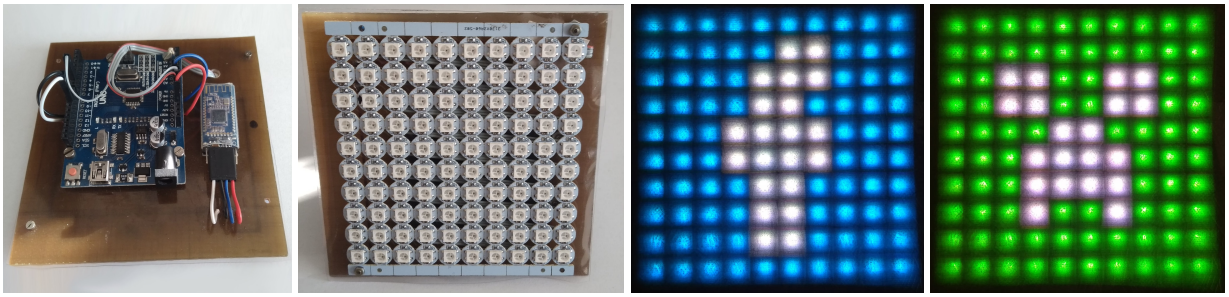
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2020
http://excel.fit.vutbr.cz

Hardwarový indikátor oznámení pro telefony se systémem Android

Roman Bártl*



Abstrakt

Cílem této práce je navrhnout a vytvořit notifikační zařízení, které bude zobrazovat nejrůznější upozornění z chytrých telefonů. Jedná se o upozornění na příchozí hovory, stav baterie a notifikace nainstalovaných aplikací. Jádrem zařízení je jednodeskový počítač Arduino. Součástí práce je i implementace aplikace pro operační systém Android, která slouží jak pro komunikaci s tímto zařízením, tak i k samotnému zachytávání zmíněných událostí. Pro komunikaci je použita technologie Bluetooth Low Energy. Je kladen důraz na jednoduchost a srozumitelnost celého systému včetně zajištění co největší možné kompatibility napříč verzemi systému Android.

Klíčová slova: Arduino — Android — Bluetooth Low Energy — Vestavěný systém — Notifikace

Přiložené materiály: Demonstrační video

*xbartl06@fit.vutbr.cz, *Fakulta informačních technologií, Vysoké učení technické v Brně*

1. Úvod

Motivací pro tuto práci je fakt, že se v poslední době začalo ustupovat od notifikačních LED diod u většiny mobilních zařízení kvůli tzv. „bezrámečkovým“ displejům, a uživatelé tak ztrácí možnost být bez vyrušení informování o příchozích upozorněních. A i když mobilní zařízení tuto diodu obsahuje, bývá často její nastavení dosti omezené a pro některé uživatele toto může být nedostačující. Proto podobné notifikační zařízení může přijít náročnějším uživatelům velmi vhod. Dále může toto zařízení například sloužit sluchově postiženým, jež jsou závislí na vizuálních podnětech.

Vestavěný systém je tvořen ze dvou částí – zobrazovací zařízení a aplikace pro telefony se systémem Android. Základním prvkem zobrazovacího zařízení je malý jednodeskový počítač Arduino. Zobrazovacím prvkem je matice tvořená z RGB LED diod. Tento maticový displej zobrazuje ikony, které indikují jednotlivá upozornění. Pro komunikaci mezi tímto zařízením a aplikací se využívá bezdrátová technologie Bluetooth Low Energy (dále jen BLE), proto musí zařízení obsahovat BLE modul.

V aplikaci má uživatel možnost specifikovat, jaká upozornění se budou zobrazovat a těm nastavit libovolnou ikonu, která bude promítána na maticovém displeji zařízení. Mezi zachytávaná upozornění patří příchozí hovory, stav baterie nebo notifikace nainstalovaných aplikací. K zachytávání těchto událostí využívá aplikace komponenty *BroadcastReceiver* a *NotificationListenerService* (viz kapitola 2.2). Pro dosažení perzistentního spojení k zařízení je součástí aplikace i služba na pozadí, která zůstane spuštěna i v případě

vypnutí aplikace (respektive všech aktivit aplikace).

