ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Knihovna funkcí pro OS Android umožňující řízení vývojového kitu Arduino
Student:	Filip Šmíd
Vedoucí:	Ing. Pavel Kubalík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Prostudujte existující řešení.
Navrhněte a implementujte knihovnu pro platformu Android zajišťující komunikaci mezi zařízením se systémem Android a vývojovým kitem Arduino.
Knihovna bude umožňovat komunikaci pomocí USB, Bluetooth a Wi-Fi na protokolu TCP/IP.
Pro vytvořenou knihovnu napište testovací aplikaci, která bude prezentovat funkčnost komunikace.
Aplikace bude napsána tak, aby bylo možné ji snadno rozšířit o další funkční prvky.
Aplikace a knihovna by měla podporovat specifickou architekturu pro vývoj Android aplikací, jako MVP, nebo MVVM, pro oddělení logiky z prezentační vrstvy a testovatelnost kódu.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 4. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Knihovna funkcí pro OS Android umožňující řízení vývojového kitu Arduino

Filip Šmíd

Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

15. května 2017

Poděkování

Nejprve bych rád poděkoval vedoucímu práce Ing. Pavlovi Kubalíkovi, Ph.D za užitečné rady a za čas, který mi věnoval, během konzultačních setkání. Dále bych rád poděkoval Vaškovi za poskytnutí potřebného hardwaru pro vytvoření bakalářské práce a na závěr bych rád poděkoval Lucce, za projevenou ochotu a čas strávený při korektuře práce.

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 15. května 2017 .....

Odkaz na tuto práci

Šmíd, Filip. *Knihovna funkcí pro OS Android umožňující řízení vývojového kitu Arduino*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Bakalářská práce se zabývá možnostmi řízení vývojového kitu Arduino pomocí zařízení s operačním systémem Android. Dále popisuje technologie a nástroje dostupné pro vývoj řešení popsaného problému. Cílem práce je analýza možností řešení problému, návrh a implementace knihovny funkcí pro operační systém Android umožňující řízení (komunikaci pomocí příkazů) Arduina a návrh a implementace Android aplikace demonstrující funkčnosti implementované knihovny. Hlavní důraz byl kladen na rozšiřitelnost knihovny a aplikace. V návrhu a samotné implementaci byly využity a zohledněny informace získané během analýzy problému. Výsledkem práce je prototyp aplikace snadno rozšiřitelný a modifikovatelný pro konkrétní použití a knihovna funkcí zajišťující funkčnost komunikace.

Klíčová slova Android, Arduino, řízení, komunikace, USB, Bluetooth, WiFi

Abstract

The bachelor thesis deals with ways of controlling Arduino development kit using a device running an Android operating system. The available development tools and technologies are alse described. The goal of the thesis is analysing different solutions to the problem, designing and implementing an Android library to provide communication with Arduino devices and an Android application to demonstrate the functionality of the library. The main emphasis was put on the library's and the application's extensibility. The application design and implementation is based on the prior analysis of the problem. Extensible and modifiable application prototype and the Android library providing communication is the result of this thesis.

Keywords Android, Arduino, control, communication, USB, Bluetooth, WiFi

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Úvod

„Chytrá“ mobilní zařízení jsou všude kolem nás. Dnes už ani nepotkáte člověka, který by nepoužíval mobilní telefon, tablet, či „chytré“ hodinky. „Chytrá“ mobilní zařízení se zkrátka stala běžnou součástí našeho života.

Můžeme však jít ještě mnohem dále. Představte si, že pomocí svého mobilního telefonu otevřete domovní dveře, nastartujete auto, necháte si uvařit kávu, či nastavit termostat na chatě, na kterou právě vyrážíte. Nejedná se ovšem o žádnou utopii, tzv. internet věcí (IoT) je moderní a masově se rozrůstající oblast ve světě informačních technologií.

Proč ale platit nemalé peníze za tato chytrá řešení, když většinu běžných úkonů lze vyřešit levněji s trochou zručnosti a vývojovým kitem Arduino. Pomocí vašeho telefonu a Arduina se senzorem teploty, tlaku a slunečního svitu si snadno můžeme postavit domácí meteostanici. Nyní už stojíme pouze před otázkou, jak naměřená data přenést do našeho telefonu. Lze využít například řešení, které představuje tato bakalářská práce.

Náplní práce je implementovat knihovnu pro platformu Android, poskytující programátorovi rozhraní pro komunikaci s Arduinem. Knihovna bude umožňovat komunikaci pomocí USB, Bluetooth a WiFi. Nad knihovnou bude vytvořena aplikace reprezentující práci s knihovnou a zvolenými možnostmi komunikace.

Práce by měla primárně sloužit pro studenty hardwarových oborů, kteří by pomocí aplikace, případně vlastní aplikace, postavené nad knihovnou, mohli řešit jen odesílání a přijímaní dat mezi zařízeními, nemuseli už řešit samotnou komunikaci a navazování spojení a mohli se tak zaměřit na problémy spojené přímo s hardwarem zařízení.

Cílem rešeršní části práce je prostudovat již existující řešení umožňující řízení vývojového kitu Arduino, zhodnotit jejich výhody a nevýhody a určit použitelnost při vývoji požadované knihovny, resp. aplikace. Cílem analýzy problému je najít a zhodnotit možnosti požadovaných technologií pro komunikaci, jak z hlediska Androidu, tak Arduina a představit možnosti vývoje pro Android a Arduino. Na závěr analýzy určit jednotlivé požadavky systému.

Cílem návrhu je, na základě získaných informací z provedené analýzy, navrhnout knihovnu na řízení (komunikaci) mezi zařízením se systémem Android a vývojovým kitem Arduino a současně navrhnout aplikaci demonstrující funkčnost knihovny. Cílem implementační části je na základě návrhu implementovat navrženou knihovnu a ukázkovou aplikaci. Posledním cílem je pak výsledný produkt otestovat a představit výsledky testů.

Kapitola 1

Existující řešení

Komunita kolem platformy Android, jak uživatelská, tak vývojářská, je velice rozsáhlá. To samé bychom mohli říci o komunitě kolem vývojového kitu Arduino a podobných vývojových kitů. Proto lze nalézt velké množství aplikací a knihoven, které se zabývají řešením komunikace mezi těmito druhy zařízení.

1.1 Aplikace

V této podkapitole představím existující aplikace, které jsou volně dostupné ke stažení přes Google Play[1].

1.1.1 Arduino bluetooth controller

Aplikace umožňuje komunikaci s Arduinem pomocí Bluetooth. Aplikace zobrazí všechna nalezená zařízení v dosahu, uživatel si zvolí zařízení a připojí se k němu[14]. Poté může zasílat jednoduché příkazy pomocí textového pole a konfigurovatelných tlačítek.

1.1.2 Arduino Total Control free

Aplikace umožňuje komunikaci pomocí Bluetooth, WiFi a Ethernet na protokolu TCP/IP a to oběma směry (odesílání i přijímaní dat)[10]. Aplikace má mnoho možností pro nastavení ovládacích prvků. Pro komunikaci aplikace definuje svůj specifický protokol.

1.1.3 Arduino Smart Home Automation

Komplexnější řešení, které je zaměřeno na IoT, umožňuje připojení a kontrolu několika zařízení v reálném čase. Aplikace komunikuje pomocí WiFi nebo internetu na protokolu TCP/IP[2].

[1] Obchod s android aplikacemi

1.1.4 Arduino Uno Communicator

Aplikace, zastupující řešení komunikace přes USB port telefonu, umožňuje odesílat a přijímat data, ale neumožňuje nastavení modulační rychlosti, podporuje pouze 9600 Bd[28].

1.2 Knihovny

V této části se zabývám volně dostupnými knihovnami, které umožňují některé z požadovaných druhů komunikace. Výhoda těchto řešení spočívá v tom, že je lze použít v aplikaci, tudíž není potřeba nabízenou funkčnost implementovat znovu.

1.2.1 UsbSerial

Knihovna poskytující rozhraní pro komunikaci se zařízením připojeným přes USB port. Jedná se o wrapper (obálku) nativního USB API Androidu. Knihovna umožňuje konfiguraci spojení, synchronní a asynchronní komunikaci pro všechny úkony potřebné pro komunikaci, jako otevření a zavření spojení, přijmutí a odeslání dat. Knihovna podporuje různé druhy připojených zařízení.

1.2.2 usb-serial-for-android

Knihovna poskytující rozhraní pro komunikaci se zařízením připojeným přes USB port. Jedná se o wrapper nativního USB API Androidu. Knihovna umožňuje konfiguraci spojení. Na rozdíl od předchozí knihovny lze obsluhovat spojení pouze synchronně a podporuje méně druhů připojených zařízení.

1.3 Arduino ADK

Arduino Accessory Development Kit je sada nástrojů, která umožňuje vývojářům hardwaru vytvářet příslušenství pro Android, jako například dokovací stanice, meteorologické stanice atd. Zařízení, implementující Android Open Accessory protokol (zkráceně AOA), může pak komunikovat se systémem Android prostřednictvím USB nebo Bluetooth technologie[17].

Arduino nabízí hned několik desek a rozšiřujících modulů implementující AOA protokol. Jedná se o desky Mega ADK a Micro ADK a dále pak o modul USB Host shield[3].

1.4 Shrnutí

U představených aplikací se většinou jednalo o řešení připravené pro konkrétní konfiguraci Arduina. Za alespoň částečně univerzální řešení lze pova-

žovat aplikaci Arduino bluetooth controller. Implementovanou funkčnost jednotlivých aplikací jsem shrnul v tab. 1.1.

	USB	Bluetooth	WiFi	Konfigurovatelnost	Dostupnost zdrojových kódů
Arduino bluetooth controller	✓			✓	
Arduino Total Control free		✓	✓		
Arduino Smart Home Automation			✓	✓	
Arduino Uno Communicator	✓				✓

Tabulka 1.1: Porovnání aplikací

Kapitola 2

Analýza a návrh

V této kapitole nejdříve představím platformu Android, vývojový kit Arduino a jimi nabízené možnosti pro podporu požadovaných druhů komunikace. Dále obecně shrnu technologie a nástroje, které lze použít při vývoji aplikací pro Android, nebo Arduino. V rámci softwarové analýzy se zaměřím na možnosti volby architektury pro vývoj Android aplikací. Ze získaných informací pak definuji požadavky na systém (aplikaci). Dále se zaměřím na aplikaci jako celek a definuji požadavky ze strany uživatele. Na závěr představím modely nejdůležitějších procesů systému.

2.1 Android

Android OS je operační systém pro mobilní telefony, televize, „chytré“ hodinky a řídící systémy automobilů založený na linuxovém jádře. V současné době se jedná o nejrozšířenější operační systém pro mobilní zařízení, o čemž svědčí i fakt, že každý den je aktivováno více než jeden milión nových zařízení se systémem Android[20]. V porovnání s operačním systémem iOS, druhým nejpoužívanějším operačním systémem, se jedná o více než trojnásobek používaných zařízení, jak lze vidět na obr. 2.1.

Operační systém android existuje v mnoha různých verzích. Výhodou je, že poskytuje zpětnou kompatibilitu, což znamená, že pokud vyvíjíme aplikaci pro nejnovější verzi OS, lze aplikaci, až na některá omezení, nainstalovat i na zařízení se starší verzí OS.

Na začátku vývoje aplikace je potřeba určit, jakou nejstarší verzi OS bude aplikace podporovat. Dle [24] je dobrým zvykem podporovat okolo 90 % aktivních zařízení. To v současné době znamená, podporovat alespoň verzi 4.4 (API level 19).

Verze	Název verze	API	Rozdělení [%]
2.3.3 - 2.3.7	Gingerbread	10	0,9
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0,9
4.1.x	Jelly Bean	16	3,5
4.2.x		17	5,1
4.3		18	1,5
4.4	KitKat	19	20,0
5.0	Lollipop	21	9,0
5.1		22	23,0
6.0	Marshmallow	23	31,2
7.0	Nougat	24	4,5
7.1		25	0,4

Tabulka 2.1: Zastoupení verzí operačního systému Android, upraveno podle [16]

2.1.1 Komunikace

Platforma Android poskytuje mnoho možností pro komunikaci se vzdálenými zařízeními, od mobilní komunikace GSM, přes bezdrátovou komunikaci pomocí Bluetooth, nebo WiFi až po komunikaci pomocí NFC čipu, nebo standardní sériovou komunikací přes USB port zařízení.

