FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Android aplikace pro sdílení proměnných s platformou Arduino prostřednictvím Bluetooth
Student:	Natanael Jokl
Vedoucí:	Ing. Pavel Kubalík, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce zimního semestru 2022/2023

Pokyny pro vypracování

1) Prozkoumejte existující řešení.
2) Pomocí metod softwarového inženýrství navrhněte vlastní řešení vyhovující níže uvedeným požadavkům.
3) Navržené řešení naprogramujete, zdokumentujte a řádně otestujte.
4) Požadavky:
   - vyberte vhodný programovací jazyk pro OS Android a platformu Arduino,
   - sdílené proměnné mezi oběma zařízeními budou přenášený s pomocí formátu JSON,
   - aplikace pro Android umožní připojení k zařízení Arduino přes Bluetooth,
   - zařízení Arduino bude podporovat tyto periferie: tlačítka, displej, LED dioda, GSM modem,
   - připojení bude provedeno pomocí uživatelského jména a hesla,
   - aplikace bude umožňovat více uživatelů s různým typem oprávnění,
   - aplikace umožní sledovat proměnné na zařízení Arduino,
   - aplikace umožní měnit hodnotu proměnných na zařízení Arduino v závislosti na oprávnění přihlášeného uživatele.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 19. února 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Android aplikace pro sdílení proměnných s platformou Arduino prostřednictvím Bluetooth

Natanael Jokl

Katedra softwarového inženýrství
Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

6. ledna 2022

Poděkování

Na tomto místě bych chtěl poděkovat mému vedoucímu práce, Ing. Pavlu Kubalíkovi, Ph.D., se kterým jsem mohl průběh celé práce konzultovat. Dále bych chtěl poděkovat mojí rodině a přátelům za podporu a modlitby.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, 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 zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). 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 zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. ledna 2022 . . . . . .

Odkaz na tuto práci

Jokl, Natanael. *Android aplikace pro sdílení proměnných s platformou Arduino prostřednictvím Bluetooth* . Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Tato bakalářská práce řeší problematiku sdílení proměnných mezi zařízeními Android a Arduino pomocí technologie Bluetooth. Sdílení proměnných či dat obecně mezi zařízením Android a dalším zařízením je v dnešní době běžně používáno. Můžeme jej naleznout u mnoha chytrých zařízení, která nás obklopují. Tato práce umožňuje širší veřejnosti vytvořit vlastní chytré zařízení, které vznikne upravením tohoto typového řešení. Vytvořená aplikace umožňuje uživateli se přihlásit k zařízení Arduino pomocí uživatelského účtu, následně si v něm zobrazit Arduinem sdílené proměnné a případně je podle uživatelských práv upravit.

Pro tvorbu aplikace byly použity metody softwarového inženýrství – analýza, návrh, implementace a testování. Výsledkem této práce je aplikace pro operační systém Android a API běžící na zařízení Arduino, které umožňuje sdílení a úpravu proměnných. Pro komunikaci mezi zařízeními byl použit datový formát JSON. Dílčím výsledkem práce je popsané zapojení a použití LED diody, tlačítek, displeje a GSM modulu na desce Arduino.

Klíčová slova Android, Arduino, Bluetooth, JSON, sdílené proměnné

Abstract

This bachelor thesis deals with the problematic of sharing variables between Android device and Arduino board via Bluetooth. Sharing either variables or any kind of data with Android device is in these days very common. We can see that in a variety of smart gadgets that surround us. This project facilitates people to make smart gadgets of their own. After logging in through the Android app to the Arduino is user enabled to see and edit the shared variables. The appearance and the editing of the variables are based on user access.

In the process of making the Android app were used methods of software engineering – analysis, design, implementation and testing. The results of this project are Android app and Arduino API that enable user to share variables and edit them. The communication between those devices is realised through JSON data format. Additional result of this project is documented wiring and use of LED diode, buttons, display and GSM modem on the Arduino board.

Keywords Android, Arduino, Bluetooth, JSON, Shared variables

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Úvod

Používání technologie Bluetooth na našich mobilních zařízeních se pro nás v dnešní době stalo samozřejmostí. Využíváme dnes řadu zařízení, která propojujeme se svým mobilním telefonem – např. chytré hodinky, fitness náramek, chytrou domácnost – a tato zařízení nám usnadňují každodenně vykonávané aktivity.

Příchod platformy Arduino nám umožnil, abychom si např. takovou chytrou domácnost poměrně jednoduše vytvořili sami z demo projektů dostupných na internetu. V oblasti Bluetooth existuje řada takových projektů. Problémem je však najít projekt, který více rozvinul myšlenku sdílení jednoduchých dat na sdílení dat složitějších právě přes technologii Bluetooth.

Toto téma jsem jsi proto tedy zvolil, abych nabídl veřejnosti projekt umožňující sdílet strukturovaná data mezi platformami Arduino a Android pomocí technologie Bluetooth.

Cílem této práce je vytvořit aplikaci pro mobilní zařízení s operačním systémem Android, která umožní komunikovat se zařízením Arduino skrze technologii Bluetooth. Pro umožnění komunikace bude zařízení Arduino poskytovat odpovídající API.

Pro přenos zpráv mezi těmito dvěma zařízeními bude použit standard JSON. Aplikace uživateli umožní připojení k zařízení Arduino, a to pomocí přihlášení uživatelským jménem a heslem. Uživatelé budou rozlišeni podle různých typů oprávnění. Nezbytnou součástí aplikace bude zobrazení a případné úpravy proměnných sdílených mezi zmíněnými platformami. Úpravy hodnot jednotlivých proměnných budou umožněny na základě oprávnění uživatele. Zařízení Arduino bude podporovat tyto periferie: tlačítka, displej, LED diodu a GSM modul.

Kapitola 1

Rešerše

Tato kapitola obsahuje základní informace o operačním systému Android, platformě Arduino, technologii Bluetooth a datovém formátu JSON. Dále je zaměřena na použitá vývojová prostředí a také popisuje existující řešení aplikací pro OS Android ovládajících platformu Arduino.

1.1 Android

Operační systém Android je v dnešní době jedním z nejvíce používaných operačních systémů. Je používán na přibližně 70 % mobilních zařízení. [1] Android je založený na Operačním systému Linux, [2] který je veřejnosti již několik desítek let známý.

1.1.1 Historie Android

Počátek operačního systému Android se datuje na rok 2003, kdy byla založena společnost Android Inc. Tuto společnost spolu založili Andy Rubin, Rich Miner, Nick Sears a Chris White. Společnost pak v srpnu 2005 byla odkoupena známou společností Google Inc.

První verze tohoto operačního systému byla velice závislá na hardwarové složce mobilního zařízení HTC Dream. Již tato verze však disponovala velkou sadou funkcí, mezi nimiž bychom nalezli také technologie Wi-Fi a Bluetooth. K dalšímu kroku v souvislosti s technologií Bluetooth došlo ve verzi Android 2.2 Froyo, ve které byla přidána podpora pro Bluetooth handsfee v autech a docích. Další novinka související s Bluetooth se objevila ve verzi 4.1 Jelly Bean. Zde pokrok v technologii Bluetooth souvisel s přenosem dat funkcí Android Beam. Ve verzi Android 4.4 KitKat byla přidána podpora profilu MAP pro Bluetooth. [3] Současnou nejnovější verzí Androidu je Android 12, který je v současné době dostupný pouze na mobilní zařízení Pixel společnosti Google. [4]

1.1.2 Android Studio

Android Studio je oficiálním vývojovým prostředím pro vývoj aplikací na OS Android. Android Studio bylo vytvořeno na základě vývojového prostředí IDEA od společnosti IntelliJ. Nabízí vývojáři spoustu užitečných vylepšení, která zvyšují produktivitu při tvoření aplikace. Mezi těmito vylepšeními můžeme najít například build systém založený na nástroji Gradle, emulátor, jednotné prostředí pro všechna zařízení Android, podporu C++ a NDK. [5]

Projekty v Android Studiu obsahují jeden nebo více modulů, které jsou složeny ze souborů se zdrojovým kódem a soubory se zdroji. Existují různé druhy těchto modulů – knihovnové moduly, moduly aplikace, Google App Engine moduly. Soubory pro sestavení aplikace jsou uloženy pod sekcí Gradle Scripts. Zdrojové soubory jsou typicky napsané v programovacím jazyku Java, či Kotlin. Android Studio také umožňuje podporu VCS, která se hodí při práci na projektu, který je uložen na vzdáleném repozitáři. [5]

Důležitou součástí Android Studia je ADB (Android Device Bridge). Tento nástroj zprostředkovává komunikaci s fyzickým zařízením Andriod. Díky tomu může vývojář testovat vyvíjenou aplikaci přímo na cílovém zařízení. Tato funkce je nezbytná při práci s bezdrátovými technologiemi, jejichž funkčnost nedokáže emulátor napodobit. Dále díky němu můžeme na našem zařízení provést debugging aplikace. To vše se dá realizovat po zapnutí ladění na použitém Android zařízení. [6]

1.2 Arduino

Arduino je elektronická open-source platforma. Je založena na jednoduchém používání hardwaru a softwaru. Desky této platformy umožňují číst z různých vstupů – např. ze světelného senzoru, zmáčknutého tlačítka – a proměnit je na výstupy – např. rozběhnutí motoru, zapnutí LED diody. Desku můžeme ovládat pomocí sady instrukcí, které se posílají mikrokontroleru na desce. K tomu použijeme programovací jazyk Arduino a jejich vývojové prostředí. [7]

1.2.1 Historie Arduino

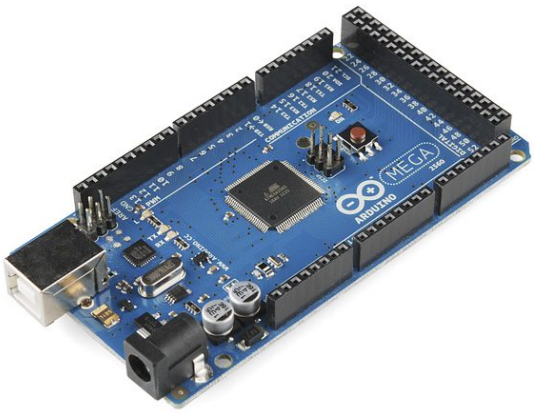
Práce na vývoji prvního Arduina začala v roce 2005, kdy se International Design Institute z italského města Ivrea rozhodl udělat vývojový set, který by byl finančně dostupný pro studenty. V té době byly populární desky BASIC Stamp, které byly ovšem pro většinu studentů příliš drahé. Arduino se mezi studenty brzy uchytilo, a tak se vývojový tým rozhodl desky Arduino nabídnout široké veřejnosti po celém světě. Programová složka Arduina byla navržena na základě programovacího jazyka Pocessing, který se využívá k výuce programování. Od vytvoření první desky Arduino se jich po celém světě prodalo několik stovek tisíc, což zajisté svědčí o jejich popularitě. [8]

1.2.2 Vývojové desky

Arduino má v dnešní době řadu různých desek, které se liší velikostí, pamětí, použitím a dalšími jinými parametry. Pravděpodobně nejznámější deskou je Arduino Uno (viz obrázek 1.1). Dnes se setkáme nejčastěji s třetí verzí této desky. Deska obsahuje napájecí konektor a umožňuje kompatibilitu s mnoha rozšiřovacími deskami (tzv. shieldy). Dále obsahuje 6 analogových vstupních pinů, 14 digitálních vstupně-výstupních pinů a je řízena mikrokontrolerem ATmega328. [9]

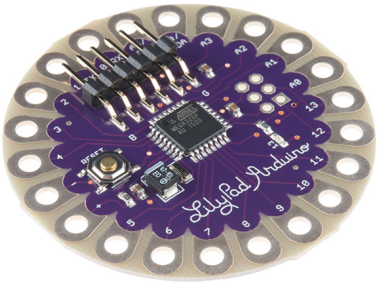
Obrázek 1.1: Vývojová deska Arduino Uno, převzato z [9]

Druhou velmi známou deskou je Arduino Mega (viz obrázek 1.2), která se dnes rovněž nejčastěji prodává v 3. verzi (Rev3). Ta je oproti desce Arduino Uno trochu větší, ale zároveň také výkonnější. Deska je řízena procesorem ATmega2560, který disponuje osmkrát větší pamětí oproti ATmega328. Dále na této desce najdeme 54 digitálních pinů a 16 analogových vstupů. Deska se používá např. v automatizaci, či jiných prostorově náročnějších projektech, které bývají méně přenosné. [9]



Obrázek 1.2: Vývojová deska Arduino Mega, převzato z [9]

Další zajímavou deskou je třeba LilyPad Arduino. Tato deska se používá v e-textilu – kombinace elektroniky a textilu. Za pomocí této desky je možné vyvinout např. chytrou zástěru. Výčet desek Arduino touto deskou rozhodně nekončí. Existuje řada dalších oficiálních desek – Arduino Mini, Arduino Nano, Arduino Micro, Arduino Fio, atd. – a rovněž klonů těchto desek, či jejich modifikací, které mají své specifické uplatnění. [8][9]



Obrázek 1.3: Vývojová deska Arduino LilyPad, převzato z [9]

1.2.3 Arduino Software

Arduino Software je oficiálním vývojovým prostředím pro práci s deskami Arduino. Obsahuje textový editor, okno se zprávami, textovou konzoli a lištu s tlačítky pro nejběžnější akce – verifikaci kódu, nahrání zkompilovaného kódu na desku a další. Vývojáři dále poskytuje snadný přístup ke knihovnám, výběr z velkého množství desek pro připojení a sériový monitor. Díky sériovému monitoru může vývojář pomocí USB komunikovat s deskou Adruino formou textových zpráv. [10]

1.2.4 CLion a PlatformIO

Vývojové prostředí CLion od společnosti IntelliJ se používá pro vývoj aplikací v programovacích jazycích C a C++. Nabízí řadu chytrých funkcí pro efektivnější psaní kódu. Upozorňuje vývojáře na překlepy, dále mu umožňuje generovat jednoduché celky kódu (konstruktor, getter, setter, atd.), jednoduchý refactoring, rychlou, snadnou dokumentaci kódu, analýzu kódu za běhu. Poskytuje také zabudovaný debugger a umožňuje instalaci různých pluginů. Jedním z těchto pluginů je PlatformIO. Díky němuž může vývojář kompilovat a nahrávat kód například na desky Arduina. [11]

PlatformIO samo o sobě je profesionální nástroj pro vývoj embeded systémů. Funguje na mnoha platformách. Umožňuje debugging, unit testing, automatizovanou analýzu kódu a remote management. Kombinace těchto dvou

nástrojů může vývojáři značně ulehčit práci s programováním desky Arduino. [12]

1.3 Bluetooth

Bluetooth je technologií, která se používá pro bezdrátové propojení zařízení. Dnes ji běžně můžeme najít např. na mobilním telefonu, počítači a chytrých zařízeních. Byl proveden odhad, že za rok 2018 se celosvětově prodalo přibližně 10 miliard zařízení s touto technologií. [13]

1.3.1 Historie Bluetooth

Tento standard vznikl v roce 1988 jako snaha sjednotit v té době velmi nejednotné množství standardů pro bezdrátovou komunikaci vyvinutých různými výrobci. Mezi první členy sdružení Bluetooth Special Interest Group patřili společnosti Ericsson, IBM, Intel a Toshiba. Počet členů velmi rapidně rostl a již po jednom roce bylo ve společnosti okolo čtyř tisíc členů. Za formální značku pak byla technologie registrována roku 1998 a to pod názvem Bluetooth . [13]

Verze Bluetooth 1.X přinesly přenos zvuku, který se používal pro hovory. Nebyl však dostatečně kvalitní pro přenos zvuku za účelem poslechu hudby. Následující verze Bluetooth 2.X přinesly rychlejší přenos dat, bezpečné párování (SSP) a zvýšení maximálního dosahu na cca 30 m. Za důležitou zmínku stojí pak verze Bluetooth 4.X, které přinesly rozdělení Bluetooth na Bluetooth Classic (viz 1.3.3), Bluetooth High Speed a Bluetooth Low Energy (viz 1.3.4). Dosah signálu ve volném prostoru se zvýšil na 50 m a v interiéru na zhruba 10 m. Aktuální verze Bluetooth 5 tato čísla přibližně zčtyřnásobila. K tomu přidala možnost mít v zařízení více připojených zařízení. [13]

1.3.2 Základy funkčnosti

Technologie Bluetooth obecně operuje v frekvenčním pásmu 2,4 GHz, které je také využíváno například technologií Wi-Fi. Tím, že je toto pásmo veřejné, tak často dochází k rušení jinými vysílači. Proto Bluetooth využívá technologii FHSS (Frequency Hopping Spread Spectrum). Ta umožňuje toto rušení minimalizovat. [14]

Dalším možným omezujícím faktorem kvality signálu je vzdálenost. Ta je závislá na překážkách, které mohou dosah signálu výrazně snížit, a dále také na výkonnostní třídě. Výkonnostní třídy byly vytvořeny z důvodu úspory energie, která je u malých přenosných zařízení poměrně dost důležitá. Přehled výkonnostních tříd nalezneme v tabulce 1.1. [14]

Technologie Bluetooth podporuje více typů komunikace. Mezi nimi bychom našli Point-to-Point (obsahující piconet), Broadcast a Mesh. V případě komunikace Point-to-Point rozlišujeme dva případy. Prvním případem je klasické

Výkonnostní třída	Výkon	Dosah
Class 1	10 mW	až cca 100 m
Class 2	2,5 mW	až cca 10 m
Class 3	1 mW	cca 1 m

Tabulka 1.1: Výkonnostní třídy Bluetooth, upraveno podle [14]

spojení pouze dvou zařízení. V druhém případě máme propojených více zařízení, kdy jedno je označeno za hlavní – role master – a ostatní za vedlejší – role slave. Takto uspořádané spojení definujeme jako *piconet*. Tento *piconet* může obsahovat až 255 zařízení, která jsou možná rozlišit pomocí jejich unikátní adresy Bluetooth Device Address. Piconety pak můžeme skládat do větších síťových celků zvané *scatternet*. [14][15]

