UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2017 Daniel Míček

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Dálkové řízení modelu osobního automobilu se senzory

Daniel Míček

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Daniel Míček**
Osobní číslo:	**I14143**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Dálkové řízení modelu osobního automobilu se senzory**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit model osobního automobilu z vybrané stavebnice (např. Lego, Merkur apod.), toto vozidlo vybavit základními senzory pro implementaci základní inteligence (např. zastavení před detekovanou překážkou) a dále navrhnout a implementovat dálkové řízení tohoto vozidla pomocí mobilního telefonu nebo tabletu.

Jako řídící vozidla je doporučena jednotka Lego EV3 a doporučeným programovacím jazykem je vyšší programovací jazyk Java.

Text bakalářské práce bude obsahovat princip funkčnosti vybraných senzorů, popis programových prostředků pro jejich obsluhu a také popis řešení této úlohy.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. BAGNALL, Brian. Intelligence unleashed: creating LEGO NXT robots with Java. Winnipeg, Manitoba: Variant Press. ISBN 978-098-6832-208
2. KISZKA B. 1001 tipů a triků pro jazyk Java. Computer Press, 2009. ISBN 9788025124673.

Vedoucí bakalářské práce: **Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2016**
Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Mgr. Josef Horálek, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 10. 05. 2017

Daniel Míček

PODĚKOVÁNÍ

Na tomto místě bych chtěl poděkovat panu Ing. Michaelu Bažantovi, Ph.D. za poskytnuté materiály a pomoc při řešení problémů. Dále bych chtěl poděkovat své přítelkyni a rodině za podporu.

ANOTACE

Cílem bakalářské práce je sestrojit robota s využitím stavebnice LEGO Mindstorms EV3, který komunikuje s mobilním telefonem s operačním systémem Android pomocí Bluetooth. Robot využívá senzorů vzdálenosti, kompasu a úhlové rychlosti a zobrazuje měřená data v mobilním telefonu uživatele. Programovacím jazykem robota je Java.

KLÍČOVÁ SLOVA

Robot, lejos, ev3, lego mindstorms, android, bluetooth, senzor

TITLE

Lego robot controlled by smartphone.

ANNOTATION

The goal of this thesis is to create a robot using LEGO Mindstorms EV3, which is communicating with an Android smartphone using Bluetooth. The robot is using a compass, a distance and a angle sensor. The data measured by the sensors are sent to the phone and are presented to the user. Java is chosen as the programming language for this task.

KEYWORDS

Robot, lejos, ev3, lego mindstorms, android, bluetooth, sensor

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

GUI	Graphical User Interface
JRE	Java Runtime Environment
JVM	Java Virtual Machine
PAN	Personal Area Network
RCX	Robotic Command eXplorers
RNDIS	Remote Network Driver Interface Specification
SDK	Software Development Kit
SIG	Bluetooth Special Interest Group
SPP	Serial Port Profile
VM	Virtual machine (Virtuální stroj)

0 ÚVOD

V bakalářské práci *Dálkové řízení modelu osobního automobilu se senzory* se budu zaměřovat na možnosti využití LEGO Mindstorms pro vytvoření modelu auta ovládaného mobilním telefonem s operačním systémem Android za využití bezdrátové technologie Bluetooth. K programování automobilu využiji programovací jazyk Java na firmware leJOS. Android aplikace bude programována ve vývojovém prostředí Android Studio. Automobil bude využívat kompasu, vzdálenostního a úhlového senzoru a tří motorů. Dva velké motory, které budou použity pro pohon a jeden střední motor, který bude použit pro zatáčení.

V teoretické části je představeno LEGO Mindstorms, jako platforma pro tvorbu robotů a vzdělávání v oblasti robotiky, a jeho evoluce až k současné verzi EV3. Dále jsou představeny periferie pro tvorbu robotů této platformy, kterými jsou senzory a motory. Následuje kapitola popisující programovací možnosti pro vývojáře robotů. Samostatnou kapitolu tvoří náhradní firmware mého výběru, kterým je leJOS. V ní jsou popisovány kroky instalace na operačním systému Microsoft Windows 10 a vývoj programu robota v programovacím jazyce Java s využitím vývojového prostředí Eclipse. V neposlední řadě přestavím mobilní operační systém Android a vývoj aplikací pro tuto platformu ve vývojovém prostředí Android Studio. Posledním tématem teoretické části je bezdrátová komunikační technologie Bluetooth.

V praktické části bude detailně představeno zadání. Následuje programově popsané navázání a komunikování pomocí technologie Bluetooth ze strany jak EV3, tak Androidu. Dále je popsána struktura programu EV3Server, jeho třídy a jejich metody. V poslední části praktické části je zdokumentovaná aplikace EV3Client a její aktivity.

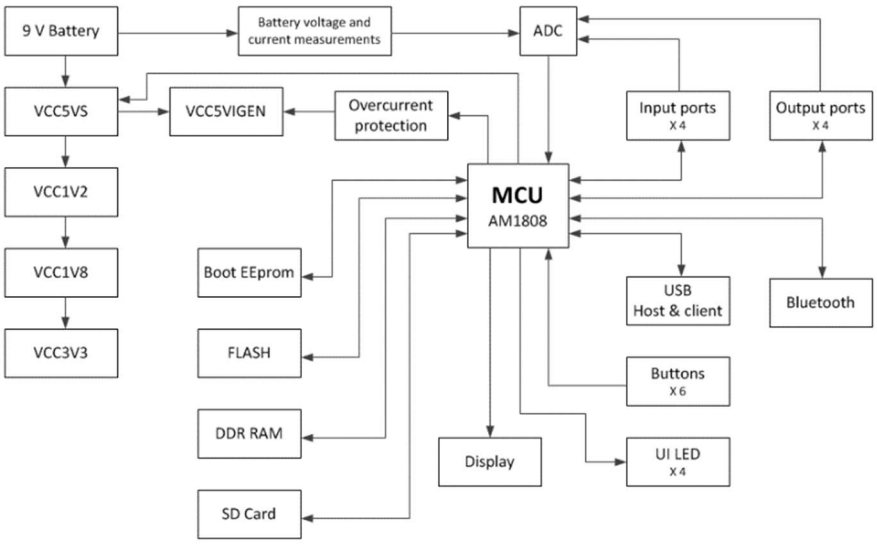
1 LEGO MINDSTORMS

LEGO Mindstorms je souprava programovatelných LEGO robotických stavebnic. Nejdůležitější část soupravy je programovatelná kostka, která ovládá motory a získává informace ze senzorů. Konstrukční část pak mají na starost součástky LEGO Technic.

První generace LEGO Mindstorms nazvaná RCX (zkratka Robotic Command eXplorers) vyšla v roce 1998. Centrální kostka RCX obsahuje 8-bitový Renesas H8/300 microcontroller procesor a 32K paměti RAM, která uchovávala firmware a uživatelské programy. Programování kostky probíhalo nahráním uživatelského programu do RAM přes speciální infračervený port, přes který mohou dvě kostky komunikovat i mezi sebou. Kromě IR portu RCX obsahovalo tři vstupní senzorové porty a tři výstupní porty, které řídí motory. Pro výstupní informace uživateli sloužil integrovaný LCD displej. [1]

Druhou generací bylo vydání s názvem NXT, které bylo dostupné ve dvou verzích – spotřebitelské balení a výuková verze. NXT vyšlo v roce 2006 a o tři roky později bylo nahrazeno verzí NXT 2.0. Balení obsahovalo kromě stavebních kostek také tři servo motory a sensor dotyku, světla, zvuku a vzdálenosti. Ovšem nejdůležitější součást balení byla programovatelná centrální kostka, vybavená 32-bitovým procesorem ARM7TDMI-core Atmel AT91SAM7S256 s 64 KB operační paměti RAM, 256 KB FLASH paměti pro lokální uchování programů. Programování kostky bylo primárně zprostředkováno pomocí grafického programovacího prostředí NXT-G, alternativně v software LabVIEW, na jehož základě bylo NXT-G vyvinuto. Oproti RCX se nyní senzory s kostkou propojovaly pomocí modifikovaného konektoru RJ12. Těchto portů NXT obsahovalo sedm, přičemž čtyři jako vstupní pro data ze senzorů a tři jako výstupní pro pohon servo motorů. Pro uživatelský výstup sloužila LCD obrazovka s rozlišením 100x60 pixelů a reproduktor. Přímý uživatelský vstup zprostředkovala čtyři tlačítka umístěná pod obrazovkou. [2]

