

UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Dálkové ovládání modelu robota

David Kudláček

Bakalářská práce

2012

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **David Kudláček**
Osobní číslo: **I07696**
Studijní program: **B2646 Informační technologie**
Studijní obor: **Informační technologie**
Název tématu: **Dálkové ovládání modelu robota**
Zadávající katedra: **Katedra informačních technologií**

Z á s a d y p r o v y p r a c o v á n í :

Cílem práce je naprogramování dálkového ovládání modelu robota s elektronickým diferenciálem.

Teoretická část:

V teoretické části práce rozpracovat metody řízení modelu robota dálkovým ovladačem. Především se požaduje navrhnout strukturu příkazů na dálkové ovládání a algoritmy jejich provedení pro model robota. Požaduje se rozpracování principu "elektronického diferenciálu", který umožní plynulé projíždění zatáček.

Praktická část:

V praktické části práce se požaduje naprogramování dálkového ovládání modelu robota ze stavebnice Lego pomocí mobilního telefonu s operačním systémem Android podle rozboru a návrhu z teoretické části práce tj. implementace struktury příkazů a elektronického diferenciálu. Program pro řídicí jednotku Lego musí být pod operačním systémem LeJOS. Jako jedno z možných rozšíření funkcí se požaduje, aby pro řízení pohybu robota bylo využito inerciálních senzorů v mobilním telefonu, jako jsou akcelerometry a gyroskopy.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Pecinovský, Rudolf, Návrhové vzory, Computer Press, 2007 vydání první, ISBN 978-80-251-1582-4

Page-Jones, Meilir, Základy objektově orientovaného návrhu v UML, Grada 2001, ISBN 80-247-0210-X

Stránky s operačním systémem LeJOS <http://lejos.sourceforge.net/>

Vedoucí bakalářské práce:

Ing. Karel Šimerda

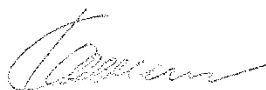
Katedra softwarových technologií

Datum zadání bakalářské práce:

16. prosince 2011

Termín odevzdání bakalářské práce:

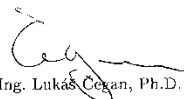
11. května 2012



prof. Ing. Simeon Karamazov, Dr.
děkan



L.S.



Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 30. března 2012

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 16. 08. 2012

David Kudláček

Poděkování

Rád bych poděkoval panu Ing. Karlu Šimerdovi za cenné rady a připomínky, které mi ochotně dával po celou dobu vypracovávání mé bakalářské práce. Dále bych chtěl poděkovat své rodině za podporu během mého studia.

Anotace

Cílem práce je naprogramování dálkového ovládání modelu robota s elektronickým diferenciálem. V teoretické části práce rozpracovat metody řízení modelu robota dálkovým ovladačem. Především se požaduje navrhnout strukturu příkazů na dálkové ovládání a algoritmy jejich provedení pro model robota. Požaduje se rozpracování principu "elektronického diferenciálu", který umožní plynulé projíždění zatáček. V praktické části práce se požaduje naprogramování dálkového ovládání modelu robota ze stavebnice Lego pomocí mobilního telefonu s operačním systémem Android podle rozboru a návrhu z teoretické části práce tj. implementace struktury příkazů a elektronického diferenciálu. Program pro řídicí jednotku Lego musí být pod operačním systémem LeJOS. Jako jedno z možných rozšíření funkcí se požaduje, aby pro řízení pohybu robota bylo využito inerciálních senzorů v mobilním telefonu, jako jsou akcelerometry a gyroskopy.

Klíčová slova

Robot, elektronický diferenciál, Android, NXT, leJOS, mindstorms

Title

Remote control of a robot model

Annotation

The thesis attempts to find a feasible solution for programming remote control of model robot with an electronical differential. The theoretical part of the thesis comprises various methods of driving a model robot using a remote control. Further, it describes the structure of orders for the remote control and algorithms of their performance. The theoretical parts concludes by bringing in the principle of "electronical differential" which enables the robot to go along curves smoothly. The practical part begins with programming the remote controller for the model robot consisting of LEGO parts. The platform for the remote control is based on a mobile phone using Android operating system in accordance with the design of electronical orders and differential suggested in the theoretical part. The program for the LEGO device is operated by LeJOS operating system. As an extra solution for the robot functions the thesis also suggests to use inertial sensors in the mobile phone to control robot's movement, such as accelerometers and gyroscopes.

Keywords

Robot, electronic differential, Android, NXT, leJOS, mindstorms

Obsah

Seznam zkratk	8
Seznam obrázků	9
Seznam tabulek	9
Úvod	10
1 LeJOS	11
1.1 LEGO Mindstorms NXT	11
1.1.1 LeJOS NXJ	11
1.2 Řídící jednotka LEGO NXT Brick.....	12
1.2.1 Instalace	12
1.2.2 Ovládání řídící jednotky	12
1.3 Ovládání motorů	13
1.3.1 Třída NXTRegulatedMotors.....	14
1.3.2 Další třídy pro práci s NXT motory	15
2 Senzory	16
2.1 Analogové senzory	17
2.1.1 Light Detector.....	17
2.1.2 Color Detector	18
2.1.3 Touch Sensor	18
2.1.4 Sound Sensor	18
2.2 Senzory používající I ² C protokol	18
2.2.1 Ultrasonic Sensor.....	18
3 Vzdálená komunikace s NXT	20
3.1 Bluetooth komunikace s PC	20
3.2 Architektura komunikace	20
3.3 Propojení Android smartphone a Lego NXT	21
4 Android	23
4.1 Vysvětlení pojmu Android	23
4.2 Uvedení do problematiky programování Android aplikací.....	23
4.3 Základní komponenty aplikace.....	24
4.3.1 Activities.....	24
4.3.2 Services.....	24

4.3.3	Content providers	24
4.3.4	Broadcast receivers	24
4.4	Manifest File	24
4.5	Aplikační zdroje	24
4.6	Uživatelské rozhraní	25
5	Pohybování modelu robota	27
5.1	Rovný přímý pohyb	27
5.2	Průjezd obloukovou zatáčkou	27
5.3	Ackermannovo řízení	29
5.3.1	Matematický rozbor Ackermannova řízení	30
5.4	Elektronický diferenciál	31
5.4.1	Matematický model elektronického diferenciálu	31
6	Popis aplikační části	35
6.1	Požadavky	35
6.2	Použitý model robota	35
6.3	Použité technologie	36
6.4	Architektura klient-server	36
6.5	Klientská část aplikace	36
6.5.1	Diagram tříd klientské části aplikace	37
6.6	Serverová část aplikace	38
6.6.1	Diagram tříd serverové části aplikace	39
6.7	Komunikace klientské a serverové části	40
	Závěr	42
	Literatura	43
	Příloha A – Obsah přiloženého CD	45

Seznam zkratk

ADT	Android Development Tools
API	Application Programming Interface
IDE	Integrated Development Environment
JDK	Java Development Kit
JVM	Java Virtual Machine
LCP	LEGO Communications Protocol
LED	Light-Emitting Diode
OS	Operační Systém
PC	Personal Computer
RAM	Random Access Memory
SD	Secure Digital
SDK	Software Development Kit
UML	Unified Modeling Language
USB	Universal Serial Bus
WPAN	Wireless Personal Area Network
XML	Extensible Markup Language

Seznam obrázků

Obrázek 1 – Řídící jednotka NXT Brick (zdroj: [5])	12
Obrázek 2 – LEGO NXT motor (zdroj: [6]).....	14
Obrázek 3 – Kompas senzor a ultrasonic senzor (zdroj: [7] [8]).....	16
Obrázek 4 – Architektura komunikace (zdroj: vlastní)	21
Obrázek 5 – Vztah mezi objekty View a ViewGroup (zdroj: [12])	25
Obrázek 6 – Grafický editor pro návrh rozvržení Android aplikace (zdroj: vlastní)	26
Obrázek 7 – Průjezd obloukovou zatáčkou (zdroj: vlastní)	28
Obrázek 8 – Ackermannova geometrie řízení (zdroj: 12)	29
Obrázek 9 – Matematický model Ackermannova řízení (zdroj: vlastní)	30
Obrázek 10 – Pravoúhlý trojúhelník modelu Ackermannova řízení (zdroj: vlastní)	30
Obrázek 11 – Matematický model elektronického diferenciálu (zdroj: vlastní)	32
Obrázek 12 – Rovnoramenný trojúhelník modelu elektronického diferenciálu (zdroj: vlastní)	33
Obrázek 13 – Použitý model robota (zdroj: vlastní).....	35
Obrázek 14 – Rozměry použitého modelu robota (zdroj: vlastní)	36
Obrázek 15 – Uživatelské rozhraní aplikace (zdroj: vlastní)	37
Obrázek 16 – Diagram tříd klientské části aplikace (zdroj: vlastní)	38
Obrázek 17 – Displej řídicí jednotky čekající na nové spojení (zdroj: vlastní)	38
Obrázek 18 – Displej řídicí jednotky po úspěšném navázání spojení (zdroj: vlastní)	39
Obrázek 19 – Displej řídicí jednotky po natočení přední nápravy (zdroj: vlastní)	39
Obrázek 20 – Diagram tříd serverové části aplikace (zdroj: vlastní)	40

Seznam tabulek

Tabulka 1 – Rozbor pohybů modelu robota (zdroj: vlastní)	27
--	----

Úvod

Bakalářskou práci na toto téma jsem si zvolil především proto, že v dnešní době je velmi populární programování mobilních aplikací a zaujala mě možnost zkoumání pohybů robota automobilového typu.

Cílem teoretické části je seznámení se všemi potřebnými souvislostmi, které jsem využil při následné tvorbě části aplikační.

Teoretická část je rozdělena do šesti ucelených kapitol. V té první jsou vysvětleny pojmy potřebné ke snadnému orientování se při používání LEGO NXT Mindstorms robotické stavebnice, je popsán operační systém leJOS, řídicí jednotka NXT a možnosti ovládání motorů. Druhá kapitola se věnuje přípojným senzorům a jejich funkcím. Ve třetí kapitole popisují vzdálenou komunikaci NXT řídicí jednotky na straně serveru a PC nebo mobilního telefonu s operačním systémem Android na straně klienta. Čtvrtá kapitola uvádí do problematiky programování Android aplikací a seznamuje s potřebnými základy pro jejich tvorbu. V páté kapitole jsem se věnoval rozboru pohybů robota, především pak pohybu po oblouku a matematickým závislostem při využití Ackermannovy geometrie řízení a elektronického diferenciálu. Šestá kapitola uzavírá práci popisem aplikační části.

1 LeJOS

1.1 LEGO Mindstorms NXT

LEGO Mindstorms NXT je název pro série sad stavebnic od firmy LEGO, které obsahují programové vybavení (anglicky software) a fyzické komponenty (anglicky hardware) pro tvorbu a ovládání malých, přizpůsobivých a programovatelných robotů. Hlavní částí sady je NXT Brick (dále jen NXT), neboli řídicí jednotka, která je ve své podstatě počítač ovládající připojené modulární senzory, motory a další části stavebnice, typicky shodné se sérií LEGO Technic. K této stavebnici je dodáván standardní firmware od firmy LEGO, který má ovšem omezené možnosti a pokud chce programátor povely robotovi řídit a programovat sám, je možné ho kompletně nahradit jiným, který bude umožňovat programování robota v konkrétním programovacím jazyce. Na výběr máme několik takových systémů, jako např.: NXT-G, jaraco.nxt, Lego.NET, LEGO::NXT, LegoNXTRemote, nebo leJOS NXJ. [1]

1.1.1 LeJOS NXJ

První dvě písmena jsou ve slově leJOS umístěna jako zkratka firmy LEGO, zbytek slova JOS je pak odkazem na operační systém javaOS, vyvinutý společností Sun Microsystems. LeJOS NXJ lze chápat jako softwarový balíček, nebo také programovací prostředí, které umožňuje programovat LEGO Mindstorms NXT roboty pomocí programovacího jazyku Java. Využívá jeho standardní verzi a díky tomu nabízí kompletní podporu programování v tomto programovacím jazyce. Skládá se z:

- programového vybavení, které kompletně nahrazuje standardní LEGO firmware pro NXT a zahrnuje Java Virtual Machine,
- knihovny Java tříd classes.jar, která implementuje leJOS NXJ rozhraní pro programování aplikací (tzv. API),
- linkeru, který linkuje uživatelské Java třídy s koncovkou .jar a classes.jar do podoby binárního souboru, který je možné nahrát a spustit na NXT,
- nástrojů pro PC, které umožňují nahrát leJOS firmware na NXT a následně spouštět programy na NXT,
- API pro PC, díky němuž lze psát na PC programy komunikující s leJOS NXJ programy běžícími na NXT pomocí datových proudů přes bezdrátové připojení Bluetooth nebo USB kabel
- a mnoha hotových příkladů programů.

