 2019-08-12]. Dostupné z: https://www.bluetooth.com/blog/an-intro-to-bluetooth-mesh-part1/

[8] AFANEH, Mohammad. *The Ultimate Bluetooth Mesh Tutorial (Part 1)*. [online]. 03.09.2018 [cit. 2019-08-12]. Dostupné z: https://www.novelbits.io/bluetooth-mesh-tutorial-part-1/

[9] AFANEH, Mohammad. *The Ultimate Bluetooth Mesh Tutorial (Part 2)*. [online]. 10.09.2018 [cit. 2019-08-12]. Dostupné z: https://www.novelbits.io/bluetooth-mesh-tutorial-part-2/

[10] AFANEH, Mohammad. *The Ultimate Bluetooth Mesh Tutorial (Part 3)*. [online]. 13.09.2018 [cit. 2019-08-12]. Dostupné z: https://www.novelbits.io/bluetooth-mesh-tutorial-part-3/

[11] Nordic Semiconductor, *About Nordic Semiconductor* [online]. [cit. 2019-08-12]. Dostupné z: https://www.nordicsemi.com/About-us

[12] Nordic Semiconductor, *nRF51822 product brief* [online]. [cit.2018-07-16]. Dostupné z: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF51822

[13] Nordic Semiconductor, *nRF52840 product brief* [online]. [cit. 2019-08-12]. Dostupné z: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840

[14] Nordic Semiconductor, *nRF51 DK* [online]. [cit. 2018-07-16]. Dostupné z: https://www.nordicsemi.com/Software-and-Tools/Development-Kits/nRF51-DK

[15] Nordic Semiconductor, *nRF52840 DK* [online]. [cit. 2019-08-12]. Dostupné z: https://www.nordicsemi.com/Software-and-Tools/Development-Kits/nRF52840-DK

[16] Nordic Semiconductor, *Software* [online]. [cit. 2019-08-13]. Dostupné z: https://www.nordicsemi.com/Software-and-Tools/Software

[17] GUPTA, Naresh C. *Inside Bluetooth Low Energy*. Boston: Artech House, 2013. ISBN 9781608075799.

[18] GEORGEL, Bogdan Alexandru. *A view on Bluetooth Low Energy stack roles* [online]. 08.03.2018 [cit. 2019-08-14]. Dostupné z: https://community.nxp.com/thread/332319

[19] Nordic Semiconductor, *SoftDevice* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/struct_nrf52/struct/nrf52_softdevices.html

[20] Nordic Semiconductor, *S130 SoftDevice* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/struct_nrf51/struct/s130.html

[21] Nordic Semiconductor, *S140 SoftDevice* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/struct_nrf52/struct/s140.html

[22] Nordic Semiconductor, *Database Discovery Module* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.3.0/lib_ble_db_discovery.html

[23] Nordic Semiconductor, *Connection Parameters Negotiation* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.3.0/lib_ble_conn_params.html

[24] Nordic Semiconductor, *Peer* Manager [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.3.0/lib_peer_manager.html

[25] Nordic Semiconductor, *Scanning Module* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.3.0/lib_ble_scan.html

[26] Nordic Semiconductor, *Timer Library* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.3.0/lib_timer.html

[27] BUI, Hung. *Bluetooth Smart and the Nordic's Softdevices – Part 1 GAP* Advertising [online]. 10.12.2015 [cit. 2019-08-14]. Dostupné z: https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/bluetooth-smart-and-the-nordics-softdevices-part-1

[28] Carles. *Re: GAP Address types* [příspěvek v diskuzním fóru]. In devzone.nordicsemi.com [online]. 03.04.2014, 09:29 [cit.2019-08-14]. Dostupné z: https://devzone.nordicsemi.com/f/nordic-q-a/2084/gap-address-types?ReplySortBy=CreatedDate&ReplySortOrder=Ascending

[29] Nordic Semiconductor, *Functions* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.s140.api.v6.0.0/group__b_l_e__g_a_p__f_u_n_c_t_i_o_n_s.html

[30] Bui Hung. *Bluetooth Smart and the Nordic's Softdevices - Part 2* [online]. 01.02.2016 [cit. 2019-08-13]. Dostupné z: https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/bluetooth-smart-and-the-nordics-softdevices-part-2

[31] Martin BL. *Bluetooth low energy Characteristics, a beginner's tutorial* [online]. 18.03.2016 [cit. 2019-08-14]. Dostupné z: https://devzone.nordicsemi.com/nordic/short-range-guides/b/bluetooth-low-energy/posts/ble-characteristics-a-beginners-tutorial

[32] Martin BL. *Bluetooth low energy Services, a beginner's tutorial* [online]. 26.08.2015 [cit. 2019-08-14]. Dostupné z: https://devzone.nordicsemi.com/nordic/short-range-guides/b/bluetooth-low-energy/posts/ble-services-a-beginners-tutorial

