FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Sběr dat pomocí Raspberry Pi a technologie Bluetooth pro Internet věcí
Student:	Ondřej Pírko
Vedoucí:	Ing. Radomír Polách
Studijní program:	Informatika
Studijní obor:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	Do konce zimního semestru 2019/20

Pokyny pro vypracování

Nastudujte princip komunikace pomocí technologie Bluetooth v prostředí platforem Raspberry Pi a Android pro potřeby Internetu věcí.
Navrhněte, analyzujte, implementujte platformu pro získávání dat z Raspberry Pi pomocí technologie Bluetooth.
Při implementaci platformy se zaměřte na snadné získávání a přenos senzorických dat z Raspberry Pi a snadné uložení, zpracování a analýzu dat ze strany operačního systému Android. V úvahu berte také bezpečnost a možnost propojení většího množství Raspberry Pi k jednomu zařízení s operačním systémem Android pro získávání dat z více zdrojů.
Implementaci důkladně testujte pomocí vhodné demo aplikace.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Karel Klouda, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 2. března 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Sběr dat pomocí Raspberry Pi a technologie Bluetooth pro Internet věcí

Ondřej Pírko

Katedra aplikované matematiky
Vedoucí práce: Ing. Radomír Polách

13. května 2018

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2018 .....

Odkaz na tuto práci

Pírko, Ondřej. *Sběr dat pomocí Raspberry Pi a technologie Bluetooth pro Internet věcí*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato bakalářská práce se zabývá vytvořením platformy k zprostředkování komunikace přes Bluetooth mezi jedním Android zařízením a jedním či více Raspberry Pi (Linux zařízeními). Android zde funguje jako master, který sbírá data z připojených zařízení a zpracovává je. Tato komunikace je vedená obousměrně. Na straně Androidu je předmětem této práce knihovna v Javě a na straně Linuxu knihovna v Pythonu. Knihovny zjednodušují tento typ komunikace mezi zařízeními.

Klíčová slova Platforma pro získávání dat, Bluetooth, IOT, Android, Linux, Raspberry Pi, Java, Python

Abstract

This bachelor thesis deals with the creation of a platform, which is used to provide communication through Bluetooth between an Android device and one or more Raspberry Pis (Linux devices). Android figures here as a master who collects data from connected devices and processes them. This communication is bidirectional. On the Android side, the subject of this work is a Java library and on the Linux side it is a Python library. The libraries simplify this type of communication between the devices.

Keywords Platform for data retrieval, Bluetooth, IOT, Android, Linux, Raspberry Pi, Java, Python

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V současné době jsou počítače mnohem menší, výkonnější a levnější než kdy předtím. Díky tomuto faktu se na trhu objevují malé počítače, které je možno používat k řadě věcí, které předtím nebyly zcela možné. Příkladem takového malého počítače může být Raspberry Pi [1]. Představme si případ, kdy k jednomu Raspberry Pi máme připojených několik teploměrů nebo jiných senzorů a data z nich jsou tímto počítačem sbírána, případně i do jisté míry zpracovávána. Pokud bychom však získaná data chtěli analyzovat, či na ně nějak reagovat, musíme tak učinit přímo na tom počítači, který je získal. To není vždy praktické. Zvlášť když budeme chtít zpracovávat data z více počítačů sériově či paralelně. Je tudíž moudré data přenést na nějaké centrální zařízení, kde je možné nasbírat a zpracovat data hned z několika počítačů. Záměrem mého projektu je tudíž vyřešit tento problém tak, že realizuji platformu, díky které je možná snadná komunikace Raspberry Pi (případně i jiného Linuxového počítače) s zařízením, které běží na operačním systému Android. Komunikace mezi zařízeními bude probíhat s použitím technologie Bluetooth. Co se týče bezdrátové výměny dat, tak jde o alternativu k technologii Wi-Fi. Má smysl Bluetooth používat místo Wi-Fi pro tento účel? Pokud ano, tak za jakých podmínek? Tyto otázky a mnohé další zodpovím v této práci.

Kapitola 1

Cíle práce

1.1 Rešeršní část

1.1.1 Seznámení se s technologií Bluetooth

Prvně je třeba se seznámit s tím, jak vlastně technologie Bluetooth funguje a zda se dá účinně použít pro komunikaci mezi zařízeními. Je také nutno zvážit výhody a nevýhody této technologie proti podobným technologiím, příkladně Wi-Fi.

1.1.2 Prozkoumání platforem

Tato práce je rozdělena na dvě části. První část je navrhnout a implementovat knihovnu na Android. Druhou částí práce je udělat to samé, nicméně na Raspberry Pi, respektive na Linux. Tudíž je nutné se seznámit s programováním na těchto platformách a rozhodnout o tom, které programovací postupy na tento projekt pasují nejvíce.

1.1.3 Návrh

Dále je důležité navrhnout strukturu knihoven a jakým způsobem bude komunikace přes Bluetooth řešena. Způsoby je třeba popsat a vybrat, který je pro účely této práce nejlepší.

1.2 Praktická část

1.2.1 Implementace

Po navržení projektu přichází čas k implementaci vybraného řešení. V této fázi se využívá již vybraných programovacích jazyků k implementaci komunikace mezi zařízeními skrze užití protokolu Bluetooth, který byl vybrán v rešeršní

části. Implementuje se tedy master zařízení na straně Androidu a slave zařízení na straně Raspberry Pi (Linuxu).

Kapitola 2

Použité technologie

2.1 Bluetooth

V této sekci jsou informace brány z [2], pokud ovšem není uvedeno jinak.

Bluetooth je zabezpečený bezdrátový standard pro odesílání a přijímání dat, který operuje v 2.4Ghz ISM pásmu. Toto pásmo používají i jiné technologie, například Wi-Fi. Tato technologie je založena na bázi paketů a využívá master-slave architekturu. K jednomu master zařízení je možno připojit až sedm slave zařízení ve stejnou dobu, nicméně slave zařízení může být připojeno pouze na jedno master zařízení. Toto síťové schéma se nazývá piconet (schéma je vidět na obrázku 2.1). Komunikace je vedena obousměrně, každé zařízení tedy může posílat i přijímat data.

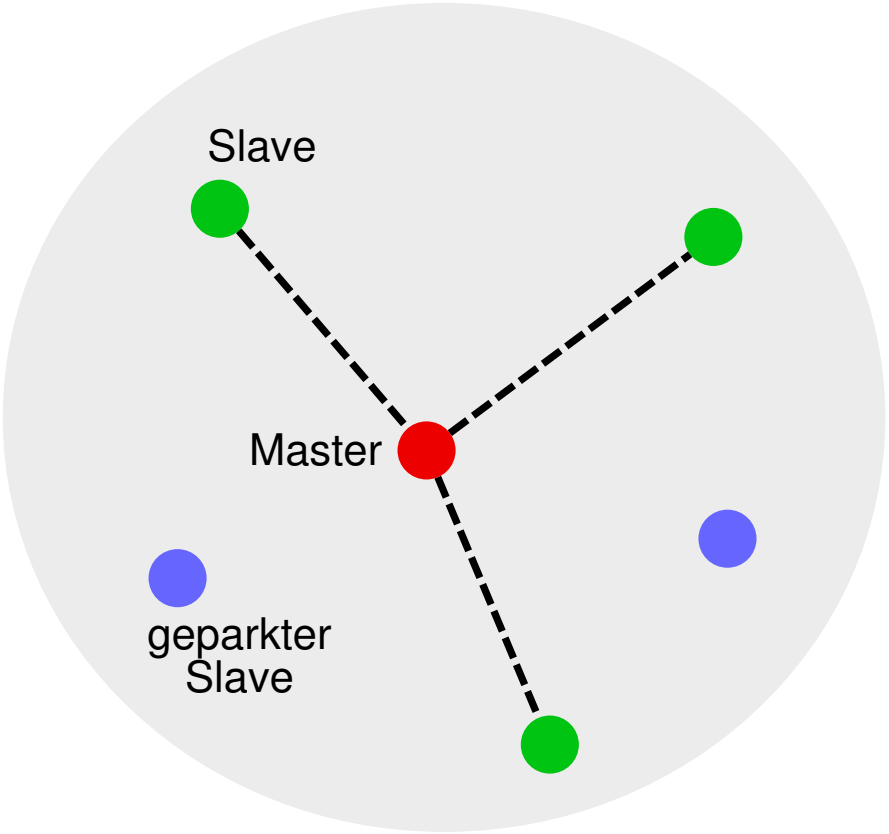
2.1.1 Srovnání verzí

- **Bluetooth 1.0**
  První verze Bluetooth, prakticky šlo jenom o „proof of concept“, funkčnost byla velmi limitovaná.

- **Bluetooth 1.1**
  Byly vyladěny problémy, které mělo Bluetooth 1.0 a byla přidána možnost nešifrovaných komunikací.

- **Bluetooth 1.2**
  Poslední verze řady 1.x a také první verze, která byla natolik dobrá, aby se dala používat pro praktické účely. Přenosová rychlost byla vyšší než kdy předtím: 721 kb/s s dosahem až 10 metrů.

- **Bluetooth 2.0 a 2.1 + EDR**
  Tyto verze přinesly především EDR (Enhanced Data Rate) a jednoduché párování (SSP). EDR zvýšilo rychlost přenosů dat na 2.1 Mb/s



Obrázek 2.1: Schéma sítě piconet. Označení „geparkter Slave“ znamená typ slavea, který není v tuto chvíli připojen. Převzato z [3].

a jednoduché párování zjednodušilo párování zařízení. Zároveň se proces párování upravil do takové míry, aby díky němu bylo připojení bezpečnější. Teoretický dosah se také zvýšil na až 100 metrů.

- **Bluetooth 3.0 + HS**
  Zde se přenosová rychlost razantně změnila na až 24 Mb/s. Pokud bylo využito HS (High Speed), tak Bluetooth pouze navázalo spojení mezi zařízeními, ale data byla následně přenášena pomocí Wi-Fi. Bez funkce HS má tato verze stejnou přenosovou rychlost jako verze předchozí. Je zde ovšem pár věcí navíc, zejména menší spotřeba a další profily připojení.