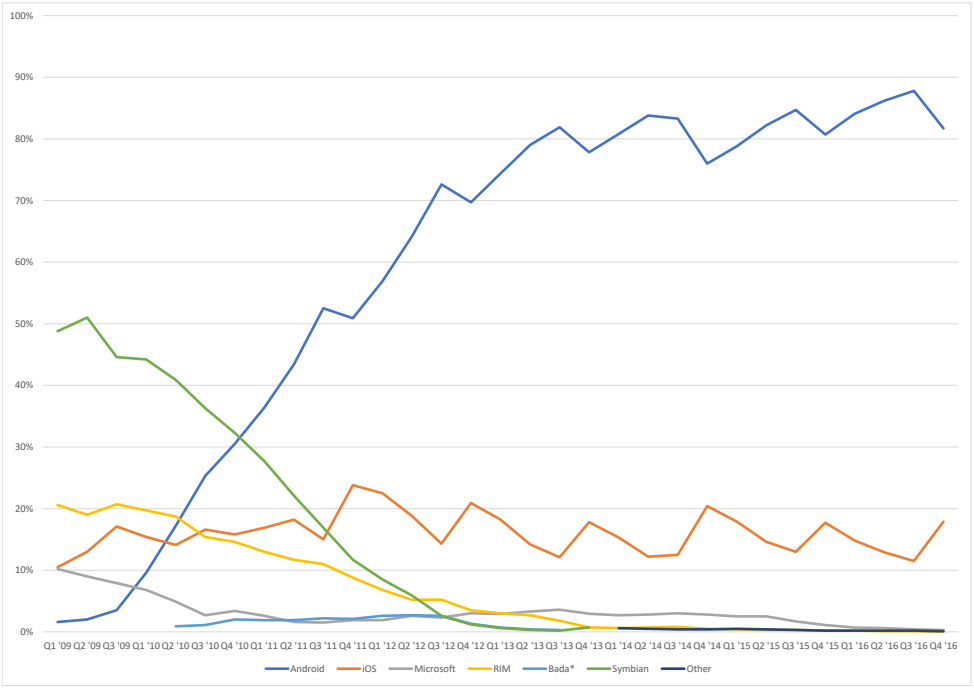
Všechny komunikační technologie požadované, v rámci specifikace problému, jsou systémem Android podporovány. Lze nalézt i zařízení, které některou z technologií nepodporují, neboť nemají k dispozici potřebný hardware. Aby aplikace, dostupná z Google Play, nebyla uživateli s nepodporujícím zařízením nabízena, lze v takovém případě při vývoji definovat, které hardware komponenty musí být součástí uživatelova zařízení.

2.1.1.1 USB

Android podporuje různé druhy USB periferií a Android USB příslušenství (hardware, který implementuje Android accessory protokol) prostřednictvím dvou režimů: USB Accessory a USB host[25]. Jednotlivé režimy se odlišují podle toho, jaké zařízení poskytuje napájení viz. obr. 2.2.

- **USB Accessory**
  V režimu Accessory poskytuje napájení zařízení připojené do portu USB, tzn. připojené zařízení není napájeno přes USB port. Jedná se například o dokovací stanice, čtečky karet nebo diagnostické nástroje.

- **USB Host**
  V režimu Host připojenému zařízení poskytuje napájení zařízení se systémem Android. V tomto režimu jsou například externí klávesnice, myši nebo další herní ovladače.



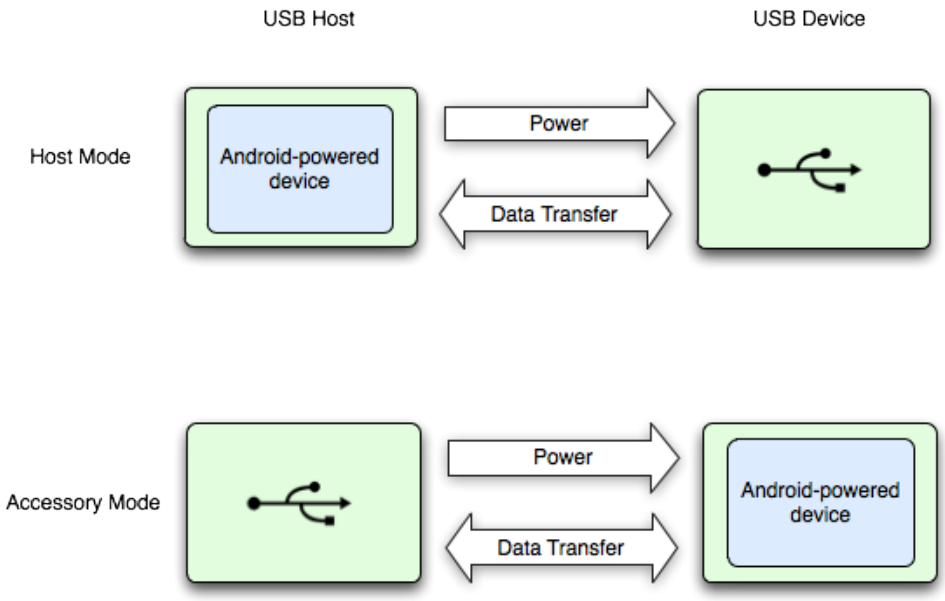
Obrázek 2.1: Porovnání používaných operačních systémů, upraveno z [13]

2.1.1.2 Bluetooth

Platforma Android umožňuje bezdrátovou komunikaci pomocí připojení Bluetooth. Aplikace využívající Bluetooth API umožňuje point-to-point a multipoint připojení. Bluetooth API poskytuje následující funkčnosti:

- vyhledávání zařízení Bluetooth,
- dotaz na lokální adaptér zařízení pro spárovaná zařízení,
- zavedení RFCOMM kanálů,
- připojení k nalezeným zařízením,
- přenášení dat mezi zařízeními,
- správa násobného připojení[21].

Zařízení, které pro komunikaci používá Bluetooth API se nachází ve dvou režimech. V prvním režimu se zařízení chová jako klient, tzn. vyhledá dostupné zařízení v režimu server, zahájí a inicializuje spojení. Ve druhém režimu se zařízení chová jako server, tzn. čeká na připojení vzdáleného zařízení a po



Obrázek 2.2: Schéma USB režimů, převzato z [26]

navázaní spojení s ním komunikuje. Po ukončení spojení čeká na připojení dalšího zařízení.

Navázat spojení mezi zařízeními lze pouze v případě, že jsou tato zařízení spárovaná (tvoří tzv. bond)[30]. Pokud se rozhodneme navázat spojení mezi zařízeními, nejdříve musíme zařízení spárovat a poté lze navázat spojení.

2.1.1.3 WiFi

Platforma Android může umožňovat, v závislosti na druhu zařízení, dva druhy spojení pomocí WiFi adaptéru.

- **WiFi P2P**
  Peer-to-peer připojení bez mezilehlého přístupového bodu je mnohem rychlejší a umožňuje přenos dat na větší vzdálenosti než Bluetooth. Zařízení, která spolu chtějí komunikovat přes WiFi P2P, musejí službu podporovat tzn. WiFi čipy obou zařízení musejí podporovat službu Wi-Fi Direct[22].

- **TCP/IP komunikace**
  Druhou možností WiFi spojení je komunikace pomocí protokolu TCP/IP, ať už prostřednictvím připojení do sítě internet nebo lokální sítě. Zde je situace podobná jako u technologie Bluetooth, neboť opět může být zařízení ve dvou režimech, a to jako klient, nebo server.

2.1.2 Shrnutí

U představených komunikačních technologií může zařízení vystupovat ve více režimech. Je tedy potřeba určit, který z režimů bude knihovna, resp. aplikace podporovat. Vzhledem k možnostem využití Arduina, které lze použít například pro sběr dat, nebo ovládání konkrétních prvků jako je: zámek dveří, zvonek, nebo domovní telefon, tudíž je někde trvale umístěno, se zde nabízí možnost mít Arduino v roli serveru. Android zařízení v roli klienta se pak k serveru připojuje například z důvodu předání, získání dat, či změnu nastavení Arduina. U technologie USB, jak již bylo řečeno, se režimy liší pouze v tom, které zařízení poskytuje napájení. Tento rozdíl nemá vliv na implementaci, knihovna může tedy podporovat oba režimy USB. U WiFi jsem se rozhodl pro TCP/IP komunikaci, lze tím podporovat více druhů Android i Arduino zařízení.

2.2 Arduino

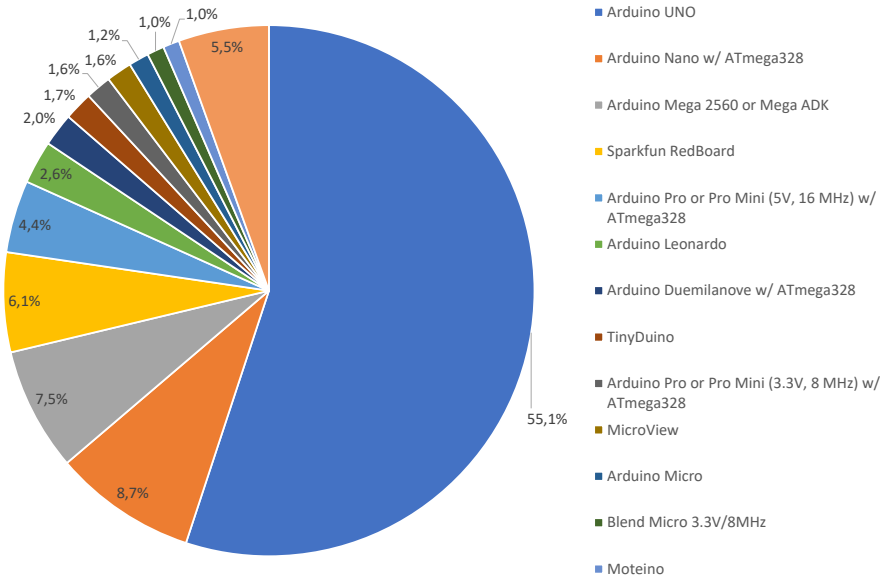
„*Arduino je otevřená elektronická platforma, založená na jednoduché počítačové desce (hardware) a vývojovém prostředí, které slouží k tvorbě software*“[4]. Pomocí Arduina můžeme sbírat a zpracovávat nepřeberné množství informací. Můžeme si vytvořit, za minimální počáteční náklady, domácí teploměr, autíčko na dálkové ovládání, nebo domovní telefon. Máme neomezené možnosti tvorby. Můžeme pořídit desku už se spoustou zabudovaných senzorů (např. Arduino Esplora), nebo desku (např. Arduino Uno, nebo Arduino Mega), ke které lze jednoduše připojit celou škálu rozšiřujících desek. „*Tyto desky označujeme jako Arduino Shieldy. Dnes je jich veliké množství a ke každému je zpravidla volně k dispozici knihovna pro okamžité použití*“[4]. Oblíbenost jednotlivých desek můžeme vidět na obr. 2.3.

2.2.1 Komunikace

Arduino umožňuje mnoho druhů komunikace. Existují desky, které mají daný čip pro komunikaci již integrován, nebo lze použít některý z dostupných Arduino Shieldů a pomocí mnoha existujících návodů a schémat dostupných na internetu připojit shield k desce. Standardně Arduino deska poskytuje několik digitálních a analogových portů a několik párů sériových portů (označovány jako RX a TX port).

2.2.1.1 USB

Součástí většiny Arduino desek je již integrovaný USB převodník, ten slouží k nahrávání programů, ale lze mít i desku bez USB převodníku. V tom případě je zde možnost využít shield s USB převodníkem (například Arduino USB TTL převodník).



Obrázek 2.3: Používání Arduino desek, upraveno podle [11]

2.2.1.2 Bluetooth

„*Arduino může prostřednictvím protokolu Bleutooth komunikovat přes externí modul. Existuje i verze shieldu s označením Arduino_BT, která obsahuje integrované Bluetooth rozhraní*“[32]. Pokud máme desku bez integrovaného Bluetooth rozhraní, můžeme použít například modul Arduino bluetooth modul HC-05, se kterým lze po připojení jednoduše komunikovat pomocí sériové linky.

2.2.1.3 WiFi

U WiFi komunikace je situace podobná jako při komunikování pomocí Bluetooth. Existují desky s integrovaným WiFi rozhraním (např. Arduino Uno WiFi), nebo můžeme použít některý z dostupných externích modulů jako například modul ESP8266, se kterým lze opět komunikovat pomocí sériové linky.

2.2.1.4 Shrnutí

Arduino nabízí mnoho různorodých technologií pro komunikaci se vzdálenými zařízeními. Pokud chceme použít některou z dostupných technologií,

lze vybrat desku, která bude mít integrované rozhraní pro námi zvolený druh komunikace, nebo použít některý z dostupných shieldů.

2.3 Technologie

V rámci analýzy požadovaného systému, je potřeba definovat a určit, co od daného systému vlastně očekáváme, s tím souvisí i volba technologie a programovacího jazyka. V následující části představím a porovnám technologie, které můžeme použít pro vývoj pro Android a Arduino.

2.3.1 Android

Pro vývoj pro platformu Android můžeme použít mnoho rozdílných technologií. Pokud máme za úkol vytvořit aplikaci dostupnou pro více mobilních platforem, můžeme použít některou z technologií pro multiplatformní vývoj.

Pro vývoj menší aplikace, která nepotřebuje využívat nativních komponent Android platformy, jako používání kamery, přístup ke kontaktům atd., lze použít některou z dostupných webových technologií například HTML5. Dle [31] je pak vývoj takové aplikace rychlejší a levnější.

Vývoj multiplatformní a současně nativní aplikace umožňuje technologie Xamarin. Xamarin je moderní nástroj pro vývoj nativních multiplatformních aplikací. „*Xamarin umožnuje vývojářům, jak vytvořit aplikace pro všechny tři nejrozšířenější platformy (iOS, Android i Windows), tak využít znalostí prostředí C# např. pro psaní různých druhů aplikací ve Visual Studiu nebo využití stávajících knihoven dostupných třeba na Nugetu*“[29].

Hlavní a nejpoužívanější technologií pro vývoj nativních aplikací je jazyk Java s využitím Android SDK. Programovací jazyk Java je oficiální jazyk pro vývoj pro Android. Pokud chceme při vývoji aplikace některé operace zefektivnit, nebo znemožnit dekompilací zneužití některé části kódu (například šifrovací algoritmus), můžeme využít nástrojů Android NDK a tyto části aplikace napsat v jazyce C/C++. V tomto případě jsme pak nuceni tyto části předkompilovat pro jednotlivé architektury procesorů používaných v zařízeních se systémem Android.