Jedná se o open source projekt, do kterého přispívá mnoho lidí a zájem o něj stoupá. Důvodem je mnoho výhod, jako např.: podpora objektově orientovaného programování, možnost práce v profesionálním vývojovém prostředí jako jsou Eclipse nebo Netbeans,

podpora napříč operačními systémy Windows, Linux a Mac OS, velmi přesná kontrola motorů, pokročilá podpora navigace a mnoho dalších. [2]

1.2 Řídící jednotka LEGO NXT Brick

Řídící jednotka je nejdůležitější částí LEGO Mindstorms NXT sady. Spolu se stavebnicí je dodáván standardní LEGO Mindstorms firmware, který je na ní nahrán, ale lze ho snadno přemazat jiným systémem, čímž dojde k úplnému přemazání a ztrátě původních dat. Fyzicky obsahuje řídící jednotka displej, ovládací tlačítka, USB port, čtyři porty pro připojení senzorů označené čísly od 1 do 4 a tři porty pro připojení motorů označené velkými písmeny A, B a C. [3] [4]

1.2.1 Instalace

Nahrání leJOS NXJ firmwaru na NXT je součástí instalace leJOS NXJ softwarového balíku. Ale ještě před samotnou instalací je třeba vykonat několik kroků. Prvním je instalace vhodného USB ovladače. Tento krok odpadá, pokud jsme již dříve instalovali standardní LEGO Mindstorms software, proběhla i instalace příslušného ovladače. Pokud ne, je možné si nainstalovat ovladač z oficiálních stránek firmy LEGO. Dále je třeba nainstalovat JDK verze 1.6 nebo vyšší a cestu ke složce /bin z nainstalovaného JDK přidat do proměnné prostředí PATH, pokud se tak nestane automaticky. LeJOS funguje pouze s 32bitovou verzí JDK, proto je bezpodmínečně nutné nainstalovat ji, i pokud používá uživatel 64bitovou verzi operačního systému. Až poté je možné přejít ke stáhnutí aktuální verze leJOS softwaru z oficiálních stránek projektu leJOS a nainstalovat ji. V závěrečné části instalace je uživatel požádán o připojení zapnuté NXT jednotky a dochází k nahrání nového firmwaru. [3]

1.2.2 Ovládání řídící jednotky

Řídící jednotka NXT Brick obsahuje displej a čtyři intuitivně vypadající tlačítka, jak je zobrazeno na obrázku 1.



Obrázek 1 – Řídící jednotka NXT Brick (zdroj: [5])

Na obrázku jsou rozpoznatelná dvě tlačítka ve tvaru šipek, sloužící pro pohyb vlevo a vpravo, či nahoru a dolů v jednotlivých úrovních menu. Dále je možné vidět oranžové

potvrzovací tlačítko, které je zároveň vyhrazeno pro spouštění jednotky při jeho delším stisknutí a dole od něj menší tlačítko pro odchod z dané úrovně menu o úroveň výš a při jeho delším stisknutí dojde k vypnutí jednotky.

Po zapnutí jednotky se zobrazí základní menu nabízející kategorie, do kterých lze vstoupit a provést patřičná nastavení. Tyto kategorie jsou:

- **run** default pro spuštění přednastaveného programu,
- **files** pro zobrazení, mazání či spuštění našich souborů, tzn. nahraných programů,
- **bluetooth** pro zapnutí či vypnutí Bluetooth, nastavení viditelnosti, změnu PIN kódu jednotky, spárování se s jiným zařízením a prohlížení spárovaných zařízení,
- **sound** pro nastavení hlasitosti zvuků,
- **system** pro zobrazení volné flash a RAM paměti, stavu baterie, naformátování jednotky, nastavení času, po kterém se má jednotka vypnout a zapnutí nebo vypnutí funkce auto run, která pokud je aktivní a je přednastaven výchozí program, tak ho při zapnutí jednotky automaticky spustí
- a **version** pro zobrazení verze firmwaru.

Po celou dobu spuštění jednotky je v horní části displeje zobrazen aktuální stav baterie, název jednotky, aktivní USB a Bluetooth připojení a ikona oznamující, zda je Bluetooth zapnuto. [4]

1.3 Ovládání motorů

Bez motorů by byl LEGO robot naprosto nepohyblivý. Motory jsou zdrojem veškerých pohybů robota, proto hrají hlavní roli v robotickém programování a vývojáři věnovali mnoho času a úsilí tvořením a vyladováním tříd, které slouží k ovládání motorů. Cílem bylo dosažení co nejpřesnějších výsledků, což se podařilo. Když například přikážeme motoru, aby se osa kola otočila o 360 stupňů, pak se tak opravdu stane, bez jakéhokoliv překročení nebo nedotočení tohoto úhlu. Motory byly testovány i pod zátěží tak, aby se dokonce i při zatížení robota zastavila osa kola na místě, na kterém se zastavit má. Dále je programátorovi poskytováno řízení rychlosti otáčení motorů a dotazování se na aktuální rychlost.

Vývojářům se podařilo vyřešit problém nakloněné dráhy. Pokud robot stoupá po nakloněné dráze a programátor mu zadá určitou rychlost, je jí zkrátka dosaženo bez toho, aby se musel programátor starat o úhel naklonění dráhy. Hnací síla, která musí působit na motory, aby se robot i po nakloněné dráze pohyboval konstantní rychlostí, která byla zadána, je vypočítána a použita bez vědomí programátora, což znamená velmi komfortní přístup k určování rychlosti pohybu robota. Aby byl omezen skluz robota a jeho smýkání se při rychlém rozjezdu ze stoji na vysokou rychlost, poskytují nám třídy i metody pro ovládání

zrychlení. Díky nim může robot nabírat rychlost postupně, což působí pozitivně na jeho předpokládané chování a je to užitečné pro dosažení přesných výsledků.

Výše popsaných funkcí je dosaženo monitorováním motoru v reálném čase. Motory obsahují optické snímače, které poskytují impulsy po každém otočení osy o jeden stupeň. Je sledována aktuální hodnota úhlu otočení kola a rychlosti motoru, i síly potřebné k dosažení této rychlosti a díky tomu jsou schopny okamžitě zareagovat na jakoukoliv změnu, například i otočením směru otáčení. Této vlastnosti se hojně využívá při vytváření robotů s robotickým ramenem, kdy se uplatňuje výše zmíněná přesnost, protože se motor otáčí o malé rozsahy úhlů a často mění směr otáčení.



Obrázek 2 – LEGO NXT motor (zdroj: [6])

1.3.1 Třída `NXTRegulatedMotors`

Třída `NXTRegulatedMotors` je hlavní třídou poskytující programátorům přístup k ovládání motorů. Obsahuje tři statické instance: `Motor.A`, `Motor.B` a `Motor.C`. Metody této třídy umožňují kontrolu nad uvedením motoru do chodu v určitém směru, změnu směru otáčení, zastavení, určení konkrétní rychlosti nebo zrychlení či otočení kola o určitý úhel. Velká písmena označující instance motorů odpovídají názvům portů pro připojení motorů k NXT řídicí jednotce.

1.3.1.1 Základní metody třídy `NXTRegulatedMotor`

```
public final boolean isMoving()
```

Sděluje, zda se motor pohybuje.

```
public final boolean isStalled()
```

Sděluje, zda je motor zablokován nebo natolik zatížen, že není schopen pohybu.

```
public final void stop()
```

Způsobí okamžité zastavení motoru.

```
public void flt()
```

Způsobí, že motor začne ztrácet výkon a pozvolna zastaví.

```
public void rotate(int angle)
```

Způsobí otočení motoru o požadovaný úhel, zadaný ve stupních.

```
public void rotateTo(int limitAngle)
```

Způsobí otočení motoru k úhlu zadanému ve stupních.

```
public void setAcceleration(int acceleration)
```

Nastaví zrychlování motoru ve stupních/sekunda/sekunda. Výchozí hodnota je 6000 a nižší hodnoty způsobí vyšší zrychlení.

```
public final void setSpeed(int speed)
```

Nastaví rychlost motoru ve stupních za sekundu. Určitou hranicí je hodnota 900 stupňů za sekundu. Tuto a vyšší rychlost je možné dosáhnout pouze s plně nabitou baterií. Aktuální maximální dosažitelná rychlost je závislá na stavu baterie a zatížení motoru.

```
public final int getSpeed()
```

Sděljuje aktuální nastavenou rychlost ve stupních za sekundu. Nesděljuje aktuální rychlost motoru, k tomu slouží metoda `getRotationSpeed()`.

```
public int getLimitAngle()
```

Sděljuje úhel ve stupních, ke kterému se motor otáčí.

```
public int getRotationSpeed()
```

Sděljuje aktuální rychlost otáčení motoru ve stupních za sekundu. Pokud metoda vrátí zápornou hodnotu, znamená to, že se motor otáčí ve zpětném směru.

```
public int getTachoCount()
```

Sděljuje, o kolik se motor otočil ve stupních.

```
public void resetTachoCount()
```

Vynuluje hodnotu, uchovávající o kolik se otočil motor.

```
public void forward()
```

Způsobí otáčení motoru ve směru dopředu, dokud není zastaven jinou metodou.

```
public void backward()
```

Způsobí otáčení motoru ve směru dozadu, dokud není zastaven jinou metodou.

1.3.2 Další třídy pro práci s NXT motory

Třída `NXTRegulatedMotors` není jedinou třídou pro práci s NXT motory. Někdy může nastat situace, kdy chce mít programátor přímý přístup k síle otáčející osou kola motoru a nestačí mu přístup přes nastavení rychlosti otáčení. Pro tento případ slouží třída `NXTMotor`, která implementuje rozhraní třídy `BasicMotor` a má navíc přístup k metodám `getTachoCount()` a `resetTachoCount()`. Dále jsou k dispozici třídy sloužící pro zachování kompatibility se sérií stavebnic LEGO RCX a jejími motory. Jsou to třídy `MotorPort` a `RCXMotor`, o kterých je možné se dozvědět více v leJOS NXJ API dokumentaci. [3]

2 Senzory

Senzory jsou velmi důležitou součástí robotů dnešního typu. Jejich využívání povýšilo roboty na takřka inteligentní stroje. Pryč jsou doby, kdy nebylo možné rozpoznat překážku dřív, než do ní robot narazil, nebo nebyl robot schopen sám zjistit, že uvízl a patřičně na to reagovat. Senzory slouží k získávání informací o okolí robota, které probíhá a vyhodnocuje se v reálném čase. Díky tomu je robot schopen dozvědět se o překážce ještě dřív, než do ní narazí nebo se díky ní převrátí. A i kdyby na překážku najel, dokáže rozpoznat, že došlo k jeho naklonění a zareagovat na to například zastavením se nebo zařazením zpětného chodu. Takto inteligentního chování lze dosáhnout i u stavebnice LEGO Mindstorms NXT.



Obrázek 3 – Kompas senzor a ultrasonic senzor (zdroj: [7] [8])

LEGO senzory vypadají na pohled všechny stejně, jako například kompas senzor, mají stejný tvar a velikost. Jediným na první pohled odlišným zástupcem je ultrasonic senzor, viz obrázek 3.

Firma LEGO uvádí všechny senzory na trh pod svou značkou, ale není skutečným výrobcem všech senzorů. Výrobu některých z nich zajišťují takzvané firmy třetích stran, jako HiTechnic nebo Mindsensors. Obě tyto firmy vyrábějí například kompas senzor. Každá ho navrhla a naprogramovala jinak, ale jeho funkce a přístup k němu musí být stejné, aby bylo možné zajistit kompatibilitu všech senzorů s leJOS NXJ API a popsat jednotný přístup v leJOS API dokumentaci. Jako řešení zvolili vývojáři sérii společných softwarových rozhraní. Každá rodina senzorů má své softwarové rozhraní, které popisuje jeho základní funkce, takže máme například stejné softwarové rozhraní pro všechny kompas senzory, nehlédě na jejich výrobce. Jednotlivé rodiny se společným rozhraním jsou:

- Accelerometer,
- ColorDetector,
- DirectionFinder,
- Gyroscope,
- LightDetector,

- PressureDetector,
- RangeFinder,
- Tachometer,
- a Touch.