LEGO Mindstorms EV3 je třetí generací stavebnic vydané 1. září 2013. EV3 je pokračování vývoje NXT, oproti kterému má několik zlepšení. Prvním z těchto zlepšení je slot na karty formátu Micro SDHC, který umožňuje bootování vlastního firmware z karty bez přepsání originálního. Dalším novou vlastností je hostitelský USB port, díky kterému je možné použít Wi-Fi přijímač a komunikovat s kostkou prostřednictvím Wi-Fi na delší vzdálenost, než nabízí Bluetooth. Počet vstupních portů zůstal stejný, ale přibyl jeden výstupní port. Konektorem portů zůstal RJ12, díky němuž je zachována přímá kompatibilita s NXT. Kostka také dostala robustnější hardware, a to 32-bitový ARM9 procesor Texas Instrument AM1808, 64 MB DDR RAM paměti, 16 MB FLASH paměti a 256 KB EEPROM paměti. I uživatelské rozhraní se dočkalo zlepšení v podobě většího černobílého LCD displeje s rozlišením 178x128 obrazových bodů a šest navigačních tlačítek. Jako NXT je i EV3 nabízeno ve více baleních, lišících se od sebe použitými senzory a konstrukčními bloky. [3][4]



Obrázek 1 – Blokový diagram EV3 kostky

Zdroj obrázku: LEGO MINDSTORMS EV3 Hardware Developer Kit

2 PERIFERIE EV3

2.1 Senzory

Spotřebitelská verze EV3 obsahuje čtyři senzory: Dotykový, infračervený, barevný a infračervený maják. Vzdělávací verze jich obsahuje 5: Ultrazvukový, barevný, gyroskop a dva dotykové. Kromě nich je ale možné využívat jak senzory starších verzí Mindstorms, se kterými je zpětně kompatibilní, tak vlastní senzory třetích stran, ze kterých je nejznámější řada HiTechnic. Pro jejich použití s oficiálním programovacím prostředím je nutné stáhnout konkrétní zásuvný modul z webových stránek výrobce. Senzory a motory se do kostky zapojují prostřednictvím modifikovaných kabelů RJ12, stejně jako u předchozího NXT. Pro použití senzorů RCX je nutné dokoupit převáděcí kabel. Některé ze senzorů mohou fungovat ve více módech, a tím nabízí rozšířené možnosti využití. Platforma EV3 podporuje funkci Auto-ID sloužící k identifikování periferií zapojených do vstupních i výstupních portů. Funkce Auto-ID podporuje všechny originální EV3 senzory a vybrané senzory NXT. [3]

2.1.1 Senzor dotyku

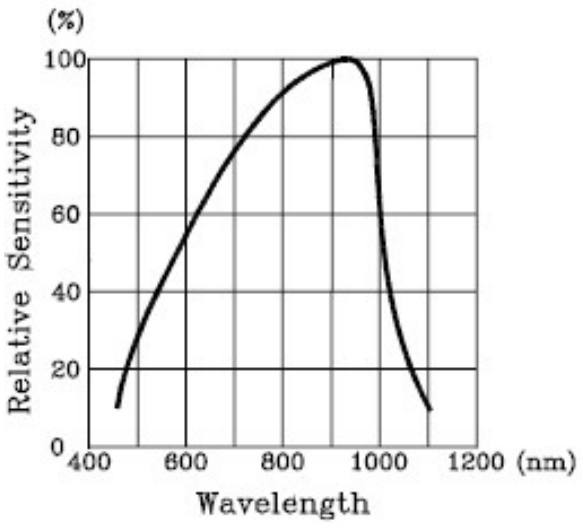
Senzor dotyku obsahuje analogový mechanický spínač, který lze programovat pomocí tří stavů: stisknutí, uvolnění nebo náraz (stisknutí a uvolnění). Možným využitím tohoto senzoru jsou tlačítka, či detektor kolize. [3]

2.1.2 Gyroskopický senzor

EV3 gyroskopický senzor je schopný snímat otáčení rychlostí až 440 stupňů za sekundu s přesností 3 stupňů na každých 90 stupňů. Obsahuje tři módy. Prvním z nich je mód měření relativního úhlu oproti poslednímu resetování nulového úhlu. Ačkoliv senzor tento mód nabízí, doporučuje se pro tuto aplikaci zvolit senzor kompas, který nabízí přesnější výsledky. Druhým módem je snímání současné úhlové rychlosti a posledním módem je možnost snímat obě předchozí hodnoty současně. Vhodné implementace gyroskopického senzoru jsou samo vyvažující roboti na principu kyvadla. [3]

2.1.3 Barevný senzor

Na rozdíl od NXT se barevný senzor u EV3 nachází v každém balení. NXT obsahovalo pouze světelný senzor a barevný bylo nutné samostatně dokoupit. Barevný senzor je schopný fungovat ve třech módech. Prvním z nich je měření okolního světla, při kterém se informační LED zbarví modře. Druhým je měření odraženého světla vyzařovaného červenou LED. Senzor vyruší jakékoliv vedlejší světlo pomocí rychlého blikání červeného podsvícení a odečtení naměřených hodnot s podsvícením i bez něj. Toto je prováděno 1000krát za vteřinu. Populární využití tohoto módu jsou roboti sledující černou linku. Třetím z nich je barevný mód, kdy senzor měří hodnoty intenzity odraženého světla jednotlivých světelných složek RGB modelu, čímž se nabízí využití jako skener, případně jako robot řešící Rubikův hlavolam. V kombinaci se standardním nemodifikovaným programovacím prostředím EV3 je senzor schopný rozlišit pouze 7 různých barev, kterými jsou černá, modrá, zelená, žlutá, červená, bílá a hnědá. Pro čtení intenzit jednotlivých složek RGB spektra je proto nutné použít dodatečný programovací blok nebo zvolit jiný programovací jazyk. Senzor je nejvíce citlivý na světlo o vlnové délce přibližně 900 nm a doporučuje se ho umístit zhruba centimetr od sledovaného povrchu. [3][5]

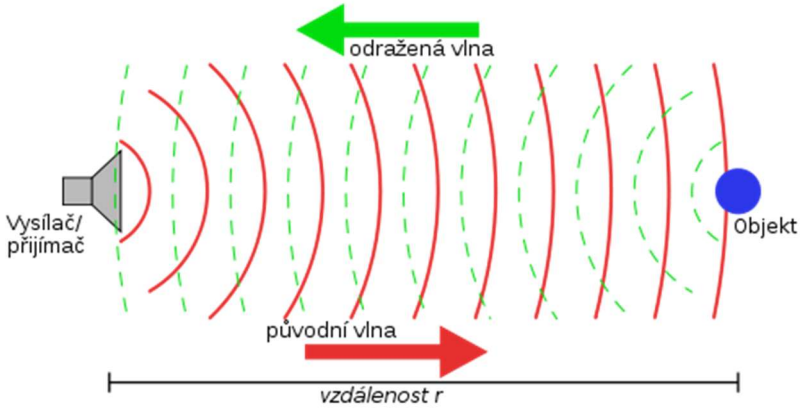


Obrázek 2 – Relativní senzitivita barevného senzoru

Zdroj obrázku: LEGO MINDSTORMS EV3 Hardware Developer Kit

2.1.4 Ultrazvukový senzor vzdálenosti

Jedním ze základních senzorů každého vydání Mindstorms je ultrazvukový vzdálenostní senzor. Senzor funguje na principu sonaru, což znamená, že vysílá ultrazvukové vlny a podle měření prodlevy jejich odrazu určuje vzdálenost předmětu, od kterého se vlny odrazily. Senzor dokáže měřit až na vzdálenost 250 cm s přesností 1 cm. Senzor je schopný snímat až 1000 hodnot za vteřinu a může fungovat ve třech módech. První i druhý nepřetržitě měří vzdálenost nejbližšího objektu pomocí vysílání a přijímání odražených ultrazvukových vln, lišící se pouze v jednotkách měření, kde první mód měří v centimetrech a druhý v palcích. Pokud je jeden z těchto módů aktivní, projevuje se stálým červeným podsvícením převodníků. Třetím módem je pouhé naslouchání vln vysílaných jiným ultrazvukovým senzorem. Tento mód lze poznat podle blikajícího podsvícení převodníků. [3]



Obrázek 3 – Princip sonaru

Zdroj obrázku: https://commons.wikimedia.org/wiki/File:Sonar_Principle-cs.svg

2.1.5 HiTechnic úhlový senzor

Úhlový senzor od společnosti HiTechnic umožňuje pomocí mechanizmu s nízkým třením měřit úhel a rychlost otáčení křížové osy. Jako i jiné senzory je schopný fungovat ve třech módech. Prvním z nich je měření absolutního úhlu od 0 do 359 stupňů s přesností na jeden stupeň. Dalším módem je měření akumulovaného úhlu vícero otočení od posledního resetování nulového úhlu. Třetím módem je měření úhlové rychlosti otáčení. [6]

