FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Platforma pro komunikace mezi Android a Arduino
Student:	Juraj Filan
Vedoucí:	Ing. Radomír Polách
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Nastudujte princip komunikace pomocí technologie Bluetooth v prostředí platforem Android a Arduino pro potřeby Internetu věcí.
Navrhněte a implementujte platformu pro komunikaci mezi Android a Arduino pomocí technologie Bluetooth.
Zaměřte se na bezpečnost a možnost propojení většího množství Arduino k jednomu zařízení s operačním systémem Android.
Implementaci důkladně testujte pomocí vhodné demo aplikace a řádně ji zdokumentujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 4. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Platforma pre komunikáciu medzi Android a Arduino

Juraj Filan

Katedra softwarového inženýrství
Vedúci práce: Ing. Radomír Polách

15. mája 2019

Poďakovanie

Rád by som sa poďakoval svojmu vedúcemu práce Ing. Radomírovi Poláchovi za rady a čas, ktorý mi venoval. Taktiež sa chcem poďakovať rodine a priateľom za podporu pri písaní tejto práce.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 15. mája 2019 ……

Odkaz na túto prácu

Filan, Juraj. *Platforma pre komunikáciu medzi Android a Arduino.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Táto bakalárska práca opisuje proces tvorby platformy pre komunikáciu medzi mobilným zariadením Android a väčším množstvom zariadení Arduino pomocou bezdrôtovej technológie Bluetooth. Hlavnou témou práce je analýza, návrh a implementácia Android aplikácie, pomocou ktorej bude používateľ riadiť túto komunikáciu. Práca taktiež opisuje vývoj jednoduchého programu pre Arduino, ktorý bude schopný prijímať a odosielať dáta z tejto mobilnej aplikácie. Nakoniec bude tento proces komunikácie dôkladne otestovaný.

Klíčová slova Android, Arduino, Bluetooth, mobilná aplikácia, bezdrôtová komunikácia

Abstract

This bachelor thesis describes process of creation of platfrom for communication between Android mobile device and bigger amount of Arduino devices through wireless technology Bluetooth. The main topic of the thesis is analysis, design and implementation of Android application, by which a user will be able to control this communication. The thesis also describes a development of simple program for Arduino, which will be able to send and receive data from this mobile application. Finally the process of communication will be thoroughly tested.

Keywords Android, Arduino, Bluetooth, mobile application, wireless communication

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam zdrojových kódov

Úvod

V dnešnom modernom svete sme obklopení rôznymi technológiami, ktoré nám uľahčujú život. Jedným z najpoužívanejších technických zariadení je smartfón. Pri tomto zariadení strávi moderný človek denne až príliš mnoho času. Hlavným dôvodom je využívanie týchto zariadení na komunikáciu medzi ľuďmi. Predstavme si ale, že by sme boli schopní pomocou svojho smartfónu ovládať rôzne mechanizmy v našom okolí, ktoré každodenne využívame. Pod týmito mechanizmami si môžeme predstaviť napríklad zapnutie a vypnutie žiarovky, otvorenie a zatvorenie brány od domu.

Na ovládanie rôznych mechanizmov je možné využiť Arduino, čo je jednoduchá elektronická platforma, na ktorú používateľ môže pripojiť a nainštalovať rôzne komponenty. S Arduinom bude komunikovať smartfón bežiaci na operačnom systéme Android. Komunikácia bude prebiehať pomocou bezdrôtového pripojenia Bluetooth. Obidve zariadenia budú pomocou tejto komunikácie schopné prijímať a odosielať dáta, ktoré vykonajú nejakú inštrukciu. Pomocou smartfónu bude možné komunikovať s väčším množstvom zariadení Arduino. Táto komunikácia bude riadená pomocou mobilnej aplikácie nainštalovanej na Android zariadení.

Kapitola 1

Cieľ práce

Hlavným cieľom práce je na základe naštudovania komunikácie pomocou Bluetooth na platformách Android a Arduino navrhnúť a implementovať mobilnú aplikáciu Android, ktorá bude schopná pripojiť sa a komunikovať s väčším množstvom Arduino zariadení. Aplikácia bude schopná komunikovať obojstranne, to znamená, že bude môcť prijímať aj odosielať dáta. K tejto aplikácií je taktiež potrebné navrhnúť a implementovať program pre zariadenie Arduino. Na záver bude táto aplikácia spolu s programom dôkladne otestovaná.

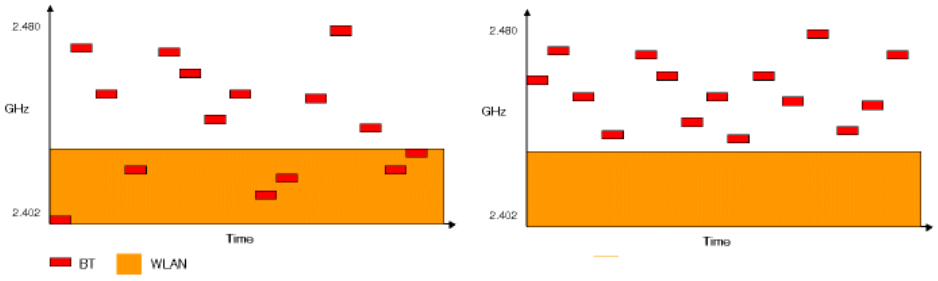
Čiastkové ciele sú založené na analýze už vytvorených riešení. Jedným z čiastkových cieľov je nájsť dostupné knižnice tretej strany pre Android, ktoré uľahčujú implementáciu Bluetooth komunikácie. Taktiež bude potrebné vyhľadať a otestovať už vytvorené mobilné aplikácie na túto tému a zanalyzovať ich funkcionalitu.

Kapitola 2

Rešerš

2.1 Bluetooth technológia

Bluetooth je bezdrôtová komunikačná technológia na krátku vzdialenosť, ktorej hlavnou úlohou je používateľov odbremeniť od káblov. Príkladmi môžu byť bezdrôtová myš, klávesnica alebo slúchadlá. Komunikácia prebieha pomocou techniky `Adaptive Frequency Hopping (AFH)`, ktorá má na starosti rozdeľovanie frekvencie v intervale 2,402-2,480 GHz do 79 fyzických kanálov s frekvenciou šírky 1 MHz. Tento skok vo frekvencií sa nazýva `(hop)`. Rovnaký interval využíva aj bezdrôtové pripojenie Wi-Fi.[1] Bluetooth zariadenie vďaka tejto funkcií taktiež zistí, ktoré kanály sú využívané inými zariadeniami a také kanály vynechá. Týmto sa vyhne kolízií znázornenej na obrázkoch 2.1, vďaka čomu nedôjde k narušeniu signálu. Túto sekvenciu skokov pozná odosielateľ aj prijímateľ. [2] Počet skokov za sekundu je 1600, takže jeden kanál existuje iba 0.625 ms. [3]

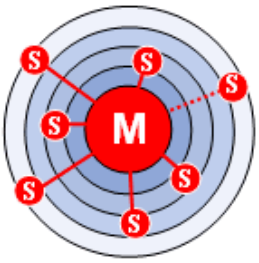


Obr. 2.1: Rozdiel použitia AFH pri náhodných skokoch. Vľavo nebola použitá a vpravo bola použitá funkcia AFH.[4]

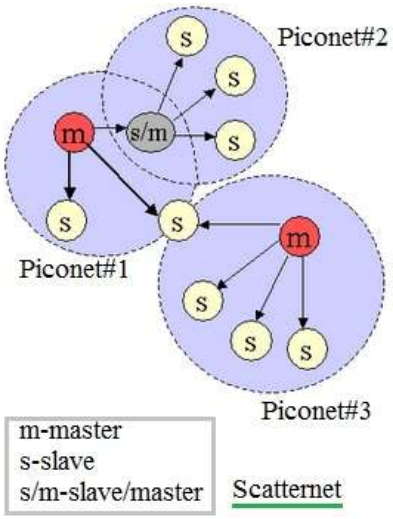
2.1.1 Princíp komunikácie

Bluetooth spojenie viacerých zariadení využíva `master-slave model` v komunikačnej sieti, ktorá sa nazýva `piconet`. `Master` zariadenie je schopné komunikovať až so siedmimi rôznymi `slave` zariadeniami, zatiaľ čo ony nemôžu komunikovať medzi sebou. Komunikácia prebieha cez obojstranné prijímanie a odosielanie dát. [5] V praxi môže byť `master` zariadenie pripojené práve s jedným `slave` zariadením a pri komunikácií s väčším množstvom zariadení dochádza k rýchlemu prepínaniu medzi zariadeniami. [1] Táto sieť je znázornená na obrázku 2.2.

Sieť pozostávajúca z viacerých sietí `piconet` sa nazýva `scatternet`. Môže vzniknúť buď tak, že sa jedno zariadenia bude správať v dvoch rôznych sietiach `piconet` ako `slave` alebo v jednej ako `slave` a v druhej ako `master`, pozri obrázok 2.3. [6]



Obr. 2.2: Komunikačná sieť piconet s master (M) zariadením a slave (S) zariadeniami [1]



Obr. 2.3: Komunikačná sieť scatternet [6]

2.1.2 História technológie Bluetooth

Prvá verejná verzia Bluetooth 1.0 bola zverejnená v roku 1999. Rok po roku sa táto technológia postupne vyvíjala a vznikali nové verzie, ktoré nadobúdali na rýchlosti a vzdialenosti pripojenia. Tieto rozdiely rôznych verzií sú uvedené v tabuľke 2.1.

Od verzie 2.0 bol zavedený nový systém zvaný `Enhanced Data Rate (EDR)`, ktorý výrazne zvýšil rýchlosť prenosu dát.