- **Bluetooth 4.0 + LE**
  Tato verze obsahuje tyto tři hlavní protokoly: Bluetooth classic (klasické), Bluetooth high speed (vysokorychlostní) a Bluetooth low energy (nízkoenergetické). První dva protokoly již známe z předchozích verzí, nicméně objevuje se zde zcela nové Bluetooth LE. Cíl tohoto nového protokolu je

signifikantně snížit spotřebu energie. Toto však přichází se sníženou rychlostí přenosu dat na pouhých 0.27 Mb/s a zároveň s mírným snížením dosahu. Spotřeba energie se u BLE (Bluetooth low energy) snížila na 1-50 % spotřeby původního Bluetooth.

- **Bluetooth 5**
  Nejnovější verze Bluetooth, podpora této verze mezi zařízeními je zatím malá. Zde již nemáme zápis verze „5.0“ jak bychom čekali, nýbrž pouze „5“. Tato změna je čistě z marketingových důvodů. Inovace v této verzi jsou především pro BLE, kde se objevilo několik nových možností. Je možno zvýšit přenosovou rychlost za cenu snížení dosahu nebo naopak zvýšit dosah za cenu snížení rychlosti. [4]

2.1.2 Přehled protokolů

Bluetooth používá pro komunikaci několik různých protokolů. Tato sekce popisuje některé z nich, a to zejména ty relevantní k této práci. [5]

- **L2CAP (Logical link control and adaptation protocol)**
  Hlavní protokol, který funguje na linkové vrstvě a slouží jako základ pro protokoly ve vyšších vrstvách. Jednou z jeho funkcí je rozdělování a skládání paketů, které mohou být velké až 64 kB. Další funkcí je například multiplexování protokolů.

- **SDP (Service discovery protocol)**
  Tento protokol je používán k objevování Bluetooth služeb v okolí a zjištění informací o nich. Je schopno zjistit například jméno služby, adresu zařízení se službou a UUID (Universally unique identifier) služby.

- **RFCOMM (Radio frequency communication)**
  Transportní protokol postavený nad protokolem L2CAP, který emuluje RS-232 sériové porty. Podobně jako TCP poskytuje jednoduchý a spolehlivý přenos dat, přičemž v jednu chvíli lze mezi dvěma zařízeními mít až 60 RFCOMM spojení.

2.1.3 Srovnání s Wi-Fi

Nejprve je vhodné porovnat Bluetooth s nějakou konkurenční technologií podobného rázu. Populární technologií na bezdrátovou výměnu dat je Wi-Fi. Je nutné posoudit, zda není Bluetooth podřadnou technologií vůči Wi-Fi. V tabulce 2.1 najdete srovnání technologie Bluetooth s Wi-Fi. [6] Z tabulky můžeme vidět, že Bluetooth vskutku horší technologií ve všech směrech vůči Wi-Fi není. Bluetooth se oproti Wi-Fi vyplatí používat hlavně v případech, kdy vám stačí malý dosah a relativně malá přenosová rychlost. Na druhou stranu za to dostanete snazší konfiguraci sítě, menší spotřebu energie a bude vás to stát méně.

Tabulka 2.1: Porovnání technologií Bluetooth a Wi-Fi.

Báze porovnání	Bluetooth	Wi-Fi
Pásmo	2.4 GHz	2.4 GHz a 5 GHz
Cena[1]	Nízká	Vysoká
Bezpečnost	Méně bezpečné	Bezpečnější
Dosah	Malý	Velký
Spotřeba energie	Malá	Velká
Maximální rychlost přenosu dat	2,1 Mb/s	200 Mb/s
Jednoduchost použití	Relativně jednoduché na užívání	Náročnější, je třeba více konfigurace hardwaru a softwaru

2.2 Android

V této sekci jsou informace čerpány z [7], pokud není uvedeno jinak.

Android je operační systém především na mobilní telefony a tablety založený na Linuxovém kernelu (jádře). Poslední dobou se však Android začíná objevovat i u chytrých televizí či hodinek. Ve sféře mobilních telefonů je Android pravděpodobně nejperspektivnějším operačním systémem. Jako argument k podložení tohoto tvrzení je možno uvést například jaký má obrovský podíl na trhu operačních systémů chytrých telefonů. To můžeme vidět na obrázku 2.2.

Period	Android	iOS	Windows Phone	Others
2016Q1	83.4%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%
2016Q4	81.4%	18.2%	0.2%	0.2%
2017Q1	85.0%	14.7%	0.1%	0.1%

Obrázek 2.2: Podíl operačních systémů na trhu chytrých telefonů. Převzato z [8].

[1] Bere se cena zprovoznění fungující sítě. To znamená cena příslušného hardwaru.

2.2.1 Programovací jazyky

K tomu, aby bylo vůbec možné napsat knihovnu pro Android je třeba vědět, v jakém programovacím jazyce je vhodné projekt psát. Na Android existuje velké množství jazyků, se kterými na něj můžete programovat aplikace, nicméně oficiálně podporované jsou pouze tři. Vývoj aplikací je totiž nejjednodušší s použitím oficiálního IDE (Integrated development environment), které má název Android Studio. A právě toto IDE podporuje pouze Javu, C/C++ a Kotlin. Nejrozumnější možností se zde jeví Java, která je na této platformě nejpoužívanějším programovacím jazykem. Dává tedy smysl vyvíjet knihovnu právě sem jestliže chci, aby se má knihovna používala co nejvíce. Následující subsekce s označením 2.2.2 je tedy vytvořená primárně pro popis struktury Java aplikace. [9]

2.2.2 Struktura aplikace

Aplikace se může skládat z čtyř různých komponent[2]:

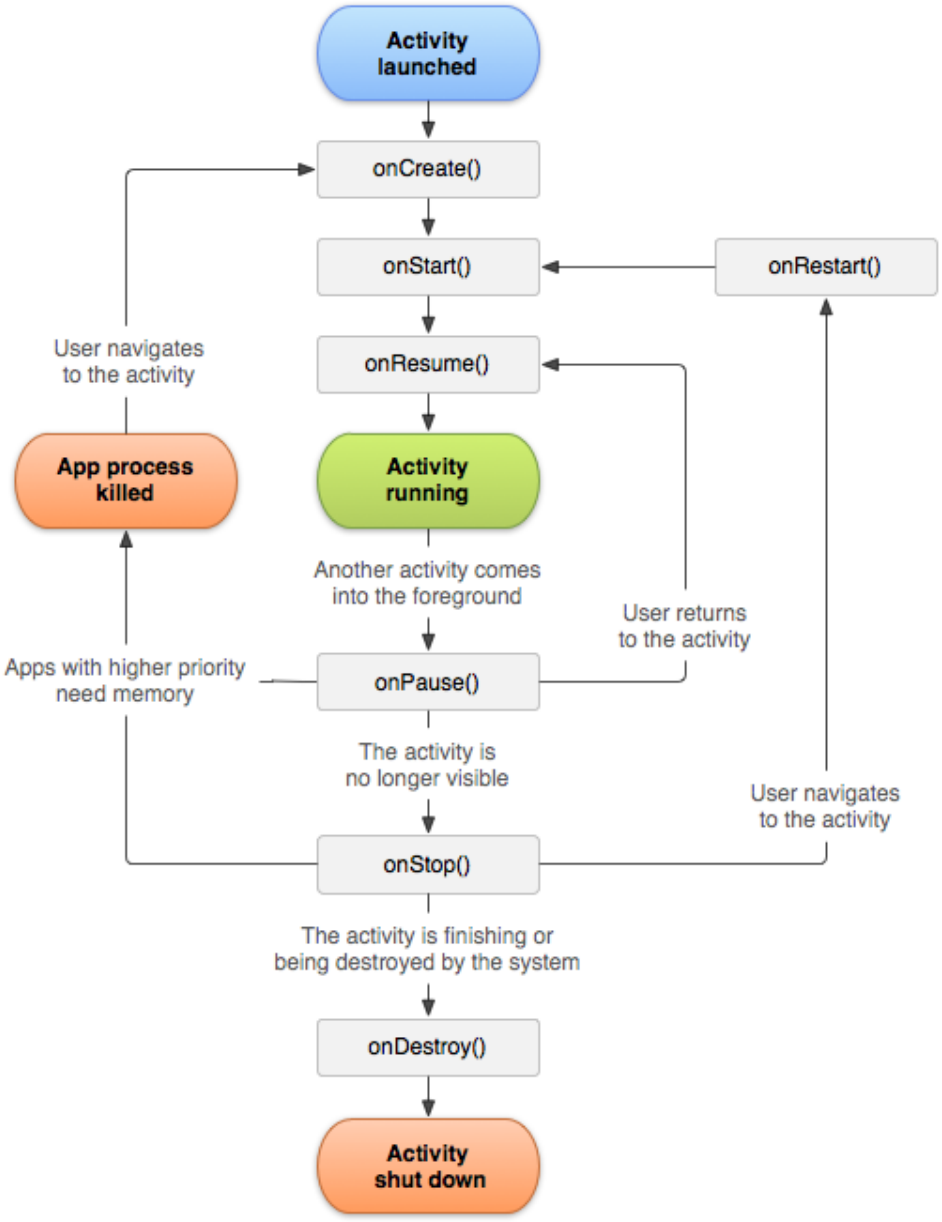
- Activities (Aktivity) jsou obrazovky s uživatelským rozhraním.
- Services (Služby) jsou komponenty, které běží na pozadí. Například jsou schopné pouštět hudbu.
- Broadcast receivers (Přijímače vysílání) umožňují systému posílání událostí aplikacím, poskytují jim tak způsob, jak reagovat na systémové oznámení.
- Content providers (Poskytovatelé obsahu) spravují data, která jsou sdílena mezi aplikacemi.

Nejdůležitější komponentou je aktivita. Ta má jasně daný životní cyklus, který je vidět na obrázku 2.3. Kdy se jaká metoda z tohoto cyklu volá je popsáno níže.

Třída *Activity* je navržená tak, aby se při každé změně stavu zavolalo jemu příslušné zpětné volání (callback). Následuje popis stavů aktivity. [11]

- **onCreate()**
  Tato metoda se zavolá jako první po vzniku aktivity. Píše se sem pouze to, co je zavoláno jen jednou na začátku běhu, například počáteční inicializace proměnných nebo deklarace uživatelského rozhraní. Po dokončení se okamžitě volá metoda *onStart()*.
- **onStart()**
  Metoda je zavolaná vždy, kdy se aktivita zviditelní pro uživatele (dostane do popředí) a stane se interaktivní. Po svém dokončení, stejně jako předchozí metoda, hned zavolá další metodu, kterou je *onResume()*.