Jako poslední představím technologii Kotlin. Kotlin je objektový programovací jazyk vyvinutý firmou JetBrains. Je plně kompatibilní s Javou, tzn. lze z kódu napsaném v Kotlinu volat komponenty napsané v Javě a současně z kódu napsaného v Javě volat komponenty napsané v Kotlinu. Kotlin přejímá typické konstrukce z ostatních programovacích jazyků a zavádí také vlastní konstrukce pro zjednodušení vývoje. Kód v Kotlinu je zpravidla mnohem více úspornější než kód napsaný v Javě, jak je vidět na příkladu 2.1.

```java
// java
// Sort by row and columns
List<SeatDO> sortedSeats
    = seats.stream()
           .sorted(comparing(SeatDO::getRow))
           .thenComparing(SeatDO::getColumn)
           .collect(Collectors.toList());

// kotlin
// Sort by row and columns
seats.sortWith(compareBy({it.row}, {it.column}))
```

Zdrojový kód 2.1: Řazení v Javě vs. v Kotlinu, upraveno z [35]

2.3.2 Arduino

Programovací jazyk Arduino, který je založen na jazyce Wiring, slouží pro vytváření programů pro Arduino mikrokontrolér. Wiring je open–source framework, který podporuje programovat velké množství desek s mikrokontroléry.

2.4 Vývojové prostředí a nástroje

V následující části představím nástroje, které nám pomáhají a usnadňují vývoj výsledných produktů. Samozřejmě lze i nadále používat obyčejný textový editor, ale pokud chceme zefektivnit a zjednodušit vývoj aplikace, bez pořádných editorů se neobejdeme. Dokonce při psaní programu pro některou z Arduino desek se při nahrávání vytvořeného programu nenahrává pouze náš program, ale spoustu další řídících příkazů typických pro konkrétní desku.

2.4.1 Android

Oficiálním IDE pro Android je Android Studio, vývojové prostředí od společnosti Google založené na IntelliJ IDEA od společnosti JetBrains, vývojovém prostředí pro Javu. Android Studio nahradilo dříve používané Eclipse IDE, které sice umožňovalo vytvářet Android aplikace, ale už jen samotná instalace rozšíření pro Android byla velice obtížná. Android Studio ve spojení s Gradle umožňuje rychlé sestavování aplikace a nabízí spoustu nástrojů usnadňující vývoj, ladění, testování a publikaci aplikace.

2.4.1.1 Gradle

Nástroj Gradle slouží pro správu závislostí, automatizaci sestavení, testování, publikaci aplikace a podporu multi-jazykových[2] aplikací[27]. Jedná se o nástupce build automation tools (nástrojů pro automatické sestavení aplikace) jako jsou například Ant a Maven.

2.4.1.2 Android Debug Bridge (ADB)

ADB slouží pro správu a komunikaci s připojeným zařízením (fyzickým i emulovaným). Umožňuje instalaci a odinstalaci aplikace, ladění nebo správu zařízení. Funguje na podobném přincipu jako klient–server program[18].

- **Klient**
  Běží na počítači vývojáře, komunikuje se serverem a odesílá na něj příkazy.
- **Server**
  Běží na počítači vývojáře, řídí komunikaci mezi klientem a démonem. Komunikace může probíhat prostřednictvím USB, nebo TCP/IP. Současně může obsluhovat více klientů.
- **Démon**
  Proces běžící na pozadí každého Android zařízení, vykonává obdržené příkazy.

Schéma komunikace mezi jednotlivými částmi lze vidět na C.1.

2.4.1.3 Android monitor

Android monitor slouží ke sledování a optimalizaci aplikace. Umožňuje monitorování využití paměti, CPU, GPU a síťové komunikace. Důležitou funkčností je také zobrazování systémových a aplikačních logů. Dále umožňuje analyzovat data, se kterými aplikace pracuje, nebo zachytit obraz běžící aplikace[19].

2.4.1.4 Android emulátor

Android emulátor umožňuje spouštění virtuálních zařízení na počítači vývojáře a emulování specifických podmínek např. omezené a nestabilní internetové připojení, omezení velikosti RAM paměti atd., na zařízení. Emulátor neumožňuje simulaci:

- WiFi,
- Bluetooth,

[2] Z hlediska programovacích jazyků

- NFC,
- vyjmutí/vložení SD karty,
- připojení sluchátek,
- USB[23].

2.4.1.5 Android Virtual Device Manager (AVD Manager)

AVD vytváří a spravuje virtuální zařízení spustitelná v Android emulátoru.

2.4.1.6 SDK manager

SDK manager slouží pro správu nativních knihoven, obrazů zařízení a vývojářských nástrojů.

2.4.2 Arduino

Pro tvorbu programů pro Arduino existuje celá řada vývojových prostředí, která se často liší jen uživatelským rozhraním. Důležitou a nepostradatelnou částí takového prostředí je pak AVR kompilátor a programátor, který slouží pro kompilaci a nahrání kódu do mikroprocesoru.

Do kategorie vývojových prostředí pro Arduino patří například Visual Studio. Jedná se o vývojové prostředí od společnosti Microsoft nabízející podporu pro vývoj například v C/C++, C#, již zmíněném Xamarinu, nebo webových technologií jako Node.js, nebo JavaScript a po instalaci rozšíření (pluginu) i pro Arduino. Výhodou Visual Studia je používání jednoho nástroje pro mnoho různých technologií.

Mezi volně dostupná a multiplatformní vývojová prostředí patří například Atmel Studio, nebo PlatformIO IDE. Zajímavou technologií může být CodeBender, který umožňuje editovat a nahrávat kód do Arduina pouze prostřednictvím webového prohlížeče (zde stačí doinstalovat potřebné rozšíření do prohlížeče).

Asi nejznámější a současně nejpoužívanější vývojové prostředí pro Arduino je Arduino IDE. Jedná se o open–source a multiplatformní vývojové prostředí napsané v jazyce Java. Výhodou používaní Arduino IDE je integrace velkého množství nástrojů pro vývoj, nahrávání a ladění přímo do prostředí (například Serial Monitor, Manager desek, nebo Programmer). Další výhodou je velká databáze ukázkových řešení pro různé druhy desek a periférií[33].

2.4.2.1 Sériový Monitor

Nástroj pro sériovou komunikaci s Arduino deskou, umožňuje přijímat a odesílat zprávy na zvoleném sériovém portu, definovat baud rate[3] a komunikovat s moduly desky.

2.4.2.2 Programmer

Umožňuje výběr programátora hardwaru pro programování desky, nebo čipu bez použití integrovaného USB na desce[5].

2.4.2.3 Emulátory a simulátory

Arduino emulátory a simulátory umožňují virtualizaci části funkčnosti desky, ale nedokáží simulovat reálnou desku. Lze je použít k testování určité funkčnosti kódu, či prezentaci jednoduché aplikace. Jedná se například o Simuino, Autodesk 123D circuits, nebo VBB4Arduino[34].

2.5 Architektura

Při vývoji softwarového, ale i hardwarového řešení je důležité si předem určit architekturu a další architektonické styly. Vhodně zvolená architektura urychluje vývoj a zpřehledňuje výsledný produkt a při dalším rozvoji ušetří nejen čas, ale i peníze. Každý vyvíjený systém by měl být založen na nějaké architektuře, nebo alespoň na přesně definovaných pravidlech například: jak bude daný systém rozdělen do balíků, tříd, funkcí, jaké budou použité jmenné konvence, atd. Systém, který tato pravidla postrádá je dlouhodobě neudržovatelný a jeho budoucí rozvoj je mnohem náročnější. Při vývoji většího systému se může zavedení architektury zdát jako velice nákladné, ale toto zdání není pravdivé, protože v celkovém výsledku se náklady na vývoj nezvýší.

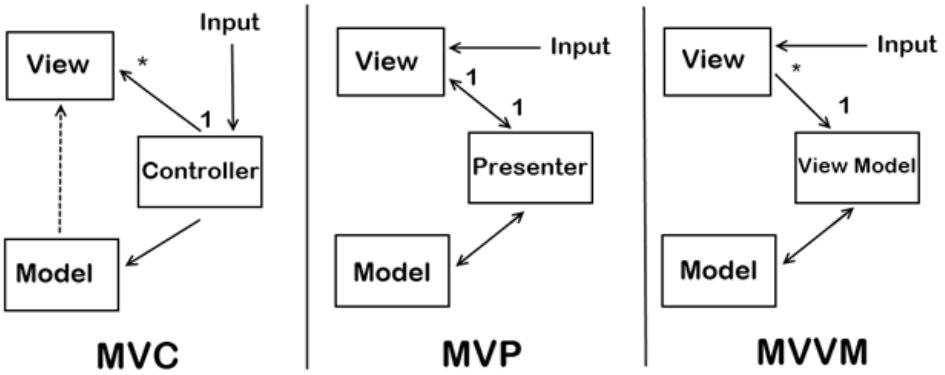
Při výběru architektury pro Android aplikaci musíme brát v potaz, že Android obsahuje komponenty, které jsou svázané některými pravidly, takže použití konkrétní architektury by mohl být problém v navázaní na tyto komponenty.

V následující části představím jedny z nejpoužívanějších architektonických vzorů pro rozdělení prezentační vrstvy u vícevrstvé architektury. Rozdělení prezentační vrstvy jednotlivých architektonických vzorů lze vidět na schématu 2.4.

2.5.1 MVC

Prezentační vrstva je rozdělena na následující komponenty.

[3] Modulační rychlost sériové linky



Obrázek 2.4: Porovnání architektur pro rozdělení prezentační vrstvy, převzato z [12]

- **Model**
  Zpracovává veškeré požadavky, pracuje s daty a komunikuje se službami třetích stran. Po zpracování požadavků informuje view.

- **View**
  Získává data z modelu a zobrazuje je uživateli. V případě změny dat je modelem o změně informován.

- **Controller**
  Propojuje model a view. Zachytává požadavky od uživatele a deleguje je na model. Controller může být sdílen mezi více view, na které si drží odkaz, a dále si drží odkazy na modely, se kterými interaguje.

2.5.2 MVP

Prezentační vrstva je rozdělena na následující komponenty.

- **Model**
  Pracuje s daty a komunikuje se službami třetích stran. Výsledná data předává na presenter.

- **View**
  Zachytává požadavky od uživatele a deleguje je na presenter. Data získané od komponenty presenter zobrazuje uživateli.

- **Presenter**
  Propojuje model a view. Zpracovává požadavky od view a předává data z modelu zpět na view. Řeší veškerou logiku.

2.5.3 MVVM

Používá se nejčastěji ve spojení s Android data binding. Prezentační vrstva je rozdělena na následující komponenty.

- **Model**
  Pracuje s daty a komunikuje se službami třetích stran. Výsledná data předává na viewmodel.
- **View**
  Zachytává požadavky od uživatele a deleguje je na viewmodel.
- **Viewmodel**
  Propojuje model a view. Zpracovává požadavky od view. Řeší veškerou logiku. Pomocí databinding aktualizuje view.

2.5.4 Shrnutí

Architektonický vzor MVC je současně nejvíce využíván při psaní webových aplikací. Při tvorbě Android aplikace, kde používáme aktivity, resp. fragmenty nastává problém definování hranice mezi view a controller.

Architektonické vzory MVP a MVVM si jsou velice podobné. Hlavním rozdílem je násobná vazba mezi view a viewmodel, na rozdíl od vazby 1:1 mezi view a presenter, jak lze vidět na obr. 2.4. Z tohoto důvodu jsem se rozhodl pro použití architektonického vzoru MVVM, protože při použití fragmentů v roli view mezi nimi mohu jednoduše sdílet jediný viewmodel a není potřeba pro každé view vytvářet příslušný presenter, jak by to bylo u vzoru MVP. Mezi další výhodu pak lze počítat automatické aktualizace view při použití Android data binding.

2.6 Analýza požadavků

„*Analýza požadavku je nezbytnou součástí procesu vývoje, realizace a následné implementace většiny softwarových řešení*“[36]. Na základě stanovených požadavků lze určit a následně ověřit, co má daný systém splňovat. Dle předchozí analýzy již mohu definovat požadavky na systém.

2.6.1 Funkční požadavky

Funkční požadavky popisují to, co by měl daný systém dělat, tzn. popisují funkce systému[6]. Seznam funkčních požadavků pro aplikaci resp. knihovnu je následující.

- **F1 USB spojení**
  Aplikace bude umožňovat komunikovat pomocí USB. Aplikace umožní zahájit a ukončit spojení, přijímat a odesílat data.

- **F2 Bluetooth spojení**
  Aplikace bude umožňovat komunikovat pomocí technologie Bluetooth. Aplikace umožní zahájit a ukončit spojení, přijímat a odesílat data.

- **F3 WiFi spojení**
  Aplikace bude umožňovat komunikovat pomocí WiFi na protokolu TCP/IP. Aplikace umožní zahájit a ukončit spojení, přijímat a odesílat data.

- **F4 USB konfigurace**
  Aplikace bude umožňovat definovat parametry USB spojení.

- **F5 Bluetooth konfigurace**
  Aplikace bude umožňovat definovat identifikátor zařízení.

- **F6 WiFi konfigurace**
  Aplikace bude umožňovat definovat údaje o vzdáleném zařízení, jako IP adresu, heslo a timeout spojení.

- **F7 Vyhledaní dostupných zařízení**
  Aplikace umožní vyhledat dostupná vzdálená zařízení a zobrazit je.

2.6.2 Nefunkční požadavky

Nefunkční požadavky jsou podmínky omezující daný systém, specifikují způsob, jakým bude systém implementován, a definují podmínky na systém, které nesouvisí s uživatelským používáním systému[6]. Vzhledem k velikosti vyvíjeného systému převládají nefunkční požadavky nad požadavky funkčními. Jedná se o tyto požadavky.