Kromě návrhu a implementace tohoto vestavěného sytému bude v tomto článku i zmínka o testování na různých modelech telefonů a verzí systému Android.

2. Funkce systému Android

2.1 Bezdrátová komunikace

Je zřejmé, že pro dlouhodobě fungující aplikaci je vhodné využít komunikační technologii, která je zaměřena na úsporu baterie. Jednou z těchto technologií je NFC (Near Field Communication), která umožňuje oboustranné zasílání dat mezi zařízeními s krátkým dosahem (4 cm). Na platformě Android se můžeme s touto technologií setkat od verze 4.0, kde je označována jako funkce Android Beam [1]. Zasílání dat pomocí Android Beam se aktivuje přiložením dvou telefonů s podporou NFC zády k sobě. Teprve po té se na zařízení, ze kterého chceme odesílat data, objeví výzva „Tap to beam“. Klepnutím na obrazovku se data začnou přenášet do cílového zařízení. Není tedy možné pomocí této technologie vytvářet automatizované systémy.

Druhou komunikační technologií zaměřenou na šetření baterie je BLE. To je zcela odlišné od klasického Bluetooth a nejsou kompatibilní. Proto je nutné zkontrolovat, zda smartphone tuto funkci umožňuje. Mezi základní prvky architektury BLE patří ATT (Attribute Protocol) a jeho implementační rozšíření GATT (Generic ATT) [2]. ATT je jednoduchý bezstavový protokol klient/server. Každý server má data organizovaná ve formě atributů a každému z nich je přiřazeno identifikační číslo UUID (Universally Unique Identifer). Jednotlivé identifikátory určují typ přenášených dat. Tyto UUID jsou vysílány serverem za účelem poskytnutí informací o svých nabízených atributech (typech dat). GATT přidává abstrakční model atributů. Hlavní logická jednotka protokolu se nazývá služba. Každá služba se váže ke konkrétní funkcionalitě zařízení a skládá se z charakteristik, které určují různé hodnoty služby. Příkladem služby může být *Monitor srdečního tepu* obsahující charakteristiku *Frekvence srdečního tepu*. Výchozí délka paketu přenášených zpráv (MTU – Maximum Transmission Unit) je nastavena na 20 bajtů. V některých případech lze MTU nastavit na vyšší hodnotu, ovšem za předpokladu, že to hardware umožňuje.

2.2 Zachytávání systémových událostí

Na operačním systému Android existuje mnoho způsobů, jak lze sledovat nejrůznější události, které mohou na zařízení nastat [3].

BroadcastReceiver (dále jen receiver) je komponenta, která slouží k přijímání broadcast zpráv zasílaných ostatními částmi aplikace, jinými aplikacemi nebo systémem samotným. Při registraci receiverů se určuje, jakým typům broadcast zpráv budou naslouchat. Registrace může být buď statická, nebo dynamická. Statická registrace se provádí v manifestačním souboru *AndroidManifest.xml*[1]. Tyto receivery je možné spouštět i v případě, že je aplikace vypnutá. Když systém vytvoří nějakou událost, které staticky registrovaný receiver naslouchá, vytvoří se jeho instance a příslušné operace se provedou na pozadí. Oproti tomu dynamické registrování se provádí přímo v kódu aplikace nejčastěji pak v aktivitách nebo službách. Výhodou oproti statické registraci je možnost odregistrace receiveru v případě, že již není potřebný.

NotificationListenerService [4] (dále jen NLS) je systémový nástroj, umožňující aplikacím zachytávat informace o všech příchozích notifikacích v zařízení. Při nově příchozím upozornění je službě prostřednictvím systému zaslán objekt, zapouzdřující veškeré informace, které jsou prezentovány uživateli ve stavovém řádku systému. Jedná se například o název balíčku, nadpis a text notifikace, čas zobrazení, případně ikonu. Na rozdíl klasických oprávnění automaticky vyžadovaných prostřednictvím systémových dialogových oken je pro NLS zavedena zvláštní část v nastavení systému. Jedná se o zvláštní opatření, které zabraňuje úniku citlivých údajů, které mohou notifikace poskytovat (například zprávy SMS). Toto může být pro běžného uživatele velice nepohodlné, ale naštěstí je možné toto zjednodušit pomocí odkazu v aplikaci, který uživatele do nastavení přímo nasměruje.