2.1.6 HiTechnic kompas

Tento senzor obsahuje digitální magnetický kompas, který měří magnetické pole země a vypočítává úhel směru. Frekvence snímání pro tento senzor je 100 Hz. Kvůli principu magnetického kompasu podléhá tento senzor efektům elektromagnetickým polím komponentů, jako jsou baterie nebo elektromotory. Tyto negativní efekty se dají do určité míry negovat

pomocí vestavěné funkce kalibrace. Kalibrace se provádí pomalým otáčením celého robota alespoň o 720 stupňů. Pro minimalizování potřeby kalibrace se doporučuje umístit senzor alespoň 10 až 15 cm od programovatelné kostky či elektromotoru. [7]

2.2 Motory EV3

Oproti NXT, které bylo dodáváno se třemi velkými motory, balení EV3 obsahuje velké motory pouze dva. Třetí velký motor byl nahrazen jedním středním motorem. Oba motory obsahují tachometrickou funkcionalitu s přesností na jeden stupeň otočení, což umožňuje přesnější ovládání rychlosti motoru. Velký motor je schopný vyvinout až 175 otáček za minutu s kroutícím momentem 20 N/cm. Od svého předchůdce kostky NXT se liší minimálně, pouze konstrukčními úchyty a barevným schématem. Střední motor EV3 dokáže vyvinout rychlost otáčení až 250 otáček za minutu s kroutícím momentem 8 N/cm. Kromě výkonu se od velkého motoru liší ještě posazením otočné osy. Oba motory disponují funkcí Auto-ID, která slouží pro identifikaci zapojených komponent do EV3 kostky.

3 PROGRAMOVÁNÍ EV3

Jako NXT nabízí i EV3 mnoho možností pro vývojáře. Firmware EV3 je založen na Linuxu. Výchozím způsobem programování je grafická bloková aplikace LEGO Mindstorms EV3 Software, která je založena na software LabVIEW. Aplikace je dostupná na standardních platformách stolních počítačů, jako je Microsoft Windows a Mac. Aplikací lze kromě kostek EV3 do určité míry programovat i starší kostky NXT. LEGO však také poskytuje možnost programování EV3 robotů s využitím tabletů s operačními systémemy Android a iOS pomocí aplikace EV3 Programmer. Programování robotů je v obou případech realizováno přetáhnutím a puštěním programovacích bloků na programovací plátno. Mobilní aplikace se od aplikace pro stolní počítače liší pouze v několika zjednodušeních, jako například absence datových bloků a kalkulací. [8]

Další oficiální způsob programování EV3 je pomocí přídavného modulu pro software LabVIEW. LabVIEW je grafické vývojové prostředí pro návrh systému, které využívá grafického jazyka pojmenovaného G. Tato volba rozšiřuje možnosti programování, a zároveň neodebírá grafické rozhraní. [9]

Náročnější uživatelé se ovšem s grafickým prostředím nespokojí. Existuje proto několik možností, jak využít populární programovací jazyky při programování EV3 kostky. Jednou z těchto možností je upravený firmware RobotC, který umožňuje kostce spouštět programy napsané v jazyce C. RobotC je používán pro programování robotů napříč platformami. Kromě robotů LEGO Mindstorms podporuje i roboty VEX Robotics či platformu Arduino. Dalším náhradním firmwarem je ev3dev, založený na Linuxové distribuci Debian. Ten podporuje kromě EV3 i platformu BrickPi založenou na populárním jednočipovém počítači Raspberry Pi s využitím kostek LEGO. Ev3dev zprostředkovává programování kostky ve více možných jazycích. Jazyky, které je možné použít při zvolení firmware ev3dev, jsou Python, JavaScript, C, C++ a další. V neposlední řadě je možné využít k programování firmware leJOS, který umožňuje programování robota v programovacím jazyce Java. [10] [11]

4 LEJOS

4.1 Představení

LeJOS je náhradní firmware pro programovatelné kostky LEGO Mindstorms. Existují verze pro všechny vydání LEGO Mindstorms. Verze pro EV3 vznikla v roce 2013. LeJOS obsahuje Java Virtual Machine (zkráceně JVM), který umožňuje spouštět programy napsané v programovacím jazyce Java.

LeJOS je odvozen z projektu TinyVM a byl původně vyvíjen jako open source Josém Solórzanen od roku 1999. Jméno leJOS je odvozeno z anglického „legos“, zkratky Java Operating Systém a španělského slova „lejos“, odkud si bere i výslovnost. Mezi vlastnosti leJOSu patří: objektově orientovaný jazyk Java, preemptivní vlákna, vícerozměrná pole, rekurze, synchronizace, výjimky, primitivní datové typy a zdokumentovaná API. [12]

Součástí balíku leJOS jsou i pomocné nástroje umožňující ovládání zařízení. Jednou z těchto aplikací je EV3 Control Center, pomocí které lze mimo jiné číst data ze senzorů, spouštět motory, spouštět programy či navigovat se menu z počítače připojeného ke kostce. Dalším důležitým nástrojem je EV3 Console, která slouží k přesměrování programového výstupu z LCD displeje kostky do příkazové řádky pro snadnější ladění chyb. V neposlední řadě je nutné zmínit nástroj EV3 SD Card Creator, sloužící k vytvoření bootovací Micro SD karty pro EV3 kostku.

4.2 Instalace na Windows 10

V následující části bude popsána instalace balíku leJOS na operačním systému Windows 10, jakožto nejpoužívanějšího operačního systému stolních počítačů. Kromě zmiňovaných Windows 10 lze balík nainstalovat i na MacOS či Linux. Kompletní průvodce instalací na všech podporovaných platformách je dostupný v angličtině na oficiálních webových stránkách leJOSu. Firmware se instaluje na Micro SD kartu, proto je k instalaci nutná čtečka těchto karet. Tím, že se firmware instaluje na externí kartu se docílí toho, že se nepřepíše původní firmware, ale pouze toho, že kostka bude bootovat upravenou verzi firmware z karty, čímž by se dal tento přístup nazvat dual-bootem. [13]

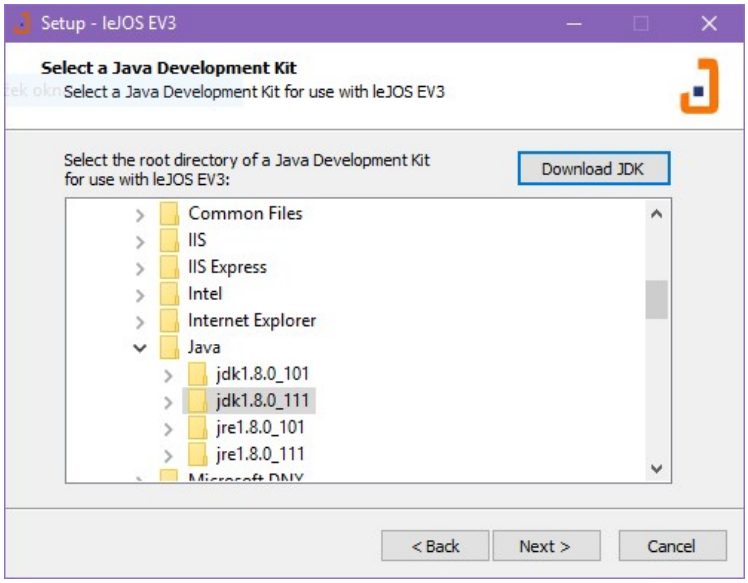
4.2.1 Stažení instalátoru leJOS

Instalátor nejnovější verze, kterou aktuálně je leJOS_EV3_0.9.1-beta_win32_setup.exe, je možné stáhnout ze složky leJOS na portálu sourceforge.net. Tímto instalátorem se nainstalují kromě instalačních souborů pro firmware kostky, i nástroje pro správu kostky.