[2] Píši zde pouze hrubou sumarizaci.



Obrázek 2.3: Životní cyklus aktivity. Převzato z [10].

- **onResume()**
  Když už je aktivita v popředí a začíná interagovat s uživatelem, je zavolána právě tato metoda. Stav se tu nemění, pokud uživatel stále používá pouze tuto aktivitu v popředí.

- **onPause()**
  Pokud se něco dostane do popředí před aktivitu a ta již není plně aktivní, zavolá se tato metoda. Příkladem může být případ, kdy uživateli někdo zavolá nebo se uživateli objeví nějaký dialog, který „vyskočí“ před aktivitu.

- **onStop()**
  Tato metoda se zavolá, pokud již aktivita není viditelná pro uživatele (je na pozadí). Je také zavolána, když se aktivita ukončuje. Zde by se měly uvolnit všechny zdroje, které nejsou nutně potřeba, jelikož uživatel v tuto chvíli s aktivitou neinteraguje.

- **onDestroy()**
  Poslední zavolaná metoda v životním cyklu aktivity. Pokud v aktivitě potřebujete před jejím zánikem něco vyřešit, zde máte tu možnost.

Aplikace na Androidu fungují na bázi intents (úmysl či záměr). Intent je objekt, který umožňuje vyžádání nějaké akce od aplikační komponenty. To znamená příkladně že pokud chce jistá aplikace zapnout Bluetooth, tak vytvoří intent, který požádá objekt *BluetoothAdapter* o zapnutí Bluetooth. Jak a kdy operace (vyžádaná intentem) skončila můžeme zjistit pomocí vhodného broadcast recieveru. Příklad formulace jednoduchého intentu na zapnutí Bluetooth můžeme vidět níže.

```
Intent enableBtIntent = new
    Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
```

Další důležitá část aplikace je manifest. Za pozornost stojí existence systému povolení. Ten spočívá v tom, že pokud chce aplikace používat například Bluetooth nebo fotoaparát, tak se musí zeptat uživatele o svolení. Ten má možnost to aplikaci buď povolit, nebo nikoli. Tato povolení se u aplikace zapisují právě do zmíněného aplikačního manifestu, který je psaný v XML. Je zde také napsáno, jakou minimální úroveň API (Application programming interface) aplikace vyžaduje, či seznam komponent. Zde můžeme vidět kus manifestu, který deklaruje, že aplikace vyžaduje povolení na kontrolu Bluetooth[3]:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

2.2.3 Zavádění aplikace

Android SDK (Software development kit) zkompiluje kód včetně všech závislostí do souboru zvaného APK (Android package). Tento soubor je pak použit na

[3]Druhý řádek dává aplikaci svolení k objevování a párování zařízení.

zařízeních s Androidem k instalaci aplikace. Standardně je v systému zakázáno instalovat APK z jiného zdroje než je Google Play, což je oficiální služba digitální distribuce pro Android[4]. Vnitřně Android funguje jako Linuxový systém s několika navzájem koexistujícími uživateli, kde uživateli jsou aplikace, které jsou identifikované pomocí ID. Je také možné, aby dvě aplikace měly stejné ID. To je k užitku například v případě, že by mezi sebou chtěly jednoduše sdílet soubory.

2.3 Raspberry Pi

Raspberry Pi je malý jednodeskový počítač, který byl vyvinut britskou charitativní organizací Raspberry Pi Foundation. Toto zařízení bylo především zaměřené pro výuku informatiky na školách. Jelikož ale toto zařízení bylo velmi levné a malé, tak začalo být populární i jinde. Jako příklady lze uvést připojení k TV (efektivně z ní dělá chytrou televizi), či velmi levné PC pro základní použití. Raspberry Pi je uzpůsobené pro běh na Linuxových systémech. Nejpopulárnější je Raspbian, který je založen na operačním systému Debian. Pokud Debian není nic pro vás, druhou nejpopulárnější variantou je Pidora, která je podobná operačnímu systému Fedora. Všechny dosavadní modely byly založeny na SoC (System on a chip) od firmy Broadcom s integrovanou CPU (Central processing unit) a GPU (Graphics processing unit). Nezbytnou součástí každého Raspberry Pi je také SD karta, která pro něj funguje jako hlavní úložiště dat včetně operačního systému. [12]

Pro programování na Raspberry Pi jsou doporučeny zejména tři platformy: Python, Wolfram Mathematica a Scratch. Raspberry Pi však podporuje nespočet dalších jazyků. Nicméně je nutno poukázat, že vzhledem k omezenému výkonu zařízení je vhodné používat takový jazyk, který není příliš paměťově a výpočetně náročný. [13]

V tabulce 2.2 najdete srovnání těch modelů Raspberry Pi, které se v současné době prodávají. V jednom sloupci můžete vidět „RAM“, to je zkratkou pro random-access memory. V rámci bezdrátové sítě je zkratkou „BT“ míněna technologie Bluetooth. Ve sloupci s označením „Spotřeba“ je udána typická spotřeba bez připojených periferií.

2.3.1 Podobnosti s klasickými počítači

Raspberry Pi se od klasických počítačů příliš neliší. Samozřejmě je pravdou, že má menší velikost a tudíž menší výkon, ale principiálně funguje stejně. Proto by právě jakákoli Bluetooth knihovna vyvinutá na Raspberry Pi měla teoreticky fungovat i na stolních počítačích či noteboocích, které běží na Linuxu. Je ovšem samozřejmé, že kromě menšího výkonu má Raspberry Pi

[4]Dá se to ale samozřejmě změnit.

[5]Oficiální zdroje toto neudávají, hodnota je tudíž aproximovaná.

Tabulka 2.2: Porovnání modelů Raspberry Pi. [14] [15]

Model	CPU	RAM	Bezdrátová síť	Spotřeba
Zero	1GHz single-core	512 MB	Není	100mA
Zero W	1GHz single-core	512 MB	Wi-Fi a BT 4.1	150mA
1 Model A+	700MHz single-core	512 MB	Není	180mA
1 Model B+	700MHz single-core	512 MB	Není	330mA
2 Model B	900MHz 32-bit quad-core	1 GB	Není	350mA
3 Model B	1,2GHz 64-bit quad-core	1 GB	Wi-Fi a BT 4.1	400mA
3 Model B+	1,4GHz 64-bit quad-core	1 GB	Wi-Fi a BT 4.2	550mA[5]

obecně také menší úložiště a horší konektivitu. Vyjímkou zde ale jsou GPIO (General-purpose input/output) piny, které jsou právě největší výhodou tohoto zařízení co se týče konektivity. Na GPIO piny totiž lze připojit velké množství příslušenství počínaje od LED (Light-emitting diode) až po všemožné senzory. Lze tak sbírat všemožná data, která může Raspberry Pi zpracovat a nějak na ně reagovat. Jako příklad by se dala uvést situace, kdy v domě máte světelné senzory připojené na Raspberry Pi a chcete, aby se zatáhly závěsy právě když bude na tyto strategicky umístěné senzory svítit určité množství světla. [16]

Kapitola 3

Analýza a návrh

3.1 Současný stav řešení problematiky

Ze strany platformy pro Android je mnoho projektů, které vytváří knihovnu na jednoduché používání Bluetooth. Většinou mají ale omezenou funkcionalitu. Například tak, že umí komunikovat pouze jako klient, či zvládají spojení pouze dvou zařízení. Příkladem může být knihovna Bluetooth-Library, více zde [17]. Tato knihovna má však to omezení, že umí číst z Bluetooth socketu pouze data, která končí „newline“. Ještě k tomu zde ani není podpora pro připojení více zařízení.

Co se týče strany Raspberry Pi, tak tam jsem žádnou knihovnu na komunikaci přes Bluetooth nenašel. Jelikož je na Raspberry Pi primárně doporučen jazyk Python, tak jsem hledal knihovny hlavně v něm.

Pokud se podíváme mimo Bluetooth, tak na protokolu TCP/IP (Transmission control protocol/Internet protocol) je postaven protokol MQTT (Message queuing telemetry transport), který se používá právě na řešení této problematiky. Funguje na bázi jednoho centrálního zařízení (brokeru), které se stará o výměnu dat v síti. Je zde využit princip přihlašování zařízení k určitým tématům, které pak následně od brokera dostávají, takže dostanou pouze to, co chtějí. [18]

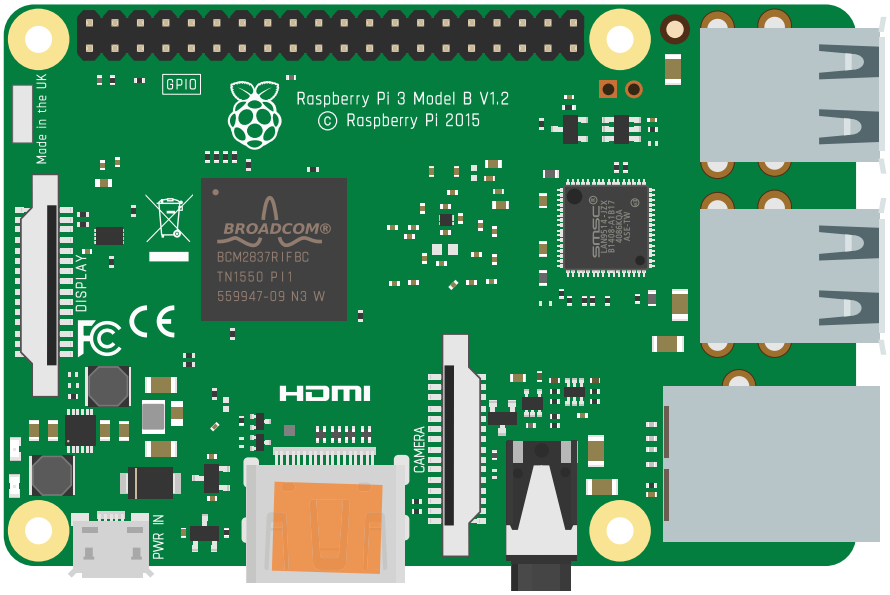
3.2 Moje řešení

Díky mým knihovnám by měla být komunikace přes Bluetooth co nejsnadnější. Budu se tedy snažit, aby pochopení a užívání mé knihovny bylo jednoduché. Myslím si, že je také důležité zanechat uživateli svobodu v tom, jak chce data přenášet. Proto je nejlepším způsobem nevyžadovat po uživateli žádný konkrétní formát, ale přenášet pole bytů. Jak už s tím polem bytů uživatel naloží je na něm. Tímto způsobem se dá poslat prakticky cokoli, uživatel si dokonce může vymyslet svůj vlastní formát. Na Androidu je master zařízení, takže by bylo dobré, aby bylo schopno držet více připojení najednou. Raspberry Pi

jako slave drží pouze jedno připojení. Vzhledem k tomu, že Android zařízení bude data hlavně číst, tak je nutné vymyslet efektivní způsob, jak to dělat. Nabízí se vytvoření jednoho dedikovaného vlákna, které bude stále číst data ze socketu. Raspberry Pi bude data hlavně posílat, takže tam není nutno čtení nijak složitě řešit.