Všechny senzory se připojují k NXT stejným způsobem, zapojením do pro ně určených portů, které jsou označeny čísly 1 až 4. Standardně máme tedy možnost připojit maximálně čtyři senzory k jedné řídicí jednotce. V kódu programu musí být označeno, který port je používán. K tomu slouží třída SensorPort. Tato třída obsahuje čtyři statické objekty SensorPort.S1, SensorPort.S2, SensorPort.S3 a SensorPort.S4, které odpovídají čtyřem výše zmíněným fyzickým portům. Dále obsahuje mnoho metod umožňujících přímý přístup k datům přicházejícím na port buď pomocí I²C protokolu nebo z analogových senzorů. Tyto metody většinou programátor přímo nepoužívá, musí pouze předat konkrétní port jako vstupní parametr konstruktoru daného senzoru. Jedním z možných rozdělení senzorů je právě kritérium, zda se jedná o analogové senzory, nebo senzory využívající ke komunikaci I²C protokol.

2.1 Analogové senzory

Analogové senzory vracejí hodnoty na základě aktuálního napětí na senzoru. Tento způsob je velice rychlý, neboť zde není použita sběrnice, která by přenos zpomalovala. Dochází pouze ke kontrole napětí na senzoru a vrácení této hodnoty. Zástupci této skupiny senzorů jsou:

2.1.1 Light Detector

Toto rozhraní je implementováno dvěma třídami. Ze sady NXT verze 1.0 je to LightSensor a ze sady NXT verze 2.0 pak ColorSensor, který obsahuje některá vylepšení, ale základní funkce mají obě třídy stejné. Oba senzory slouží k rozeznávání slunečního svitu a dokáží změřit jeho intenzitu. Jsou vybavené červenou LED diodou a dokáží rozeznat i okem neviditelné infračervené světlo.

Zástupci rodiny LightDetector nabízejí dva režimy:

active mode, který lze poznat podle rozsvícené LED diody a v tomto aktivním režimu je senzor schopen například sledovat černou čáru, podle níž se robot pohybuje, nebo rozeznávat objekty tak, aby se jim robot mohl vyhnout a nenarazil do nich

a **passive mode**, který lze poznat podle zhasnuté LED diody a v tomto pasivním režimu je senzor schopen rozeznávat sluneční svit. Toho je využíváno například, pokud se robot pohybuje na plošině, která je výš než její okolí a nechceme, aby z ní sjel. Takovou plošinou může být stůl a okolním prostředím podlaha. Stůl, po kterém se robot pohybuje, odráží mnohem intenzivněji světlo, než vzdálená podlaha a tím vzniká hranice, kterou senzor bezpečně rozezná.

2.1.2 Color Detector

NXT verze 2.0 obsahuje jeden color senzor a další je vyráběn firmou HiTechnic. Oba implementují rozhraní rodiny ColorDetector a jsou podobní zástupcům výše uvedené rodiny LightDetector, ale hlavní rozdíl je v jejich schopnosti rozeznat jednotlivé barevné složky podle RGB modelu, zjistit jejich hodnoty a určit o kterou barvu z palety barev se jedná.

2.1.3 Touch Sensor

NXT verze 1.0 obsahuje jeden touch senzor a NXT verze 2.0 pak další dva touch sensory. Všechny implementují rozhraní rodiny Touch a jejich hlavní součástí je tlačítko, které se tlakem přepne. Zda bylo tlačítko přepnuto lze zjistit zavoláním metody isPressed(), která je jedinou metodou tohoto rozhraní a vrací jednoduše hodnotu datového typu boolean.

2.1.4 Sound Sensor

NXT verze 1.0 obsahuje jeden sound senzor, který slouží k měření hlasitosti zvuku. Jednotkou měření je decibel. Pokud je zdroj zvuku umístěn blíže, je hlasitost zvuku vyšší, pokud dále, je tomu naopak. Pomocí tohoto principu dokáže senzor vystopovat zdroj zvuku. Dále nám třída SoundSensor nabízí také druhý režim měření, který neměří přímo decibely, ale hlasitost zvuku v závislosti na rozsahu slyšitelnosti lidského ucha. Taková hodnota je pak vrácena v procentech.

2.2 Senzory používající I²C protokol

Tato skupina senzorů využívá ke komunikaci protokol I²C, který je všemi NXT porty určenými k připojení senzorů podporován. I²C je počítačová sériová sběrnice vyvinutá firmou Philips, která umožňuje přenos dat mezi NXT řídicí jednotkou a senzory. Tuto technologii využívají především senzory vyrobené jinými firmami, než je LEGO, tedy tzv. firmami třetích stran. Přenos dat je pomalejší, než u analogových senzorů, ale není třeba se obávat, že by tento způsob citelně snížil výpočtovou rychlost. Návrátové časy takového spojení jsou kolem 2 milisekund, což je naprosto dostačující. Dokonce mnohé senzory nejsou schopny vracet data takto rychle a tím samy zpomalují rychlost výpočtu, což je patrné například u ultrasonic senzoru, který vysílá zvukové vlny, které se odrazí od objektu a vrátí se zpět na senzor za určitý čas, který neumožňuje senzoru odpovědět rychleji. Značnou výhodou I²C protokolu je, že takto mohou být vráceny bohatší informace, než jen jedna hodnota, jako je tomu u analogových senzorů.

I²C Sensor je abstraktní třída pro množství senzorů připojených touto cestou. To znamená, že všechny instance těchto senzorů mohou volat metody této třídy. Typickými zástupci jsou UltrasonicSensor, CompassHTSensor a ColorHTSensor.

2.2.1 Ultrasonic Sensor

Je jediným z I²C senzorů, který je umístován přímo v sadách LEGO NXT Mindstorms. Měří svou vzdálenost od pevných objektů v palcích nebo v centimetrech. Je schopen zachytit objekt vzdálený až 255 centimetrů, ale přesné výsledky, na které se můžeme

spolehnout, vrací ve vzdálenosti od 6 do 180 centimetrů. Tím se rozumí odchylka měření do 3 centimetrů.

Ultrasonic sensor patří do rodiny RangeFinder, takže implementuje její rozhraní a funguje tak, že vysílá pro lidské ucho téměř neslyšitelný zvukový signál, který se od pevného objektu odrazí a vrátí se na senzor. Ten změří, jak dlouho to zvukovému signálu trvalo a díky znalosti rychlosti zvuku vypočítá vzdálenost objektu. Senzor snímá objekty v rámci kuželovitého výřezu, který se rozevírá pod úhlem 30 stupňů a ve vzdálenosti 180 cm, která je horní hranicí vzdálenosti, ve které je schopen dosahovat přesných výsledků, má kužel tedy 90 cm v průměru. Takto veliký kuželový výřez je dostačující k tomu, aby se robot vyhnul většině kolizí. [3]

3 Vzdálená komunikace s NXT

Motivací k vytvoření fungujícího spojení mezi NXT řídicí jednotkou a jiným zařízením je zpravidla potřeba ovládat robota dálkově, tedy bez osobního kontaktu. Třídy sloužící pro komunikaci jsou navrženy tak, aby byl kód nezávislý na druhu spojení. Ovládacím zařízením může být náš osobní počítač, se kterým lze realizovat spojení pomocí USB kabelu, což je často nevhodné řešení, protože lze zařízení od sebe vzdálit jen na vzdálenost nepřesahující délku kabelu. Mnohem pohodlnějším řešením je využití Bluetooth komunikace, kdy se data přenášejí bezdrátově a to nám umožňuje větší svobodu v pohybech robota nezávisle na poloze ovládacího zařízení. Bluetooth protokol umožňuje vytvoření osobní bezdrátové sítě (anglicky Wireless Personal Area Network, zkratka WPAN) mezi několika zařízeními. Pokud jsou všechna zařízení v síti přenosná, pak je přenosná celá síť.

3.1 Bluetooth komunikace s PC

Bezdrátová komunikace mezi PC a NXT vyžaduje na straně počítače použití Bluetooth adaptéru verze 2.0 nebo vyšší a nainstalování příslušného ovladače.

Programy mohou být spouštěny přímo na řídicí jednotce, ale není to jediný možný způsob. Existuje zrcadlový obraz NXJ API, který umožňuje spuštění programu na PC. Nazývá se PC Comms a je to balík obsahující téměř všechny třídy ze standardní NXJ API určené pro programy spouštěné přímo na řídicí jednotce a k tomu ještě několik tříd navíc. Je možné si ho detailně prozkoumat na oficiálních stránkách projektu leJOS. Hlavním rozdílem tedy je, že programy neběží na NXT jednotce, ale na PC, ze kterého jsou vysílány jednotlivé příkazy robotovi. Třetím způsobem je kombinace obojího, kdy běží na každém zařízení vlastní program a vzájemně spolu komunikují.

Před samotným použitím Bluetooth je nutné spárovat obě zařízení. Předpokladem úspěšného párování je zapnuté Bluetooth na obou zařízeních. Jedná se o důležitý bezpečnostní prvek, kdy je při připojování se k NXT vyžádán jeho PIN kód, který je možné zjistit nebo změnit přímo na NXT. Tím je zabezpečeno, že se k NXT připojuje jeho skutečný vlastník, který k němu má i fyzicky přístup. Po úspěšném párování je možné nahrávat programy na NXT přes Bluetooth.

3.2 Architektura komunikace

Nejsložitějším a zároveň nejkomplexnějším typem komunikace je vykonávání programu na NXT a PC zároveň. Často se na NXT provádějí ty části kódu, které obsahují časově náročné operace, aby proběhly co nejrychleji. Realizace takového spojení má podobu klient-server aplikace, přičemž předpokladem úspěšného ustálení spojení je již dříve zmíněné párování zařízení. Zařízení na serverové straně čeká na připojení, které iniciuje druhé zařízení, tedy strana klienta, kterou může reprezentovat nejen počítač, ale v dnešní době třeba také smartphone, neboli „chytrý telefon“.



Obrázek 4 – Architektura komunikace (zdroj: vlastní)

Do pozice serveru je typicky voleno NXT, protože jednak neposkytuje možnost vytvoření uživatelského rozhraní a také se výborně hodí pro splnění úkolů, které jsou od serveru v této architektuře komunikace očekávány:

- čekání na Bluetooth připojení,
- po úspěšném připojení čekání na příkazy od klienta,
- vykonání přijatých příkazů, typicky např. pohyb některé části robota,
- uzavření spojení a čekání na nové připojení.

V pozici klienta nám PC nabízí možnost pohodlného vytvoření uživatelského rozhraní pomocí vývojového prostředí. Budoucí uživatel tak může zasílat jednotce příkazy pomocí dnes tolik rozšířené aplikace okenního typu. Pokud máme na klientské straně jiné zařízení, jako např. smartphone, tak i ten nám nabízí uživatelské rozhraní aplikace a tento úkol splňuje. [3]

3.3 Propojení Android smartphone a Lego NXT

LeJOS třídy mohou být využívány jako součást android aplikací skrze leJOS PC API, která komunikuje s leJOS NXJ programy běžícími na NXT, na kterém je nahráný leJOS a nebo komunikuje se zařízením pomocí LCP protokolu (anglicky celým názvem LEGO Communications Protocol), při jehož použití nemusí být na NXT nahrán leJOS a lze s ním tedy komunikovat i pokud má nainstalovaný standardní LEGO firmware.

Ke spouštění a programování android aplikací je potřeba dodržet několik pokynů, jejichž kompletní seznam a popis je k dispozici ve výukovém tutoriálu věnovaném programování leJOS aplikací, konkrétně v sekci „Using leJOS with Android“. Hlavními body jsou:

- stáhnout si z oficiálních stránek androidu SDK vhodné pro operační systém, který uživatel vlastní,
- dále je vysoce doporučováno využít některé vývojové prostředí pro jednodušší programování a sestavování aplikací, přičemž nejvíce doporučovaným prostředím je v současné době Eclipse,
- do něj je třeba doinstalovat plugin umožňující programování android aplikací,
- po založení nového projektu importovat do složky /lib soubor pccomms.jar, který nalezneme ve složce, kam jsme nainstalovali leJOS a přidat cestu k němu do cest pro sestavování projektu, tzv. „build path“,
- v kořenovém adresáři SD karty vytvořit složku leJOS a
- provést několik změn přímo na zařízení s operačním systémem Android, především povolit ladění aplikací při připojení USB kabelu a povolit instalaci aplikací z jiných trhů.