Vo verzií 2.1 prvý krát zaviedli metódu párovania `Secure Simple Pairing`, ktorá je bližšie opísaná v sekcii 2.2.2. [7] Od tejto verzie bola taktiež zavedená funkcia `Sniff subrating`, ktorá má na starosti rozhodnúť, ako dlho bude čakať na poslanie informačnej správy, či je zariadenie stále pripojené. Touto funkciou bol znížený výdaj batérie až o 500%. [8]

Verzia 3.0 priniesla funkciu `High-speed (HS)`, teda vysoko-rýchlostný prenos dát. To docielila vďaka protokolu 802.11 a zvýšila rýchlosť pripojenia až 8 -násobne na 24 Mbps . [7]

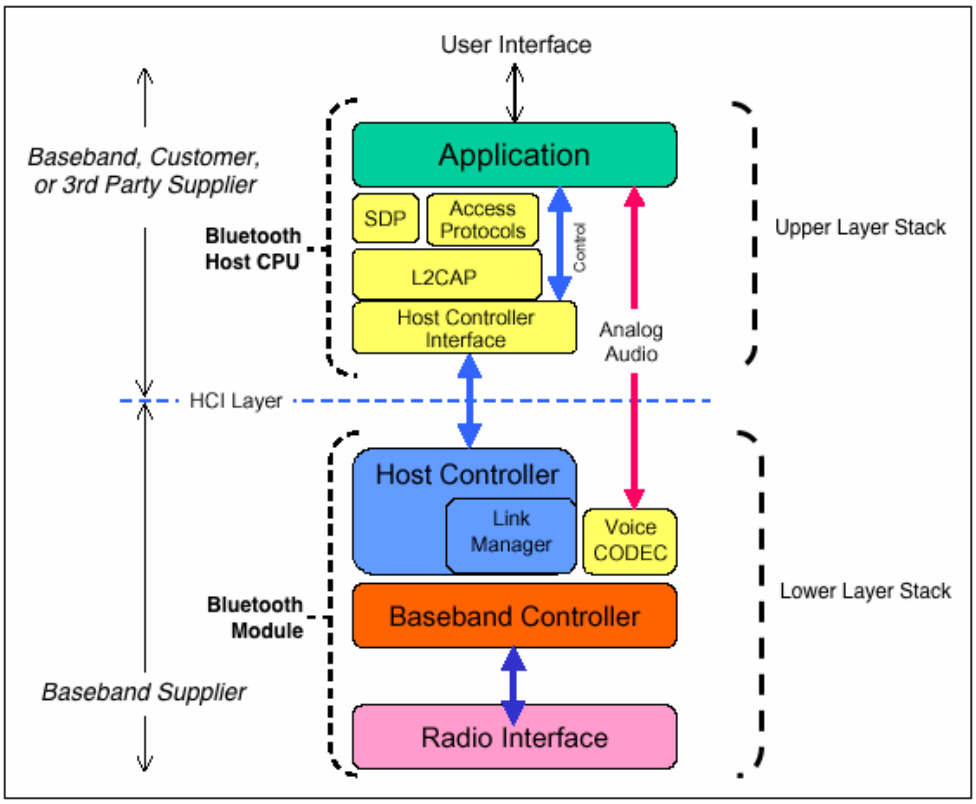
S verziou 4.0 prišla na trh technológia `Bluetooth Low Energy (BLE)`, ktorá ako z názvu vyplýva, funguje s nižšou spotrebou energie. Zariadeniam využívajúce tento typ komunikácie vydrží batéria oveľa dlhšie. Medzi tieto zariadenia patria napríklad chytré hodinky, ktoré sú dnes veľkým hitom, ale aj rôzne senzorové zariadenia. Využíva sa pri prenose malého množstva dát. [9]

Tabuľka 2.1: História rýchlosti a vzdialenosti pripojenia Bluetooth [10]

	Maximálna rýchlosť pripojenia	Maximálna vzdialenosť pripojenia
Bluetooth 1.0 (1999)	0.7 Mbps	10 m
Bluetooth 2.0 + EDR (2004)	1 Mbps 2.1 Mbps s EDR	30 m
Bluetooth 2.1 + EDR (2004)	1 Mbps 3 Mbps s EDR	30 m
Bluetooth 3.0 + HS (2009)	3 Mbps s EDR	30 m
Bluetooth 4.0 + LE (2013)	3 Mbps s EDR 1 Mbps s Low Energy	60 m
Bluetooth 5 (2017)	3 Mbps s EDR 2 Mbps s Low Energy	240 m

2.1.3 Architektúra Bluetooth

Architektúra využitá v technológií Bluetooth je rozdelená na dve hlavné časti pozostávajúce z hornej a dolnej vrstvy. Horná vrstva je riadená hostiteľom, ktorý zabezpečuje spoluprácu rôznych protokolov, ide hlavne o adaptačný protokol (L2CAP) a `Service Discovery Protocol (SDP)`. Všetky funkcie vykonané na tejto vrstve má na starosti samotné zariadenie (smartfón, laptop). Dolnú vrstvu riadi modul, ktorý je priamo integrovaný v zariadení alebo môže byť zapojený externe, napríklad cez USB. Táto vrstva má na starosti samotný bezdrôtový prenos pomocou rádiových vĺn. Komunikáciu medzi týmito dvoma vrstvami riadi `host controller`. Celá architektúra je znázornená na obrázku 2.4. [11]



Obr. 2.4: Architektúra Bluetooth opisujúca vzťah medzi zariadením a Bluetooth modulom [11]

Fyzická linka využíva dva typy spojení:

- Synchronous Connection-Oriented (SCO)
- Asynchronous Connection-Less (ACL)

`Master` môže pri komunikácií využiť až 3 SCO spojenia s jedným až troma `slave` zariadeniami. Využíva sa hlavne pri Bluetooth zariadeniach na hlasovú komunikáciu. ACL sa využíva pri dátovom prenose v sieti `piconet` medzi všetkými zúčastnenými zariadeniami. Môže existovať práve jedno ACL spojenie. ACL má podporu opakovaného prenosu paketov, zatiaľ čo SCO túto funkciu nepodporuje. [12]

2.2 Bezpečnosť Bluetooth komunikácie

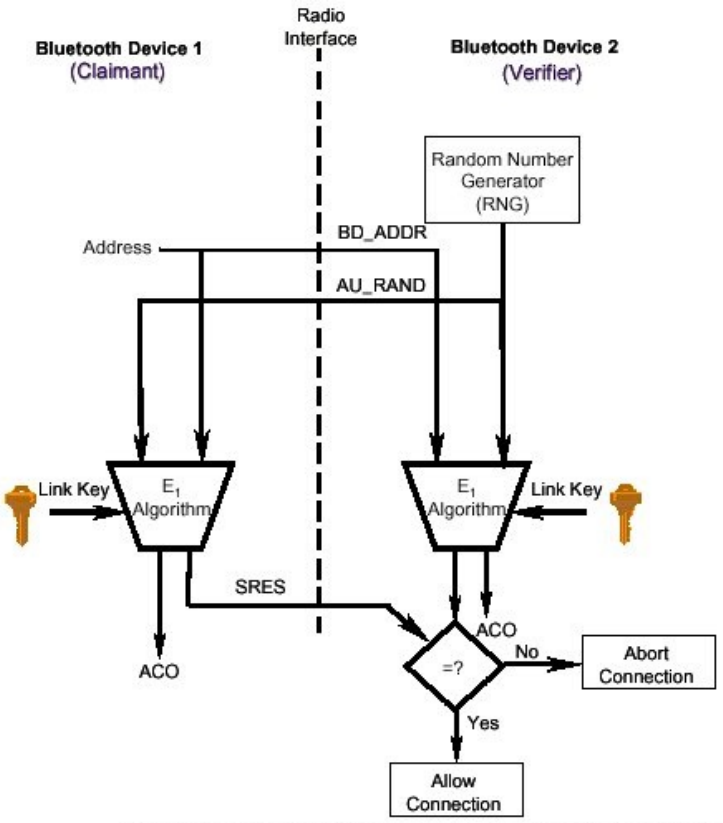
Technológia Bluetooth využíva na komunikáciu bezdrôtové pripojenie prenášané rádiovými vlnami, ktoré je ľahkým terčom hackerov. Z tohto dôvodu boli vývojári tejto technológie, ktorú využíva obrovský počet moderných zariadení, nútení dbať na bezpečnosť svojho produktu.

Medzi základné prvky bezpečnosti technológie Bluetooth patrí:

Autentifikácia – Overuje, či zariadenia poznajú linkový kľúč. Zariadenie pri pokuse o komunikáciu pošle svoju Bluetooth adresu druhému zariadeniu, ktoré spätne pošle náhodne vygenerovanú 128-bitovú hodnotu. Potom sa spustí na oboch zariadeniach šifrovací algoritmus `E1` na túto náhodnú hodnotu, adresu a linkový kľúč, Výsledkom je 32-bitová hodnota, ktorá sa na konci porovná, pozri obrázok 2.5. [13]

Šifrovanie – Zabraňuje úniku dát pri komunikácií tým, že zaručuje, aby mali prístup k preneseným dátam iba autorizované zariadenia cez nejaký druh šifrovania. Tento prvok je bližšie opísaný v sekcii 2.2.3.

Autorizácia – Proces, ktorý má na starosti kontrolu, či bolo zariadenie pred pripojením ku komunikácií autorizované na využívanie služieb pomocou autentifikácie. [14]



Obr. 2.5: Autentifikácia Bluetooth zariadení [13]

Každé Bluetooth zariadenie využíva pre svoju bezpečnosť jeden zo štyroch bezpečnostných módov:

Mód 1 – Tento mód nie je bezpečný, nevyužíva žiaden prvok bezpečnosti. Zariadenia, ktoré ho používajú nie sú ničím chránené. Tento mód je chránený v praxi jedine na miestach, kde si je používateľ istý, že sa do jeho blízkosti nedostanú žiadne iné zariadenia. Je podporovaný iba verziou Bluetooth 2.0 + EDR a staršími verziami.

Mód 2 – Pri tomto móde má na starosti bezpečnosť aplikácia, ktorej softvér sa rozhodne až po spojení zariadení, či chce využiť nejaký zo základných prvkov opísaných v sekcii 2.2. Bezpečnostné procesy sú spustené až po fyzickom a logickom spojení.

Mód 3 – Zariadenia s týmto módom využívajú na spojenie autentifikáciu a ich komunikácia je zašifrovaná. Bezpečnosť je zaručená ešte pred samotným fyzickým spojením, vďaka čomu sa dá na niektorých zariadeniach nastaviť viditeľnosť.

Mód 4 – V poslednom rade je mód 4, ktorý ako mód 2 spustí bezpečnostné procesy až po fyzickom a logickom spojení. Mód je podporovaný verziou Bluetooth 2.1 + EDR a novšími verziami. Tento mód využíva metódu párovania `Secure Simple Pairing (SSP)` opísanú v sekcii 2.2.2. Pri tomto móde nemusí byť linkový kľúč autentifikovaný.

Dôležitou súčasťou autentifikácie a šifrovania je vytvorenie linkového kľúča. V bezpečnostných módoch 2 a 3 je tento kľúč vytvorený pomocou `PIN` kódu, ktorý sa zadáva pri párovaní zariadení. V móde 4 sa na vytvorenie tohto kľúča využíva metóda `SSP`. [15] [16]

2.2.1 Párovanie pomocou PIN kódu

Na jednoduché, rýchle a bezpečné pripojenie zariadení sa pri technológií Bluetooth využíva metóda párovania. Hlavnou úlohou párovania je vytvorenie linkového kľúča. Základným krokom je vyplnenie správneho `PIN` kódu, ktorý je nastavený v zariadení. Predvolená hodnota `PIN` kódu býva nastavená na 0000. Po úspešnom zadaní `PIN` kódu si zariadenia vytvoria linkový kľúč, ktorý bude potrebný na komunikáciu zariadení. Táto metóda sa využíva vo verzii Bluetooth 2.0 + EDR a v starších verziách. [13]

2.2.2 Secure Simple Pairing

Táto metóda je nevyhnutnou súčasťou Bluetooth verzie 2.1 + EDR a novších verzií. Tieto verzie sú schopné využívať staršiu metódu párovania pomocou PIN kódu iba so staršími verziami, kde nie je SSP podporované. [17] Táto metóda je flexibilná a závisí od možností zariadenia, ako napríklad či má zariadenie klávesnicu alebo displej. Táto metóda sa využíva s bezpečnostným módom 4.

Modely, ktoré využíva SSP sú:

Jednoducho funguje – Ako vyplýva z názvu tento, model je ten najjednoduchší. Využíva sa, keď aspoň jedno zo zariadení nemá k dispozícií ani displej ani klávesnicu. Tento model ako jediný nezaisťuje autentifikovaný linkový kľúč.