3.3 Výběr Bluetooth

K dispozici mám telefon Samsung Galaxy S7, který disponuje Bluetooth 4.2. Na druhé straně komunikace plánuji používat Raspberry Pi 3 Model B s Bluetooth 4.1 (obrázek 3.1) a pro testovací účely také notebook Toshiba Tecra Z50-A, který má Bluetooth 4.0. Vzhledem k těmto okolnostem musím bohužel konstatovat, že Bluetooth 5 nebudu moci použít, jelikož bych neměl prostředky, jak knihovny otestovat. Tudíž mi zbývá rozhodnout se mezi BLE a Bluetooth classic[6].



Obrázek 3.1: Ilustrace Raspberry Pi 3 Model B. Obrázek z [19].

Výhody a nevýhody BLE oproti Bluetooth classic:

- **Výhody**
  Menší spotřeba energie (1-50% oproti klasickému Bluetooth).
  Teoreticky vyšší počet aktivních připojení, přičemž klasické Bluetooth má maximum 7 aktivně připojených zařízení.
  Menší latence.

[6]Bluetooth HS nepočítám, jelikož hlavní část komunikace zde probíhá přes Wi-Fi.

- **Nevýhody**
  Je novější, takže nižší podpora ze stran zařízení. Klasické Bluetooth je oproti tomu podporované prakticky na každém zařízení s technologií Bluetooth.
  Menší přenosová rychlost. Bluetooth classic disponuje až téměř osminásobnou rychlostí.
  Menší dosah signálu.

3.3.1 Zhodnocení

Jelikož jsou chytrý telefon i Raspberry Pi zařízení, kde se děje spousta mnohem energeticky náročnějších událostí[7], tak můžeme říci, že pro naše účely není spotřebový rozdíl příliš relevantní. Více k energetické spotřebě Raspberry Pi zde [20]. Podobně se můžeme podívat i na latenci, kde je rozdíl opravdu malý a jestli data přijdou byť o vteřinu[8] déle nás nezajímá. Vyšší počet současně připojených zařízení je samozřejmě vždy k užitku, ale je těžké představit si situaci, kdy by k jednomu telefonu bylo z nějakého důvodu připojeno sedm Raspberry Pi v jednu chvíli. Dále máme podporu zařízení s Bluetooth starším než 4.0. To je jedině dobrá vlastnost i přes to, že takových zařízení v dnešní době není mnoho. Když se podíváme na rychlost, tak ta pro nás hraje roli pouze tehdy, kdy bychom chtěli stále přesunovat velké množství dat. Představme si případ, kdy by k Android zařízení bylo připojeno sedm Raspberry Pi. Kdyby každé posílalo 5 kB dat za sekundu, tak již přenosová rychlost BLE nebude stačit. V takovýchto případech by Bluetooth classic získalo velkou převahu. Jako poslední musíme zhodnotit relevanci dosahu signálu, ten je pro nás důležitý, jelikož chceme mít možnost posílat data na co nejvyšší vzdálenosti.

3.3.2 Výsledek

Po zvážení všech pro a proti v minulé sekci jsem se rozhodl využít technologii Bluetooth classic. Dalším důvodem, který jsem ještě nezmínil je to, že tento typ Bluetooth je už starou technologií, takže si myslím, že bude vyladěnější a spolehlivější oproti BLE.

3.4 Výběr platformy na Android

Oficiálně doporučeným programem na vývoj aplikací je IDE Android Studio, rozhodl jsem se tedy programovat právě v něm. K automatizaci sestavování aplikace je vhodné používání Gradle, jelikož je v rámci IDE používán standardně. V zadání práce se také píše, že je její součástí vyvinutí demo aplikace. To znamená, že bude zároveň potřeba SDK, které je v rámci Android Studia také standardem.

[7] Příkladně režie OS či zapnuté Wi-Fi.
[8] I když to ani ta vteřina nebude.

3.4.1 Programovací jazyk

Co se týče programovacího jazyku, tak zde jsem přemýšlel nad Javou a Kotlinem. Kotlin je nově podporovaným programovacím jazykem v rámci programování na Androidu. Prý je také jednodušší na naučení než Java. S Kotlinem nemám naprosto žádné zkušenosti a navíc je nově podporovaným jazykem v rámci vývoje aplikací na Android, takže pokud s něčím máte problém a nevíte jak na to, tak pro vás bude těžší hledat řešení. Co se týče Javy, tak ta je nejpopulárnějším programovacím jazykem vývoje na Androidu. Je k němu rovněž největší podpora a nejvíce návodů. Dalším bodem pro Javu je fakt, že jsem v Javě už pár projektů programoval, takže s ní mám jisté zkušenosti. Rozhodl jsem se proto vyvíjet knihovnu a demo aplikaci v Javě.

3.5 Výběr platformy na Raspberry Pi

Raspberry Pi je zařízení, které má relativně vůči stolním počítačům, notebookům a dokonce i velké části chytrých telefonů slabý výkon. Kvůli tomuto faktu je třeba vybrat takovou platformu na Raspberry Pi, která umí efektivně hospodařit s omezeným výpočetním výkonem a malou pamětí. V úvahu zde připadá zejména Python a C/C++. Oba tyto jazyky na Linuxu používají „Bluez“. BlueZ je totiž oficiální Linuxový Bluetooth stack, což je v podstatě sada ovladačů a rozhraní. Tento software zpřístupňuje na Linuxu služby, respektive protokoly technologie Bluetooth. [21]

3.5.1 C/C++

Právě C a C++ jsou výborné jazyky, se kterými je možno naprogramovat prakticky cokoli jakýmkoli způsobem, pokud člověk ví co dělá. To je také jeho nevýhodou, jelikož pokud programátor přesně nechápe, co jeho kód dělá, tak zde může udělat a velice pravděpodobně udělá velké množství chyb. Čtenáře nebudu zdržovat hlubším popisem těchto jazyků, jelikož jsou pro každého, kdo se pohybuje v IT naprostým základem. Pokud si ale chcete přečíst o C++, ideální volbou bude kniha od autora C++ (kniha [22]) Během programování v C/C++ to typicky dopadá tak, že když nezkušený člověk programuje nějaký složitější program v těchto jazycích, tak mu dlouho nebude fungovat. Když funkčnost programu konečně vyřeší, tak se jistě objeví minimálně pár chyb v kódu, které mohou ohrozit dlouhodobou funkčnost a bezpečnost tohoto programu.

Na druhou stranu je dobré podotknout, že mám s používáním obou jazyků už nějaké zkušenosti, a to minimálně z předmětů BI-PA1, BI-PA2, BI-AG1 a BI-OSY. Bohužel také musím konstatovat, že co se týče síťové komunikace jsem v těchto jazycích nic nepsal až na jedinou výjimku, tou byla domácí úloha na BI-PSI.

3.5.2 Python

Python je objektově orientovaný vysokoúrovňový jazyk. Syntakticky je to jednoduchý jazyk. Právě proto je snadné se ho naučit a zároveň je lehké kód číst a porozumět mu, což ulehčuje ladění programu. Narozdíl od jazyků C a C++ je tento jazyk interpretovaný, což znamená, že se nekompiluje. V současnosti je to také jeden z nejpopulárnějších programovacích jazyků. [23]

Já osobně mám s Pythonem relativně menší zkušenosti a před touto prací jsem v něm de facto nic nepsal. Tento jazyk mi ale přijde zajímavý a myslím si, že má smysl se ho naučit. V Pythonu existuje modul PyBluez [24], což je modul psaný v C, který poskytuje přístup k Bluetooth službám v objektově orientovaném stylu. Můžeme si ho představit jako takovou nadstavbu pro BlueZ, která je navržená specificky pro Python. Díky tomuto softwaru je zprostředkování komunikace přes Bluetooth snadné. Dle mého názoru snazší než v C/C++.

3.5.3 Výsledek

Nejprve jsem se rozhodl vyzkoušet pro účely tohoto projektu jazyk C. Naneštěstí se mi ale stalo to, čeho jsem se bál. Navázání Bluetooth spojení je zde totiž těžší, než se na první pohled zdá. Příliš mi to nefungovalo a nemohl jsem nijak ručit za plnou funkčnost či bezpečnost programu. Tudíž po přihlédnutí k popularitě Pythonu a jeho doporučení na oficiálních stránkách Raspberry Pi jsem se po pečlivém rozmýšlení nakonec rozhodl pro Python.

Kapitola 4

Realizace

Byly napsány dvě knihovny. Jedna v Javě na Android a druhá v Pythonu na Raspberry Pi (nebo jiný Linuxový počítač). Ke každé knihovně jsem také napsal vzorovou aplikaci/skript. Tyto vzory ukazují způsob, jak mohou být knihovny použity. Tato dema jsou jednoduchá, jelikož pouze demonstrují funkce knihoven a mají být snadná na pochopení uživatelem.