2.3 Služby

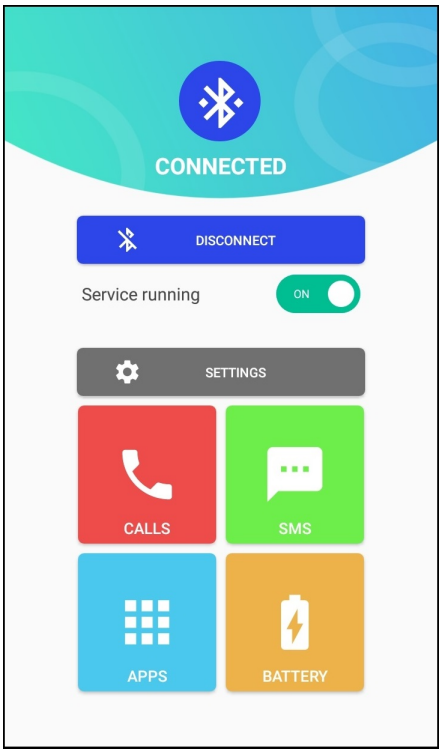
Služba [5] je komponenta aplikace, která umožňuje provádět libovolné operace na pozadí bez nutnosti interakce s uživatelem. Speciálním typem služby je tzv. *ForegroundService*. Tento typ je dle oficiální dokumentace doporučeno používat vždy, když aplikace provádí nějakou dlouhodobou operaci, o které by měl uživatel vědět (indikováno nesmazatelnou notifikací ve stavovém řádku systému). Důležitou metodou, kterou služba může implementovat, je *onStartCommand()*. Ta vrací jednu z konstant, které určují, jestli se služba restartuje, když ji systém ukončí kvůli nedostatku prostředků.

[1] Tento soubor musí obsahovat každý projekt ve svém kořenovém adresáři. Poskytuje základní informace o aplikaci samotnému systému. Obsahuje seznam všech aktivit a služeb, ale také i vyžadovaná oprávnění.

3. Návrh aplikace

U grafické stránky aplikace je kladen důraz především na jednoduchost a srozumitelnost a aby byly veškeré prvky co nejvíce intuitivní. Uživatel bude ve většině případů aplikaci používat pro připojování k notifikačnímu zařízení. Zbylé procento vykonaných akcí bude nastavování, jak se má tento systém chovat.

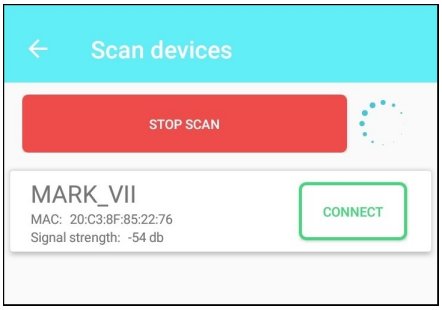
Nejvyužívanější částí aplikace bude hlavní aktivita[2] (viz obrázek 1). V této aktivitě se uživatelé připojují k zařízení. V horní části obrazovky je zobrazen stav připojení (*Připojeno*/*Připojování*/*Odpojeno*). Dále aktivita obsahuje odkazy do dalších aktivit, které slouží k nastavování upozornění. Uživatel může nastavovat upozornění pro příchozí hovory, stav baterie a notifikace nainstalovaných aplikací na telefonu. Dále si může uživatel určit jas LED diod (tlačítko *Settings*).



Obrázek 1. Hlavní aktivita aplikace, ve které se uživatel bude připojovat k zařízení. Obsahuje odkazy do aktivit pro nastavení upozornění.