- **N1 Android**
  Aplikace bude vytvořena jako nativní pro Android. Minimální podporovaná verze systému bude 4.0.3. (API level 15).

- **N2 Jazyk**
  Aplikace bude napsána v jazyce Java.

- **N3 Knihovna**
  Funkčnost vyhledávaní zařízení a navazování komunikace bude odděleno do samostatné knihovny.

- **N4 Architektura**
  Aplikace bude napsána s využitím vícevrstvé architektury, kde pro rozdělení prezentační vrstvy bude použit architektonický vzor MVVM.

- **N5 Bluetooth**
  Aplikace se bude chovat, při volbě technologie Bluetooth, jako klient.

- **N6 WiFi**
  Aplikace se bude chovat, při volbě technologie WiFi, jako klient.
- **N7 Překlad**
  Aplikace bude v českém a anglickém jazyce.
- **N8 Stabilita**
  Aplikace se musí vypořádat se změnou orientace, nebo omezením prostředků a obnovit se do původního stavu.
- **N9 Výkon**
  Aplikace nebude při komunikaci blokovat hlavní vlákno.

2.7 Model případů užití

Modelování případů užití je doplňkovým způsobem získání a dokumentování požadavků. Modelování případů užití se skládá z nalezení:

- hranic systému,
- aktérů,
- případů užití[7].

V případě potřeb provádíme toto modelování opakovaně. Výstupem je model případů užití.

Jak už bylo několikrát zmíněno, implementovaný systém (aplikace) je do rozsahu velice malý, dle [8] není v takovém případě modelování případů užití příliš vhodné, i přesto, hlavně z ilustračních důvodů, jsem se rozhodl toto modelování provést.

2.7.1 Aktéři

„*Aktéři jsou role, přidělené osobám, nebo předmětům používající daný systém*“[7]. V našem systému se vyskytuje pouze jeden aktér.

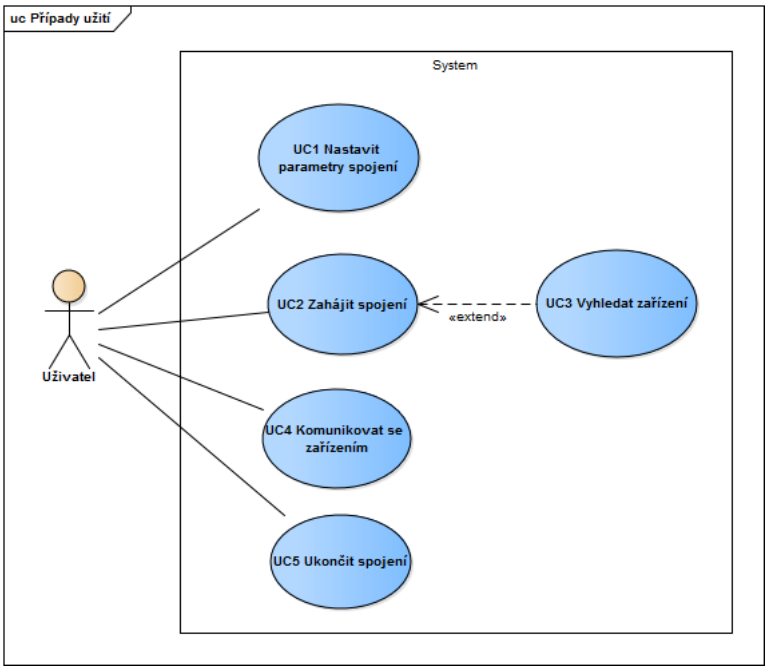
2.7.1.1 Uživatel

Uživatel je osoba používající systém. Osoba se stává uživatelem po nainstalovaní a spuštění aplikace.

2.7.2 Případy užití

Popis jednotlivých případů užití z obr. 2.5.

- **UC1 Nastavit parametry spojení**
  Systém umožní uživateli nakonfigurovat daný typ spojení.



Obrázek 2.5: Model případů užití (Use cases model)

- **UC2 Zahájit spojení**
  Systém umožní uživateli navázat spojení s vybraným vzdáleným zařízením.

- **UC3 Vyhledat zařízení**
  Systém umožní uživateli vyhledat vzdálená zařízení v dosahu, aby mohl s některým z nich navázat spojení.

- **UC4 Komunikovat se zařízením**
  Systém umožní uživateli komunikovat se zařízením, se kterým uživatel navázal spojení.

- **UC5 Ukončit spojení**
  Systém umožní uživateli ukončit navázané spojení.

2.7.3 Pokrytí případů užití

Nyní už máme definováno, co má daný systém splňovat z pohledu uživatele (případy užití) i z pohledu samotného systému (funkční požadavky). Abychom dostali úplný pohled na fungování požadovaného systému, je výhodné oba získané pohledy propojit. Za tímto účelem vytváříme tzv. matici pokrytí. Na základě matice pokrytí lze jednoduše určit, zda máme pokryté všechny poža-

davky, případně na který požadavek jsme zapomněli. Matici pokrytí lze vidět v tab. 2.2.

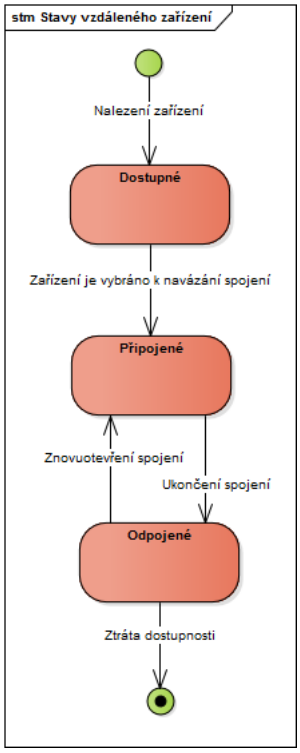
	UC1	UC2	UC3	UC4	UC5
F1		✓		✓	✓
F2		✓		✓	✓
F3		✓		✓	✓
F4	✓				
F5	✓				
F6	✓				
F7			✓		

Tabulka 2.2: Matice pokrytí případů užití

2.8 Diagram aktivit

Diagramy aktivit jsou vývojové diagramy, které slouží k modelování obchodních procesů. Každý diagram aktivit slouží k popsání konkrétního aspektu chování systému[9]. Diagram aktivit zachycuje tok mezi jednotlivými aktivitami daného obchodního procesu. Přechody mezi jednotlivými aktivitami bývají často doprovázeny změnami stavů vstupních objektů. Diagramy, které zachycují všechny přechody mezi jednotlivými stavy objektu, se nazývají stavové diagramy. Proto při modelování diagramů aktivit dochází často k vytváření stavových diagramů příslušných objektů.

Aplikace obsahuje několik obchodních procesů, rozhodl jsem se vytvořit diagram pouze pro hlavní obchodní proces, kterým je komunikace mezi zařízeními. Aktivity diagram pro proces komunikace lze vidět na diagramu C.3. Vstupní objekt v procesu navázání komunikace je vzdálené zařízení. Změny stavů vzdáleného zařízení lze vidět na stavovém diagramu 2.6.



Obrázek 2.6: Stavový diagram vzdáleného zařízení

Kapitola 3

Implementace

V této kapitole popíši řešení, které jsem v rámci bakalářské práce implementoval. Řešení se skládá z knihovny, která umožňuje navázání spojení mezi zařízením se systémem Android a Arduinem, dále pak z aplikace, která simuluje použití knihovny a demonstruje funkčnost implementované knihovny a poslední částí řešení je aplikace pro vývojový kit Arduino na demonstrování a otestování funkčnosti knihovny, respektive aplikace. Poslední části kapitoly jsou věnované problémům a omezením, které jsem musel řešet při vytváření aplikace, a v závěru představím knihovny použité při implementaci.

3.1 Knihovna

Knihovna Coupler[4] zajišťuje komunikaci se vzdáleným Arduino zařízením pomocí USB, Bluetooth, nebo WiFi. Při implementaci jsem volil taková řešení, aby se použití konkrétního druhu komunikace nelišilo od ostatních druhů. Proto jsem mohl využít podobnosti rozhraní jednotlivých tříd zajišťující konkrétní druh komunikace a pro společnou funkčnost vytvořit rodičovskou třídu. Hlavní funkce knihovny lze rozdělit na následující úkoly:

- vyhledání dostupných zařízení,
- navázání spojení,
- komunikace a obsluha spojení.

Pro vytváření instancí tříd zajišťující určitý druh spojení jsem použil návrhový vzor builder (stavitel), který umožňuje jednodušší a přehlednější inicializaci. Generická třída `BaseBuilder` zajišťuje inicializaci společných atributů pro jednotlivé druhy komunikace a definuje rozhraní, které musejí potomci implementovat, pro vytvoření instance, jak lze vidět na příkladech vytvoření

[4]Pracovní název knihovny

instance třídy zajišťující spojení (USB spojení: 3.1, WiFi spojení: D.1, Bluetooth spojení: D.2). Rozhraní rodičovské třídy je popsáno níže.

- `BaseBuilder.setInformationCallback(Information)`
  Nastaví referenci pro informování uživatele o probíhajících operacích viz. 3.1.3.1.
- `BaseBuilder.setDiscoveringCallback(DiscoveredDevice)`
  Nastaví referenci pro informování o dostupných zařízeních viz. 3.1.3.3.
- `BaseBuilder.setResponseCallback(DeviceResponse)`
  Nastaví referenci pro předání odpovědi z připojeného zařízení viz. 3.1.3.2.
- `BaseBuilder.create(Context)`
  Vytvoří a inicializuje instanci.

```
new UsbBuilder().setInformationCallback(this)
                .setDiscoveringCallback(this)
                .setResponseCallback(this)
                .setBaudRate(9600)
                .setDataBits(UsbSerialInterface.DATA_BITS_8)
                .setStopBits(UsbSerialInterface.STOP_BITS_1)
                .setParity(UsbSerialInterface.PARITY_NONE)
                .setFlowControl(UsbSerialInterface.FLOW_OFF)
                .restoreState(savedInstanceState)
                .create(getApplicationContext());
```

Zdrojový kód 3.1: Vytvoření instance třídy pro USB spojení

3.1.1 Coupler

Třídy mající v názvu Coupler jsou hlavními komponentami knihovny. Spojují dohromady všechny potřebné komponenty pro vytvoření a obsluhu příslušného druhu spojení. Poskytují rohraní pro vyhledávání dostupných zařízení a rozhraní pro komunikaci se zařízením. Jedná se o tři třídy, `UsbCouler`, `BluetoothCouler` a `WifiCouler`, dle druhu komunikace, který jednotlivé třídy zajišťují. Společnou funkčnost zajišťuje rodičovská třída `BaseCoupler`. Rodičovská třída registruje příslušný BroadcastReceiver[5] (druh je definován potomkem) pro získávání informací o změnách dostupných zařízeních. Třída si drží aktuální informace o připojeném zařízení a v případě potřeby tyto informace uloží, aby mohlo dojít ke korektnímu obnovení spojení. Toto je důležitá funkčnost, protože systém Android může běžící aplikaci „restartovat“ (děje se například při změně orientace displaye) a tím dojde k vytvoření nové instance třídy pro komunikaci. K obnovení původního stavu slouží uložené informace.

[5] Android komponenta poslouchající změny systému

3.1.2 Service

Třída, která má v názvu CouplerService, slouží k obsluze spojení a poskytuje rozhraní pro obsluhu spojení. Poskytované rozhraní lze vidět na 3.2. Jedná se o třídy:

- `UsbCouplerService`,
- `BluetoothCouplerSevice`,
- `WifiCouplerService`,

Společná funkčnost je implementována v rodičovské třídě `BaseCouplerService`.

```
    /**
     * Zahájí již vytvořené spojení.
     */
    void start();

    /**
     * Ukončí spojení.
     */
    void stop();

    /**
     * Odešle zprávu vzdálenému zařízení.
     *
     * @param bytes Zpráva v bytech
     */
    void send(@NonNull byte[] bytes);

    /**
     * Vytvoří a zahájí spojení pro vzdálené zařízení.
     * Ukončí poslední probíhající spojení, pokud je to potřeba.
     *
     * @param device Vzdálené zařízení
     */
    void startForNewDevice(@NonNull T device);
```

Zdrojový kód 3.2: Rozhraní pro obsluhu spojení

3.1.3 Callbacks

Důležitou a nedílnou součástí navržené knihovny jsou tzv. callbacks. Slouží pro předávání informací zpět k volajícímu funkcí knihovny, nebo přímo uživateli. Knihovna definuje hned několik druhů callbacks, které se liší podle

předávané informace. Takovéto rozdělení jsem zvolil záměrně, aby bylo možné delegovat společné funkce na rozdílné třídy.

3.1.3.1 InformationCallback

Callback slouží pro předávání informací o probíhajících změnách, jako například čekání na spojení, vytvoření, nebo ukončení spojení. Definované rozhraní a popis funkcí lze vidět na ukázce kódu 3.3.

```
/**
 * Předá uživateli zprávu.
 *
 * @param message Text zprávy
 */
void inform(@NonNull String message);

/**
 * Spustí indikaci čekání se zprávou.
 *
 * @param message Text titulku indikátoru
 */
void startProgress(@NonNull String message);

/**
 * Ukončí běžící indikaci čekání.
 *
 * @param message Text výsledku operace,
 *                může být {@code null}
 */
void stopProgress(@Nullable String message);
```

Zdrojový kód 3.3: Rozhraní InformationCallback

3.1.3.2 ResponseCallback

Callback slouží pro komunikaci se vzdáleným zařízením. Umožňuje předání zprávy od vzdáleného zařízení. Definované rozhraní a popis funkcí lze vidět na ukázce kódu 3.4.