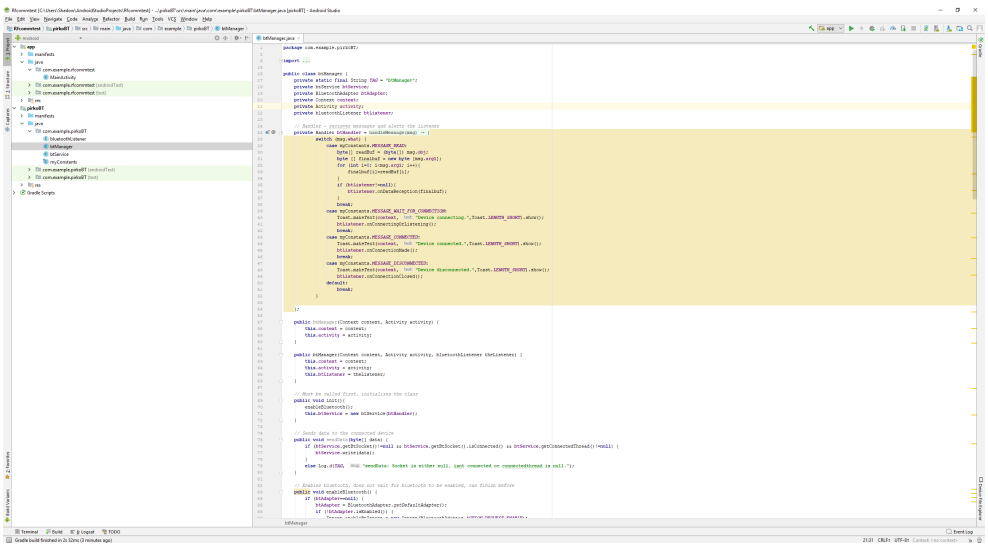
Komunikace přes Bluetooth pomocí mých knihoven je vedena skrze použití protokolu RFCOMM. Knihovny jsou napsány pro populární programovací jazyky na daných platformách a to znamená, že uživatelský potenciál je zde velký. Komunikace s užitím mých knihoven byla navržena tak, aby:

- Byla obousměrná.
- Mohla být iniciována z obou stran. To znamená, že libovolné zařízení může být klientem nebo serverem.
- Master zařízením mohlo být pouze zařízení s Androidem.
- Bylo možné k master zařízení připojit více slave zařízení.
- Byla jednoduchá na použití.
- Omezovala uživatele co nejméně. Knihovny jsou navrženy tak, aby komunikovaly data ve formě pole bytů. To znamená, že uživatel si může dle libosti navrhnout, v jakém formátu chce data posílat. Soubory jsou totiž v esenci také pouze pole bytů.
- Měla co nejvyšší dosah.
- Počítala s posíláním velkého množství dat ze strany Raspberry Pi.

4.1 Android strana

Knihovna i demonstrační aplikace byly vyvinuty v IDE Android Studio 3.1 (obrázek 4.1) v programovacím jazyce Java. Aplikace byla vyvinuta specificky pro chytré telefony s Androidem. Knihovnu je však možno použít i u jiných Android zařízení, například televizí. K sestavení programu byly použity nejnovější SDK nástroje a Gradle. Nejprve byla samozřejmě vyvinuta samotná knihovna, která byla napsána metodou shora dolů. To znamená, že vývoj začal implementací vyšších vrstev knihovny a postupovalo se postupně dolu. Knihovnu[9] je možno používat za pomocí více vláken. Je totiž odladěná tak, aby byla thread-safe (vláknově bezpečná). Je rovněž možno mít k Android zařízení připojeno až sedm Raspberry Pi. Bohužel jsem neměl tolik zařízení k dispozici, abych to důkladně otestoval. Nicméně testované bylo připojení a posílání dat od dvou zařízení zároveň a to fungovalo.

Knihovna byla testována na Androidu 7.0 (API 24). Měla by dobře fungovat na starší verze Androidu až k verzi 4.4 (API 19), kde přestává fungovat metoda párování. Používám zde totiž metodu *createbond()*, která je součástí třídy *BluetoothDevice*. Ta byla totiž přidána až v API 19. [25]



Obrázek 4.1: IDE Android Studio 3.1, na kterém probíhal vývoj knihovny a demo aplikace pro Android.

Vstupní třídou pro uživatele je zde třída *btManager*, která spravuje právě jednu Bluetooth komunikaci. Chce-li tedy uživatel navázat více spojení, musí mít aktivních více instancí této třídy. Po vytvoření instance této třídy je k jejímu použití nutné zavolat metodu „btManager.init()“, která inicializuje třídu. V tu chvíli je vše připraveno k navázání komunikace. Po ukončení ko-

[9] Je myšleno především posílání dat skrze knihovnu.

munikace je nutno zavolat metodu „btManager.end()“, která po třídě uklidí. Níže je ukázán vzor inicializace a zrušení třídy *btManager* ve vzorové aktivitě. Konkrétní popis metod *bluetoothListener* je v subsekci 5.1.2.

```
// ExampleActivity.java
private btManager btmanager;
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   btmanager = new btManager(this, this, new bluetoothListener() {
      @Override
      public void onDataReception(byte [] bytes){
      // zpracovani prichozich dat
      }
      @Override
      public void onConnectionMade() {
      // akce po navazani spojeni
      }
      @Override
      public void onConnectionClosed() {
      // akce po preruseni spojeni
      }
      @Override
      public void onConnectingOrListening() {
      // akce po zahajeni naslouchani ci pripojovani na socketu
      }
      @Override
      public void onDiscoveryStarted() {
      // akce po zahajeni objevovani zarizeni
      }
      @Override
      public void onDiscoveryFinished() {
      // akce po dokonceni objevovani zarizeni
      }
   });
   btmanager.init();
   // odtud je mozno tridu pouzivat
}

@Override
protected void onDestroy() {
   super.onDestroy();
   btmanager.end();
}
```

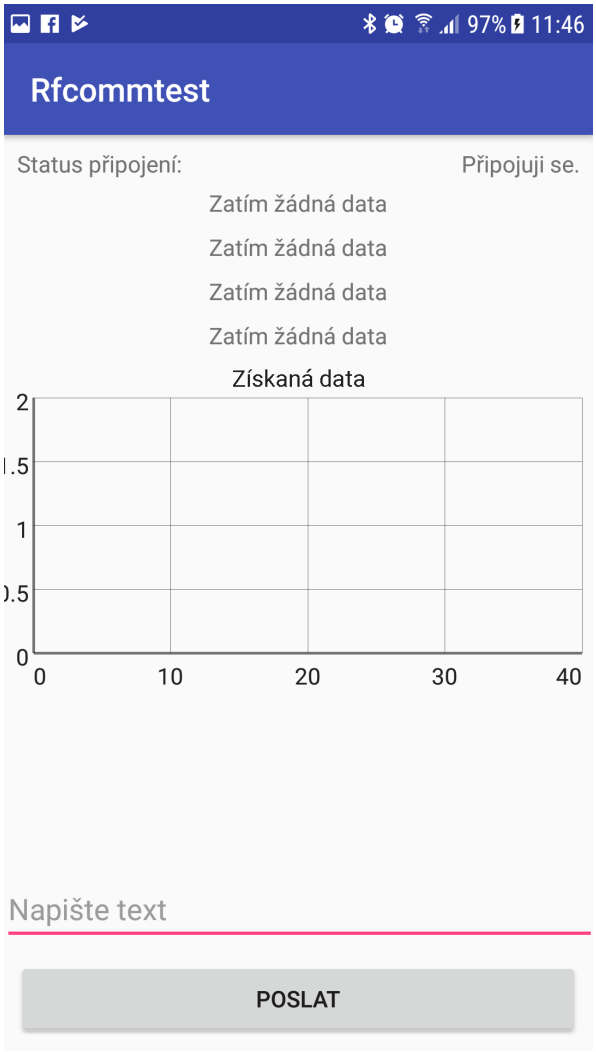
Knihovna má následující funkce:

- Zapne nebo vypne Bluetooth se svolením či bez svolení uživatele[10].
- Zviditelní zařízení pro ostatní Bluetooth zařízení na libovolnou dobu.
- Najde viditelná zařízení používající Bluetooth v okolí.
- Přeruší hledání viditelných Bluetooth zařízení v okolí.
- Najde spárovaná zařízení dle daného jména či MAC adresy.
- Spáruje zařízení.
- Zkontroluje, zda je dané zařízení spárované.
- Naváže Bluetooth spojení jako server či klient.
- Pošle či přijme data skrze Bluetooth.
- Přeruší spojení.
- Zjistí stav připojení.
- Zjistí informace o připojeném zařízení.
- Nastaví uživatelem definovaný Listener (naslouchač). To je třída, které se v rozhodujících momentech připojení volají uživatelem definované metody.

4.1.1 Demo aplikace

Demonstrační aplikace je řešena pomocí jedné aktivity, která začne přijímáním připojení. Když se nějaké zařízení připojí, tak od něj přijímá data. Tato data vypisuje do vrchní části obrazovky nad graf. Vypíšou se maximálně poslední čtyři. Po příchodu pátého čísla se zobrazená data smažou a páté číslo se vypíše jako první. V pravém rohu vidíme text, který určuje v jakém stavu se připojení v tento moment nachází. V již zmíněném grafu jsou ilustrovaná příchozí data. Zde je x souřadnicí pořadí, v jakém číslo přišlo a y souřadnicí samotné číslo. Tento graf se aktualizuje vždy, když se datová tabulka nad grafem zaplní a přijde další číslo. Ve spodní části obrazovky je pole, kam je možno napsat text, který se pošle přes Bluetooth druhému zařízení po stisku tlačítka „POSLAT“. Aplikace je řešena tak, aby neustále přijímala data. Konec nastane pouze tehdy, kdy druhé zařízení ukončí komunikaci. K tomu je uzpůsobený Python skript na druhé straně (straně klienta), který čte příchozí data a pokud dostane příkaz „Stop“, tak komunikaci přeruší. Formát je v této demo aplikaci daný tak, že každá zpráva je oddělena znakem „;“. Toto se ale řeší vnitřně a koncový uživatel aplikace o tom vůbec neví. K vytvoření grafu bylo využito knihovny *GraphView*. Více na [26].

[10] V případě vypnutí Bluetooth se uživatele neptá.