Číselne porovnanie – Tento model sa používa, ak sú obidve zariadenia schopné na displeji zobraziť 6-ciferné číslo a majú tlačidlá na potvrdenie a zamietnutie. Pri pokuse o spárovanie sa na displeji oboch zariadení zobrazí číslo, ktoré používateľ porovná a ak sa zhodujú, môže párovanie potvrdiť. Tento mód sa líši od párovania pomocou `PIN` kódu tým, že sa v tomto prípade číslo (kód) nevyužije na vytvorenie linkového kľúča, takže hacker nie je schopný zistiť linkový a šifrovací kľúč na základe tohto čísla.

Zadanie univerzálneho kľúča – Pri tomto modeli musia mať obidve zariadenia displej a klávesnicu, na ktorých je schopné zobraziť a zadať 6-ciferné číslo. Požívateľ musí pre spárovanie zariadení zadať na obidvoch zariadeniach to isté číslo. Ak jedno zo zariadení nemá takú klávesnicu, toto číslo sa vygeneruje a zobrazí na displeji, ktoré potom zadá na druhom zariadení. Toto číslo taktiež nie je súčasťou linkového a šifrovacie kľúča.

Out of Band (OOB) – Ak zariadenia podporujú iný druh drôtového alebo bezdrôtového pripojenia, využije sa tento model. Napríklad pri technológií NFC, ktorá sa využíva v bezkontaktných kreditných kartách a niektorých smartfónoch, sa zariadenia spárujú pri blízkom fyzickom kontakte. [17] [16]

2.2.3 Šifrovanie

Keď chcem odosielať správu s tým, aby nikto iný okrem mňa a prijímateľa nevidel jej obsah, je nutné takúto správu zašifrovať. Takisto aj Bluetooth bojuje proti riziku úniku prenesených dát pomocou šifrovania.

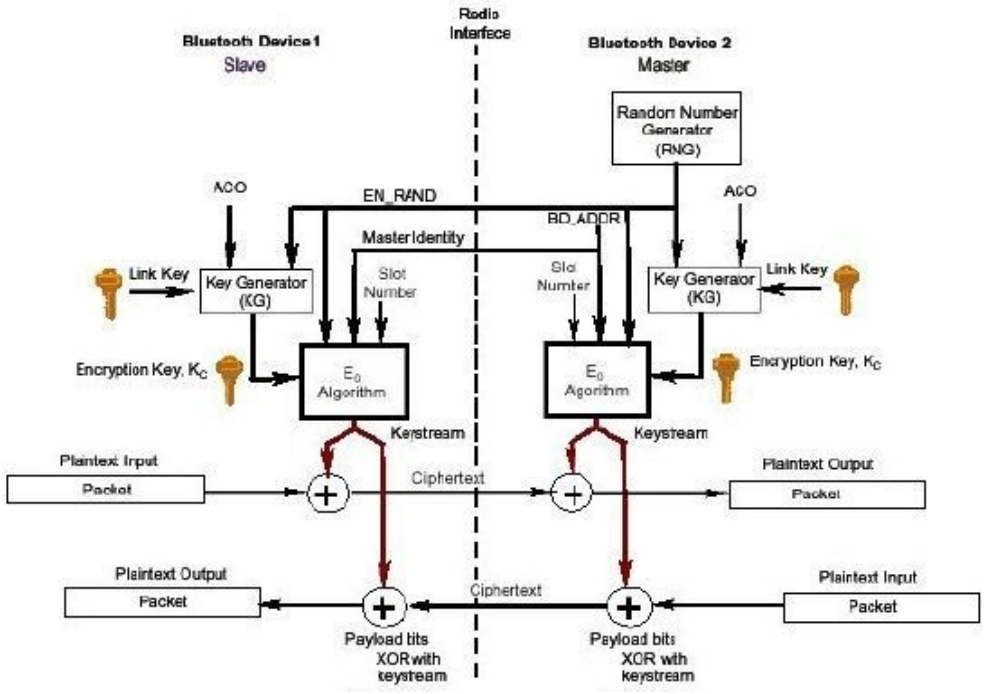
Okrem módov bezpečnosti, Bluetooth ponúka 3 módy šifrovania:

Šifrovací mód 1 – Žiadne šifrovanie.

Šifrovací mód 2 – Zašifrované sú iba vybrané prenosy pomocou šifrovacieho kľúča založeným individuálnymi linkovými kľúčmi.

Šifrovací mód 3 – Celý prenos je zašifrovaný šifrovacím kľúčom na základe linkového kľúča `master` zariadenia

Šifrovacie módy 2 a 3 využívajú ten istý algoritmus šifrovania. Ako je vidieť na obrázku 2.6, algoritmus zoberie na začiatku adresu zariadenia, náhodne vygenerovanú 128-bitovú hodnotu, číslo slotu a linkový kľúč. Na základe týchto hodnôt vygeneruje šifrovací kľúč. Tento kľúč je pre každý paket iný, pretože číslo slotu sa časovo mení. [16]



Obr. 2.6: Šifrovanie Bluetooth komunikácie [13]

2.3 Bluetooth na platforme Android

Technológia Bluetooth je nevyhnutnou súčasťou všetkých smartfónov. Používateľom dáva možnosť využívať rôzne príslušenstvá, ako napríklad bezdrôtové slúchadlá alebo chytré hodinky, ale aj notebook a iný smartfón. Funkcia Bluetooth komunikácie na tejto platforme je založená na párovaní zariadení. Pred tým, ako používateľ môže zahájiť bezdrôtovú komunikáciu s iným zariadením, je nútený s ním svoj smartfón spárovať.

Ako spárovať Android zariadenie s iným Bluetooth zariadením je opísané v týchto krokoch:

1. Otvorte nastavenia svojho Android zariadenia
2. Vyberte možnosti: `Pripojené zariadenia`, potom `Predvoľby pripojenia` a potom `Bluetooth`. Pokiaľ nevidíte možnosť `Predvoľby pripojenia` prejdite na 3. krok.
3. Vyberte možnosť `Spárovať nové zariadenie`. Týmto sa začne vyhľadávanie dostupných Bluetooth zariadení vo vašom okolí.
4. Zvoľte zariadenie, s ktorým sa chcete spárovať

S verziou Android 6.0 prišla nová funkcia rýchleho párovania. Používateľ sa musí uistiť, či jeho zariadenie, ktoré chce spárovať so svojim smartfónom, podporuje túto funkciu. Potom stačí na smartfóne zapnúť rozhranie Bluetooth a určovanie polohy. Po blízkom kontakte so zariadením pripraveným na párovanie vyskočí na smartfóne notifikácia s potvrdením. [18]

Vývojár mobilnej aplikácie Android môže na prístup k Bluetooth využiť `Android Bluetooth API`. Vďaka nemu môže jednoducho vyhľadať Bluetooth zariadenia v okolí, pomocou adaptéra zobraziť spárované zariadenia, pripojiť sa k zariadeniu, odosielať a prijímať dáta. Taktiež je možné sa spojiť a komunikovať s väčším množstvom zariadení. Od verzie Android 4.3 je možné využiť `BLE`. [19]

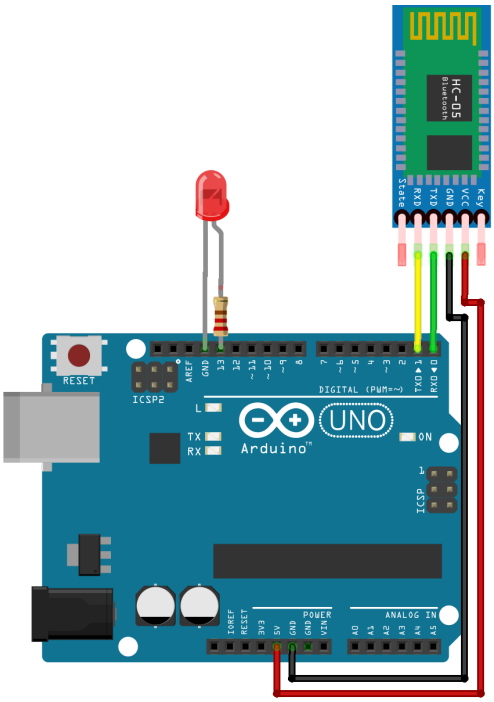
2.4 Bluetooth na platforme Arduino a iných zariadeniach Internetu vecí

Keďže na trhu nie sú k dispozícií Arduino zariadenia s integrovaným Bluetooth, je používateľ nútený k svojmu zariadeniu dokúpiť Bluetooth modul. Medzi najpredávanejšie moduly patria `HC-05` a `HC-06`. Modul `HC-05` na rozdiel od `HC-06` má resetovacie tlačidlo a môže slúžiť ako `master` zariadenie, zatiaľ čo `HC-06` môže byť iba `slave`. [20] Ako pripojiť Bluetooth modul do Arduino zariadenia je nakreslené na obrázku 2.7.

Existujú však zariadenia podobné Arduinu, ako napríklad `Raspberry Pi 3` a `ESP32`, ktoré sú na rozdiel od Arduina s integrovaným Bluetooth. V obidvoch zariadeniach je Bluetooth 4.2, ktorý podporuje aj `BLE`. [21] [22]

Pre spárovanie so zariadeniami Internetu vecí (IoT) je niekedy potrebné zadať `PIN` kód, ktorý býva prednastavený na 0000 alebo 1234. Po zadaní správneho `PIN` kódu sa zariadenia spárujú. Pri niektorých moduloch obsahujúcich blikajúce `LED` svetlo sa po nadviazaní spojenia zmení frekvencia blikania.

Komunikácia s iným zariadením pripojeným pomocou Bluetooth prebieha pomocou sériovej linky.



Obr. 2.7: Pripojenie Bluetooth modulu do Arduino zariadenia [23]

Kapitola 3

Analýza

Táto kapitola sa zaoberá analýzou platforiem Android a Arduino. Pre obidve platformy sú zanalyzované správne postupy pri tvorbe softvéru na tieto platformy. Pre platformu Android sú taktiež zanalyzované existujúce riešenia.

3.1 Mobilná aplikácia Android

Táto sekcia je zameraná na analýzu tvorby mobilnej aplikácie Android. Ako prvé som sa zameral na analýzu existujúcich riešení. Na základe toho budú vytvorené funkčne požiadavky, ktoré implementujem v návrhu mojej aplikácie. sú taktiež zanalyzované vývojové prostredia a typy architektúr pri vývoji Android aplikácie.

3.1.1 Analýza existujúcich riešení

Komunita zaoberajúca sa vývojom na platforme Android je veľmi rozsiahla, preto je na internete veľké množstvo tutoriálov, knižníc a aplikácií. Pred realizáciou vlastného návrhu som sa zameral na vyhľadávanie a analýzu aplikácií a knižníc tretej strany pre Android, ktoré sa zaoberajú komunikáciou s Arduinom pomocou pripojenia Bluetooth.

3.1.1.1 Aplikácie

Všetky aplikácie som vyhľadával pomocou `Google Play Store`.