3.1.3.3 DiscoveringCallback

Callback definuje rozhraní pro informovaní o změnách dostupnosti vzdálených zařízeních. Definované rozhraní a popis funkcí lze vidět na ukázce kódu 3.5.

```java
/**
 * Předá přijatou zprávu od vzdáleného zařízení
 *
 * @param bytes Zpráva v bytech
 */
void onReceivedData(byte[] bytes);
```

Zdrojový kód 3.4: Rozhraní ResponseCallback

```java
/**
 * Informuje o nalezení nového zařízení.
 *
 * @param device Dostupné vzdálené zařízení
 */
void addDevice(@NonNull T device);

/**
 * Informuje o ztrátě zařízení.
 *
 * @param device Již nedostupné vzdálené zařízení
 */
void removeDevice(@NonNull T device);

/**
 * Informuje o dostupných zařízeních.
 * Využíváno pokud jsou volány metody na manuální
 * prohledávání okolí, nebo při první inicializaci knihovny.
 *
 * @param devices Dostupná vzdálená zařízení
 */
void updateKnowDevices(@NonNull List<T> devices);
```

Zdrojový kód 3.5: Rozhraní DiscoveringCallback

3.1.4 Použití

Knihovna nabízí, kromě již zmíněného použití přímého vytvoření instance příslušného druhu spojení, využít už předpřipravených řešení. Pro každý druh spojení knihovna nabízí příslušnou aktivitu, resp. fragment. V takovém případě stačí pouze podědit z příslušné třídy a přetížit potřebné metody.

3.1.5 Omezení

Android zařízení se pro bezdrátové komunikace chová jako klient, tudíž provádí obsluhu spojení. Aby mohla komunikace probíhat neomezeně až do doby, kdy se ji klient rozhodně ukončit, udržuje klient spojení stále otevřené. Z tohoto důvodu není schopen poznat situaci, kdy dojde k přerušení spojení na straně serveru. Tuto situaci je schopen poznat až v případě, že se pokusí se serverem komunikovat.

3.2 Android aplikace

Aplikace nabízí uživateli volbu druhu komunikace. Dále umožňuje v nastavení definovat parametry pro jednotlivé druhy komunikace. Hodnoty parametrů jsou uloženy, takže při opětovném spuštění aplikace dojde k obnovení hodnot podle poslední změny.

Uživatelské rozhraní je jednotné pro všechny druhy komunikace, s výjimkou komunikace pomocí USB, která nemá volbu manuálního vyhledávání nových zařízení, protože v případě připojení nového zařízení do portu USB, knihovna automaticky informuje aplikaci.

Po zvolení druhu komunikace zobrazí aplikace uživateli dostupná vzdálená zařízení, se kterými může uživatel navázat spojení. Současně může být otevřené pouze jedno spojení. Po výběru zařízení může uživatel libovolně otevírat, nebo uzavírat spojení, komunikovat se zařízením, nebo změnit vybrané zařízení.

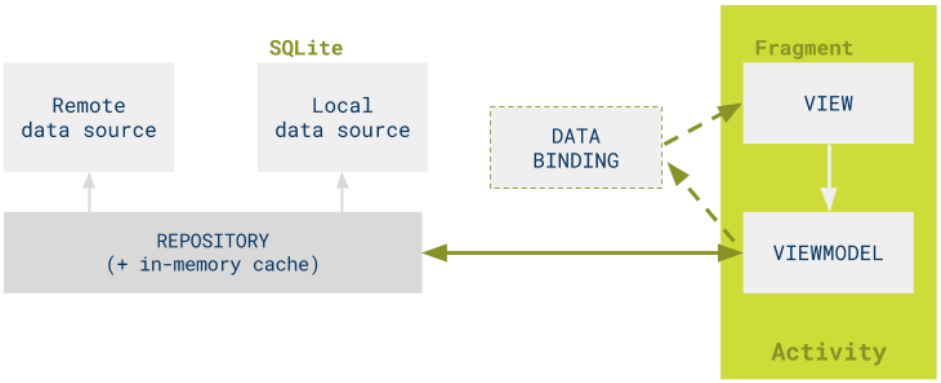
Ke komunikaci slouží editovatelné pole, kam může uživatel zadat desítkové číslo v intervalu $\langle 0, 9999 \rangle$ a odeslat ho do zařízení. Odpověď od zařízení se uživateli zobrazí v příslušném textovém poli. Aplikace nespecifikuje žádný komunikační protokol, pouze odesílá a přijímá data. Aplikace je založena na architektuře MVVM s Android data binding podle obr. 3.1.

3.2.1 View

View komponenty jsou všechny aktivity, fragmenty a dialogy. Propojení mezi view a viewmodel probíhájí na aktivitě, která si drží referenci na svůj viewmodel a deleguje na něj potřebné metody životního cyklu.

3.2.2 Viewmodel

Viewmodel je implementovaný jako retain fragment, díky tomu nezaniká při změně orientace displaye, tudíž si může držet konkrétní stavy aplikace. Aplikace obsahuje celkem tři viewmodely. První viewmodel je používán na hlavní obrazovce. Neobsahuje žádný model, pouze řeší požadavky, které na něho deleguje view, konkrétně se jedná o zobrazení správné obrazovky po



Obrázek 3.1: Schéma architektury aplikace, převzato z [15]

výběru položky z menu. Druhý viewmodel na obrazovce s nastavením řeší požadavky z view, jako získání a uložení dat nastavení, a komunikuje s modelem pro obsluhu lokálního úložiště. Poslední viewmodel zajišťuje obsluhu samotné komunikace se vzdáleným zařízením, tzn. komunikuje s modelem pro získávání dat ze vzdálených úložišť.

3.2.3 Model

Aplikace obsahuje dva modely. První slouží pro získávání dat z lokálního úložiště, druhý pro získávání dat ze vzdáleného úložiště. Jako lokální úložiště jsem použil Android SharedPreferences[6], kam jsou ukládány volitelné parametry spojení. Vzdálené úložiště je vlastně Arduino zařízení, se kterým probíhá komunikace přes funkce vytvořené knihovny.

3.3 Arduino aplikace

Aplikace pro Arduino vznikla především z důvodu otestování funkčnosti knihovny, resp. Android aplikace. Aplikace je napsaná pro Arduino Mega, nebo jiné druhy Arduina, které mají alespoň tři dvojice RX, TX pinů pro sériovou komunikaci.

Jedná se o jednoduchou aplikaci, která přijme desítkové číslo na intervalu $\langle 0, 9999 \rangle$ a zobrazí ho na sedmisegmentovém displayi. Po přijetí čísla, na některém vstupním sériovém pinu, aplikace odešle, na příslušný výstupní sériový pin, zprávu s informací, na kterém pinu bylo číslo přijato.

Pro testování WiFi komunikace byla vytvořena vlastní aplikace, protože komunikace s modulem ESP8266 probíhá pomocí speciálních příkazů (AT

[6] Úložiště pro data aplikace, bez nutnosti používání vlastní databáze

commands), která pouze přeposílá data mezi sériovými porty TX, RX a sériovými porty TX2, RX2. Po připojení Arduina k počítači pomocí USB, pak posíláním příkazů přes Serial Monitor komunikujeme s WiFi modulem.

3.4 Životní cyklus

Nedílnou součástí Android aplikace je její tzv. životní cyklus. V podstatě se jedná o sled metod konkrétní Android komponenty, kde každá metoda má svoji funkci a je volána v konkrétních případech (například metoda `Activity.onResume()` je volána, když daná aktivita přechází do popředí, tzn. je viditelná uživatelem). Nejčastěji se lze potkat s životním cyklem aktivity, nebo fragmentu, které lze pro ilustraci vidět na schématu C.2.

V našem případě bylo nutné volání některých metod ze životního cyklu delegovat na třídy knihovny vytvářející spojení. Daný problém je možný vyřešit několika způsoby, při vývoji knihovny jsem se musel rozhodnout, který způsob použiji.

3.4.1 ActivityLifecycleCallbacks

První možností, která se nabízela, bylo implementovat rozhraní `Application.ActivityLifecycleCallbacks` a následně tyto callbacks zaregistrovat na Application třídě. Problém tohoto řešení je v tom, že každá aktivita v aplikaci má tyto callbacks zaregistrované. Pokud jsem například přecházel z aktivity, která zajišťovala spojení, zpět do aktivity, která zobrazovala menu, došlo na třídě k volání metod, jak z důvodu opouštění původní aktivity, tak z důvodu přechodu na novou aktivitu. Z tohoto důvodu jsem dané řešení zavrhl.

3.4.2 Fragment

Další možností bylo, aby třída zajišťující spojení byla implementovaná jako fragment a při vytváření instance se takový fragment přidal na aktivitu. Pokud bychom navíc takový fragment implementovali jako tzv. retain fragment, nebylo by potřeba řešit ukládání stavů. Z důvodu větší použitelnosti knihovny (použití knihovny by bylo možné pouze na aktivitě) jsem se pro toto řešení nerozhodl.

3.4.3 Volání metod

Poslední a mnou zvolené řešení bylo všechny potřebné metody provázat s knihovnou manuálně, při použití knihovny, jak lze vidět na příkladu D.3. Jedná se sice o nejméně automatizované řešení, ale z důvodů zmíněných výše jediné možné.

3.5 Oprávnění aplikace (permission)

Oprávnění aplikace (angl. permission) slouží k vymezení pravomocí aplikace. Každá Android aplikace má definovaná oprávnění, která požaduje udělit. Uživatel při instalaci takové aplikace vidí, k čemu všemu aplikace požaduje přístup. Android definuje oprávnění například pro čtení kontaktů, uskutečňování hovorů, nebo přístup k lokaci zařízení. Oprávnění aplikace jsou jedním z bezpečnostních prvků systému Android.

Do Android OS verze 6.0 musel uživatel při instalaci všechna oprávnění potvrdit, jinak si aplikaci nemohl nainstalovat. Pokud aplikace podporuje operační systém verze 6.0 a vyšší, je potřeba zahrnout ošetření těchto oprávnění. Od již zmíněné verze 6.0 se oprávnění rozdělily na následující dvě skupiny.

3.5.1 Normal permission

Oprávnění, které mají stejné chování jako do verze 6.0, tzn. uživatel je potvrzuje při instalaci aplikace. V knihovně se jedná konkrétně o tato oprávnění.

- ACCESS_WIFI_STATE – povolení pro přístup k WiFi,
- BLUETOOTH – povolení pro přístup k Bluetooth,
- BLUETOOTH_ADMIN – povolení pro zapnutí Bluetooth,
- CHANGE_WIFI_STATE – povolení pro zapnutí WiFi,
- INTERNET – povolení pro získání informace o připojení.

3.5.2 Runtime permission

Oprávnění, u kterých má uživatel volbu, zda se je rozhodne udělit, nebo ne. Své rozhodnutí může ovšem kdykoliv změnit a aplikace na to musí umět reagovat. Pokud aplikace pro vykonání určitého požadavku potřebuje oprávnění, které nemá, může se uživatele dotázat a zachovat se podle výsledku. V knihovně se jedná konkrétně o toto oprávnění.

- ACCESS_COARSE_LOCATION – povolení pro určení polohy.

3.6 Použité knihovny

V této podkapitole představím volně dostupné knihovny použité při vývoji aplikace. Všechny použité knihovny jsou do projektu přidány ve formě dynamických závislostí, tzn. při sestavování aplikace jsou knihovny vyhledávány a následně stahovány ze vzdálených repozitářů. Příklad definice dynamické závislosti lze vidět na 3.6.

```
compile 'com.jakewharton.timber:timber:4.5.1'
//nebo
compile group: 'com.jakewharton.timber',
        name: 'timber',
        version: '4.5.1'
```

Zdrojový kód 3.6: Definice závislosti v Gradle

3.6.1 Gradle Retrolambda Plugin

Rozšíření pro sestavovací nástroj Gradle umožňují využívat funkčností Javy verze 6 a vyšší. Android standardně podporuje Javu verze 6 s některými rozšířeními z vyšších verzí. V aplikaci využívám z důvodu podpory funkcionálního programování a lambda funkcí, která byla přidána do Javy verze 8.

3.6.2 v7 Support Libraries

Support Libraries rozšiřují základní funkčnost z Android SDK a jsou nezávislé na verzi systému Android. Lze je použít pro Android 2.3 (API level 9) a vyšší. Z dostupných knihoven jsem použil následují tři knihovny.

- **v7 appcompat library**
  Přidává podporu ActionView, material design a AppCompat komponent.
- **v7 cardview library**
  Slouží pro docílení efektu karty pod UI komponentou.
- **v7 recyclerview library**
  Obsahuje UI komponentu pro vytváření seznamů. `RecyclerView` umožňuje efektivně zobrazovat seznamy s mnoha položkami.

3.6.3 Annotations Support Library

Java anotace slouží k vyšší specifikaci chování parametrů, nebo metod, umožňují statickou analýzu kódu a celkově zpřehledňují výsledný kód. Knihovna definuje anotace, jako například `@NonNull`, `@Nullable`, nebo Android anotace pro statické zdroje (texty aplikace, velikosti UI elementů, atd.), jako například `@StringRes`, `@DrawableRes` nebo `@LayoutRes`.

3.6.4 Design Support Library

Jedná se o rozšíření standardních Android UI komponent o Android material design komponenty, jako floating action button (plovoucí tlačítko), navigation drawers (navigační postranní lišta), nebo záložky. V aplikaci používám `TextInputLayout` v kombinaci s `TextInputEditText`, umožňující zobrazení textu nápovědy zadávacího pole, při psaní, nad polem.