[33] AFANEH, Mohammad. *The Ultimate Bluetooth Mesh Tutorial (Part 4)*. [online]. 17.9.2018 [cit. 2019-08-14]. Dostupné z: https://www.novelbits.io/bluetooth-mesh-tutorial-part-4/

[34] Nordic Semiconductor, *Light switch example* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.meshsdk.v3.2.0/md_examples_light_switch_README.html

[35] AFANEH, Mohammad. *The Ultimate Bluetooth Mesh Tutorial (Part 6)*. [online]. 08.10.2018 [cit. 2019-08-14]. Dostupné z: https://www.novelbits.io/bluetooth-mesh-tutorial-part-6/

[36] Remi. Re: *External programming Using nRF52-DK* [příspěvek v diskuzním fóru]. In devzone.nordicsemi.com [online]. 09.10.2016, 11:47 [cit. 2019-08-13]. Dostupné z: https://devzone.nordicsemi.com/f/nordic-q-a/14058/external-programming-using-nrf52-dk

[37] DOWNEY, Chris. *Understanding Wireless Range Calculations* [online]. 03.04.2013 [cit. 2019-08-13]. Dostupné z: https://www.electronicdesign.com/communications/understanding-wireless-range-calculations

[38] Nordic Semiconductor, *ble_gap_adv_params_t Struct Reference* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.s140.api.v6.0.0/structble__gap__adv__params__t.html

[39] Nordic Semiconductor, *nrf_ble_scan_init_t Struct Reference* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.3.0/structnrf__ble__scan__init__t.html

[40] Nordic Semiconductor, *ble_gap_scan_params_t Struct Reference* [online]. [cit. 2019-08-14]. Dostupné z: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.s140.api.v6.0.0/structble__gap__scan__params__t.html

10 Přílohy

Obsah přiloženého CD:

- Adresář Nordic Semi obsahuje 6 podadresářů
  - HEX: Vygenerované .hex soubory jednotlivých programů pro snadnější naprogramování vývojového kitu.
  - Manual: Manuál pro zprovoznění aplikací.
  - nRF5_SDK_12.3.0: Zde se v podadresáři examples/ble_peripheral/03_range_test_nrf51822 nachází zdrojový kód vytvořeného demonstračního příkladu pro testovaní dosahu 1Mbps (Periferie, nRF51822)
  - nRF5_SDK_15.3.0: Zde se v podadresářích složky examples nachází zdrojové kódy, a projekty pro SEGGER Embedded Studio
    - ble_peripheral/03_APP – Vytvořený demonstrační příklad pro skenování okolních zařízení (nRF52840)
    - ble_peripheral/01_ble_range_test – Vytvořený demonstrační příklad pro testovaní dosahu 1Mbps (Periferie, nRF52840)
    - ble_peripheral/02_ble_long_range_test – Vytvořený demonstrační příklad pro testovaní Bluetooth Long Range (Periferie, nRF52840)
    - ble_central/01_ble_range_test_c – Vytvořený demonstrační příklad pro testovaní dosahu 1Mbps (Central, nrf52840)
    - ble_central/02_ble_long_range_test_c – Vytvořený demonstrační příklad pro testovaní Bluetooth Long Range (Central, nRF52840)
  - nrf5SDKforMeshv320src
    - Zde se v podadresáři examples/light_switch_example nachází kopie ukázkového projektu od Nordic Semiconductor pro Bluetooth Mesh
  - Python
    - Python script pro výpis hodnot charakteristik pro 03_APP

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2017/2018

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Mádr Stanislav	Pod Černým vrchem 226, Horní Maršov	I1600572

TÉMA ČESKY:

Vývoj Bluetooth Low Energy zařízení na bázi chipů NRF5x

TÉMA ANGLICKY:

Development of Bluetooth Low Energy Device Using NRF5x Chip

VEDOUCÍ PRÁCE:

Ing. Pavel Kříž, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl: seznámit s chipy společnosti Nordic Semiconductor, technologiemi Bluetooth a Bluetooth Low Energy, a následně s jejich využitím při vývoji aplikací. Popsat vývojové prostředí NRF5 SDK. Praktickou část práce tvoří vývoj aplikace (firmware) pro Bluetooth Low Energy zařízení na bázi chipu NRFx.

Osnova:
1. Úvod
2. Technologie Bluetooth
3. Bluetooth chipy Nordic Semiconductor
4. Vývoj aplikací, demonstrační příklad(y)
5. Výsledky testování
6. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

Zdroje:
https://www.nordicsemi.com/eng/Products/Bluetooth-5
https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK
https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v13.0.0%2Flib_crypto.html
https://www.bluetooth.com/

Podpis studenta: Mádr S. ..... **Datum:** 10.10.18

Podpis vedoucího práce: ..... **Datum:** 10.10.18