Vybral som a zanalyzoval tie, ktoré mali najviac stiahnutí a boli používateľmi najlepšie hodnotené:

Arduino bluetooth controller [24] – Po otvorení aplikácie sa zobrazí zoznam spárovaných Bluetooth zariadení. Po vybraní zariadenia si používateľ zvolí,či chce s Arduinom komunikovať pomocou rôznych foriem, napríklad tlačidla, slideru alebo terminálu. Keď chce používateľ zmeniť formu komunikácie, musí sa vrátiť späť do zoznamu spárovaných zariadení a znova vybrať zariadenie zo zoznamu, čo spôsobuje odpojenie Arduina.

Arduino bluetooth [25] – Aplikácia funguje na podobnom princípe ako aplikácia `Arduino bluetooth controller`. Rozdiel je v tom, že keď chceme zmeniť formu komunikácie, nemusíme ísť späť do zoznamu zariadení, takže sa neodpojí spojenie s Arduinom.

Arduino Bluetooth Terminal [26] – Zaujímavá jednoduchá aplikácia, kde po vybraní zariadenia zo zoznamu spárovaných Bluetooth zariadení má používateľ na výber z dvoch možností. Prvá možnosť je `receiver`, kde sa zobrazujú dáta prijaté z Arduina. Druhá je terminál, ktorý je zaujímavý hlavne vzhľadom a pripomína `chat`. Aplikácia neobsahuje žiadne tlačidlá na ovládanie.

Arduino Bluetooth Serial monitor [27] – Jednoduchá aplikácia, v ktorej cez terminál komunikujeme s jedným zo spárovaných Bluetooth zariadení.

Zanalyzované aplikácie mali jednoduchý a prehľadný dizajn. Avšak po dôkladnom naštudovaní uvedených aplikácií som zistil, že žiadna z nich nemá funkciu komunikácie s viacerými zariadeniami Arduino. Ďalším nedostatkom bola chýbajúca možnosť uložiť si svoje nastavenia.

3.1.1.2 Knižnice

Našiel som desiatky knižníc tretej strany, ktoré zjednodušujú prácu s Bluetooth pripojením a komunikáciou s iným zariadením. Mojim cieľom bolo nájsť knižnicu, ktorú by som mohol využiť pri návrhu mojej aplikácie.

Na základe využiteľnosti som vybral tieto knižnice:

Android Bluetooth Library [28] – Jednoduchá knižnica, ktorá vytvorí spojenie so spárovaným zariadením, vytvorí triedu `BluetoothSocket`, ktorým komunikuje obojstranne pomocou triedy `BufferedReader`.

BluetoothKit [29] – Taktiež veľmi jednoduchá knižnica, ktorá sa stará o pripojenie a komunikáciu cez triedu `BluetoothSocket`.

BluetoothHelper [30] – Umožňuje prístup k Bluetooth, ovláda jeho zapínanie a vypínanie. Taktiež slúži na vyhľadávanie Bluetooth zariadení v blízkosti zariadenia.

Android-Multi-Bluetooth-Library [31] – Knižnica slúžiaca na komunikáciu medzi väčším množstvom Android zariadení pomocou Bluetooth, kde sa jeden chová ako server a ostatní ako klienti.

Android-bluetooth-serial [32] – Slúži na komunikáciu s väčším množstvom spárovaných zariadení.

Zo všetkých zanalyzovaných knižníc najviac vyhovuje môjmu riešeniu knižnica `Android-bluetooth-serial`.

3.1.2 Funkčné požiadavky mobilnej aplikácie

Hlavnou funkciou mobilnej aplikácie je možnosť pripojiť sa pomocou technológie Bluetooth na spárované Arduino zariadenia a vymieňať si s nimi dáta. Po analýze existujúcich aplikácií v sekcii 3.1.1.1 som zvolil ako ďalšiu funkciu aplikácie možnosť pridať a odobrať používateľom upraviteľné komponenty na komunikáciu, ktoré si môže uložiť a neskôr načítať. Pri komunikácií s väčším množstvom zariadení je potrebné dbať na jednoznačné určenie, s ktorým zariadením chce používateľ práve komunikovať.

3.1.3 Vývojové prostredie

Na vývoj mobilných aplikácií Android sa používa vývojové prostredie `Android Studio`. Toto prostredie bolo vytvorené spoločnosťou `JetBrains` v spolupráci s `Google`. Obsahuje všetký potrebné funkcie k vývoju ako `debugger`, `logger` a `code editor`. Prostredie je možné nainštalovať na Windows, macOS a aj Linux, čo je obrovskou výhodou. Priamo v prostredí si môže vývojár nájsť a nainštalovať rôzne doplnky, ktoré pomáhajú pri vývoji.

Taktiež obsahuje `AVD manager`, ktorý slúži na spravovanie emulátorov. Emulátor je výborná pomôcka počas implementácie a testovania. Vývojár si tak môže vytvoriť Android zariadenie zo zoznamu, ktoré tento manažér ponúka. Takto si môže vytvoriť nielen Android smartfón, ale aj tablet, hodinky a televíziu. Okrem výberu presného typu zariadenia si používateľ môže vybrať zo všetkých verzii operačného systému Android. [33]

3.1.4 Architektúra

Pri vývoji softvéru je potrebné brať ohľad na čitateľnosť kódu. Preto je potrebné držať sa rôznych vzorov objektovo orientovaného programovania. Obzvlášť ak sa jedná o rozsiahly softvér, v ktorom sú implementované databáza, `business logika` a `UI`. Takýto softvér je aj mobilná aplikácia Android, kde je väčšinou samotné `UI` najdôležitejšia súčasť aplikácie. Pre lepšiu obsluhu, rozširovanie a testovanie kódu aplikácie sa databáza, `business logika` a `UI` zvyknú rozdeliť do vrstiev.

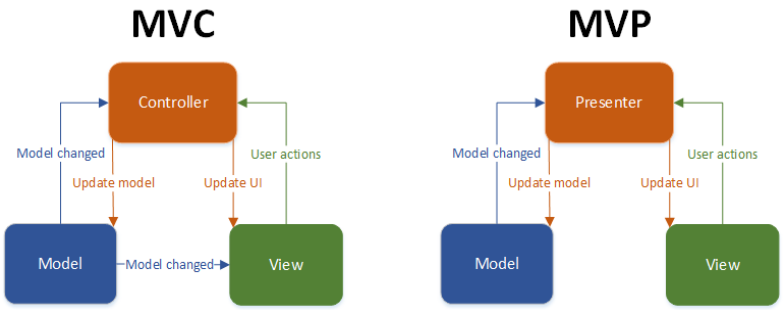
Taktiež z dôvodu životného cyklu Android aplikácií je potrebné správne implementovať `UI` vrstvu a dáta s ním viazané. Na to je možné použiť niektorú z týchto architektúr:

MVC – Model - View - Controller

MVP – Model - View - Presenter

MVVM – Model - View - Viewmodel

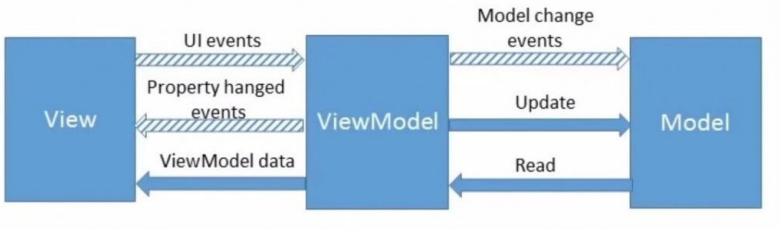
Architektúry `MVC` a `MVP` sú veľmi podobné. V obidvoch prípadoch je `Model` poskytovateľ dát a ako jediný komponent komunikuje so samotnou databázou. `View` je komponent, ktorý má na starosti zobrazovanie dát používateľovi, ktoré poskytuje `Model` a nemala by v ňom byť implementovaná žiadna `business logika`. `Controller` a `Presenter` sú komponenty, v ktorých je implementovaná všetka `business logika`. [34] [35]



Obr. 3.1: Diagram MVC a MVP architektúr [36]

V architektúre `MVC` komponent `Model` upozorňuje na svoje zmeny samotný `View`, kvôli tomu musí byť vo `View` implementovaná logika, ktorá je schopná reagovať na tieto zmeny. V `MVP` je tento proces rozdielny a zmeny v komponente `Model` sleduje `Presenter` a ten potom upozorňuje `View`. Tým sa `View` vyhne všetkej logike a môže sa venovať iba svojej funkcií a to zobrazovaniu dát. Celý proces komunikácie medzi komponentami obidvoch architektúr je znázornený na obrázku 3.1. [36]

Architektúra `MVVM` je podobná `MVP` s tým rozdielom, že medzi komponentami `View` a `Model` je `ViewModel`, ktorý na rozdiel od komponentu `Presenter` nemá referenciu na `View`. Výhodou tejto architektúry je natívna podpora pri implementácií na Android a jej jednoduché testovanie.



Obr. 3.2: Diagram MVVM architektúry [37]

3.2 Program pre zariadenie Arduino

Program pre zariadenie Arduino je písaný v jazyku C, respektívne C++. Namiesto funkcie `main`, ktorá je automaticky volaná po spustení programov implementovaných v týchto jazykoch, Arduino po spustení zavolá funkciu `setup`, ktorá môže slúžiť napríklad na inicializáciu objektov. Po tejto funkcií sa v nekonečnom cykle volá funkcia `loop`, v ktorej je implementovaná väčšina logiky, ako napríklad interakcia s používateľom a následná reakcia Arduino zariadenia. Samozrejme je možné vytvárať iné funkcie, premenné, globálne premenné a využívať rôzne prvky objektovo orientovaného programovania.

3.2.1 Vývojové prostredie

Na vývoj programov pre Arduino sa používa vývojové prostredie `Arduino IDE`. Toto prostredie umožňuje písanie programu a jeho nahratie na Arduino zariadenie. Ďalšou funkciou je možnosť sťahovania rôznych knižníc pomocou manažéra knižníc. Vývojár si môže toto prostredie stiahnuť na oficiálnej stránke. Taktiež môže využiť online verziu, ktorú otvorí vo svojom prehliadači. Tým má vývojár zaručené ukladanie programov do `cloud` úložiska a taktiež bude vždy vyvíjať v najnovšej verzií prostredia. [38]

Kapitola 4

Návrh

Táto kapitola je zameraná na návrh platformy pre komunikáciu medzi Android a väčším množstvom Arduino zariadení pomocou bezdrôtového pripojenia Bluetooth. Samotná Android aplikácia bude rozsiahlejšia ako Arduino program. Z toho dôvodu bude väčšia časť tejto kapitoly zameraná práve na návrh tejto aplikácie.

4.1 Návrh mobilnej aplikácie Android

V tejto sekcii je opísaný návrh mobilnej aplikácie Android, ktorej primárnou funkciou je nadviazať spojenie s väčším množstvom spárovaných Arduino zariadení pomocou technológie Bluetooth a následne s nimi komunikovať.