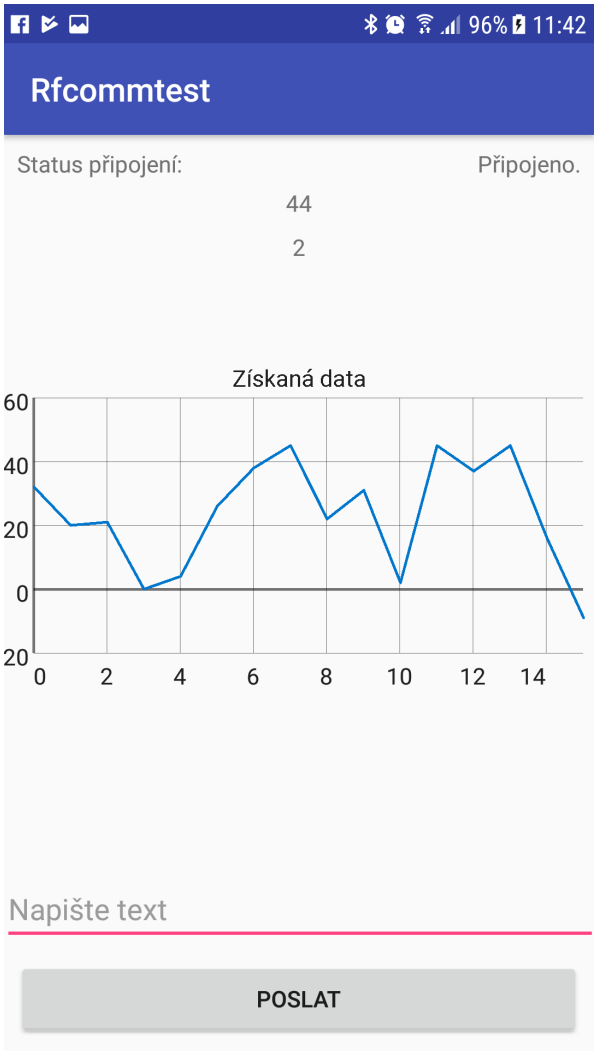
Obrázek 4.2: Počáteční stav aplikace.

Na obrázku 4.2 můžeme vidět počáteční stav aplikace. Zařízení čeká na připojení a nejsou zde žádná data. Po připojení je zde vedena krátká komunikace, kdy klient posílá čísla[11]. Na obrázku 4.3 můžeme vidět aplikaci v průběhu komunikace.

4.2 Raspberry Pi (Linux) strana

Knihovna byla vyvinuta v programovacím jazyce Python, konkrétně v softwaru Komodo Edit [27]. Knihovna byla napsána a testována v Pythonu 2.

[11]Tato čísla mohou být například z teploměru na Raspberry Pi.



Obrázek 4.3: Stav aplikace po přenosu několika dat.

Měla by ale fungovat i v Pythonu 3, nicméně nebyla v něm testována. Dále je nutno podotknout, že k plnému fungování knihovny na Raspberry Pi jsou potřeba následující balíčky:

- Python 2.3 nebo novější
- Python-dev
- BlueZ a BlueZ-tools
- PyBluez
- Rfkill

Jako vstupní bod zde figuruje třída *btManager*, kterou není třeba inicializovat ani ukončovat žádnou její metodou. O to se postará konstruktor a destruktor. Jelikož mi přišlo nevhodné zapínat Bluetooth pokaždé, kdy je tato třída inicializována, tak je třeba se před použitím metod třídy ujistit, zda je zapnuté Bluetooth. Níže můžete vidět jeden z možných postupů. V tomto příkladu si chci deklarovat a iniciovat na začátku všechny proměnné, aby v tom byl pořádek.

```
# Example.py
from pirkoBT import *      # importuje knihovnu
btman = btManager()
# zde muze byt nejaky kod, ktery nesouvisi s btManager
# ...
# v tuto chvili uz chci btman pouzit
enableBluetooth()
# tady se uz mohu pripojit a komunikovat
```

Knihovna umí následující:

- Zapne či vypne Bluetooth.
- Objeví a vypíše zařízení v okolí.
- Najde adresu zařízení podle jména.
- Objeví a vypíše Bluetooth služby v okolí.
- Naváže Bluetooth spojení jako server či klient.
- Pošle či přijme data skrze Bluetooth.
- Přeruší spojení.
- Zjistí stav připojení.
- Nastaví timeout Bluetooth socketu.
- Vypíše spárovaná zařízení.
- Zkontroluje, zda je dané zařízení spárované.

4.2.1 Demo skript

Demonstrační skript na Pythonu funguje jako klient demo aplikace na Androidu. Nejprve najde podle jména chytrý telefon a následně se k němu připojí. Dalším krokem je nastavení timeoutu socketu na jednu vteřinu. Tělo skriptu potom spočívá v tom, že zde běží nekonečná smyčka. V této smyčce se čte z Bluetooth socketu a za podmínky, že se přečte „Stop“ skript skončí. Pokud se tak ale nestane, tak se pošle nějaké vygenerované číslo druhému zařízení.

Tady konkrétně jde o simulaci čtení teploty z teploměru připojeného skrze GPIO piny. Je to řešeno tímto způsobem, aby bylo možno demonstrovat fungování bez připojování jakéhokoli příslušenství.

Spouští se následovně:

```
python main.py
```

Pokud vám toto nebude fungovat, doporučuji se podívat na následující subsekci, případně před příkaz napsat „sudo“.

4.2.2 Problém s BlueZ 5

Zde je důležité poukázat, že v nejnovější verzi BlueZ (verze 5) je jeden pro nás důležitý bug. Prakticky jde o to, že některé služby v rámci BlueZ už nefungují. BlueZ 5 totiž opustilo C interface */var/run/sdp*, který potřebujeme. Naštěstí je tento problém relativně snadný na vyřešení. Tato subsekce vás provede řešením tohoto problému.

Nejprve editujeme soubor */etc/systemd/system/dbus-org.bluez.service* a změníme v něm řádek

```
ExecStart=/usr/lib/bluetooth/bluetoothd
```

na následující:

```
ExecStart=/usr/lib/bluetooth/bluetoothd --compat
```

Dále musíme provést následující dva příkazy:

```
systemctl daemon-reload
systemctl restart bluetooth
```

Nyní by mělo SDP fungovat. Problémem je, že po každé aktualizaci BlueZ se tato změna přepíše původní hodnotou, takže s tím je třeba počítat. Ještě zde je možnost, že jsou u */var/run/sdp* nastavená nesprávná přístupová práva. To vám vadí, pokud vše nepouštíte jako root (správce). Řešení je jednoduché:

```
chmod 777 /var/run/sdp
```

Teď by mělo SDP fungovat pro všechny uživatele.

4.3 Bezpečnost

Jelikož je zde použito Bluetooth classic, tak budu vypisovat konkrétní informace o bezpečnosti tohoto protokolu.

Zabezpečení komunikace je jednak pomocí toho, že když zařízení na jakékoli straně přijímá připojení, tak se nejprve podívá, zda je připojované zařízení spárované. Pokud tomu tak není, komunikaci zavře. Toto je moje opatření, které

v protokolu není standardně řešeno, ten totiž umožňuje i nešifrované spojení s nespárovanými zařízeními. Další bezpečnostní metoda spočívá v procesu párování, kdy se vytvoří klíč na kódování a dekódování komunikace. Proces párování a s ním i tvorba tohoto klíče se liší dle verze Bluetooth. Níže jsou napsány podrobnosti k různým verzím.

- **Verze 2.0 a dříve**
  Zde je klíč generován ze sdíleného PIN kódu, masterova Bluetooth časovače (clock), masterovy Bluetooth adresy a náhodného vyměněného čísla. Pokud infiltrátor tyto informace má, může si spočítat klíč. Probíhá zde informační výměna mezi masterem a slavem, kdy si vymění náhodné číslo a kombinovaný klíč. Díky tomu jsou schopni si vytvořit společný klíč. [28]

- **Verze 2.1 a dále**
  Tady je využito SSP (Secure simple pairing), které využívá kryptografie eliptických křivek. Společný klíč se vytváří Diffieho–Hellmanovou výměnou klíčů (klíčem je 192-bit číslo). Každé zařízení má privátní a veřejný klíč, kde veřejný je komunikován protějšku. Tímto způsobem je zajištěno, že i když infiltrátor tyto klíče zachytí, tak mu to k ničemu není. [29]

Více o bezpečnosti Bluetooth ve publikaci NIST (National Institute of Standards and Technology) zde [30].

4.4 Využití

Využití mých knihoven je velmi široké. Počínaje jednoduchými aplikacemi, jako například vyvinutí chatovací aplikace přes Bluetooth nebo posílání teplot z teploměrů. Je taktéž možné na nich postavit komplikovanější systémy. Představme si, že bychom chtěli zpracovávat nějaká velká data na několika Raspberry Pi. Data jsou rozdělitelná na menší části, které se dají zpracovat samostatně, tudíž využijeme výpočetního výkonu více Raspberry Pi k jejich zpracování. Tato data následně posíláme z Raspberry Pi do Android zařízení, které je spojuje dohromady. Tímto způsobem by šly řešit výpočetně složité problémy rozložením výpočtů na vícero zařízení.

Kapitola 5

Dokumentace ke knihovnám

V této kapitole leží popis těch částí knihovny, které se uživatele týkají. To znamená popis veřejných metod tříd, přidružených funkcí nebo pro uživatele relevantních konstant.

5.1 Android knihovna

V této Java knihovně jsou čtyři důležité třídy, níže následuje jejich popis.

5.1.1 Třída btManager

Hlavní třídou je třída *btManager*. Slouží jako vstupní třídou pro programátora. Její veřejné metody jsou následující:

- **btManager(Context context, Activity activity)**
  Konstruktor třídy. Prvním argumentem je kontext aktivity a druhým samotná aktivita. V praxi oba dva argumenty bývají *this*, když se tento konstruktor volá v aktivitě.
- **btManager(Context context, Activity activity, bluetoothListener theListener)**
  Konstruktor podobný konstruktoru výše. Jediným rozdílem je další argument, kterým je *theListener* (naslouchač). V tomto naslouchači si uživatel nastaví, co se má dít při určitých událostech. Návod k němu najdete v sekci 4.1.
- **init()**
  Inicializuje třídu a také zapne Bluetooth. Tato iniciační metoda se neptá uživatele, jestli smí zapnout Bluetooth. Pokud se uživatele zeptat chcete, použijte metodu níže.
- **init(boolean askForBt)**
  Alternativní inicializační metoda. Zde argument *askForBt* udává, zda se

budeme ptát uživatele na spuštění Bluetooth. Pokud je argument *true*, uživatele se ptáme, pokud je *false*, neptáme se.