3.6.5 Timber

Knihovna slouží pro vytváření aplikačních logů. Pro zaznamenávání logů využívá standardní Android `Log` třídu a přidává další funkčnosti, jako automatické přiřazení místa volání logu, nebo možnost zpracování zachycených logů před samotným zaznamenáním logu.

3.6.6 UsbSerial

Popis knihovny je v rámci části 1.2.1. Použití knihovny jsem zvolil z důvodu velké variability pro definovaní USB spojení.

3.6.7 ReactiveX

Knihovna přidává podporu reaktivního programovaní. Reaktivní programování je určitá nástavba funkcionálního programování, která rozšiřuje funkcionální programovaní o zpracování v čase. Přesněji řečeno umožňuje zpracovávat data a po dokončení požadovaných operací informovat o změně.

- **RxJava2**
  Knihovna přidává podporu reaktivního programování pro jazyk Java. Hlavními komponentami jsou například `Observable`, nebo `Schedulers`.

- **RxAndroid2**
  Rozšiřuje funkčnost knihovny RxJava a umožňuje vytvářet reaktivní komponenty v Android aplikacích. Přidává například komponentu `AndroidSchedulers` pro provádění funkcí na hlavním vlákně aplikace.

3.6.8 LeakCanary

Nástroj slouží k analyzování práce s pamětí aplikace. Umožňuje odhalit tzv. memory-leaks (místa, která drží referenci na objekt, takže paměť tohoto objektu nemůže být uvolněna).

3.6.9 Knihovny pro testování

Níže zmíněné knihovny nejsou součástí standardního sestavení aplikace. K použití knihoven dochází pouze v případě sestavení aplikace pro spouštění automatického testování.

- **JUnit**
  Knihovna podporuje tvorbu jednotkových testů v jazyce Java.

- **mockito-core**
  Knihovna obsahuje základní komponenty pro vytváření zástupných objektů požadovaných tříd a vytváření volání metod.

Kapitola 4

Testování

V kapitole představím metody testování výsledné aplikace, použitá testovací zařízení a dva hlavní testovací scénáře. V závěru shrnu problémy a chyby, na které jsem narazil během testování.

4.1 Automatické testování

Před spuštěním automatického testovaní je nejprve potřeba provést kroky definované v programátorské příručce v kapitolách F.1 a F.2, kde není potřeba provádět instalaci Android Studia, ale stačí jen instalace Android SDK. Po nastavení systémového prostředí lze pak testy pustit následujícími příkazy (v závislosti na operačním systému).

```
#UN*X
cd <cesta_k_adresáři_s_projektem>/ConnEx
chmod u+x gradlew
./gradlew test
```

```
@rem Windows
cd /d <cesta_k_adresáři_s_projektem>\ConnEx
gradlew.bat test
```

4.2 Uživatelské testování

Uživatelské testování probíhalo, podle testovacích scénářů definovaných níže, na několika různých zařízení. Pro testování bylo použito Arduino Mega s Bluetooth modulem BT_BOARD v1.5 a s WiFi modulem ESP8266. Pro testování WiFi komunikace bylo použito i druhé Android zařízení, na kterém byl spuštěn jednoduchý server pro obsluhu požadavků.

4.2.1 Testovací zařízení

Uživatelské testy byly prováděny na třech zařízeních různých značek a verzí operačního systému a na jednom emulovaném zařízení. Testovací zařízení a jejich parametry jsou shrnuty v tab. 4.1. Na emulovaném zařízení bylo otestováno pouze správné chování UI aplikace, testy komunikace nebyly prováděny z důvodů uvedených v části 2.4.1.4.

4.2.2 Testovací scénáře

Testování aplikace proběhlo na základě následujících testovacích scénářů:

- Komunikace
  1. Výběr technologie komunikace (USB, Bluetooth, WiFi)
  2. Vybrat zařízení
     - Známé zařízení
     - Vyhledat dostupná zařízení
  3. Odeslání zprávy
  4. Kontrola odpovědi
  5. Ukončení spojení
  6. Obnovení spojení
  7. Výběr jiného zařízení
  8. Opakovat body 3, 4, 5, 6
  9. Změnit orientaci zařízení
  10. Opakovat body 3, 4, 5, 6
- Nastavení
  1. Vstoupit do nastavení
  2. Změnit vybranou možnost
  3. Vrátit se do menu
  4. Vstoupit do nastavení a zkontrolovat změnu
  5. Provést další změnu
  6. Ukončit aplikaci (úplné zastavení aplikace)
  7. Otevření aplikace
  8. Vstoupit do nastavení a zkontrolovat změnu

4.3 Zjištěné nedostatky

Na základě testování byly zjištěny některé nedostatky a chyby. Pokud to bylo možné, byly chyby opraveny a nedostatky zapracovány do aplikace. Všechny informace a důvody jsou shrnuty v tab. 4.2.

Výrobce	**Model**	**Verze systému**	**API level**	**Rozlišení displaye ["]**	**Podporovaná komunikace**
LG	Nexus 5	6.0.1 Marshmallow	23	4,95	USB, WiFi, Bluetooth
HTC	Desire X	4.1.1 Jelly Bean	16	4	USB, WiFi, Bluetooth
SAMSUNG	Galaxy Tab A	6.0.1 Marshmallow	23	10,1	USB,WiFi, Bluetooth
	Emulátor	4.0.3	15	3,2	

Tabulka 4.1: Zařízení použité při testování

Chyba/Nedostatek	**Příčina**	**Opraveno**
Po manuálním vyhledání WiFi sítí, není nalezena žádná sít	Chyba od Android verze 6, musejí být povoleny polohové služby	Ano
Nezdařené připojení k neznámé WiFi sítí	Připojení k sítí probíhá asynchronně	Ano
Duplikace stejných zařízení v seznamu	Chyba implementace	Ano
Po změně orientace nedojde k obnovení spojení	Chyba Android Bluetooth API	Ano
Špatná indikace veřejných sítí	Chyba implementace	Ano

Tabulka 4.2: Výsledky testování

Závěr

Cílem bakalářské práce bylo analyzovat, navrhnout a implementovat knihovnu pro zařízení se systémem Android na řízení vývojového kitu Arduino. To znamená umožnit (zprostředkovat) komunikaci pro předávání příkazů mezi zařízeními. Součástí tohoto cíle je také výsledný produkt otestovat a představit výsledky provedených testů. Informace, jako možnosti, ale hlavně omezení zařízení Android, zařízení Arduino, či jednotlivých technologií, zjištěné během analýzy problému, byly zohledněny a využity při samotném návrhu a implementaci řešení. Vytvořená knihovna, resp. aplikace umožňuje všechny požadované druhy komunikace (USB, WiFi, Bluetooth) a je navržena tak, aby byla snadno rozšiřitelná o další dostupný druhu komunikace, nebo aby pro komunikaci mohl být použit specifický protokol pro požadovanou funkčnost.

Pro budoucí rozvoj by mohla být knihovna rozšířena o některou z dalších dostupných komunikačních technologií, nebo prototyp aplikace rozšířit o možnosti dynamického uživatelského prostředí. Uživatelské prostředí by mohlo nabízet množství konfigurovatelných tlačítek pro přidávání požadované funkčnosti bez nutnosti úpravy zdrojového kódu. V takovém případě by pak aplikace mohla být využívána pro řízení Arduina i bez znalostí vývoje aplikací pro platformu Android.

Literatura

[1] Android Debug Bridge. In: *www.xda-developers.com*, [online], xda-developers, 2015, [cit. 2017-05-01]. Dostupné z: `https://forum.xda-developers.com/wiki/Android_Debug_Bridge`

[2] AMPHAN: *Arduino Smart Home Automation – Aplikace pro Android ve službě Google Play*. [online], Google, Inc., 2007, ©Google, [cit. 2016-11-10]. Dostupné z: `https://play.google.com/store/apps/details?id=arduino.smarthome.automation`

[3] ARDUINO: *Arduino ADK*. [online], Arduino, 2017, ©Arduino, [cit. 2017-05-05]. Dostupné z: `https://www.arduino.cc/en/Guide/ArduinoADK`

[4] ARDUINO.CZ: *Co to je Arduino?*. [online], Arduino.cz, 2014, [cit. 2017-04-29]. Dostupné z: `https://arduino.cz/co-je-to-arduino/`

[5] ARDUINO.CZ: *Arduino IDE*. [online], Arduino.cz, 2017, ©Arduino.cz, [cit. 2017-05-08]. Dostupné z: `https://arduino.cz/arduino-ide/`

[6] ARLOW, Jim a NEUSTADT, Ila: *UML 2 a unifikovaný proces vývoje aplikací: Objektově orientovaná analýza a návrh prakticky*. 2. aktual. a dopl. vyd., Brno: Computer Press, a.s., 2008, ISBN 978-80-251-1503-9, 80 s.

[7] ARLOW, Jim a NEUSTADT, Ila: *UML 2 a unifikovaný proces vývoje aplikací: Objektově orientovaná analýza a návrh prakticky*. 2. aktual. a dopl. vyd., Brno: Computer Press, a.s., 2008, ISBN 978-80-251-1503-9, 91 s.

[8] ARLOW, Jim a NEUSTADT, Ila: *UML 2 a unifikovaný proces vývoje aplikací: Objektově orientovaná analýza a návrh prakticky*. 2. aktual. a dopl. vyd., Brno: Computer Press, a.s., 2008, ISBN 978-80-251-1503-9, 112 s.

[9] ARLOW, Jim a NEUSTADT, Ila: *UML 2 a unifikovaný proces vývoje aplikací: Objektově orientovaná analýza a návrh prakticky*. 2. aktual. a dopl. vyd., Brno: Computer Press, a.s., 2008, ISBN 978-80-251-1503-9, 306 s.

[10] ATC INTEGRATIONS: *Arduino Total Control free – Aplikace pro Android ve službě Google Play*. [online], Google, Inc., 2007, ©Google, [cit. 2016-11-10]. Dostupné z: `https://play.google.com/store/apps/details?id=com.apps.emim.btrelaycontrolfree`

[11] CHELSEA THE DESTROYER: Arduino Board Usage. In: *www.sparkfun.com*, [online], SparkFun Electronics, 2015, [cit. 2017-05-02]. Dostupné z: `https://www.sparkfun.com/news/1982`

[12] D'ARCY: MVVM Compared To MVC and MVP. In: *www.geekswithblogs.net*, [online], Geekswithblogs.net, 2009, [cit. 2017-04-30]. Dostupné z: `http://www.geekswithblogs.net/dlussier/archive/2009/11/21/136454.aspx`

[13] GARTNER: Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 4th quarter 2016. In: *www.statista.com*, [online], Statista, 2017, [cit. 2017-04-28]. Dostupné z: `https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/`

[14] GIUMIG APPS: *Arduino bluetooth controller – Aplikace pro Android ve službě Google Play*. [online], Google, Inc., 2007, ©Google, [cit. 2016-11-10]. Dostupné z: `https://play.google.com/store/apps/details?id=com.giumig.apps.bluetoothserialmonitor`

[15] GOOGLE, I.: todo-mvvm-databinding. In: *github.com*, [online], GitHub, Inc., 2017, [cit. 2017-05-06]. Dostupné z: `https://github.com/googlesamples/android-architecture/tree/todo-mvvm-databinding`

[16] GOOGLE Inc.: Platform Versions. In: *developer.android.com*, [online], Google, Inc., [cit. 2017-04-28]. Dostupné z: `https://developer.android.com/about/dashboards/index.html`

[17] GOOGLE, Inc.: *Accessory Development Kit*. [online], Google, Inc., [cit. 2017-05-05]. Dostupné z: `https://developer.android.com/adk/index.html`

[18] GOOGLE, Inc.: *Android Debug Bridge*. [online], Google, Inc., [cit. 2017-05-01]. Dostupné z: `https://developer.android.com/studio/command-line/adb.html`

[19] GOOGLE, Inc.: *Android Monitor Overview*. [online], Google, Inc., [cit. 2017-04-27]. Dostupné z: https://developer.android.com/studio/profile/android-monitor.html

[20] GOOGLE, Inc.: *Android, the world's most popular mobile platform*. [online], Google, Inc., [cit. 2017-04-28]. Dostupné z: https://developer.android.com/about/android.html

[21] GOOGLE, Inc.: *Bluetooth*. [online], Google, Inc., [cit. 2016-11-10]. Dostupné z: https://developer.android.com/guide/topics/connectivity/bluetooth.html

[22] GOOGLE, Inc.: *Peer-to-Peer*. [online], Google, Inc., [cit. 2016-11-10]. Dostupné z: https://developer.android.com/guide/topics/connectivity/wifip2p.html

[23] GOOGLE, Inc.: *Run Apps on the Android Emulator*. [online], Google, Inc., [cit. 2017-05-01]. Dostupné z: https://developer.android.com/studio/run/emulator.html

[24] GOOGLE, Inc.: *Supporting Different Platform Versions*. [online], Google, Inc., [cit. 2017-04-28]. Dostupné z: https://developer.android.com/training/basics/supporting-devices/platforms.html

[25] GOOGLE, Inc.: *USB Host and Accessory*. [online], Google, Inc., [cit. 2016-11-10]. Dostupné z: https://developer.android.com/guide/topics/connectivity/usb/index.html

[26] GOOGLE, Inc.: USB Host and Accessory. In: *developer.android.com*, [online], Google, Inc., 2017, [cit. 2016-11-10]. Dostupné z: https://developer.android.com/guide/topics/connectivity/usb/index.html

[27] GRADLE: *GitHub - gradle/gradle: Adaptable, fast automation for all*. [online], GitHub, Inc., 2017, ©GitHub, Inc., [cit. 2017-04-27]. Dostupné z: https://github.com/gradle/gradle

[28] JEPPSSON & LÖVSTRÖM: *Arduino Uno Communicator – Aplikace pro Android ve službě Google Play*. [online], Google, Inc., 2007, ©Google, [cit. 2016-11-10]. Dostupné z: https://play.google.com/store/apps/details?id=com.primavera.arduino.listener

[29] MÁDR, Vojtěch: Xamarin: Představujeme nástroj pro multiplatformní vývoj mobilních aplikací (díl 1). In: *www.eman.cz*, [online], eMan s.r.o., 2016, [cit. 2017-05-02]. Dostupné z: https://www.eman.cz/blog/xamarin-predstavujeme-nastroj-pro-multiplatformni-vyvoj-mobilnich-aplikaci-dil-1/

[30] MEIER, Reto: *Professional Android^{TM} 2 Application Development*. Indianapolis: Wiley Publishing, Inc., 2010, ISBN 978-0-470-56552-0, 433 s.