Obrázek 4 – Uvítací okno instalace

4.2.2 Volba JDK

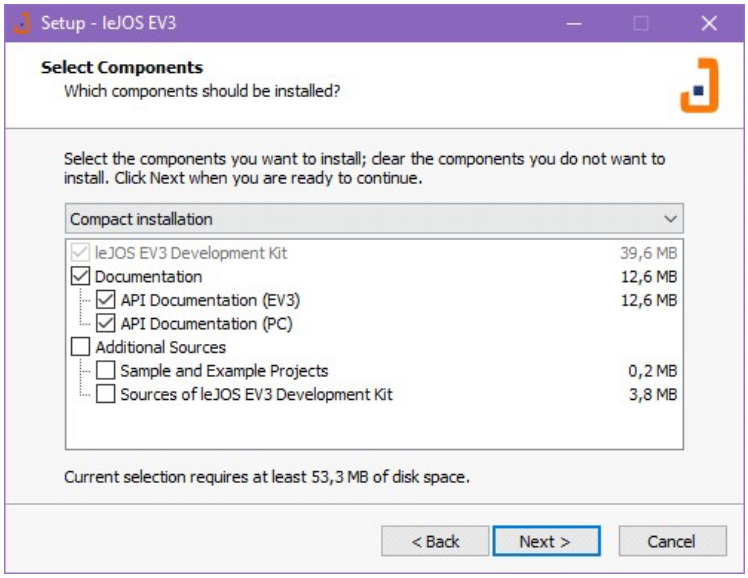
V dalším kroku je nutné vybrat Java Development Kit (zkráceně JDK). V případě, že žádné JDK nainstalované nemáme, pomůže nám tlačítko, které nám v prohlížeči zobrazí webové stránky Oracle, odkud je možné nejnovější JDK stáhnout. Adresa zmíněných stránek je http://www.oracle.com/technetwork/java/javase/downloads/



Obrázek 5 – Okno volby JDK

4.2.3 Volba dodatečných komponentů

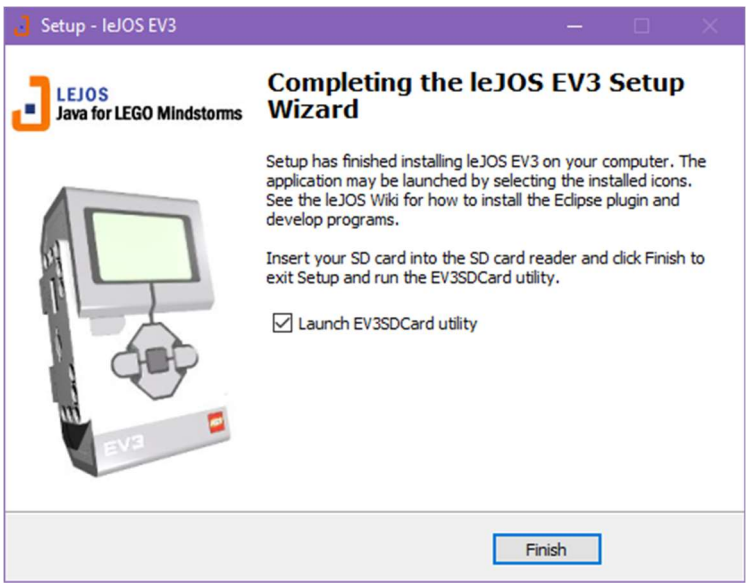
Zde je možné si vybrat, jestli chceme nainstalovat volitelné komponenty, kterými jsou dokumentace, zdrojové kódy vzorových programů a zdrojové kódy leJOS EV3 Development Kit.



Obrázek 6 – Okno volby umístění

4.2.4 Dokončení instalace

Tímto krokem je instalace leJOSu na Windows hotová. Po dokončení se nabídne nástroj EV3 SD Card Creator, kterým se vytváří bootovatelná Micro SD karta.



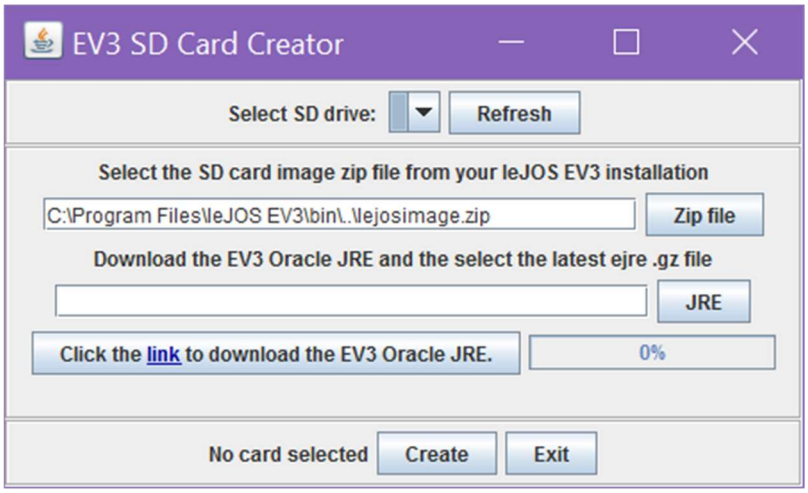
Obrázek 7 – Okno dokončení instalace

4.3 Instalace na EV3 kostku

Pro instalaci firmware na kostku je nutné mít Micro SD kartu o kapacitě alespoň 2 GB s formátem FAT32. Zapsáním se na kartu nahraje modifikovaná verze standardního LEGO firmware.

4.3.1 Vytvoření bootovatelné Micro SD karty.

V této fázi je na disku rozbalená utilita EV3 SD Card Creator na vytvoření bootovatelné karty a zdrojové soubory pro instalaci. Ve výchozím stavu po instalaci se předvolí cesta k souboru s obrazem leJOSu. Dále je nutné vybrat cestu k souboru s JRE (Java Runtime Environment). V případě, že se žádná verzi na disku nenachází, je možné ji stáhnout pomocí tlačítka aplikace, které vede na webovou stránku http://www.java.com/legomindstorms. Z důvodů kompatibility je doporučené stáhnout ověřenou stabilní verzi *7u60*.



Obrázek 8 – Nástroj tvorby bootovatelné Micro SD karty

V případě, že se takto vytvořený systém na kostce zasekne při bootování, je nutné vytvořit bootovatelnou kartu manuálně. Postup manuálního vytvoření bootovatelné karty je následující. Nejprve je nutné si nainstalovat aplikaci Win32 Disk Imager (dostupné z: https://sourceforge.net/projects/win32diskimager/). V dalším kroku je nutné zformátovat kartu a pomocí stažené aplikace zapsat na kartu soubor *sd500.img*, který se nachází ve složce, kam jsme nainstalovali leJOS (standardně C:\Program Files\leJOS EV3). Dále je nutné rozbalit do kořenového adresáře karty obsah komprimovaného souboru *lejosimage.zip*, nacházející se ve stejné složce jako předchozí soubor. Posledním krokem je vložení archivního souboru JRE do karty.

V této chvíli už zbývá jen kartu zasunout do slotu na kostce a spustit jí. Kostka spustí instalaci leJOSu, která trvá přibližně 10 minut, během které se dále karta zformátuje a zmenší se její

kapacita. Pro naformátování karty do původní velikosti pro jiné použití je nutné použít externí aplikaci nebo nástroj *diskpart*. Po dokončení instalace se na kostce zobrazí menu leJOSu.

4.4 Instalace vývojového prostředí

Oficiálně doporučeným vývojovým prostředím je Eclipse (dostupné z: http://www.eclipse.org/downloads/). Po nainstalování Eclipse je nutné nainstalovat leJOS EV3 Eclipse plugin, který najdeme pomocí vestavěné možnosti „Help → Install New Software“. Aby manažer pluginů námi požadovaný plugin našel, je nutné přidat repozitář http://lejos.sourceforge.net/tools/eclipse/plugin/ev3. Přidáním pluginu se nám do nastavení přidá položka leJOS EV3, kde je důležité nastavit do „Name“ IP adresu USB rozhraní kostky (ve výchozím stavu 10.0.1.1). Tímto se umožní Eclipse nahrávat a spouštět kompilované programy.

4.4.1 Spojení EV3 a počítače prostřednictvím USB.

Při prvním připojení kostky s nainstalovaným leJOSem si systém Windows automaticky nainstaluje ovladače. LeJOS s USB pracuje jako s Ethernetovým připojením prostřednictvím USB. Je tak možné, že Windows nainstaluje nesprávné ovladače a tím znemožní tento mód připojení. V tomto případě je nutné ovladače RNDIS (zkratka Remote Network Driver Interface Specification) nainstalovat manuálně. Že se nainstalovaly správné ovladače, lze poznat tak, že při pravém stisku myši na zařízení v „Zařízení a tiskárny“ se zobrazí volba „Nastavení sítě“. V nastavení sítě lze tak vidět nové Ethernetové připojení. Nyní je vhodné nové připojení pojmenovat a u standardního připojení k internetu nastavit sdílení připojení pro EV3. Komunikaci počítače s kostkou můžeme otestovat pomocí příkazu *ping 10.0.1.1*. [14]

4.5 Vývoj programu robota

Po splnění předchozích kroků instalace by mělo být správně nakonfigurované vývojové prostředí Eclipse a s EV3 navázaná komunikace prostřednictvím USB. Následně už zbývá jen naprogramovat kostce chování.

LeJOS obsahuje spoustu vestavěných tříd pro ovládání motorů, sbírání dat ze senzorů či vykreslování na displej. Všechny třídy lze dohledat v obsáhlé leJOS API. Syntax programu je standardní syntax programovacího jazyka Java. Zde uvedu příklad jednoduchého programu, který slouží jako demonstrace pracování s vestavěnými třídami leJOSu.

```
package Priklad;

import lejos.hardware.Sound;
import lejos.hardware.lcd.LCD;
import lejos.hardware.motor.Motor;
import lejos.utility.Delay;

public class Priklad {
	public static void main(String[] args) {
		// nastaví hlasitost na 20
		Sound.setVolume(20);

		// vydá krátké pípnutí
		Sound.beep();

		// nastaví rychlost v deg/s motoru zapojeném v portu A
		Motor.A.setSpeed(50);

		// zapne motor
		Motor.A.forward();

		// výpis na LCD
		System.out.println("Hello world");

		// alternativní výpis pomocí statické třídy LCD
		LCD.drawString("Alt hello world", 0, 4);

		// prodleva 7 vteřin
		Delay.msDelay(7000);

		// zastaví motor
		Motor.A.stop();

		// vypne program a vrátí se do hlavního menu
		System.exit(0);
	}
}
```

Kód 1 – Příklad programu pro leJOS

Prostřednictvím třídy *Sound* přistupujeme k jejím statickým metodám *setVolume(int vol)* a *beep()*. První z nich nastaví hlasitost v rámci programu, čímž nepřepíše hlasitost v nastavení kostky. Druhá vydá krátké pípnutí pomocí vestavěného reproduktoru kostky. Další řádky se týkají motorů.

Třída *Motor* obsahuje čtyři statické instance třídy *NXTRegulatedMotor* pojmenované podle písmen portů. Metodami *setSpeed(int speed)*, *forward()* a *stop()* ovládáme rychlost a pohyb motoru daného portu.

Výpis na obrazovku je možné řešit implicitně, pomocí klasického Javového výpisu do výstupní konzole využitím *System.out.println()*. Alternativně lze využít třídu LCD a její statickou metodu *drawString(String str, int x, int y)*, která vypíše zadaný řetězec na místo určené souřadnicemi. Tímto způsobem lze dosáhnout lepší kontroly nad výstupem na obrazovku.

5 ANDROID

5.1 Představení

Pro splnění zadání je nutné ještě představit platformu Android. V současnosti je Android nejpopulárnější mobilní platformou zahrnující jak chytré telefony, tak tablety. Kromě těchto platforem se Android dostal už i do chytrých hodinek, televizí a dokonce automobilů. Jeho jádro je založené na Linuxu a je distribuovaný pod open-source licencí.

Vznik operačního systému Android se spojuje s vznikem stejnojmenné firmy v roce 2003. Zakladateli této firmy byli Andy Rubin, Rich Miner, Nick Sears a Chris White. O dva roky později v roce 2005 byla firma koupena firmou Google. V roce 2008 byl vydán první mobilní telefon s verzí Android 1.0 HTC Dream. S verzí 1.5 zvanou Cupcake, vydanou v roce 2009, začala tradice pojmenování verzí podle sladkostí začínajících písmenem v pořadí dle abecedy. Současnou nejnovější verzí je Android 7.0 Nougat. [15]

Android dominuje trhu chytrých telefonů s tržním podílem 86,8 % ke třetímu kvartálu 2016 podle International Data Corporation. Jeho nejbližší soupeř je iOS od firmy Apple s 12,5 %. Důvody dominance Androidu je mnoho. Jedním z hlavních důvodů je dostupnost. Zařízení s operačním systémem Android vyrábí drtivá většina výrobců chytrých telefonů všech cenových kategorií. Dalším důvodem je bohatý ekosystém aplikací s aplikacemi placenými i aplikacemi zdarma, které ovšem mohou obsahovat reklamy. [16]

5.2 Vývoj aplikace

Nativní aplikace pro Android jsou převážně založené na jazyku Java. Kromě jazyku Java je možné použít od verze 2.2 i jazyk C++, který například umožňuje vyšší výkon v určitých druzích aplikací či zjednodušuje přenášení (portování) existujících aplikací v C++ na platformu Android. Pro prezentační část aplikace (anglicky front end) se využívá značkovacího jazyka XML. Druhou možností jsou aplikace webové, čímž se rozumí multiplatformní aplikace založené na jazycích HTML5, CSS a Javascript. Hlavní výhoda tohoto přístupu vyvíjení mobilních aplikací je v tom, že není nutné pro každý mobilní operační systém vytvářet vlastní verzi, protože využívají vestavěný webový prohlížeč daného operačního systému. Nevýhodou je neschopnost využít některé hardwarové prostředky konkrétní mobilní platformy a výkon při běhu aplikace.

Pro vývoj aplikace jsem zvolil nativní aplikaci ve vývojovém prostředí Android Studio. Android studio je vyvíjeno společností Google od roku 2013 a je založené na IntelliJ IDEA od

firmy JetBrains. Díky tržnímu podílu Androidu je možností více. Mezi další populární vývojová prostředí patří Eclipse či IntelliJ IDEA. Důvodů proč zvolit Android Studio je mnoho. Díky specializaci na vývoj aplikací pro Android nabízí vyšší míru integrace ihned po nainstalování, bez nutnosti stahovat rozšíření navíc. To se projevuje v relevantnějším našeptávání kódu specifického pro Android. Další výhodou je přehlednější zobrazení struktury Android projektu. Android Studio využívá k sestavování aplikace (anglicky build) zabudovaný automatizační systém sestavování Gradle, pomocí kterého lze deklarovat závislost na externí knihovně. Gradle se postará o její zahrnutí do výsledného souboru s koncovkou *.apk* po sestavení aplikace. Pro vyvíjení a ladění aplikací lze využít intuitivní vestavěný emulátor.

5.3 Obecná struktura projektu v Android studio

Projekt aplikace pro Android se od obyčejného Java projektu liší v několika částech. Android Studio obsahuje možnost zobrazení optimalizované pro Android aplikace, které zjednodušuje a zpřehledňuje strukturu souborů projektu. V následující části bude tato zjednodušená struktura projektu popsána.

Modul obaluje zdrojový kód aplikace, zdrojové soubory a nastavení aplikace (např. soubor Android manifest). Pomocí modulů je možné v jednom projektu vytvořit verzi aplikace na více zařízení (např. pro Android TV nebo Android Wear platformy). Každý modul je možno samostatně sestavovat (build), testovat a ladit. Tímto se podobají projektu v terminologii Eclipse. Pro vytvoření nového modulu je nutné prvně zvolit typ modulu z několika předvoleb. Tyto předvolby obsahují kromě aplikací na mobilní zařízení Android i moduly pro Google Cloud a knihovny. Pro vytvoření modulu mobilního zařízení je nutné ještě zvolit nejnižší podporovanou verzi SDK (Software Development Kit) pro daný modul. Modulům jsou podřazené skupiny *java*, *res* a *manifests*.

Skupina *java* obsahuje balíčky a jim podřazené třídy. Což znamená, že se zde nachází veškerý programový kód aplikace. Zároveň se zde nachází jednotkové testy frameworku JUnit. Speciálními třídami specifickými pro Android jsou aktivity. Aktivity jsou třídy, které interagují s uživatelem. Každá obrazovka aplikace je jimi tvořena. Pro přepínání mezi aktivitami se používá objekt zvaný Intent, který s sebou dokáže nést data, které cílová aktivita může zpracovat.

Skupina *res* obsahuje veškeré neprogramové zdroje, které aplikace používá. Příkladem zdrojů mohou být XML soubory schématu rozložení, texty pro uživatelské rozhraní a lokalizaci nebo bitmapové obrázky. Tento adresář je dále dělen na složky:

- Drawable – obsahuje vykreslitelné XML soubory a bitmapové obrázky. Každý obrázek si vytvoří složku, která může obsahovat více verzí pro různé úrovně rozlišení displeje cílového zařízení.
- Layout – obsahuje XML soubory definující rozložení komponent.
- Mipmap – obsahuje bitmapové ikony pro spouštěč aplikace. Stejně jako u drawable obrázky mohou existovat ve více velikostech.
- Values – obsahuje XML soubory jednoduchými definovanými konstantami pro barvy, texty.