4.1.1 Funkcie

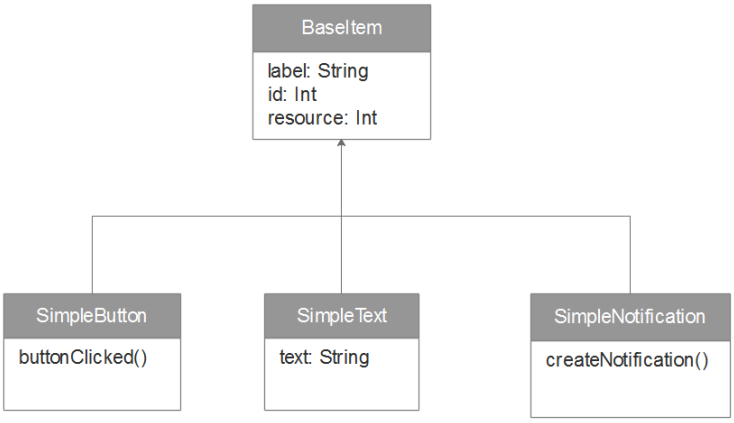
Pred samotným návrhom bolo potrebné zanalyzovať mobilné aplikácie s podobnou funkcionalitou. Tieto aplikácie sú zanalyzované v sekcii 3.1.1.1. Na základe tejto analýzy boli vytvorené funkčné požiadavky mnou navrhnutej mobilnej aplikácie, ktoré sú opísané v sekcii 3.1.2.

4.1.1.1 Spojenie s Arduino zariadeniami

Pred samotnou komunikáciou musí používateľ vybrať zo zoznamu spárovaných zariadení, s ktorými zariadeniami chce komunikovať. Samotné spárovanie zariadení je potrebné vykonať mimo aplikácie v nastaveniach Android zariadenia, viac v sekciách 2.3 a 2.4.

4.1.1.2 Komunikačné komponenty

Používateľ si môže vytvoriť rôzne komponenty slúžiace na komunikáciu s Arduinom pomocou Bluetooth. Komponent je po uložení umiestnený na jednej z troch stránok, ktoré môže používateľ využívať. Samotný návrh týchto komponentov je znázornený na obrázku 4.1.



Obr. 4.1: Doménový model komponentov

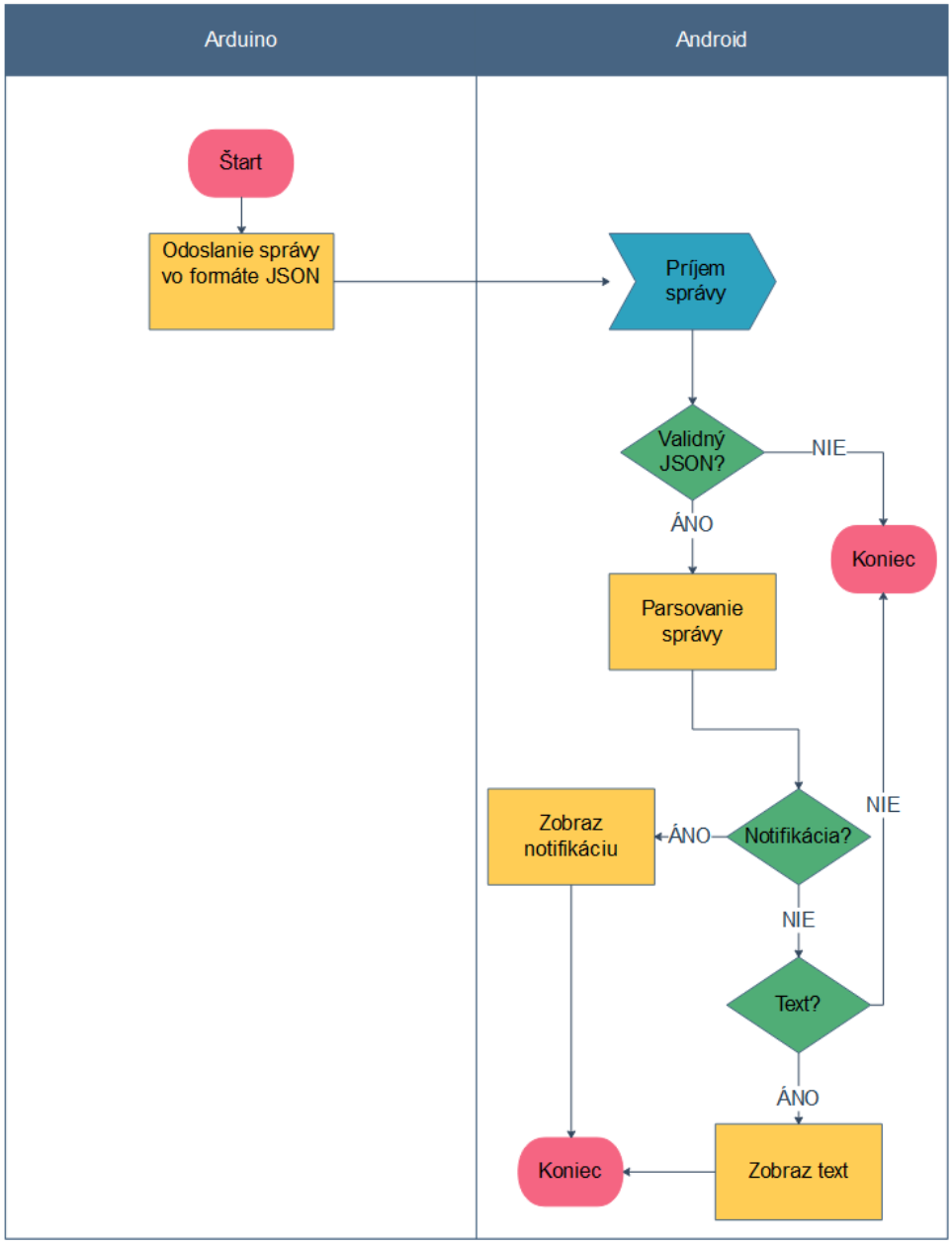
Používateľ má na výber z troch komponentov:

Button – Tlačidlo so zvoleným názvom. Po kliknutí sa pošle správa všetkým pripojeným zariadeniam.

Text – Komponent zložený z dvoch `TextView`. Jeden je názov komponentu, ktorý si zvolil používateľ. Pod ním je druhý, ktorý vypíše prijatú správu.

Notification – Tento komponent sa vizuálne javí iba ako jeden `TextView` so zvoleným názvom. Po prijatí správy sa zobrazí notifikácia so správou.

Celá komunikácia prebieha správami vo formáte `JSON`. Každá prijatá sprava z Arduina obsahuje `id` komponentu, pomocou ktorého sa jasne určí, ktorý komponent má zobraziť správu pod hodnotou kľúča `text`. Prijatie správy z Arduina je znázornené na diagrame 4.2.



Obr. 4.2: Prijatie správy z Arduina v Android aplikácií

4.1.1.3 Ukladanie komponentov

Ďalšou dôležitou funkciou je ukladanie komponentov. Používateľ má po vytvorení rozhrania, tvoreného zo zvolených a upravených komponentov, možnosť uložiť si toto sebou vytvorené rozhranie. Samozrejme môže toto rozhranie neskôr načítať alebo odstrániť. Komponenty sa ukladajú do natívnej databázy Android aplikácie `SharedPreferences` vo forme `JSON`. Táto databáza sa vymaže, keď sa používateľ rozhodne vymazať aplikáciu, alebo v nastaveniach Android zariadenia vyčistí dáta danej aplikácie.

Vo vysúvacom menu má používateľ na výber z týchto funkcií:

Save – Uloží dané rozhranie. Ak rozhranie pred tým nebolo uložené, správa sa ako funkcia `Save as`.

Save as – Je potrebné zadať názov, pod ktorým bude uložené rozhranie.

Load – Zobrazí všetky uložené rozhrania podľa zvoleného názvu pri uložení.

Remove – Odstráni práve zvolené rozhranie.

Devices – Zobrazí zoznam spárovaných zariadení, viac v sekcii 4.1.1.1.

4.1.2 Kotlin

Pred samotným vývojom aplikácie je potrebné vybrať programovací jazyk. Pri vývoji Android aplikácií je možné vybrať si z dvoch jazykov. Jedným z nich je jazyk `Java`, v ktorom je vyvinutá väčšina dnešných aplikácií a knižníc. Pre vývoj Android aplikácie je nevyhnutné ovládať tento jazyk.

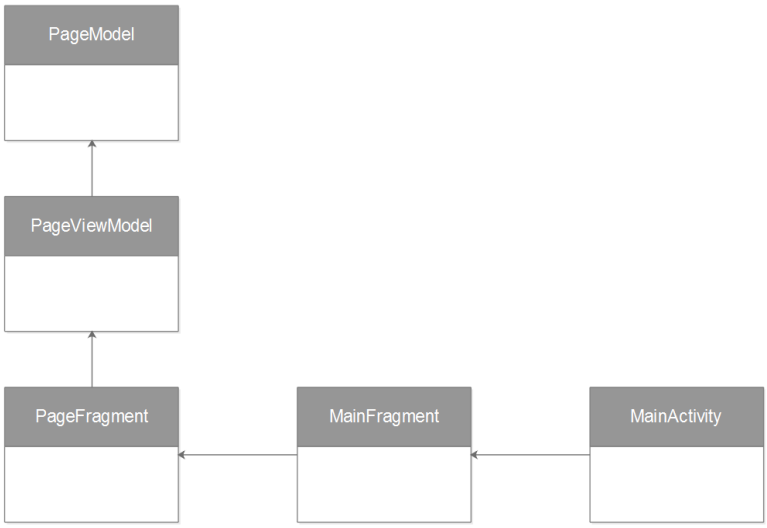
Druhý jazyk je `Kotlin`, ktorý sa pri vývoji Android aplikácií stal veľmi populárnym. Samotný `Google` oznámil na konferencií v máji 2019, že `Kotlin` sa oficiálne stáva preferovaným jazykom pre vývoj Android aplikácií. Stalo sa tak iba 2 roky po tom, ako sa tento jazyk stal plne podporovaný vo vývojovom prostredí `Android Studio`. Podľa štatistík až polovica profesionálnych vývojárov na platforme Android používa tento jazyk. `Kotlin` je interoperabilný s jazykom `Java`. To znamená, že ak sa rozhodne vývojár písať v jazyku `Kotlin`, môže využívať knižnice napísané v jazyku `Java`. [39] [40] [41]

Z toho vyplýva, že `Kotlin` je jednoznačne budúcnosť pri vývoji mobilných aplikácií Android. Ďalšou výhodou je kratší zdrojový kód, `null safety` a `lambda funkcie`. Na základe toho som sa rozhodol vyvíjať aplikáciu práve v tomto jazyku.

4.1.3 Architektúra

Všetky typy architektúr, ktoré sa používajú pri vývoji Android aplikácií sú opísané a zanalyzované v sekcii 3.1.4. Po analýze týchto architektúr som na vývoj mojej aplikácie zvolil architektúru `MVVM`. Táto architektúra je odporúčaná pri vývoji Android aplikácií. Hlavnou výhodou je jednoduchá implementácia, testovanie a natívna podpora. Ďalšou výhodou je využitie knižnice `Data Binding`, ktorá uľahčuje prepojenie `UI` komponentov s dátami na zobrazenie.

Moja aplikácia je navrhnutá tak, ako je vidieť na diagrame 4.3, že neobsahuje veľký počet `UI` obrazoviek (aktivita alebo fragment). Síce sa to pri tak malom počte nevyžaduje, no aj napriek tomu som sa rozhodol použiť tento typ architektúry.