[31] OGBO, Obaro: HTML5 vs Native Android App. In: *www.androidauthority.com*, [online], Android Authority, 2015, [cit. 2017-05-01]. Dostupné z: `http://www.androidauthority.com/html-5-vs-native-android-app-607214/`

[32] SELECKÝ, Matúš: *Arduino: Uživatelská příručka*. Brno: Computer Press, a.s., 2016, ISBN 978-80-251-4840-2, 228 s.

[33] SELECKÝ, Matúš: *Arduino: Uživatelská příručka*. Brno: Computer Press, a.s., 2016, ISBN 978-80-251-4840-2, 50–66 s.

[34] SELECKÝ, Matúš: *Arduino: Uživatelská příručka*. Brno: Computer Press, a.s., 2016, ISBN 978-80-251-4840-2, 23–26 s.

[35] SOUHRADA, Václav: Kotlin – A Language we should know it exists. In: *medium.com*, [online], A Medium Corporation, 2017, [cit. 2017-04-30]. Dostupné z: `https://medium.com/@v.souhrada/kotlin-a-language-we-should-know-it-exists-c1fafe39f9b6`

[36] SOVA NET, s.r.o.: *Analýza požadavků*. [online], SOVA NET, s.r.o., ©SOVA NET, s.r.o., [cit. 2017-04-29]. Dostupné z: `https://www.sovanet.cz/analyza-pozadavku/`

[37] WANG, P.: android, how to do findViewById in Activity when you are using Fragment? (activity and fragment lifecycle). In: *hi.baidu.com*, [online], hi.baidu.com, 2015, [cit. 2017-05-06]. Dostupné z: `http://baiduhix.blogspot.cz/2015/08/android-how-to-do-findviewbyid-in.html`

Příloha **A**

Seznam použitých zkratek

ADK Accessory Development Kit

AOA Android Open Accessory

API Application Programming Interface

CPU Central Processing Unit

GPU Graphic Processing Unit

GSM Global System for Mobile Communications

HTML5 Hyper-Text Markup Language verze 5

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

IoT Internet of Things

IP Internet Protocol

LAN Local area network

MVC Model-View-Controller

MVP Model-View-Presenter

MVVM Model-View-Viewmodel

NDK Native Development Kit

NFC Near-Field Communication

OS Operating System

P2P Peer-to-Peer

RAM Random Access Memory

RFCOMM Radio Frequency Communication

RX Receiver

SD Secure Digital

SDK Software Development Kit

TCP Transmission Control Protocol

TX Transmitter

UI User Interface

USB Universal Serial Bus

Příloha B

Slovník pojmů

Android data binding

Knihovna umožňuje vytváření automatického mapování objektů na XML prvky UI návrhu. Umožňuje automatické aktualizace dat na obrazovce.

Aktivita

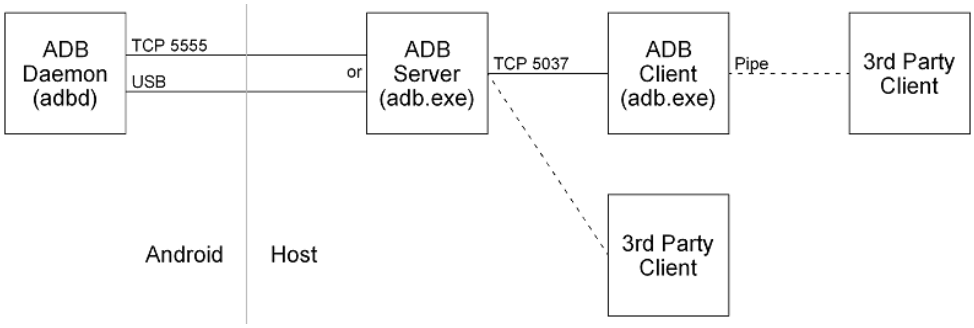
Aktivita je základní stavební prvek Android aplikace. Umožňuje zobrazit uživatelské rozhraní.

Fragment

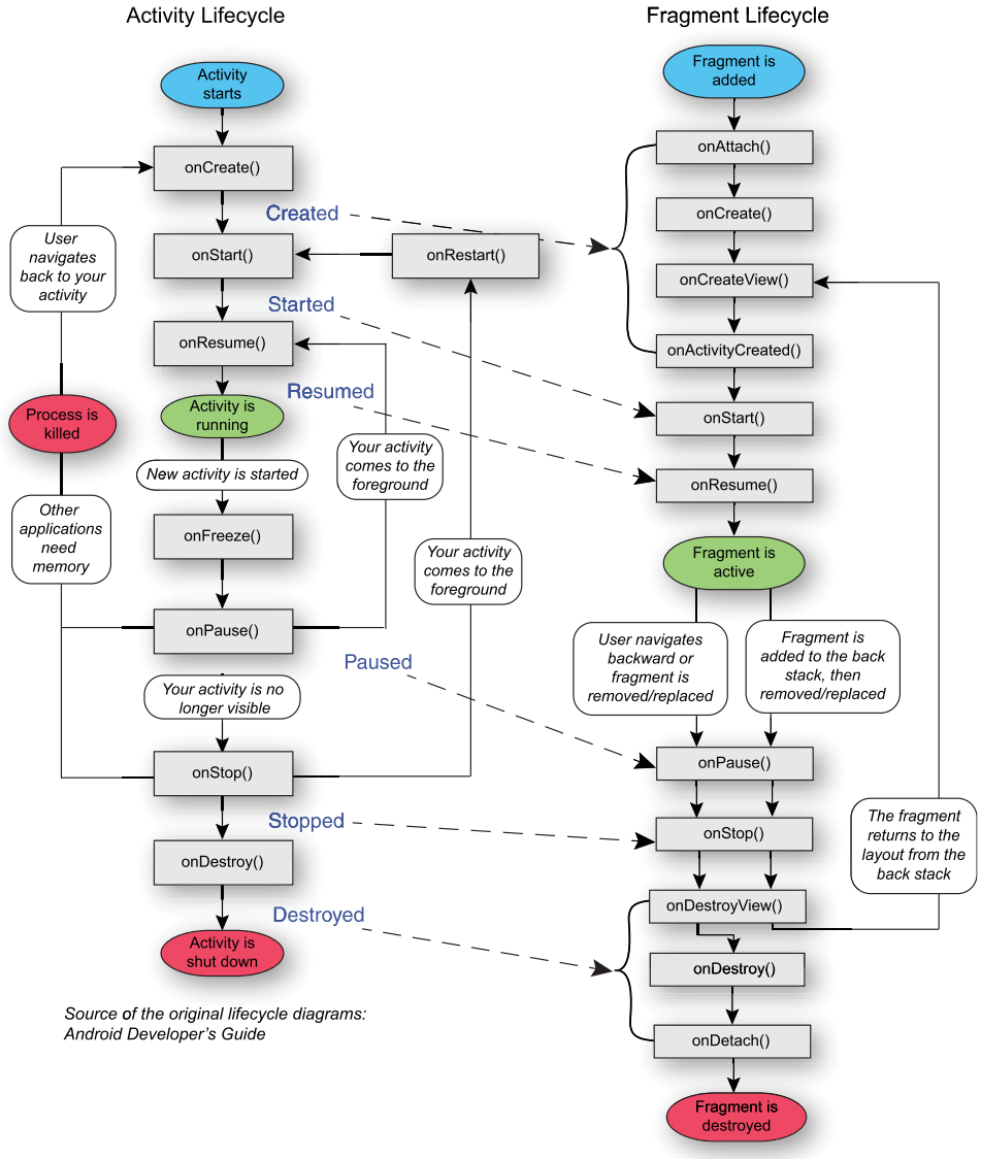
Fragment je další ze základních Android komponent, je součástí aktivity, umožňuje zobrazit uživatelské rozhraní, umožňuje tvorbu dynamického UI.

Příloha C

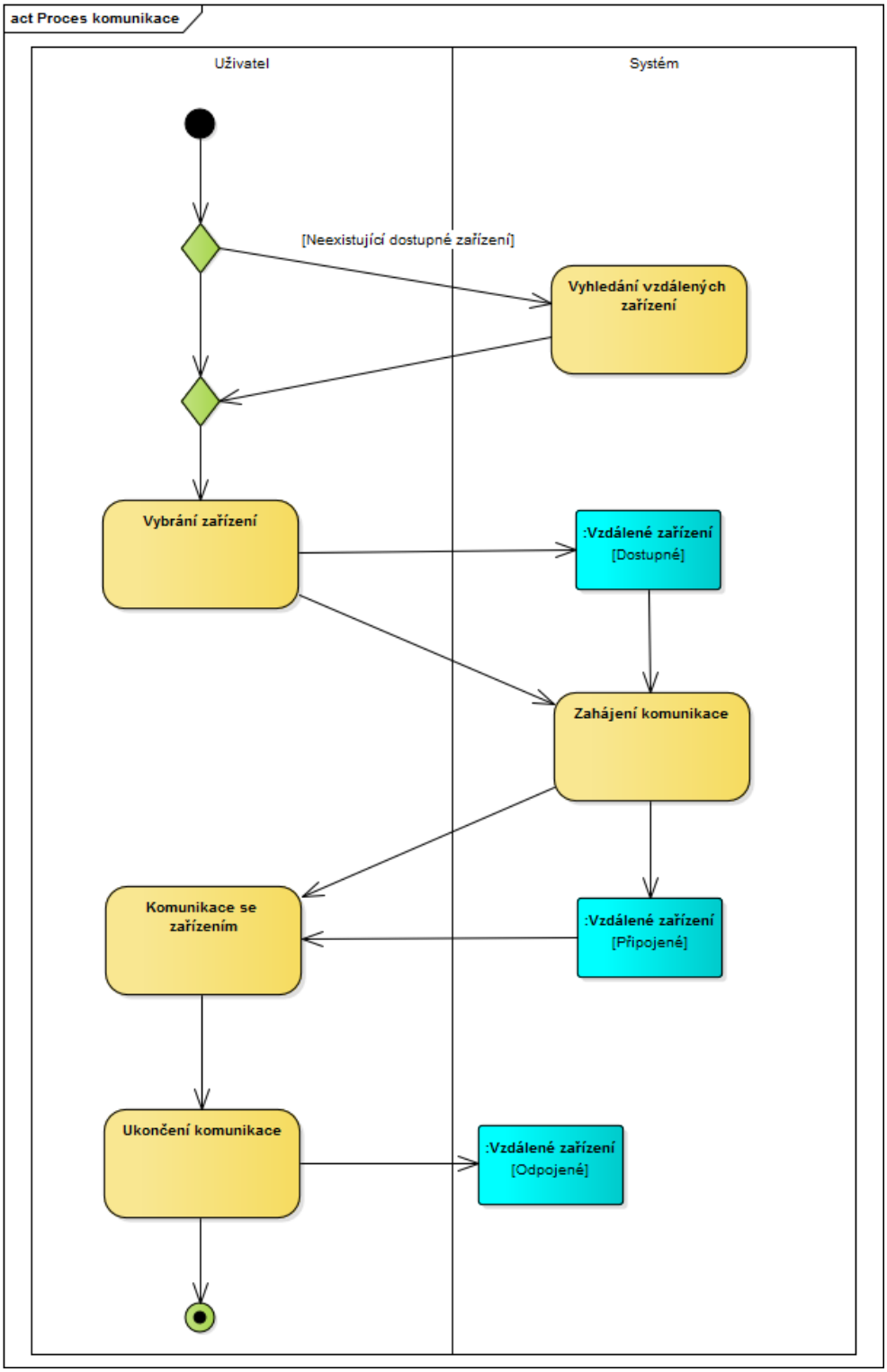
Obrázky a schémata



Obrázek C.1: Android Debug bridge schéma, převzato z [1]



Obrázek C.2: Životní cyklus aktivity a fragmentu, převzato z [37]



Obrázek C.3: Aktivity diagram procesu komunikace

Příloha D

Ukázky zdrojových kódů

```
new WifiBuilder().setInformationCallback(this)
                 .setDiscoveringCallback(this)
                 .setResponseCallback(this)
                 .setAddress("192.168.0.1")
                 .setPort(5555)
                 .setTimeout(10000)
                 .restoreState(savedInstanceState)
                 .create(getApplicationContext());
```

Zdrojový kód D.1: Vytvoření instance třídy pro WiFi spojení

```
new BluetoothBuilder().setInformationCallback(this)
                      .setDiscoveringCallback(this)
                      .setResponseCallback(this)
                      .setUuid(getDeviceUuid())
                      .restoreState(savedInstanceState)
                      .create(getApplicationContext());
```