Poslední skupinou je skupina *manifests*, která obsahuje důležitý soubor *AndroidManifest.xml*, který poskytuje zařízení důležité informace o dané aplikaci, nutné pro správný běh aplikace. Příkladem takových informací je například název aplikace, její ikona, nejnižší podporovaná verze API či nutná oprávnění přístupu k hardwarovým prostředkům. [17] [18]

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.upce.micek.EV3Client">

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.upce.micek.EV3Client.MenuActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.upce.micek.EV3Client.ControlActivity"
            android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
        </activity>
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
    </application>

</manifest>
```

Kód 2 – Příklad obsah souboru AndroidManifest.xml

6 BLUETOOTH

Důležitou součástí praktické části je přenos dat pomocí technologie Bluetooth. Bluetooth bylo vytvořeno v roce 1994 firmou Ericsson a jeho cílem bylo nahradit dosavadní drátové sériové rozhraní RS-232. Od svého vývoje prošlo Bluetooth několika iteracemi, než dosáhlo současné verze s číslem 5. Standard, který Bluetooth definuje, má označení IEEE 802.15.1, ale jeho správu má na starosti organizace Bluetooth Special Interest Group (zkratka SIG).

Komunikace je zprostředkována radiovými vlnami a funguje na krátkou vzdálenost, čímž spadá do kategorie osobních počítačových sítí PAN (Personal Area Network). Bluetooth pracuje ve frekvenčním pásmu ISM 2,4 GHz (což je stejné pásmo jako Wi-Fi) a využívá FHSS, díky které je během vteřiny provedeno až několik stovek přeskoků mezi 79 frekvenčními pásmy o šířce pásma 1 MHz (40 pásem o šířce 2 MHz pro Bluetooth LE), čímž zamezuje rušení či odposlechu. Pro vzájemnou komunikaci dvou zařízení je nutné prvně, aby se zařízení spárovala, čímž si navzájem potvrdí identitu a uloží si záznam do paměti pro příští připojení.

Od verze 2.0 existuje specifikace Bluetooth EDR (Enhanced Data Rate) umožňující dvojnásobnou přenosovou rychlost oproti předchozí Bluetooth BR (Basic Rate) specifikaci, čímž efektivně snižuje spotřebu energie. S verzí 4.0 byla představena specifikace Bluetooth LE (Low Energy), dříve také jako Bluetooth Smart, která byla zaměřena na IoT (Internet of Things) zařízení a chytré hodinky. Zařízení s Bluetooth LE tráví většinu času v režimu spánku a budí se až po navázání kontaktu, což společně s kratší dobou komunikace má za výsledek nižší spotřebu energie. [19]

7 PRAKTICKÁ ČÁST

7.1 Zadání

Cílem praktické části bylo sestrojit robota s využitím senzorů ovládaného mobilním telefonem. Jako senzory jsem zvolil EV3 ultrazvukový. Tento senzor zamezí kolizím omezením rychlosti, v případě, že se robot bude blížit překážce. Dalším zvoleným senzorem je HiTechnic úhlový, který pomocí matematické operace bude vypisovat rychlost v centimetrech za sekundu. Posledním zvoleným senzorem je HiTechnic kompas, sloužící k určování orientace robota. Robot je poháněn dvěma velkými EV3 motory a zatáčení je řešeno pomocí středního EV3 motoru a převodu různoběžných os. Ovládání pohybu bude zprostředkováno pomocí touchpadu a jednotlivých tlačítek pro speciální funkce. Data sbíraná senzory budou uživateli prezentována jak v textové formě, tak graficky.

7.2 Bluetooth spojení

Na straně robota, který zaujme pozici serveru, je nutné vyčkat spojení. Pro použití potřebných metod je nutné importovat tyto třídy.

```
import lejos.hardware.Bluetooth;
import lejos.remote.nxt.BTConnection;
import lejos.remote.nxt.NXTConnection;
```

Kód 3 – Importované třídy pro umožnění Bluetooth spojení

Samotné připojení vypadá takto. Čas na vypršení čekání na připojení je nastaven na 0, tímto bude kostka čekat donekonečna, než se připojí Bluetooth klient. V připojení je zvolen typ komunikace *RAW* neboli hrubý mód. V tomto módu spojení se přenáší pouze vstupní a výstupní byty bez jakékoliv hlavičky.

```
BTConnection btc = (BTConnection) Bluetooth.getNXTCommConnector().waitForConnection(0,
NXTConnection.RAW);
```

Kód 4 – Vyčkání spojení na straně serveru

Na straně klienta, tedy telefonu, je nutné zvolit, ke kterému zařízení se chceme připojit. Výpis spárovaných zařízení provedeme následovně.

```
ArrayList list = new ArrayList();
pairedDevices = btAdapter.getBondedDevices();

for (BluetoothDevice bt : pairedDevices)
    list.add(bt.getName() + "\n" + bt.getAddress());
```

Kód 5 – Výpis spárovaných zařízení

Dále adresu zvoleného zařízení předáme vláknu spojení Bluetooth, které se nachází v ovládací aktivitě, v třídě *BluetoothConnectionThread*. V konstruktoru této třídy vytvořím socket k

Bluetooth zařízení pomocí známého unikátního identifikátoru pro připojení profilu Bluetooth SPP (Serial Port Profile).

```
    public BluetoothConnectionThread(BluetoothDevice device) {
        // Použití dočasné proměnné, protože btSocket je finální proměnná
        BluetoothSocket tmp = null;
        btDevice = device;

        try {
            // Získání BluetoothSocketu kterým se připojíme k zařízení
            // Jako UUID je zvolený prověřený string.
            // Při připojení k jinému android zařízení je nutné vygenerovat vlastní.
            tmp = device.createRfcommSocketToServiceRecord( UUID.fromString("00001101-0000-
1000-8000-00805F9B34FB"));
        } catch (IOException e) {
            // Vytvoření socketu není možné aplikace se vrátí do přechozí aktivity.
            finish();

        }
        btSocket = tmp;
    }

    public void run() {
        // Zrušení vyhledávání pro zlepšení rychlosti.
        btAdapter.cancelDiscovery();

        try {
            // Připojení k zařízení přes socket. Blokující volání.
            btSocket.connect();
        } catch (IOException connectException) {
            // Spojení se nezdařilo
            System.out.println("Exception: " + connectException.toString());
            try {
                btSocket.close();
            } catch (IOException closeException) {
                finish();
            }
            return;
        }
        // Úspěšně připojeno, práce se socketem se provádí v jiném vlákně.
    }
```

Kód 6 – Třída Bluetooth spojení na straně klienta

Po úspěšném navázání spojení je ze socketu vlákna vytvořen vstupní a výstupní datový proud, pomocí kterých se přenáší informace do a ze zařízení respektivně.

```
btThread = new BluetoothConnectionThread(device); // vytvoření instance vlákna připojení
bluetooth
btThread.run(); // spuštění vlákna

try {
    btOutStream = btThread.btSocket.getOutputStream(); // výstupní proud
} catch (IOException e) {
    e.printStackTrace();
}

try {
    btInStream = btThread.btSocket.getInputStream(); // vstupní proud
} catch (IOException e) {
    e.printStackTrace();
}
```

Kód 7 – Vytvoření vstupního a výstupního datového proudu ze Socketu

7.3 Struktura programu EV3Server

Program EV3Server je program chování robota vyvíjený ve vývojovém prostředí Eclipse s zásuvným modulem leJOS. Jeho struktura je rozdělena do tří tříd: *Sensors.java*, *Motors.java* a hlavní třída *Server.java*. Vedlejší třídy obsahují statické metody, aby mohly spolu komunikovat přímo. Pro synchronizování vláken při ukončení je využit semafor.

7.3.1 Sensors.java

Třída *Sensors.java* obsahuje statické atributy použitých senzorů, které tak jsou přístupné z ostatních tříd a statický atribut *distance*, jenž slouží k ukládání hodnot senzoru vzdálenosti pro regulaci rychlosti motoru v blízkosti překážky. Použitými třídami senzorů jsou *HiTechnicAngleSensor*, *HiTechnicCompass* a *EV3UltrasonicSensor*. Mezi metody této třídy patří *initializeSensors()*, ve které inicializuji každému senzoru požadovaný mód. Pro vzdálenostní senzor byl zvolen výchozí mód čili měření vzdálenosti. Stejně tak pro kompas, tedy měření úhlů v rozsahu 0 až 359 stupňů. Úhlový senzor využívá módu 2, tedy měření rychlosti otáčení. Metoda *getSamples(float[] array)* předává pole datového typu *float* k naplnění metodám *fetchSample(float[] sample, int offset)* jednotlivých senzorů. Poslední metodou této třídy je metoda *closeSensors()*, kterou se ukončí práce se senzory.

7.3.2 Motors.java

Třída *Motors.java* zahrnuje statické atributy motorů a statické atributy uchovávající zvolenou rychlost a směr. Pro hlavní pohonné motory je použitá třída *UnregulatedMotor* a motor ovládající zatáčení je zprostředkován třídou *EV3MediumRegulatedMotor*, která umožňuje přesnější nastavení otočení. Metody této třídy ovládají pohyb všech motorů robota. Její první metodou je třída *initializeMotors()*, jejíž nejdůležitější funkcí je nastavit střed zatáčení a rychlost zatáčejícího motoru. Pohybová metoda *goForward()* nastaví motorům rychlost podle atributu třídy *forwardSpeed* v případě, že hodnota *distance* ve třídě *Sensors.java* je větší než hodnota konstanty *minimalDistance* též třídy, tedy 25. V opačném případě nastaví proporciálně rychlost podle výpočtu na ukázce Kód 8. Dále metoda nastaví atribut *direction* na 1 a zapne oba motory.

```
speed = (int)(forwardSpeed * (Sensors.distance - offset) / (float)Sensors.minimalDistance);
```

Kód 8 – Výpočet rychlost vpřed v blízkosti překážky

Metoda *goBackward()* je jednoduší v tom, že jako rychlost se vždy nastavuje atribut *backwardSpeed*. Po nastavení rychlosti následuje nastavení *direction* na -1 a zapnutí obou motorů. Mezi další metody této třídy patří metoda *idle()*, sloužící k doběhnutí motorů, aniž by

robot zastavil na místě. Volání této metody nastaví *direction* na 0. Metoda *turnTo(int value)* s použitím neblokujícího zatáčení zatočí na úhel *value*, který v záporných hodnotách představuje zatáčení doleva a v kladných doprava. Další metoda *straighten()* se od předchozí liší tím, že blokuje další akce než je tato dokončena a slouží ke srovnání kol na nulový úhel. Metodou *calibrate()* je uživatel schopný nastavit nulový bod zatáčení. Poslední metodou této třídy je metoda *closeMotors()*, která ukončí práci s motory a uvolní použité prostředky.

7.3.3 Server.java

Hlavní třída *Server.java* obsahuje hlavní metodu, v které zahájím čekání na spojení Bluetooth. Po úspěšném spojení vytvořím vstupní a výstupní datový proud, se kterými pak pracuji v příjímacím a odesílacím vláknu, které následně vytvořím. Po vytvoření oznámím zvukově uživateli připravenost programu.

Přijímací vlákno serveru *recievingThread* slouží k ovládání robota na základě zprávy získané z mobilního telefonu. Nejprve se deklaruje proměnná typu bytového pole *message* o dvou prvcích a inicializují se motory. Následuje cyklus čtení vstupního proudu, uvnitř kterého se zapíší hodnoty do bytového pole *message* a podle hodnot jeho prvků rozhodne o další akci. První prvek *message* rozhoduje, zdali se jedná o obyčejný pohyb nebo jednu ze tří speciálních funkcí. V následující tabulce je znázorněno rozdělení hodnot prvního prvku *message* a jejich význam. Nevyužité hodnoty je možné v budoucnu využít pro rozšíření aplikace o dodatečné speciální funkce.

Tabulka 1 – Tabulka možných hodnot prvku zprávy a jejich význam

Hodnota message[0]	Činnost
103	Srovnání zatáčení
102	Kalibrace zatáčení
101	Ukončení programu
<1;100>	Pohyb dopředu danou rychlostí
0	Volnoběh
<-100;-1>	Pohyb dozadu danou rychlostí

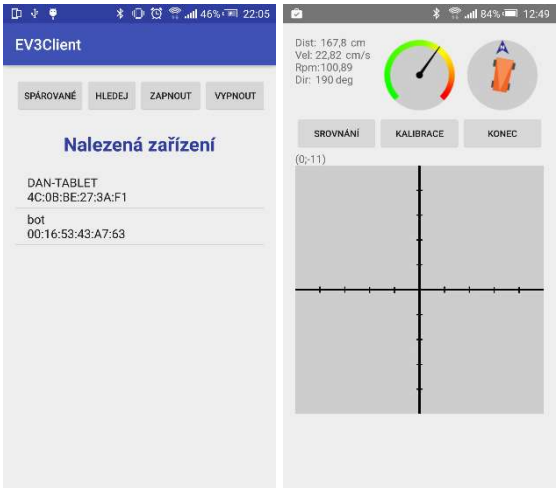
V případě, že hodnota prvního prvku je v rozmezí od -100 do 100, nastaví se otočení kol na úhel rovný hodnotě druhého prvku. Pokud zpráva obsahuje kód ukončení 101, zavolá se metoda *exit()*, při níž se srovná řízení, uzavřou se všechny senzory i motory a program se ukončí.

Vlákno *sendingThread* neboli odesílací vlákno slouží ke čtení hodnot senzorů a jejich zapisování do výstupního datového proudu. Po vytvoření vlákna se inicializují senzory a

deklaruje se proměnná *sensorValues* typu pole čísel float o třech prvcích, sloužící k ukládání naměřených dat ze senzorů. Dále jsou v cyklu zapsány hodnoty ze senzorů do pole *sensorValues* a hodnota vzdálenosti se zapíše do statické proměnné *distance* ve třídě senzorů pro omezení maximální rychlosti v blízkosti objektů. Před ukončením těla cyklu jsou hodnoty zapsány do výstupního proudu.

7.4 Struktura programu EV3Client

EV3Client je nativní mobilní aplikace pro platformu Android, která umožňuje se připojit k EV3 kostce, na které běží program EV3Server pomocí bezdrátového spojení Bluetooth. Aplikace je vytvořena ve vývojovém prostředí Android Studio 2.3.1. Po připojení aplikace zprostředkovává uživateli ovládání kostky a zobrazuje naměřená data jejích senzorů. Aplikace obsahuje dvě aktivity *MenuActivity.java* a *ControlActivity.java*.



Obrázek 9 – Snímek hlavní obrazovky (vlevo) a ovládací obrazovky (vpravo)

7.4.1 MenuActivity.java

Aktivita *MenuActivity* je výchozí aktivita, která se zobrazí při spuštění aplikace. Její obsah tvoří tlačítka pro ovládání Bluetooth adaptéru a prostor pro výpis dostupných zařízení. Dvě z nich jsou dedikovaná tlačítka pro vypnutí a zapnutí Bluetooth telefonu přímo z aplikace. Zbylá dvě tlačítka slouží k výpisu seznamu zařízení. Tlačítko „Spárované“ zobrazuje všechny zařízení, které má mobilní telefon mezi spárovanými. Tlačítko „Hledej“ vypíše všechna dostupná zařízení v dosahu Bluetooth. Při kliknutí na vypsané zařízení je jeho jméno a adresa předána ovládací aktivitě *ControlActivity.java*, ve které se provede pokus o spojení.

7.4.2 ControlActivity.java

Ovládací aktivita *ControlActivity.java* slouží jako rozhraní mezi robotem a uživatelem. Jejím úkolem je navázání spojení mezi robotem a uživatelem a následné ovládání robota.

Ve vrchní části rozložení aktivity se nachází informační část. Ta obsahuje textový výpis hodnot senzorů vzdálenosti, úhlové rychlosti, lineární rychlosti a kompasu. Lineární rychlost je uváděna v centimetrech za sekundu a hodnota je pomocí vzorce odvozována z úhlové rychlosti v otáčkách za minutu. Vedle textového výpisu se nachází animované znázornění hodnot rychlosti a směru kompasu. Tyto hodnoty jsou obnovovány z přijímacího vlákna *RecievingThread*, které s využitím datového vstupního proudu očekává zprávu obsahující tři čísla typu *float*.

Pro aktualizování grafického uživatelského rozhraní (GUI) z tohoto vlákna je nutné přistoupit ke komponentám pomocí metody *runOnUiThread()*, kterou nabízí třída *Activity*. Tato metoda vloží do fronty vlákna uživatelského rozhraní objekt typu *Runnable* s naším kódem v případě, že vlákno, ve kterém se metoda nachází, není vlákno GUI. V opačném případě se daný objekt provede ihned. V tomto konkrétním případě metoda *runOnUiThread()* zavolá metodu *updateUI(float dist, float vel, float dir)*, která nastaví textovému výpisu požadované hodnoty a nastaví indikátorům tachometru a kompasu úhel otočení. Výstup úhlového senzoru je v otáčkách za sekundu, proto je pro získání požadované hodnoty nutné vynásobit původní hodnotu pomocí konstant ze vzorců.