Zabezpečení je u Bluetooth řešeno metodou párování. Typicky zařízení vyžadující spojení odešle na druhé zařízení žádost o párování. Při párování se potvrzuje totožnost a zadává se společný tajný klíč. Potvrzením zadaného klíče vniká mezi zařízeními šifrované spojení, které je šifrováno 48 bitovým klíčem. [14]

1.3.3 Bluetooth Classic

Technologie Bluetooth Classic, často označovaná také jako *„Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR)“*, je nízkoenergetický vysílač, který umožňuje přenášet data přes 79 kanálů ve frekvenčním pásmu 2,4 GHz, které je označováno jako nelicencované průmyslové, vědecké a lékařské pásmo (ISM). Bluetooth Classic podporuje komunikaci zařízení typu Point-to-Point, které se v dnešní době hlavně používá pro bezdrátové streamování zvuku. To můžeme pozorovat například u bezdrátových sluchátek, reproduktorů nebo bluetooth rádií zabudovaných v autech. Dále se používá pro přenos dat, jehož ukázkou je například mobilní tisk. [16]

1.3.4 Bluetooth Low Energy

Bluetooth LE (Low Energy) je technologie navržená pro provoz s nízkou spotřebou energie. Vysílač Bluetooth LE přenáší data přes 40 kanálů v nelicencovaném frekvenčním pásmu ISM 2,4 GHz. To poskytuje vývojářům velké množství flexibility při vytváření produktů, které splňují trhem poptávané požadavky připojitelnosti. Bluetooth LE podporuje více komunikačních topologií. Podporuje komunikační typ Point-to-Point, dále pak Broadcast, nebo také Mesh, který umožňuje vytvoření rozsáhlých spolehlivých sítí zařízení. Dále je dnes také široce používán pro přesné určování polohy uvnitř budov. Nejprve podporoval pouze přibližné určení přítomnosti a blízkosti, nyní podporuje *„Bluetooth Direction Finding“* a brzy začne s podporou velmi přesného měření vzdálenosti. [16]

1.4 JSON

JSON (JavaScript Object Notation) je formát pro snadnou výměnu dat. Je pro člověka dobře čitelný a zároveň i snadno zapisovatelný. Pro počítač je zase snadno generovatelný či parsovatelný. Je založen na standardu programovacího jazyka JavaScript ECMA-262 třetího vydání z prosince roku 1999. Přestože je tento textový formát jazykově nezávislý, je pro programátory používající jazyky z rodiny jazyka C velmi intuitivní na používání. Tyto vlastnosti dělají z JSON ideální jazyk pro výměnu dat. [17]

1.4.1 Základy formátu JSON

JSON se skládá ze dvou typů struktur. Prvním typem je kolekce jména proměnné a její hodnoty. Množina těchto kolekcí se nazývá objekt. Druhým typem je seřazený seznam proměnných/hodnot/objektů. Tu nazýváme pole. Obecně je důležité poznamenat, že název proměnné je vždy řetězec znaků, ale hodnota může nabývat různých hodnot – čísla (celá, desetinná), řetězce symbolů (string), pravdivostní hodnoty (true / false), prázdné hodnoty (null), pole, nebo objektu (viz zdrojový kód 1). [17]

```
{
    "users": [
        {
            "name": "Josef Novak",
            "username": "pepa01"
        },
        {
            "name": "Jana Dvorakova",
            "username": "janadvor"
        },
    ]
}
```

Zdrojový kód 1: Ukázka datového formátu JSON

1.4.2 Knihovny

Pro vývoj aplikací pro OS Android máme datový formát podporovaný již od API levelu 1. Tato zabudovaná knihovna nám umožňuje práci v datovém formátu JSON v jeho plném rozsahu. Pro platformu Arduino existuje podobná externí knihovna ArduinoJson. Tato knihovna umožňuje kromě tvorby JSON objektů také serializaci a deserializaci těchto objektů. Má velice dobrou podporu a snadnou instalaci. Vývojář si stáhne hlavičkový soubor a ten přiloží do svého projektu. Tím je instalace hotová. Tyto dvě knihovny jsou nejpo-

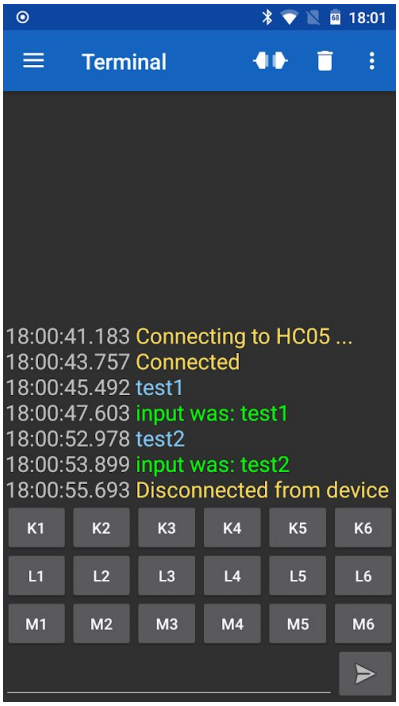
užívanějšími knihovnami pro práci s datovým formátem JSON na daných platformách. [18][19]

1.5 Existující řešení

Tato kapitola se zabývá existujícími řešeními problematiky sdílení dat (resp. proměnných) pomocí technologie Bluetooth. Řešení se nacházejí jak mezi mobilními aplikacemi, tak mezi odbornými pracemi. Následující aplikace pro mobilní zařízení s OS Android jsou dostupné na Obchod Play a zprostředkovávají komunikaci se zařízením Arduino pomocí technologie Bluetooth.

1.5.1 Serial Bluetooth Terminal

Aplikace Serial Bluetooth Terminal [20] umožňuje rozpoznat nejbližší zařízení Bluetooth, spárovat se s nimi a následně se k nim připojit. Okolní zařízení rozlišuje na ty, která mají klasický Bluetooth a ty, která obsahují Bluetooth LE (Low Energy). Aplikace dále umožňuje posílat si mezi zařízeními zprávy.



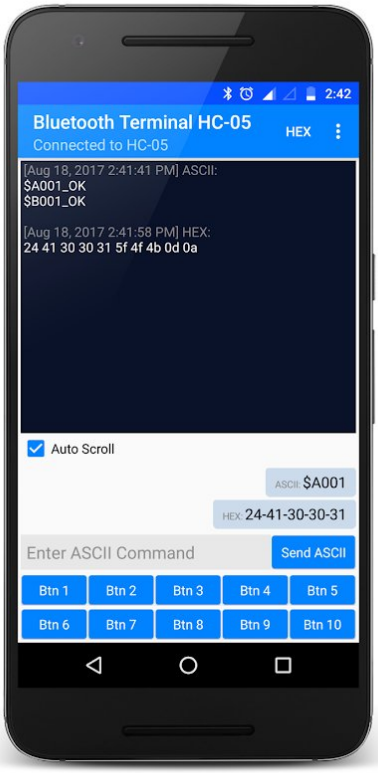
Obrázek 1.4: Ukázka aplikace Serial Bluetooth Terminal, převzato z [20]

Z hlediska uživatelského rozhraní je aplikace rozdělena na 4 základní aktivity. Úvodní aktivita obsahuje prostředky pro odesílání zpráv. Druhá aktivita obsahuje dostupná zařízení k připojení. Další aktivita obsahuje nastavení

aplikace a poslední aktivita informace o aplikaci. Pro přepínání mezi jednotlivými aktivitami je v aplikaci boční rozkládací menu. Díky němu je aplikace přehledná a snadná k používání.

1.5.2 Bluetooth Terminal HC-05

Aplikace Bluetooth Terminal HC-05 [21] je především pro připojování k bluetooth modemu HC-05. Umožňuje s ním následně posílání zpráv.



Obrázek 1.5: Ukázka aplikace Bluetooth Terminal HC-05, převzato z [21]

Aplikace se v základu skládá ze dvou na sebe navazujících aktivit. První aktivita umožňuje zobrazit dostupná zařízení pro komunikaci pomocí Bluetooth. Při zvolení zařízení se aplikace pokusí připojit k danému zařízení a přesměruje uživatele na druhou aktivitu. V druhé aktivitě jsou nástroje pro posílání textových zpráv. Tato návaznost aktivit se jeví jako logická a dobré přímočaré řešení pro používání aplikace uživatelem.

1.5.3 Odborné práce

Na téma sdílení dat/proměnných pomocí technologie Bluetooth již bylo napsáno několik odborných prací. Za zmínku stojí například bakalářská práce s názvem *„Platforma pre komunikáciu medzi Android a Arduino“* [22], nebo práce *„Meteorologická stanice v chytré domácnosti založená na platformě Arduino“* [23].

První zmíněná práce se zabývá technologií Bluetooth pro komunikaci na více teoretické úrovni, ale obsahuje také praktickou ukázku pomocí aplikace pro OS Android. Práce druhá se naopak zaměřuje na velmi praktickou aplikaci technologie Bluetooth pro vytvoření meteorologické stanice.

Další práce zabývající se komunikací mezi zařízeními Arduino a Android pomocí Bluetooth je *„Knihovna funkcí pro OS Android umožňující řízení vývojového kitu Arduino“* [24]. Tato práce se zabývá různými typy komunikace - USB, Bluetooth a Wi-Fi. Práce má větší záběr komunikačních technologií, a proto není tolik zacílena na Bluetooth.

Práce, kterou je určitě důležité zmínit, je bakalářská práce s názvem *„Aplikace android pro komunikaci s platformou Arduino protokolem Bluetooth“* [25]. Tato bakalářská práce se zabývá sdílením proměnných mezi zařízeními Android a Arduino. Pro komunikaci používá datový formát JSON. Práce se dále zabývá především sdílením proměnných primitivních datových typů a jejich polí. Je zde rovněž přihlašování k uživatelskému účtu pomocí jména a hesla. Aplikace pro Android, která je výsledkem této práce uživateli zobrazuje tyto proměnné, umožňuje proměnné upravovat a měnit pohledy (seznamy proměnných) stavu procesoru Arduina. Tato práce je určitým nezávislým řešením podobné problematiky. Rozdílnost lze nalézt ve větší rozmanitosti uživatelských přístupů k proměnným, dále pak v možnosti sdílení jednovrstvých struktur (záznamů).

Kapitola 2

Analýza

V této kapitole je provedena analýza řešení problematiky, která se skládá z funkčních a nefunkčních požadavků, z případů užití a použitých technologií. V závěru kapitoly jsou analyzovány datové typy přenášených dat.

2.1 Funkční požadavky

Funkční požadavky udávají, které funkčnosti bude výsledná aplikace obsahovat. Android aplikace tedy bude splňovat následující:

- **F1 – Vyhledání zařízení**
  Android aplikace vyhledá spárovaná a v okolí dostupná zařízení pro připojení přes Bluetooth.
- **F2 – Zobrazení zařízení**
  Android aplikace zobrazí spárovaná a v okolí dostupná zařízení pro připojení přes Bluetooth.
- **F3 – Připojení k zařízení**
  Aplikace uživateli umožní se pomocí Bluetooth spojit se zařízením Arduino.
- **F4 – Zobrazení neúspěchu připojení**
  Aplikace bude uživatele informovat o neúspěšném pokusu o připojení se k zařízení.
- **F5 – Přihlášení uživatele**
  Aplikace uživateli umožní přihlásit se pomocí uživatelského jména a hesla a také se odhlásit.
- **F6 – Různí uživatelé**
  Aplikace bude podporovat možnost více uživatelů, kteří budou mít různé typy oprávnění přístupu k sdíleným proměnným.

- **F7 – Sdílení proměnných**
  Platforma Arduino bude zprostředkovávat sdílení proměnných s platformou Android.
- **F8 – Zobrazení proměnných**
  Aplikace bude zobrazovat stav sdílených proměnných podle oprávnění daného uživatele.
- **F9 – Úprava proměnných**
  Aplikace uživateli umožní upravovat sdílené proměnné na základě jeho oprávnění.

2.2 Nefunkční požadavky

Nefunkční požadavky jsou další částí specifikující limity a omezení výsledného produktu. Pro aplikaci bude platit následující:

- **N1 – Vzájemné připojení**
  Spojení aplikace a zařízení Arduino bude realizováno pomocí technologie Bluetooth.
- **N2 – Přihlášení uživatele**
  Pro práci se sdílenými proměnnými se bude uživatel muset přihlásit pomocí uživatelského jména a hesla.
- **N3 – Různí uživatelé**
  Aplikace bude podporovat alespoň dva uživatele a alespoň dva různé typy přístupu k sdíleným proměnným.
- **N4 – Formát komunikace**
  Komunikace mezi oběma zařízeními bude implementována pomocí datového formátu JSON.

2.3 Případy užití

Případy užití specifikují funkcionality aplikace z pohledu aktérů - uživatele a aplikace. Níže uvedený seznam popisuje případy užití z diagramu 2.1.

- **UC1 – Připojit se k zařízení**
  Umožňuje uživateli se připojit k zařízení, se kterým je mobilní zařízení spárované, či je v okolí dostupné. Tento případ užití začíná poté, co uživatel otevře aplikaci.

  Hlavní scénář:

  1. Uživateli se zobrazí seznam spárovaných zařízení (viz UC2).

2. Uživatel si ze seznamu spárovaných zařízení vybere zařízení, ke kterému se chce připojit.
3. Při úspěšném připojení je uživatel přesměrován na přihlašování. Při neúspěchu je informován o neúspěšném pokusu o připojení.

Alternativní scénář:

1. Uživatel klikne na tlačítko, které zahájí prohledávání okolí pro dostupná zařízení.
2. Uživateli se zobrazí seznam dostupných zařízení (viz UC2).
3. Dále pokračuje bodem 2 z hlavního scénáře.

- **UC2 – Zobrazit dostupná zařízení**
  Aplikace může uživateli zobrazit spárovaná resp. v okolí dostupná zařízení.

  Hlavní scénář:
  Tento scénář začíná poté, co uživatel otevře aplikaci.

  1. Aplikace vyhledá spárovaná zařízení.
  2. Aplikace zařízení zobrazí.

  Alternativní scénář:
  Tento scénář začíná po případu užití UC3.

  1. Aplikace prohledá okolí pro dostupná zařízení.
  2. Při zachycení zařízení v okolí aplikace toto zařízení přidá do seznamu dostupných zařízení.
  3. Aplikace zobrazí seznam dostupných zařízení.

- **UC3 – Vyhledat dostupná zařízení**
  Umožní uživateli zahájit hledání zařízení, která jsou v okolí dostupná. Tento případ užití začíná poté, co uživatel otevře aplikaci a přejde do záložky pro vyhledávání dostupných zařízení.

  Hlavní scénář:

  1. Uživatel klikne na tlačítko pro vyhledání bluetooth zařízení dostupných v okolí zařízení.
  2. Aplikace začne vyhledávat tato zařízení.

- **UC4 – Přihlásit se**
  Umožní uživateli se pomocí uživatelského jména a hesla přihlásit do svého účtu.

Hlavní scénář:
Tento scénář začíná poté, co aplikace připojí k zařízení Arduino.

1. Uživatel do textových polí zadá své uživatelské jméno a heslo.
2. Klikne na tlačítko pro přihlášení.
3. V zařízení Arduino dojde k ověření identity.
4. V případě validních údajů je uživatel přihlášen.

Alternativní scénář:
Tento scénář je pokračováním bodu 3 z hlavního scénáře. Scénář se provede v případě, že ověřování identity odhalí, že kombinace přihlašovacích údajů není validní.

1. Uživatel je aplikací informován o nesprávnosti kombinace uživatelského jména a hesla.

- **UC5 – Zobrazit sdílené proměnné**
Umožní aplikaci zobrazení sdílených proměnných. Tento případ užití začíná poté, co uživatel otevře aplikaci a přejde do záložky pro vyhledávání dostupných zařízení.

  Hlavní scénář:

  1. Aplikace odešle na zařízení Arduino žádost o všechny proměnné.
  2. Aplikace přijme seznam sdílených proměnných.
  3. Aplikace zobrazí sdílené proměnné uživateli.

- **UC6 – Upravit sdílenou proměnnou**
Umožní uživateli upravit zvolenou sdílenou proměnnou.

  Hlavní scénář:
  Tento scénář začíná poté, co se uživateli zobrazí seznam sdílených proměnných.

  1. Uživatel v seznamu zvolí proměnnou, kterou chce upravit a klikne na tlačítko pro její úpravu.
  2. Aplikace uživateli poskytne prostředí pro úpravu proměnné – textové pole pro novou hodnotu proměnné, tlačítko nastavení nové hodnoty a tlačítko pro zrušení úpravy.
  3. Uživatel zadá novou hodnotu proměnné do textového pole.
  4. Uživatel klikne na tlačítko pro nastavení nové hodnoty proměnné.
  5. Na zařízení Arduino je odeslána žádost o úpravu s novou hodnotou.

6. Uživatel je informován o úspěšném či neúspěšném nastavení nové hodnoty proměnné.

Alternativní scénář:
Scénář začíná v okamžiku, kdy uživatel klikne na tlačítko pro úpravu sdílené proměnné (viz bod 1 hlavního scénáře).

1. Uživatel klikne na tlačítko pro zrušení úpravy.
2. Aplikace uživateli skryje prostředí pro úpravu vybrané sdílené proměnné.

- **UC7 – Odhlásit se**
  Umožní uživateli odhlásit se z uživatelského účtu (a případně se přihlásit k jinému). Tento případ užití začíná poté, co se uživatel úspěšně přihlásí.

  Hlavní scénář:

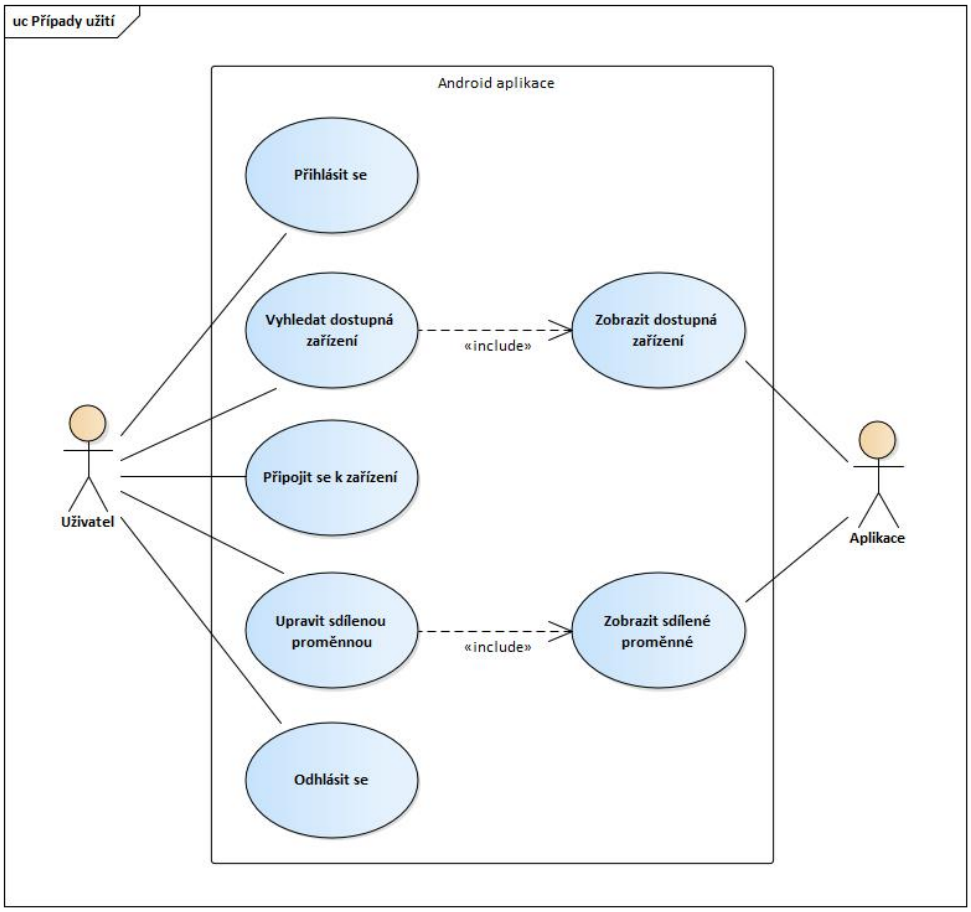
  1. Uživatel klikne na tlačítko pro odhlášení.
  2. Uživatel je přesměrován na přihlašovací obrazovku.

2.3.1 Kontrola

Na závěr modelování případů užití provedeme kontrolu. Ta nám ukáže, zda máme všechny definované funkční požadavky pokryté jednotlivými případy užití. Jedna metoda, jak provést kontrolu, je pomocí tabulky jejíž sloupce jsou případy užití a řádky funkční požadavky. Políčko příslušící požadavku označíme za pokryté, pokud je splněné daným případem užití (viz tabulka 2.1).

	UC1	UC2	UC3	UC4	UC5	UC6	UC7
F1		✓	✓				
F2		✓					
F3	✓						
F4	✓						
F5				✓			✓
F6				✓			✓
F7					✓	✓	
F8					✓	✓	
F9						✓	

Tabulka 2.1: Kontrola splnění funkčních požadavků



Obrázek 2.1: Diagram případů užití

2.4 Technologie

Pro vývoj Android aplikace a Arduino API budou použity následující technologie:

- **Bluetooth**
  Technologie Bluetooth bude použita pro zprostředkování komunikace mezi zařízeními.
- **Bluetooth modul**
  Pomocí Bluetooth modulu hc-05 bude mít zařízení Arduino umožněno používat technologii Bluetooth.
- **Arduino**
  Deska Arduino Mega 2560 bude použita pro vývoj API.

- **Android**
  Aplikace bude fungovat na zařízení Android 5.0 a vyšší.
- **JSON**
  Datový formát bude použit pro komunikaci mezi zařízeními a pro přenos sdílených dat.
- **GSM modul**
  Pro demonstraci podpory GSM modulu na desce Arduino bude použit GPRS GSM modul SIM800L. [26]
- **LED dioda**
  Pro demonstraci podpory LED diody na desce Arduino bude použit RGB LED modul KY-016. [27]
- **OLED displej**
  Pro demonstraci podpory displeje na desce Arduino bude použit I2C OLED displej 128x64. [28]
- **Tlačítka**
  Pro demonstraci podpory tlačítek na desce Arduino bude použita membránová klávesnice se čtyřmi tlačítky. [29]

2.5 Datové typy

Jak již bylo zmíněno v podkapitole 1.4.1, tak hodnota JSON objektu může nabývat různých hodnot. Pro přenos dat mezi zařízeními Arduino a Android proto bude nutné zvolit vhodné primitivní datové typy pro zaznamenání a přenos hodnot. Datové typy se odvíjí od používaného programovacího jazyka a v některých případech se implementace datových typů u programovacích jazyků pro Android (Java) a Arduina (C++) liší.

Jako relevantní hodnoty JSON objektu na Arduino nalezneme primitivní datové typy *bool*, *int*, *long*, *float*, *double*, *char* a *String*. Datové typy *float* a *double* však na desce Arduino Mega 2560 mají stejný rozsah. Proto budeme operovat pouze s datovým typem *float*. [30]

Relevantní datové typy musíme nalézt i pro jazyk Java. Tam bychom nalezli *Boolean*, *Integer*, *Long*, *Float*, *Double*, *Character*, *String*. Žel však nemůžeme provést jednoduché namapování datových typů z jazyka Arduino do jazyka Java, protože některé datové typy jsou jinak implementované. Párování bude fungovat pro typy *bool* a *Boolean*, *float* a *Float*, *char* a *Character*, *String* a *String* a v poslední řadě *long* a *Integer*. Tyto typy budou sobě odpovídat velikostí i rozsahem. Pro zachování datového typu *int* se jej můžeme rozhodnout namapovat s datovým typem *Integer*, kde nám nemusí vadit lehké plýtvání pamětí na straně Android aplikace. [30][31]

Kapitola 3

Návrh

Tato kapitola popisuje návrh řešení bakalářské práce. Řešení se bude skládat za dvou hlavních komponent – Android aplikace a API poskytovaného zařízením Arduino. Práce bude pracovat na principu klienta a serveru. Aplikace bude představovat klienta, který se bude dotazovat na server. Zařízení Arduino zase bude představovat server, který bude poskytovat rozhraní pro dotazování klienta. Na serveru budou uložena data. Tento vztah by se dal rovněž popsat Bluetooth terminologií jako master (server) a slave (klient).

3.1 Aplikace

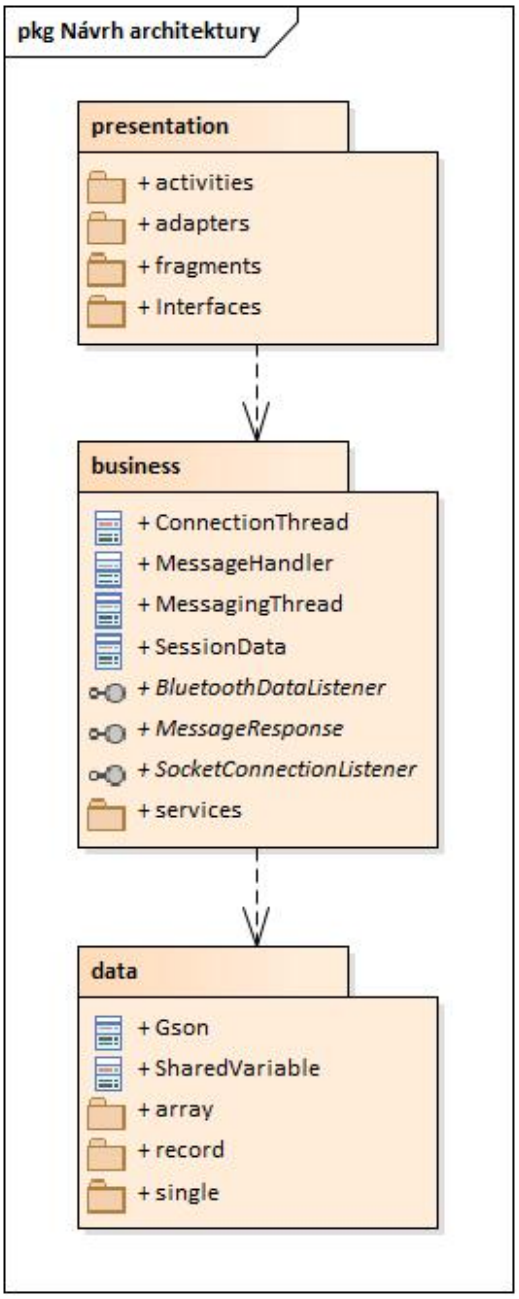
Návrh aplikace se skládá ze dvou hlavních částí – návrhu architektury a návrhu uživatelského prostředí. Architektura popisuje základní návrh fungování aplikace. Návrh uživatelského prostředí zase popisuje, jak bude aplikace graficky navržena pro snadné používání. Pro vytvoření diagramů této podkapitoly byl požit nástroj Enterprise Architect [32].

3.1.1 Architektura

Pro vytvoření Android aplikace bude použita striktní třívrstvá architektura. Ta se skládá z prezentační, business a datové vrstvy. Prezentační vrstva je závislá na business vrstvě, business vrstva je zase závislá na datové (viz obrázek 3.1).

Prezentační vrstva bude mít na starosti zobrazování dat uživateli, která získá od business vrstvy, a reagování na uživatelské vstupy. Business vrstva bude mít na starosti logiku celé aplikace. Tu bude převážně tvořit komunikace se zařízením Arduino pomocí jeho API. Pro přeposílání dat ve správném formátu bude business vrstva využívat datovou vrstvu. Komunikace datové vrstvy se bude skládat ze zpráv v datovém formátu JSON. Tyto zprávy bude business vrstva posílat a přijímat. Na přijaté zprávy bude automaticky reagovat. Datová vrstva bude poskytovat nástroje pro serializaci a deserializaci

zpráv a dat v nich uložených. Kromě toho bude v datové vrstvě uložena struktura dat – sdílených proměnných.

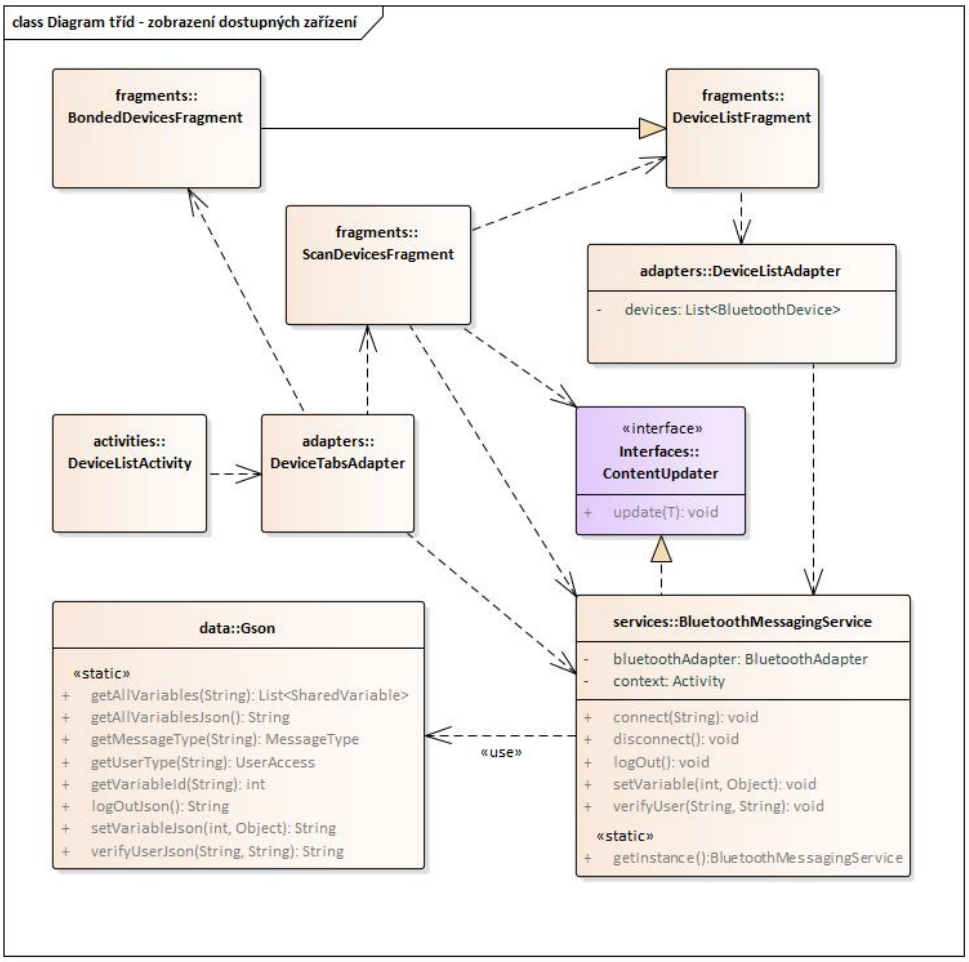


Obrázek 3.1: Návrh architektury aplikace

3.1.2 Návrhové modely tříd

Návrhové modely tříd slouží ke zdokumentování provedených architektonických rozhodnutí. Dále pomáhají vizuálně zachytit jednotlivé vazby mezi třídami, jejich atributy, metody, atd. a pochopit budoucí fungování aplikace.

První model tříd 3.2 zachycuje návrh tříd pro zobrazení dostupných zařízení. Středem tohoto diagramu je aktivita *DeviceListActivity*, která se bude skládat ze dvou fragmentů - *ScanDeviceFragment* a *BondedDeviceFragment*. Každý z těchto fragmentů bude umístěn do jedné záložky, které budou řízeny třídou *DeviceTabsAdapter*. Oba tyto fragmenty využívají třídu *DeviceListFragment*, která jim pomáhá se zobrazováním seznamu zařízení.



Obrázek 3.2: Model tříd – zobrazení dostupných zařízení

Pro zobrazení spárovaných zařízení si třída *DeviceTabsAdapter* získá od

BluetoothMessagingService seznam spárovaných zařízení, předá je fragmentu *BondedDeviceFragment* a ten je zobrazí.

Zobrazení v okolí dostupných zařízení bude rovněž provedeno dotazováním se na *BluetoothMessagingService*. V případě, že dojde k objevení nového zařízení v okolí, bude *ScanDeveiceFragment* pomocí listeneru *ContentUpdater* informován o novém zařízení a seznam aktualizuje.

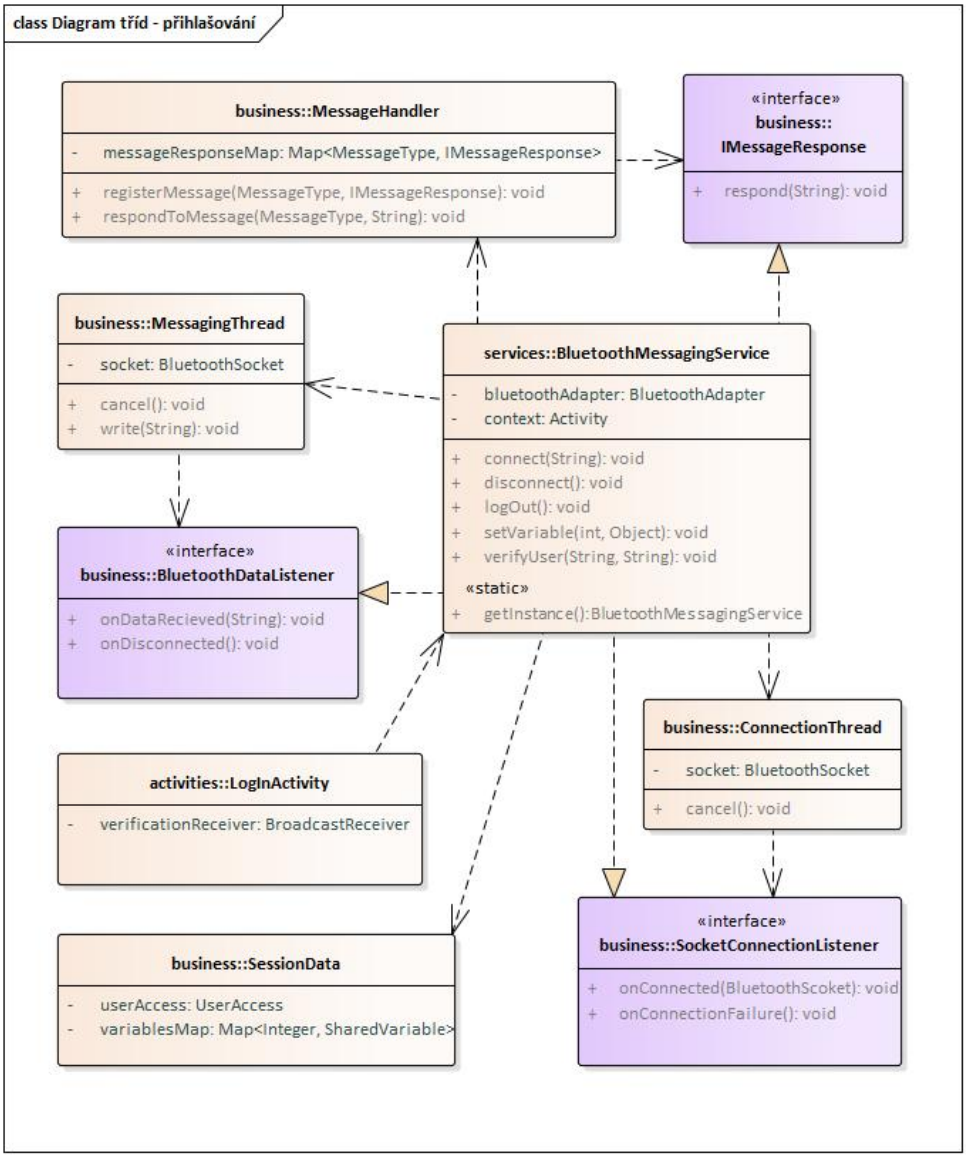
Při stisknutí tlačítka pro připojení v jedné ze dvou záložek dojde k použití metody *connect(String deviceAddress)* z *BluetoothMessagingService* a bude proveden pokus o připojení k zařízení dané adresy.

Další diagram (viz obrázek 3.3) zachycuje návrh tříd pro realizaci přihlašování uživatele. Je zde rovněž zachycen způsob, jakým bude aplikace komunikovat se serverem.

O připojení k serveru se postará třída *ConnectionThread*, která pomocí rozhraní *SocketConnectionListener* definovaného v třídě *BluetoothMessagingService* informuje o neúspěšném připojení metodou *onConnectionFailure()*, či o úspěšném připojení metodou *onConnected(BluetoothSocket socket)*. Při úspěšném připojení bude hodnota parametru*socket* předána třídě *MessagingThread*. Tato třída bude přijímat data a informovat o nich třídu *BluetoothMessagingService* pomocí metody *onDataReceived(String data)* z rozhraní *BluetoothDataListener*. Dále tuto třídu bude informovat v případě přerušení spojení pomocí metody *onDisconnected()*.

Pro automatické zpracování dat bude sloužit třída *MessageHandler*. V ní budou pomocí metody *registerMessage(MessageType type, IMessageResponse response)* registrovány odpovědi na jednotlivé typy zpráv. Na konkrétní zprávy se pak bude odpovídat pomocí metody *respondToMessage(MessageType type, String jsonMessage)*. Tato metoda bude moct být volána hned po přijetí dat/zprávy. K registraci jednotlivých odpovědí na různé typy zpráv dojde ve třídě *BluetoothMessagingService*. Ta data ze zprávy nejprve zpracuje pomocí metod třídy *Gson*, pak je uloží do instance třídy *SessionData* a informuje o nich prezentační vrstvu pomocí broadcast zprávy.

Příkladem tohoto mechanismu je přihlašování. Uživatel z aktivity *LogInActivity* odešle pomocí metody *verifyUser(String username, String password)* z třídy *BluetoothMessagingService* na server žádost o verifikaci onoho uživatele. Zpráva bude pomocí metody *verifyUserJson(String username, String password)* převedena do datového formátu JSON. Server uživatele prověří a odpoví. *MessagingTherad* odpověď serveru zachytí, *BluetoothMessagingService* zprávu zpracuje (pomocí metod z třídy *Gson*) a pošle broadcast zprávu o úspěšném, či neúspěšném přihlášení. *LogInActivity* pomocí instance třídy *BroadcastReciever* toto zachytí a zobrazí příslušný obsah uživateli.

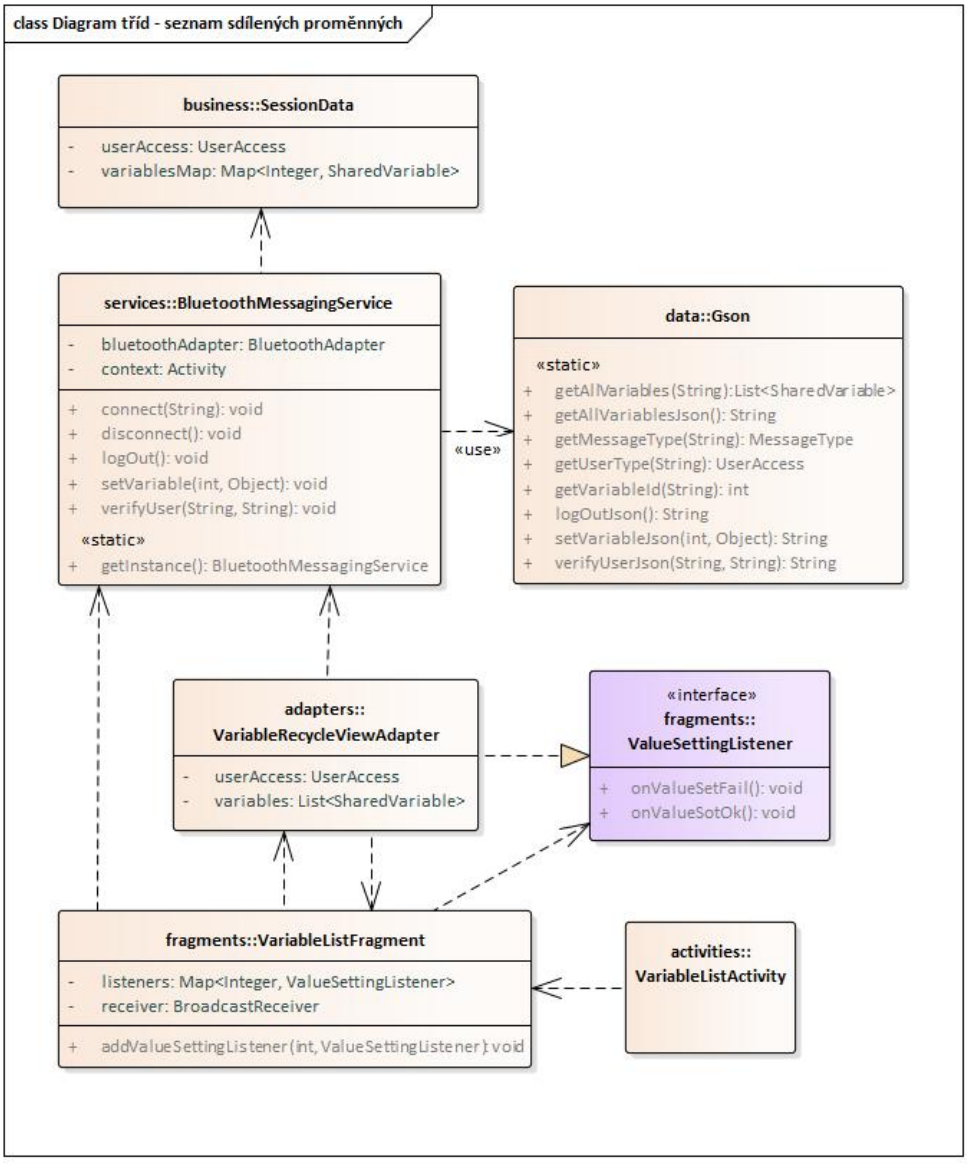


Obrázek 3.3: Model tříd – přihlašování

Návrh pro realizaci zobrazení sdílených proměnných je ilustrován pomocí diagramu 3.4. V aktivitě *VariableListActivity* bude umístěn fragment *VariableListFragment*. Jeho obsah bude zobrazován pomocí adaptéru *VariableRecycleViewAdapter*. Ten bude řídit, které proměnné se uživateli zobrazí na základě jeho uživatelského oprávnění. Proměnné budou zobrazené jako seznam položek v *RecycleView*.

Pro úpravu těchto položek bude sloužit metoda *setVariable(int id, Ob-*

ject value), která pošle na server žádost o změnu dané proměnné. Server pak informuje o úspěchu, či neúspěchu nastavení proměnné třídu *BluetoothMessagingService*. Ta pak pomocí braodcast zprávy informuje *VariableListFragment*, který následně s použitím instance třídy *ValueSettingListener* informuje *VariableRecycleViewAdapter* o úspěchu (či neúspěchu) při nastavování dané proměnné.

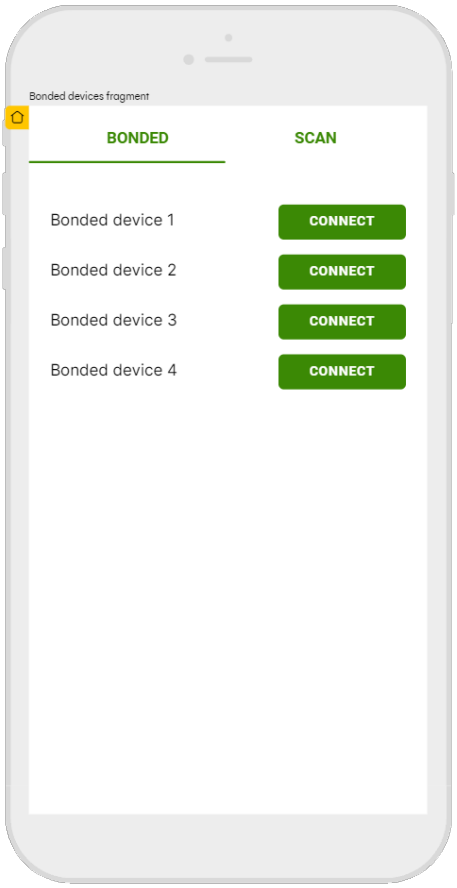


Obrázek 3.4: Model tříd – zobrazení sdílených proměnných

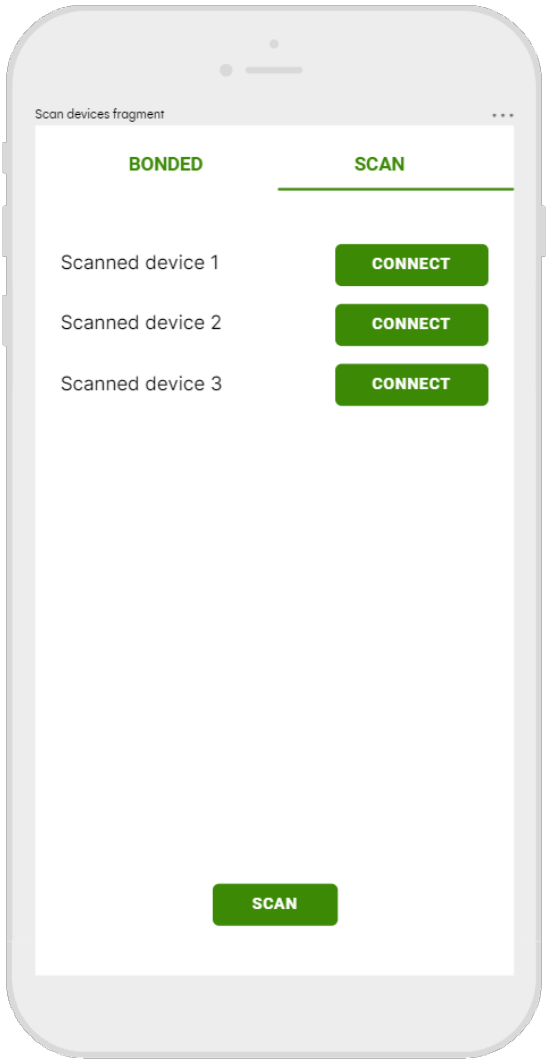
3.1.3 Uživatelské rozhraní

Uživatelské rozhraní aplikace se bude skládat z různých aktivit, které budou zobrazovány na obrazovku. Tyto aktivity se typicky skládají z jednoho či více fragmentů, které pak obsahují jednotlivé zobrazované prvky – např. tlačítka, textová pole, atd. Pro tvorbu rozložení uživatelského prostředí byl použit nástroj *Uizard.io* [33].

První aktivita, která se uživateli zobrazí při otevření aplikace bude aktivita se dvěma záložkami – záložkou se spárovanými zařízeními (viz obrázek 3.5) a záložkou s možností vyhledání v okolí dostupných zařízení (viz obrázek 3.6). Vyhledávání zařízení se v příslušné záložce spustí stisknutím tlačítka *scan* ve spodní části obrazovky. Pomocí těchto dvou záložek resp. fragmentů bude uživateli umožněno se připojit k jinému zařízení dle výběru. Připojení uživatel zahájí stisknutím tlačítka *connect* u příslušného zařízení.

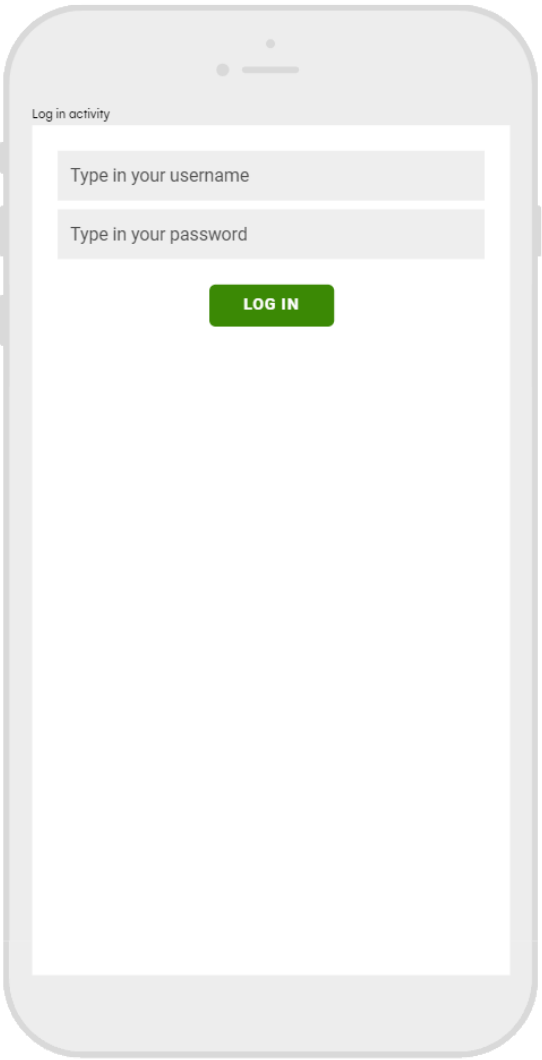


Obrázek 3.5: Úvodní aktivita – spárovaná zařízení



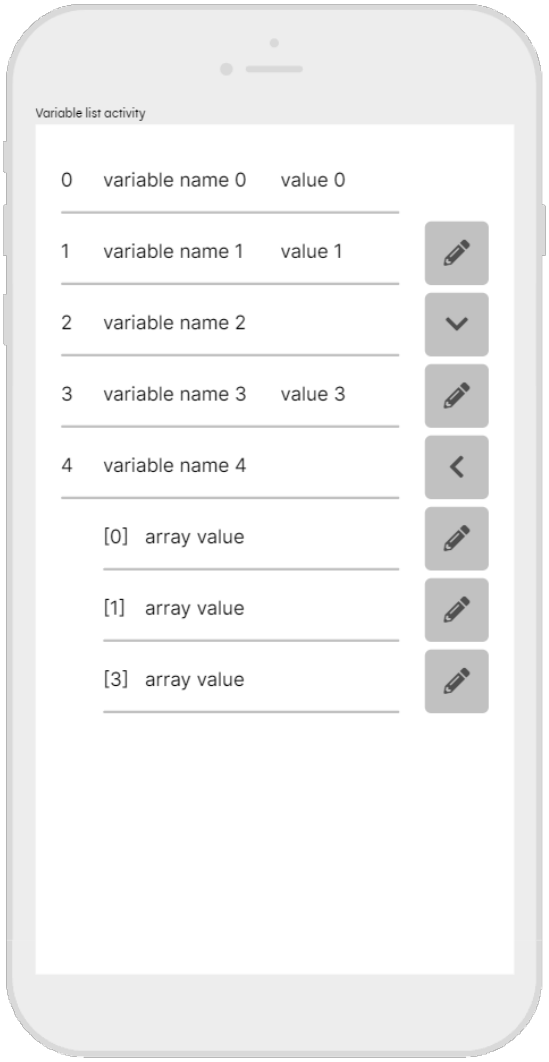
Obrázek 3.6: Úvodní aktivita – dostupná zařízení

Při úspěšném připojení bude uživatel přesměrován na aktivitu s přihlašovacím formulářem (viz obrázek 3.7). V něm bude od uživatele požadováno jeho uživatelské jméno a heslo. Stisknutím tlačítka *log in* bude provedena autentifikace uživatele. Při nesprávné kombinaci uživatelského jména a hesla bude uživateli zobrazeno upozornění.



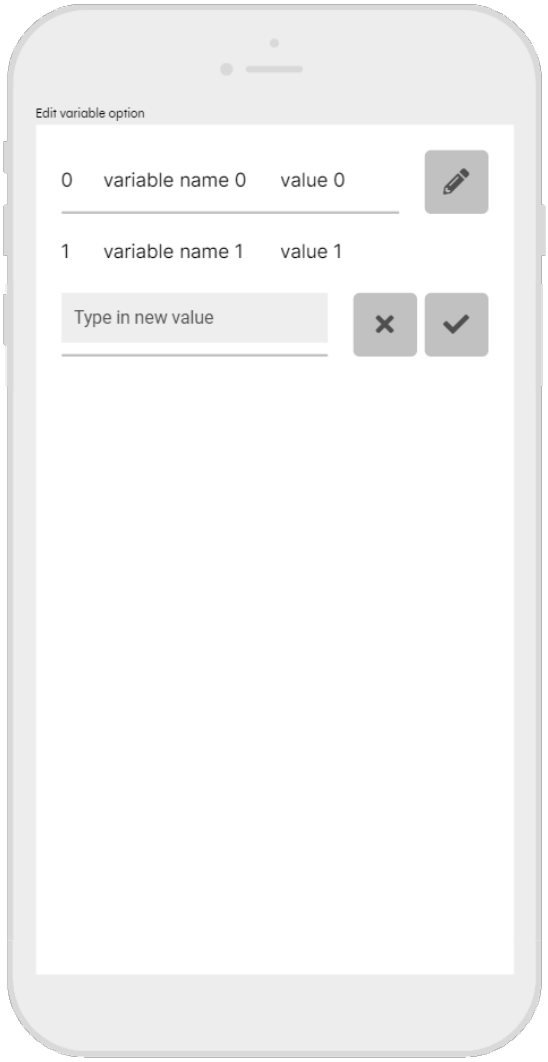
Obrázek 3.7: Přihlašovací aktivita

Po úspěšném přihlášení bude uživatel přesměrován na aktivitu se sdílenými proměnnými (viz obrázky 3.8 a 3.9). Proměnné budou uživateli zobrazeny jako položky v seznamu v pořadí podle jejich číselného identifikátoru – id. Položka seznamu se obecně bude skládat z id, názvu proměnné, hodnoty proměnné a tlačítka. Hodnota proměnné bude zobrazena v daném řádku v případě, že se nejedná o strukturu či pole hodnot. Pokud se bude jednat o strukturu nebo pole hodnot, tak jednotlivé položky takto složené proměnné budou možné zobrazit kliknutím na tlačítko s ikonou šipky směřující dolů. Naopak pro skrytí bude potřeba stisknout tlačítko s ikonou šipky směřovanou doleva v příslušném řádku (viz obrázek 3.8).



Obrázek 3.8: Aktivita se seznamem sdílených proměnných

Jednotlivé hodnoty proměnných bude možné upravit pomocí tlačítka s ikonou tužky v příslušném řádku. Po stisknutí se uživateli zobrazí textové pole, do kterého může zadat novou hodnotu proměnné a dvě tlačítka (viz obrázek 3.9). Tlačítka slouží pro potvrzení, či zrušení volby nastavení nové hodnoty do proměnné. Při potvrzení dojde k validaci vstupu a zapsání proměnné na server. Při zrušení bude formulář pro editaci skryt a bude opět zobrazeno tlačítko pro případnou úpravu. Tlačítko pro úpravu se uživateli zobrazí na základě jeho oprávnění (např. v obrázku 3.8 uživatel nemá oprávnění k editaci proměnné s id rovné 0).



Obrázek 3.9: Úprava sdílené proměnné

3.2 API

Zařízení Arduino bude poskytovat API (Application Programming Interface), které umožní aplikaci připojit se k zařízení Arduino a dále s ním komunikovat pomocí zpráv. Dále umožní tvorbu sdílených proměnných. Podle typu zprávy bude zařízení reagovat na zbytek zprávy.

API bude rozdělené do tří balíčků – gson, managers a shared_variables. Řídící třídou API bude třída *BluetoothManager*. Ta zajistí fungování všech částí dohromady. Třídu nalezneme v balíčku managers. Další důležitou třídou bude třída *UserManager*. Ta bude mít na starosti přidávání a ověřování

uživatelů. Třída *MessageManager* bude mít zase na starosti automatické odpovídání na jednotlivé zprávy dle jejich typu. Pro správu sdílených proměnných tu bude třída *SharedVariablesManager*, která umožní přidávání a upravování proměnných.

Balíček gson bude obsahovat třídu *Gson*, která bude mít na starosti serializaci a deserializaci zpráv. K určení typu zprávy bude sloužit jejich výčet uložený v třídě *MessageType*.

Posledním důležitým celkem je balíček shared_variables. V tom jsou analogicky jako v Android aplikaci strukturovány sdílené proměnné do balíčků single, array a record.

3.3 Struktura sdílených proměnných

Strukturu sdílených proměnných můžeme sledovat na obrázku 3.10. Všechny sdílené proměnné budou mít tyto atributy:

- **Id** – unikátní číselný identifikátor,
- **name** – název proměnné,
- **readAccess** – uživatelský přístup/oprávnění ke čtení proměnné,
- **writeAccess** – uživatelský přístup/oprávnění k zápisu do proměnné.

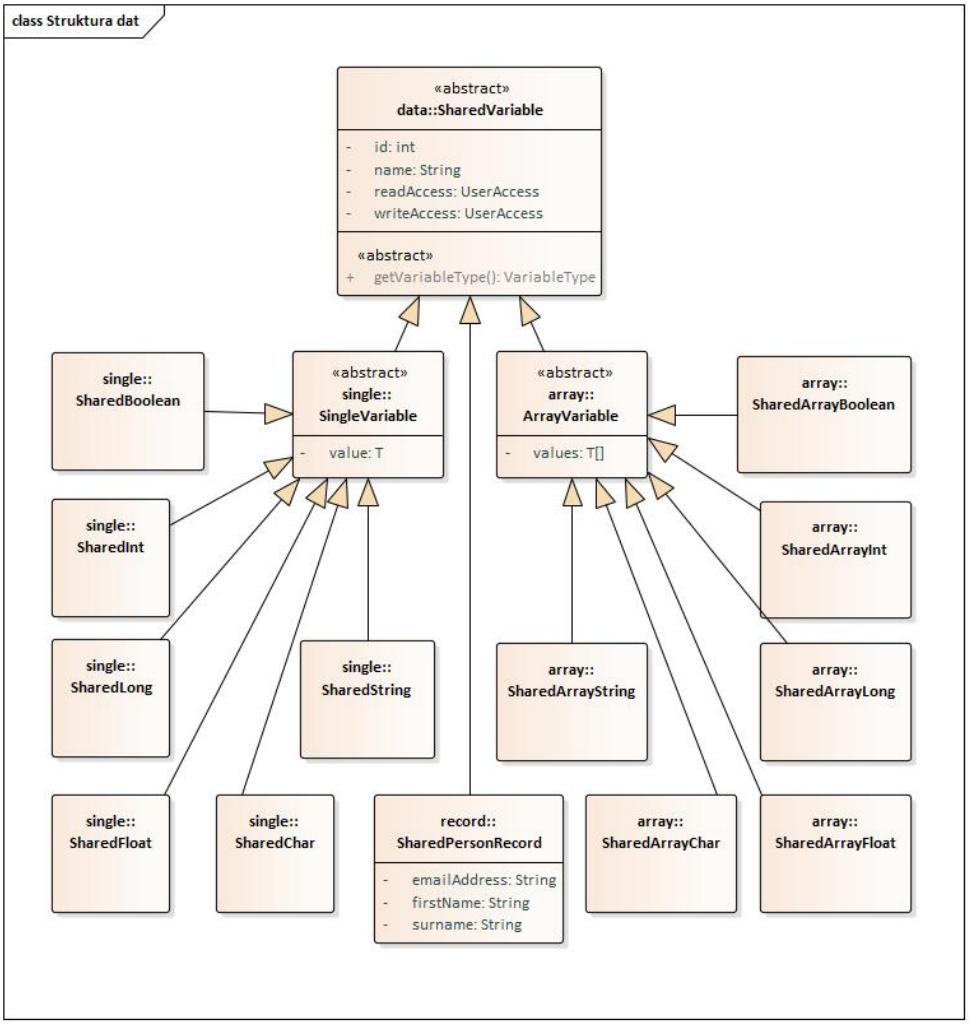
Jsou to atributy, které bude mít společný předek všech sdílených proměnných – SharedVariable. Kromě toho bude mít abstraktní metodu *getVariableType()*, která bude vracet typ sdílené proměnné. To umožní snadné rozeznání proměnné a ulehčí i serializaci a deserializaci dat. Třída bude umístěna v balíčku data (viz obrázek 3.1).

Další důležitou třídou je abstraktní třída *SingleVariable*. V té bude nový generický atribut *value*, který bude obsahovat danou hodnotu proměnné. V této třídě zůstane abstraktní metoda *getVariableType()* neimplemetnována. K implementaci metody dojde až u potomků třídy *SharedBoolean*, *SharedInt*, *SharedLong*, *SharedFloat*, *SharedChar* a *SharedString*. Dle názvu třídy bude zvolen vhodný datový typ hodnoty a i implementace metody *getVariableType()*. Třída *SingleVariable* a její potomci budou umístěni v balíčku single, který bude vnořen do balíčku data (viz obrázek 3.1).

Třídě *SingleVariable* bude podobna abstraktní třída *ArrayVariable*. Ta bude obsahovat generický atribut *values*, který bude držet pole hodnot. Abstraktní metoda *getVariableType()* bude v této třídě rovněž neiplementována. Implementovat ji budou potomci této třídy *SharedArrayBoolean*, *SharedArrayInt*, *SharedArrayLong*, *SharedArrayFloat*, *SharedArrayChar* a *SharedArrayString*. Dle názvu dané třídy bude zvolen vhodný datový typ pro hodnoty a implementace metody *getVariableType()*. Třída *ArrayVariable* a její potomci

budou umístěni v balíčku array, který bude vnořen do balíčku data (viz obrázek 3.1).

Třetím typem implementace třídy *SharedVariable* budou záznamy uložené v balíčku record, který bude vnořen v balíčku data. Záznamy uložené v tomto balíčku nebudou mít žádné specifické společné vlastnosti, tudíž budou přímo dědit ze třídy *SharedVariable*. Příkladem záznamu je třída *SharedPersonRecord* v obrázku 3.10.



Obrázek 3.10: Model tříd – struktura dat

Kapitola 4

Implementace

V této kapitole je popsána implementace Android aplikace a Arduino API. Dále se kapitola zabývá zapojením jednotlivých periferií – LED diody, displeje, tlačítek a GSM modulu – na desku Arduino.

4.1 Android aplikace

Aplikace je dle návrhu strukturovaná do striktní třívrstvé architektury. Ta je složena z prezentační vrstvy (balíček presentation), business vrstvy (balíček business) a datové vrstvy (balíček data).

Vedle těchto balíčků je ještě balíček enums, který obsahuje výčty používané v aplikaci. Jedná se o výčet typů zpráv (enum *MessageType*), výčet typů uživatelských přístupů (enum *UserAccess*) a výčet typů sdílených proměnných (enum *VariableType*).

Vzhledem k použití technologie Bluetooth bylo zapotřebí přidat do aplikace správná oprávnění. Oprávnění se přidávají do souboru AndroidManifest.xml, který nalezneme ve složce app/manifests. První oprávnění má název BLUETOOTH. To umožňuje připojit se ke spárovaným zařízením. Dalšími nezbytnými oprávněními pro chod aplikace jsou BLUETOOTH_ADMIN a ACCESS_FINE_LOCATION. Jejich úloha je zjišťovat v okolí dostupná zařízení a umožnit se nimi spárovat. [34]

4.1.1 Prezentační vrstva

Prezentační vrstva přijímá uživatelské vstupy, předává je business vrstvě a zobrazuje odpověď na uživatelský vstup. Prezentační vrstva se skládá ze dvou hlavních komponent – ze souborů s rozložením komponent a z tříd, které ovládají chování těchto komponent.

V balíčku presentation bychom mohli najít čtyři vnořené balíčky – activities, fragments, adapters a interfaces. Balíček activities obsahuje jednotlivé aktivity, které se zobrazují uživateli. Aktivita je v zásadě jedna scéna,

která zabírá celou obrazovku zařízení s běžící aplikací. Aktivita se typicky skládá z jednoho či více fragmentů, které nalezneme právě v balíčku fragments. Ve fragmentech se už většinou nacházejí jednotlivé komponenty (např. tlačítka, textová pole, atd.), jejichž rozložení ve fragmentu je popsáno v balíčku app/res/layout. V něm najdeme XML soubory popisující rozložení komponent na obrazovce. Chování jednotlivých komponent je popsáno v příslušné třídě pro danou aktivitu/fragment obsahující tuto komponentu.

Některé aktivity/fragmenty využívají složitější rozložení. Mezi takové patří záložky s fragmenty, či seznamy různých položek generovaných dynamicky (v kódu). Pro používání těchto rozložení je zapotřebí různých adaptérů. Ty najdeme v balíčku adapters.

Posledním balíčkem je balíček interfaces, který obsahuje různá rozhraní, používaná k aktualizování zobrazovaného obsahu.

První použitou aktivitou v aplikaci je aktivita *DeviceListActivity*. Ta zobrazuje seznamy zařízení, ke kterým je možné se připojit. Je složena ze dvou záložek, které obsahují fragmenty *BondedDevicesFragment* a *ScanDevicesFragment*. Pro obsluhu záložek aktivita využívá adaptér *DeviceTabsAdapter*. Oba fragmenty používají *DeviceListFragment* pro zobrazování seznamu zařízení. *BondedDevicesFragment* je jeho přímým potomkem a zobrazuje seznam spárovaných zařízení, ke kterým je možné se připojit. *ScanDevicesFragment* využívá *DeviceListFragment* jako většinovou součást záložky pro zobrazování seznamu dostupných zařízení. Kromě toho obsahuje tlačítko pro zahájení vyhledávání v okolí dostupných zařízení. Pro zobrazování správných položek využívá *DeviceListFragment* adaptér *DeviceListAdapter*.

Druhou aktivitou je *LogInActivity*. Ta neobsahuje žádný fragment, ale má komponenty umístěné přímo do layout souboru pro aktivitu. *LogInActivity* představuje přihlašovací formulář pro uživatele. Skládá se z dvou upravitelných textových polí – jedno pro jméno uživatele a druhé pro heslo – a tlačítka pro odeslání údajů k autentifikaci. Po stisknutí tlačítka uživatel čeká na přesměrování na další aktivitu, nebo zprávu o chybných přihlašovacích údajích. Pro zachycení odpovědi serveru aktivita používá *BroadcastReceiver* (viz zdrojový kód 2). Díky němu může zachytit broadcast zprávu vyslanou z prezentační vrstvy a podle ní zareagovat.

Poslední aktivitou je *VariableListActivity*. Ta obsahuje jeden fragment třídy *VariableListFragment*. Ten je již o něco komplexnější. *VariableListFragment* zobrazuje uživateli sdílené proměnné. Po vytvoření čeká na přijetí zprávy o získání všech sdílených proměnných na severu. Jakmile zprávu obdrží, zažádá si o proměnné *BluetoothMessagingService* a následně je zobrazí. Další důležitou funkcí tohoto fragmentu je přijímání zprávy o změně některé z proměnných. Když takovou zprávu obdrží, tak pomocí instance rozhraní *ValueSettingListener* informuje příslušný adaptér – *VariableRecyclerViewAdapter* pro jednoduché proměnné, *ArrayVariableRecyclerViewAdapter* pro proměnné obsahující pole hodnot, nebo *RecordVariableRecyclerViewAdapter* pro proměnné obsahující záznam – o úspěšném, či neúspěšném pokusu o nastavení

```
private final BroadcastReceiver verificationReceiver =
    new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            MessageType extra =
                (MessageType) intent.getSerializableExtra(
                    BluetoothMessagingService.VERIFY_EXTRA);

            if (extra.equals(MessageType.VERIFY_FAIL)) {
                // ...
            } else if (extra.equals(MessageType.VERIFY_OK)) {
                // ...
            }
        }
    };
```

Zdrojový kód 2: Ukázka využití instance třídy *BroadcastReicievер*

proměnné. Instance rozhraní *ValueSettingListener* jsou pro jednoduché proměnné a záznamy uložené v mapě. Klíčem je id proměnné. U proměnných obsahující pole hodnot jsou instance rozhraní *ValueSettingListener* uloženy do tabulky. Díky ní se při změně jedné hodnoty z pole nemusí aktualizovat celé pole ale jen jedna jeho hodnota. Pro implementaci tabulky byla použita knihovna Guava [35] od společnosti Google Inc.

Třída *VariableRecyclerViewAdapter* se stará o správné zobrazování jednotlivých proměnných. Zobrazuje celé jednoduché proměnné pomocí třídy *SingleViewHolder*, a také název záznamů a pole proměnných pomocí *ArrayViewHolder*. Třída *VariableRecyclerViewAdapter* zajišťuje možnost upravování jednoduchých proměnných a kroky s tím spojené – validaci nové hodnoty, odeslání hodnoty na server, reakci na nastavení či nenastavení proměnné. Řídí také jednotlivé grafické prvky používané při úpravě proměnné. Pro záznamy a proměnné s polem hodnot se o to starají třídy *RecordVariableRecyclerViewAdapter* a *ArrayVariableRecyclerViewAdapter*.

Při úpravě jakékoliv proměnné je používán jednotný layout soubor layout_edit_single_variable, který je vždy nejprve skrytý a je zobrazen v momentě, kdy uživatel upravuje danou proměnnou. Tím dochází k úspoře napsaného kódu. Pro vkládání tohoto souboru na příslušná místa v XML se používá element typu include (viz zdrojový kód 3). Přes jeho instanci, kterou můžeme získat na základě jeho id, pak můžeme přistupovat k jeho jednotlivým komponentám a nastavit jejich chování.

```
<include
        android:id="@+id/editVariableInclude"
        layout="@layout/layout_edit_single_variable"
        android:visibility="gone" />
```

Zdrojový kód 3: Ukázka využití elementu typu include

4.1.2 Business vrstva

Business vrstva je hlavním řídícím prvkem celé aplikace. Je uložena v balíčku business a má jeden vnořený balíček services.

Nejdůležitější třídou business vrstvy je třída *BluetoothMessagingService*. Tato třída nemá veřejný konstruktor. Místo toho se k jejím metodám přistupuje pomocí statické metody *getInstance(Activity context)*, kde parametr *context* reprezentuje kontext, ve kterém se instance třídy používá. Instance třídy je inicializována při jejím prvním použití. Tomuto typu inicializace se říká *lazy initialization*. *BluetoothMessagingService* zajišťuje zapnutí Bluetooth, párování zařízení, připojení k serveru, posílání a přijímání zpráv, ukládání dočasných dat a reakci na přijaté zprávy. Pro většinu z těchto úkonů využívá jiné třídy z business vrstvy.

První funkcionalitou *BluetoothMessagingService* je obsluha Bluetooth. Při spuštění aplikace je kontrolováno zapnutí Bluetooth. V případě vypnutí je uživateli zobrazeno dialogové okno s možností zapnutí. O to se postará privátní metoda *turnBluetoothOn()*. V momentě, kdy je Bluetooth zapnutý, je aktualizován seznam spárovaných zařízení. O to se stará *BroadcastReceiver*, který zachytává změny v stavu zapnutí Bluetooth. V případě, že je Bluetooth vypnut, jsou skryta všechna zařízení pro připojení. Zahájení vyhledávání v okolí dostupných zařízení je realizováno privátní metodou *startDiscovery()*, která je využita ve veřejné metodě *findDevices()*.

Další funkcionalitou třídy je připojování se k zařízení/serveru. Pro tyto účely se používá metoda *connect(String deviceAddress)*. Ta využívá vláken *ConnectionThread* a *MessagingThread* pro připojení k zařízení definovaného parametrem *deviceAddress* a následného posílání a přijímání zpráv. O stavu připojování je třída *BluetoothMessagingService* informována prostřednictvím rozhraní *SocketCreationListener* a *SocketConnectionListener*. Pro informování o přijetí dat či ztrátě spojení slouží zase rozhraní *bluetoothDataListener*. V něm je po přijetí dat volána metoda *respondToMessage(MessageType messageType, String jsonMessage)* z další důležité třídy *MessageHandler*.

Třída *MessageHandler* má na starosti odpovídání na příchozí zprávy. Obsahuje mapu, do které se registrují odpovědi na jednotlivé typy zpráv. Klíčem k dané odpovědi je tedy typ zprávy a hodnotou je instance implementovaného rozhraní *IMessageResponse*. K registraci a implementace jednotlivých odpovědí na různé typy zpráv dochází v třídě *BluetoothMessagingService* v privátní metodě *createMessageHandler()* (viz zdrojový kód 4). Metodou se inicializuje

atribut třídy *BluetoothMessagingService*. K inicializaci dochází v konstruktoru. Díky tomu pak na daný typ zprávy může být automaticky provedena odpověď.

```
private MessageHandler createMessageHandler() {
    MessageHandler handler = new MessageHandler();
    handler.registerMessage(MessageType.VERIFY_OK,
            (String jsonMessage) -> {
                // implementace odpovědi na zprávu jsonMessage
            });
    // ...
    return handler.
}
```

Zdrojový kód 4: Ukázka implementace odpovědi na zprávu

Pro komunikaci mezi serverem a klientem jsou použity tyto typy zpráv:

- **VERIFY**
  Zpráva odeslaná od klienta na server s žádostí o ověření uživatele.
- **VERIFY_OK**
  Zpráva odeslaná klientovi ze serveru oznamující úspěšnou verifikaci uživatele a následné přihlášení.
- **VERIFY_FAIL**
  Zpráva odeslaná klientovi ze serveru oznamující neúspěšnou verifikaci uživatele.
- **LOG_OUT**
  Zpráva odeslaná od klienta na server oznamující odhlášení uživatele.
- **GET_ALL_VARIABLES**
  Zpráva odeslaná od klienta na server vyžadující všechny sdílené proměnné dle oprávnění uživatele.
- **ALL_VARIABLES**
  Zpráva odeslaná klientovi ze serveru obsahující vyžádané sdílené proměnné.
- **SET_VARIABLE**
  Zpráva odeslaná od klienta na server vyžadující nastavení proměnné na novou hodnotu.
- **SET_VARIABLE_OK**
  Zpráva odeslaná klientovi ze serveru potvrzující úspěšné nastavení proměnné.

- **SET_VARIABLE_FAIL**
  Zpráva odeslaná klientovi ze serveru oznamující neúspěšné nastavení proměnné.

- **UPDATE_VARIABLE**
  Zpráva odeslaná klientovi ze serveru oznamující o změně hodnoty proměnné.

Po připojení k serveru dochází k přihlašování uživatele. Pro vyslání žádosti o ověření uživatele na serveru slouží metoda *verifyUser(String username, String password)*. Odpověď na tuto žádost je pak poslána broadcast zprávou, kterou odchytává prezentační vrstva. Třída *BluetoothMessagingService* rovněž poskytuje metodu *logOut()*, která informuje server o odhlášení uživatele a smaže lokálně uložená uživatelská data.

Server po úspěšném přihlášení odešle seznam sdílených proměnných. Proměnné jsou uloženy v instanci třídy *SessionData*. Prezentační vrstva k těmto datům pak může skrze metodu *getSessionData()* z třídy *BluetoothMessagingService* přistupovat. Úprava proměnných je realizována metodami *setSingleVariable(int id, T newValue)*, setArrayVariable(int id, int index, T newValue) a *setRecordVariable(int id, RecordVariable variable)*. Tyto metody nastavují hodnotu sdílených proměnných na serveru a po schválení je poslána broadcast zpráva prezentační vrstvě.

4.1.3 Datová vrstva

Datová vrstva obsahuje třídy sdílených proměnných, třídu *Gson* a *Limits*. Hierarchie dědičnosti proměnných je skoro stejná, jako je uvedena v návrhu. Společným předkem všech sdílených proměnných je třída *SharedVariable*. Ta má atributy id, název proměnné, a přístupy ke čtení proměnné a zápisu do proměnné. Dále obsahuje důležitou abstraktní metodu *getVariableType()*. Z této abstraktní třídy pak dědí tři další abstraktní třídy – *SingleVariable*, *ArrayVariable* a *RecordVariable*.

Abstraktní třída *SingleVariable* má generický parametr *T*, který udává typ uložené hodnoty v proměnné. Z této třídy pak dědí třídy *SharedBoolean*, *SharedInt*, *SharedLong*, *SharedFloat*, *SharedChar* a *SharedString*. Každá z těchto tříd implementuje abstraktní třídu *getVariableType()* a vrací příslušnou hodnotu vzhledem k datovému typu hodnoty uložené v proměnné.

Na stejném principu je strukturováno dědění z abstraktní třídy *ArrayVariable*. Ta však obsahuje místo jedné hodnoty celé pole hodnot.

Realizace záznamů se od původního návrhu liší tím, že jednotlivé typy záznamů nedědí přímo z abstraktní třídy *SharedVariable*. Je zde mezistupeň, a tím je abstraktní třída *RecordVariable*. Ta obsahuje seznam záznamových položek reprezentovaných třídou *RecordComponent*. Položky se skládají z názvu a hodnoty proměnné. Podle typu hodnoty položky můžeme rozlišit na

BooleanComponent, *IntComponent*, *LongComponent*, *FloatComponent*, *CharComponent* a *StringComponent*. Jednotlivé sdílené záznamy pak dědí z třídy *RecordVariable* a v konstruktoru určují, které položky budou v daném záznamu (viz zdrojový kód 5). Záznamy jsou pouze jednoúrovňové. Tento přístup byl zvolen pro jednodušší přidávání dalších typů záznamů. Kdyby byl dodržen původní návrh, tak by programátor musel pro každý nový typ záznamu vytvářet nový XML layout a následně implementovat chování jeho jednotlivých komponent. Tímto zvoleným řešením tato povinnost odpadá.

```
public SharedPersonRecord(int id, String name,
                          UserAccess readAccess,
                          UserAccess writeAccess,
                          String firstName, String surname,
                          String emailAddress) {
    super(id, name, readAccess, writeAccess);
    components.add(
                new StringComponent("first_name", firstName));
    components.add(
                new StringComponent("surname", surname));
    components.add(
                new StringComponent("email", emailAddress));
}
```

Zdrojový kód 5: Ukázka implementace konstruktoru sdíleného záznamu

Další důležitou součástí datové vrstvy je třída *Gson*. Jejím úkolem je serializace a deserializace zpráv. Třída obsahuje mnoho statických metod právě pro tyto účely. Mezi těmito metodami bychom našli například metodu *verifyUserJson(String username, String password)*, která vytvoří textovou zprávu ve formátu JSON pro ověření uživatele. Obecně v této třídě platí, že metody, jejichž název končí na *Json()* vytvářejí textové zprávy ve formátu JSON. Ostatní veřejné metody jsou používány pro deserializaci dat – převedení zprávy na data.

V datové vrstvě bychom našli ještě jednu třídu spojenou se sdílenými proměnnými. Tou třídou je třída *Limits*. V ní jsou definovány limitní hodnoty pro sdílené číselné proměnné – pro *SharedInt*, *SharedArrayInt*, *SharedLong*, *SharedArrayLong*, *SharedFloat* a *SharedArrayFloat*. Pomocí příslušné statické metody je pak možné zkontrolovat validitu sdílené proměnné. Toho se využívá v prezentační vrstvě při úpravě proměnné. Je to určitá kontrola nově nastavené hodnoty před odesláním na server.

4.1.4 Uživatelské prostředí

Výsledná podoba uživatelského prostředí se oproti původnímu návrhu téměř nezměnila. Jedinou změnou, kterou můžeme vidět je realizace obrazovky zob-

razující sdílené proměnné. Pro větší přehlednost byl odstraněn sloupec s id proměnných a názvy jednotlivých proměnných jsou zvýrazněny. Tím je dosažena větší přehlednost při zobrazování proměnných obsahujících pole hodnot a také záznamů.

4.2 Arduino

Řešení zadání je pro Arduino rozděleno do dvou částí. První částí je API, které je použito pro komunikaci s aplikací. Druhou je zapojení jednotlivých periferií k desce Arduino. Pro ukázky zapojení byl použit nástroj Fritzing [36].

4.2.1 API

Arduino API je rozděleno do tří balíčků – gson, managers a shared_variables.

Balíček gson obsahuje soubory, které se starají o serializaci a deserializaci dat. Najdeme zde třídu *Gson* a výčtovou třídu *MessageType*. Třída *MessageType* obsahuje množinu typů zpráv, které server používá pro komunikaci s klientem. Třída *Gson* provádí serializaci a deserializaci dat ve formátu JSON. Při vytváření zpráv pro odeslání se snaží zbytečně neplýtvat již tak omezenou pamětí. To zajišťuje počítáním kapacity posílané zprávy. Pro spočítání kapacity slouží makra poskytnutá knihovnou. Pokud zpráva obsahuje textové řetězce, tak je nutné započítat jejich dálku do výsledné kapacity. Naopak při přijímání dat je snaha mít připravený dostatečně velký buffer pro přijetí celé zprávy. Formát posílaných zpráv můžeme vidět v ukázce kódu 6.

```
{
    "message_type":"SET_VARIABLE",
    "variable": {
        "id":0,
        "value":117
    }
}
```

Zdrojový kód 6: Ukázka zprávy – žádost o nastavení proměnné

Struktura sdílených proměnných je uložena v balíčku shared_variables. Ten obsahuje další tři balíčky – single, array a record. Ty popořadě obsahují implementace jednoduchých proměnných, pole proměnných a záznamů. Společným předkem všech proměnných je třída *SharedVariable*. Její implementace je obdobná jako u stejnojmenné třídy v Android aplikaci. Podobnost je i u implementace jednoduchých proměnných a pole proměnných. Rozdíly můžeme najít pouze v syntaxi a případně v lehce jiném názvu třídy. Proto zde třídy nejsou detailně rozepsány. Rozdílnost je u tříd z balíčku record. Zde

záznamy dědí přímo z třídy *SharedVariable*. Paměťová náročnost jednotlivých proměnných byla zkoumána pomocí knihovny Arduino-MemoryFree [37].

Jádro celé aplikace se nachází v balíčku managers, konkrétně ve třídě *BluetoothManager*, která řídí celou komunikaci s klientem přes Bluetooth. Pro odesílání a přijímání zpráv třída používá *MessageManager*. Třída *MessageManager* funguje podobně jako třída *MessageHandler* v Android aplikaci. Obsahuje seznam reakcí na zprávy. Reakce na zprávu je reprezentována pomocnou strukturou *MessageAction*. Reakce jsou pomocí lambda výrazu definovány v metodě *registerMessageActions()* třídy *BluetoothManager* (viz zdrojový kód 7). Při přijetí zprávy je zjištěn typ zprávy, a následně je vyvolána příslušná reakce.

```
void BluetoothManager::registerMessageActions() {
    messageManager
            ->addMessageAction(
                MessageType::LOG_OUT,
                [](const String &message, void *context) {
        auto *tmp = static_cast<BluetoothManager *>(context);
        tmp->isUserLoggedIn = false;
    });
    // ...
}
```

Zdrojový kód 7: Ukázka automatické reakce serveru na odhlášení

Třída *BluetoothManager* dále využívá *UserManager*. Ten, jak již jeho název napovídá, spravuje uživatelské účty. Umožňuje přidávání nových uživatelů (metodou *add(User &user)*) a jejich autentifikaci (metodou *verifyUser(const User &user)*).

Další důležitou třídou používanou v třídě *BluetoothManager* je třída *SharedVariablesManager*. Ta umožňuje přidávání nových sdílených proměnných do kolekce proměnných (metodou *add(SharedVariable &variable)*) a následnou práci s nimi.

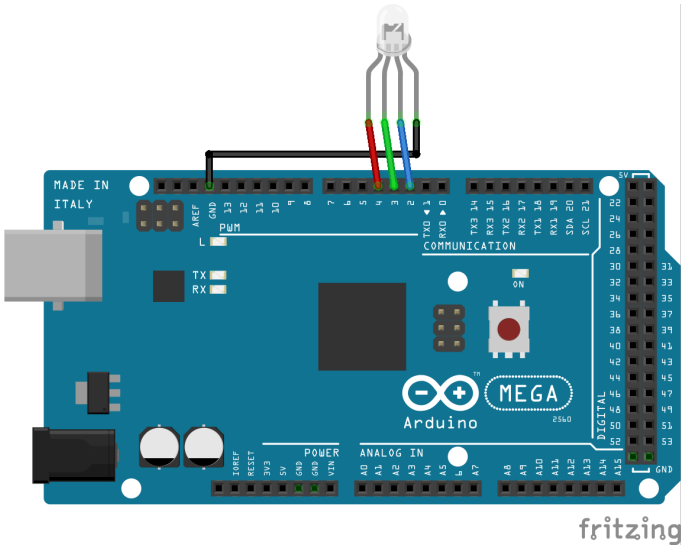
Použití celého API v nějaké hlavní třídě main.cpp pro Arduino desku je následující. Nejprve je nutné provést import hlavičkového souboru třídy *BluetoothManager*. Pak můžeme vytvořit globální instance všech manažerských tříd – *BluetoothManager*, *UserManager*, *MessageManager* a *SharedVariablesManager*. U manažerských tříd *UserManager* a *SharedVariablesManager* je možné v konstruktoru nastavit přesný počet použitých uživatelů/sdílených proměnných. To zajistí lepší práci s pamětí.

Ve třídě main.cpp se budou nacházet dvě metody – *setup()* a *loop()*. V metodě *setup()* provedeme základní nastavení manažerů. Tím je myšleno přidání uživatelů do instance třídy *UserManager*, přidání proměnných do instance třídy *SharedVariablesManager* a nastavení instance třídy *BluetoothManager*

ostatními třemi manažerskými třídami. V metodě *loop()* pak již stačí jen zavolat metodu *manage()* na instanci třídy *BluetoothManager*. Po zkompilování a nahrání kódu na desku Arduino je již pak možné komunikovat s Android aplikací.

4.2.2 LED dioda

Zde je uveden příklad zapojení a použití LED diody. Pro zmíněné účely byl použit RGB LED modul KY-016. Na modulu jsou 4 kontakty – R pro červenou barvu, G pro zelenou, B pro modrou a GND pro uzemnění. Kontakty zapojíme k desce Arduino dle obrázku 4.1. Kontakty pro barvy jsou zapojeny do PWM pinů a kontakt GND je zapojen k jednomu z uzemnění desky.



Obrázek 4.1: Zapojení LED modulu KY-016, upraveno podle návodu z [27]

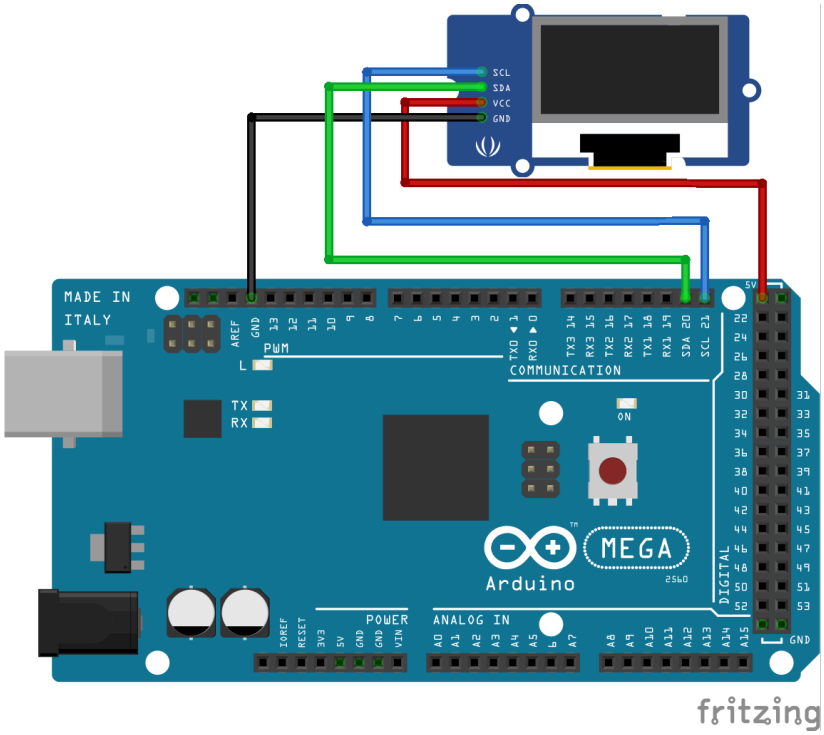
```
void setup() { // R - 4, G - 3, B - 2
    pinMode(4, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(2, OUTPUT);
}
void loop() {
    analogWrite(4, 255); delay(1000);
    analogWrite(3, 255); delay(1000);
    analogWrite(2, 255); delay(1000);
}
```

Zdrojový kód 8: Ukázka práce s RGB modulem KY-016, upraveno podle návodu z [27]

Pak je potřeba napsat kód, který bude s modulem pracovat. Použitím funkce *pinMode* nastavíme piny pro všechny tři barvy. To provedeme ve funkci *setup()*. Ve funkci *loop()* pak může například postupně problikat všechny barvy. To provedeme pomocí funkce *analogWrite*, jejíž první parametr udává, do kterého pinu budeme zapisovat, a druhý nastavuje intenzitu svitu LED diody. Intenzita se pohybuje mezi čísly 0 a 255, kde 0 značí žádný svit a 255 svit maximální. Příklad použití můžeme vidět v zdrojovém kódu 8.

4.2.3 Displej

Zde je uveden příklad zapojení a použití OLED displeje. Piny na displeji zapojíme tak, že GND na displeji je zapojeno do GND na desce. To samé platí pro piny SCL a SDA. Pin VCC na desce zapojíme do 5V napájení (viz obrázek 4.2).



Obrázek 4.2: Zapojení OLED displeje, upraveno podle návodu z [28]

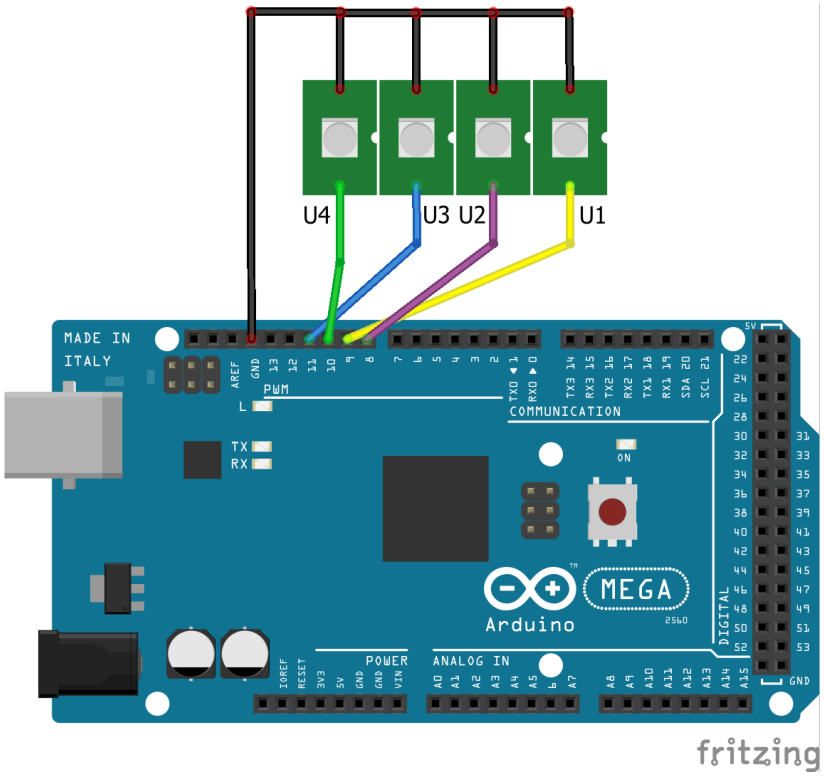
Pro zobrazování textu na displeji použijeme knihovnu U8glib. [38] Nejprve inicializujeme OLED displej, pak u něj nastavíme velikost fontu, obnovíme obsah displeje a na danou pozici vykreslíme zvolený text (viz zdrojový kód 9). Knihovna nám nikterak nehlídá přetékání textu mimo obrazovku. To je nutné si při programování pohlídat.

```
#include "U8glib.h"

U8GLIB_SSD1306_128X64 oled(U8G_I2C_OPT_NONE);

void setup() {
    oled.setFont(u8g_font_5x7);
    oled.firstPage();
    oled.setPrintPos(0, 10);
    oled.print("Adruino OLED test");
}
```

Zdrojový kód 9: Ukázka použití OLED displeje, upraveno podle návodu z [28]

4.2.4 Tlačítka

Zde je popsáno zapojení membránové klávesnice se čtyřmi tlačítky a práce s ní. Zapojení je znázorněno v obrázku 4.3.



Obrázek 4.3: Zapojení membránové klávesnice, upraveno podle návodu z [29]

Zvláštností při zapojování membránové klávesnice je překřížení kontaktů pro první a druhé a pro třetí a čtvrté tlačítko. Tlačítka totiž nemají piny

seřazené podle jejich pořadí.

Pro čtení stisknutí tlačítka jej nejprve nastavíme pomocí funkce *pinMode*, a pak z nich můžeme číst funkcí *digitalRead*. Pokud funkce *digitalRead* vrátí konstantu LOW, je tlačítko stisknuto. Pokud vrátí hodnotu HIGH, tlačítko stisknuté není. To můžeme vidět v zdrojovém kódu 10. Ukázka popisuje rozpoznávání puštění stisknutého tlačítka.

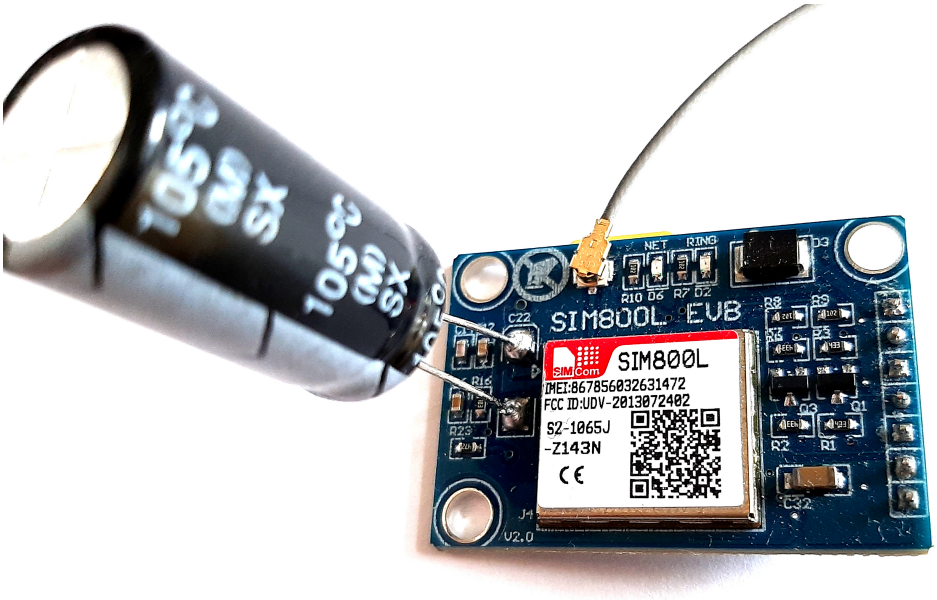
```
#define btn_1 9
#define btn_2 8
#define btn_3 11
#define btn_4 10
#define btn_count 4

const int buttons[] = {btn_1, btn_2, btn_3, btn_4};
bool pressed[] = {false, false, false, false};

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < btn_count; ++i) {
    pinMode(buttons[i], INPUT_PULLUP);
  }
}

void printButton(int buttonNumber) {
  Serial.print("Button ");
  Serial.print(buttonNumber);
  Serial.println(" pressed");
}

void loop() {
  for (int i = 0; i < btn_count; ++i) {
    if (digitalRead(buttons[i]) == LOW) {
      pressed[i] = true;
    }
    if (digitalRead(buttons[i]) == HIGH && pressed[i]) {
      pressed[i] = false;
      printButton(btn_count - i);
    }
  }
}
```

Zdrojový kód 10: Ukázka použití membránové klávesnice, upraveno podle návodu z [29]

4.2.5 GSM modul

Z použitých periferií je GSM modul nejsložitější na použití. Modul vyžaduje stabilní napájení, které se dá zajistit externím napájením například z Li-po baterie, nebo pomocí převodníku proudu ze sítě. Pokud modul nemá stabilní napájecí zdroj, tak dochází k jeho častému resetování. To znemožňuje využívání jeho funkcí. [39]

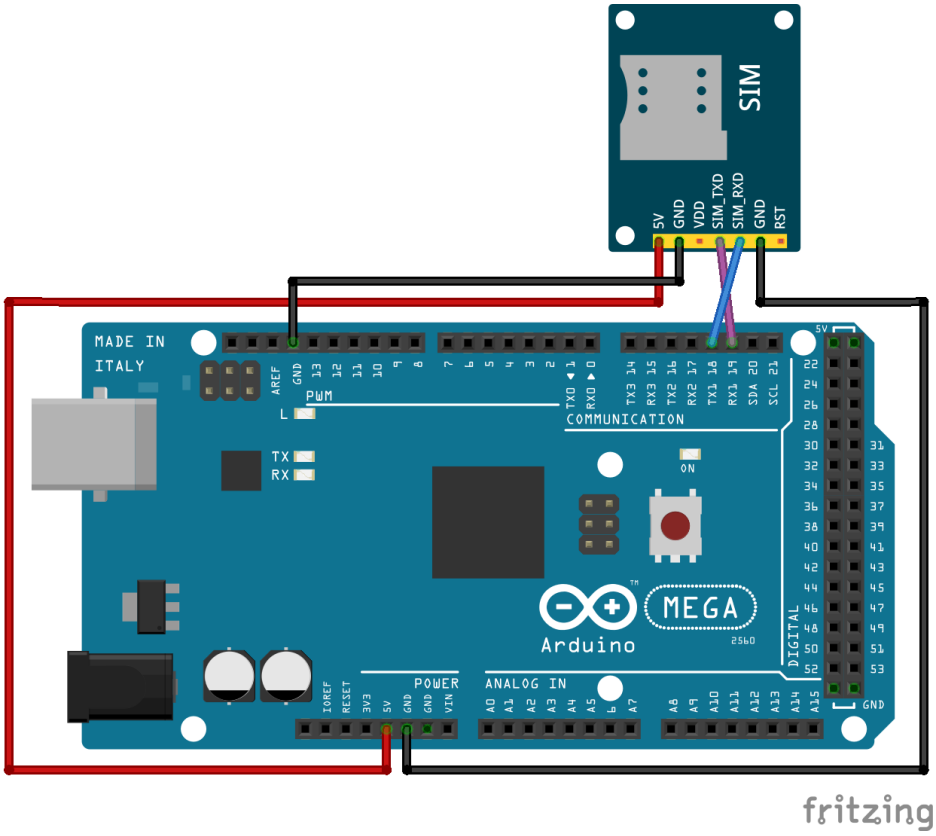
Pro vyřešení tohoto problému byl použit jiný postup, než výše zmíněné pomocí externího napájení. Na modulu se nacházel kondenzátor, označený jako C22 (viz obrázek 4.4), který měl velmi malou kapacitu. Výměnou kondenzátoru za jiný s větší kapacitou docílíme toho, že můžeme modul napájet přímo z desky Arduino. V tomto případě byl dán kondenzátor s kapacitou 4700 μF. V řešení, podle kterého byla výměna kondenzátoru provedena, byl použit menší kondenzátor s kapacitou 2200 μF. [40]



Obrázek 4.4: Výměna kondenzátoru na modulu GSM

Zapojení GSM modulu je pak znázorněno v obrázku 4.5. Pin na modulu pro napájení je zapojen do 5V napájení na desce a oba dva piny GND na modulu jsou zapojeny do uzemnění na desce. Pin SIM_TXD je zapojen do RX1 a SIM_RXD je zapojen do TX1. Piny RX1 a TX1 dohromady na desce Arduino Mega 2560 odpovídají hardwarovému sériovému monitoru označovanému názvem Serial1.

Programování modulu je znázorněno v zdrojovém kódu 11. Pro ovládání GSM modulu se zde používají AT příkazy. Jejich kompletní výčet AT příkazů



Obrázek 4.5: Zapojení modulu GSM, upraveno podle [41]

pro práci s modulem bychom našli v jeho dokumentaci. [42] V zdrojovém kódu 11 můžeme vidět příklad odesílání a příjem SMS zpráv. Nejprve jsou ve funkci *setup()* nastaveny sériové monitory pro práci s konzolí a pro práci s GSM modulem. Ve funkci *loop()* pak dochází ke čtení uživatelského vstupu ze sériového monitoru.

Pokud je na vstup posláno písmeno s, je zavolána funkce *sendMessage()*. Ta na GSM modulu zapne mód pro práci s textovými zprávami a nastaví číslo příjemce (příkaz AT+CMGS). Po zalomení řádky je zadán text zprávy. Nakonec je zpráva odeslána přidáním znaku CTRL-Z, který má v ASCII tabulce číslo 26. Ve funkci *loop()* také dochází ke čtení stavu GSM modulu a jeho vypisování do sériového monitoru. Proto je v něm po úspěšném odeslání zprávy zobrazena zpráva o úspěchu.

V případě, že je na vstupu napsáno písmeno r, tak je zavolána funkce *recieveMessage()*. Ta umožňuje přijímání zpráv. Nejprve je zapnut mód pro práci s textovými zprávami (příkaz AT+CMGF=1), pak je zapnuto živé přijímání zpráv. Přijaté zprávy jsou zobrazovány do sériového monitoru.

```
#define sim Serial1
String number = "+420xxxxxxxxx"; // Telefonní číslo

void setup() {
  Serial.begin(9600);
  sim.begin(9600);
  Serial.println("Type s to send a SMS, r to receive a SMS");
}

void loop() {
  if (Serial.available() > 0)
    switch (Serial.read()) {
      case 's':
        sendMessage();
        break;
      case 'r':
        recieveMessage();
        break;
    }
  if (sim.available() > 0)
    Serial.write(sim.read());
}

void sendMessage() {
  sim.println("AT+CMGF=1"); //Nastaví GSM modul do Text módu
  delay(200);
  sim.println("AT+CMGS=\"" + number + "\"\r");
  delay(200);
  sim.println("My SMS text");
  delay(100);
  sim.println((char)26); // CTRL+Z v ASCII kódu
  delay(200);
}

void recieveMessage() {
  sim.println("AT+CMGF=1");
  delay (200);
  sim.println("AT+CNMI=1,2,0,0,0"); // Přijímání SMS
  delay(200);
}
```

Zdrojový kód 11: Ukázka použití GSM modulu, upraveno podle [41]

4.3 Přidání záznamu

Vzhledem k očekávanému rozšiřování aplikace o další typy záznamů se tato kapitola zaměřuje na popis jednotlivých kroků pro přidání nového záznamu. Záznam je potřeba přidat jak do Android aplikace, tak do Arduino API. Je nutné podotknout, že záznam může obsahovat pouze množinu atributů jednoduchých datových typů. V případě přidávání složitějších záznamů, nebo nových typů sdílených proměnných je nutné navíc doplnit zobrazování proměnných. Tím se tato kapitola nezabývá. Pro názornost bude použit záznam trouba. Ten bude mít atributy zapnutí (boolean) a nastavenou teplotu (int).

4.3.1 Android Aplikace

Přidání nového typu záznamu do aplikace začíná vytvořením třídy pro záznam do balíčku data/shared_variables/record. V tomto případě je potřeba vytvořit třídu *SharedOvenRecord*, která bude dědit ze třídy *RecordVariable*. Zde je potřeba implementovat konstruktor. Ten bude obsahovat parametry jeho předka a také parametry inicializující jeho atributy. Atributy jsou u záznamů realizovány pomocí komponent, které jsou uloženy v seznamu. Konstruktor záznamu trouby by tedy mohl vypadat jako v zdrojovém kódu 12. Třída *SharedOvenRecord* bude dále implementovat get metody pro atributy třídy a metodu *getVariableType()*. Metoda bude vracet nový typ proměnné SHARED_RECOR_OVEN.

```
public SharedPersonRecord(int id, String name,
                          UserAccess readAccess,
                          UserAccess writeAccess,
                          boolean isOn, int temperature) {
    super(id, name, readAccess, writeAccess);
    components.add(new BooleanComponent(IS_ON, isOn));
    components.add(new IntComponent(TEMPERATURE, temperature));
}
```

Zdrojový kód 12: Ukázka konstruktoru záznamu trouby

Dalším krokem bude implementace serializace a deserializace proměnné. K těmto úkonům dojde prostřednictvím třídy *Gson*. Pro deserializaci bude nutné přidat *case* scénář pro nový typ proměnné a to do metody *getVariable(JSONObject variableJson)*. Serializace bude provedena vytvořením nové statické metody *setOvenRecordJson(int id, SharedOvenRecord value)*. Předlohou pro implementaci této metody může být například metoda *setPersonRecordJson(int id, SharedPersonRecord value)*, která plní stejnou funkci pro záznam osoby. Tuto metodu budeme volat v třídě *BluetoothMessagingService* v metodě *setRecordVariable(int id, RecordVariable variable)*, kam přidáme nový *case* scénář pro nový typ záznamu.

4.3.2 Arduino API

Pro přidání nového typu záznamu do API je nejprve potřeba vytvořit nový hlavičkový soubor. V našem případě jej pojmenujeme *SharedOvenRecord.h*. V něm definujeme třídu *SharedOvenRecord*, která bude dědit od třídy *SharedVariable*. Bude zde implementace konstruktoru, který bude obsahovat stejné parametry jako konstruktor jeho předka. Navíc k nim bude mít parametry *isOn* a *temperature*, teré budou nastavovat stejnojmenné privátní atributy. Pro tyto atributy budou vytvořeny také metody typu get a set. Dále bude implementována metoda *getType()*, která bude vracet nový typ SHARED_RECOR_OVEN. Ten bude vytvořen ve třídě *VariableType*. Budou pro něj rovněž přidány překlady na textový řetězec do metody*toString(Type type)* a z textového řetězce do metody *fromString(const String &type)*. Tím bude pokryto vytvoření nového záznamu.

Dalším krokem bude vytvoření funkcí pro serializaci a deserializaci. Ve třídě *Gson* vytvoříme dvě nové statické funkce. První z jich bude veřejná funkce *getOvenRecord(const String &jsonMessage)*. Ta převede zprávu ve formátu JSON na novou instanci záznamu trouby a vrátí ji. Funkce bude použita v třídě *BluetoothManager* v metodě *registerMessageActions* při registrování zpráv pro nastavování sdílených proměnných (typ SET_ VARIABLE). Hned v první konstrukci *switch* se přidá nový *case* pro nový typ záznamu. Jeho implementace je zachycena v zdrojovém kódu 13.

```
case VariableType::SHARED_RECORD_OVEN:
    SharedOvenRecord *oven =
                        ((SharedOvenRecord *) manager->get(id));
    SharedOvenRecord tmpoven = Gson::getOvenRecord(message);
    oven->setOn(tmpoven.isOn());
    oven->setTemperature(tmpoven.getTemperature());
    break;
```

Zdrojový kód 13: Ukázka použití funkce *getOvenRecord*

Druhou funkcí bude privátní funkce *createNestedOven(SharedOvenRecord *sharedVariable, JsonArray &context)*. Jejím cílem bude vytvořit JSON reprezentaci záznamu trouby do JSON pole sdílených proměnných *context*. Funkce bude použita ve funkci *getMessageAllVariables* podle vzoru ostatních typů sdílených proměnných. Kromě toho bude nutné do funkce *getCapacityForAllVariables* přidat výpočet velikosti sdílené trouby. V případě záznamu sdílené trouby budou do celkové kapacity připočítány délky znakových řetězců, které označují hodnoty jejích atributů.

Po dokončení těchto kroků bude záznam úspěšně přidán do Arduino API.

Kapitola 5

Testování

Tato kapitola se zabývá testováním aplikace. Jelikož se v aplikaci nenachází žádné algoritmické, či výpočetní problémy, tak nebyly použity Unit testy. Naopak, pro správné fungování aplikace a ověření funkčnosti komunikace mezi zařízeními, byly zapotřebí uživatelské testy. Provedené testovací scénáře jsou uvedeny v následujících podkapitolách.

5.1 Zobrazení zařízení

Pro použití tohoto scénáře je potřeba mít nainstalovanou aplikaci na mobilní zařízení.

Po spuštění aplikace dojde ke kontrole zapnutí Bluetooth. Pokud je Bluetooth vypnutý, zobrazí se možnost zapnout Bluetooth. V opačném případě jsou zobrazena spárovaná zařízení a je uživateli umožněno vyhledávání v okolí dostupných zařízení.

V momentě, kdy je aplikace zapnuta a uživatel vypne Bluetooth, jsou mu všechna zařízení skryta. Po opětovném zapnutí se uživateli obnoví seznam spárovaných zařízení a je rovněž umožněno vyhledávání zařízení.

Pokud je aplikace spuštěna a uživatel nemá zapnutý Bluetooth, tak je mu při vyhledávání v okolí dostupných zařízení zobrazena žádost o zapnutí Bluetooth. Při zapnutí započne vyhledávání zařízení a jejich zobrazení.

5.2 Připojení a odpojení

Tento testovací scénář vyžaduje nainstalovanou aplikaci na mobilní zařízení a využití API na zařízení Arduino.

Po kliknutí na tlačítko pro připojení k zařízení dojde o pokus o připojení. Při neúspěchu je uživatel informován. Pokud se podaří uživateli připojit, je přesměrován na přihlašovací formulář. Při stisknutí tlačítka zpět dojde k odpojení od zařízení. Tento postup lze libovolně opakovat.

5.3 Přihlášení a odhlášení

Tento testovací scénář vyžaduje nainstalovanou aplikaci na mobilní zařízení a využití API na zařízení Arduino. Dále je potřeba vytvořit instanci alespoň jednoho uživatele. Uživatel musí být přidán do instance třídy *UserManager*. Ta je nastavena v instanci třídy *BluetoothManager*, která je dále použita.

Po připojení k zařízení je uživateli nabídnut přihlašovací formulář. Po jeho vyplnění a odeslání dojde ke kontrole. V případě, že kombinace uživatelského jména a hesla není správná, je uživatel o tomto faktu informován. Když je kombinace správná, je uživatel přesměrován na obrazovku se sdílenými proměnnými. Odtud se tlačítkem zpět dostane zpět na přihlašovací formulář. Zmíněný postup lze libovolně opakovat.

5.4 Zobrazení proměnných

Tento testovací scénář vyžaduje nainstalovanou aplikaci na mobilní zařízení a využití API na zařízení Arduino. Dále je potřeba vytvořit instanci alespoň dvou různých uživatelů s různými typy oprávnění a dvě sdílené proměnné. Jejich oprávnění pro čtení se musí lišit. Je dobré, pokud jejich oprávnění pro čtení kopírují oprávnění uživatelů. Uživatelé a proměnné musejí být přidány do příslušných manažerů.

Po úspěšném přihlášení jsou uživateli zobrazeny proměnné, jejichž oprávnění pro čtení je menší nebo rovné oprávnění přihlášeného uživatele. Po odhlášení a opětovnému přihlášení k jinému uživatelskému účtu, je zobrazen seznam proměnných, který se liší od předchozího. Rozdílnost je zajištěna různými typy oprávnění uživatelů a přístupů proměnných ke čtení.

5.5 Úprava proměnných

Pro tento testovací scénář je potřeba nainstalovaná aplikace na mobilní zařízení a využití API na zařízení Arduino. Dalším krokem je vytvoření alespoň jednoho uživatele a dvou různých sdílených proměnných. Oprávnění proměnných pro zápis se musí lišit – jedno uživateli umožní zápis a druhé ne. Oprávnění pro čtení musí uživateli umožňovat zobrazení proměnných. Uživatel a proměnné musejí být přidány do příslušných manažerů, které jsou dále použity.

Po zobrazení proměnných jsou uživateli u jednotlivých proměnných zobrazeny ikony pro úpravu. Pokud uživatel nemá k dané proměnné oprávnění pro zápis, tak je ikona skryta. Kliknutím na ikonu se uživateli zobrazí formulář pro úpravu hodnoty proměnné. Po nastavení proměnné a jejím odeslání na server dojde ke kontrole. Pokud dojde k chybě, je o tom uživatel informován. Při úspěchu dojde k nastavení proměnné na novou hodnotu a k jejímu zobrazení.

Odhlášením a opětovným přihlášením k uživatelskému účtu můžeme ověřit, že došlo k úspěšnému uložení nové hodnoty proměnné.

5.6 Přerušení spojení

V tomto testovacím scénáři se testuje reakce aplikace na přerušení spojení se serverem. Pro tento testovací scénář je nutné mít nainstalovanou aplikaci na mobilní zařízení a použité API na Arduino desce.

Po úspěšném připojení nastává možnost přerušení spojení. To může nastat vypnutím Bluetooth na mobilním zařízení, či chybou na serveru – například ztrátou napájení desky. Při přerušení je aplikace přesměrována ze současné obrazovky na hlavní obrazovku s dostupnými zařízeními. Dále je uživatel informován o ztrátě spojení.

Závěr

Cílem bakalářské práce bylo vytvořit Android aplikaci pro sdílení proměnných s platformou Arduino prostřednictvím Bluetooth. Aplikace měla podporovat uživatelské účty s různými stupni oprávnění. Dále měla umožňovat zobrazování a úpravu proměnných podle uživatelských oprávnění. Dalším kritériem bylo zapojení periferií na desku Arduino.

Pro potřeby práce byla provedena rešerše zkoumající potřebné technologie a existující řešení problematiky. Pro tvorbu aplikace byly použity metody softwarového inženýrství – analýza, návrh, implementace a testování.

Výsledkem této bakalářské práce je aplikace pro operační systém Android. Aplikace podporuje více uživatelských účtů s různými typy oprávnění. Oprávnění mají tři stupně (ROOT, REGULAR, GUEST), ale je zde možnost je rozšířit o libovolné další mezistupně. Uživatelé mají své jméno a heslo.

Aplikace umožňuje zobrazování a úpravu proměnných uložených na desce Arduino dle oprávnění uživatele. Proměnné mohou být také pouze pro čtení. Sdílené proměnné můžeme rozdělit na tři kategorie – jednoduché proměnné, proměnné s polem hodnot a záznamy. Jednoduché proměnné obsahují pouze jednu hodnotu. Záznamy mají pouze jeden stupeň zanoření. Rozsah datových typů hodnot v proměnných je ve všech třech kategoriích stejný. Je možné ukládat čísla (int, long, float), pravdivostní hodnotu (bool), znaky (char) a znakové řetězce (String).

Druhou částí práce je API pro Arduino desku. Použitím API je aplikaci umožněno komunikovat se deskou Arduino. Realizovaná komunikace je pomocí technologie Bluetooth v datovém formátu JSON.

Poslední realizovanou částí práce je použití periferií – LED diody, klávesnice, displeje a GSM modulu. Všechny zmíněné periferie byly použity. Dále bylo zdokumentováno jejich zapojení a příklad jejich použití.

Práci by bylo možné rozšířit o nové typy uživatelských oprávnění a proměnných, zejména víceúrovňových záznamů. Výsledek práce je však nosným základem, který se dá použít pro tvorbu vlastních chytrých zařízení.

Bibliografie

1. PAVLÍČEK, Jaroslav. Statistiky Android, iOS a modely telefonů v půlce roku 2021. In: *Inited.cz* [online]. INITED Solutions s. r. o., 2021 [cit. 2021-11-18]. Dostupné z: `https://inited.cz/2021/07/23/statistiky-android-ios-modely-2021/`.
2. BRODKIN, Jon. Google and Samsung soar into list of top 10 Linux contributors. In: *ArsTechnica.com* [online]. WIRED Media Group, 2013 [cit. 2021-11-18]. Dostupné z: `https://arstechnica.com/information-technology/2013/09/google-and-samsung-soar-into-list-of-top-10-linux-contributors/`.
3. KILIÁN, Karel. Historie Androidu v kostce aneb Od verze 1.0 až po Android M. In: *SvetAndroida.cz* [online]. SvetAndroida.cz, 2015 [cit. 2021-11-18]. Dostupné z: `https://www.svetandroida.cz/historie-androidu/`.
4. BROWN, C. Scott. Android 12 release date: When can you expect it on your phone?: It will be a bit longer before we see it on non-Google phones. In: *AndroidAuthority.com* [online]. Authority Media, 2021 [cit. 2021-11-18]. Dostupné z: `https://www.androidauthority.com/android-12-release-date-1202179/`.
5. GOOGLE INC. Meet Android Studio? In: *Developer.android.com* [online]. Google Inc., 2021 [cit. 2021-11-29]. Dostupné z: `https://developer.android.com/studio/intro`.
6. GOOGLE INC. Android Debug Bridge (adb). In: *Developer.android.com* [online]. Google Inc., 2021 [cit. 2021-11-29]. Dostupné z: `https://developer.android.com/studio/command-line/adb`.
7. ARDUINO.CC. What is Arduino? In: *Arduino.cc* [online]. Arduino.cc, 2020 [cit. 2021-11-19]. Dostupné z: `https://www.arduino.cc/en/guide/introduction#`.

8. VODA, Zbyšek; HW KITCHEN. Průvodce světem Arduina. In: *RobotikaBrno.cz* [online]. Robotika Brno, 2014, kap. 1-2 [cit. 2021-11-19]. Dostupné z: `https://www.robotikabrno.cz/docs/arduino/Pr%C5%AFvodce-sv%C4%9Btem-Arduina-CZ.pdf`.

9. LASKA, Konstantin. Krátký přehled Arduino desek. In: *Blog.laskarduino.cz* [online]. laskarduino.cz, 2016 [cit. 2021-11-22]. Dostupné z: `https://blog.laskarduino.cz/kratky-prehled-arduino-desek/`.

10. ARDUINO.CC. Arduino Software (IDE). In: *Arduino.cc* [online]. Arduino.cc, 2018 [cit. 2021-11-29]. Dostupné z: `https://www.arduino.cc/en/Guide/Environment`.

11. JETBRAINS S.R.O. CLion. In: *Jetbrains.com* [online]. JetBrains s.r.o., [n.d.] [cit. 2021-11-29]. Dostupné z: `https://www.jetbrains.com/clion/`.

12. PLATFORMIO.ORG. What is PlatformIO? In: *Docs.platformio.org* [online]. Platformio.org, 2014 [cit. 2021-11-29]. Dostupné z: `https://docs.platformio.org/en/latest/what-is-platformio.html`.

13. KROMPOLC, Tomáš. Bluetooth je tu s námi 20 let: vše, co o této technologii potřebujete vědět. In: *Smartmania.cz* [online]. SMARTmania s.r.o., 2019 [cit. 2021-11-19]. Dostupné z: `https://smartmania.cz/bluetooth-je-tu-s-nami-20-let-vse-co-o-teto-technologii-potrebujete-vedet/`.

14. KOVAŘÍK, David. Bluetooth – modrozub pod drobnohledem (vědecké okénko). In: *Mobilizujeme.cz* [online]. Mobilizujeme.cz, 2011 [cit. 2021-11-22]. Dostupné z: `https://mobilizujeme.cz/clanky/bluetooth-modrozub-pod-drobnohledem-vedecke-okenko`.

15. BARRY, Peter; CROWLEY, Patrick. *Modern Embeded Computing*. Waltham, MA: Morgan Kaufmann, 2012. ISBN 978-0-12-391490-3.

16. BLUETOOTH SIG, INC. Bluetooth® Wireless Technology. In: *Bluetooth.com* [online]. Bluetooth SIG, Inc., 2021 [cit. 2021-11-19]. Dostupné z: `https://www.bluetooth.com/learn-about-bluetooth/tech-overview/`.

17. JSON.ORG. Introducing JSON. In: *Json.org* [online]. Json.org, [n.d.] [cit. 2021-11-19]. Dostupné z: `https://www.json.org/json-en.html`.

18. GOOGLE INC. JSONObject. In: *Developer.android.com* [online]. Google Inc., 2021 [cit. 2021-11-29]. Dostupné z: `https://developer.android.com/reference/org/json/JSONObject`.

19. ARDUINOJSON.ORG. ArduinoJson. In: *ArduinoJson.org* [online]. ArduinoJson.org, [n.d.] [cit. 2021-11-29]. Dostupné z: `https://arduinojson.org/`.

20. MORICH, Kai. *Serial Bluetooth Terminal* [online]. 2021. Ver. 1.36 [cit. 2021-11-24]. Dostupné z: `https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&hl=cs&gl=US`. Pro verze Android 4.3 a vyšší.

21. MIGHTYIT. *Bluetooth Terminal HC-05* [online]. 2019. Ver. 1.20 [cit. 2021-11-24]. Dostupné z: `https://play.google.com/store/apps/details?id=project.bluetoothterminal&hl=cs&gl=US`. Pro verze Android 4.0.3 a vyšší.

22. FILAN, Juraj. *Platforma pro komunikace mezi Android a Arduino*. Praha, 2019. Bakalářská práce. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství.

23. ŠVEC, Michal. *Meteorologická stanice v chytré domácnosti založená na platformě Arduino*. Praha, 2019. Bakalářská práce. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství.

24. ŠMÍD, Filip. *Knihovna funkcí pro OS Android umožňující řízení vývojového kitu Arduino*. Praha, 2017. Bakalářská práce. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství.

25. DENISSYUK, Andrey. *Aplikace android pro komunikaci s platformou Arduino protokolem Bluetooth*. Praha, 2021. Bakalářská práce. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství.

26. In: *Dratek.cz* [online]. ECLIPSERA s.r.o., [n.d.] [cit. 2021-12-01]. Dostupné z: `https://dratek.cz/arduino/7569-bezdratovy-gsm-gprs-modul-sim800l-v2.0.html`.

27. In: *Dratek.cz* [online]. ECLIPSERA s.r.o., [n.d.] [cit. 2021-12-01]. Dostupné z: `https://dratek.cz/arduino/1403-ky-016-rgb-led-modul-3-barvy-pro-arduino-avr-pic-raspberry.html`.

28. In: *Dratek.cz* [online]. ECLIPSERA s.r.o., [n.d.] [cit. 2021-12-01]. Dostupné z: `https://dratek.cz/arduino/1978-eses-0-96-i2c-oled-display-pro-jednodeskove-pocitace.html`.

29. In: *Dratek.cz* [online]. ECLIPSERA s.r.o., [n.d.] [cit. 2021-12-01]. Dostupné z: `https://dratek.cz/arduino/1006-klavesnice-membranova-1x4-pro-arduino.html`.

30. ARDUINO.CC. Language Reference: Variables. In: *Arduino.cc* [online]. Arduino.cc, 2021 [cit. 2021-12-04]. Dostupné z: `https://www.arduino.cc/reference/en/`.

31. W3SCHOOLS. Bluetooth® Wireless Technology. In: *W3schools.com* [online]. W3schools.com, 2021 [cit. 2021-12-04]. Dostupné z: `https://www.w3schools.com/java/java_data_types.asp`.

32. SPARX SYSTEMS PTY LTD. *Enterprise Architect* [online]. 2018. Ver. 14 [cit. 2021-12-29]. Dostupné z: https://www.sparxsystems.com/products/ea/14/index.html.

33. APP.UIZARD.IO. In: *App.uizard.io* [online]. app.uizard.io, [n.d.] [cit. 2021-12-21]. Dostupné z: https://app.uizard.io.

34. GOOGLE INC. Manifest.permission. In: *Developer.android.com* [online]. Google Inc., 2021 [cit. 2021-12-28]. Dostupné z: https://developer.android.com/reference/org/json/JSONObject.

35. GOOGLE INC. *Guava* [online]. 2021. Ver. 31.0.1 [cit. 2021-12-25]. Dostupné z: https://github.com/google/guava.

36. FRITZING GMBH. *Fritzing* [online]. 2021. Ver. 0.9.8 [cit. 2021-12-29]. Dostupné z: https://fritzing.org/download/.

37. MPFLAGA. *Arduino-MemoryFree* [online]. 2021 [cit. 2021-12-28]. Dostupné z: https://github.com/mpflaga/Arduino-MemoryFree.

38. OLIKRAUS. *U8glib* [online]. 2021 [cit. 2021-12-28]. Dostupné z: https://github.com/olikraus/u8glib.

39. Send Receive SMS & Call with SIM800L GSM Module & Arduino. In: *Lastminuteengineers.com* [online]. Lastminuteengineers.com, 2021 [cit. 2021-12-29]. Dostupné z: https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial/.

40. MIŘEJOVSKÝ, Jakub. SIM800L restart automatically. In: *Stackexchange.com* [online]. Stackoverflow.com, 2019 [cit. 2021-12-29]. Dostupné z: https://electronics.stackexchange.com/questions/361924/sim800l-restart-automatically.

41. MILIOHM.COM. SIM800L V2 tutorial with arduino. In: *Miliohm.com* [online]. Miliohm.com, 2020 [cit. 2021-12-29]. Dostupné z: https://miliohm.com/sim800l-v2-tutorial-with-arduino/.

42. SIMCOM. SIM800 Series AT Command Manual. In: *Elecrow.com* [online]. Elecrow.com, 2015 [cit. 2021-12-29]. Dostupné z: https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf.

Příloha A

Seznam použitých zkratek

BLE Bluetooth Low Energy

OS Operační systém

API Application Programming Interface

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `android` ..... aplikace Android
  - `app.apk` ..... instalovatelná aplikace
  - `arduinoremotecontroller` ..... Android Studio projekt
  - `doc` ..... dokumentace aplikace
- `arduino` ..... Arduino API
  - `arduinoserver` ..... CLion projekt
  - `doc` ..... dokumentace API
- `text` ..... text bakalářeské práce
  - `BP_Jokl_Natanael_2022.pdf` ..... PDF bakalářské práce