Seznam nainstalovaných aplikací se generuje pomocí *PackageManageru*, který získává informace o nainstalovaných aplikacích. Ke každé aplikaci ukládá nastavení do lokální databáze SQLite[3]. Uživatel také může nastavit pro jednotlivá upozornění (i jednotlivé aplikace), zda je chce na zařízení zobrazovat.

Před samotným zahájením komunikace je zapotřebí se k zařízení připojit. Nejprve musí aplikace najít všechna dostupná zařízení pomocí skenování. Ta zobrazí uživateli do seznamu seřazeného dle síly signálu (viz obrázek 2). Uživatel si následně vybere, ke kterému zařízení se chce připojit. Pro komunikaci se zařízením využívá speciální formát zpráv. Tento protokol umí rozlišovat jednotlivé typy upozornění. Důvod pro toto rozdělení je především kvůli prioritám těchto typů upozornění. Například příchozí hovory se budou na displeji zobrazovat tak dlouho, jak bude vyzvánět telefon. Notifikace aplikací se budou zobrazovat do doby, než budou smazány ze stavového řádku telefonu.



Obrázek 2. Aktivita pro skenování zobrazující dostupná zařízení seřazené sestupně podle síly signálu. Uživatel může vybírat, které z nich chce používat.

4. Návrh zařízení

Zařízení se skládá ze tří částí – klon Arduino UNO, BLE modul a maticový RGB LED displej. U výběru komponent bylo zapotřebí si dávat pozor na spotřebu obou modulů, protože maximální výstupní proud desky Arduino je 500 mA při napájení z USB. V případě Bluetooth modulu to nebyl žádný problém, jelikož jeho odběr činí maximálně 400 μA. Ovšem odběr vybraného maticového modulu může dosahovat až 6 A – displej se skládá z 10×10 LED diod typu *WS2812B* a každá z nich při nejvyšším jasu odebírá 60 mA (každá barevná složka RGB 20 mA). Při průběžném testování diod jsem ale zjistil, že maximální jas je velmi oslnivý a pro oči nepříjemný. Maximální jas, který půjde nastavit bude zhruba 7,8%[4] maximálního možného jasu LED diod a tedy výsledný odběr modulu bude přibližně 468 mA. Pro zjednodušení práce těmito diodami navrhla společnost Adafruit Industries knihovnu *NeoPixels* [7]. Pomocí PWM (pulzně šířkové modulace) umožňuje jednotlivým diodám snadno měnit barvu i jas.

Datový vodič pro LED diody musí být zapojen do pinu, který umožňuje PWM (viz obrázek 3). Pro

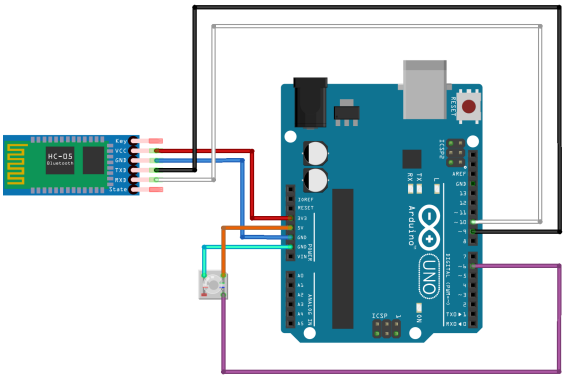
[2] Aktivity umožňují uživatelům skrze grafické rozhraní ovládat aplikaci a komunikovat s ní.

[3] SQLite je relativně malá, výkonná multiplatformní knihovna. Samotná data jsou uložena v lokálním souboru, ke kterým se přistupuje za pomoci klientské aplikace [6].

[4] Maximální hodnota jasu, která lze za pomoci knihovny *NeoPixels* nastavit je 255, v případě tohoto zařízení je hodnota nastavena na 20

datové přenosy mezi BLE modulem a Arduinem se využívá sériové komunikace, proto datové vodiče mohou být připojeny na jakékoliv digitální piny desky Arduino.

Pro zařízení bude také vytištěné pouzdro, které je prozatím ve fázi návrhu.



