ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDICÍ TECHNIKY

BAKALÁŘSKÁ PRÁCE

Využití robotů LEGO MINDSTORMS – návrh soutěžní úlohy (robotický fotbal)

Praha, 2011

Autor: Adam Šolc

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze, dne _____ _____

podpis

Poděkování

Děkuji především vedoucímu bakalářské práce panu Ing. Martinu Hlinovskému, Ph.D za poskytnuté materiály a za ochotu při řešení problémů. Dále bych rád poděkoval své rodině za podporu při studiu.

Abstrakt

Cílem této bakalářské práce je navrhnout soutěžní úlohu robotický fotbal, která bude mít obdobná pravidla jako klasický stolní fotbal. Roboti budou sestaveni ze stavebnice Lego Mindstroms a ovládáni šesti hráči mobilními telefony. Komunikace mezi roboty a mobilními telefony bude provozována na bezdrátovém rozhraní Bluetooth. Práce rozebírá jak konstrukční řešení robota, tak i návrh řídicího programu. Práce popisuje také návrh aplikace pro ovládání robota pro mobilní zařízení podporující Java ME.

Abstract

The aim of this thesis is to suggest the robotic soccer competition, which will have similar rules as the classic table football. Robots will be assembled from Lego Mindstroms kit, and controlled by six players with mobile phones. Communication between robots and mobile phones will be operated on the wireless Bluetooth technology. The thesis discusses structure solution of the robot, and the design of control programs. It also includes design of application to control the robot for mobile devices supporting Java ME.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Adam Šolc**

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný
Obor: Kybernetika a měření

Název tématu: **Využití robotů LEGO MINDSTORMS – návrh soutěžní úlohy (robotický fotbal)**

Pokyny pro vypracování:

1. Seznamte se s možnostmi robota LEGO Mindstorms (současný stav, HW a SW vybavení).
2. Nastudujte možnosti programování robota LEGO Mindstorms a seznamte se s možnostmi využití ovládání LEGO robotů pomocí bluetooth komunikace s použitím mobilního telefonu.
3. Proveďte návrh úlohy – robotický fotbal včetně dokumentace s konstrukcí robota (vytvořené v LEGO designéru) a způsobu jeho programování.
4. Vytvořte webovou stránku k navržené soutěžním úlohám.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Martin Hlinovský, Ph.D.

Platnost zadání: do konce zimního semestru 2011/2012

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 12. 10. 2010

Obsah

OBSAH ..... **VIII**

SEZNAM OBRÁZKŮ ..... **XI**

SEZNAM TABULEK ..... **XII**

1 ÚVOD DO BAKALÁŘSKÉ PRÁCE **1**

1.1 CÍL PRÁCE 1

1.2 OBSAH PRÁCE 1

2 STAVEBNICE LEGO MINDSTORMS **2**

2.1 PŘEDSTAVENÍ STAVEBNICE 2

2.2 INTELIGENTNÍ ŘÍDICÍ JEDNOTKA NXT 3

2.3 AKČNÍ ČLENY STAVEBNICE 4

2.3.1 SERVOMOTOR 4

2.4 SENZORY 5

2.4.1 SVĚTELNÝ SENZOR 5

2.4.2 ZVUKOVÝ SENZOR 6

2.4.3 DOTYKOVÝ SENZOR 6

2.4.4 ULTRAZVUKOVÝ SENZOR 7

2.4.5 PROPOJOVACÍ KABELY 8

2.5 PROGRAMOVÁNÍ NXT ŘÍDICÍ JEDNOTKY ROBOTA 8

2.5.1 NXT-G 8

2.5.2 NXC 9

2.5.3 LEJOS NXJ 9

2.5.4 MATLAB TOOLBOX 9

3 TECHNOLOGIE BLUETOOTH **10**

3.1 DŮVOD VZNIKU TECHNOLOGIE BLUETOOTH 10

3.2 HISTORIE BLUETOOTH 10

3.3 VLASTNOSTI TECHNOLOGIE 11

3.4 STRUKTURA PROTOKOLŮ 11

3.5 PROFILY BLUETOOTH 13

4 JAVA ME **14**

4.1 KONFIGURACE JAVA ME 15

4.1.1 CDC ... 15
4.1.2 CLDC ... 15
4.2 PROFILY JAVA ME ... 16
4.2.1 MIDP ... 16
4.3 JAVA ME A BLUETOOTH ... 18
5 NÁVRH SOUTĚŽNÍ ÚLOHY ROBOTICKÝ FOTBAL ... 19
5.1 PRAVIDLA A HŘIŠTĚ ... 19
5.2 KONSTRUKCE ROBOTA ... 20
5.2.1 POHON ROBOTA ... 20
5.2.2 ODPALOVACÍ MECHANIZMUS ... 21
5.2.3 KONSTRUKCE PRO SLEDOVÁNÍ ČÁRY ... 22
5.2.4 OMEZENÍ POHYBU ROBOTA ... 22
5.2.5 SESTAVENÍ ROBOTA ... 22
6 APLIKACE PRO OVLÁDÁNÍ ROBOTA ... 24
6.1 PŘÍPRAVA VÝVOJOVÉHO PROSTŘEDÍ PRO NÁVRH APLIKACE ... 24
6.2 NÁVRH APLIKACE „ROBOT CONTROL“ ... 25
6.2.1 TŘÍDA MIDLET ... 26
6.2.2 TŘÍDA BLUETOOTH ... 26
6.2.3 TŘÍDA CONTROL ... 27
6.3 VYUŽITÍ APLIKACE ... 27
6.4 ZÁVĚR ... 28
7 NÁVRH PROGRAMU A NASTAVENÍ ROBOTA ... 29
7.1 PŘÍPRAVA PROSTŘEDÍ PRO NÁVRH ALGORITMU ... 29
7.2 SLEDOVÁNÍ ČÁRY ... 29
7.2.1 VÝBĚR REGULÁTORU A JEHO POPIS ... 30
7.2.2 TŘÍDA LINEFOLLOWER ... 31
7.3 OTÁČENÍ ROBOTA ... 32
7.3.1 TŘÍDA ROBOTROTATING ... 32
7.4 BLOKOVÁNÍ POHYBU ROBOTA ... 35
7.5 HLAVNÍ TŘÍDA PROGRAMU ROBOTA ... 35
7.5.1 TŘÍDA FOTBALLPLAYER ... 36
7.6 UVEDENÍ ROBOTA DO PROVOZU ... 37
7.6.1 LOGOVÁNÍ DAT DO SOUBORU ... 38
7.6.2 NASTAVENÍ REFERENČNÍ HODNOTY REGULÁTORU ... 38
7.6.3 NASTAVENÍ PRAHU PŘEKLOPENÍ PŘI OTÁČENÍ ROBOTA ... 39
7.6.4 NASTAVENÍ REGULÁTORU ... 39

7.7 ZMĚNA JMÉNA ROBOTA 41

8 NÁVRH WEBOVÝCH STRÁNEK 42

9 ZÁVĚR 43

LITERATURA 44

A PŘÍLOHA I

Seznam obrázků

Obrázek 2.1: Příklad robota sestaveného ze stavebnice Lego Mindstorms 2
Obrázek 2.2: Blokové schéma NXT inteligentní jednotky 3
Obrázek 2.3: NXT inteligentní řídicí jednotka 4
Obrázek 2.4: Průřez NXT servomotorem 5
Obrázek 2.5: Pulzně šířková modulace střídy 20:80, dvě periody 5
Obrázek 2.6: Světelný senzor 6
Obrázek 2.7: Zvukový senzor 6
Obrázek 2.8: Dotykový senzor 7
Obrázek 2.9: Ultrazvukový senzor 7
Obrázek 2.10: Ukázka programu NXT-G 8
Obrázek 3.1: Struktura vrstev technologie Bluetooth 12
Obrázek 4.1: Edice jazyka Java 14
Obrázek 4.2: Životní cyklus MIDletu 17
Obrázek 5.1: Povrch hřiště pro robotický fotbal 19
Obrázek 5.2: Konstrukční provedení pohonu 20
Obrázek 5.3: Odpalovací mechanizmus před a po odpálení míčku 21
Obrázek 5.4: Konečná podoba robota: pohled zleva a zprava 23
Obrázek 6.1: Životní cyklus aplikace 25
Obrázek 6.2: Ukázka grafického rozhraní aplikace 28
Obrázek 7.1: Vývojový diagram sledování čáry 31
Obrázek 7.2: Rozdělení stavů při otáčení 33
Obrázek 7.3: Vývojový diagram otáčení robota pro jeden směr 34
Obrázek 7.4 Intenzita světla při otáčení robota 39
Obrázek 7.5: Odezva robota při regulaci 41

Seznam tabulek

TABULKA 3.1: PŘEHLED TŘÍD BLUETOOTH 11
TABULKA 7.1: PŘIŘAZENÍ KÓDU TLAČÍTKŮM KLÁVESNICE MOBILU A JEJICH FUNKCE 36
TABULKA 7.2: KONSTANTY PROGRAMU POTŘEBNÉ NADEFINOVAT 37
TABULKA 7.3: PRAVIDLA NASTAVENÍ REGULÁTORU 40

1 Úvod do bakalářské práce

1.1 Cíl práce

Tato práce je zaměřená na využití stavebnice Lego Mindstorms pro realizaci soutěžní úlohy robotický fotbal. Robotický fotbal bude sloužit pro demonstraci možností této stavebnice na katedře řídicí techniky, fakulty elektrotechnické, českého učení technického v Praze. Pravidla hry jsou obdobná jako u klasického stolního fotbalu. Tým se bude skládat ze tří robotů. Robot smí jezdit pouze po vyznačené černé linii a nesmí se dotýkat okolních robotů. Každý robot bude ovládán samostatně jedním hráčem pomocí bezdrátové technologie Bluetooth z mobilního telefonu podporující Java micro edition (Java ME). Cílem této práce je tedy navrhnout a sestavit funkčního robota, který bude konstrukčně schopný pohybovat se po černé linii a odpalovat míček, dále pak naimplementovat software pro robota a navrhnout aplikaci v jazyku Java ME pro mobilní telefony, která naváže spojení s robotem a následně ho bude ovládat.

1.2 Obsah práce

Celá práce se skládá z devíti kapitol. První a poslední kapitola se věnuje úvodu a zhodnocení. Druhá kapitola popisuje stavebnici Lego Mindstorms, možnosti funkcí a programování stavebnice a jednotlivě její součásti. Kapitola také popisuje možnosti komunikace s robotem. Třetí kapitola popisuje technologii Bluetooth. Čtvrtá kapitola obsahuje vlastnosti Java ME aplikací a použití komunikace Bluetooth. Pátá kapitola se věnuje Návrhu soutěžní úlohy, konstrukci a sestavení robota. Šestá kapitola popisuje návrh a implementaci aplikace Java ME pro ovládání robota. Sedmá kapitola obsahuje popis navrženého programu pro robota, způsob jak ho naprogramovat a nastavit. A nakonec osmá kapitola je o webových stránkách.