```
final int distanceConversion = 100;
final int speedometerGaugeOffset = 45;
final int speedometerGaugeMultiplier = 45;

// převod úhlové rychlosti v otáčkách za minutu na lineární rychlost v cm/s
// v = r × RPM × 0.10472
// (0.0432 / 2) * 0.10472 * 100
final double velocityConversion = 0.2261952;

// převod z otáček za vteřinu na otáčky za minutu, opačný směr a převod ozubených kol
// 60 * (16 / 36) * (-1);
final double rpmConversion = -26.6666666667;

public void updateUI(float dist, float vel, float dir){
    ((TextView) findViewById(R.id.botInfo)).setText(
            "Dist:\t" + new DecimalFormat("#.##").format(
                (double) (dist * distanceConversion)) +"\t"+"cm\n"+
            "Vel:\t" + new DecimalFormat("#.##").format(
                 (double) (vel * velocityConversion * rpmConversion)) +"\tcm/s\n"+
            "Rpm:\t" + new DecimalFormat("#.##").format(
                 (double) (vel * rpmConversion)) +"\t\n"+
            "Dir:\t" + new DecimalFormat("#.##").format((double) (dir)) + "\tdeg"
    );
    ((ImageView) findViewById(R.id.speedometerNeedle)).setRotation(
                (Math.abs(vel) * speedometerGaugeMultiplier) + speedometerGaugeOffset);
    ((ImageView) findViewById(R.id.compassIndicator)).setRotation(dir);
}
```

Kód 9 – Aktualizace výpisu na straně klienta

Pod informační částí se nachází část ovládací. Ta se skládá ze tří tlačítek pro speciální funkce, kterými jsou srovnání kol, kalibrace a ukončení programu. Tyto tlačítka reagují na kliknutí

pomocí implementace *View.OnClickListener* a jeho metody *onClick()*, díky které odešlou kostce odpovídající kód podle tabulky 1. Při stisknutí tlačítka konec se ještě navíc po odeslání kódu aktivita ukončí a uživatel se přesune zpět do aktivity menu. Zbytek aktivity vyplňuje ovládací dotyková oblast, která implementuje *View.OnTouchListener* a jeho metodu *onTouch()*, která snímá souřadnice doteku uživatele a odesílá je robotovi jako oddělené hodnoty bytového pole *message*. Přesné snímané souřadnice odeslané v message jsou vypisovány nad levým horním rohem dotykové oblasti. Metoda *onTouch()* také rozeznává událost zvednutí prstu z dotykové obrazovky pomocí *MotionEvent.ACTION_UP* a při jejím nastání odešle jako hodnotu pohybu 0, čímž dá kostce pokyn nastavit volnoběh. Všechny ovládací prvky využívají ovládací metody *sendMessage()*, sloužící k odeslání současného stavu pole *message* skrze výstupní proud kostce.

8 ZÁVĚR

Cílem bakalářské práce bylo využití LEGO Mindstorms pro vytvoření modelu auta ovládaného mobilním telefonem s operačním systémem Android za využití bezdrátové technologie Bluetooth. Práce byla vypracována podle zadání a konzultací s vedoucím práce.

V teoretické části byla představena použitá platforma vývoje robotů LEGO Mindstorms, její třetí vydání s názvem EV3 a její periferie. V další kapitole byly popsány programovací možnosti EV3 kostky. V samostatné kapitole byl popsán firmware leJOS, s podkapitolami věnujícími se představení, instalaci a vývoji programu. Další kapitolou byl představen mobilní operační systém Android, a vývoj nativní aplikace pomocí vývojového prostředí Android Studio. V poslední kapitole je popsána bezdrátová technologie Bluetooth.

V praktické části byl popsán způsob navázání komunikace mezi programem pro firmware leJOS a aplikací pro operační systém Android prostřednictvím technologie Bluetooth. V další kapitole byl detailně představen program kostky EV3Server, jeho třídy a metody. Na závěr byla popsána Android aplikace EV3Client a její aktivity.

Přestože program EV3Server je hotový, je možné robota v budoucnu rozšířit o jeden senzor, jelikož jeden senzorový port zůstal nevyužitý. Případně ho lze rozšířit i o více senzorů za použití senzorového multiplexoru. To samé platí i pro aplikaci EV3Client, u které je možné přidat nové speciální příkazy pomocí neobsazených hodnot Bluetooth zprávy.

Mezi další možné zlepšení, které by uživatelům poskytlo intuitivní ovládání robota, patří využití kombinace akcelerometru a gyroskopického senzoru telefonu pro ovládání robota pomocí naklánění telefonu. Toto řešení je v současnosti hojně využíváno s populárními dálkově ovládanými kvadrokoptérami.

9 ZDROJE

[1] CAPRANI, Ole. RCX Manual. In: *LEGO Lab, University of Aarhus* [online]. [cit. 2017-05-09]. Dostupné z: http://legolab.daimi.au.dk/CSaEA/RCX/Manual.dir/RCXManual.html

[2] THE LEGO GROUP. *LEGO MINDSTORMS NXT: Hardware Developer Kit*. 2006.

[3] THE LEGO GROUP. *LEGO MINDSTORMS EV3: Hardware Developer Kit*. 2013.

[4] SHERRAD, Ann a Amy RHODES. Comparison of the LEGO Mindstorms NXT and EV3 Robotics Education Platforms. *Journal of Extension* [online]. 2014, **52**(5) [cit. 2017-05-09]. ISSN 1077-5315. Dostupné z: https://www.joe.org/joe/2014october/tt9.php

[5] How to build MindCub3r for LEGO MINDSTORMS EV3. In: *MindCuber for EV3 and NXT* [online]. [cit. 2017-05-09]. Dostupné z: http://mindcuber.com/mindcub3r/mindcub3r.html

[6] NXT Angle Sensor. In: *HiTechnic Products* [online]. [cit. 2017-05-09]. Dostupné z: https://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NAA1030

[7] NXT Compass Sensor. In: *HiTechnic Products* [online]. [cit. 2017-05-09]. Dostupné z: https://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NMC1034

[8] THE LEGO GROUP. *Uživatelská příručka.* 2013. s. 69.

[9] NI LabVIEW Module for LEGO® MINDSTORMS®. In: *National Instruments* [online]. [cit. 2017-05-09]. Dostupné z: http://sine.ni.com/nips/cds/view/p/lang/cs/nid/212785

[10] ROBOTC Downloads. In: *ROBOTC.net* [online]. [cit. 2017-05-09]. Dostupné z: http://www.robotc.net/download/

[11] *Ev3dev* [online]. [cit. 2017-05-09]. Dostupné z: http://www.ev3dev.org/

[12] LeJOS EV3. In: *LeJOS: Java for Lego Mindstorms* [online]. [cit. 2017-05-09]. Dostupné z: http://www.lejos.org/ev3.php

[13] Windows installation. In: *SourceForge* [online]. [cit. 2017-05-09]. Dostupné z: https://sourceforge.net/p/lejos/wiki/Windows%20Installation/

[14] Connect via USB. In: *Lego-Robotics and Turtle-Graphics with Java* [online]. [cit. 2017-05-09]. Dostupné z: http://clab2.phbern.ch/lego/legoEnglish/index.php?inhalt_links=home/nav_home.inc.php&inhalt_mitte=ev3install/usb.inc.php

[15] Google's Android OS: Past, Present, and Future. In: *Phone Arena* [online]. 2011 [cit. 2017-05-09]. Dostupné z: http://www.phonearena.com/news/Googles-Android-OS-Past-Present-and-Future_id21273

[16] Smartphone OS Market Share, 2016 Q3. In: *IDC* [online]. [cit. 2017-05-09]. Dostupné z: http://www.idc.com/promo/smartphone-market-share/os

[17] App Manifest. In: *Android Developers* [online]. [cit. 2017-05-09]. Dostupné z: https://developer.android.com/guide/topics/manifest/manifest-intro.html

[18] HLAVÍK, Jiří. 2. díl - Android programování - Vývojové prostředí. In: *ITnetwork.cz* [online]. [cit. 2017-05-09]. Dostupné z: https://www.itnetwork.cz/java/android/tutorial-programovani-pro-android-v-jave-vyvojove-prostredi

[19] What is Bluetooth?: How It Works. In: *Bluetooth Technology Website* [online]. [cit. 2017-05-09]. Dostupné z: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works