Obrázek 3. Diagram zapojení modulů k desce Arduino. Maticový modul je zde reprezentován jednou diodou. Datový vodič diody je připojen na pin 6, který umožňuje PWM. Datové vodiče BLE modulu jsou zapojeny na piny 9 a 10. Zbylé vodiče slouží pro napájení modulů.

5. Implementace

V této části článku se zaměřím na realizaci nejdůležitějších částí aplikace a zařízení.

5.1 Služby na pozadí

Aplikace obsahuje dvě služby. První z nich je využita pro komunikaci se zařízením pomocí Bluetooth. Přijímá broadcast zprávy od ostatních komponent aplikace a provádí určité akce Jedná se například připojení se k zařízení nebo odeslání zprávy na zařízení po té, co byla zachycena některá z událostí. Tato služba je spouštěna jako *ForegroundService*. Její implementace obsahuje metodu *onStartCommand()*. Aby služba zůstala běžet i v případě ukončení aplikace uživatelem, musí tato metoda vracet konstantu `START_STICKY`, která oznamuje systému, aby byla služba vždy restartována [5].

Druhá služba fungující na pozadí je potomek třídy *NotificationListenerService*. Jejím jediným úkolem je zasílat broadcast zprávy ohledně zachycení přidané nebo smazané notifikace výše zmíněné službě pro komunikaci. Ve zprávě se posílá název balíčku aplikace, která notifikaci vytvořila, a informaci, zda byla vytvořena nová nebo odebrána již existující.

Řešení problémů u testovaných zařízení je popsáno v kapitole 6.1.

5.2 Zachytávání událostí

Zachytávání notifikací, jak již bylo zmíněno, probíhá pomocí služby *NotificationListenerService*. Pro získávání stavu baterie a zachytávání příchozích hovorů jsou využity staticky registrované *BroadcastReceivery*. Přístup k informacím o příchozích hovorech je limitován na aplikace, kterým uživatel přidělil oprávnění, jelikož se jedná o citlivé údaje. Toto oprávnění se uvádí v manifestačním souboru a je pak vyžadováno po uživateli při spuštění aplikace pomocí systémového dialogového okna.

5.3 Komunikace přes BLE

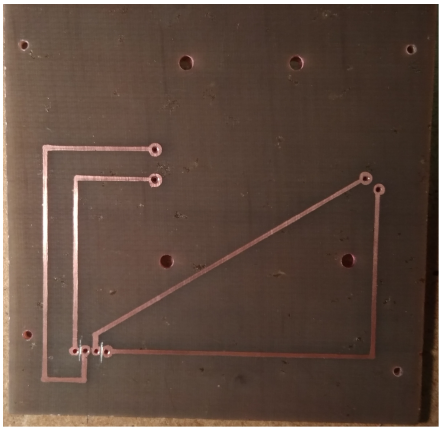
Před zahájením komunikace je nutné zjistit všechna dostupná zařízení v okolí pomocí skenování. Aby však aplikace našla pouze zařízení pro zobrazování notifikací, bude nejefektivnější změnit BLE modulům jejich názvy. Aplikace pak při skenování bude ignorovat veškerá zařízení s rozdílným názvem – tzv. *filtrované skenování*. Díky skenování pak aplikace může zjistit MAC adresu zvoleného zařízení (modulu), která je využitá pro připojení.

Jakmile se podaří úspěšně k zařízení připojit (indikováno pomocí callbacku), pokusí se aplikace najít službu a její charakteristiku, která bude využívána pro přenos dat na zařízení. Využitý BLE modul poskytuje službu pro sériovou komunikaci a charakteristiku pro zasílání dat na modul. Služba i charakteristika jsou pro všechny moduly tohoto typu identifikovány pomocí stejných *UUID*, proto je možné tyto hodnoty „natvrdo“ napsat do kódu aplikace. Nejprve potřeba pomocí *UUID* nalézt danou službu a až díky ní její charakteristiku. Po nalezení charakteristiky pomocí metody *BluetoothGattService.getCharacteristic(UUID uuid)*, vrátí tato metoda objekt, který se bude využívat pro zasílání dat na zařízení.