- **sendData(byte [] data)**
  Pokud je navázaná komunikace přes Bluetooth tak pošle data daná argumentem druhému zařízení. Argument *data* je pole bytů, to znamená, že pokud chcete poslat cokoli jiného, tak je nutné to převést právě na pole bytů.

- **enableBluetooth()**
  Ukáže uživateli dialog, který si vyžádá spuštění Bluetooth. Uživatel může přijmout či zamítnout. Nečeká na uživatele, takže se ukončí prakticky vždy dříve, než uživatel stačí dialogem projít. S tím je nutno počítat a přizpůsobit se tomu. V mezidobí je možno dělat něco jiného a čas od času sledovat, jestli je Bluetooth zapnuté. To se dá dělat například sledováním *BluetoothAdapter.isEnabled()*. Rozumný způsob ale je vytvoření broadcast recieveru, který vás upozorní až se Bluetooth zapne.

- **forceEnableBluetooth()**
  Nikoho se na nic neptá a zapne Bluetooth.

- **forceDisableBluetooth()**
  Nikoho se na nic neptá a vypne Bluetooth.

- **makeDiscoverable(int duration)**
  Zviditelní zařízení pro ostatní v rámci Bluetooth. Argument *duration* určuje dobu v sekundách, po kterou bude zařízení viditelné.

- **startDiscovery()**
  Zahájí hledání Bluetooth zařízení v okolí. Toto hledání trvá přibližně 12 sekund. Dá se zkrátit zavoláním metody *cancelDiscovery()*.

- **cancelDiscovery()**
  Zruší hledání Bluetooth zařízení v okolí.

- **findDeviceByName(String theName)**
  Prohledá seznam spárovaných zařízení a pokusí se najít zařízení dané argumentem *theName*, které označuje jeho jméno. Pokud je zařízení nalezeno, metoda ho vrací ve formě objektu *BluetoothDevice*. Pokud zařízení nalezeno není, je vrácen *null*.

- **findDeviceByName(String theMAC)**
  Funguje identicky jako metoda výše s jedinou výjimkou. Místo jména zařízení je zde argument *theMac*, který označuje MAC adresu zařízení.

- **pairDevice(BluetoothDevice device)**
  Spáruje toto a zařízení dané argumentem *device*. Zde je stejně jako

v metodě *enableBluetooth()* potřeba uživatelské interakce. Metoda se ukončuje okamžitě po zahájení procesu párování. Je tedy potřeba s tím počítat a zařídit se dle toho. Doporučoval bych zde sledovat ukončení párování s pomocí broadcast recieveru.

- **checkIfPaired(BluetoothDevice device)**
  Projde seznam spárovaných zařízení a zkontroluje, zda zařízení dané argumentem *device* se rovná některému ze spárovaných. Metoda vrací *true* za podmínky, že *device* je spárované. Pokud zařízení spárované není, je vráceno *false*.

- **createBtServer()**
  Zastaví všechna běžící vlákna a spustí nové vlákno *AcceptThread*. Co toto vlákno dělá se můžete dočíst níže v subsekci týkající se třídy *btService*.

- **connectToDevice(BluetoothDevice theDevice)**
  Zastaví všechna běžící vlákna a spustí nové vlákno *ConnectThread*, které se připojí k zařízení daném v argumentu *theDevice*. Pokud vás zajímají konkrétnější informace, tak ty se dají zjistit níže v subsekci týkající se třídy *btService*.

- **connectToDevice(String MACaddr)**
  Funguje stejně jako metoda výše. Jedinou výjimkou je to, že je zařízení tentokrát určeno MAC adresou danou argumentem *MACaddr*.

- **disconnect()**
  Ukončí všechna vlákna. To znamená, že pokud má třída čekat na spojení, připojovat se nebo být připojena, tak po zavolání této metody tomu již tak nebude.

- **end()**
  Uklidí po třídě. Je třeba volat vždy na konci používání třídy.

- **setListener(bluetoothListener btlistener)**
  Nastaví třídní naslouchač na ten, který je daný argumentem *btlistener*. Je vhodné použít v případě, že si přejete změnit chování programu při událostech jako je například příjem dat přes Bluetooth.

- **getConnectedDevice()**
  Vrací připojené zařízení ve formě objektu *BluetoothDevice*. Pokud není aktivní žádné spojení vrací *null*.

- **getConnectionState()**
  Vrací aktuální status připojení ve formě *int*. Co dané číslo znamená je popsané ve třídě *myConstants* v rámci knihovny.

- **getBtAdapter()**
  Vrací objekt ve formě *BluetoothAdapter*. Tento objekt reprezentuje standardní lokální Bluetooth adaptér.

5.1.2 Třída bluetoothListener

Abstraktní třída *bluetoothListener* slouží jako pomocnou třídou ke třídě *btManager*. Definuje několik metod, které jsou volány v rozhodujících okamžicích komunikace přes Bluetooth. Je doporučeno, aby si uživatel implementoval ty metody, které jsou pro něj důležité. To se provádí v rámci konkrétní instance uvnitř *btManager*. Níže následuje popis metod.

- **onConnectionMade()**
  Zavolá se po spuštění vlákna, které se stará o správu navázaného připojení. To znamená, že je aktuální zařízení již připojeno a připraveno posílat či číst data.

- **onConnectionClosed()**
  Zavolá se ve dvou případech. Prvním je případ, kdy je připojení k protějšímu zařízení ztraceno. Druhým je zase případ, kdy je připojení přerušeno z našeho konce, například zrušením vlákna starajícího se o připojení.

- **onConnectingOrListening()**
  Tato metoda se zavolá také ve dvou případech. Prvním je, když se spustí vlákno zodpovědné za přijímání připojení. Druhým je, když se naopak spustí vlákno zodpovědné za připojení k danému zařízení. Primární účel této metody je, alespoň pro mě, možnost sdělit uživateli stav připojení.

- **onDataReception(byte [] bytes)**
  Když se přečtou nějaká příchozí data z Bluetooth socketu, zavolá se právě tato metoda. V argumentu *bytes* jsou právě tato data ve formě pole bytů. Těchto bytů bude nejvíce 1024. To proto, že ze socketu se čte do bufferu, který má právě tuto velikost.

- **onDiscoveryStarted()**
  Zavolá se po spuštění objevování okolních Bluetooth zařízení.

- **onDiscoveryFinished()**
  Zavolá se po zrušení objevování okolních Bluetooth zařízení.

5.1.3 Třída btService

Privátní instance třídy *btService* je použita v rámci třídy *btManager*. Funguje zde jako o úroveň nižší třída, ve které běží různá vlákna a spravuje se zde připojení. Řeší se tu ty věci, se kterými nechceme programátory otravovat. Například vybírání UUID nebo spravování naslouchacího, připojovaného

a připojeného vlákna. Na tuto třídu by se vůbec nemělo sahat, takže sem kompletní dokumentaci psát nemá smysl. Za zmínku ale stojí tři typy vláken a jedna s nimi související metoda.

- **ConnectThread (vlákno)**
  V argumentu konstruktoru dostane zařízení ve formě *BluetoothDevice*. Dále se zde vytvoří *BluetoothSocket*, který se připraví na navázání bezpečné komunikace přes Bluetooth. Je zde využito UUID k identifikaci služby na druhé straně pomocí protokolu SDP. Když je hotovo, pokusí se přes tento socket navázat spojení. Pokud všechny tyto kroky proběhly hladce, tak se zavolá metoda *manageConnectedSocket* a vlákno se ukončí.

- **AcceptThread (vlákno)**
  Zde se pro změnu vytvoří *BluetoothServerSocket*. Ten se připraví na naslouchání bezpečné RFCOMM komunikace. Dále se zde v nekonečném cyklu volá metoda zmíněného server socketu *accept()*. Pokud se některé zařízení připojí, zavolá se stejně jako výše metoda *manageConnectedSocket* a vlákno se ukončí.

- **ConnectedThread (vlákno)**
  V argumentu konstruktoru dostane socket ve formě *BluetoothSocket*. Z něj se pak získají vstupní a výstupní proudy (input a output streamy). Následně začne běh nekonečného cyklu, kde se čtou ze socketu data. Pokud nějaká data přijdou, jsou poslány skrze handler do třídy *btManager*, kde se zavolá příslušná metoda listeneru. Během průběhu tohoto cyklu lze také posílat data. K tomu je v rámci vlákna metoda *write*. Ta je ale uzpůsobena k tomu, aby bylo možno ji volat z vnějšku vlákna.

- **manageConnectedSocket(BluetoothSocket socket, BluetoothDevice device)**
  Tato metoda nejprve zruší accept a connect vlákna (pokud existují). Následně vytvoří a spustí *ConnectedThread* se socketem, který byl předán v argumentu *socket*.

5.1.4 Třída btConstants

Poslední třídou je třída *btConstants*. V této třídě jsou uloženy všemožné konstanty, které jsou použity v rámci knihovny. Jediné, co by uživatele knihovny mohlo zajímat, jsou číselná označení stavů připojení. Část kódu, která je určuje můžete vidět níže.

```
// stavy z btConstants.java
public static final int STATE_NONE = 0;
public static final int STATE_LISTENING = 1;
public static final int STATE_CONNECTING = 2;
```

```
public static final int STATE_CONNECTED = 3;
```

5.2 Raspberry Pi (Linux) knihovna

V této Python knihovně je použita pouze jedna třída a několik přidružených funkcí.