Obr. 4.3: Diagram MVVM architektúry použitej v aplikácií

4.1.4 Verzia systému

Každá nová verzia operačného systému Android so sebou prináša nové funkcie a zmeny. Vývojári sa viac stretávajú s označením `API level` alebo `SDK verzia`.

Vývojár je povinný nastaviť pre aplikáciu tieto hodnoty:

Minimum SDK version – Určuje najnižšiu možnú verziu Android zariadenia, na ktorú je možne nainštalovať aplikáciu. Pri výbere tejto verzie treba brať ohľad na oficiálne štatistky používaných Android zariadení.

Target SDK version – Touto verziou označuje vývojár, pre ktorú verziu Android zariadenia je vývoj aplikácie cielený. Hodnota tejto verzie musí byť väčšia alebo rovná `Minimum SDK version`. Pri nahraní aplikácie na `Google Play Store` je podmienka minimálnej hodnoty tejto verzie.

Compile SDK version – Hodnota tejto verzie musí byť rovnaká ako hodnota `Target SDK version`.

V čase vývoja tejto aplikácie využívalo verziu Android 6.0 a novšie 74,8 % všetkých používaných Android zariadení. [42] Táto aplikácia je určená pre klientelu tvorenú technicky zdatnými ľuďmi. Preto predpokladám, že ich verzia Android zariadenia bude dostatočná. Z toho dôvodu som sa rozhodol nastaviť hodnotu `Minimum SDK version` na 23.

Od 1. 8. 2019 musia mať všetky nové aplikácie nahrané na `Google Play Store` nastavenú hodnotu `Target SDK version` minimálne na 28. Vzhľadom na to, že mnou navrhnutá aplikácia je nová a hodnota `Target SDK version` nijako neovplyvní samotný vývoj aplikácie, rozhodol som sa nastaviť túto hodnotu na 28.

4.2 Návrh programu pre zariadenie Arduino

Táto sekcia sa zaoberá návrhom programu na Arduino. Hlavnou myšlienkou pri návrhu bolo vytvoriť jednoducho rozšíriteľný program.

4.2.1 Funkcie

Tento program je schopný prijímať správy z mobilnej aplikácie Android vo formáte `JSON`. Program obsahuje validáciu správy, no následné parsovanie a vykonanie funkcií je na samotnom vývojárovi. Program je tiež schopný odosielať správy. Táto správa je taktiež vo formáte `JSON`.

Správa na odoslanie musí obsahovať 2 kľúče:

id – Hodnota tohto kľúča je typu `int`, čiže celé číslo. Toto číslo zodpovedá zadanému `id` pri tvorbe komunikačných komponentov `Notification` alebo `Text`, ktoré sú navrhnuté v sekcii 4.1.1.2.

text – Tento kľúč obsahuje hodnotu typu `String`. Jedná sa o samotný text správy, ktorý sa má zobraziť.

Po odoslaní správy sa táto funkcia na Android zariadení príjme a zvaliduje. Tento proces je vysvetlený v sekcii 4.1.1.2.

4.2.2 Spojenie s Android zariadením

Arduino je potrebné pred samotnou komunikáciou spárovať s Android zariadením. Toto párovanie je potrebne vykonať v nastaveniach Android zariadenia, pozri sekciu 4.1.1.1. Možnosti využitia bezdrôtového pripojenia pomocou technológie Bluetooth na platforme Arduino sú opísané v sekcii 2.4.

Kapitola 5

Implementácia

V tejto kapitole sú opísané použité knižnice v mobilnej aplikácií Android a programe pre zariadenie Arduino. Taktiež obsahuje zoznam použitých technológií a postupy riešenia dôležitých funkcií.

5.1 Mobilná aplikácia Android

V tejto sekcii je okrem zoznamu knižníc a technológií, opísaný postup riešenia pri pripojení a komunikácií s väčším množstvom zariadení Arduino pomocou technológie Bluetooth.

5.1.1 Použité knižnice

Pri vývoji Android aplikácií je možné využiť veľké množstvo knižníc. Počas analýzy som sa zameral na analýzu knižníc tretej strany, ktoré uľahčujú implementáciu Bluetooth pripojenia. Analýza týchto knižníc je uvedená v sekcii 3.1.1.2. Nakoniec som sa rozhodol nepoužiť ani jednu z týchto knižníc a využiť iba natívne triedy Bluetooth knižnice. Rozhodol som sa však použiť iné knižnice, ktoré mi uľahčili prácu pri implementovaní a testovaní.

5.1.1.1 Android JetPack

`Android Jetpack` [43] je kolekcia väčšieho množstva Android knižníc a tried, ktoré využíva každý vývojár na tejto platforme. Hlavným cieľom je zjednodušiť vývoj zložitejších problémov pri vývoji Android aplikácií. Medzi tieto problémy patrí napríklad implementácia architektúry, komunikácia s Android servismi (notifikácie, povolenia) a taktiež zjednodušuje implementáciu `UI`.

V mojej aplikácií mi táto knižnica pomohla hlavne pri implementácií architektúry MVVM, ktorá je navrhnutá v sekcii 4.1.3. Pri implementácií som využil nasledujúce triedy a knižnice:

LiveData – Trieda s podobnou funkciou ako trieda `Observable`. To znamená, že upozorňuje sledovateľa na zmenu dát. Táto trieda však pozná logiku životného cyklu Android aplikácie a upozorňuje na zmeny iba sledovateľov v aktívnom stave. [44] Túto triedu som využil pri všetkých premenných, ktorých zmena ovplyvňuje `UI` alebo nejakú funkciu aplikácie.

Data Binding – Táto knižnica uľahčuje prepojenie `UI` komponentov s dátami na zobrazenie. Namiesto prepojenia v zdrojovom kóde aktivity, fragmentu alebo iného `UI` komponentu, je toto prepojenie realizované priamo v `XML layout`. [45] Toto prepojenie som využil pri implementácií komunikačných komponentov, ktoré sú navrhnuté v sekcii 4.1.1.2. Implementácia tejto knižnice v mojej aplikácií je znázornená v zdrojovom kóde 1 reprezentujúci `XML layout` komunikačného komponentu `Text`.

```
{
   <data>
        <variable
                name="item"
                type="cz.cvut.fit.aba.item.SimpleText"/>
    </data>
    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="@{item.label}"/>

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:textSize="16sp"
            android:text="@{item.text}"/>
}
```

Zdrojový kód 1: Ukážka implementovanej knižnice Data Binding

ViewModel – Trieda vytvorená na uchovávanie a spravovanie dát spojených s `UI`. Vďaka tejto triede je možné jednoducho zachovať dáta pri rotácií obrazovky Android zariadenia. [46] Táto trieda je súčasťou architektúry MVVM, ktorú som zanalyzoval v sekcii 3.1.4. Na základe analýzy vznikol návrh s využitím tejto triedy pre moju aplikáciu v sekcii 4.1.3.

5.1.1.2 Timber

Táto knižnica je určená pre zjednodušenie testovania počas implementácie zdrojového kódu. Je to jednoduchá nadstavba triedy `Log`, ktorá ma na starosti výpis v `logger` vývojového prostredia. [47]

5.1.1.3 Tray

`Tray`[48] je knižnica, ktorá uľahčuje prácu s `SharedPreferences`. Táto knižnica je použitá pri ukladaní komunikačných komponentov, ktorých návrh je opísaný v sekcii 4.1.1.3. Je použitá napríklad pri funkcií odstránenia rozhrania komunikačných komponentov, ktorá je znázornená v zdrojovom kóde 2.

```
fun removeData(name:String, context: Context) {
    val appPreferences = AppPreferences(context)
    val files = appPreferences.getString("files", "empty")

    val jsonArray = if (files.equals("empty")) JSONArray()
                    else JSONArray(files)

    val newArray = JSONArray()

    for (i in 0 until jsonArray.length()) {
        val file = jsonArray.getJSONObject(i)
        if (name != file.getString("name")) {
            newArray.put(file)
        }
    }
    appPreferences.put("files", newArray.toString())
}
```

Zdrojový kód 2: Funkcia na odstránenie rozhrania komunikačných komponentov

5.1.1.4 Anko

`Anko` je `Kotlin` knižnica pre vývoj Android aplikácií, ktorá výrazne zjednodušuje vývoj na tejto platforme. [49] V Android aplikácií sa všetky zmeny v `UI` vykonávajú na hlavnom vlákne a všetky ostatné zložité funkcie by mali prebiehať na iných vláknach. Tým sa zabezpečí plynulosť aplikácie. Na to je potrebné použiť asynchrónne programovanie. Vďaka tejto knižnici je to oveľa jednoduchšie, pozri zdrojový kód 3.

5.1.2 Pripojenie Arduino zariadenia

Pri implementácií pripojenia s Arduino zariadením pomocou technológie Bluetooth som využil natívne triedy tejto technológie pre vývoj Android aplikácií. Pomocou týchto tried je možné spojiť sa so spárovaným zariadením a následne s ním komunikovať. Zariadenia však používateľ musí spárovať v samotných nastaveniach svojho smartfónu, pozri sekciu 2.3.

Pri implementácií boli využité tieto dve triedy:

BluetoothDevice – Táto trieda reprezentuje Bluetooth zariadenie a obsahuje jeho názov a adresu. Získava sa z objektu `BluetoothAdapter` pomocou adresy.

BluetoothAdapter – Trieda, ktorá slúži na vyhľadávanie spárovaných Bluetooth zariadení. Taktiež vracia pole adries a názvov spárovaných zariadení.

BluetoothSocket – Táto trieda sa využíva pri nadviazaní spojenia so spárovaným zariadením. Obsahuje triedy `OutputStream` a `InputStream`, pomocou ktorých je možné so vzdialeným zariadením komunikovať. Taktiež vyhadzuje výnimky pri zlyhaní komunikácie.

Všetky tieto triedy som využil vo funkcií na spojenie so spárovaným zariadením, pozri zdrojový kód 3. Na zobrazenie spárovaných zariadení som použil triedu `AlertDialog`.

```
private fun connect(address: String, name: String) {
    doAsync {
        uiThread {
          progressDialog.show()
        }
        try {
            // vytvorí objekt zariadenia na základe adresy
            val device = bluetoothAdapter
            .getRemoteDevice(address)
            // vytvorý objekt BletoothSocket
            val btSocket = device
                .createRfcommSocketToServiceRecord(myUUID)
            bluetoothAdapter.cancelDiscovery()
            btSocket.connect()
            connectedDevices.add(BluetoothDevice(btSocket,
            address, name, this@BluetoothManager))
            uiThread {
              makeToast(fragment.getString(R.string.connected))
            }
        } catch (e: IOException) {
            uiThread {
              makeToast(fragment.getString(R.string.error))
            }

        } finally {
            uiThread {
              progressDialog.dismiss()
            }
        }
    }
}
```

Zdrojový kód 3: Funkcia na pripojenie zariadenia pomocou technológie Bluetooth

5.1.3 Komunikácia s Arduino zariadením

Pred samotnou komunikáciou s Arduino zariadením je potrebné, aby bolo nadviazané spojenie. Implementácia pripojenia S Arduino zariadením pomocou technológie Bluetooth je opísaná v sekcii 5.1.2. Po úspešnom spojení je možné prijímať a odosielať dáta vo formáte `JSON` pomocou komunikačných komponentov, ktoré sú navrhnuté v sekcii 4.1.1.2.