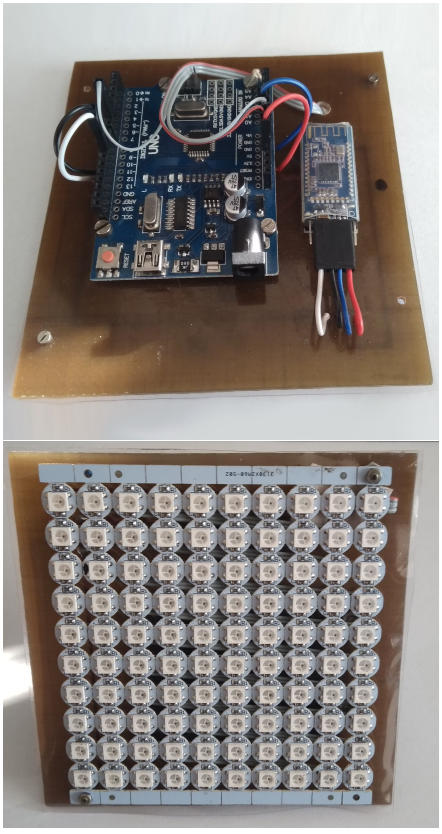
Jak jsem uvedl v kapitole 2.1, výchozí velikost jednotlivých packetů v případě technologie BLE je 20 bajtů a ne všechen hardware podporuje zvýšení MTU (včetně telefonů) [8]. Dle dokumentace mnou vybraný modul tuto možnost neposkytuje. Také nelze předpokládat, že každý telefon, na kterém aplikace bude běžet, bude umožňovat změnu MTU. Proto jsem se rozhodl vždy posílat data po packetech délky 20 bajtů. Po odeslání dat na zařízení, je zavolán callback označující stav přenosu dat (metoda *onWriteCharacteristics()*). Pokud přenos packetu proběhl v pořádku, odešle aplikace následujících 20 bajtů zprávy, v případě neúspěšného přenosu se pokusí odeslat packet znovu. Protože může nastat situace, že přijde nová notifikace v době, kdy již probíhá přenos dat, použil jsem semafor, který vždy pozastaví nově vytvořené vlákno do doby, než budou data odeslána.

5.4 Základní deska a finální zapojení

Kvůli jednoduchému přenášení zařízení jsem umístil všechny moduly na jednu základní desku vytvořenou z cuprextitu, jež obsahuje plošné spoje (viz obrázek 4). Tyto spoje slouží k přehlednému propojení mezi komponenty. Propojení plošných spojů a komponent je tvořeno drátovými propojkami (viz obrázek 5).



Obrázek 4. Základní deska s plošnými spoji propojující jednotlivé moduly zařízení.



Obrázek 5. Nahoře: zadní strana základní desky s připojenými moduly k desce Arduino. Dole: na přední straně základní desky je připevněn maticový RGB LED displej.

6. Testování

6.1 Testování kompatibility

Testování doposud proběhlo na osmi různých zařízeních s různými verzemi operačního systému Android:

- Xiaomi Redmi 4a (Android 7.1)
- Xiaomi Redmi 4a (Android 6.0)
- Xiaomi Redmi 7a (Android 9.0)
- Xiaomi Redmi Note 5 (Android 9.0)
- Doogee X5 Pro (Android 5.1)
- Zopo C2 (Android 6.0)
- Samsung Galaxy J5 2016 (Android 7.1)
- Lenovo A2010 (Android 5.1)

Důvod proč jsem pro testování vybral ve většině smartphony vyrobené v Číně (*Xiaomi*, *Doogee*, *Zopo*), je ten, že většina nadstaveb na Android u těchto telefonů obsahuje sytém šetření baterie, který automaticky vypíná všechny procesy běžící na pozadí v případě ukončení aktivit aplikace [9]. Tím pádem není možné ponechat službu typu *ForegroudService* běžet bez aktivit, pokud nemá uživatelem přidělená patřičná oprávnění. Stejně je na tom i NLS, která je od aplikace odpojena, a tedy nemůže zasílat aplikaci informace o zachytávaných notifikacích.