5.2.1 Třída btManager

Jedinou a hlavní třídou knihovny je třída *btManager*. Tato třída se chová jako zprostředkovatel připojení k dalšímu zařízení skrze Bluetooth. Její veřejné metody jsou následující:

- **createBtServer(self)**
  Funguje obdobně jako metoda Android knihovny stejného názvu. Poslouchá na socketu a zveřejní službu v protokolu SDP. Skončí ve chvíli, kdy se připojí nějaké zařízení.
- **connectToDevice(self, addr)**
  Také funguje podobně jako metoda Android knihovny pod stejným názvem. Pomocí SDP najde službu této knihovny na MAC adrese dané argumentem *addr* a k ní se následně připojí.
- **closeConnection(self)**
  Přeruší spojení k zařízení.
- **isConnected(self)**
  Zjistí, zda je připojené nějaké zařízení. Vrací *True* pokud připojené je, *False* pokud nikoli.
- **setSocketTimeout(self, tmout)**
  Nastaví timeout socketu připojení na číslo dané argumentem *tmout*. Toto číslo je dáno v sekundách. Dává smysl použít ve spojení s metodou *read()*, která se obyčejně zablokuje do té doby, než se na socketu objeví něco ke čtení. Pokud je nastavený timeout, pak metoda *read()* skončí až uplyne daný čas (pokud ovšem dříve nedostane data).
- **read(self)**
  Snaží se přečíst data ze socketu. Každý read přečte standardně maximálně 1024 bytů. Pokud nějaká data přečte, tak je vrátí. Když na socketu žádná data nejsou pak čeká, než se tam nějaká objeví. Toto se dá obejít nastavením timeoutu, který byl vysvětlen v metodě výše.
- **write(self, data)**
  Pokusí se poslat data daná argumentem *data*. Vrací *True* pokud se data podařilo poslat, *False* pokud nikoli.

5.2.2 Přidružené funkce

- **enableBluetooth()**
  Zapne Bluetooth.
- **disableBluetooth()**
  Vypne Bluetooth.
- **checkIfPaired(devName, devAddress)**
  Zkontroluje, zda zařízení dané argumenty *devName* a *devAddress* je spárované. Vrací *True* pokud jsou, *False* pokud nikoli.
- **findAddressByName(thename)**
  Prohledá spárovaná zařízení a když najde zařízení se jménem daném argumentem *thename*, tak vrátí jeho MAC adresu. Pokud zařízení s daným jménem nenalezne, vrací *None*.
- **listPairedDevices()**
  Vypíše seznam spárovaných zařízení.
- **listNearbyDevices(duration=8)**
  Zahájí průzkum okolních Bluetooth zařízení a pak je vypíše. Argument *duration* udává délku v jednotkách 1.28 sekundy. Délka 8 je doporučená a využita, pokud je funkce zavolána bez argumentu.
- **listServices(name=None, uuid=None, address=None)**
  Vypíše Bluetooth služby v okolí pomocí protokolu SDP. Hledá služby pouze dle daných argumentů. To znamená, že pokud je tato funkce zavolána bez argumentů, tak vypíše všechny dostupné služby.

Závěr

Cílem práce bylo vytvořit platformu k umožnění sběru dat přes Bluetooth z Raspberry Pi do Android zařízení. Výsledný projekt se skládá ze dvou knihoven, díky kterým je komunikace těchto zařízení přes Bluetooth výrazně usnadněná a přizpůsobena k tomuto účelu. Analýza a zpracování příchozích dat na Androidu jsou s použitím této platformy jednoduché. Byla vyvinuta rovněž demo aplikace na Android a demo skript na Raspberry Pi, které využití knihoven demonstrují. Na knihovnách se dají postavit složitější programy, které si uživatel napíše dle vlastního gusta pro jeho konkrétní použití. Při vývoji byl kladen důraz na jednoduchost použití a co nejmenší omezování uživatele. Je zde samozřejmě prostor pro rozšíření knihoven. Zcela jistě by bylo zajímavé dělat podobný projekt s využitím Bluetooth 5. Bohužel však žádné současné Raspberry Pi Bluetooth 5 nepodporuje. Dále by bylo možné přidělat do knihoven další třídu, kde by ale komunikace byla řešena s použitím BLE. Tak by si uživatelé mohli zvolit to Bluetooth, které jim pro jejich konkrétní situaci vyhovuje nejvíce.

Bibliografie

1. RASPBERRY PI FOUNDATION. *Raspberry Pi* [online] [cit. 2018-05-12]. Dostupné z: https://www.raspberrypi.org/.
2. BLUETOOTH SPECIAL INTEREST GROUP. *Specification of the Bluetooth System, v5.0* [online] [cit. 2018-04-27]. Dostupné z: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=421043.
3. GOEBEL, Hartmut. *Scheme of a Bluetooth Piconet* [online] [cit. 2018-05-02]. Dostupné z: https://commons.wikimedia.org/wiki/File:BluetoothPiconet-de.svg.
4. SIMS, Gary. *The truth about Bluetooth 5 – Gary explains* [online] [cit. 2018-04-28]. Dostupné z: https://www.androidauthority.com/bluetooth-5-speed-range-762369/.
5. INSTITUT FÜR ANGEWANDTE MIKROELEKTRONIK UND DATENTECHNIK. *Service Discovery Protocol (SDP)* [online] [cit. 2018-04-28]. Dostupné z: https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth_info/sdp.html.
6. DIGI INTERNATIONAL INC. *An Introduction to Wi-Fi* [online] [cit. 2018-05-06]. Dostupné z: http://ftp1.digi.com/support/documentation/0190170_b.pdf.
7. GOOGLE LLC. *Application Fundamentals* [online] [cit. 2018-04-29]. Dostupné z: https://developer.android.com/guide/components/fundamentals.
8. IDC CORPORATE USA. *Smartphone market share* [online] [cit. 2018-04-28]. Dostupné z: https://www.idc.com/promo/smartphone-market-share/os.

9. SINICKI, Adam. *I want to develop Android Apps — What languages should I learn?* [online] [cit. 2018-04-29]. Dostupné z: https://www.androidauthority.com/develop-android-apps-languages-learn-391008/.
10. N., Ramkumar. *Trip on Android Activity Life Cycle* [online] [cit. 2018-04-30]. Dostupné z: https://android.i-visionblog.com/trip-on-android-activity-life-cycle-%EF%B8%8F-3ea59a3261fb.
11. GOOGLE LLC. *The activity lifecycle* [online] [cit. 2018-04-30]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle.
12. ANONYM. *What is a Raspberry Pi?* [online] [cit. 2018-04-30]. Dostupné z: https://opensource.com/resources/raspberry-pi.
13. RASPBERRY PI FOUNDATION. *Usage* [online] [cit. 2018-05-01]. Dostupné z: https://www.raspberrypi.org/documentation/usage/.
14. RASPBERRY PI FOUNDATION. *FAQS* [online] [cit. 2018-04-30]. Dostupné z: https://www.raspberrypi.org/help/faqs.
15. RASPBERRY PI FOUNDATION. *Buy a Raspberry Pi* [online] [cit. 2018-04-30]. Dostupné z: https://www.raspberrypi.org/products/.
16. RASPBERRY PI FOUNDATION. *GPIO* [online] [cit. 2018-05-01]. Dostupné z: https://www.raspberrypi.org/documentation/usage/gpio/.
17. AFLAK, Omar. *Bluetooth knihovna na Android* [online] [cit. 2018-04-18]. Dostupné z: https://github.com/OmarAflak/Bluetooth-Library.
18. MALÝ, Martin. *Protokol MQTT: komunikační standard pro IoT* [online] [cit. 2018-04-18]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/.
19. BUTIX. *Vector illustration of a Raspberry Pi 3* [online] [cit. 2018-05-01]. Dostupné z: https://commons.wikimedia.org/wiki/File:Raspberry_Pi_3_illustration.svg.
20. GEERLING, Jeff. *Power Consumption Benchmarks* [online] [cit. 2018-05-01]. Dostupné z: https://www.linuxexpres.cz/hardware/raspberry-pi-vstup-vystup-gpio.
21. BLUEZ PROJECT. *About* [online] [cit. 2018-05-02]. Dostupné z: http://www.bluez.org/about/.
22. STROUSTRUP, Bjarne. *The C++ Programming Language*. Čtvrtá edice. Texas: Addison-Wesley, 2013. ISBN 978-0321563842.
23. PYTHON SOFTWARE FOUNDATION. *About Python* [online] [cit. 2018-05-02]. Dostupné z: https://www.python.org/about/.

24. GITHUB KONTRIBUTOŘI. *Bluetooth Python extension module* [online] [cit. 2018-04-18]. Dostupné z: `https://github.com/pybluez/pybluez`.

25. GOOGLE LLC. *Android reference, createbond* [online] [cit. 2018-05-04]. Dostupné z: `https://developer.android.com/reference/android/bluetooth/BluetoothDevice.html#createBond()`.

26. GEHRING, Jonas. *GraphView - open source graph plotting library for Android* [online] [cit. 2018-05-04]. Dostupné z: `http://www.android-graphview.org/`.

27. ACTIVESTATE SOFTWARE INC. *DOWNLOAD KOMODO EDIT* [online] [cit. 2018-05-05]. Dostupné z: `https://www.activestate.com/komodo-ide/downloads/edit`.

28. TELEDYNE LECROY, INC. *Legacy Pairing (Bluetooth 2.0 and earlier)* [online] [cit. 2018-05-03]. Dostupné z: `http://www.fte.com/webhelp/bpa500/Content/Documentation/WhitePapers/BPA600/Encryption/LegacyPairing.htm`.

29. HONEYWELL INTERNATIONAL INC. *Bluetooth Secure Simple Pairing'(SSP)* [online] [cit. 2018-05-03]. Dostupné z: `https://support.honeywellaidc.com/s/article/Bluetooth-Secure-Simple-Pairing-SSP`.

30. PADGETTE, John et al. *Guide to Bluetooth Security* [online] [cit. 2018-05-11]. Dostupné z: `https://csrc.nist.gov/publications/detail/sp/800-121/rev-2/final`.

Příloha A

Obsah přiloženého datového média

- `readme.txt` ..... stručný popis obsahu datového média
- `Rfcommtest.apk` ..... instalační balíček demo aplikace na Android
- `src`
  - `android` ..... adresář s demo aplikací a knihovnou na Android
  - `rasppi` ..... adresář s demo skriptem a knihovnou na Raspberry Pi
  - `thesis` ..... adresář se zdrojovým souborem .tex a obrázky
- `BP_Pirko_Ondrej_2018.pdf` ..... text práce ve formátu PDF