Po úspešnom spojení s Arduino zariadením je možné čítať bajty pomocou triedy `InputStream`, ktorú nám poskytuje vytvorený `BluetoothSocket`. Po získaní dát aplikácia vykoná tieto kroky:

1. Validáciu správy pomocou algoritmu, ktorého úlohou je počítanie pravých a ľavých zložených zátvoriek. Na základe rovnosti ich počtu zistí, či sa jedná o validný `JSON`.

2. Parsovanie správy podľa kľúčov `id` a `text`.

3. Priradenie správy správneho komunikačného komponentu podľa hodnôt `id`, pozri zdrojový kód 4.

4. Zobrazenie hodnoty kľúča `text` pomocou komunikačného komponentu `Text` alebo `Notification`.

Pomocou komunikačného komponentu `Button` je používateľ schopný odosielať správy vo formáte `JSON`. Správa sa z formátu `String` prevedie na pole bajtov a pošleme ju zariadeniu Arduino pomocou triedy `OutputStream`, ktorú nám poskytuje vytvorený `BluetoothSocket`. Tento krok sa vykoná pre všetky pripojené zariadenia.

```
private fun dataReceived(data: String) {
    doAsync {
        val jsonObject = JSONObject(data)
        val text = jsonObject.getString("text")
        val id = jsonObject.optInt("id",-1)

        DataHolder.getInstance().items.forEach { array ->
            array.forEach { item ->
                if (item.id == id) {
                    if (item is SimpleNotification) {
                        uiThread {
                            item.createNotification(name, text)
                        }
                    } else if (item is SimpleText) {
                        uiThread {
                            item.text.value = text
                        }
                    }
                }
            }
        }
    }
}
```

Zdrojový kód 4: Funkcia na parsovanie a priradenie textovej správy správnemu komunikačnému komponentu

5.2 Program pre zariadenie Arduino

Táto sekcia sa zaoberá implementáciou programu pre zariadenie Arduino, ktorého funkciou je prijímať a odosielať správy vo formáte `JSON`. Pri implementácií som kládol dôraz na čitateľnosť zdrojového kódu a možnosť ho jednoducho rozšíriť. Návrh tohto programu je opísaný v sekcií 4.2.

5.2.1 Použité knižnice

Oficiálne vývojové prostredie pre vývoj programov na Arduino obsahuje veľa knižníc, ktoré možno využiť. Analýza tohto vývojového prostredia je v sekcií 3.2.1.

Pri implementácií som nepoužil žiadne špeciálne knižnice ponúkané vývojovým prostredím. Použil som ale knižnicu tretej strany `ArduinoJson` [50]. Pomocou tejto knižnice som bol schopný jednoducho pracovať so správami vo formáte `JSON`.

5.2.2 Komunikácia s mobilnou aplikáciou Android

Komunikácia s mobilnou aplikáciou Android pomocou technológie Bluetooth je implementovaná podľa návrhu v sekcií 4.2.1.

5.2.2.1 Serial

Prijímanie a odosielanie dát medzi zariadeniami cez Bluetooth prebieha pomocou sériovej linky. Na riadenie jej funkčnosti sa využíva objekt `Serial`, ktorý sa môže využívať aj na testovanie pri vývoji, sledovaním výpisov v sériovom monitore. Pre tento objekt je pred použitím nutné nastaviť správnu rýchlosť prenosu dát, ktorú podporuje použité Arduino zariadenie. Táto hodnota je vo väčšine prípadov 9600.

5.2.2.2 Prijímanie a odosielanie správ

Program som implementoval tak, aby vývojár mohol jednoducho rozšíriť tento program, hlavne čo sa týka samotnej komunikácie.

Implementácia prijatia správy je zložitejšia, pretože je potrebné správne zvalidovať prijatú správu. Po úspešnej validácií sa z prijatej správy vytvorí pomocou knižnice `ArduinoJson` objekt `JSON`. Tento objekt potom môže vývojár sparsovať a na základe toho vykonať rôzne funkcie.

Pre odoslanie správy je vytvorená funkcia, ktorú môže vývojár použiť, pozri zdrojový kód 5. V Android aplikácií potom stačí vytvoriť komunikačný komponent `Text` alebo `Notification` so zhodným `id`.

```
void sendMessage(int id, String text) {
  const int capacity = JSON_OBJECT_SIZE(10);
  StaticJsonDocument<capacity> docToSend;

  docToSend["id"] = id;
  docToSend["text"] = text;

  String json;
  serializeJson(docToSend, json);
  Serial.println(json);
}
```

Zdrojový kód 5: Funkcia na odoslanie správy z Arduino zariadenia

Kapitola 6

Testovanie

Súčasťou vývoja aplikácií a programov je ich testovanie. Testované boli samotné funkcie zdrojového kódu. Komunikácia bola nakoniec dôkladne otestovaná na zariadeniach, ktoré simulovali reálne využitie aplikácie a programu.

6.1 Unit testy

Pri testovaní funkcií, ktoré slúžia na zložitejší výpočet, som využil knižnicu `JUnit`. Pomocou tejto testovacej knižnice som otestoval funkcie, ktoré mali na starosti validáciu prijatej správy. Na využitie tejto metódy testovania je potrebné vytvoriť testovaciu triedu v adresári určeného pre testovacie súbory. V tejto triede môže vývojár vytvárať testovacie funkcie, ktoré musia obsahovať anotáciu `@Test`. Túto metódu testovania som použil pri porovnávaní počtu ľavých a pravých zložených zátvoriek správy, pozri zdrojový kód 6.

```
@Test
fun testMatch() {
    val test1 = "{\"id\":\"5\""
    val test2 = "{\"id\":\"5\"}"
    val test3 = "{\"id\":{}"
    val test4 = "{\"id\":{}}"

    assert(!JsonHelper.sameNumberOfBrackets(test1))
    assert(JsonHelper.sameNumberOfBrackets(test2))
    assert(!JsonHelper.sameNumberOfBrackets(test3))
    assert(JsonHelper.sameNumberOfBrackets(test4))
}
```

Zdrojový kód 6: Testovacia funkcia na kontrolu správnosti

6.2 Logger

Pri vývoji mobilnej aplikácie Android je možné vďaka oficiálnemu vývojovému prostrediu `Android studio` využiť `logger`. Tento zabudovaný systém zobrazuje systémove správy práve bežiacej aplikácie na pripojenom zariadení alebo emulátore. Spolu s knižnicou `Timber` opísanej v sekcii 5.1.1.2, som bol schopný jednoducho a efektívne testovať aplikáciu počas implementácie.

6.3 Testovacie zariadenie

Na testovanie som využil Arduino zariadenie s Bluetooth modulom `HC-05`, rotačným motorom a rotačným enkóderom. Pri implementácií tohto zariadenia som využil môj rozšíriteľný program na komunikáciu s mojou aplikáciou.

6.3.1 Implementácia testovacieho zariadenia

Komponenty zapojené do Arduino zariadenia sú naprogramované následovne:

Rotačný motor – Zapnutie, vypnutie a smer rotácie motora sú ovládané pomocou Android aplikácie.

Rotačný enkóder – Po otočení v smere hodinových ručičiek sa zvýši rýchlosť motora, v opačnom smere sa rýchlosť zníži.

V mojej Android aplikácií bolo potrebné vytvoriť všetky komunikačné komponenty pre správnu komunikáciu s Arduino zariadením. Jedná sa o tieto komponenty:

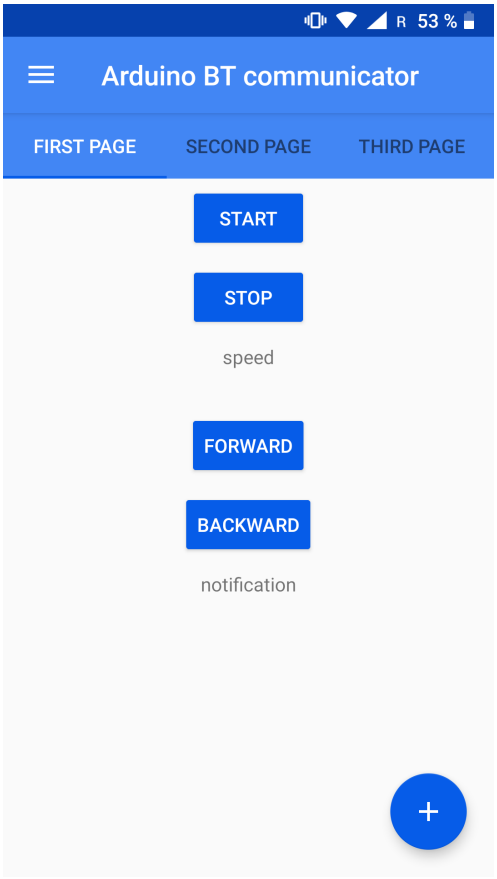
Button – Na ovládanie rotačného motora som použil tieto tlačidlá:

- Start – Zapne motor.
- Stop – Vypne motor.
- Forward a Backward – Nastavia smer otáčania motora.

Text – Tento komponent slúži na zobrazenie rýchlosti motora.

Notification – Po prekročení kritických hodnôt rýchlosti motora sa zobrazí notifikácia.

Tieto komponenty vytvorené v mobilnej aplikácií sú na obrázku 6.1. Pri implementácií programu na Arduino a vytváraní komunikačných komponentov v Android aplikácií je potrebné zadať správne hodnoty kľúča `id`.



Obr. 6.1: Ukážka aplikácie na testovanie

6.3.2 Testovanie zariadenia

Po úspešnej implementácií Arduino zariadenia a nastavenia Android aplikácie bolo možné ich komunikáciu otestovať. Pri testovaní som skúšal rôzne variácie klikaní na `Button` komponenty v mobilnej aplikácií. Takisto som skúšal aj otáčanie rotačného enkódera. Test vyhovel aj rýchlemu klikaniu a otáčaniu. Pri testovaní som zaznamenal menšie spomalenie prenosu dát pri používaní väčšieho množstva znakov v správe. Pri reálnom využití aplikácie je ale zbytočné využívať také množstvo znakov.

Na testovanie som taktiež použil zariadenie `ESP32`, na ktoré je tiež možné skompilovať môj program. S tým rozdielom, že na Bluetooth komunikáciu treba využiť knižnicu `BluetoothSerial.h` a namiesto objektu `Serial` používať objekt `BluetoothSerial`.

Nakoniec som v mobilnej aplikácií pripojil obidve zariadenia a otestoval komunikáciu s väčším množstvom zariadení. Test tejto komunikácie prebehol taktiež úspešne.

Záver

Cieľom tejto práce bolo na základe naštudovania Bluetooth komunikácie na platformách Android a Arduino, navrhnúť a implementovať mobilnú aplikáciu Android, ktorá bude schopná pripojiť sa a komunikovať s väčším množstvom Arduino zariadení. Pred implementáciou som zanalyzoval existujúce riešenia. Po tejto analýze boli navrhnuté funkčné požiadavky tejto aplikácie. Taktiež som zanalyzoval ako správne vytvoriť modernú Android aplikáciu a na základe toho som sa rozhodol implementovať túto aplikáciu v jazyku Kotlin. Táto aplikácia umožňuje komunikovať so zariadeniami pomocou upraviteľných komunikačných komponentov.