Aby mohla služba zůstat nepřetržitě spuštěná, je nutné v případě telefonů značky *Xiaomi* přidělit aplikaci oprávnění k automatickému spouštění. Do kódu aplikace jsem přidal odkaz do tohoto nastavení podobně jako v případě NLS. Tento odkaz může být použit pouze u telefonů této značky, čehož je docíleno pomocí třídy *Build*, která zpřístupňuje informace o Android zařízení, na kterém je aplikace právě spuštěna. Bohužel v případě smartphonu *Xiaomi Redmi 4a* s verzí Androidu 7.1 se mi nepodařilo, aby služba na popředí fungovala, když uživatel aplikaci vypne, a to ani po udělení oprávnění. Stejně tak přestane fungovat i NLS. Takovéto chování je nejspíše způsobené chybou dané verze operačního systému (případně nadstavby). U ostatních modelů *Xiaomi* fungovaly služby na pozadí dle očekávání po udělení práv.

V případě telefonu *Doogee X5 Pro* se mi opět nepodařilo, aby *ForegroundService* fungovala i bez zapnuté aplikace. Také jsem nenašel v nastavení žádná oprávnění k automatickému spouštění jako tomu bylo v případě telefonů *Xiaomi*. Paradoxně však služba NLS zůstala vždy spuštěná (otestováno ladícími zprávami v Android Studiu).

Dalším problémem, na který jsem v průběhu testování narazil, byl u telefonu *Zopo C2*. Na tomto zařízení nebyla aplikace schopna najít služby a charakteristiky BLE modulu hned po ustavení spojení. Efektivním

řešením bylo začít hledat služby (a tedy charakteristiky) až po uplynutí určité doby po ustavení spojení s modulem [10]. V aplikaci jsem nastavil čekání na 1 s.

U zbylých testovaných zařízeních nebyly nalezeny žádné další problémy.

6.2 Testování uživatelů

V této části textu se zmíním o testování layoutu uživateli a jeho následné úpravě, a také o požadavcích uživatelů na přidání nových funkcí do aplikace/zařízení.

Největším problémem původního návrhu grafického uživatelského rozhraní bylo, že uživatelé nevěděli, jak se k zařízení připojit. Ikona, která nyní indikuje stav připojení k zařízení, fungovala původně i jako tlačítko pro manuální připojení/odpojení se k/od zařízení. Toto řešení se ukázalo jako nepoužitelné, protože žádný z testovaných uživatelů nebyl schopen poznat, že se jedná o tlačítko. Proto jsem nakonec do layoutu přidal pro tyto účely separátní tlačítko, jak je zobrazeno v konečném návrhu.

Druhá změna GUI se týkala výběru číselných hodnot v případě nastavení priority notifikací a procentuálního stavu baterie, při kterém se pošle upozornění na zařízení. Původní komponenta pro výběr čísel byla *NumberPicker*, která se zobrazovala v dialogovém okně. Dle některých testerů by bylo lepší spíše využít posuvník (*SeekBar*), který nakonec v konečném návrhu layoutu zůstal, protože je na první pohled srozumitelnější.

Některým uživatelům přišlo nepohodlné a zdlouhavé hledání aplikace pro SMS zprávy mezi ostatními systémovými aplikacemi, proto jsem přidal do hlavní aktivity odkaz v podobě tlačítka.

Jedním z požadavků na funkcionalitu celkového systému bylo, aby si uživatelé mohli vyzkoušet vzhled ikony, která se bude zobrazovat na maticovém displeji. Proto jsem do aktivit pro nastavení upozornění přidal tlačítko, které odešle ikonu daného upozornění na zařízení, kde bude zobrazena.

Dalším žádaným rozšířením byla možnost upravit jas LED diod, protože se některým uživatelům stále připadaly diody velice jasné, i přes to, že byly sníženy na téměř 8% své maximální svítivosti.