Zdrojový kód D.2: Vytvoření instance třídy pro Bluetooth spojení

```
public class BaseCouplerActivity<C extends BaseCoupler>
        extends AppCompatActivity implements Information,
                                             DeviceResponse {

    @Nullable
    protected C mCoupler;

    @Override
    protected void onStart() {
        super.onStart();
        if (mCoupler != null) {
            mCoupler.onStart();
        }
    }

    @Override
    protected void onStop() {
        super.onStop();
        if (mCoupler != null) {
            mCoupler.onStop();
        }
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        if (mCoupler != null) {
            mCoupler.onSaveInstanceState(outState);
        }
    }
.
.
.
}
```

Zdrojový kód D.3: Delegace metod životního cyklu

Příloha E

Uživatelská příručka

E.1 Požadavky na systém

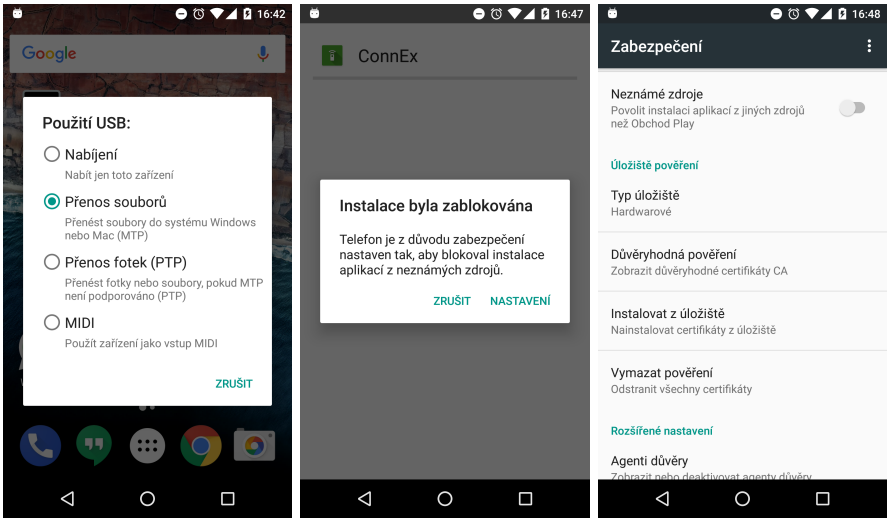
Pro instalaci aplikace je potřeba zařízení s operačním systémem Android. Minimální verze podporovaného systému je 4.0.3 (API level 15). Zařízení musí obsahovat hardware podporu pro technologie USB, Bluetooth a WiFi.

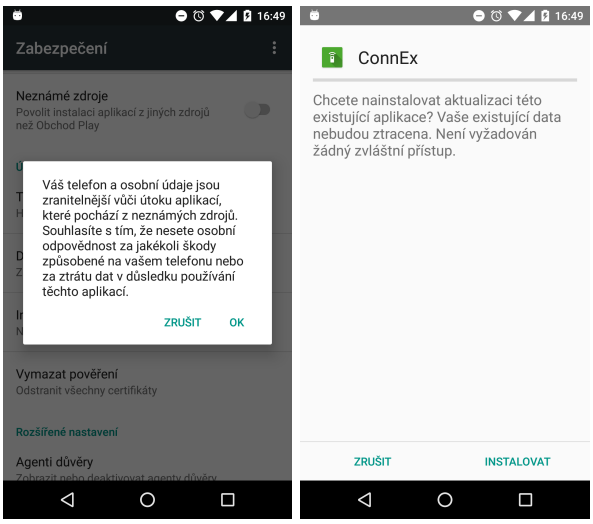
E.2 Instalace aplikace

Některé kroky instalace se mohou nepatrně lišit v závislosti na zařízení. Pro instalaci aplikace je potřeba připojit telefon pomocí USB kabelu k počítači a mít v telefonu nainstalovanou aplikaci pro správu souborů. Pro instalaci je potřeba provést následující kroky:

1. nastavení přenosu souborů,
2. nakopírování souboru *ConnEx.apk* do telefonu,
3. výběr souboru *ConnEx.apk* v úložišti telefonu,
4. zablokování instalace,
5. vstup do nastavení,
6. povolení instalace z cizích zdrojů,
7. zobrazení instalačního dialogu,
8. potvrzení instalace.

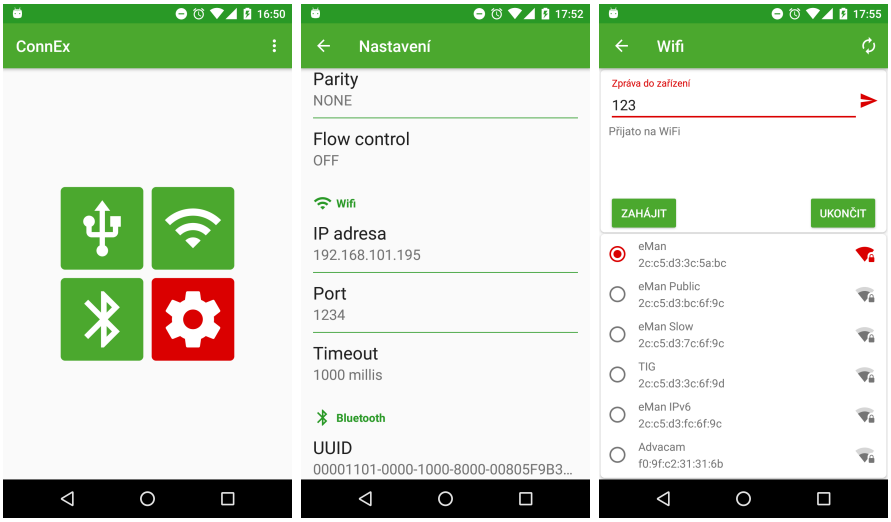
Pokud již byla instalace z cizích zdrojů povolena, body 4–6 nenastanou.





E.3 Hlavní obrazovka, obrazovka nastavení a obrazovka spojení

Na hlavní obrazovce je možné zvolit požadovaný druh komunikace, nebo přejít do nastavení. Na obrazovce nastavení je možné provést konfigurace jednotlivých druhů spojení. Obrazovka spojení slouží k obsluze spojení a zobrazování dostupných zařízení.


16:50
ConnEx
17:52
Nastavení
Parity
NONE
Flow control
OFF
Wifi
IP adresa
192.168.101.195
Port
1234
Timeout
1000 millis
Bluetooth
UUID
00001101-0000-1000-8000-00805F9B3...
17:55
Wifi
Zpráva do zařízení
123
Přijato na WiFi
ZAHÁJIT
UKONČIT
eMan
2c:c5:d3:3c:5a:bc
eMan Public
2c:c5:d3:bc:6f:9c
eMan Slow
2c:c5:d3:7c:6f:9c
TIG
2c:c5:d3:3c:6f:9d
eMan IPv6
2c:c5:d3:fc:6f:9c
Advacam
f0:9f:c2:31:31:6b

Příloha F

Programátorská příručka

F.1 Instalace JDK a nastavení systémových proměnných

Instalace JDK a nastavení systémových proměnných se liší dle operačního systému, proto uvedu pouze body, které je nutné splnit. V případě potřeby lze dohledat návod podle příslušného operačního systém.

1. Instalace JDK verze 8
2. Přidání cesty k JDK do systémové proměnné **PATH**
3. Nastavení cesty k JDK do systémové proměnné **JAVA_HOME**

F.2 Android Studio, Android SDK a SDK tools

Dostupné z `https://developer.android.com/studio/index.html`, kde je potřeba si stáhnout instalační balík dle operačního systému. Po nainstalování Android Studia lze pomocí SDK Manager (integrovaný přímo ve studiu) stáhnout potřebné balíky. Jsou to hlavně tyto balíky:

- Android SDK Platform 25
- Android SDK Build-Tools 25.0.3
- Android Support Repository 47.0.0

V případě potřeby lze stáhnout další balíky. Doporučuji řídit se průvodcem při instalaci Android Studia.

Pokud nepotřebujeme a nechceme, například z důvodu spouštění testů, instalovat Android Studio, lze stáhnout pouze SDK tools a pomocí SDK Manageru, který je součástí SDK tools stáhnout potřebné balíky.

F.3 Import projektu

Pro otevření projektu v Android Studiu je třeba provést následují kroky:

1. *File→Open...*
2. Vyhledání kořenového adresáře projektu
3. Rozbalení kořenového adresáře projektu
4. Vybrat soubor **build.gradle**
5. Potvrdit

Při prvním otevření dojde k provedení build procesu, při kterém dojde ke stažení potřebných knihoven. Pokud nastane nějaká chyba, pak je to nejčastěji z důvodu chybějící verze **buildTools**, nebo že se nepodařilo stáhnout některou knihovnu. V takovém případě doporučuji postupovat dle dostupných návodů na webu.

F.4 Struktura projektu

```
ConnEx ..... kořenový adresář projektu
├── app ..... modul aplikace
│   ├── src
│   │   └── main
│   │       ├── java ..... adresář se zdrojovými Java kódy aplikace
│   │       └── res ..... adresář se zdroji aplikace
│   └── build.gradle ..... build script modulu app
├── coupler ..... modul knihovny
├── build.gradle ..... hlavní build script
└── settings.gradle ..... soubor s použitými moduly
```

F.5 Jmenné konvence

Při psaní zdrojového kódu jsou dodržovány jmenné konvence dle `https://google.github.io/styleguide/javaguide.html`. Pravidla jsou definovány v kořenovém adresáři projektu v souboru **checkstyle_rules.xml**. Pro automatickou kontrolu dodržování jmenných konvencí lze do Android Studia doinstalovat rozšíření (angl. plugin) **CheckStyle-IDEA**.

F.6 Přidání nového druhu komunikace

Pro rozšíření knihovny o nové druhy komunikace je potřeba řídit se následujícími body.

- **Třída `Coupler`**
  Prvním bodem je vytvoření potomka třídy `BaseCoupler`, který zajišťuje obsluhu požadavků, vyhledává a informuje o dostupnosti vzdálených zařízení. Dále je potřeba určit třídu, která definuje vzdálené zařízení.

- **Třída `CouplerService`**
  Dalším bodem je vytvoření potomka třídy `CouplerService`. Třída pak provádí obsluhu samotného spojení. Při vytváření spojení je důležité, vytvoření spojení v novém vlákně, aby nedocházelo k blokování hlavního vlákna. Lze k tomu použít tzv. `Executors`. Ukázku lze vidět už v existujících třídách.

- **Třída `Builder`**
  Pro dodržení společného rozhraní pro použití je potřeba vytvořit potomka třídy `BaseBuilder`, který bude umožňovat vytváření nové instance třídy obsluhující spojení.

- **Třídy `CouplerActivity` a `CouplerFragment`**
  Posledním bodem je vytvoření aktivity a fragmentu pro nový druh komunikace. Opět lze využít rodičovské třídy s již implementovanou funkčnost.

F.7 Rozšíření aplikace

- **Přidání nového druhu komunikace**

1. **Aktualizace knihovny Coupler**
   Aktualizaci provedeme nahrazením modulu s knihovnou novou verzí knihovny.
2. **Vytvoření aktivy a list fragmentu**
   Dalším krokem je vytvoření aktivy a fragmentu zobrazující seznam dostupných zařízení pro nový druh komunikace. S tím souvisí i definovaní potřebných UI komponent a příslušného viewmodelu.
3. **Změny stávající aplikace**
   Následným krokem je přidání ikonky, symbolizující druh komunikace, do hlavního menu. Rozšíření zobrazení správné aktivity po stisku ikonky. V případě potřeby přidat parametry spojení do nastavení. Všechny parametry jsou definovány v enum třídě `SettingsMenu`.
4. **Task**
   Na závěr je potřeba vytvoření třídy pro obsluhu spojení tzv. task. Příklad implementace lze vidět například ve třídě `UsbTask`.

- **Rozšíření/změna podporované funkčnosti**

  1. **Úprava UI a view komponent**
     Rozložení a prvky UI jsou definovány v příslušném XML souboru v balíku *res/layout*. Soubory XML jsou použity v příslušných aktivitách, resp. fragmentech, které v tomto případě plní roli komponenty view. Veškerá obchodní logika by měla být z view delegována na viewmodel.
  2. **Viewmodel**
     Viewmodel je nutno rozšířit o metody pro obsluhu delegovaných obchodních procesů. Předávání data zpět na view je řízeno pomocí Android data binding, nebo lze použít rozhraní `Navigator`, jak lze vidět například ve třídě `MenuViewModel`

- **Definice komunikačního protokolu**
  Pro definování protokolu je potřeba ve třídě `BaseCouplerViewModel` implementovat v metodách `onSendClick()` a `receiverData(String)` navržený protokol.

Příloha **G**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `ConnEx.apk` ..... spustitelná forma implementace
- `src`
  - `ConnEx` ..... kořenový adresář se zdrojovými kódy aplikace
    - `app` ..... modul aplikace
    - `coupler` ..... modul knihovny
  - `arduino`
    - `connectionChecker` ..... Arduino USB, Bluetooth aplikace
      - `connectionChecker.ino` ..... zdrojový kód aplikace
    - `wifiChecker` ..... Arduino WiFi aplikace
      - `wifiChecker.ino` ..... zdrojový kód aplikace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
    - `BP_SMID_FILIP_2017.tex` ..... hlavní zdrojový soubor práce
- `BP_SMID_FILIP_2017.pdf` ..... text práce ve formátu PDF