LeJOS je open source projekt uložený na přístupném úložišti a po zaregistrování se do něj mohou vývojáři přispívat a sdílet vlastní programy. Z úložiště je také možné stáhnout si balík „LeJOSDroid samples“, který obsahuje několik příkladů, které ulehčí začínajícím programátorům začátek programování android aplikací komunikujících s NXT. [9]

4 Android

4.1 Vysvětlení pojmu Android

Android je software pro mobilní telefony zahrnující operační systém. V současné době je využíván stovkami milionů uživatelů ve více než 190 zemích světa a každým dnem jeho popularita roste. Jedná se o nejrychleji se rozpínající operační systém pro mobilní telefony. S každou novou verzí je uživateli poskytována rychlejší a hladší reakce na jeho dotazy. Spolu se svými partnery Android neustále posouvá hardwarové a softwarové hranice a přináší nové možnosti pro uživatele i vývojáře. Android SDK je volně dostupné ke stažení na oficiálních stránkách projektu a poskytuje nástroje a API potřebné k vyvíjení aplikací na platformě Android pomocí programovacího jazyku Java. Vývojářům je tedy umožněno vytváření výkonných aplikací, které používají nejnovější mobilní technologie. K vývoji lze použít standardní Java IDE jako je Eclipse. Všechny aplikace je poté možné okamžitě distribuovat pomocí internetového obchodu Google Play. Stačí se jen zaregistrovat, seznámit se s všeobecnými obchodními podmínkami a zaplatit registrační poplatek 25 dolarů.

4.2 Uvedení do problematiky programování Android aplikací

Android aplikace se programují v programovacím jazyce Java. Android SDK zkompiluje všechny zdrojový kód do jednoho Android balíčku s příponou .apk a ten může být poté využit zařízením s nainstalovaným operačním systémem Android k instalaci aplikace. Po instalaci je možné aplikaci kdykoliv spustit.

Android je Linuxový víceuživatelský operační systém, ve kterém každá aplikace je jiný uživatel, kterému systém přiřadí jedinečné uživatelské identifikační číslo. Toto číslo existuje pouze v rámci systému a pro aplikaci samotnou je neznámé. Je využíváno tak, že systém nastaví právo přístupu k souborům dané aplikace pouze pro aplikaci samotnou a ostatní aplikace přístup k těmto souborům nemají. Android zavádí princip minimálních oprávnění, což znamená, že každá aplikace má přístup pouze ke svým souborům a do součástí systému, které si vyžádá pro svou práci a získá k nim oprávnění. Například může aplikace požádat o oprávnění pro přístup k datům zařízení, jako jsou uživatelské kontakty, SMS zprávy, data na SD kartě a další. Všechna oprávnění jsou aplikaci přidělena již v době její instalace. To vytváří velmi bezpečné prostředí, ve kterém aplikace nemůže získat přístup k části systému, pro kterou nemá dané povolení.

Ve výchozím nastavení běží všechny aplikace ve vlastním linuxovém procesu. Android spouští proces, když některá ze součástí aplikace potřebujete být vykonána, poté proces ukončí, když už není potřeba, aby nadále běžel, nebo když musí systém získat paměť pro jiné aplikace. Každý proces má svůj vlastní virtuální stroj (VM), proto běží kód každé aplikace v izolaci od jiných aplikací. Při potřebě sdílet soubory je využito toho, že je možné, aby více aplikací sdílelo stejné identifikační číslo a tudíž mají vzájemný přístup ke svým souborům.

4.3 Základní komponenty aplikace

4.3.1 Activities

Aplikace se skládá z jedné či více aktivit a každá aktivita představuje jednu obrazovku s uživatelským rozhraním. Přestože aktivity společně vytvářejí soudržné uživatelské prostředí, každá z nich je nezávislá na ostatních, což znamená, že jiné aplikace mohou spustit jakoukoliv z těchto aktivit, pokud jim to aplikace, jejíž je aktivita součástí, dovolí.

4.3.2 Services

Service je komponenta běžící na pozadí aplikace, která se používá k provádění dlouhotrvajících operací a neposkytuje žádné uživatelské rozhraní. Příkladem může být přehrávání hudby na pozadí, zatímco uživatel využívá jinou aplikaci.

4.3.3 Content providers

Content provider neboli „poskytovatel obsahu“ spravuje sdílené soubory dat aplikace. Je možné ukládat data v systému souborů, v databázi SQLite nebo v jiném trvalém úložišti dat, ke kterému má aplikace přístup. Prostřednictvím této komponenty mohou ostatní aplikace zaslat dotaz na požadovaná data nebo je dokonce změnit. Například systém Android poskytuje content provider, který spravuje informace o kontaktech. Z toho vyplývá, že se každá aplikace s příslušnými oprávněními může dotázat na tyto data a například číst a zapisovat informace o konkrétní osobě.

4.3.4 Broadcast receivers

Broadcast receiver neboli všesměrový přijímač je komponenta, která reaguje na hlášení, které obdrží a to v rámci celého systému. Mnoho takových hlášení pochází přímo ze systému, jako například, že se vypla obrazovka, baterie je vybitá a další. Aplikace může také vyslat hlášení, například tak může dát vědět ostatním aplikacím, že byla stažena určitá data a jsou pro ně k dispozici. I když tyto komponenty nezobrazí uživatelské rozhraní, tak mohou vytvořit oznámení upozorňující uživatele, že došlo k dané události.

4.4 Manifest File

Aby mohl systém spustit některou z komponent aplikace, musí nejprve vědět o její existenci a to se dozví právě z tohoto souboru, který musí být umístěn v kořenovém adresáři projektu. Musí zde být deklarovány všechny komponenty, které jsou součástí dané aplikace. Dále manifest file uchovává všechna přístupová oprávnění, která aplikace požaduje, obsahuje také minimální verzi systému, která musí být na zařízení nainstalována, aby bylo možné aplikaci nainstalovat či všechna potřebná zařízení, která aplikace ke své funkci vyžaduje.

4.5 Aplikační zdroje

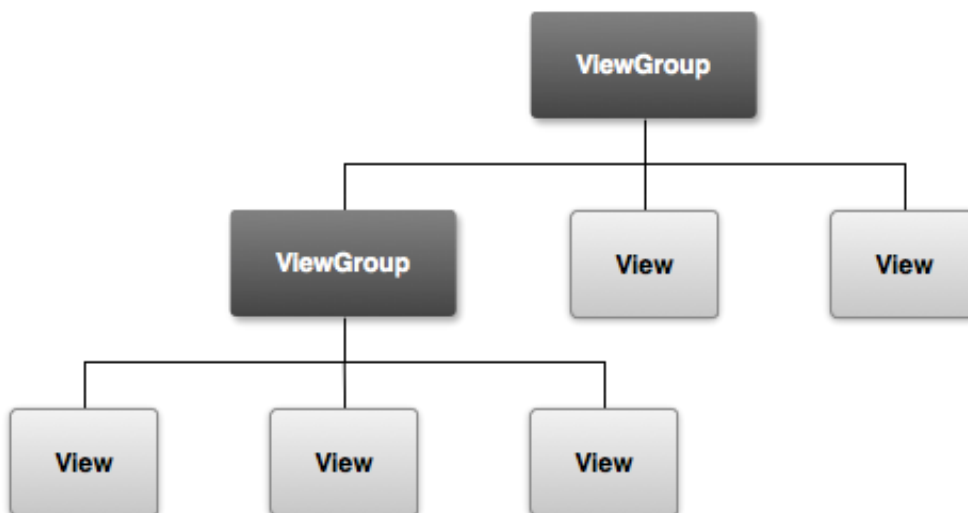
Android aplikace obsahují kromě kódu, který obsluhuje jejich průběh, také takzvané aplikační zdroje, které se využívají k oddělení prvků uživatelského rozhraní od samotného kódu aplikace a mají formát XML souboru. Jsou určeny k tomu, aby obsahovaly všechny

obrázky, zvukové soubory a vše co je spojené s vizuálním zobrazením aplikace. Dále by v nich mělo být definováno celkové rozvržení uživatelského rozhraní a všechny animace, styly, textové řetězce, barvy a všechna použitá menu. Toto oddělení zdrojového kódu od zbytku aplikace je na první pohled složité, ale při jeho využívání nabízí programátorovi velké výhody. Umožňuje aktualizaci různých zobrazení bez úprav ve zdrojovém kódu a využití alternativních zdrojů umožňuje optimalizovat aplikace pro různé konfigurace zařízení, jako například různé jazyky a velikosti obrazovky.

Pro každý zdroj, který programátor zahrne do svého projektu, je vygenerován jedinečný celočíselný identifikátor, kterým je dále možné se odkazovat na daný zdroj, jak ze zdrojového kódu aplikace, tak i z jiných zdrojů definovaných v XML. Například, pokud by aplikace obsahovala obrázek s názvem obr.png (uložený v res/drawable/adresář), nástroje SDK vygenerují identifikační číslo zdroje a pojmenují ho názvem R.drawable.obr, který může být poté použit jako odkaz na obrázek a pomocí něj lze obrázek vložit v uživatelském rozhraní. [10]

4.6 Uživatelské rozhraní

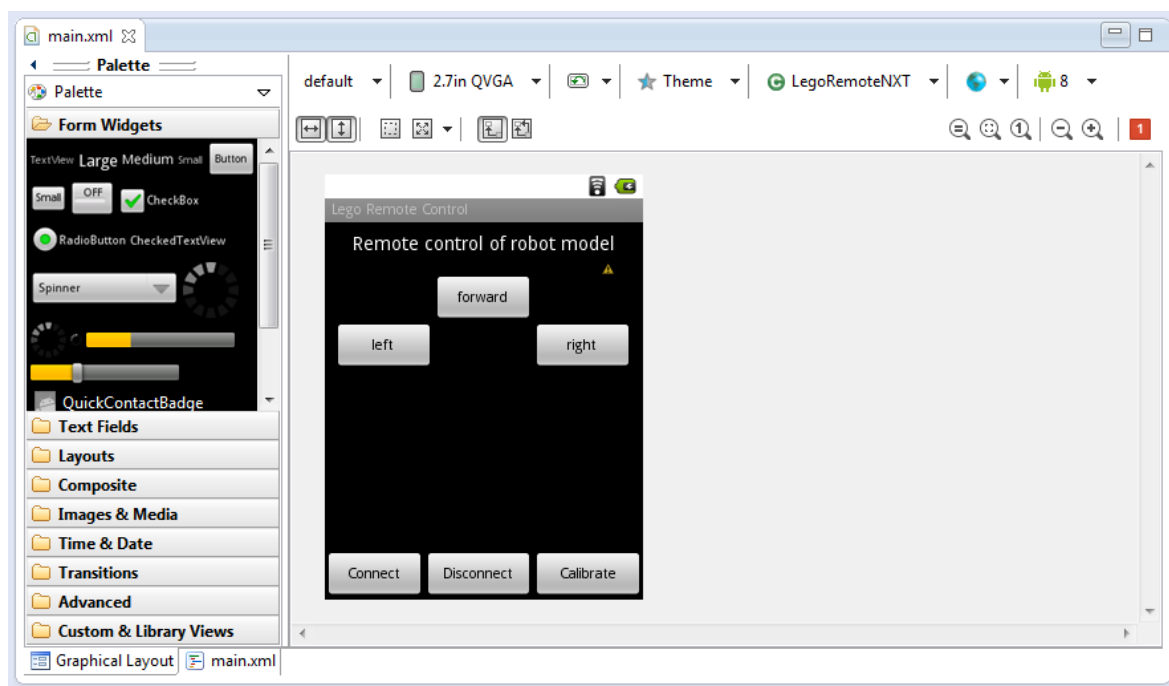
Všechny prvky uživatelského rozhraní Android aplikací jsou vystavěny pomocí objektů View (česky pohled) a ViewGroup (česky skupina pohledů). Pohled si můžeme představit jako jakýkoliv objekt vykreslený na obrazovce zařízení, například text nebo tlačítko. Tyto objekty jsou sdružovány do skupin pohledů, které je možné do sebe vnořovat a tím tvořit různá rozvržení uživatelského rozhraní aplikace. Vztah mezi těmito objekty je názorně zobrazen na obrázku 5.



Obrázek 5 – Vztah mezi objekty View a ViewGroup (zdroj: [12])

Je velmi doporučováno vytvářet rozvržení mimo zdrojový kód aplikace, za pomoci aplikačních zdrojů, konkrétně jako XML soubor s na první pohled čitelnou strukturou v podsložce kořenového adresáře /res/layout.

Pokud programátor využívá k vyvíjení své aplikace doporučené vývojové prostředí Eclipse a má v něm doinstalovaný ADT plugin, který poskytuje grafické rozhraní pro přístup k mnoha nástrojům příkazové řádky Android SDK a tím výrazně ulehčuje vývojářům práci, tak jednou z těchto pomůcek je i grafický editor pro tvorbu návrhu uživatelského rozhraní, který slouží jako velmi intuitivní nadstavba pracující nad XML souborem s rozvržením aplikace a umožňuje výběr z několika druhů rozvržení a přidávání komponent pouze pomocí tažení myši, viz obrázek 6. [11]



Obrázek 6 – Grafický editor pro návrh rozvržení Android aplikace (zdroj: vlastní)

5 Pohybování modelu robota

Jak bylo již dříve zmíněno, robot se může pohybovat pomocí tříd ovládajících motory jednoduše tak, že každý motor má schopnost otáčet se dopředu či dozadu. Avšak pokud chceme, aby se pohyboval po přesně dané dráze, je potřeba mít naprostou kontrolu nad každým jeho pohybem, nad ujetou vzdáleností a úhlem, pod kterým se pohybuje. Každý jakkoliv složitý pohyb se skládá ze 4 základních typů pohybu, kterými jsou: rovný přímý pohyb, pohyb po oblouku, otáčení se na místě kolem své osy a zastavení. Pohybující se robot vždy vykonává jeden z těchto pohybů a jejich sekvencí lze přesunout robota po jakkoliv navržené trase z bodu A do bodu B. Každý z těchto pohybů se navíc skládá ze dvou částí, které probíhají zároveň a charakterizují ho. Jsou jimi odjetí vzdálenosti a úhel, pod kterým je pohyb veden, přičemž při každém druhu pohybu jsou zastoupeny obě části, pouze jedna, nebo žádná, jak ukazuje tabulka níže.

Tabulka 1 – Rozbor pohybů modelu robota (zdroj: vlastní)

Typ pohybu	Vzdálenost	Úhel	Obměny
Pohyb rovně	+ nebo -	0	2
Otočení na místě	0	+ nebo -	2
Pohyb po oblouku	+ nebo -	+ nebo -	4
Zastavení	0	0	1

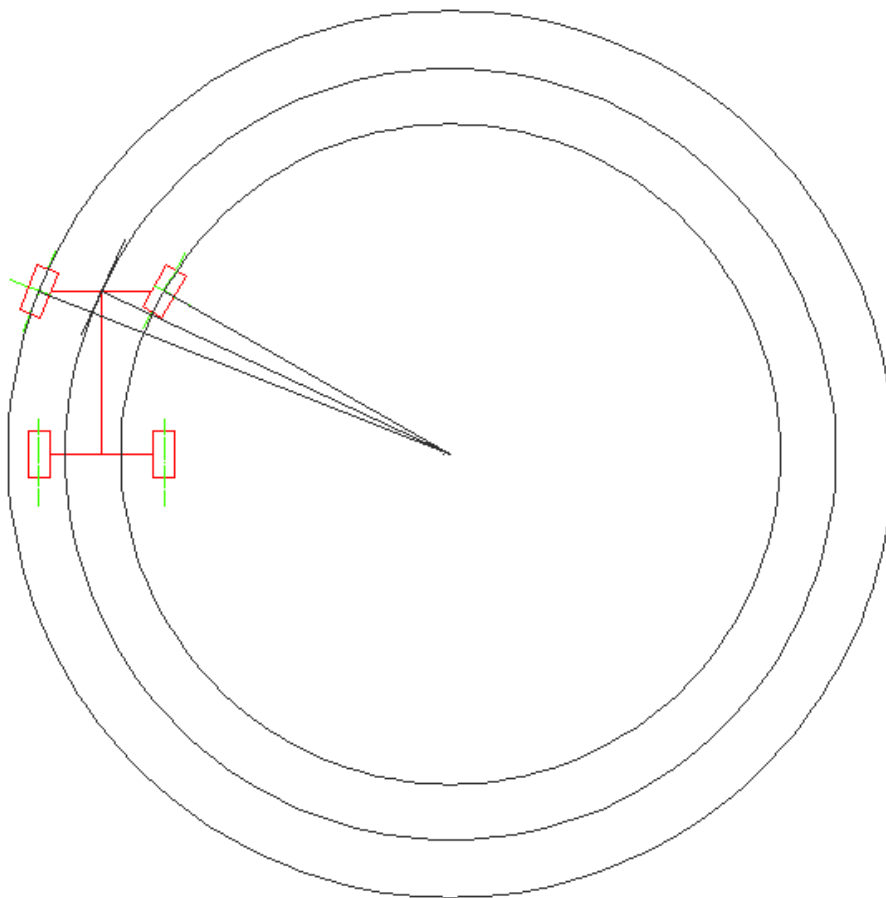
Plus symbolizuje pohyb vpřed a kladný úhel, symbolem mínus je naopak označen záporný úhel a pohyb vzad. Z tabulky jsou patrné dva druhy odjeté vzdálenosti, tedy vzdálenost vpřed a vzad a dva druhy úhlu, po kterém se robot pohybuje a to úhel ve směru hodinových ručiček a úhel proti směru hodinových ručiček.

5.1 Rovný přímý pohyb

Stavebnice LEGO Mindstorms NXT nabízí uživatelům ovladatelné motory, přičemž každý motor je možné ovládat zvlášť. Pro dosažení rovného přímého pohybu tedy platí, že motory pohánějící obě kola hnací nápravy se musí otáčet stejnou rychlostí. Pokud chceme ovládat model robota, který má kromě hnací nápravy ještě druhou zatáčecí nápravu, je potřeba nastavit této nápravě úhel tak, aby byla všechna čtyři kola v rovnoběžném stavu.

5.2 Průjezd obloukovou zatáčkou

Pokud se robot pohybuje po oblouku a plynule tak projíždí zatáčkou, opisují všechna jeho kola kružnici, jak je patrné z obrázku níže.



Obrázek 7 – Průjezd obloukovou zatáčkou (zdroj: vlastní)

U dvoukolových robotů s jednou hnací nápravou, bez řídicí nápravy, lze plynulou zatáčku vykonat pouze tím, že jeden motor otáčí kolem rychleji, než druhý. Naproti tomu u robotů s řídicí nápravou a hnanou zadní nápravou, které nápadně připomínají automobil, je situace značně složitější. Řešení pomocí natočení přední nápravy o určitý úhel a stejná rychlost otáčení kol hnací nápravy není vhodným řešením. První problém je s koly umístěnými na zadní nápravě. Problém nastává při zatočení předních kol. Zadní kola jsou na stejné nápravě a robot se má pohybovat po oblouku, což znamená, že vnitřní kolo musí opisovat kružnici o menším poloměru, než je poloměr kružnice, kterou opisuje kolo vnější. Neboli vnitřní kolo musí urazit za určitý čas menší dráhu než kolo vnější. Pokud by byla obě zadní kola poháněna stejnou rychlostí při zatočení přední nápravy, docházelo by ke značnému smyku a robot by projížděl zatáčkou více ze široka, než by měl. V důsledku toho by docházelo k výraznému opotřebování pneumatik a nápor na zadní nápravu by ji mohl v krajním případě i poškodit. Druhý problém se týká přední zatáčecí nápravy. I tady platí, že vnitřní kolo opisuje menší kružnici, než vnější. Pokud by byla obě kola natočena pod stejným úhlem, měla by tendenci opisovat kružnici o stejném poloměru. Proto musí být vnitřní kolo zatočené pod ostřejším úhlem, než kolo vnější, aby byla tato vlastnost potlačena. V opačném případě by opět docházelo k rychlejšímu opotřebení pneumatik. Jako vhodné řešení těchto problémů se jeví aplikace elektronického diferenciálu, který

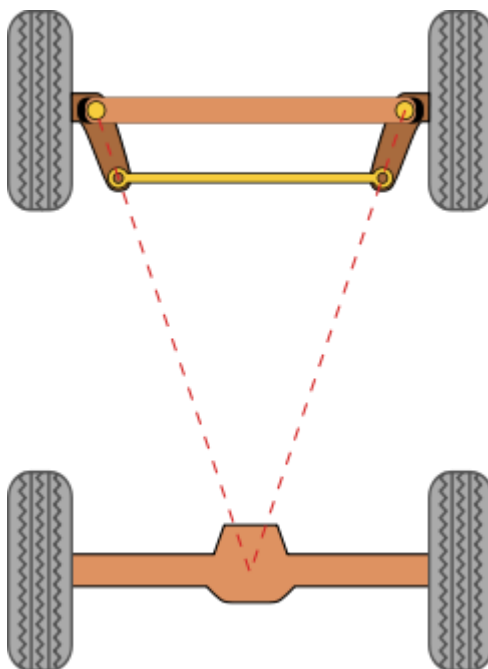
vypočítá rychlost každého z kol hnací nápravy v závislosti na úhlu natočení přední nápravy, v kombinaci s Ackermannovým řízením. [3]

5.3 Ackermannovo řízení

Ackermannova geometrie řízení je geometrické uspořádání přední nápravy vozidla, které řeší potřebu různého úhlu natočení vnitřního a vnějšího kola, která musí při plynulém průjezdu zatáčkou opisovat kružnice o různých poloměrech. Toto řešení vynalezl Georg Lankensperger v roce 1817 v Mnichově a následně ho v roce 1818 patentoval jeho zástupce v Anglii, Rudolf Ackermann.

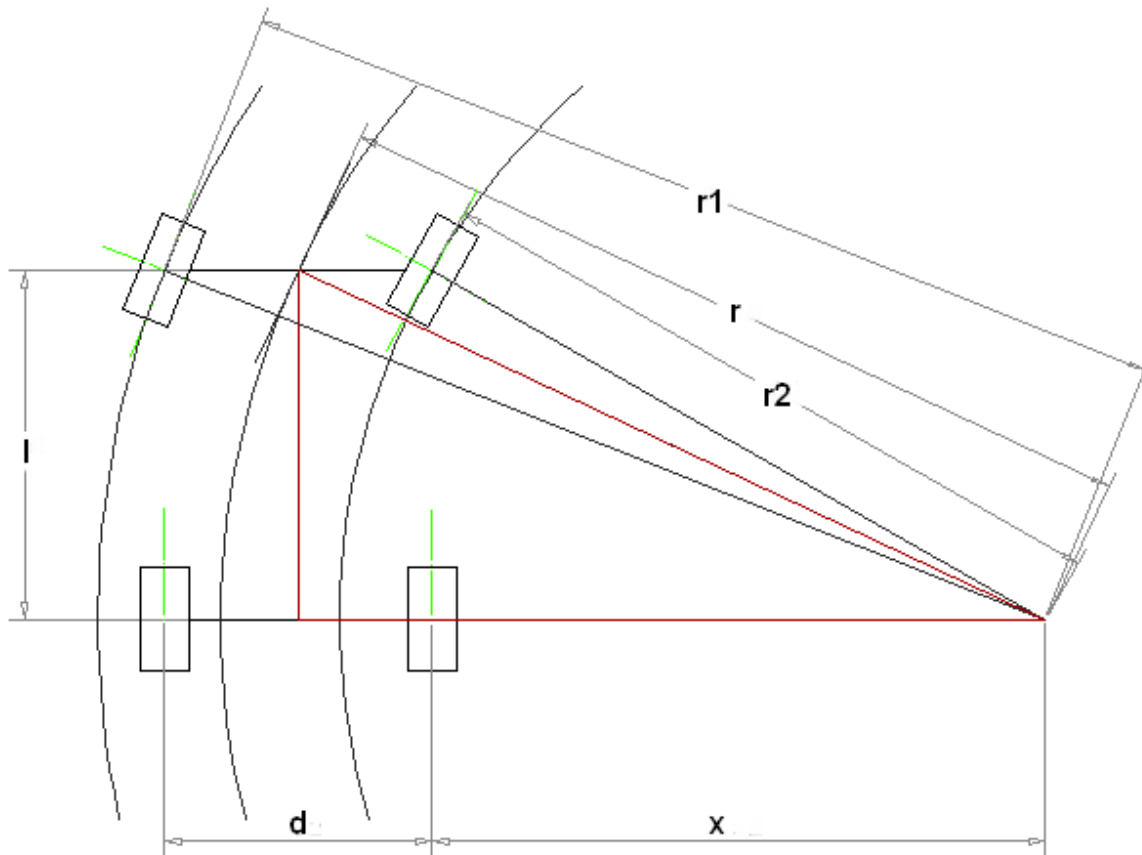
Záměrem Ackermannovy geometrie řízení je předejít skluzu pneumatik při průjezdu obloukovou zatáčkou. Toto geometrické řešení způsobí, že jsou osy všech kol uspořádány jako poloměry kružnice se společným středem. Jelikož zadní kola jsou na pevné nápravě, musí střed této kružnice ležet na jejich ose a je určen průsečíkem této osy s osami kol přední otočné nápravy. Jak je možné vidět na obrázku 9, je tím dosaženo různých úhlů natočení vnitřního a vnějšího kola přední nápravy. Kolo na vnitřní straně je natočeno pod větším úhlem, než kolo na vnější straně. Úhel natočení celé přední nápravy je dán aritmetickým průměrem úhlů natočení jejích kol.

Na obrázku 8 je zobrazeno dokonalé řešení Ackermannovy geometrie řízení, kdy se osy ramen přední nápravy střetávají ve středu osy zadní nápravy vozidla. Právě při takovém řešení je střed všech kružnic, které opisují jednotlivá kola, přesně ve stejném bodě. V praxi je ovšem technicky velmi obtížné tohoto řešení dosáhnout. [12]

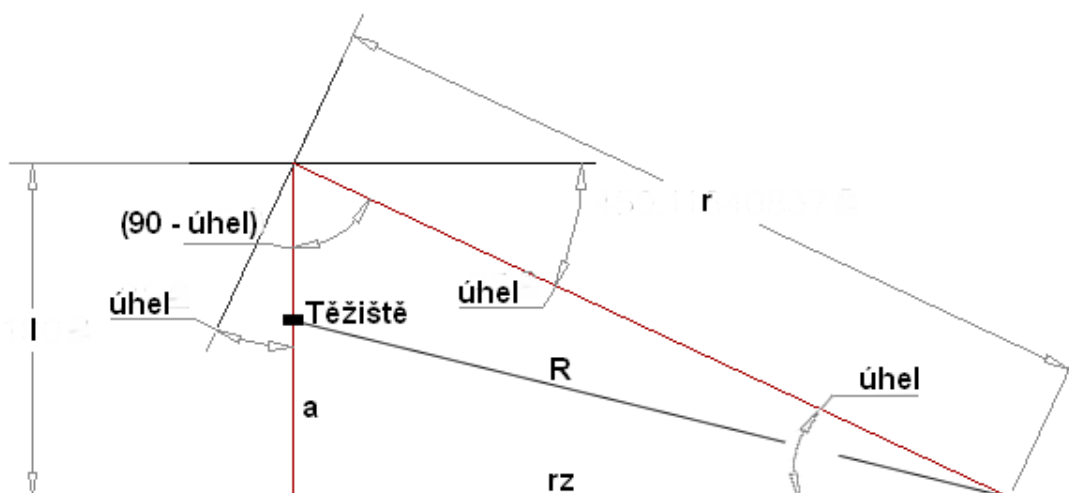


Obrázek 8 – Ackermannova geometrie řízení (zdroj: 12)

5.3.1 Matematický rozbor Ackermannova řízení



Obrázek 9 – Matematický model Ackermannova řízení (zdroj: vlastní)



Obrázek 10 – Pravoúhlý trojúhelník modelu Ackermannova řízení (zdroj: vlastní)

Základem pro výpočet poloměru otáčení u Ackermannova řízení, je sestavení pravoúhlého trojúhelníku, viz obrázek 10. Při znalosti vzdálenosti os náprav, na obrázku označené jako

l , a znalosti úhlu natočení přední nápravy, lze dopočítat poloměr otáčení zadní nápravy vztahem:

$$r_z = l \cdot \operatorname{tg}(90 - \text{úhel})$$

Poloměry otáčení kol zadní nápravy lze poté odvodit, pokud je známá vzdálenost otisků pneumatik nápravy, a to pouhým posunem o $d/2$. Výsledný vztah je:

$$r_{z_{1,2}} = l \cdot \operatorname{tg}(90 - \text{úhel}) \pm d / 2$$

Poloměr otáčení přední nápravy lze taktéž odvodit pomocí goniometrických funkcí v pravoúhlém trojúhelníku vztahem:

$$r = \frac{l}{\sin(\text{úhel})}$$

K dopočítání skutečného poloměru otáčení R je potřeba znát vzdálenost těžiště vozidla od osy zadní nápravy. Na obrázku je tato vzdálenost označena jako a . Poté výsledný vztah pro výpočet poloměru otáčení je: [13]

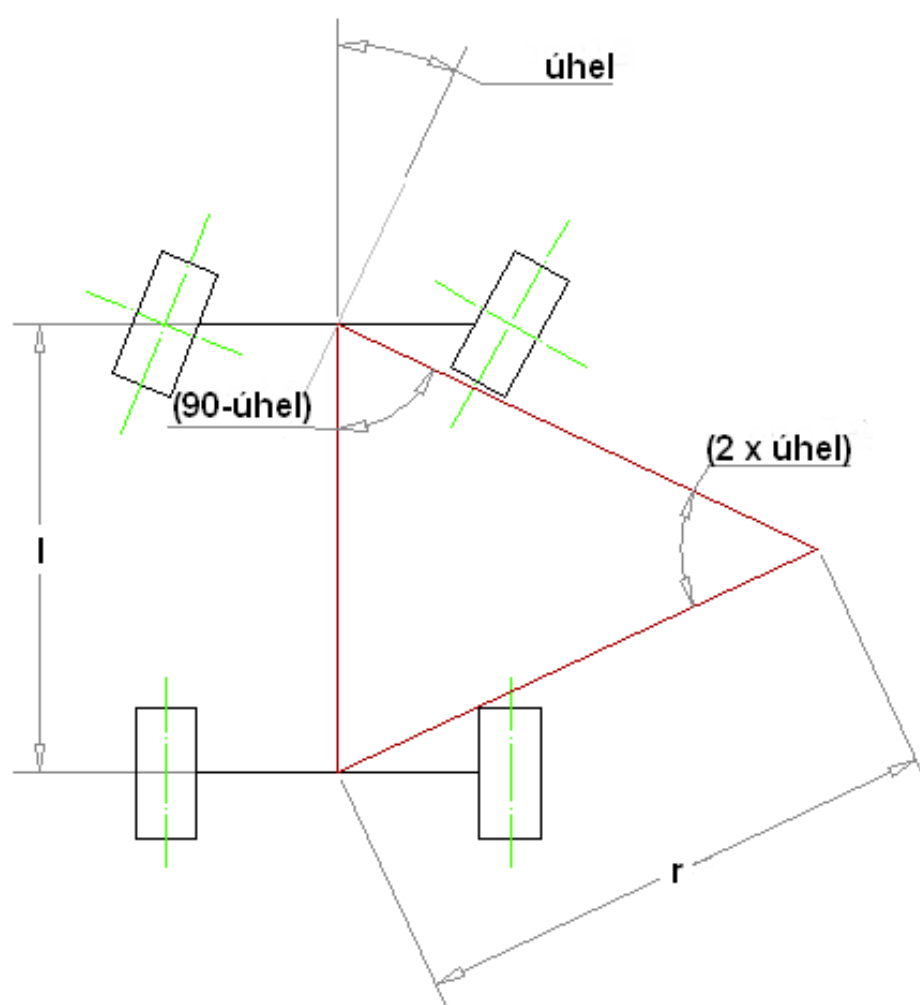
$$R = \sqrt{a^2 + l^2 \cdot \cot^2(\text{úhel})}$$

5.4 Elektronický diferenciál

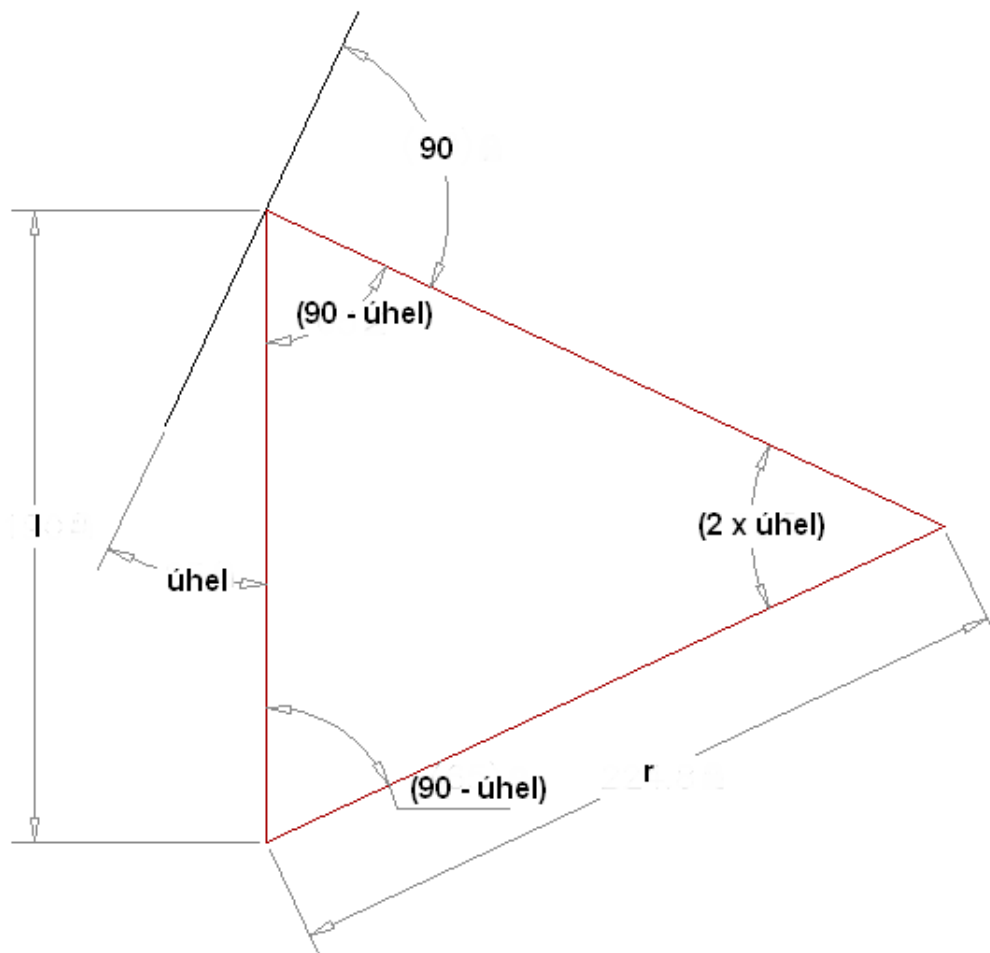
Elektronický diferenciál se využívá k určení rychlostí samostatně poháněných kol zadní hnací nápravy v závislosti na úhlu natočení přední nápravy. Vychází z toho, že při zatočení se změní dráha, kterou musí každé kolo urazit za daný čas. Postup je takový, že se vyhodnotí úhel natočení kol a z něj se vypočítá poloměr zatáčení a upraví se rychlost jednotlivých kol. Často se elektronický diferenciál používá ke zlepšení jízdních vlastností vozů s Ackermannovým řízením. Při jeho použití se zvýší ovladatelnost podvozku vozidla a naopak se sníží namáhání jeho kostry, zamezí se prokluzování kol a tím se předchází opotřebování pneumatik. Dále nedochází k vzájemnému přetlačování se hnacích motorů, ani při prudkém zatočení, a tím se snižuje jejich namáhání i spotřeba energie. [14]

5.4.1 Matematický model elektronického diferenciálu

Při převedení problému do matematické úrovně jsou potřebnými vstupními parametry výpočtu požadovaná dopředná rychlost vozidla a úhel aktuálního natočení přední nápravy. Ve výpočtu musí být jako konstanty zahrnuty parametry podvozku: vzdálenost os přední a zadní nápravy a vzdálenost středů otisků pneumatik jedné nápravy. Požadovanými výstupy jsou rychlosti levého a pravého zadního kola. Základním cílem je potlačení smyku a prokluzu kol. K odvození závislostí matematického modelu byly využity obrázky 11 a 12.



Obrázek 11 – Matematický model elektronického diferenciálu (zdroj: vlastní)



Obrázek 12 – Rovnoramenný trojúhelník modelu elektronického diferenciálu (zdroj: vlastní)

Základem je sestavení rovnoramenného trojúhelníku, jehož základna má délku vzdálenosti os náprav a ramena mají délku poloměru otáčení vozidla. Úhel, který svírají ramena trojúhelníku, má pak velikost dvojnásobku úhlu zatočení a úhly přiléhající k základně lze vypočítat odečtením úhlu zatočení od velikosti pravého úhlu. Pokud je známa velikost základny a úhlu natočení nápravy, lze ze sestaveného trojúhelníku vypočítat poloměr otáčení vztahem:

$$r = \frac{l \cdot \sin(90 - \text{úhel})}{\sin(2 \cdot \text{úhel})}$$

Následně při znalosti vzdálenosti os otisků pneumatik lze vypočítat poloměry otáčení pro jednotlivá kola vztahem:

$$r_{1,2} = r \cdot \left(1 \pm \frac{d}{2 \cdot r}\right)$$

Dále vycházíme z vědomí, že za stejný čas musí jednotlivá kola urazit odlišnou vzdálenost. Výsledný vzorec zahrnuje jako vstupní parametr požadovanou dopřednou rychlost, která je

v závislosti na úhlu zatočení jednotlivých kol přední nápravy přepočítána a výslednými hodnotami jsou rychlosti levého a pravého kola hnací zadní nápravy. Výsledný vztah je:

$$v_{1,2} = v \cdot \left(1 \pm \frac{d}{2 \cdot l \cdot \sin(90 - \text{úhel})}\right) \cdot \sin(2 \cdot \text{úhel})$$

Nevýhodou takto realizovaného matematického modelu elektronického diferenciálu je neschopnost reagovat na změnu podmínek, které působí na jednotlivá kola, jak to dokáže mechanický diferenciál. [14] [15]

6 Popis aplikační části

6.1 Požadavky

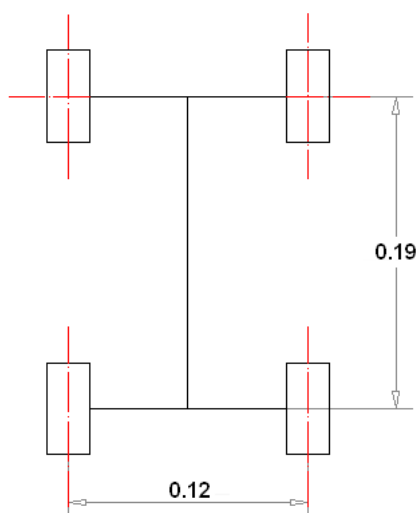
Hlavním cílem bylo implementovat matematický model elektrického diferenciálu rozebraný v kapitole 5.4, který umožňuje plynulé projíždění zatáček. Program pro řídicí jednotku Lego měl být naprogramován pro operační systémem LeJOS. Ovládání robota mělo být realizováno pomocí aplikace pro mobilní telefon s operačním systémem Android.

6.2 Použitý model robota

Byl použit model robota sestavený ze školní stavebnice Lego Mindstorms NXT, jehož podvozek měl přední nápravu sestavenou podle Ackermannovy geometrie řízení. Celkem robot obsahoval tři motory. Jeden ve smyslu zatáčení přední nápravy pomocí páky a zbylé dva na oddělený pohyb kol zadní hnací nápravy. Digitálním posuvným měřidlem byly naměřeny důležité hodnoty pro výpočet: vzdálenost os náprav o velikosti 0.19 m a vzdálenost os pneumatik zadní nápravy o velikosti 0.12 m, viz obrázky 13 a 14.



Obrázek 13 – Použitý model robota (zdroj: vlastní)



Obrázek 14 – Rozměry použitého modelu robota (zdroj: vlastní)

6.3 Použité technologie

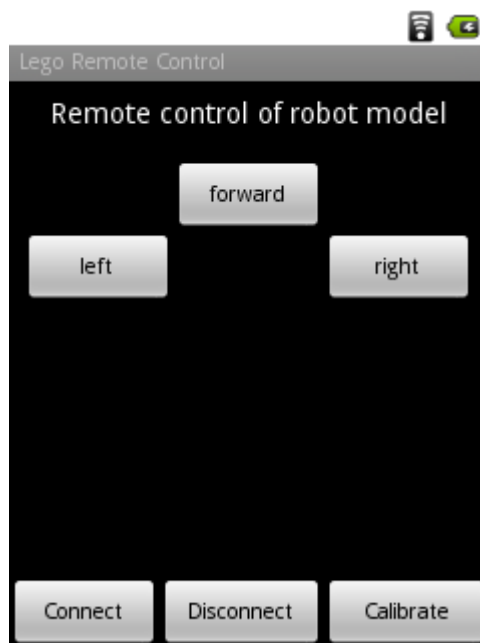
Pro programování všech částí aplikace bylo využito doporučené vývojové prostředí Eclipse Classic 4.2 ve 32bitové verzi. Do něj byly doinstalovány pluginy pro programování leJOS aplikací, pro programování Android aplikací a ObjectAid UML plugin pro pohodlné vytvoření UML diagramu tříd a vygenerování jejich závislostí. Android aplikace byla po celou dobu vývoje testována na mobilním telefonu Samsung Galaxy Ace s verzí operačního systému Android 2.2.1 Froyo, což odpovídá vývojovému stupni API 8 (anglicky API level). Na NXT jednotku byla nainstalována v současné době nejnovější verze operačního systému leJOS 0.9.1 beta.

6.4 Architektura klient-server

Aplikace odpovídá architektuře klient-server, kde klientem poskytujícím uživatelské rozhraní je aplikace běžící na mobilním telefonu s operačním systémem Android a serverová část je reprezentována NXT řídicí jednotkou, na které běží aplikace pod operačním systémem leJOS. Více bylo o dané architektuře popsáno v kapitole 3.2.

6.5 Klientská část aplikace

Po instalaci aplikace na mobilní telefon a jejím spuštění se uživateli zobrazí grafické rozhraní s intuitivním ovládacím menu v angličtině, viz obrázek 15.



Obrázek 15 – Uživatelské rozhraní aplikace (zdroj: vlastní)

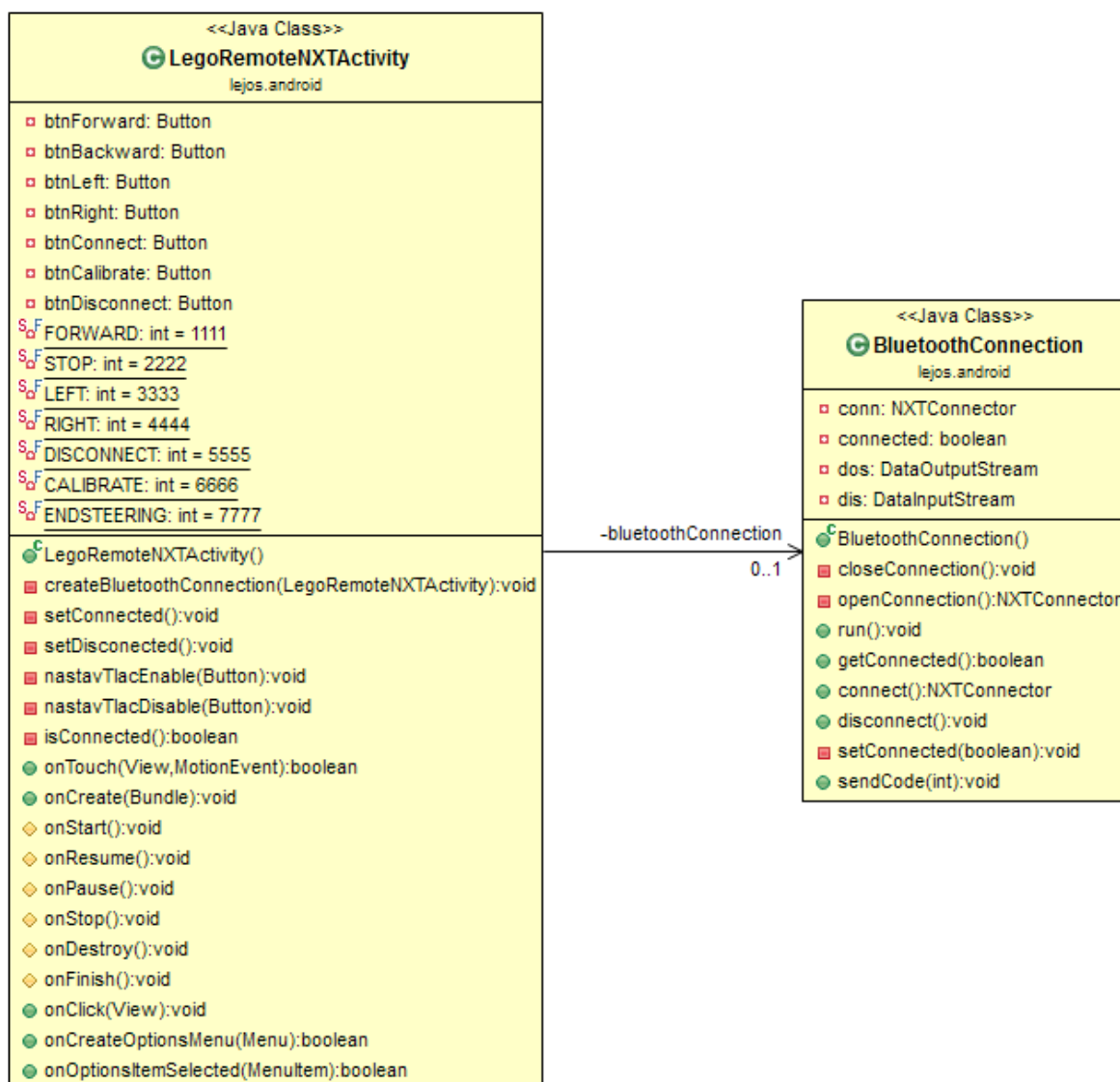
Menu obsahuje dvě části. V horní části jsou tlačítka pro řízení pohybu robota a ve spodní části se nacházejí tlačítka pro obsluhu připojení, odpojení a kalibraci. Menu se může nacházet ve dvou stavech. V prvním stavu, který je aktivní při spuštění aplikace a aktivuje se vždy, když dojde k odpojení aplikace od NXT, je fungující pouze tlačítko pro připojení se zobrazeným textem „Connect“ (česky připojit) a na tlačítku pro odpojení je zobrazen text „Disconnected“ (česky odpojeno). Kliknutím na tlačítko Connect se jeho text změní na „Connected“ (česky připojeno) a tlačítko se stane nefunkčním, naopak ostatní tlačítka přejdou do funkčního stavu a text tlačítka pro odpojení se změní na „Disconnect“ (česky odpojit).

Po úspěšném připojení by měl uživatel pomocí tlačítek left (česky vlevo) a right (česky vpravo) pro natočení nápravy srovnat přední nápravu robotického vozítka tak, aby byla natočena rovně a všechna čtyři kola byla tudíž rovnoběžně. V tomto stavu je nutné stisknout tlačítko Calibrate pro kalibraci, což způsobí, že se daný úhel natočení motoru ovládajícího přední řídicí nápravu vynuluje a tento stav se tedy v NXT uloží jako výchozí stav rovného pohybu vpřed. Po dokončení této akce je již bez problémů možné tlačítky left a right zatáčet přední nápravou a tlačítkem forward (česky vpřed) rozjet robota ve směru vpřed. Všechna tlačítka ovládající řízení fungují tak, že pohyb je vykonáván po dobu stisknutí tlačítka.

6.5.1 Diagram tříd klientské části aplikace

Tato část aplikace obsahuje dvě třídy. První se nazývá LegoRemoteNXTActivity a je rozšířením třídy Activity (česky aktivita), která se v programování Android aplikací používá jako základní stavební kámen zastupující jednu plochu zobrazenou na obrazovce, což znamená, že obsahuje uživatelské rozhraní. Tato třída v sobě uchovává instanci třídy

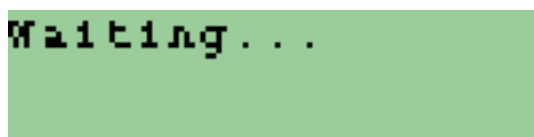
BluetoothConnection, která je rozšířením třídy Thread (česky vlákno) a stará se o vytvoření spojení s NXT, jeho udržení po dobu komunikace a následné odpojení.



Obrázek 16 – Diagram tříd klientské části aplikace (zdroj: vlastní)

6.6 Serverová část aplikace

Serverovou část aplikace tvoří program spustitelný na NXT, který po svém spuštění čeká na spojení, což dává najevo zobrazením textu „Waiting“ (česky čekající).



Obrázek 17 – Displej řídicí jednotky čekající na nové spojení (zdroj: vlastní)

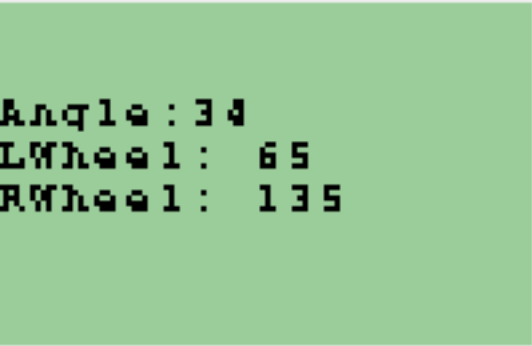
Pokud klient iniciuje spojení a dojde k jeho úspěšnému navázání, zobrazí se na displeji potvrzující text „Connected“ (česky připojeno).



```
Connected
```

Obrázek 18 – Displej řídicí jednotky po úspěšném navázání spojení (zdroj: vlastní)

Po navázání spojení se spustí cyklus, který trvá až do ukončení spojení a v rámci tohoto cyklu přijímá tato část aplikace od klienta povely a vykonává k nim příslušné akce. Když klient zašle povel pro kalibraci, dojde k vynulování údaje, který uchovává aktuální natočení přední nápravy. Poté se pokaždé po přijetí informace o tom, že byl změněn úhel natočení přední nápravy, vykoná přepočítání rychlostí jednotlivých kol zadní nápravy a na displeji se zobrazí jak aktuální úhel zatočení nápravy, tak vypočítaná rychlost pro obě kola zadní nápravy, přičemž zkratka „LWheel“ symbolizuje levé kolo a „RWheel“ kolo pravé.



```
Angle: 34  
LWheel: 65  
RWheel: 135
```

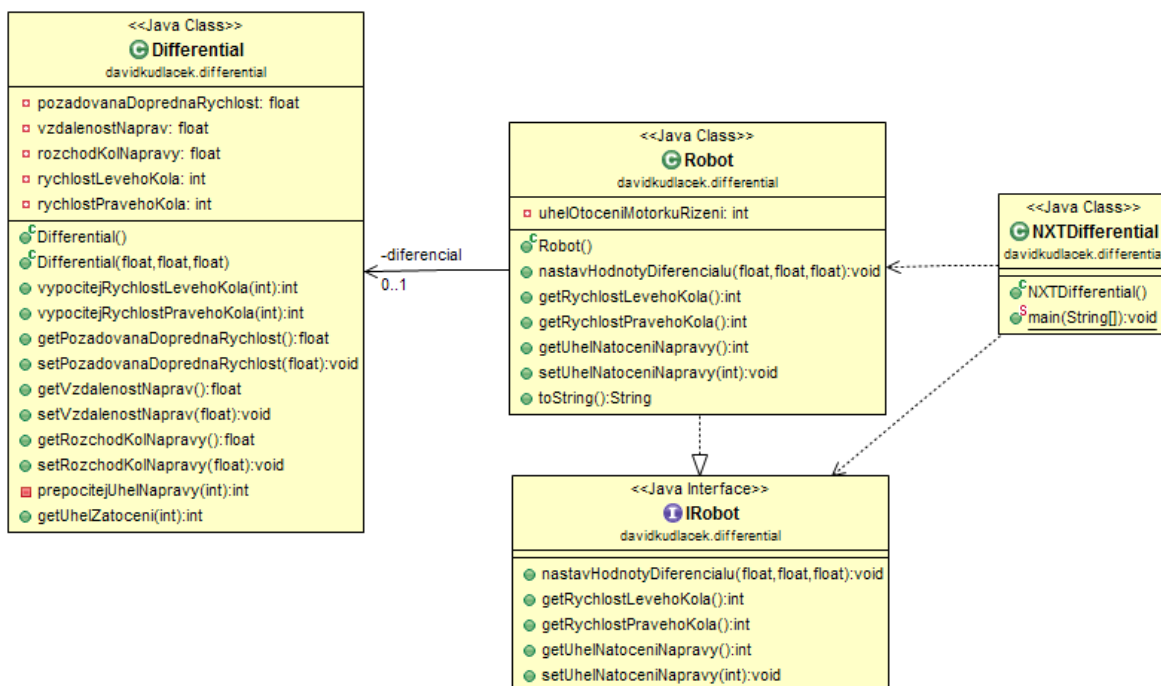
Obrázek 19 – Displej řídicí jednotky po natočení přední nápravy (zdroj: vlastní)

Při přijmutí povelu pro rozjetí robota ve směru vpřed se robot pohybuje vždy s aktuálně přepočítanými hodnotami rychlostí zadních kol.

Pokud klient žádá o ukončení spojení, je spojení ukončeno a aplikace se vrací do stavu, kdy čeká na nové spojení.

6.6.1 Diagram tříd serverové části aplikace

Tato část aplikace je tvořena ze tří tříd. Spouštěcím bodem je třída NXTDifferential se vstupním bodem programu, metodou main, která uchovává instanci rozhraní IRobot a obsahuje nekonečný cyklus čekající na příchozí spojení od klienta, vnořený cyklus reagující na pokyny klienta a také ukončuje spojení, pokud si o to klient zažádá. Třída Robot implementuje rozhraní IRobot a uchovává v sobě hodnotu aktuálního natočení přední nápravy a instanci třídy Differential, jejíž využívá metody. Třída Differential se stará o výpočty rychlostí zadních kol a je v ní tedy implementován matematický model elektronického diferenciálu.



Obrázek 20 – Diagram tříd serverové části aplikace (zdroj: vlastní)

6.7 Komunikace klientské a serverové části

Klientská Android aplikace si vyžádá spojení.

```

private NXTConnector openConnection() {
    NXTConnector conn = new NXTConnector();
    conn.setDebug(true);
    this.connected = conn.connectTo("btspp://");
    dos = new DataOutputStream(conn.getOutputStream());
    dis = new DataInputStream(conn.getInputStream());
    return conn;
}
  
```

Na přijmutí spojení čeká serverová část.

```

BTConnection btc = Bluetooth.waitForConnection();
  
```

Klient uchovává prostřednictvím konstant kódy, které reprezentují požadavky uživatele přijaté prostřednictvím stisku nebo uvolnění konkrétního tlačítka.

```

private static final int FORWARD = 1111;
private static final int STOP = 2222;
private static final int LEFT = 3333;
private static final int RIGHT = 4444;
private static final int DISCONNECT = 5555;
private static final int CALIBRATE = 6666;
private static final int ENDSTEERING = 7777;
  
```

Tyto kódy odesílá metoda reagující na změnu stavu tlačítka.

```

public boolean onTouch(View v, MotionEvent event) {
    int a = event.getAction();
  
```

```

    if (v == btnForward) {
        if (a == MotionEvent.ACTION_DOWN) {
            bluetoothConnection.sendCode(FORWARD);
        } else if (a == MotionEvent.ACTION_UP) {
            bluetoothConnection.sendCode(STOP);
        }
    } else if (v == btnLeft) {
        if (a == MotionEvent.ACTION_DOWN) {
            bluetoothConnection.sendCode(LEFT);
        } else if (a == MotionEvent.ACTION_UP) {
            bluetoothConnection.sendCode(ENDSTEERING);
        }
    } else if (v == btnRight) {
        if (a == MotionEvent.ACTION_DOWN) {
            bluetoothConnection.sendCode(RIGHT);
        } else if (a == MotionEvent.ACTION_UP) {
            bluetoothConnection.sendCode(ENDSTEERING);
        }
    }
    return true;
}

```

Ta volá metodu, která odesílá kód, který přijme jako vstupní parametr.

```

public void sendCode(int code) {
    if (getConnection()) {
        try {
            System.out.println("Sending " + code);
            dos.writeInt(code);
            dos.flush();
        } catch (IOException ioe) {
            System.out.println("IO Exception writing bytes:");
            System.out.println(ioe.getMessage());
        }
    }
}

```

Na serverové části běží cyklus, který po otevření vstupního proudu přijme kód.

```

DataInputStream dis = btc.openDataInputStream();
int n = dis.readInt();

```

Porovná přijatý kód v konstrukci switch s očekávanými kódy. Při shodě je vykonána příslušná akce. Pokud je přijatým kódem požadováno ukončení spojení, je spojení ukončeno a NXT se vrací do stavu, kdy čeká na nové spojení.

Závěr

Práce je zajímavá tím, že se skládá z několika tematických celků, které by sami o sobě mohly být náplní bakalářské práce. Snažil jsem se dát jim ve své práci stejný prostor a dívat se na každý z nich v souvislosti s jeho využitím v mé práci. Základní logické celky odpovídají osnově, přičemž pouze použití senzorů jsem si prakticky nevyzkoušel.

Dalším přínosem této práce je, že jsem se nesetkal s uceleným materiálem v českém jazyce, který by se zabýval tématy: programování Android aplikací, LEGO NXT Mindstorms a jejich vzájemná komunikace, což dokládá i výrazná převaha pramenů v angličtině.

V teoretické části bylo poskytnuto seznámení s robotickou stavebnicí LEGO NXT Mindstorms a bylo vysvětleno její možné využití při komunikaci s mobilním telefonem, který využívá operační systém Android. Dále byl stručně popsán úvod do programování Android aplikací a byly detailněji rozebrány matematické modely Ackermannovy geometrie řízení a elektronického diferenciálu. Taktéž byla popsána aplikační část.

V rámci aplikační části byly vytvořeny dvě aplikace. První je klient v podobě Android aplikace, poskytující uživatelské rozhraní a druhá je program spouštěný na NXT řídicí jednotce zastupující server, který se stará o výpočet rychlosti každého z kol zadní hnací nápravy v závislosti na úhlu natočení přední nápravy robota automobilového typu. Bylo dosaženo správné implementace elektronického diferenciálu podle matematického modelu rozebraného v teoretické části.

Tento projekt byl vytvořen pouze na výzkumné účely a nemá motivaci komerčního využití, jeho největší přínos je ve vědomostech, které jsem při jeho tvorbě nabyl. Jako velmi zajímavé rozšíření do budoucna se nabízí využití inerciálních senzorů v mobilním telefonu, jako jsou akcelerometry a gyroskopy.

Literatura

- [1] Lego Mindstorms NXT. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, poslední aktualizace 13. srpna 2012 [cit. 2012-08-13]. Dostupné z: http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT
- [2] The leJOS NXJ Tutorial: What is leJOS NXJ. *LEJOS: Java for Lego Mindstorms. The leJOS NXJ Tutorial* [online]. 2012 [cit. 2012-08-14]. Dostupné z: <http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/Intro.htm>
- [3] BAGNALL, Brian. *Intelligence Unleashed: Creating LEGO NXT robots with Java*. Winnipeg, Manitoba: Variant Press, 2009. ISBN 978-0-9868322-0-8.
- [4] The leJOS NXJ Tutorial: The leJOS NXJ Menu System. *LEJOS: Java for Lego Mindstorms. The leJOS NXJ Tutorial* [online]. 2012 [cit. 2012-08-14]. Dostupné z: <http://lejos.sourceforge.net/nxt/nxj/tutorial/MenuSystem/MenuSystem.htm>
- [5] RCX vs. NXT. *University of Nebraska Omaha* [online]. [cit. 2012-08-15]. Dostupné z: <http://aimforthestars.unomaha.edu/pages/rcxnxt.php>
- [6] Lego Mindstorms NXT 2.0. *Computero* [online]. [cit. 2012-08-15]. Dostupné z: <http://blog.computero.com.br/lego-mindstorms-nxt-2-0/>
- [7] Sensors for Lego NXT: Lego Mindstorms NXT ultrasonic sonar sensor. *Generation robots* [online]. [cit. 2012-08-15]. Dostupné z: <http://www.generationrobots.com/sonar-sensor-robot-lego-mindstorms-nxt,us,4,Capteur-Ultrasons-NXT.cfm>
- [8] Lego Mindstorms NXT. *NEXT SYSTEM Robotics Learning Center* [online]. [cit. 2012-08-15]. Dostupné z: <http://nextsys.web.id/edukasi/lego-mindstorms-nxt>
- [9] The leJOS NXJ Tutorial: Using leJOS with Android. *LEJOS: Java for Lego Mindstorms. The leJOS NXJ Tutorial* [online]. 2012 [cit. 2012-08-14]. Dostupné z: <http://lejos.sourceforge.net/nxt/nxj/tutorial/Android/Android.htm>
- [10] Application Fundamentals. *Android: Develop* [online]. 2012 [cit. 2012-08-14]. Dostupné z: <http://developer.android.com/guide/components/fundamentals.html>
- [11] User Interface Overview. *Android: Develop* [online]. 2012 [cit. 2012-08-15]. Dostupné z: <http://developer.android.com/guide/topics/ui/overview.html>
- [12] Ackermann steering geometry. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, poslední aktualizace 28. července 2012 [cit. 2012-08-14]. Dostupné z: http://en.wikipedia.org/wiki/Ackermann_steering_geometry

[13] Steering Dynamics. *Institute for Dynamic Systems and Control* [online]. [cit. 2012-08-14]. Dostupné z:

http://www.idsc.ethz.ch/Courses/vehicle_dynamics_and_design/11_0_0_Steering_Theroy.pdf

[14] SITTA, Michal. Elektronický diferenciál. *TIM2 Robotics* [online]. poslední aktualizace 2010 [cit. 2012-08-14]. Dostupné z: <http://www.tim2.wz.cz/diferencial.php>

[15] SITTA, Michal. *Elektronika a řídicí systém mobilního robotu*. Brno, 2010. Dostupné z: <http://www.tim2.wz.cz/download/dp/xsitta00.pdf>. Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce doc. Ing. Luděk Žalud, Ph.D.

Příloha A – Obsah přiloženého CD

Příložené CD má následující adresářovou strukturu:

/pdf/	Obsahuje elektronickou verzi tohoto souboru ve formátu PDF.
/javadoc/	Obsahuje vygenerovanou dokumentaci zdrojového kódu aplikace.
/nxt/	Obsahuje serverovou část aplikace běžící na NXT.
/android/	Obsahuje klientskou část aplikace pro operační systém Android.