Taktiež bol navrhnutý a implementovaný program pre zariadenie Arduino, ktorý je schopný komunikovať s touto mobilnou aplikáciou. Cieľom bolo, aby tento program bol jednoducho rozšíriteľný a upraviteľný. Po implementácií boli tieto zariadenia úspešne otestované.

V budúcnosti bude cieľom pridať do mobilnej aplikácie ďalšie komunikačné komponenty, upraviť dizajn aplikácie a nahrať túto aplikáciu spolu s návodom do Google Play Store.

Literatúra

[1] Bluetooth - How it works. *CCM*, [online] [cit. 2019-04-20]. Dostupné z: https://ccm.net/contents/69-bluetooth-how-it-works

[2] What is Bluetooth Adaptive Frequency Hopping (AFH)? *Honeywell*, [online] [cit. 2019-04-20]. Dostupné z: https://support.honeywellaidc.com/s/article/What-is-Bluetooth-Adaptive-Frequency-Hopping-AFH?s

[3] Bluetooth Physical Layer. *RF Wireless World*, [online] [cit. 2019-04-25]. Dostupné z: http://www.rfwireless-world.com/Tutorials/Bluetooth-physical-layer.html

[4] Adaptive Frequency Hopping for Reduced Interference between Bluetooth® and Wireless LAN. *Design Reuse*, [online] [cit. 2019-04-20]. Dostupné z: https://www.design-reuse.com/articles/5715/adaptive-frequency-hopping-for-reduced-interference-between-bluetooth-and-wireless-lan.html

[5] Bluetooth Basics. *Sparkfun*, [online] [cit. 2019-04-20]. Dostupné z: https://learn.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works

[6] Piconet vs Scatternet-Difference between Piconet and Scatternet. *RF Wireless World*, [online] [cit. 2019-04-24]. Dostupné z: http://www.rfwireless-world.com/Terminology/difference-between-piconet-and-scatternet-in-bluetooth.html

[7] Bluetooth Technology: What Has Changed Over The Years. *medium*, [online] [cit. 2019-04-20]. Dostupné z: https://medium.com/jaycon-systems/bluetooth-technology-what-has-changed-over-the-years-385da7ec7154

[8] sniff subrating. *Search mobile computing*, [online] [cit. 2019-04-20]. Dostupné z: https://searchmobilecomputing.techtarget.com/definition/sniff-subrating

[9] Bluetooth Vs. Bluetooth Low Energy: What's The Difference? *Link Labs*, [online] [cit. 2019-04-22]. Dostupné z: https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy

[10] A quick history of Bluetooth. *Android authority*, [online] [cit. 2019-04-20]. Dostupné z: https://www.androidauthority.com/history-bluetooth-explained-846345/

[11] Bluetooth 101 – Part VI – Bluetooth Architecture. *Hearing Health Technology Matters*, [online] [cit. 2019-04-24]. Dostupné z: https://hearinghealthmatters.org/waynesworld/2014/bluetooth-101-part-vi/

[12] Bluetooth Baseband. *University of Colorado Boulder*, [online] [cit. 2019-04-25]. Dostupné z: http://ecee.colorado.edu/~ecen4242/marko/Bluetooth/Bluetooth/SPECIFICATION/Baseband.htm

[13] Bluetooth Security. *Stanford University*, [online] [cit. 2019-04-27]. Dostupné z: https://cs.stanford.edu/people/eroberts/courses/soco/projects/2003-04/wireless-computing/sec_bluetooth.shtml

[14] Bluetooth Security. *Electronics notes*, [online] [cit. 2019-04-23]. Dostupné z: https://www.electronics-notes.com/articles/connectivity/bluetooth/security.php

[15] How Bluetooth works. *I Programmer*, [online] [cit. 2019-04-23]. Dostupné z: https://www.i-programmer.info/programming/hardware/2602-how-bluetooth-works.html?start=1

[16] John Padgette, L. C., Karen Scarfone: Guide to Bluetooth Security. © 2012, [online] [cit. 2019-05-07]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-121r1.pdf

[17] Bluetooth pairing machanism (Legacy Pairing and Secure Simple Pairing (SSP)). *Silicon Labs*, [online] [cit. 2019-04-27]. Dostupné z: https://www.silabs.com/community/wireless/bluetooth/knowledge-base.entry.html/2015/08/06/bluetooth_pairingma-Fe61

[18] Pripojenie zariadenia s Androidom cez Bluetooth. *Google, Inc.*, [online] [cit. 2019-04-27]. Dostupné z: https://support.google.com/android/answer/9075925?hl=sk

[19] Bluetooth overview. *Google, Inc.*, [online] [cit. 2019-05-01]. Dostupné z: https://developer.android.com/guide/topics/connectivity/bluetooth

[20] HC-05 and HC-06 zs-040 Bluetooth modules. First Look. *Martyn Currey*, [online] [cit. 2019-04-27]. Dostupné z: http://www.martyncurrey.com/hc-05-and-hc-06-zs-040-bluetooth-modules-first-look/

[21] Raspberry Pi 3 Model A+ 64-bit 512MB RAM. *Arduino Shop*, [online] [cit. 2019-04-27]. Dostupné z: https://arduino-shop.cz/arduino/5400-raspberry-pi-3-model-a-64-bit-512mb-ram.html

[22] ESP32 vs ESP8266 – Pros and Cons. *Maker Advisor*, [online] [cit. 2019-04-27]. Dostupné z: https://makeradvisor.com/esp32-vs-esp8266/

[23] Arduino Bluetooth Basic Tutorial. *Arduino*, [online] [cit. 2019-04-27]. Dostupné z: https://create.arduino.cc/projecthub/mayooghgirish/arduino-bluetooth-basic-tutorial-d8b737

[24] Giumig Apps: Arduino bluetooth controller. 2016, [online] [cit. 2019-04-20]. Dostupné z: https://play.google.com/store/apps/details?id=com.giumig.apps.bluetoothserialmonitor

[25] CircuitMagic: Arduino bluetooth. 2015, [online] [cit. 2019-04-20]. Dostupné z: https://play.google.com/store/apps/details?id=com.circuitmagic.arduinobluetooth

[26] Hauke, F.: Arduino Bluetooth Terminal. 2016, [online] [cit. 2019-04-20]. Dostupné z: https://play.google.com/store/apps/details?id=com.frederikhauke.ArduTooth

[27] Poddar, R.: Arduino Bluetooth Serial monitor. 2018, [online] [cit. 2019-04-20]. Dostupné z: https://play.google.com/store/apps/details?id=appinventor.ai_poddarrupak2808.Rupak_arduino_serial_monitor

[28] OmarAflak: Android Bluetooth Library. 2016, [online] [cit. 2019-04-20]. Dostupné z: https://github.com/OmarAflak/Bluetooth-Library

[29] sirvar: BluetoothKit. 2018, [online] [cit. 2019-04-20]. Dostupné z: https://github.com/sirvar/bluetoothkit-android

[30] tlgbltcn: BluetoothHelper. 2019, [online] [cit. 2019-04-20]. Dostupné z: https://github.com/tlgbltcn/BluetoothHelper

[31] arissa34: Android-Multi-Bluetooth-Library. 2015, [online] [cit. 2019-04-20]. Dostupné z: https://github.com/arissa34/Android-Multi-Bluetooth-Library

[32] harry1453: Android-bluetooth-serial. 2018, [online] [cit. 2019-04-20]. Dostupné z: https://github.com/harry1453/android-bluetooth-serial

[33] Android Studio. *Google, Inc.*, [online] [cit. 2019-05-03]. Dostupné z: https://developer.android.com/studio

[34] Android Architecture with MVP or MVVM - Tutorial. *Vogella*, [online] [cit. 2019-05-05]. Dostupné z: https://www.vogella.com/tutorials/AndroidArchitecture/article.html

[35] The MVC, MVP, and MVVM Smackdown. *Realm*, [online] [cit. 2019-05-05]. Dostupné z: https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android

[36] MVP and MVC Architectures in Android – part 1. *TechYourChance*, [online] [cit. 2019-05-05]. Dostupné z: https://www.techyourchance.com/mvp-mvc-android-1

[37] MVP vs MVVM: A Review of Patterns for Android. *Think Mobiles*, [online] [cit. 2019-05-05]. Dostupné z: https://thinkmobiles.com/blog/mvp-vs-mvvm-android-patterns/

[38] Getting Started with Arduino and Genuino products. *Arduino*, [online] [cit. 2019-05-08]. Dostupné z: https://www.arduino.cc/en/Guide/HomePage

[39] Kotlin is now Google's preferred language for Android app development. *Tech Crunch*, [online] [cit. 2019-05-11]. Dostupné z: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/?guccounter=1

[40] Android overview. *Kotlinlang*, [online] [cit. 2019-04-20]. Dostupné z: https://kotlinlang.org/docs/reference/android-overview.html

[41] Kotlin on Android. Now official. *Jetbrains*, [online] [cit. 2019-04-20]. Dostupné z: https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/

[42] Distribution dashboard. *Google, Inc.*, [online] [cit. 2019-05-11]. Dostupné z: https://developer.android.com/about/dashboards

[43] Android Jetpack. *Google, Inc.*, [online] [cit. 2019-05-11]. Dostupné z: https://developer.android.com/jetpack

[44] LiveData Overview. *Google, Inc.*, [online] [cit. 2019-05-11]. Dostupné z: https://developer.android.com/topic/libraries/architecture/livedata

[45] Data Binding Library. *Google, Inc.*, [online] [cit. 2019-05-11]. Dostupné z: `https://developer.android.com/topic/libraries/data-binding`

[46] ViewModel Overview. *Google, Inc.*, [online] [cit. 2019-05-11]. Dostupné z: `https://developer.android.com/topic/libraries/architecture/viewmodel`

[47] Wharton, J.: Timber. 2013, [online] [cit. 2019-05-12]. Dostupné z: `https://github.com/JakeWharton/timber`

[48] grandcentrix GmbH: Tray. 2015, [online] [cit. 2019-05-12]. Dostupné z: `https://github.com/grandcentrix/tray`

[49] Kotlin: Anko. 2015, [online] [cit. 2019-05-12]. Dostupné z: `https://github.com/Kotlin/anko`

[50] Blanchon, B.: Android-bluetooth-serial. 2014, [online] [cit. 2019-04-20]. Dostupné z: `https://github.com/bblanchon/ArduinoJson`

Dodatok **A**

Zoznam použitých skratiek

UI User Interface

SSP Secure Simple Pairing

PIN Personal Identification Number

BLE Bluetooth Low Energy

IoT Internet of Things

IDE Integrated Development Environment

XML eXtensible Markup Language

JSON JavaScript Object Notation

Dodatok B

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `apk` ..... adresár so spustiteľnou formou implementácie
- `src` ..... zdrojové kódy
  - `impl` ..... zdrojové kódy implementácie
    - `android` ..... zdrojové kódy Android aplikácie
    - `arduino` ..... zdrojové kódy programu na Arduino
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF