Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Android aplikace na ovládání auta

Martin Škorník

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Martin Škorník

Informatika

Název práce

Android aplikace na dálkové ovládání auta

Název anglicky

Android application for remote car control

Cíle práce

Cílem práce je naprogramovat aplikaci na dálkové ovládání auta pro operační systém Android za pomocí jazyku Java. Jako dílčí cíl je naprogramování firmwaru pro auto s řídící jednotkou Arduino. Aplikace by dále měla zobrazit uživateli informace ze senzorů umístěném na autě. Komunikace mezi aplikací a Arduinem bude probíhat bezdrátově pomocí Bluetooth.

Metodika

Diplomová práce se bude dělit na část teoretickou a část praktickou.

V teoretické části představím způsoby programování Android aplikace ve vývojovém prostředí AndroidStudio s využitím jazyku Java. V teoretické části vysvětlím implementaci programování firmwaru pro jednotku Arduino.

V praktické části se budu zabývat vývojem Android aplikace virtuálního ovladače v prostředí AndroidStudio s možností zobrazení informací ze senzorů.

Diplomovou práci bych ukončil porovnáním výsledné aplikace s podobnými aplikaci, které jsou k dispozici.

Doporučený rozsah práce

50-60 stran

Klíčová slova

Bluetooth o, Android Studio, Arduino, dálkové ovládání, Java

Doporučené zdroje informací

Android Developers. Android Developers [online]. Dostupné z: https://developer.android.com/
HEROUT, Pavel. Učebnice jazyka Java. 5., rozš. vyd. České Budějovice: Kopp, 2010. ISBN 9788072323982.
LACKO, Ľuboslav. Vývoj aplikací pro Android. Brno: Computer Press, 2015. ISBN 9788025143476.
MONK, Simon. Arduino + Android projects for the evil genius: control Arduino with your smartphone or tablet. New York: McGraw-Hill, c2012. ISBN 978-0071775960.
VODA, Zbyšek. Průvodce světem Arduina. Vydání druhé. Bučovice: Martin Stříž, 2017. ISBN 9788087106938.

Předběžný termín obhajoby

2018/19 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 26. 3. 2019

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 26. 3. 2019

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 10. 11. 2019

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Android aplikace na ovládání auta" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 20.11.2019 _____

Poděkování

Rád bych touto cestou poděkoval svému vedoucímu diplomové práce Ing. Marku Píckovi, Ph.D., za odborné vedení při vypracování této diplomové práce.

Dále bych chtěl poděkovat rodičům za výraznou podporu, a to nejen během celého studia.

Android aplikace na ovládání auta

Abstrakt

Tato diplomová práce je zaměřena na vytvoření aplikace pro bezdrátové ovládání auta za pomocí mobilního zařízení s operačním systémem Android.

V teoretické části se práce zabývá základy programování mobilních aplikací a seznámí čtenáře s prací ve vývojovém prostředí Android studio s použitím jazyka Java. V další části se dále zaměřím na vysvětlení mikrořadiče Arduino, který zde slouží jako řídící jednotka auta včetně způsobů programování firmwaru ve vývojovém prostředí Arduino IDE.

V praktické části aplikuji poznatky z teoretické části pro vývoj virtuálního mobilního ovladače. Praktickou část začnu analýzou požadavků pro výslednou aplikaci. Na základě těchto požadavků provedu návrh aplikace a v části realizace provedu naprogramování aplikace, která bude vysílat signály řídící jednotce auta a vlastního firmwaru řídící jednotky pro příjem těchto signálů a zpětné posílání souřadnic auta.

Klíčová slova: Virtuální joystick, auto, Android aplikace, Android Studio, Java, mikrořadiče, Arduino

Android application for car control

Abstract

This diploma thesis is focused on developing application for wireless car controlling with help Android mobile device.

The theoretical part deals basics of programming mobile application and inform readers about a developing IDE Android studio. In next chapter I will focus on describe microcontroller Arduino, which in car is used as control unit and describe programming of firmware in Arduino IDE.

The practical part I applied the information from the theoretical part for the development of the mobile virtual controller. First, I start by analysis of requirements for final application. Based on these requirements, I will make the application design and in the implementation part, I will develop application for control Arduino car with a firmware for receiving instruction from a mobile device and sending back GPS car coordinates.

Keywords: Virtual joystick, car, Android application, Android Studio, Java, microcontroller, Arduino

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Seznam použitých zkratek

IDE	Integrated Development Environment
RC	Remote Control
RUP	Rational Unified Process
SDK	Software Development Kit
JDK	Java Development Kit
UML	Unified Modeling Language
UP	Unified Process

1 Úvod

Tématem této práce je naprogramování aplikace, která by byla schopna dálkově ovládat auto za pomocí mobilního zařízení se systémem Android.

Pro toto téma jsem se rozhodl z důvodu spojení dvou dnes oblíbených trendů, a to rekreační či závodní jízda s různými modely aut a snahou ovládat cokoliv na dálku za pomocí mobilního telefonu či tabletu.

Dneska existuje spousta sad, díky kterým si člověk může sestavit vlastní RC auto, ale k těmto sadám si musí uživatel ještě dokoupit pro ovládání rádiový vysílač, který je většinou i několikanásobně dražší než samotná sada. Pokud by chtěl budoucí řidič koupit spíše už hotový produkt, nevyhne se tímto i drahé investici několika tisíců korun. Aby se mohlo ušetřit a vynechat rádiový vysílač, existuje možnost auto ovládat pomocí mobilního telefonu, ale to uživateli vezme možnost sestavení vlastního modelu, neboť je aplikace pro ovládání často přizpůsobena pro modely určitého výrobce, bez možnosti použití dané aplikace pro modely jiného výrobce a bez možnosti si danou aplikaci upravit, neboť se tyto aplikace šíří s uzavřeným kódem.

Tuto nevýhodu lze odstranit s použitím různých mikrořadičů. Pomocí nich si vytvořit vlastní aplikaci s otevřeným kódem společně s vlastním firmwarem.

Diplomovou práci bych chtěl zaměřit hlavně pro ty, kteří by si chtěli sestavit vlastního auto za použití mikrořadiče Arduino, který je ve světě mikrořadičů ten nejrozšířenější a ovládat ho za použití systému Android jakožto nejrozšířenější mobilní platformy.

Teoretická část práce se zprvu věnuje postupu vývoje aplikací a jsou zde zobrazeny základní metodiky vývoje softwaru včetně úvodu do modelovacího jazyka UML. Tato část se dále zabývá programováním aplikace pro operační systém Android ve vývojovém prostředí Android Studio. Další část se zaměřuje na mikrořadič Arduino společně s programováním v nástroji Arduino IDE.

Praktická část práce obsahuje kapitoly s jednotlivými fázemi vývoje konečné aplikace pro dálkové ovládání auta s vlastním firmwarem.

Výsledná aplikace by měla být schopná pomocí Bluetooth dálkově ovládat auto a zobrazit uživateli souřadnice modelu včetně jejich zobrazení na mapě.

2 Cíl práce a metodika

2.1 Cíl práce

Jako hlavní cíl je vytvoření virtuálního ovladače pro řízení auta.

Tohoto nelze dosáhnou bez splnění těchto dílčích cílů:

- Sestavení auta s použitím mikrořadiče Arduino.
- Naprogramování ovladače ve vývojové prostředí Android studio a jazyka Java.
- Naprogramování Arduino firmwaru pro komunikaci s virtuálním ovladačem a pro řízení auta v prostředí Arduino IDE.

2.2 Metodika

Prvním krokem pro vytvoření aplikace je nastudování potřebné literatury, konkrétně obecný postup vývoje aplikací, postup vytváření aplikací pro operační systém Android za použití jazyka Java s vývojovým prostředím Android Studio a programování firmwaru mikrořadiče Arduino za pomocí nástroje Arduino IDE a jazyka C.

Pro splnění hlavního cíle je třeba nejdříve určit požadavky nové aplikace a ty rozdělit na požadavky funkční a nefunkční. Ze získaných požadavků za pomocí jazyka UML vytvořit diagramy tříd, užití a sekvenční, které poslouží k zjednodušení návrhu nové aplikace.

Z výsledných diagramů vytvořit návrh virtuálního ovladače využívající ke komunikaci s mikrořadičem Arduino bezdrátovou technologii Bluetooth. Po dokončení návrhu naprogramování samotného ovladače pro zařízení Android pomocí jazyka JAVA a firmwaru řídící jednotky Arduino. Na závěr naprogramovanou aplikaci společně s firmwarem otestovat, otestovanou aplikaci porovnat s obdobnou aplikací, která je již k dispozici.

3 Teoretická východiska

V této kapitole je provedena literární rešerše a její poznatky jsou dále použity v praktické části.

3.1 Fáze vývoje softwaru

Během vývoje software přechází do různých fází. Mezi tyto fáze řadíme analýzu požadavků, návrh systému, implementaci, testovaní, zavedení a údržbu.[1]

3.1.1 Analýza požadavků

Tato fáze se zabývá analýzou toho, jakých cílů má aplikace dosáhnout. Požadavky můžeme shrnout do dvou kategorií, funkční a nefunkční požadavky. Funkční požadavky určují hlavní cíle aplikace a vývojář by jim měl věnovat největší prioritu. Nefunkční požadavky jsou požadavky vedlejší, které se věnují spíše způsobu naprogramování aplikace, výběrem programovacího jazyka, vývojového prostředí, nebo na požadavky z pohledu uživatele, například líbivý design a jednoduché ovládání.[2]

3.1.2 Návrh systému

Vychází se zde z již získaných požadavků a úkolem je určení jednotlivých následujících postupů budoucího vývoje, včetně rozmístění požadavků do jednotlivých obrazovek aplikace. Jsou zde definovány i základy designu ve formě wireframů, či vytvořený model tříd obsahující jednotlivé objekty s vazbami mezi nimi.[3]

3.1.3 Realizace

V této fázi nastává tvorba samotné aplikace již v určeném vývojovém prostředí za pomocí odsouhlaseného programovacího jazyka. Během této fáze se pracuje také na projektové dokumentaci.[4]

[1] BUCHALCEVOVÁ, A. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky.* 1. vyd. Praha: Grada, 2005. 163 s. ISBN 978-80-247-1075-7.
[2] Tamtéž
[3] Tamtéž
[4] Tamtéž

3.1.4 Testování

Než se výsledná aplikace odevzdá zákazníkovi, je třeba ji nejdříve otestovat neboli zjistit, zda aplikace splňuje všechny požadavky zákazníka a odpovídá návrhu. Testují se zde buď jednotlivé funkcionality zvlášť, nebo se aplikace testuje jako celek. V rámci testování se doporučuje mít aplikaci izolovanou od ostatních systému, aby se zamezilo případné kolizi s ostrými daty.[5]

3.1.5 Zavedení a údržba

Zavedením softwaru se myslí odevzdání již otestované aplikace společně s dokumentací zákazníkovi. Tato fáze obsahuje také případné školení uživatelů. V rámci údržby se provádí instalace nových verzí na základě uživatelské zpětné vazby. V této fázi je proto vedeno i sledování aplikace již v ostrém provozu pro co nejlepší optimalizaci.[6]

3.2 Metodiky vývoje softwaru

Jedná se o seznam procesů a postupů pro analýzu, návrh a řízení vývoje softwaru, jednotlivé fáze vývoje byly definovány výše. V diplomové práci se provede výčet nejznámějších metodik vývoje softwaru.

3.2.1 Vodopádový model

U vodopádového modelu[7] platí, že do následující fáze vývoje software přejde až po ukončení fáze předchozí. Výhodou této metodiky je snadné pochopení, jelikož každý člen realizačního týmu ví dopředu, co a ve které fázi má dělat. Zásadní nevýhodou této metodiky je nepřítomnost zákazníka během samotného vývoje, jelikož jeho úloha končí již v první fázi, kde se ještě neprogramuje samotný software. Nevýhoda této metody spočívá v pomalé reakci na změny zákazníkových požadavků. Z důvodů jednoduchosti se tato metodika používá u jednoduchých projektů, kde dosahuje vysoké efektivnosti.

---

[5] BUCHALCEVOVÁ, A. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky.* 1. vyd. Praha: Grada, 2005. 163 s. ISBN 978-80-247-1075-7.

[6] Tamtéž

[7] Metodiky vývoje softwaru. *Moravská vysoká škola Olomouc.* [online]. [cit. 10.02.2019]. Dostupné z: https://mvso.cz/wp-content/uploads/2018/02/Metodiky-v%c3%bdvoje-software-studijn%c3%ad-text.pdf

3.2.2 Spirálový model

Spirálový model[8] vychází z vodopádového modelu, ale na rozdíl od něj se systém nejdříve rozdělí na dílčí části. V nich se uplatňuje podobný postup jako u předchozího modelu, ale než se přejde do další fáze, dochází k analýze rizik. Tak je možné vyhnout se v následující fázi předem určeným rizikovým situacím. Toto se opakuje pro každou dílčí část a z tohoto důvodu získala tato metodika svůj název.

3.2.3 RUP

Tuto metodiku vyvinula společnost IBM. Jedná se o šablonu obsahující systematický přístup k vývoji softwaru. V rámci uplatnění této metodiky používáme detailní návody a postupy pro plnění cílů, kvality, termínů a konečného rozpočtu. Tato metodika je určena pro větší a náročnější projekty. Z toho plynou dvě nevýhody, jednak to, že není k dispozici zdarma, jednak je třeba počítat s delším časem na studium všech potřebných materiálů. Pro menší projekty je možné využít metodiky UP, která je oproti metodice RUP zdarma. Výhodu této metodiky je rychlejší odhalení rizik a snadnější reakce na změny.[9]

3.2.4 Agilní manifest

Jedná se o principy[10], kterými se mají vývojáři řídit, aby mohli vyvinout ten nejlepší software. Těmito principy se řídí takzvané agilní metodiky.[11]

Agilní metodiky vesměs kopírují jednotlivé fáze vývoje softwaru, ale dochází zde k jejím přizpůsobením. U agilního vývoje se předem počítá se zapojením zákazníka do všech fází vývoje. Oproti tradičním metodikám se vývojáři snaží vyvíjet software co možná nejrychleji a zákazníkovi předávat software i nedokončený. Program postupně dokončují za provozu na základě zpětné vazby zákazníka. Výhodou a zároveň nevýhodou těchto metodik je samotný zákazník. Na jedné straně tím, že je vtažen do vývoje, je velká pravděpodobnost, že zákazník dostane přesně to, co vyžaduje. Na druhé straně však můžete narazit na

[8] Metodiky vývoje softwaru. *Moravská vysoká škola Olomouc.* [online]. [cit. 10.02.2019]. Dostupné z: https://mvso.cz/wp-content/uploads/2018/02/Metodiky-v%c3%bdvoje-software-studijn%c3%ad-text.pdf
[9] Tamtéž
[10] Manifest Agilního vývoje software. *Agilemanifesto.* [online]. [cit. 10.02.2019]. Dostupné z: http://agilemanifesto.org/iso/cs/manifesto.html
[11] Agilní vývoj není (jenom) Scrum a Kanbanl. *Jiří Knesl.* [online]. [cit. 10.02.2019]. Dostupné z: http://www.knesl.com/agilni-vyvoj-neni-jen-scrum-nebo-kanban

zákazníka, který nemá dostatek času se vývoji věnovat, anebo nemá jasnou představu, co by měl výsledný software všechno obsahovat.

Do nejznámějších metodik agilního vývoje patří Scrum, Extrémní programování a Test-Driven Development.

3.2.4.1 Scrum

Pro Scrum[12] jsou typické malé týmy vývojářů obvykle do deseti lidí, kteří jsou dobře sehraní a vědí, co mohou od sebe očekávat. Vývoj softwaru je rozdělen na kratší úseky, takzvané sprinty, kde je dán seznam úkonů a předem stanoven čas. Délka jednoho sprintu je obvykle čtrnáct dní. Během této doby se jednotliví členové vývojového týmu scházejí v tzv. „skrumážích", kde diskutují o tom, co se vše stihlo během předchozího dne a co je třeba ještě v rámci daného sprintu stihnout.

Hlavní výhodou této metodiky, jak bylo naznačeno výše, je tým vývojářů, který dokáže dotáhnout projekt do takového stavu, kdy zákazník dostane přesně to, co vyžaduje, a to do předem stanoveného termínu. Pokud však tým není dobře sladěn, může to přinést značné problémy, což lze označit za nevýhodu.

3.2.4.2 Extrémní programování

Tato metodika[13] vychází z jednotlivých vlastností agilního vývoje, které dotahuje do takzvaného extrému. U této metodiky platí udělat software co možná nejjednodušší a co nejrychleji daný program předat zákazníkovi, i po provedení menších úprav. U zákazníka se vyžaduje co možná nejaktivnější účast na vývoji a z toho důvodu mají zástupci zadavatelské společnosti vyhrazený prostor uvnitř společnosti realizující vývoj. V Extrémním programování se můžeme setkat podobně jako u Scrumu se sprinty, které na rozdíl od předchozí metodiky trvají i kratší dobu. Můžeme se zde také setkat s párovým programováním, kde u jednoho počítače pracují dva vývojáři, a zatímco jeden programuje, druhý mezitím kontroluje zdrojový kód a po nějaké době se vymění. Metodika obsahuje také Test-Driven Development.

[12] Co je Scrum? *Jiří Knesl.* [online]. [cit. 10.02.2019]. Dostupné z: http://www.knesl.com/co-je-scrum
[13] Extrémní programování (XP). *MBI - Management Byznys* [online]. [cit. 10.02.2019] Dostupné z: https://mbi.vse.cz/public/cs/obj/METHOD-92

3.2.4.3 Test-Driven Development

Tato metodika upřednostňuje zásadu psát testy ještě dříve, než se začne programovat. Zákazníkovi se tedy vždy předá odladěný software, ale v případě většího množství rozpracovaných testů hrozí nesplnění časového harmonogramu. Problémem této metodiky bývá i určení adekvátního množství testů.[14]

3.3 Jazyk UML

Jedná se o grafický jazyk sloužící k návrhu nové, či k vizualizaci již hotové aplikace, pro co možná nejjednodušší pochopení funkcionality a k znázornění obsahu aplikace nejen pro nové členy vývojového týmu.

UML se skládá ze dvou hlavních částí, diagramu struktury a diagramu chování. Diagram struktury udává strukturu systému, tj. z čeho se skládá. Diagram chování vysvětluje, jak systém pracuje a jak se využívá. Diagram chování navíc obsahuje diagramy interakcí sloužící k zachycení komunikace mezi jednotlivými součástmi systému.[15]

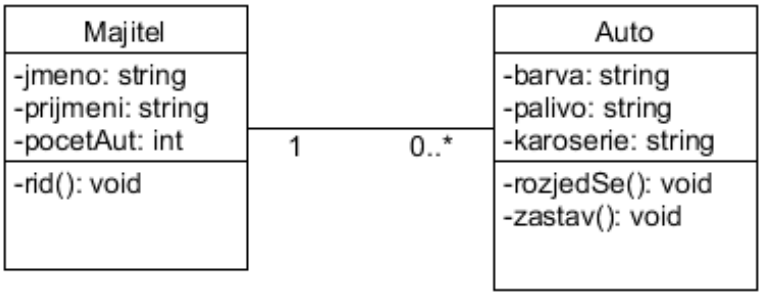
3.3.1 Diagram tříd

Patří mezi diagramy struktury a obsahuje statickou strukturu aplikace. V diagramu třídy reprezentují šablonu pro tvorbu objektů. Třída obsahuje atributy a metody, reprezentující vlastnosti a operace uvnitř dané třídy. Mezi třídami se mohou také objevovat vazby, mezi něž patří asociace, agregace a dědičnost. Při asociaci dochází pouze k výměně zpráv mezi třídami. Agregace nám určuje vazbu typu celek součást, kde je objekt dané třídy součástí jiného objektu. U dědění dochází k přenesení všech vlastností rodičovské třídy na její potomky.[16]

[14] Extrémní programování (XP). MBI - Management Byznys [online]. [cit. 10.02.2019] Dostupné z: https://mbi.vse.cz/public/cs/obj/METHOD-92

[15] Úvod do UML. *itnetwork.cz* [online]. [cit. 05.02.2019]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

[16] UML - Class diagram. *itnetwork.cz* [online]. [cit. 05.02.2019]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model



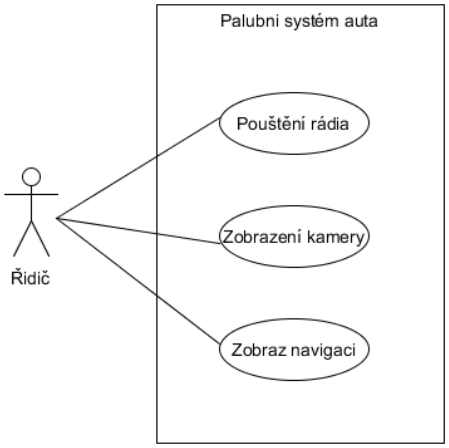
Obrázek 1: Diagram tříd

V Předchozím obrázku (Obrázek 1) je zobrazena základní struktura diagramu tříd, kde máme dvě třídy Auto a Majitel spojené mezi sebou vazbou asociace, násobnost u vazby nám zde udává, že majitel nemusí vlastnit automobil, nebo jich může vlastnit nekonečně mnoho, ale automobil musí patřit vždy jednomu majiteli. Každá třída začíná velkým počátečním písmenem, pod názvem třídy jsou definovány atributy definující vlastnosti dané třídy a pojmenovávají se podstatnými jmény. Konec diagramu patří metodám určující operace třídy, které se značí slovesem.[17]

3.3.2 Diagram užití

Patří do skupiny diagramu chování a jeho cílem je zobrazit, jak bude aktér (většinou uživatel) aplikaci využívat a co od ní má očekávat. Kromě aktéra diagram obsahuje již konkrétní případy užití ohraničené daným systémem. Jako příklad je uveden diagram (Obrázek 2), který obsahuje aktéra společně s činnostmi, které může provádět uvnitř palubního systému auta.[18]

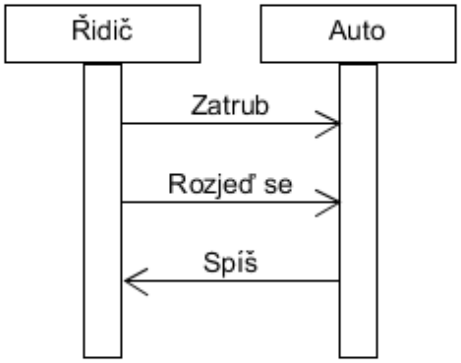
[17] UML - Class diagram. *itnetwork.cz* [online]. [cit. 05.02.2019]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model
[18] UML - Use Case Diagram. *itnetwork.cz* [online]. [cit. 05.02.2019]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram



Obrázek 2: Use Case diagram

3.3.3 Sekvenční diagram

Sekvenční diagram patří do skupiny diagramů interakcí. Zachycuje komunikaci v čase mezi jednotlivými objekty a je definován přesně pro jeden typ případu užití. V uvedeném příkladu (Obrázek 3) pro ovládání auta máme zachycenou komunikaci mezi řidičem a autem. Řidič přikazuje autu, aby se rozjelo, ale auto řidiče upozorní, že se řidiči již klíží oči, tudíž se nerozjede.[19]



Obrázek 3: Sekvenční diagram

3.3.3.1 Scénář

Scénář nám doplňuje sekvenční diagram ve snaze zachytit podrobně posloupnost jednotlivých zpráv. Jak diagram sekvenční, tak scénář se vztahuje vždy k jednomu případu užití.[20]

[19] UML - Sequence diagram. *itnetwork.cz* [online]. [cit. 05.02.2019]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-sequence-diagram
[20] Tamtéž

3.4 Operační systém Android

Operační systém Android byl vyvinut společností Google a je v současnosti nejrozšířenějším operačním systém pro mobilní telefony a tablety. Jedná se o operační systém převážně pro mobilní zařízení běžící na Linuxovém jádře. Díky tomu, že se jedná o systém s otevřeným kódem (open source), tudíž je zdarma k dispozici, může ho kdokoliv upravovat a nahrávat do vlastního mobilního zařízení. Toho často využívají výrobci těchto zařízení a vyvíjejí mnohé grafické nástavby pro odlišení od ostatních výrobců či pro přidání nových funkcí.

Pro uživatele existuje možnost instalovat si do svého zařízení různé aplikace z centra softwaru jménem Google Play, kde existuje ohromná spousta aplikací a her, které jsou k dispozici zdarma či za poplatek. Později zde přibyla hudební produkce, přehrávání filmů, čtení elektronických knih a podobně.

Se systémem Android se můžeme také setkat i v chytrých hodinkách, televizích, či coby systémem pro automobily. [21]

3.4.1 Historie systému Android

V roce 2003 vznikla společnost Android Inc., která začala vyvíjet operační systém se stejným jménem. V roce 2005 koupila tuto společnost firma Google, která se od té doby podílí na vývoji tohoto systému. V říjnu roku 2008 vznikl telefon T-Mobile G1 právě se systémem Android ve verzi 1.0 Apple Pie a zároveň byl zveřejněn balíček nástrojů pro vývojáře.[22]

3.4.2 Přehled jednotlivých verzí

Společnost Google vydává každý rok novou verzi systému, která obsahuje jak opravy chyb ze starých verzí, tak i přidané nové funkce. Každá nová verze systému se pojmenovává podle americké sladkosti v abecedním pořadím a příslušným číslem. Následující výčet udává historii jednotlivých verzí systému.[23]

[21] Android. *Mobilizujeme.cz* [online]. [cit. 20.12.2018]. Dostupné z: https://mobilizujeme.cz/stitky/android
[22] Android (operační systém). *Wikipedie* [online]. [cit. 20.12.2018]. Dostupné z: https://cs.wikipedia.org/wiki/Android_(operační_systém)
[23] Tamtéž

- **Android 1.0 Apple Pie –** První verze systému, obsahuje většinu aplikací, jaké známe doteď, jako například Google aplikace, webový prohlížeč, fotoaparát a podobně.
- **Android 1.1 Banana Bread –** Možnost přidávání příloh do zpráv či možnost zobrazení (skrytí) klávesnice.
- **1.5 Cupcake –** Přidána možnost nahrávání a zobrazení videa z kamery a jejich sdílení na YouTube, upravená klávesnice s možností našeptávání slov.
- **1.6 Donut** – Vylepšení centra softwaru jménem Android Market, došlo k přepracování vzhledu u defaultních aplikací a přidána možnost ovládání hlasem.
- **2.0/2.1 Eclair –** Nové požadavky na hardware z důvodu optimalizace chodu systému, úprava prohlížeče s přidáním podpory HTML 5. Nově podpora přisvětlovací diody k fotoaparátu včetně přidání digitálního zoomu.
- **2.2 Froyo –** Přidání možnosti instalování aplikací na paměťovou kartu. Možnost vytvoření Wi-Fi hotspotu pro sdílení mobilního internetového připojení. Zmodernizovaná aplikace fotoaparátu s více možnostmi nastavení focení.
- **2.3/2.4 Gingerbread –** Úprava softwarové klávesnice, aktualizace mapové aplikace s přidáním 3D zobrazení.
- **3.0/3.1/3.2 Honeycomb –** Nový vzhled systému, předělání webového prohlížeče a přizpůsobení systému pro tablety.
- **4.0–4.0.4 Ice Cream Sandwich –** Vylepšení rozpoznávání hlasu. Nově možnost odemknutí telefonu obličejem. Fotoaparát rozšířen o zobrazení panoramat.
- **4.1/4.2/4.3 Jelly Bean –** Přidána funkce Google Now coby virtuální asistentky jako alternativu k asistentce systému IOS společnosti Apple. Možnost přidání do mobilu další uživatelský účet a oddělení administrátorského a uživatelského účtu.
- **4.4 KitKat –** Optimalizace systému pro zařízení s menší pamětí.
- **5.0/5.1 Lollipop –** Nový vzhled systému společně se vzhledem notifikační lišty. Nově i vznik 64bitového systému. Vznik úsporného režimu pro prodloužení práce na jedno nabití.
- **6.0 Marshmallow –** Podpora USB typu C, odemykání systému za pomocí otisku prstu.
- **7.0/7.1 Nougat –** Skrytá funkce umožňující zvolit, které aplikaci má být přiděleno více výkonu. Upozornění na aplikace vytěžující baterii.

- **8.0/8.1 Oreo –** Vyšší optimalizace systému pro rychlejší start zařízení. Omezení aplikací běžících na pozadí a omezení geolokačních služeb.
- **9.0 Pie –** Zvětšení plochy pro upozornění s podrobnějším výpisem. Kompletní přepracování designu.

3.5 Programování Android aplikací

Na programování Android aplikací můžeme využít několik programových jazyků, nejčastěji se setkáme s jazykem Java či Kotlin, ale můžeme se setkat i s využitím jazyka C# či v experimentálním režimu i s jazykem Swift, který je k dispozici například ve vývojovém prostředí Scade.[24]

3.5.1 Řídící prvky Android aplikace

Každá Android aplikace se skládá jak z prvků, které ovlivňují její chod a pracují většinou na pozadí, tak z prvků ovlivňujících její design.[25] V následujícím výčtu jsou popsány jednotlivé řídící prvky aplikace.

3.5.1.1 Activity

Jedná se obrazovku uživatelského rozhraní, která existuje v aplikaci alespoň jednou. U activity je důležité spravovat a sledovat její stav pro správný běh aplikace.

Activita[26] během fungování aplikace může nabývat těchto stavů:

- **Creating** – stav pro vytvoření nové aktivity.
- **Starting** – do tohoto stavu se aktivita dostává během spuštění.
- **Running** – v tomto stavu se objevuje, když právě běží na popředí.
- **Paused** – stav, ve kterém je aktivita zastavena (poslána do pozadí) z důvodu běhu activity jiné.
- **Resume** – obnova aktivity po jejím zastavení (poslána do popředí).
- **Stopped** – v tomto stavu je zastavena a není k dispozici.
- **Destroyed** – stav, ve kterém je aktivita zničena a nedá se k ní vrátit.

[24] Application Fundamentals. *Android Developers* [online]. [cit. 20.12.2018]. Dostupné z: https://developer.android.com/guide/components/fundamentals
[25] Tamtéž
[26] Tamtéž

3.5.1.2 Service

Část programu, která existuje skrytě a pracuje bez ovlivňování uživatele. Service zajišťuje správu aplikace tehdy, kdy je aplikace překrytá aplikací jinou a zároveň není uživatelem ukončena. Například při přehrávání muziky na pozadí, kde uživatel ovládá jinou aplikaci, ale zároveň mu hraje hudba z hudebního přehrávače, kterého používal dříve, ale neukončil ho.[27]

3.5.1.3 Broadcast receiver

Používá se pro naslouchání systémových oznámení a zároveň dle potřeby na daná oznámení reaguje, například přechodem do úsporného režimu při nízké kapacity baterie.[28]

3.5.1.4 Content provider

Slouží ke sdílení dat mezi jednotlivými součástmi aplikace a jejími aktivitami. Pomocí Content providera se přistupuje k informacím i z jiných aplikací, pokud jim je udělen přístup, například zobrazení seznamu telefonních kontaktů v aplikaci emailu či aplikace pro VoIP telefonování typu Viber.[29]

3.5.2 Prvky ovlivňující design

3.5.2.1 Fragment

Fragment rozděluje uživatelské rozhraní na menší dílky, ve kterých mohou být spuštěné různé aktivity. Do jedné aktivity můžeme spustit více fragmentů a zároveň v jednom fragmentu může být spuštěno více aktivit. Příkladem využití tohoto prvku může být již zmiňovaný telefonní seznam, kdy by v jednom fragmentu byly zobrazeny kontakty a v druhém by byla zobrazena komunikace s kontaktem, na který předtím klikl uživatel.

3.5.2.2 View

Tento prvek je důležitý hlavně v uživatelském rozhraní aplikace. Definují se zde prvky vzhledu společně s jejími vlastnostmi.[30]

---

[27] Application Fundamentals. *Android Developers* [online]. [cit. 20.12.2018]. Dostupné z: https://developer.android.com/guide/components/fundamentals
[28] Tamtéž
[29] Tamtéž
[30] Tamtéž

3.5.2.3 Layout

Layout obsahuje seznam všech prvků View a reprezentuje jejich rozložení v rámci uživatelského rozhraní.[31]

3.5.2.4 Intent

Rozhoduje, co se zrovna má uživateli zobrazit a jaká operace se má právě udělat.[32]

3.5.2.5 Resources

Jsou zde uloženy všechny externí zdroje, které se použily v aplikaci.[33]

3.5.2.6 Manifest

Jedná se o konfigurační soubor, který se spustí při načítání aplikace a obsahuje informace o operacích, které se musí provést před zobrazením uživatelského rozhraní.[34]

3.6 Vývojová prostředí pro Android

Na naprogramování aplikace pro Android můžeme využít mnoho vývojových prostředí s různou podporou společností Google. Nejpoužívanějšími prostředími jsou Android studio a IntelliJ IDEA, které využívají jazyky Java a Kotlin. V historii se používala i prostředí Eclipse a NetBeans, ale pro tato prostředí se musely nahrát speciální doplňky, které již nejsou podporovány.[35]

3.6.1 Android studio

Jedná se o nejpoužívanější vývojové prostředí vyvinuté společností Google vycházející z IntelliJ IDEA, ze kterého dědí nejen design, ale většinu prvků.[36] Oproti IntelliJ IDEA lze Android studio naistalovat a využívat zdarma.

Android studio využívá tyto komponenty:

- Testování použivatelnosti pro koncová zařízení,
- podepisování aplikací,

[31] Application Fundamentals. *Android Developers* [online]. [cit. 20.12.2018]. Dostupné z: https://developer.android.com/guide/components/fundamentals
[32] Tamtéž
[33] Tamtéž
[34] Tamtéž
[35] Android Studio. *Wikipedie* [online]. [cit. 20.12.2018]. Dostupné z: https://cs.wikipedia.org/wiki/Android_Studio
[36] Tamtéž

- předpřipravené šablony designu, zde stačí jen vkládat zdrojový kód operací,
- propracovaný editor s možností navrhovat uživatelské rozhraní metodou drag-and-drop
- integrovaný emulátor prostředí Android,
- úprava uživatelského rozhraní za pomocí univerzálního XML souboru.

3.6.1.1 Proč používat Android studio?

- Lepé upravený našeptávač,
- lepší barevné rozlišování kódu,
- oficiální podpora společnosti Google,
- časté aktualizace,
- vysoká technická podpora, nejenom od společnosti Google,
- možnost zvolení tmavého pozadí,
- je k dispozici zdarma.[37]

3.6.1.2 Proč nepoužívat Android studio?

- Vysoká hardwarová náročnost pro vývojové zařízení (PC, notebook),
- vyvíjet prostředí může jenom společnost Google,
- prostředí je striktně nastaveno pouze pro programování aplikací na platformě Android,
- malá možnost využití externích modulů.[38]

3.6.2 IntelliJ IDEA

Jedná se o vývojové prostředí vyvinuté českou společností JetBrains. Jedná se o prostředí s cílem zajistit co nejvyšší programátorský komfort se snahou maximalizovat účinnost provedené práce a snaží se implementovat to, na co je vývojář zvyklý, aby nemusel měnit vývojová prostředí. Oproti Android studio je IntelliJ IDEA multi-jazykové prostředí, jde v něm programovat ve více jazycích než v Javě a Kotlinu (například v C++) a dále v něm lze programovat i pro jiné platformy než Android. Co se vlastností týče, je na tom stejně jako Android studio. Existuje jak v placené verzi, kde vývoj zajišťuje společnost JetBrains,

[37] Android Studio. *Wikipedie* [online]. [cit. 20.12.2018]. Dostupné z: https://cs.wikipedia.org/wiki/Android_Studio
[38] Tamtéž

tak i v komunitní verzi, která je zdarma a na vývoji může pracovat kdokoliv, i když novinky nejdříve přichází na placenou verzi.[39]

3.6.2.1 Proč používat IntelliJ IDEA?

- Příjemné uživatelské prostředí,
- je méně hardwarově náročné než Android studio,
- vyvíjen českou společností,
- vhodný i pro více programátorských jazyků,
- lze zde programovat i pro jinou platformu než Android.[40]

3.6.2.2 Proč nepoužívat IntelliJ IDEA?

- Jedná se o placený software, dá se částečně nahradit komunitní verzí,
- méně využívané než Android studio,
- nižší podpora ze strany společnosti Google,
- možné spekulace o odejmutí úplné Android podpory ve snaze prosadit přechod na Android studio.[41]

3.7 Nástroje nutné pro vytvoření prvního programu

Před vytvořením prvního programu je nejdříve nutné nainstalovat dodatečný software podle zvoleného vývojového prostředí a programátorského jazyka. U všech prostředí, lze tento software doinstalovat při prvním spuštění vývojového prostředí, ale lze si tento software doinstalovat předem ve verzi, která se bude aktuálně používat, neboť automaticky dojde k instalaci vždy nejnovější verze.[42]

3.7.1 Android SDK

Jedná se o sadu nástrojů umožňující vyvíjet aplikace pro dané prostředí. Každá verze systému Android používá svou vlastní SDK a vždy platí, že aplikace naprogramovaná ve staré verzi SDK je plně kompatibilní v nové verzi systému, ale aplikace naprogramovaná v nové verzi SDK není zpětně kompatibilní se starou verzí systému Android.

---

[39] IntelliJ IDEA. *Wikipedie* [online]. [cit. 20.12.2018]. Dostupné z: https://cs.wikipedia.org/wiki/IntelliJ_IDEA
[40] Tamtéž
[41] Tamtéž
[42] Projects overview. *Android Developers* [online]. [cit. 02. 01.2019]. Dostupné z: https://developer.android.com/studio/projects/

Správu SDK v Android studio má na starosti SDK Manager, kde se dají doinstalovat potřebné nástroje z SDK. Pokud se uživatel rozhodne vytvořit novou aplikaci, vývojové prostředí mu v roce 2019 nabídne SDK 28, které je přizpůsobené pro vývoj aplikací pro systém Android 9. Pie. Pokud se rozhodne vyvíjet i pro starší zařízení, lze si v SDK Manageru nainstalovat i starou verzi SDK, ale je zde podmínka, že nový vývojář může zvolit pro svůj projekt minimální verzi SDK s číslem 26 uzpůsobenou pro verzi Android 8. V případě zvolení nižší verze bude uživatel upozorněn, že tato aplikace nepůjde uložit do centra softwaru Google Play.[43]

3.7.2 Java Development Kit (JDK)

Sada nástrojů určená pro vývoj Java aplikací vyvíjená společnosti Oracle, na rozdíl od Android SDK funguje nejnovější verze JDK pro všechny verze systému Android.[44]

3.7.3 Gradle

Jedná se o nástroj pro automatické sestavení programu naprogramován v jazyce Groovy licencovaný jako open-source. Verze nástroje Gradle se určuje podle nainstalované verze SDK.[45]

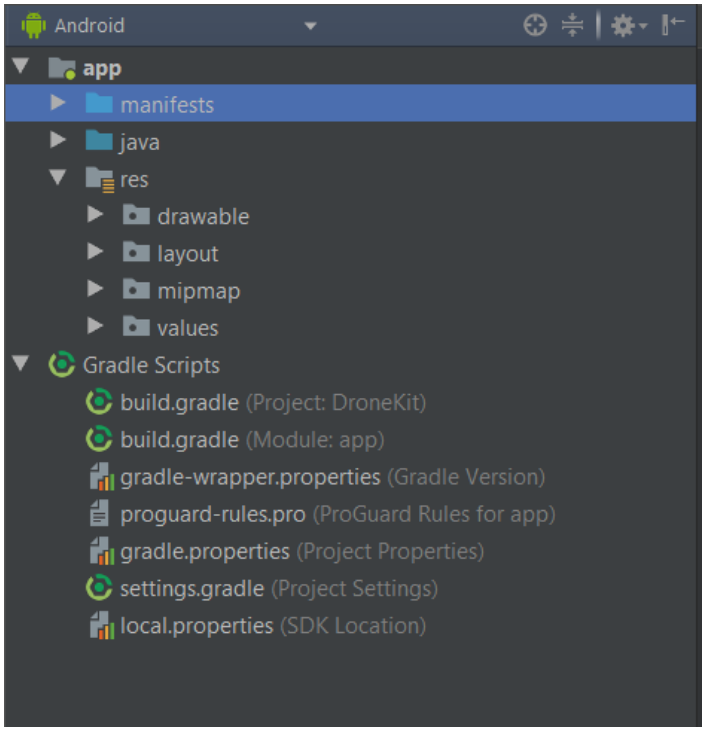
3.7.4 Emulátor

Nástroj na spuštění virtuálního prostředí systému Android na zařízení typu desktop. Využívá se ke spuštění vytvořené aplikace bez nutnosti ji předem nainstalovat na mobilním zařízení.[46]

3.8 Adresářová struktura Android programu

Každý program naprogramovaný v prostředí Android studio má následující adresářovou strukturu (Obrázek 4).[47] V následující části budou jednotlivé složky popsány.

[43] Projects overview. *Android Developers* [online]. [cit. 02. 01.2019]. Dostupné z: https://developer.android.com/studio/projects/
[44] Tamtéž
[45] Tamtéž
[46] Tamtéž
[47] Tamtéž



Obrázek 4: Adresářová struktura

3.8.1 Adresář manifests

Obsahuje soubor AndroidManifest.xml, v tomto souboru se nacházejí informace o aplikaci, které jsou zobrazeny systému před spuštěním zdrojového kódu aplikace. Definuje se zde jak samotný název aplikace, tak všechna oprávnění, která bude aplikace potřebovat.[48]

3.8.2 Adresář java

Obsahuje zdrojový kód aplikace napsaný v jazyce Java společně s logikou aplikace.[49]

3.8.3 Adresář res

Tento adresář obsahuje následující podsložky.

3.8.3.1 Podsložka drawable

Jsou zde uloženy bitmapové obrázky použité v aplikaci, například pro pozadí.[50]

[48] Projects overview. *Android Developers* [online]. [cit. 02. 01.2019]. Dostupné z: https://developer.android.com/studio/projects/
[49] Tamtéž
[50] Tamtéž

3.8.3.2 Podsložka layout

Nachází se zde soubory ve formátu XML obsahující celé uživatelské rozhraní aplikace, s možností využívat jeden layout v různých projektech.[51]

3.8.3.3 Podsložka minimap

Je zde uložena ikona programu, která se zobrazí v menu s aplikacemi uživatele v systému Android.[52]

3.8.3.4 Podsložka values

Jsou zde uložena barevná schémata, styly či typy písma použité v aplikaci.[53]

3.8.4 Složka Gradle Scripts

Obsahuje nástroj pro automatické sestavení programu Gradle společně s jeho konfiguračními soubory.[54]

3.9 Mikrořadič Arduino

S Arduinem[55] se můžeme setkat v nejrůznějších projektech, od různých hračkách pro „dospělé kluky“, až po 3D tiskárny a automatizované linky. Výhodou tohoto zařízení je nízká cena a vysoké množství příslušenství, které jde k Arduinu připojit. Příkladem mohou být různá měřící čidla využívající vstupní porty, zapojení motorů pro programování vlastního robota a jiných modulů využívající výstupní porty, s využitím modulů pro přenos bezdrátového signálu můžeme své projekty ovládat například pomocí mobilního telefonu.

Pro svou jednoduchost se stalo oblíbeným řešením pro výuku programování využívající programovacího jazyka podobnému jazyku C, nebo za použití externího příslušenství názorně ukázat výuku elektrotechniky (příkladem měření elektrických veličin). Nedostatkem Arduina na rozdíl od Raspberry Pi je ten, že Arduino nefunguje jako samostatný počítač, tudíž si na něm nespustíte operační systém Linux anebo nepodíváte na internet. Slouží převážně k řízení signálů od vstupu na výstup a naopak.

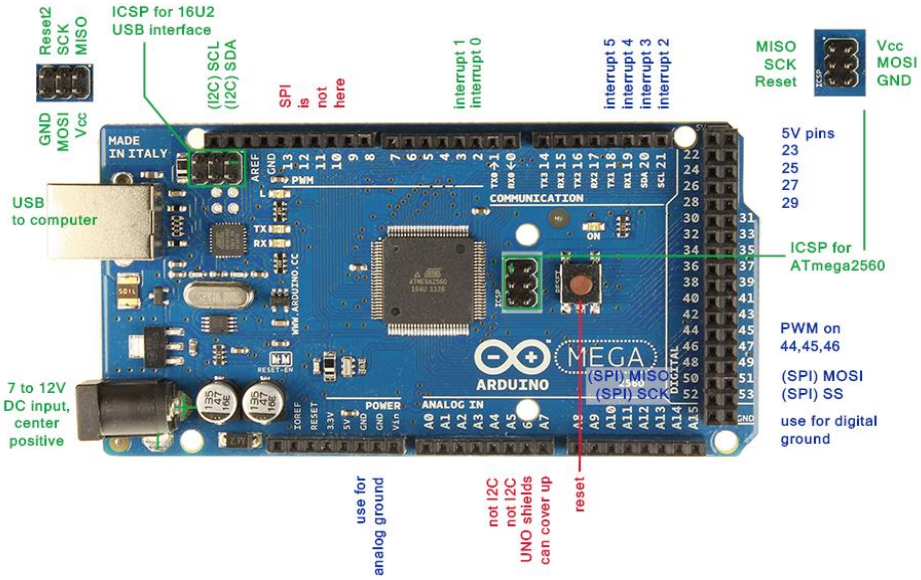
---

[51] Projects overview. *Android Developers* [online]. [cit. 02. 01.2019]. Dostupné z: https://developer.android.com/studio/projects/
[52] Tamtéž
[53] Tamtéž
[54] Tamtéž
[55] Seznámení s Arduinem. *itnetwork.cz* [online]. [cit. 03.01.2019]. Dostupné z: https://www.itnetwork.cz/hardware-pc/arduino/arduino-seznameni

Mezi základní verze patří Arduino Uno, jeho menší verze Nano a větší verze Mega. Dále je zde verze Yún využívající připojení k Wi-Fi, nebo verze Lilypad, která se používá v oděvním průmyslu, například ve svítících vestách pro cyklisty.

3.9.1 Arduino Mega

Větší verze Arduina pracující na procesoru ATmega2560 (Obrázek 5), obsahuje 54 digitálních vstupně-výstupních pinů (14 z nich lze použít jako PWM výstup), 16 analogových vstupů, 4 hardwarové sériové porty, 16 MHz krystalový oscilátor, USB port, napájecí konektor, ICSP a tlačítko reset.[56]



Obrázek 5: Arduino Mega s popisem portů

Zdroj: https://cz.pinterest.com

3.9.2 Arduino periferie

Pro Arduino existuje mnoho vstupních a výstupních periferií, které můžeme připevnit k předem připraveným pinům, tyto periferie se dají rozdělit do dvou skupin, senzory a shieldy.

Senzory nám dokáží měřit různé veličiny, jako například teplotu a vlhkost vzduchu, množství škodlivých plynů a jiné veličiny. Do této skupiny patří i moduly, ty slouží pro zobrazení aktuální polohy, či moduly pro komunikaci Arduina s okolím ať prostřednictvím Bluetooth, nebo Wi-Fi.

[56] Arduino Mega 2560 Rev3. *Arduino* [online]. [cit. 03.01.2019]. Dostupné z: https://store.arduino.cc/mega-2560-r3

Shiledy jsou přídavné periferie, které se nasazují na mikrořadič a obsahují několik senzorů naráz, příkladem je výukový shield obsahující v sobě tlačítka pro ovládání, display společně s možností ovládání motorů.[57]

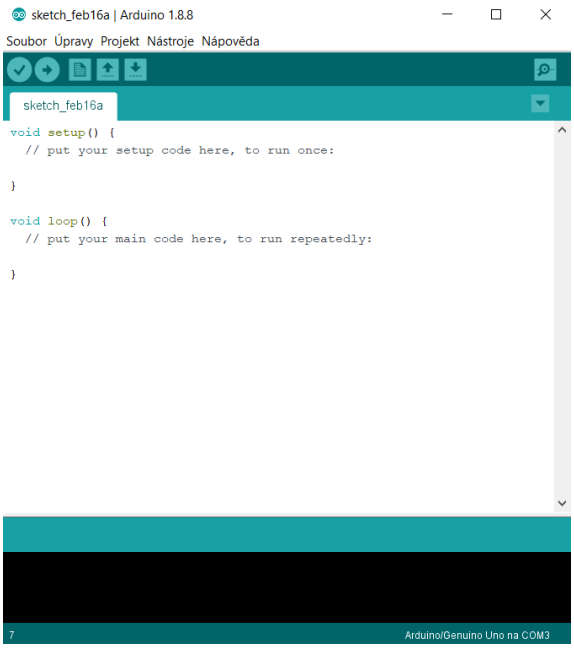
3.9.3 Arduino IDE

Jedná se o vývojové prostředí využívající jazyk C rozšířený o možnost ovládání vstupních a výstupních portů a dále o vlastní knihovny (Obrázek 6).[58]

V prvním řádku jsou umístěny navigační prvky. Jako první je zde „Soubor“, ve kterém můžeme spouštět již uložené projekty, nebo uložit projekt stávající. Následují „Úpravy“ sloužící k úpravě zdrojového projektu a jeho formátování, jako třetí je zde „Projekt“, který slouží pro odeslání již hotového kódu do zařízení Arduino, jako předposlední jsou „Nástroje“ sloužící hlavně pro výběr verze Arduina a portu, na kterém bude Arduino komunikovat s počítačem. Poslední „Nápověda“ slouží jako zdroj informací pro seznámení s Arduinem a jeho programováním.

V dalším řádku lze nalézt několik ikon. První je ikona fajfky „Ověřit“, kde po stisknutí dojde ke spuštění kontroly programu a jeho zdrojových kódů. Jestliže se najde nějaká chyba, dojde k upozornění programátora a zvýraznění dané chyby. Vedle ikony „Ověřit“ se nachází ikona „Nahrát“ ve tvaru šipky doprava. Ta spustí kontrolu programu, a posléze dojde k nahrání programu do připojeného Arduina. Vedle ní je ikona se symbolem stránky „Nový“ pro otevření nového souboru. Předposlední tlačítko v levé části se šipkou nahoru „Otevřít“ slouží pro otevření již uložených projektů. Tlačítkem se šipkou dolů „Uložit“ lze uložit současný program. Zcela vpravo na stejném řádku nalezneme ještě ikonu s lupou „Sériový Monitor“, který slouží pro sledování komunikace na sériové lince. Velký prostor s bílým pozadím slouží pro zápis zdrojového kódu a do černého prostoru ve spodní části se zobrazují chybová hlášení.[59]

[57] Seznámení s Arduinem. *itnetwork.cz* [online]. [cit. 03.01.2019]. Dostupné z: https://www.itnetwork.cz/hardware-pc/arduino/arduino-seznameni
[58] Programujeme Arduino. *Arduino.cz* [online]. [cit. 03.01.2019]. Dostupné z: https://arduino.cz/programujeme-arduino/
[59] Tamtéž



Obrázek 6: Arduino IDE

Z obrázku jsou patrné dvě struktury kódu. První z nich je blok metody setup. Mezi složené závorky se v tomto bloku píše kód provedený pouze jednou vždy na začátku programu, konkrétně po připojení ke zdroji napájení, stisknutí tlačítka pro reset, nebo opětovného nahrání kódu do mikrořadiče. Druhým blokem je metoda loop, do jehož složených závorek se zapisuje kód, který se neustále opakuje, dokud nedojde k odpojení napájení.[60]

3.10 Projekty využívající Arduino a mobilní telefon s Androidem

Na internetu se dají najít různé Android aplikace na ovládání auta s řídící jednotkou Arduino, ale tyto projekty bývají z větší části naprogramovány pomocí softwaru MIT App Inventor[61] jakožto zástupce grafického programování, kde jsou předem připravené bloky, které mají předem daný účel s cílem tyto bloky různě pospojovat.

3.10.1 mBot

Robot čínské společnosti Makeblock[62], využívající coby řídící jednotku upravenou verzi mikrořadiče Arduino. Mbot (Obrázek 7) si klade za cíl naučit děti programovat

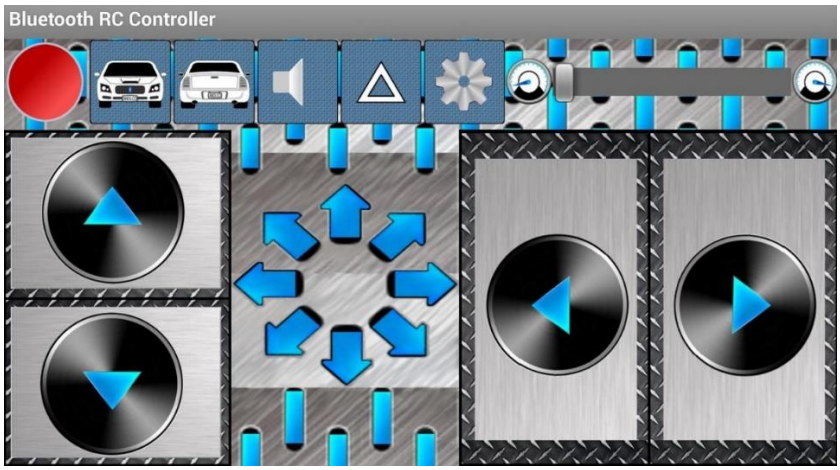
[60] Programujeme Arduino. *Arduino.cz* [online]. [cit. 03.01.2019]. Dostupné z: https://arduino.cz/programujeme-arduino/
[61] Explore MIT App Inventor. *MIT App Inventor* [online]. [cit. 5.3.2019]. Dostupné z: https://appinventor.mit.edu/
[62] Robot Kits for Kids : mBot. *Makeblock: Global STEAM Education Solution Provider* [online]. [cit. 05.03.2019]. Dostupné z: https://makeblock.com/steam-kits/mbot

a vytvářet aplikace pro operační systém Android za pomocí grafického programování ve vývojovém prostředí podobnému aplikaci Scratch. Samotný robot je k dostání s již připraveným firmwarem řídící jednotky, nebo si lze vytvořit firmware vlastní.

Obrázek 7: mBot
Zdroj: https://www.makeblock.com

3.10.2 Arduino Bluetooth RC Car

Tato aplikace[63] umožňuje ovládat auto pomocí Bluetooth (Obrázek 8). Autor tohoto programu vytvořil tuto aplikaci pomocí vyššího programovacího jazyka Java. Ovládání auta probíhá za pomocí směrových šipek, pomocí připraveného posuvníku se reguluje rychlost auta. Nachází se zde nalézají tlačítka pro zapínání světel motorů společně s tlačítkem pro klakson.



Obrázek 8: Bluetooth RC Car
Zdroj: https://play.google.com

[63] Arduino Bluetooth RC Car. *Ardumotive Arduino Greek Playground* [online]. [cit. 05.03.2019]. Dostupné z: https://www.ardumotive.com/bluetooth-rc-car.html

4 Vlastní práce

Praktická část je rozdělená do několika částí. Nejdříve je provedena analýza požadavků nové aplikace pro ovládání RC auta. Na základě analýzy požadavků se provede návrh nové aplikace. Z vypracovaného návrhu se přejde do části samotné realizace aplikace ve vývojovém prostředí Android studio a jazyka Java. V rámci realizace se popíší jednotlivé funkcionality aplikace včetně zobrazení důležitých částí kódu. Realizace se dále věnuje doprovodnému firmwaru řídící jednotky Arduino. Výsledná aplikace se následně otestuje. Konec praktické části se zaměří na porovnání výsledné aplikace s projektem Arduino Bluetooth RC Car, neboť podobně jako výsledná aplikace diplomové práce, byla naprogramována v programovacím jazyce Java a také je určená pro mobilní telefony.

4.1 Analýza požadavků

Tato diplomová práce se zabývá aplikací na ovládání RC auta, tudíž do hlavních požadavků patří schopnost ovládat model auta pomocí Android aplikace využívající pro komunikaci bezdrátovou technologii Bluetooth.

Seznam funkčních požadavků lze nalézt v následujícím výčtu:

- Schopnost připojit se pomocí Bluetooth k řídící jednotce auta,
- ovládání směrů auta pomocí dvou joysticků,
- zatáčení auta provádět za pomocí servomotoru a jeho nápravy,
- otočení nápravy přesně o daný úhel zadaný v aplikaci,
- umožnění zobrazení souřadnic ze senzoru GPS,
- ze zjištěných souřadnic zobrazit v Google mapách polohu auta,
- možnost archivaci souřadnic do souboru na SD kartě umístěné v modelu auta,
- načtení souboru se souřadnicemi do Google map vrstvy výsledné aplikace.

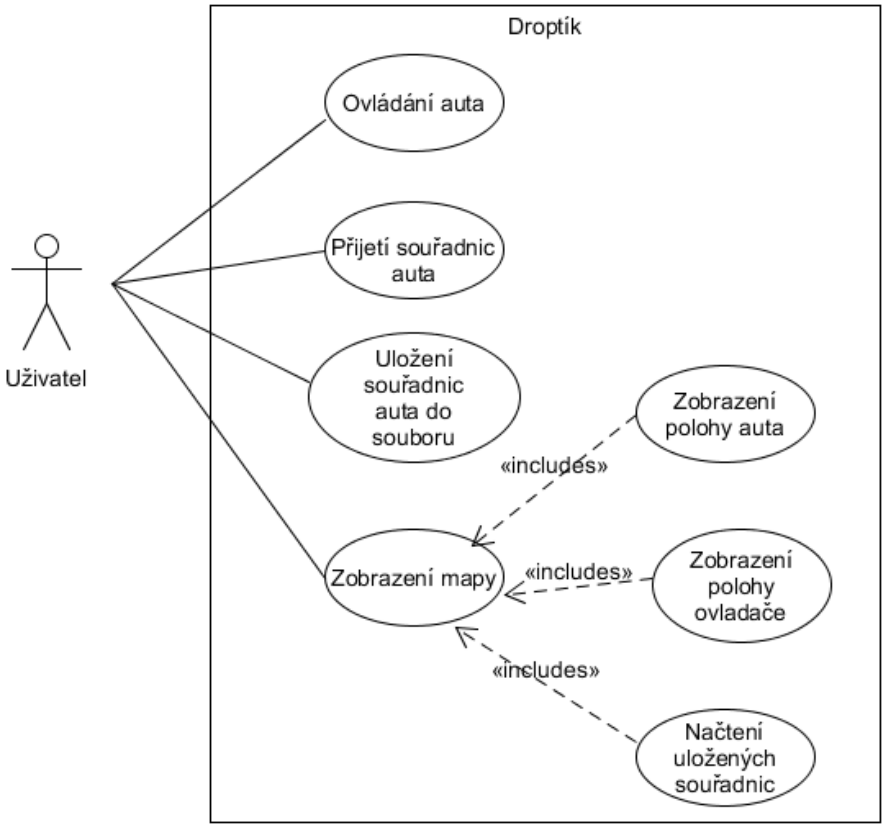
Seznam nefunkčních požadavků:

- Aplikaci naprogramovat v programovacím jazyce Java ve vývojovém prostředí Android Studio,
- firmware auta naprogramovat v nativním prostředí Arduino IDE a jeho rozšířené verze jazyka C,
- celkově sladit vzhled jednotlivých obrazovek aplikace,
- rozdělit funkcionality do více obrazovek,
- možnost přepínání obrazovek aplikace pomocí menu,

- možnost zobrazit na mapě i polohu ovladače.

4.1.1 Use Case diagram

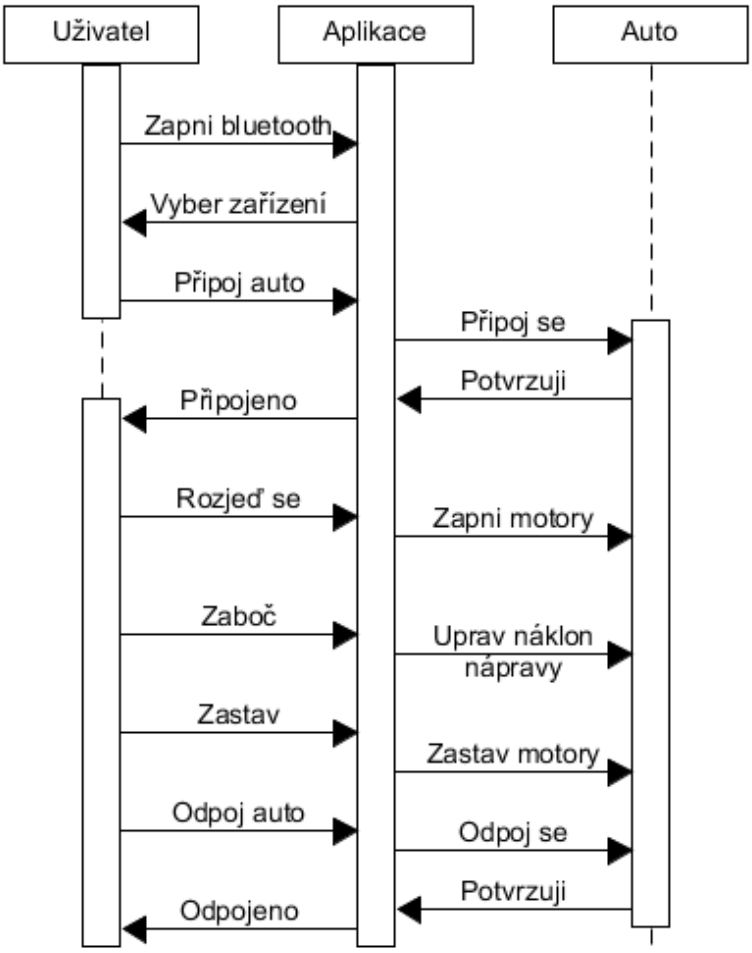
Následující diagram (Obrázek 9) zachycuje možnosti využití aplikace uživatelem coby jediným aktérem. Uvnitř systému se nachází jednotlivé případy užití, ať už pro ovládání, či zobrazení souřadnic.



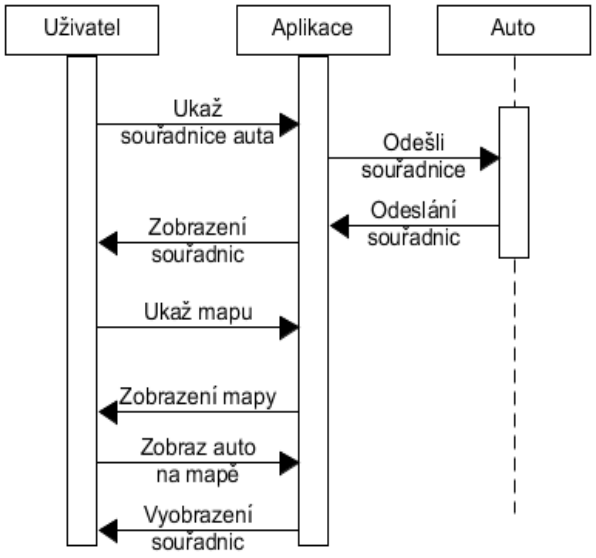
Obrázek 9: Use Case diagram aplikace

4.1.2 Sekvenční diagram

Sekvenční diagramy obsahují dva případy užití, konkrétně pro ovládání (Obrázek 10) a přijetí souřadnic (Obrázek 11).



Obrázek 10: Sekvenční diagram ovládání auta



Obrázek 11: Sekvenční diagram přijetí souřadnic

4.1.3 Scénář

Ve scénářích se uvádí posloupnost komunikace mezi uživatelem a autem, sloužící pro doplnění sekvenčních diagramů pro ovládání auta (Tabulka 1) a odesílání souřadnic (Tabulka 2).

Krok	Role	Akce
1	Uživatel	Žádá aplikaci, aby zapnula v mobilním telefonu Bluetooth, který slouží pro komunikaci s autem.
2	Aplikace	Posílá uživateli seznam spárovaných Bluetooth zařízení.
3	Uživatel	Uživatel vybere dané zařízení a požaduje zahájení komunikace.
4	Aplikace	Pošle autu požadavek o připojení a čeká na odezvu.
5	Auto	Pošle potvrzení, že je připraven.
6	Aplikace	Informuje uživatele o této skutečnosti.
7	Uživatel	Žádá, aby se auto rozjelo.
8	Aplikace	Vyšle požadavek uživatele.
9	Auto	Zvedne otáčky všech motorů pro rozjezd.
10	Uživatel	Žádá o změnu směru jízdy.
11	Aplikace	Vyšle požadavek uživatele o změně jízdy.
12	Auto	Upraví náklon nápravy podle příkazu z aplikace.
13	Uživatel	Stiskne tlačítko pro zastavení motorů.
14	Aplikace	Odešle požadavek.
15	Auto	Ukončí otáčky všech motorů.
16	Uživatel	Stiskne tlačítko pro odpojení.
17	Aplikace	Ukončí propojení s autem.
18	Auto	Potvrdí požadavek.
19	Aplikace	Informuje uživatele.

Tabulka 1: Scénář ovládání

Krok	Role	Akce
1	Uživatel	Žádá aplikaci o zobrazení souřadnice auta.
2	Aplikace	Vyšle tento požadavek autu.
3	Auto	Odešle souřadnice auta.
4	Aplikace	Vypíše informace uživateli.

5	Uživatel	Stiskne klávesnici pro zobrazení mapy.
6	Aplikace	Zobrazí mapu i s polohou uživatele.
7	Uživatel	Stiskne klávesnici pro zobrazení souřadnic auta.
8	Aplikace	Ukáže polohu auta.

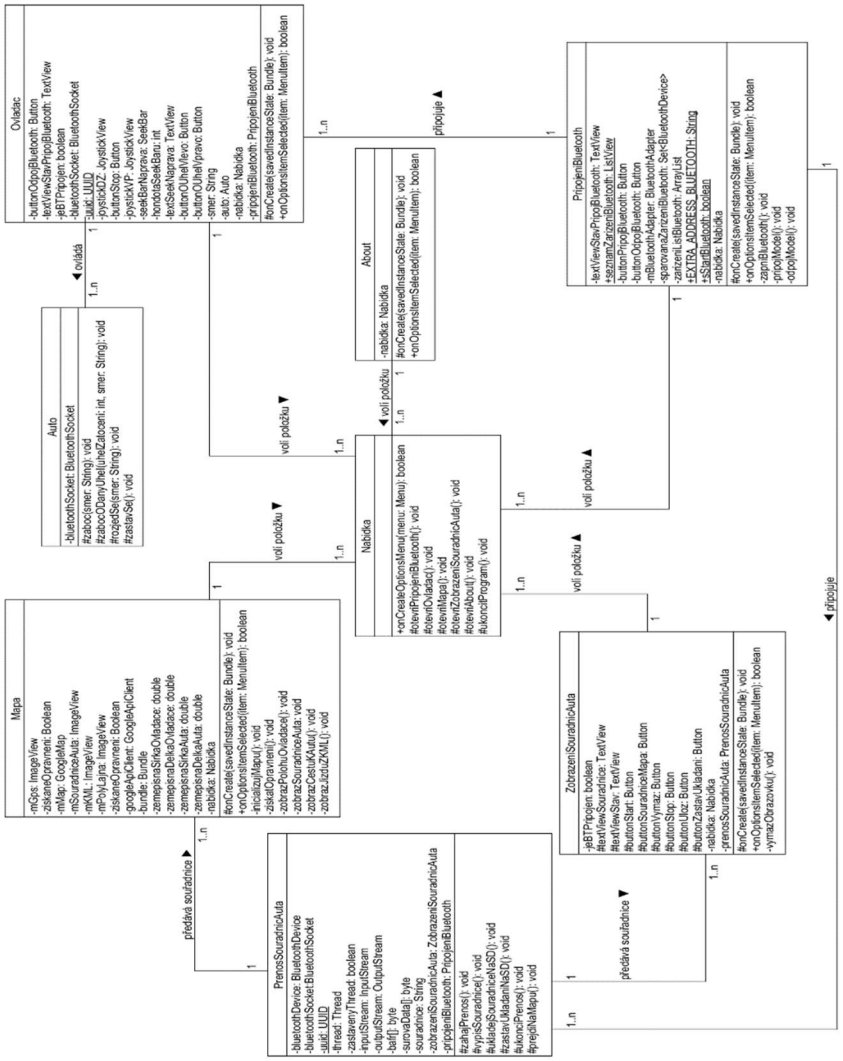
Tabulka 2: Scénář souřadnice

4.2 Návrh

Návrh aplikace zobrazuje diagram tříd mobilní aplikace společně s návrhem designu jednotlivých aktivit.

4.2.1 Diagram tříd

Diagram tříd (Obrázek 12) obsahuje jednotlivé třídy aplikace a jejich metody. Zvětšený diagram tříd lze nalézt v příloze (Příloha C).

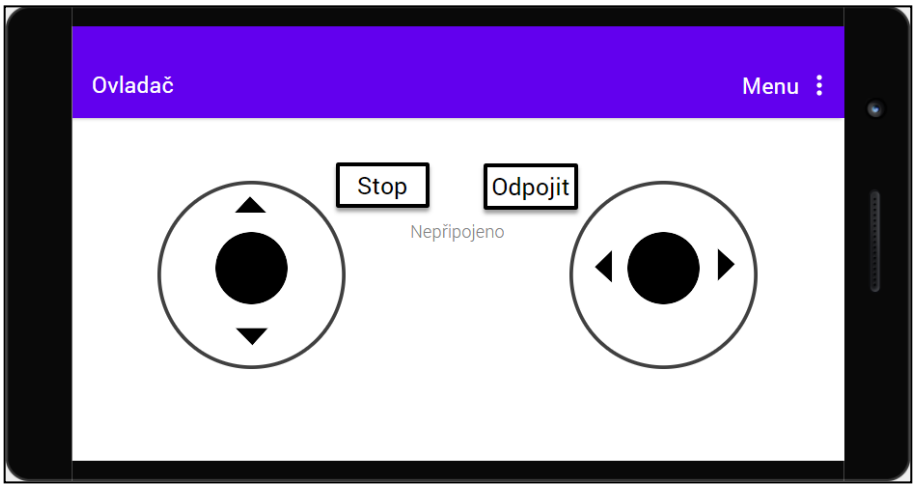


Obrázek 12: Diagram tříd

4.2.2 Návrh designu

4.2.2.1 Ovladač

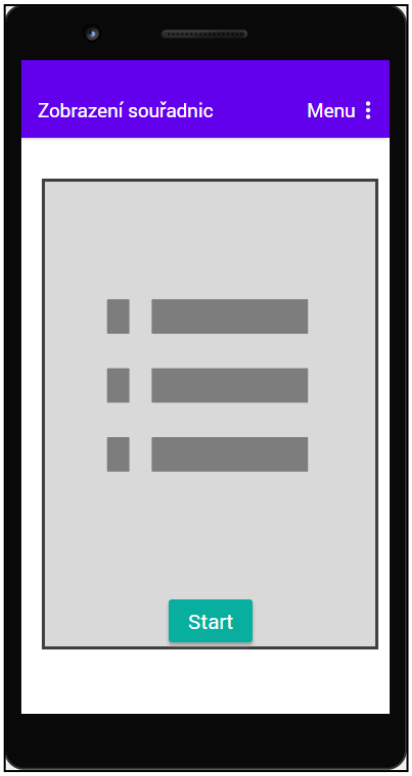
V ovladači (Obrázek 13) uživatel ovládá auto, proto zde jsou komponenty, které umožňují ovládání auta společně s ovládáním nápravy pro zatáčení.



Obrázek 13: Návrh ovladače

4.2.2.2 Zobrazení souřadnic

V okně zobrazení souřadnic auta (Obrázek 14) se vypisují souřadnice auta.



Obrázek 14: Návrh výpisu souřadnic

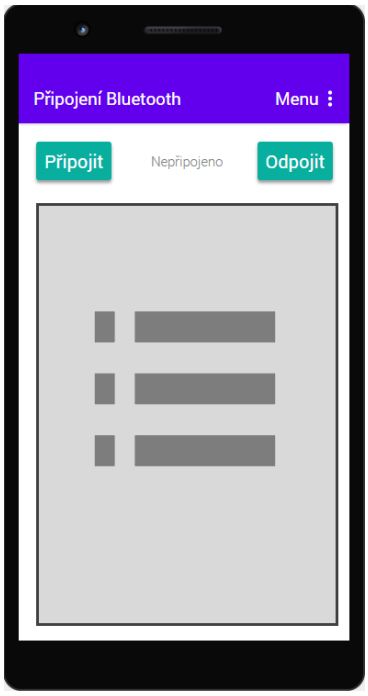
4.2.2.3 Mapy

Jakmile dojde k výpisu souřadnic auta, mohou se zobrazit do připravených Google map (Obrázek 15) společně se souřadnicemi ovladače.

Obrázek 15: Návrh aktivity mapa

4.2.2.4 Připojení Bluetooth

V této obrazovce (Obrázek 16) se zobrazí seznam všech Bluetooth zařízení, které jsou uživateli k dispozici a následně pro spojení ovladače a auta.



Obrázek 16: Návrh pro Bluetooth připojení

4.3 Realizace

Realizace probíhá ve dvou částech. První část je zaměřena na vývoj mobilní aplikace, druhá část je věnována firmwaru řídící jednotky.

4.3.1 Realizace mobilní aplikace

Tato část se zabývá jednotlivými aktivitami a pomocnými třídami mobilního ovladače. Jednotlivé soubory lze nalézt v adresářové struktuře Droptik\app\src\main. Složka java obsahuje zdrojové kódy všech tříd, tj. všech kódů neobsahující prvky designu, jelikož se během programování Android aplikací odděluje část logiky a designu, design aplikace se nachází ve složce res\layout.

4.3.1.1 Manifest

V manifestu se programuje pomocí XML skriptu, podobném jazyku HTML, a to pomocí tagů a atributů daného tagu.

V rámci manifestu je definován v tagu application název aplikace Droptík pomocí atributu android:label společně s ikonou android:icon viditelnou po instalaci v přehledu všech nainstalovaných aplikací. V tagu activity jsou definována jednotlivá okna aplikace a pro upřesnění některé vlastnosti je třeba zvolit jeho párovou alternativu. Tato možnost byla zvolena u aktivity pro připojení aplikace k autu, jelikož je určená coby spouštěcí aktivita pomocí párového tagu:

```
intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Dále se zde uvádí, které obrazovky se zobrazují na šířku a které na výšku pomocí android:screenOrientation. Pro ovladač se doporučuje mít obrazovku na šířku z důvodu lepšího uchopení mobilního telefonu oběma rukama a ovládání auta, ale pro zobrazení mapy je zase výhodnější mít obrazovku natočenou na výšku.

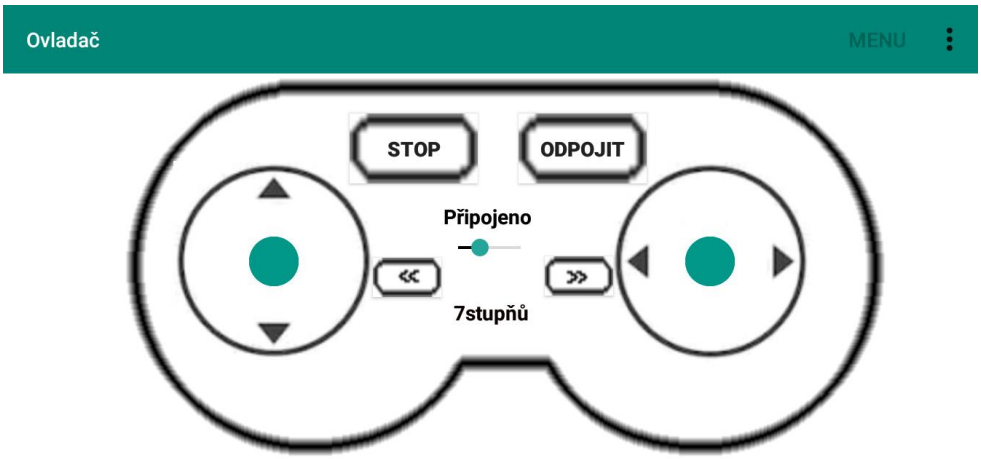
V souboru se dále zobrazují jednotlivá oprávnění požadované pro správné fungování aplikace. Aplikace pro svůj chod potřebuje získat oprávnění pro přístup k Bluetooth z důvodu ovládání auta, přístup k internetu z důvodu načtení Google mapy, přístup k aktuální poloze mobilního zařízení pro zobrazení polohy na mapě a možnost číst z interního uložiště mobilního zařízení pro zobrazení souboru se souřadnicemi.

4.3.1.2 Ovladač

Jedná se o hlavní část aplikace sloužící pro bezdrátové ovládání auta pomocí Bluetooth. Levá strana obsahuje joystick pro jízdu vpřed a vzad, pravá strana obsahuje joystick na ovládání servomotoru nápravy pro zatáčení auta. Na středu se nachází posuvník, jehož hodnota udává stupeň otočení nápravy. Pod posuvníkem se nacházejí dvě šipky znázorňující směr zatočení o zadaný úhel z posuvníku. Uživatel má dále možnost ručně se odpojit od auta. O komunikaci mezi ovladačem a modelem se stará třída Auto na základě volání metod této třídy, toto volání se odehrává v metodě onCreate. Tato metoda obsahuje takzvaný Listener[64], který reaguje na uživatelskou odezvu (stisk tlačítek, pohyb joysticku a podobně). Uvedený příklad znázorňuje Listener pro odeslání příkazu na zatočení auta (Zdrojový kód 1). Vzhled obrazovky ovladač udává (Obrázek 17).

```
joystickVP.setOnMoveListener(new JoystickView.OnMoveListener() {
    @Override
    public void onMove(int uhel, int intenzita) {
        if (uhel == 180 && intenzita == 100) { //jakmile se joystick
nakloní k levému kraji
            //volání metody třídy Auto pro zatočení doleva
            smer = "doleva";
            auto.zaboc(smer);

        }
        if (uhel == 0 && intenzita == 100) { //jakmile se joystick
nakloní k pravému kraji
            //volání metody třídy Auto pro zatočení doprava
            smer = "doprava";
            auto.zaboc(smer);
        }
    }
});
```

Zdrojový kód 1: Listener zabočení auta



Obrázek 17: Activity ovladač

[64] Česky naslouchač

4.3.1.3 Auto

Třída zajišťující odeslaní příkazů pro pohyb auta na základě stisknutí určitého tlačítka uživatelem.

Změna polohy pravého joysticku zavolá metodu zaboc(String smer) (Zdrojový kód 2), která má jako parametr směr otočení auta. Změna polohy levého joysticku zavolá metodu rozjedSe(String smer) (Zdrojový kód 3), zde parametr reprezentuje jízdu vřed a vzad. Ovladač umožňuje otočení modelu o předem stanovený úhel, za tímto účelem se zde nachází posuvník umístěný uprostřed obrazovky. Šipky kolem posuvníku odesílají úhel posuvníku modelu pomocí metody zabocODanyUhel(int uhelZatoceni, String smer) (Zdrojový kód 4), parametry znamenají velikost otočení a jeho směr. Metoda obsahuje koeficient, který se přičte k požadovanému úhlu otočení a na základě jeho velikosti auto pozná jakým směrem má natočit nápravu. Třída také obsahuje metodu pro zastavení modelu.

```
//metoda pro zatáčení auta do strany
protected void zaboc(String smer) {
    if (smer == "doleva") { //jakmile se joystick nakloní k levému
kraji
        try {
            bluetoothSocket.getOutputStream().write(7); //odeslání
příkazu pro zatočení doleva
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (smer == "doprava") { //jakmile se joystick nakloní k pravému
kraji
        try {
            bluetoothSocket.getOutputStream().write(6); //odeslání
příkazu pro zatočení doprava
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Zdrojový kód 2: Zabočení auta

```
//metoda pro řízení auta
protected void rozjedSe(String smer) {
if (smer == "dopredu") {
    try {
        bluetoothSocket.getOutputStream().write(1); //rozjed auto
dopředu pomalou rychlostí
    } catch (IOException e) {
        e.printStackTrace();
    }
}
if (smer == "dozadu") {
    try {
        bluetoothSocket.getOutputStream().write(2); //couvej pomalou
rychlostí
```

```
    } catch (IOException e) {
        e.printStackTrace();
    }
}

    if (smer == "dopreduPlna") {
        try {
            bluetoothSocket.getOutputStream().write(3); //rozjed auto
plnou rychlostí dopředu
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (smer == "dozaduPlna") {
        try {
            bluetoothSocket.getOutputStream().write(4); //couvej plnou
rychlostí
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Zdrojový kód 3: Rozjíždění auta

```
//metoda pro zatočení o daný úhel
protected void zabocODanyUhel(int uhelZatoceni, String smer) {
    if (smer == "doleva") {
        try {
            bluetoothSocket.getOutputStream().write(uhelZatoceni +
100); //přičtení koeficientu 100 k hodnotě posuvníku pro zatočení
doleva
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (smer == "doprava") {
        try {
            bluetoothSocket.getOutputStream().write(uhelZatoceni +
200); //přičtení koeficientu 200 k hodnotě posuvníku pro zatočení
doprava
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Zdrojový kód 4: Zabočení o zadaný úhel

4.3.1.4 Připojení Bluetooth

Toto okno se vždy zobrazí jako první při každém spuštění aplikace a je určeno pro připojení auta pro komunikaci s mobilním ovladačem. Při připojení nejdříve dochází k ověření, zda má ovladač zapnutý Bluetooth metodou zapniBluetooth (Zdrojový kód 5). Pokud není zapnutý, zobrazí se hlášení uživateli, zda může Bluetooth zpřístupnit. Pokud již zapnutý je, zavolá se metoda pripojModel() (Zdrojový kód 6) se seznamem všech již spárovaných zařízení. Pro fungování aplikace je třeba nejdříve spárovat mobilní telefon

s Bluetooth modulem řídící jednotky Arduino, které se provádí v nastavení telefonu. Při párování je uživatel dotázán na přístupové heslo, které je ve výchozím stavu nastaveno na 1234.

```
private void zapniBluetooth() {
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
//Bluetooth adaptér zařízení ovladače
    buttonPripojBluetooth.setOnClickListener(new View.OnClickListener()
{
        @Override
        public void onClick(View v) {
            if(mBluetoothAdapter == null) { //jestli je aplikace
nainstalována na zařízení kde není Bluetooth k dispozici
                Toast = Toast.makeText(getApplicationContext(),
"Zařízení nepodporuje Bluetooth", Toast.LENGTH_LONG);
                toast.show();
                finish();//ukončit aplikaci
            }
            if(!mBluetoothAdapter.isEnabled()){ //pokud je Bluetooth
zrovna vypnutý
                textViewStavPripojBluetooth.setText("Nepřipojeno");
                Intent intentBluetooth = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE); //zapnutí Bluetooth
                startActivityForResult(intentBluetooth,1);
                textViewStavPripojBluetooth.setText("Stiskněte znova na
Připojit");
            }
            if(mBluetoothAdapter.isEnabled()){
                seznamZarizeniBluetooth.setVisibility(View.VISIBLE);
                zobrazSparovanaZarizeni(); //volání metody pro
zobrazení spárovaných zařízení
                Toast = Toast.makeText(getApplicationContext(),
"Vyberte zařízení pro komunikaci s Bluetooth", Toast.LENGTH_LONG);
                toast.show();
            }
        }
    });
}
```

Zdrojový kód 5: Metoda pro zapnutí Bluetooth

V metodě pripojModel() (Zdrojový kód 6) se do položky ListView zobrazí seznam zařízení s Bluetooth. Jakmile uživatel stiskne jednu z nabízených možností, uloží se adresa daného zařízení a dojde k přesměrování na aktivitu s ovladačem auta.

```
private void pripojModel(){
    //Zobrazuje již spárovaná zařízení
    sparovanaZarizeniBluetooth = mBluetoothAdapter.getBondedDevices();
    zarizeniListBluetooth = new ArrayList();

    if(sparovanaZarizeniBluetooth.size()>0) {
        for (BluetoothDevice device : sparovanaZarizeniBluetooth) {
            zarizeniListBluetooth.add(device.getName() + "\n" +
device.getAddress());
        }
    }else{
        Toast = Toast.makeText(getApplicationContext(), "Žádné zařízení
k dispozici. Proveďte párování ručně!", Toast.LENGTH_LONG);
```

```
            toast.show();
        }
        final ArrayAdapter adapter = new
ArrayAdapter(this,android.R.layout.simple_list_item_1,
zarizeniListBluetooth); //určení vzhledu seznamu zařízení
        seznamZarizeniBluetooth.setAdapter(adapter);
        seznamZarizeniBluetooth.setOnItemClickListener(listClickListener);
//volání naslouchače po zvolení zařízení
}

private AdapterView.OnItemClickListener listClickListener = new
AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View, int position,
long id) {
            String info = ((TextView) view).getText().toString(); //získání
jména zařízení
            String adresa = info.substring(info.length() - 17); //získání
jeho adresy
            Intent i = new Intent(PripojeniBluetooth.this, Ovladac.class);
//přesměrování na obrazovku ovladač

            sStartBluetooth =true;
            i.putExtra(EXTRA_ADDRESS_BLUETOOTH, adresa);
            startActivity(i);
    }
};
```

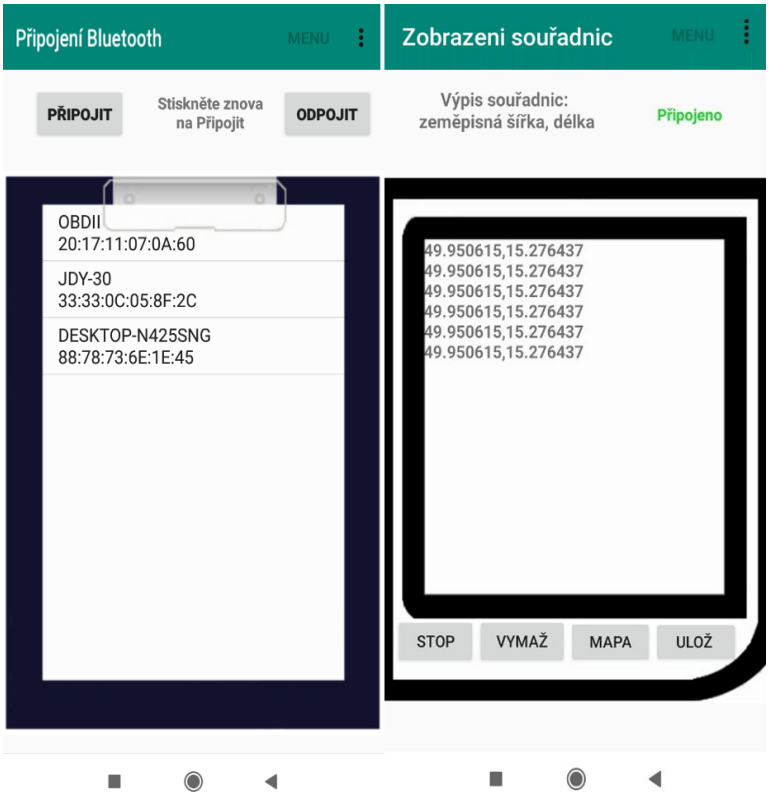
Zdrojový kód 6: Seznam spárovaných zařízení

Vzhled obrazovky připojení Bluetooth udává (Obrázek 18) společně se vzhledem obrazovky pro výpis souřadnic.

Třída dále obsahuje metodu pro odpojení modelu, která kromě odpojení zařizuje i změnu informace o stavu připojení na požadovaných obrazovkách aplikace využívajících Bluetooth.

4.3.1.5 Zobrazení souřadnic auta

Třída určená pro zobrazení aktuálních souřadnic auta senzoru GPS připevněném na autě. Uživatel může stiskem tlačítka start začít přenášet souřadnice modelu. Tlačítkem mapa dojde k přechodu na obrazovku mapa a zobrazení poslední polohy auta na mapě. Zmáčknutím ulož se odešle příkaz na archivaci souřadnic do souboru na SD kartě umístěné v autě. Souřadnice se na obrazovce zobrazují pod sebou ve formátu zeměpisná šířka, zeměpisná délka s nejnovější hodnotou vždy na konci. Stejně jako u třídy Ovladac, tak i zde komunikaci mezi modelem a aplikací provádí zvláštní třída. Vzhled obrazovky zobrazení souřadnic udává (Obrázek 18) společně s obrazovkou pro připojení k Bluetooth.



Obrázek 18: Obrazovky připojení Bluetooth a zobrazení souřadnic

4.3.1.6 Přenos souřadnic auta

Třída, která provádí operace třídy Zobrazení souřadnic. Souřadnice zachytává aplikace ve formě surových dat, která jsou následně zobrazována do připraveného pole. O zachycení přijatých dat, k jejím zobrazení v aplikaci, se využívá metoda vypisSouradnice() (Zdrojový kód 7). O přesun souřadnic do aktivity Mapa po stisknutí tlačítka z aktivity Zobrazení souřadnic se stará metoda prejdiNaMapu() (Zdrojový kód 8). Tlačítkem ulož se vyšle signál o ukládání souřadnic do souboru na SD kartě pomocí metody ukladejSouradnice (Zdrojový kód 9).

```
protected void vypisSouradnice() {
    final Handler = new Handler();
    zastavenyThread = false;
    bafr = new byte[1024];
    Thread thread = new Thread(new Runnable() {
        public void run() {
            while (!Thread.currentThread().isInterrupted() &&
!zastavenyThread) {
                try {
                    int velikostBafru = inputStream.available();
                    if (velikostBafru > 0) {
                        surovaData = new byte[velikostBafru];
                        inputStream.read(surovaData);
                        final String string = new String(surovaData,
"UTF-8");
                        handler.post(new Runnable() {
                            public void run() {
```

```
                                souradnice = string;

zobrazeniSouradnicAuta.textViewSouradnice.append(string);
                            }
                        });
                    }
                } catch (IOException ex) {
                    zastavenyThread = true;
                }
            }
        }
    });
    thread.start();
}
```

Zdrojový kód 7: Přijetí souřadnic

```
protected void prejdiNaMapu() {
    if (souradnice != null) {
        Intent intentSouradnice = new Intent(this, Mapa.class);
        intentSouradnice.putExtra("souradniceAuta", souradnice);
//předání získaných souřadnic do aktivity mapa
        startActivity(intentSouradnice);
    }
}
```

Zdrojový kód 8: Zobrazeni mapy

```
protected void ukladejSouradniceNaSD() {
    zobrazeniSouradnicAuta.buttonUloz.setVisibility(View.INVISIBLE);
zobrazeniSouradnicAuta.buttonZastavUkladani.setVisibility(View.VISIBLE)
;
    try {
        bluetoothSocket.getOutputStream().write(9);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Zdrojový kód 9: Ukládání souřadnic do souboru

4.3.1.7 Mapa

V tomto okně jsou implementovány mapy společnosti Google. Pro získání přístupu k mapám je třeba získat klíč opravňující vývojáře používat API Google maps. Ze získaného API využívám metody pro zobrazení map, u kterých je nutné mít na mobilním telefonu přístup k internetu alespoň přes Wi-Fi.

Uživatel je zde přesměrován z aktivity Zobrazení souřadnic a polohu auta lze zobrazit po stisknutí ikony auta umístěné v levém horním rohu (Zdrojový kód 10). Po načtení API se automaticky zobrazí poloha mobilního ovladače a lze se k ní kdykoli vrátit stiskem na obrázek mobilního telefonu v pravém horním rohu. Souřadnice jízdy řídící jednotka ukládá

do .csv souboru, tento soubor lze přes webovou aplikaci convertcsv [65] převést do .kml souboru a později zobrazit na mapě přes ikonu mapy v levém dolním rohu obrazovky. Pro lepší nalezení zatoulaného auto lze na zjištěné souřadnice navést pomocí Google navigace, která nalezne nejrychlejší cestu k modelu. Při zobrazení cesty ze souboru KML anebo pomocí křivky působí vrstva s mapou coby kreslící plátno a jednotlivé prvky se nanáší pomocí formátu mMap tečka a metoda, kde mMap značí mapovou vrstvu. Příkladem může být ukázka jízdy s využitím křivky (Zdrojový kód 11).

Vzhled obrazovky mapa lze vidět na (Obrázek 19), obrázek obsahuje vzhled aktivity společně s ukázkou zobrazení souřadnic auta, načtení KML souboru se souřadnicemi a zachycení cesty k autu s využitím křivky.

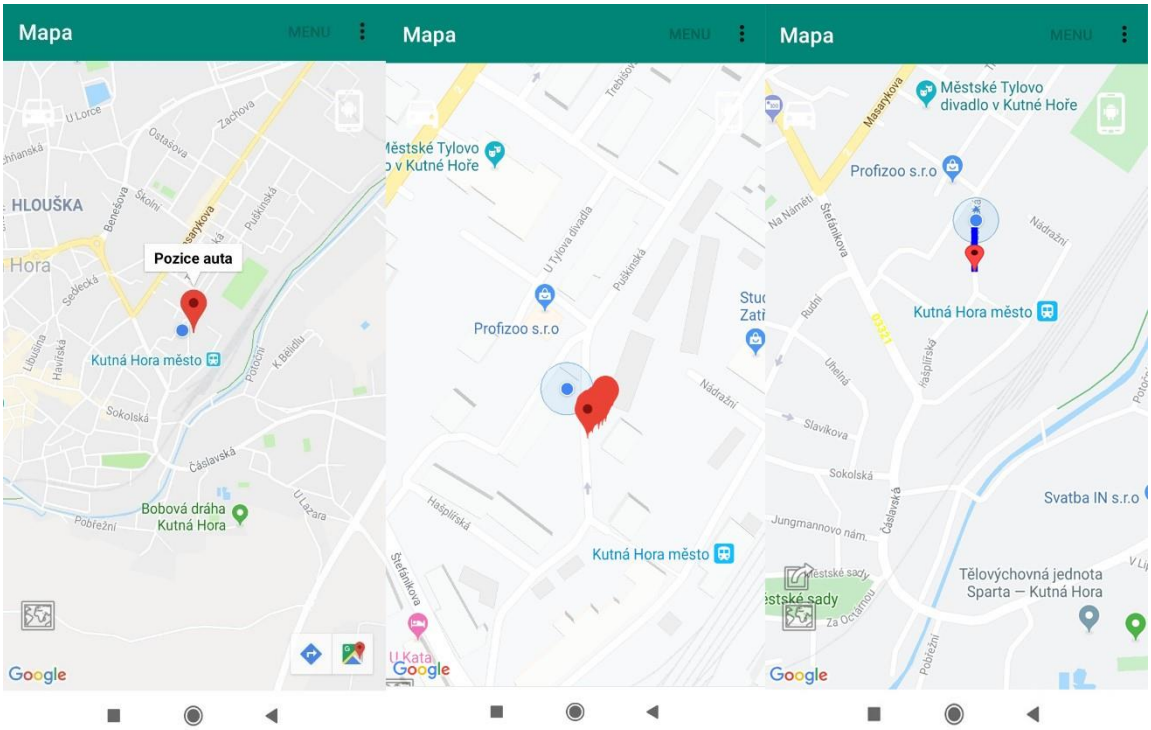
```
private void nactiSouradnice() {
    String souradniceAuta = bundle.getString("souradniceAuta");
//přijetí souřadnic z obrazovky souřadnice
    String[] tokens = souradniceAuta.split(","); //rozdělení souřadnic
na zeměpisnou šířku a délku
    zemepisnaSirka = Double.parseDouble(tokens[0]); //převod stringu na
číslo
    zemepisnaDelka = Double.parseDouble(tokens[1]); //převod stringu na
číslo
    LatLng auto = new LatLng(zemepisnaSirka, zemepisnaDelka);
//zobrazení souřadince na mapě
    mMap.addMarker(new MarkerOptions().position(auto).title("Pozice
auta")); //po stisknutí na ukazatel zobrazení hlášky
    mMap.moveCamera(CameraUpdateFactory.newLatLng(auto)); //přesun mapy
na pozici auta
}
```

Zdrojový kód 10: Zobrazení souřadnic na mapě

```
mMap.addPolyline(new PolylineOptions()
 //zobrazení aktuální polohy ovladače přes metody get,
zemepisnaSirkaAuta a delka jsou souřadnice auta
.add(new LatLng(zemepisnaSirkaOvladace, zemepisnaDelkaOvladace),new
LatLng(zemepisnaSirkaAuta, zemepisnaDelkaAuta))
.width(8f) //šířka čáry znázorňující cestu
.color(Color.BLUE) //barva čáry
);
```

Zdrojový kód 11: Vykreslení cesty k autu

[65] http://www.convertcsv.com/csv-to-kml.htm



Obrázek 19: Activity zobrazení mapy

4.3.1.8 About

Obrazovka obsahující jméno autora práce společně s názvem výsledné aplikace.

4.3.1.9 Nabídka

O vytvoření menu (Zdrojový kód 12) a přechod mezi jednotlivými obrazovkami aplikace (Zdrojový kód 13) obstarává třída Nabidka.

```
public boolean onCreateOptionsMenu(android.view.Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.hlavni_menu, menu);
    return true;
}
```

Zdrojový kód 12: Vytvoření menu

```
protected void otevriOvladac() { //otevření aktivity ovladač
    Intent intent = new Intent(getBaseContext(), Ovladac.class);
    startActivity(intent);
}
```

Zdrojový kód 13: Otevření aktivity ovladač

4.3.2 Realizace firmwaru řídící jednotky

4.3.2.1 Protokol komunikace

Komunikace mezi řídící jednotkou a ovladačem probíhá přes sériovou linku mikrořadiče Arduino. Princip spočívá v odesílání příkazu z mobilní aplikace přes Bluetooth

se speciálním číslem. Dané číslo reprezentuje jednu operaci, ať už se jedná o jízdu vpřed, čí odeslání souřadnic. Každá operace se řeší ve své vlastní metodě. Seznam příkazů a operací se nalézá v následujícím výčtu (Tabulka 3).

Číslo příkazu	Operace
1	Rozjíždění dopředu
2	Rozjíždění dozadu
3	Couvání
4	Jízda vpřed
5	Zastavení
6	Zabočení doprava
7	Zabočení doleva
8	Odeslání souřadnic
9	Ukládání souřadnic na SD kartu
10	Ukončení ukládání souřadnic

Tabulka 3: Seznam operací

Ve firmwaru nejdříve dochází k ověření, zda došlo k přijetí příkazu, následně konstrukce switch přiřadí příkaz patřičné metodě (Zdrojový kód 14).

```
if (bluetooth.available() > 0 ) { // pokud přišla hodnota k bluetooth
    hodnotaBluetooth = bluetooth.read(); //čtení hodnoty bluetooth
  } //konec bluetooth ověření

  //Na základě přijatého příkazu proveď požadovanou operaci
  switch (hodnotaBluetooth) {
    case 1: //jakmile dojde k načtení příkazu pro pomalé rozjíždění
      pomaleRozjizdeni(); //volaní metody pro rozjíždění
      break;
    case 2: //jakmile dojde k načtení příkazu pomalé couvání
      pomaleCouvani(); //volaní metody pro pomalé couvání
      break;
    case 3: //jakmile dojde k načtení příkazu pro couvání
      dozadu(); //volaní metody vzad
      break;
    case 4: //jakmile dojde k načtení příkazu pro jízdu vpřed
      dopredu(); //volaní metody vpřed
      break;
```

```
    case 5: //jakmile dojde k načtení příkazu stop
      zastav(); //volaní metody zastav
      break;
    case 6: //jakmile doje k načtení příkazu pro otočení doprava
      doprava(); //volaní metody doprava
      break;
    case 7: //jakmile doje k načtení příkazu pro otočení doleva
      doleva(); //volaní metody doleva
      break;
    case 8: //jakmile dojde k načtení příkazu odeslání souřadnic
      vypisSouradnic(); //volaní metody vypisSouradnic
      break;
    case 9: //jakmile dojde k načtení příkazu k ukládání souřadnic na SD
kartu
      if (SD.begin(pinCS)) { //pokud se načetl SD modul
        zapisNaSD();
      }
      break;
    case 10: //jakmile dojde k načtení příkazu k ukončení ukládání
souřadnic na SD kartu
      if (SD.begin(pinCS)) { //pokud se načetl SD modul
        ukonciZapisNaSD();
      }
      break;
}
```

Zdrojový kód 14: Vyhodnocení přijatého příkazu

4.3.2.2 Ovládání auta

Motory pohonu auta jsou ovládány pomocí Motor shieldu, který se nasazuje na jeho řídící jednotku. Shield obsahuje připravený firmware na ovládání motorů, tudíž při implementaci stačí používat připravené příkazy pomocí metody digitalWrite, kde se jako parametr udává směr jízdy. Rychlost motoru se udává za pomocí metody analogWrite s názvem konkrétního motoru a jeho rychlosti coby parametry. Ovládání motorů pohybu zajišťují metody dopředu (Zdrojový kód 15) a dozadu (Zdrojový kód 16). Firmware umožňuje také auto zastavit (Zdrojový kód 17).

```
void pomaleRozjizdeni() {
  digitalWrite(smerPrava, HIGH); //zapni motor pro jízdu vpřed, HIGH
  digitalWrite(smerLeva, HIGH);
  analogWrite(rychlostPrava, 190); //snížená rychlost motorů
```

```
  analogWrite(rychlostLeva, 190);
  delay(300);
}
void dopredu() {
  digitalWrite(smerPrava, HIGH); //zapni motor pro jízdu vpřed, HIGH
  digitalWrite(smerLeva, HIGH);
  analogWrite(rychlostPrava, 255); //nastaveni rychlosti na max
  analogWrite(rychlostLeva, 255);
  delay(300);
}
```

Zdrojový kód 15: Jízda dopředu

```
void pomaleCouvani() {
  digitalWrite(smerPrava, LOW); //zapni motor pro jízdu vzad, LOW
  digitalWrite(smerLeva, LOW);
  analogWrite(rychlostPrava, 190); //snížená rychlost motorů
  analogWrite(rychlostLeva, 190);
  delay(300);
}
void dozadu() {
  digitalWrite(smerPrava, LOW); //zapni motor pro jízdu vzad, LOW
  digitalWrite(smerLeva, LOW);
  analogWrite(rychlostPrava, 255); //nastaveni rychlosti couvaní na max
  analogWrite(rychlostLeva, 255);
  delay(300);
}
```

Zdrojový kód 16: Jízda dozadu

```
void zastav() {
  digitalWrite(smerPrava, LOW);//zastav motor
  digitalWrite(smerLeva, LOW);
  analogWrite(rychlostPrava, 0); //vynuluj rychlost
  analogWrite(rychlostLeva, 0);
  delay(300);
}
```

Zdrojový kód 17: Zastavení

Pro ovládání servomotoru s upevněnou nápravou se využívá metoda write z knihovny Servo.h, parametr metody udává o kolik stupňů se má servomotor natočit. V případě modelu s výchozí hodnotou devadesát stupňů se motor natáčí vždy o dvacet stupňů doleva

a doprava, toto omezení je dáno konstrukcí auta. Zatáčení auta provádějí metody doleva (Zdrojový kód 18) a doprava (Zdrojový kód 19).

```
void doleva() {
  if (servoNaprava_pozice >= 70) { //otočí servomotor lze jen od 90 do 70 stupňů, 70 stupňů je zvoleno z důvodu konstrukce auta, která nenabízí větší otočení nápravy doleva
    servoNaprava_pozice -= 1; //odečtení 1 stupně k servomotoru
    servoNaprava.write(servoNaprava_pozice);
    delay(5);
  }
}
```

Zdrojový kód 18: Zabočení doleva

```
void doprava() {
  if (servoNaprava_pozice <= 110) {
    servoNaprava_pozice += 1; //přičtení 1 stupně k servomotoru
    servoNaprava.write(servoNaprava_pozice);
    delay(5);
  }
}
```

Zdrojový kód 19: Zabočení doprava

Na získání souřadnic GPS se využívá knihovna TinyGPS++.h, v rámci metod z této knihovny dochází k překladu surových dat ze senzoru do srozumitelného formátu. Souřadnice jsou ovladači odesílány ve formátu „zeměpisná šířka,zeměpisná délka“, oba údaje jsou zkráceny na šest desetinných míst z důvodu opakování nul v dalších desetinných cifrách (Zdrojový kód 20). Zjišťování souřadnic probíhá nezávisle na uživateli vždy po každém zapnutí auta. Jakmile řídící jednotka příjme příkaz na posílání souřadnic dochází k odesílání aktuální pozice auta, další souřadnice se odesílají vždy po minutě (Zdrojový kód 21).

```
while (Serial3.available() > 0) {
    if (gps.encode(Serial3.read())) {
      zemSirka = gps.location.lat(); //získání zeměpisné šířky
      zemDelka = gps.location.lng(); //získání zeměpisné délky
      souradnice = (String(zemSirka, 6) + ',' + String(zemDelka, 6));
      delay(1000);
    }
  }
```

Zdrojový kód 20: Získání souřadnic

```
void vypisSouradnic() {
  bluetooth.println(souradnice);
  delay(60000);
}
```

Zdrojový kód 21: Odeslání souřadnic

Uživatel stiskem na tlačítko ovladače začne zaznamenávat souřadnice jízdy na SD kartu, do souboru souradnice.csv v metodě zapisNaSD (Zdrojový kód 22), ve stejném formátu, jako se posílají přes Bluetooth k ovladači.

```
void zapisNaSD() {
  soubor = SD.open("souradnice.csv", FILE_WRITE); //otevření souboru pro
zápis
  if (soubor) { //pokud se povedlo otevřít soubor
    soubor.print(souradnice); //zápis souřadnic
    soubor.print("\n"); //odřádkování, musí zde být tento znak
  }
}
```

Zdrojový kód 22: Zápis souřadnic na SD kartu

V případě otočení nápravy o stanovený úhel dochází k přičtení koeficientu k úhlu otočení, konkrétně o sto pro zatočení doleva a dvě stě pro zatočení doprava. Ve firmwaru dochází na základě koeficientu k volání patřičné metody (Zdrojový kód 23).

```
  //jakmile došel příkaz z aplikace o otočení o určitý úhel dochází
k zavolání těchto metod
  if (hodnotaBluetooth >= 100 && hodnotaBluetooth <= 120) { //první znak
znázorňuje směr otáčení, 1 doleva zbylé dva znaky znázorňují úhel
    zabocAnalogoveVlevo();
  }
  if (hodnotaBluetooth >= 200 && hodnotaBluetooth <= 220) { //první znak
znázorňuje směr otáčení, 2 doprava zbylé dva znaky znázorňují úhel
    zabocAnalogovePravo();
  }
```

Zdrojový kód 23: Volání metod pro otočení o zadaný úhel

Uvnitř těchto metod dochází k odečtení koeficientu od přijaté Bluetooth hodnoty a následně se rozdíl použije coby úhel natočení (Zdrojový kód 24).

```
void zabocAnalogoveVlevo() {
  int pomocna = hodnotaBluetooth - 100; //z poslané hodnoty se získá
o kolik stupňů se má natočit náprava
```

```
  servoNaprava_pozice = servoNaprava_pozice - pomocna; //natočení nápravy o daný úhel
  if (servoNaprava_pozice > 70) { //aby nedošlo k otočení nápravy přes povolenou hodnotu
    servoNaprava.write(servoNaprava_pozice); //na jakém úhlu se má náprava zastavit
    delay(5);
  }
} //konec metody zabocAnalogoveVlevo
void zabocAnalogovePravo() {
  int pomocna = hodnotaBluetooth - 200; //z poslané hodnoty se získá o kolik stupňů se má natočit náprava
  servoNaprava_pozice = servoNaprava_pozice + pomocna;
  if (servoNaprava_pozice < 110) { //aby nedošlo k otočení nápravy přes povolenou hodnotu
    servoNaprava.write(servoNaprava_pozice); //na jakém úhlu se má náprava zastavit
    delay(5);
  }
} //konec metody zabocAnalogovePravo
```

Zdrojový kód 24: Metody pro otočení nápravy o daný úhel

Kompletní firmware řídící jednotky auta se nalézá v přílohách (Příloha B).

4.3.3 Sestavení auta

4.3.3.1 Seznam použitých dílů

4.3.3.1.1 Motor shield Rev3

Arduino shield funguje na bázi H můstku pro ovládání až dvou motorů. H můstek se skládá z tranzistorů s cílem přepólování motorů pro určení směru otáčení.

4.3.3.1.2 Bluetooth modul HC 5

Bluetooth modul, slouží ke komunikaci s řídící jednotkou prostřednictvím bezdrátové technologie Bluetooth, konkrétně s verzí 4.0 za použití TTL příkazů.

4.3.3.1.3 GPS modul Neo 6M

Modul určený pro zjištění přesné pozice auta, konkrétně zeměpisné délky a šířky, navíc lze pomocí tohoto modulu určit také rychlost pohybu a nadmořskou výšku.

4.3.3.1.4 Modul SD karty

Modul sloužící k ukládání dat do souboru umístěném na SD kartě.

4.3.3.1.5 Podvozek se servomotorem

Speciální podvozek, do kterého se zašroubuje servomotor umožňující zabočení auta do požadovaného směru.

4.3.3.1.6 Napájení

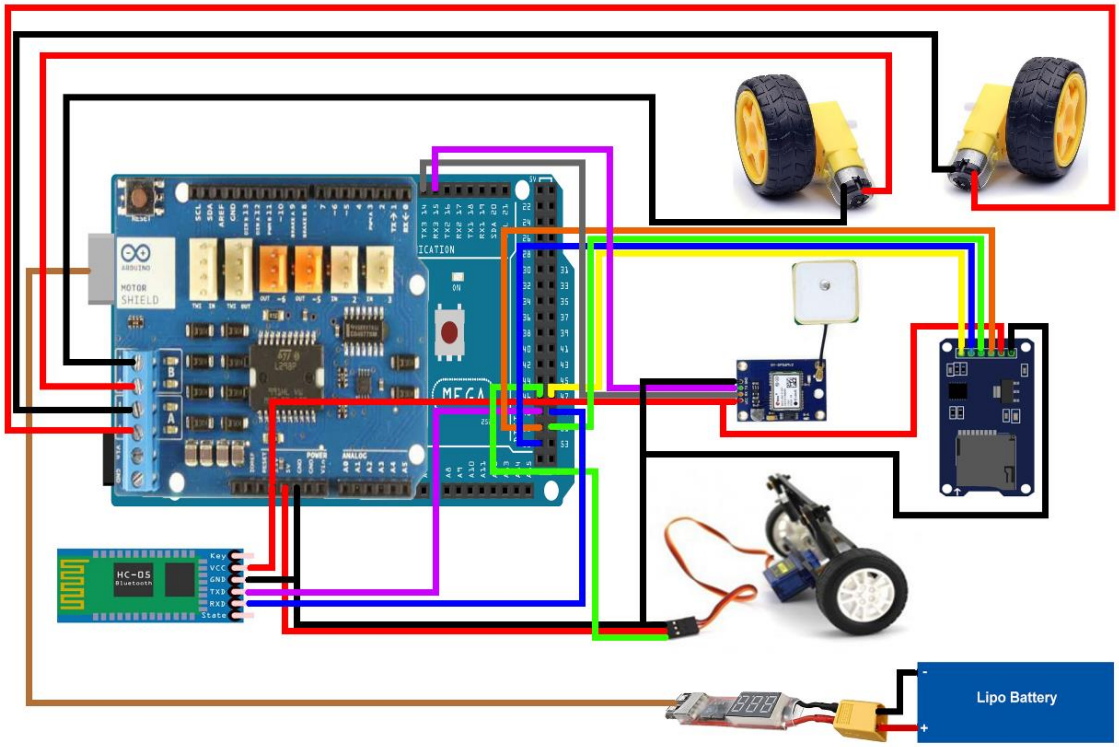
Model se napájí pomocí Li–Po akumulátoru, pro připojení akumulátoru k řídící jednotce se musí použít redukce konektoru XT60 na USB. Řídící jednotku lze napájet i použitím powerbanky.

4.3.3.2 Schéma zapojení

Schéma zapojení modelu využitý v rámci diplomové práce lze nalézt v následujícím přehledu (Tabulka 4) včetně názorného diagramu (Obrázek 20).

Zařízení	**Druh výstupu/vstupu**	**Barva**	**Číslo portu Arduina**
Bluetooth	GND (zem)	Černá	GND
Bluetooth	VCC (fáze)	Červená	3,5 V
Bluetooth	RXD (příjem dat)	Modrá	49 (digitální)
Bluetooth	TXD (odeslání dat)	Fialová	48 (digitální)
Náprava	Fáze (červená)	Červená	5V
Náprava	Zem (hnědá)	Černá	GND
Náprava	Komunikace	Zelená	46 (digitální)
GPS senzor	GND (zem)	Černá	GND
GPS senzor	VCC (fáze)	Červená	3,5 V
GPS senzor	RXD (příjem dat)	Šedá	14 (Serial 3)
GPS senzor	TXD (odeslání dat)	Fialová	15(Serial 3)
Modul SD karty	GND (zem)	Černá	GND
Modul SD karty	VCC (fáze)	Červená	3,5 V
Modul SD karty	CS (datová)	Žlutá	47 (digitální)
Modul SD karty	MISO (sběrnice)	Oranžová	50 (digitální, SPI)
Modul SD karty	MOSI (sběrnice)	Zelená	51 (digitální, SPI)
Modul SD karty	SCK (sběrnice)	Modrá	52 (digitální, SPI)

Tabulka 4: Zapojení modelu

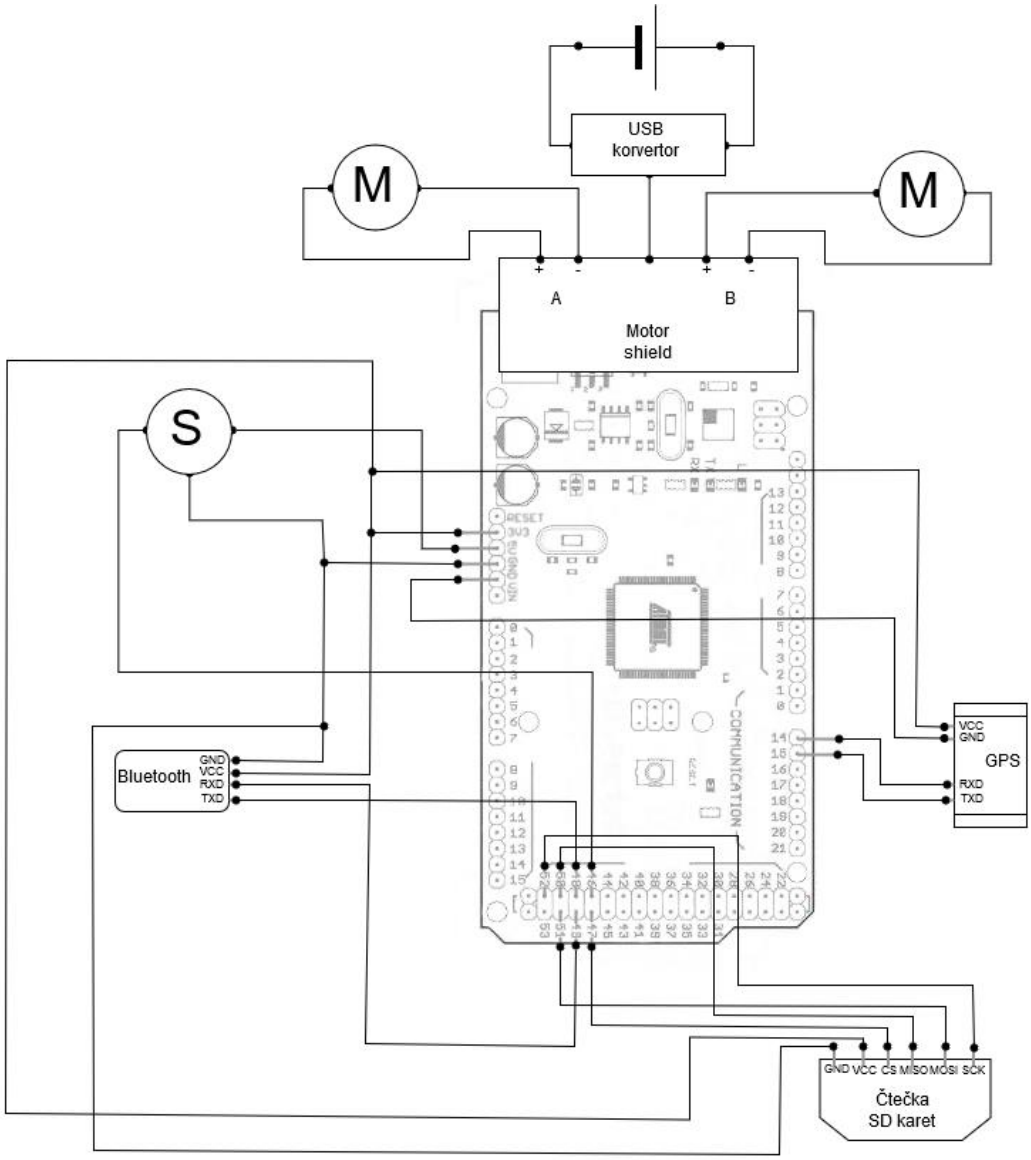


Obrázek 20: Diagram zapojení

Červená barva slouží pro napájení neboli +, černá pro zem neboli -. Ostatní barvy slouží pro přenos dat. Napájení probíhá od Li-Po baterie k převodníku X60 konektoru na USB a od něj pak USB kabelem k Arduino a následně na ostatní komponenty, v diagramu je USB kabel značen hnědou barvou, místo Li-Po baterie lze použít powerbanku.

Motory jsou připojeny k Arduinu pomocí motor shieldu v pořadí fáze motoru ke svorkovnici „+“ na shieldu a zem ke svorkovnici „–“ na shieldu, otočením pořadí by došlo k změně otáčení motoru. Motor shield se napájí z Arduina.

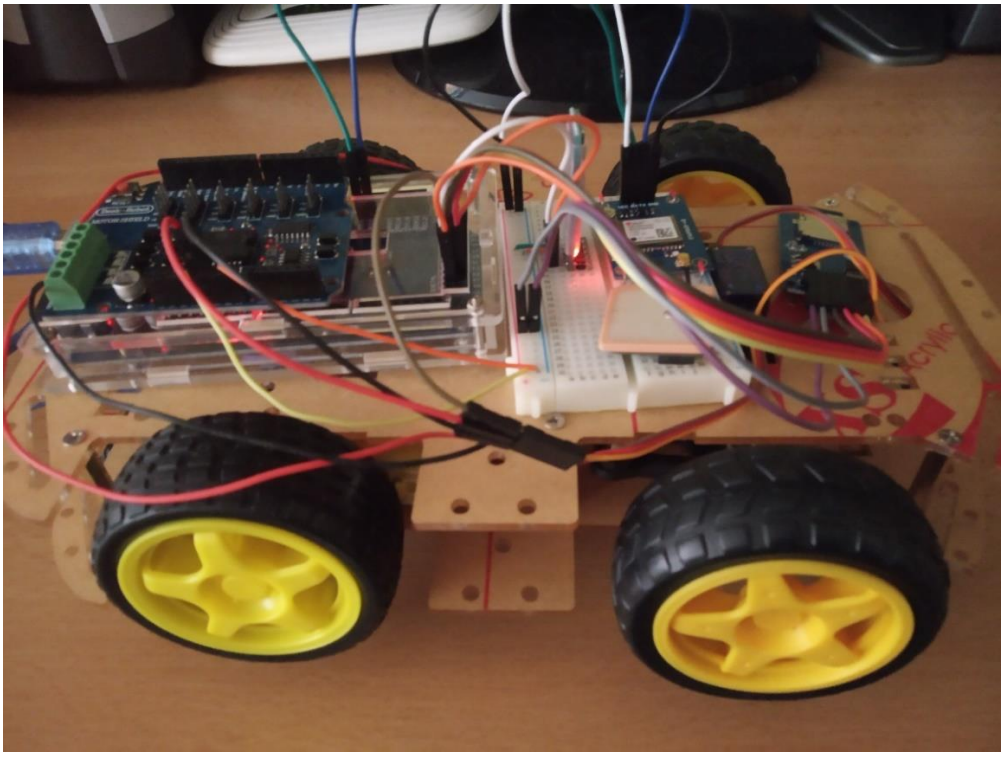
Zapojení modelu lze vidět také v elektrickém schématu (Obrázek 21).



Obrázek 21: Elektrické schéma zapojení

4.3.3.3 Foto auta

Sestavený model auta lze vidět v následujícím obrázku (Obrázek 22), další obrázek se nachází ve složce ScreenShots v přiloženém DVD.



Obrázek 22: Sestavené auto

4.4 Testování

Pro testování byl použit tento postup. Nejdříve se ozkoušel firmware mikrořadiče Arduino pro kontrolu správnosti výpisů hodnot ze senzorů přes sériový monitor v prostředí Arduino IDE. Posléze se přešlo na kontrolu aplikace, kde nejprve došlo k testování jednotlivých aktivit, bez připojení ovladače k autu přes Bluetooth, k testování propojení došlo až později. Jakmile bylo propojení úspěšné došlo k testování komunikace mezi řídicí jednotkou a mobilním telefonem. Konkrétně proběhlo testování funkčnosti motorů a zpětného odesílání dat ze senzoru GPS s následným zobrazením souřadnic v mobilní aplikaci.

4.5 Srovnání výsledné aplikace s Arduino Bluetooth RC Car

Výslednou aplikaci jsem se rozhodl srovnat s aplikací Arduino Bluetooth RC Car. Obě aplikace mají společné použití programovacího jazyka Java a komunikačního systému Bluetooth.

Obě aplikace se však v některých bodech liší. Konkurenční aplikace využívá všechny funkcionality v jednom okně, což se mi jevilo jako nepraktické řešení, neboť všechny komponenty zabírají mnoho místa a může tak snadno dojít k překlepu. Další rozdíl je ve způsobu ovládání. Ve své aplikaci místo šípek pro ovládání používám joysticky. Pro zatáčení

využívám speciální nápravy se zabudovaným servomotorem, tedy nevyužívám principu zastavení motoru jedné strany pro daný směr jízdy, kde druhý motor auta, který se stále pohybuje, dotlačí auto do požadovaného směru.

Aplikace Droptík navíc umožnuje vypsat souřadnice auta z modulu GPS a ty pak zobrazit v připravených mapách, kde se zobrazuje i aktuální poloha mobilního ovladače. Navíc zde dochází k ukládání souřadnic auta do souboru a zobrazení souřadnic daného souboru v Google mapách aplikace.

5 Výsledky

Výsledkem této práce je naprogramovaná aplikace pro zařízení se systémem Android umožňující ovládání auta s řídící jednotkou Arduino.

Aplikace pracuje coby virtuální ovládač, kde uživatel může pomocí Bluetooth ovládat auto. Auto umožňuje pomocí Arduino shieldu jízdu vpřed a vzad, pro zatáčení slouží speciální náprava se servomotorem a směr je dán náklonem této nápravy. Nápravu lze otáčet i o předem zadaný úhel. Aplikace umožnuje vypsání aktuální polohy auta a tu vyobrazit v Google mapách, tudíž během řízení nemusí být auto zrovna na očích, společně s lokalitou mobilního ovladače. Souřadnice se dále mohou ukládat do souboru na SD kartě, při načtení tohoto souboru zobrazit v mapách ujetou trasu auta. Oproti konkurenci má aplikace jednoduché uživatelské rozraní, ale výhodou je snadné ovládání, neboť pro jízdu se používají jednoduché joysticky jeden pro jízdu vpřed a vzad a druhý k otáčení nápravy. Výhodou také je, že se aplikace dodává i se zdrojovým kódem ve vyšším programovacím jazyce a patřičnou dokumentací. Jako nevýhodu považuji ovládání pomocí Bluetooth z důvodu dosahu signálu, který je omezen na okruh padesáti metrů kolem mobilního telefonu. Toto omezení lze z části kompenzovat s využitím Bluetooth verze 4.0, kde výrobci slibují dosah i do sta metrů.

K aplikaci je dále naprogramován projekt pro mikrořadič Arduino, který umožňuje provádět příkazy z aplikace pro řízení auta a odeslání GPS souřadnic.

Pro budoucí vývoj plánuji nahradit řídící jednotku Arduino za Raspberry Pi, která s využitím přizpůsobené kamery, umožňuje živý přenos záznamů za použití WiFi do PC, společně s implementací autonomní jízdy.

6 Závěr

Cílem diplomové práce bylo naprogramovat aplikaci na ovládání auta pro systém Android. Aplikace pracuje coby virtuální ovladač pro bezdrátové ovládání.

Aplikace byla naprogramovaná pomocí vývojového prostředí Android studio využívající jazyka Java. K aplikaci byl dále naprogramován firmware pro řídící jednotku Arduino ve vývojovém prostředí Arduino IDE s rozšířeným jazykem C.

V teoretické části se práce zabývá metodikami vývoje softwaru. Nachází se zde obecný postup vývoje softwaru, včetně vysvětlení jednotlivých kroků. Dále je zde napsán úvod do UML, který slouží pro snadnější pochopení výsledné aplikace a pro ukázku struktury jednotlivých částí. Uvádí se zde také základní informace pro naprogramování aplikace systému Android, včetně základních součástí aplikace, společně se souhrnem jednotlivých nejpoužívanějších vývojových prostředí. Konec teoretické části se věnuje mikrořadiči Arduino, způsobu jeho programování v prostředí Arduino IDE.

Praktická část postupuje jednotlivými kroky vývoje softwaru, konkrétně analýzou požadavků výsledné aplikace, návrhem aplikace ze získaných požadavků, realizací vývoje samotné aplikace ve vývojovém prostředí Android Studio společně s programováním firmwaru řídící jednotky. Po skončení vývoje došlo k testovaní jak firmwaru, tak samotné aplikace. Konec praktické části se věnuje porovnání výsledné aplikace s konkurenční, která je nyní k dispozici.

7 Seznam použitých zdrojů

Publikace

Android Developers. *Android Developers* [online]. Dostupné z: https://developer.android.com/

BUCHALCEVOVÁ, A. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky.* 1. vyd. Praha: Grada, 2005. 163 s. ISBN 978-80-247-1075-7.

HEROUT, Pavel. *Učebnice jazyka Java. 5., rozš. vyd.* České Budějovice: Kopp, 2010. ISBN 9788072323982.

LACKO, Ľuboslav. *Vývoj aplikací pro Android.* Brno: Computer Press, 2015. ISBN 9788025143476.

MONK, Simon. *Arduino + Android projects for the evil genius: control Arduino with your smartphone or tablet.* New York: McGraw-Hill, c2012. ISBN 978-0071775960.

VODA, Zbyšek. *Průvodce světem Arduina.* Vydání druhé. Bučovice: Martin Stříž, 2017. ISBN 9788087106938.

Elektronické zdroje

Agilní vývoj není (jenom) Scrum a Kanbanl. *Jiří Knesl* [online]. [cit. 10.02.2019]. Dostupné z: http://www.knesl.com/agilni-vyvoj-neni-jen-scrum-nebo-kanban

Android (operační systém). *Wikipedie* [online]. [cit. 20.12.2018]. Dostupné z: https://cs.wikipedia.org/wiki/Android

Android Studio. *Wikipedie* [online]. [cit. 20.12.2018]. Dostupné z: https://cs.wikipedia.org/wiki/Android_Studio

Android. *Mobilizujeme.cz* [online]. [cit. 17.02.2019]. Dostupné z: https://mobilizujeme.cz/stitky/android

Application Fundamentals. *Android Developers* [online]. [cit. 20.12.2018]. Dostupné z: https://developer.android.com/guide/components/fundamentals

Arduino Bluetooth RC Car. *Ardumotive Arduino Greek Playground* [online]. [cit. 05.03.2019]. Dostupné z: https://www.ardumotive.com/bluetooth-rc-car.html

Arduino Mega 2560 Rev3. *Arduino* [online]. [cit. 03.01.2019]. Dostupné z: https://store.arduino.cc/mega-2560-r3

Co je Scrum? *Jiří Knesl* [online]. [cit. 10.02.2019]. Dostupné z: http://www.knesl.com/co-je-scrum

Explore MIT App Inventor. *MIT App Inventor* [online]. [cit. 5.3.2019]. Dostupné z: https://appinventor.mit.edu/

Extrémní programování (XP). *MBI - Management Byznys* [online]. [cit. 10.02.2019] Dostupné z: https://mbi.vse.cz/public/cs/obj/METHOD-92

IntelliJ IDEA. *Wikipedie* [online]. [cit. 20.12.2018]. Dostupné z: https://cs.wikipedia.org/wiki/IntelliJ_IDEA

Manifest Agilního vývoje software. *Agilemanifesto* [online]. [cit. 10.02.2019]. Dostupné z: http://agilemanifesto.org/iso/cs/manifesto.html

Metodiky vývoje softwaru. *Moravská vysoká škola Olomouc* [online]. [cit. 10.02.2019]. Dostupné z: https://mvso.cz/wp-content/uploads/2018/02/Metodiky-v%c3%bdvoje-software-studijn%c3%ad-text.pdf

Programujeme Arduino. *Arduino.cz* [online]. [cit. 03.01.2019]. Dostupné z: https://arduino.cz/programujeme-arduino/

Projects overview. *Android Developers* [online]. [cit. 02. 01.2019]. Dostupné z: https://developer.android.com/studio/projects/

Robot Kits for Kids : mBot. *Makeblock: Global STEAM Education Solution Provider* [online]. [cit. 05.03.2019]. Dostupné z: https://makeblock.com/steam-kits/mbot

Seznámení s Arduinem. *itnetwork.cz* [online]. [cit. 03.01.2019]. Dostupné z: https://www.itnetwork.cz/hardware-pc/arduino/arduino-seznameni

UML - Class diagram. *itnetwork.cz* [online]. [cit. 05.05.2019]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model

UML - Use Case Diagram. *itnetwork.cz* [online]. [cit. 05.02.2019]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

UML - Sequence diagram. *itnetwork.cz* [online]. [cit. 05.02.2019]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-sequence-diagram

Úvod do UML. *itnetwork.cz* [online]. [cit. 05.02.2019]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

8 Přílohy

8.1 Příloha A: Obsah disku

Název složky	Popis
dokumentace	Složka obsahující elektronickou verzi diplomové práce.
prilohy	Přílohy obsahují návod na spuštění projektu pro Android studio a Arduina společně s instalací aplikace na mobilní telefon. Dále složka obsahuje schéma zapojení auta a obrázky aplikace.
uml diagramy	Složka obsahuje UML diagramy ve formátu uxf a png.
zdrojove kody	Obsahuje zdrojové kódy aplikace společně s Arduino projektem. Ve složce se nacházejí i instalační soubory vývojových prostředí použitých při vypracování diplomové práce.

Tabulka 5: Obsah disku

8.2 Příloha B: Firmware řídící jednotky

```
//Pro auto bylo použito Arduino MEGA
//Bluetooth se musí nejdříve připojit přes správu androidu zařízení
s odpovídající MAC adresou Bluetooth modulu, heslo pro spárování je 1234,
pak teprve připojit aplikaci
#include <NeoSWSerial.h> //knihovna pro komunikaci po sériové lince
#include <Servo.h> //servo knihovna
#include <TinyGPS++.h> //knihovna pro GPS modul
#include <SPI.h> //knihovna pro SPI sběrnici
#include <SD.h> //knihovna pro SD modul

//definovaní globálních proměnných
Servo servoNaprava; //servo pro zadní nápravu udávající směr

/*
  připojeni motoru do motorShiledu v pořadí:
  směr pravého motoru, směr levého motoru, rychlost pravého motoru,
  rychlost levého motoru, měřič proudu analog A0, A1, vpřed, vzad
*/
int smerPrava = 12;
int smerLeva = 13;
int rychlostPrava = 3;
int rychlostLeva = 11;
int proudPrava = A0;
int proudLeva = A1;

TinyGPSPlus gps; //TinyGPS++ objekt
double zemSirka, zemDelka;
String souradnice;

int bluetoothTx = 48; //bluetooth TX do pinu 48
int bluetoothRx = 49; //bluetooth RX do pinu 49
NeoSWSerial bluetooth(bluetoothTx, bluetoothRx);
int hodnotaBluetooth;

int servoNaprava_pozice = 90; //výchozí natočení servo motoru

File soubor; //definování souboru
int pinSS = 53; //SS pin SD modulu
int pinCS = 47; //CS pin SD modulu

void setup() {
  servoNaprava.attach(46); //připojeni serva pro nápravu na port 46
  servoNaprava.write(servoNaprava_pozice); //výchozí stav otočení servo
motoru nápravy

  Serial.begin(9600);
  Serial3.begin(9600); //15(GPSTX), 14(GPSRX) Arduino Mega porty použity
pro GPS
  bluetooth.begin(9600);

  //nastavení výstopu motoru
  pinMode(smerPrava, OUTPUT);
  pinMode(smerLeva, OUTPUT);

//nastavení výstupu SD modulu
  pinMode(pinSS, OUTPUT); //tento pin se nebude používat, ale musí být
definován
} //konec metody setup
```

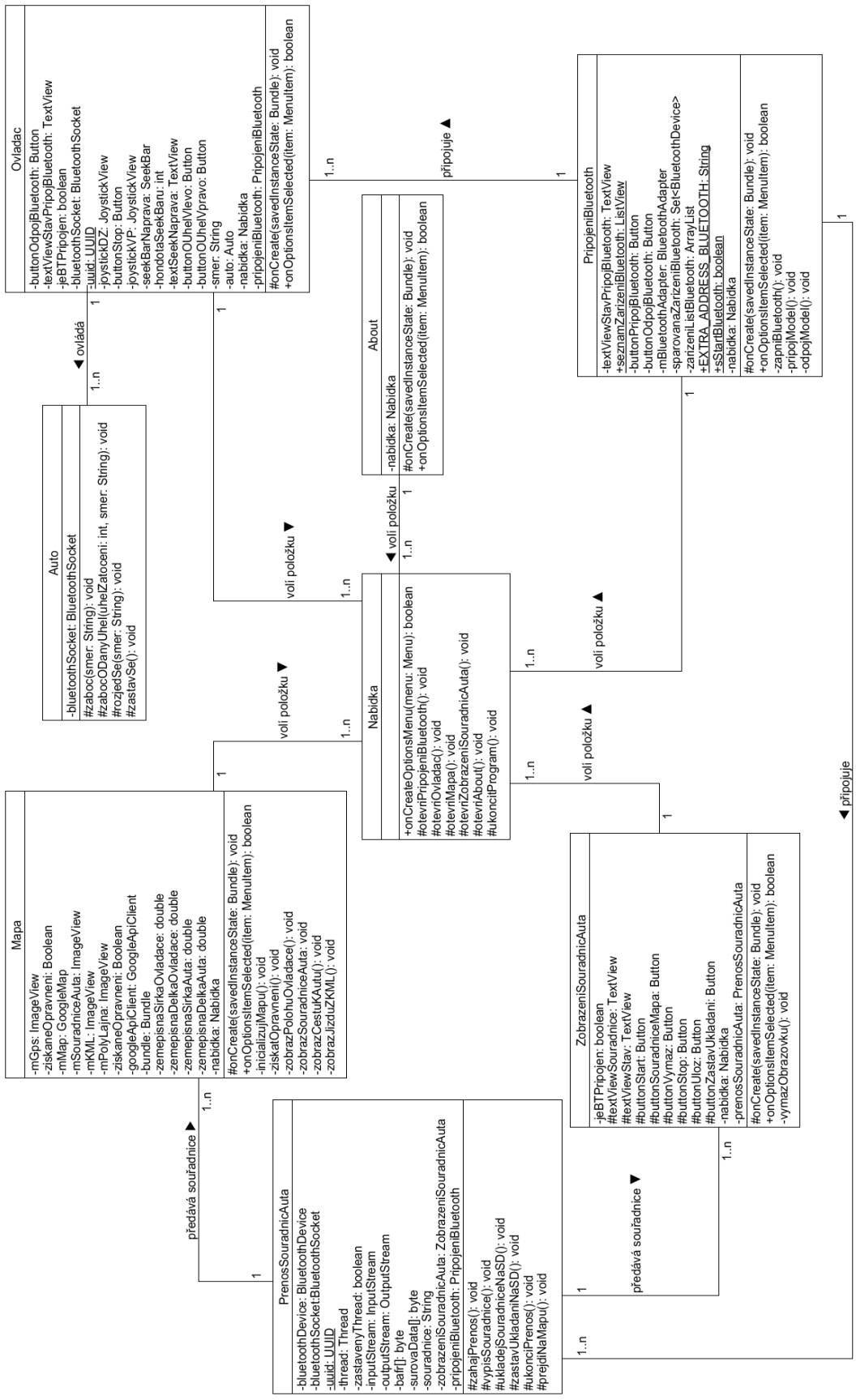
```
void loop() {
  //pokud z GPS přicházejí nějaká data, posli je do knihovny TinyGPS++
  while (Serial3.available() > 0) {
    if (gps.encode(Serial3.read())) {
      zemSirka = gps.location.lat(); //získání zeměpisné šířky pomocí
metody knihovny TinyGPS++
      zemDelka = gps.location.lng(); //získání zeměpisné délky pomocí
metody knihovny TinyGPS++
      souradnice = (String(zemSirka, 6) + ',' + String(zemDelka, 6));
//skládání souřadnice auta do požadovaného formátu na 6 desetinných míst
      delay(1000);
    }
    if (SD.begin(pinCS)) { //pokud se načetl SD modul
      zapisNaSD();
    }
  } //konec GPS ověření

  //přečti hodnotu z bluetooth
  if (bluetooth.available() > 0 ) { //pokud přišla hodnota k Bluetooth
    hodnotaBluetooth = bluetooth.read(); //čtení hodnoty Bluetooth
  } //konec Bluetooth ověření

  //na základě přijatého příkazu proveď požadovanou operaci
  switch (hodnotaBluetooth) {
    case 1: //jakmile dojde k načtení příkazu pro pomalé rozjíždění
      pomaleRozjizdeni(); //volaní metody pro rozjíždění
      break;
    case 2: //jakmile dojde k načtení příkazu pomalé couvání
      pomaleCouvani(); //volaní metody pro pomalé couvání
      break;
    case 3: //jakmile dojde k načtení příkazu pro couvání zapni oba
motory
      dozadu(); //volaní metody vzad
      break;
    case 4:  //jakmile dojde k načtení příkazu pro jízdu vpřed
      dopredu(); //volaní metody vpřed
      break;
    case 5: //jakmile dojde k načtení příkazu stop
      zastav(); //volaní metody zastav
      break;
    case 6: //jakmile doje k načtení příkazu pro otočení doprava
      doprava(); //volaní metody doprava
      break;
    case 7: //jakmile doje k načtení příkazu pro otočení doleva
      doleva(); //volaní metody doleva
      break;
    case 8: //jakmile dojde k načtení příkazu odeslání souřadnic
      vypisSouradnic(); //volaní metody vypisSouradnic
      break;
    case 9: //jakmile dojde k načtení příkazu k ukládání souřadnic na SD
kartu
      if (SD.begin(pinCS)) { //pokud se načetl SD modul
        zapisNaSD();
      }
      break;
    case 10: //jakmile dojde k načtení příkazu k ukončení ukládání
souřadnic na SD kartu
      if (SD.begin(pinCS)) { //pokud se načetl SD modul
        ukonciZapisNaSD();
      }
      break;
```

```
  } //konec přepínače

  //jakmile došel příkaz z aplikace o otočení o určitý úhel dochází
k zavolání těchto metod
  if (hodnotaBluetooth >= 100 && hodnotaBluetooth <= 120) { //první znak
znázorňuje směr otáčení, 1 doleva zbylé dva znaky znázorňují úhel
    zabocAnalogoveVlevo();
  }
  if (hodnotaBluetooth >= 200 && hodnotaBluetooth <= 220) { //první znak
znázorňuje směr otáčení, 2 doprava zbylé dva znaky znázorňují úhel
    zabocAnalogovePravo();
  }
} //konec metody loop

void pomaleRozjizdeni() {
  digitalWrite(smerPrava, HIGH); //zapni motor pro jízdu vpřed, HIGH
  digitalWrite(smerLeva, HIGH);
  analogWrite(rychlostPrava, 190);
  analogWrite(rychlostLeva, 190);
  delay(300);
} //konec metody pomaleRozjizdeni

void pomaleCouvani() {
  digitalWrite(smerPrava, LOW); //zapni motor pro jízdu vzad, LOW
  digitalWrite(smerLeva, LOW);
  analogWrite(rychlostPrava, 190);
  analogWrite(rychlostLeva, 190);
  delay(300);
} //konec metody pomaleCouvani

void dopredu() {
  digitalWrite(smerPrava, HIGH); //zapni motor pro jízdu vpřeď, HIGH
  digitalWrite(smerLeva, HIGH);
  analogWrite(rychlostPrava, 255); //nastaveni rychlosti na max
  analogWrite(rychlostLeva, 255);
  delay(300);
} //konec metody dopredu

void dozadu() {
  digitalWrite(smerPrava, LOW); //zapni motor pro jízdu vzad, LOW
  digitalWrite(smerLeva, LOW);
  analogWrite(rychlostPrava, 255); //nastaveni rychlosti couvaní na max
  analogWrite(rychlostLeva, 255);
  delay(300);
} //konec metody dozadu

void zastav() {
  digitalWrite(smerPrava, LOW); //zastav motor
  digitalWrite(smerLeva, LOW);
  analogWrite(rychlostPrava, 0); //vynuluj rychlost
  analogWrite(rychlostLeva, 0);
  delay(300);
} //konec metody zastavení

void doprava() {
  if (servoNaprava_pozice <= 110) { //otočí servomotor lze jen od 90 do
110 stupňů, 110 stupňů je zvoleno z důvodu konstrukce auta, která
nenabízí větší otočení nápravy doprava
    servoNaprava_pozice += 1; //přičtení 1 stupně k servomotoru
    servoNaprava.write(servoNaprava_pozice);
    delay(5);
```

```
  }
} //konec metody doprava

void doleva() {
  if (servoNaprava_pozice >= 70) { //otočí servomotor lze jen od 90 do
70 stupňů, 70 stupňů je zvoleno z důvodu konstrukce auta, která nenabízí
větší otočení nápravy doleva
    servoNaprava_pozice -= 1; //odečtení 1 stupně k servomotoru
    servoNaprava.write(servoNaprava_pozice);
    delay(5);
  }
} //konec metody doleva

void vypisSouradnic() {
  bluetooth.println(souradnice);
  delay(60000);
} //konec metody vypisSouradnic

void zapisNaSD() {
  soubor = SD.open("souradnice.csv", FILE_WRITE); //otevření souboru pro
zápis
  if (soubor) { //pokud se povedlo otevřít soubor
    soubor.print(souradnice); //zápis souřadnic
    soubor.print("\n"); //odřádkování, musí zde být tento znak
  }
} //konec metody zapisNaSD

void ukonciZapisNaSD() {
  soubor.close(); //uzavření souboru
}// konec metody ukonciZapisNaSD

void zabocAnalogoveVlevo() {
  int pomocna = hodnotaBluetooth - 100; //z poslané hodnoty se získá
o kolik stupňů se má natočit náprava
  servoNaprava_pozice = servoNaprava_pozice - pomocna; //natočení nápravy
o daný úhel
  if (servoNaprava_pozice > 70) { //aby nedošlo k otočení nápravy přes
povolenou hodnotu
    servoNaprava.write(servoNaprava_pozice); //na jakém úhlu se má
náprava zastavit
    delay(5);
  }
} //konec metody zabocAnalogoveVlevo

void zabocAnalogovePravo() {
  int pomocna = hodnotaBluetooth - 200; //z poslané hodnoty se získá
o kolik stupňů se má natočit náprava
  servoNaprava_pozice = servoNaprava_pozice + pomocna;
  if (servoNaprava_pozice < 110) { //aby nedošlo k otočení nápravy přes
povolenou hodnotu
    servoNaprava.write(servoNaprava_pozice); //na jakém úhlu se má
náprava zastavit
    delay(5);
  }
} //konec metody zabocAnalogovePravo
```

Zdrojový kód 25: Firmware řídící jednotky

8.3 Příloha C: Diagram tříd



Obrázek 23: Diagram tříd