Po úpravách dle výše zmíněných nedostatků a požadavků proběhlo druhé testování. Všichni uživatelé již byli schopni se bez pomoci na zařízení připojit. Dále také přidání odkazu na nastavení SMS aplikace byla velmi vítanou změnou. Největší ohlasy u uživatelů však získala možnost vyzkoušet vzhled jimi nastavených ikon.

Testování proběhlo veskrze pozitivně, protože pomohlo odhalit nedostatky a zároveň přineslo některá vylepšení pro celý tento vestavěný systém.

7. Závěr

Cílem této práce bylo navrhnout a realizovat zařízení zobrazující notifikace včetně aplikace pro telefony, která tato upozornění zachytává a zasílá na zařízení prostřednictvím Bluetooth Low Energy.

Aplikace byla testována na různých modelech chytrých telefonů s různými verzemi operačního systémů Android. Některé chyby, které byly způsobeny ať už právě konkrétní verzí operačního systému, nadstavby nebo hardwarovou implementací telefonu, byly ošetřeny.

Toto zařízení by mohlo být používáno v místech, kde by zvukové notifikace byly obtíží jako například kanceláře (samozřejmě by záleželo na jednotlivcích). Dalšími možnými uživateli by mohli být sluchově postižení, kteří jsou závislí na vizuálních podnětech. Nebo by jej mohli využívat techničtí nadšenci. Navíc díky rozšíření kompatibility napříč různými modely telefonů s operačním systémem Android může být toto zařízení používáno větším množstvím uživatelů.

Další vývoj zařízení by mohl zahrnovat podrobnější testování aplikace na vícero mobilních zařízeních a umožnit tak vícero uživatelům využívat tento vestavěný systém. Dále by se mohlo experimentovat s různými variantami zobrazování upozornění například využití LCD displejů.

Poděkování

Rád bych poděkoval mému vedoucímu bakalářské práce prof. Ing. Adamu Heroutovi, Ph.D. za odborné vedení, cenné rady a připomínky při konzultacích. Dále bych rád poděkoval všem, kteří se podíleli na testování a za jejich zpětnou vazbu a především pak Jiřímu Polákovi za ochotu při tisknutí pouzdra pro hardwarové zařízení.

Literatura

[1] Nathan Chandler. What is android beam? how stuff works? blogpost (english), Feb 2012. https://electronics.howstuffworks.com/android-beam1.htm.

[2] Kevin Townsend, Cufi Carles, Akiba, and Robert Davidson. *Getting started with Bluetooth low energy*. OReilly Media, 2014.

[3] Jan Piskáček. Možnosti sledování událostí na systému android a jejich využití pro odhalování podezřelého chování aplikací. Bakalářská práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, 2015. https://dspace.cvut.cz/bitstream/handle/10467/

61674/F3-BP-2015-Piskacek-Jan-piskaja2_2015_bp.pdf.

[4] Inc. Google. Notificationlistenerservice. online (english), 2020. https://developer.android.com/reference/android/service/notification/NotificationListenerService.

[5] Inc. Google. Services overview. online (english), 2020. https://developer.android.com/guide/components/services.

[6] Ľuboslav Lacko. *Mistrovství – Android*. Computer Press, 2017.

[7] Adafruit Industries. Some finer points of neopixels. online (english), 2020. https://learn.adafruit.com/sipping-power-with-neopixels/insights.

[8] Inc. GitHub. Mtu limit of 20 bytes on some android devices. blogpost (english), May 2016. https://github.com/don/cordova-plugin-ble-central/issues/234.

[9] Reddit. Workmanager reliability for periodic tasks on chinese roms (xiaomi, huawei, and so on). blogpost (english), Oct 2018. https://www.reddit.com/r/androiddev/comments/9ra0iq/workmanager_reliability_for_periodic_tasks_on/.

[10] Stack Overflow. onservicesdiscovered never called while connecting to gatt server. blogpost (english), Jan 2017. https://stackoverflow.com/questions/41434555/onservicesdiscovered-never-called-while-connecting-to-gatt-server.