2 Stavebnice Lego Mindstorms

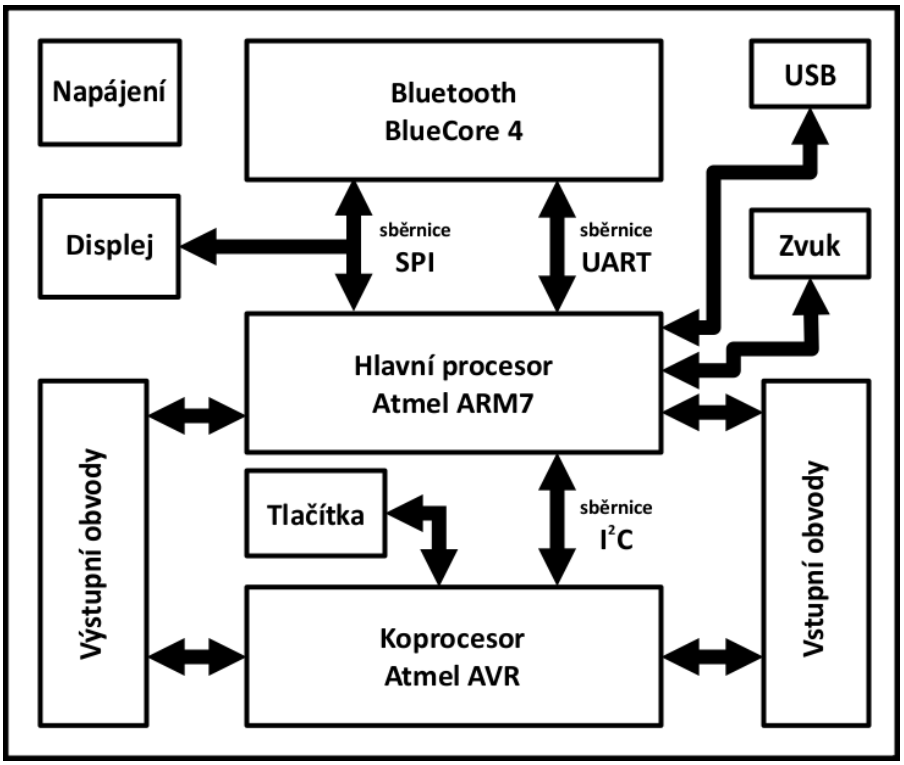
2.1 Představení stavebnice

Stavebnice Lego Mindstorms je výuková stavebnice od známé firmy Lego. První verze této stavebnice byla na trh přivedena již v roce 1998 pod názvem RIS, Robotic Invention System. První programovatelná kostka RCX byla vytvořena v laboratoři MIT. Originální stavebnice obsahovala RCX inteligentní kostku, dva motory, dva dotekové senzory a jeden světelný senzor. Nynější verze nese název Lego Mindstorms NXT 2.0 a obsahuje novou modernější kostku NXT, tři servomotory, dva dotykové senzory a po jednom kusu světelný senzor, ultrazvukový senzor a zvukový senzor. Stavebnice má konstrukční prvky známé ze stavebnic Lego Technics.

Obrázek 2.1: Příklad robota sestaveného ze stavebnice Lego Mindstorms

2.2 Inteligentní řídicí jednotka NXT

Nejdůležitějším prvkem stavebnice, který nemůže chybět v realizaci žádné úlohy, je inteligentní řídicí jednotka, často nazývána jako kostka. Bolkové schéma řídicí jednotky můžeme vidět na obrázku 2.2. Jednotka se skládá z hlavního procesoru, jímž je 32 bitový Atmel AT91SAM7S256 pracující na frekvenci 48MHz. Procesor je vybavený pamětí FLASH o velikosti 256Kb, a pamětí typu RAM veliké 64Kb. Procesor komunikuje s podpůrným 8 bitovým koprocesorem Atmel Atmega 48 pomocí sběrnice I2C. Do flash paměti procesoru jsou nahrávány vytvořené programy buď pomocí rozhraní USB 2.0 nebo pomocí Bluetooth.



Obrázek 2.2: Blokové schéma NXT inteligentní jednotky

Jednotka nabízí Bluetooth komunikace díky integrovanému čipu CSR BlueCore 4 v2.0. Čip má v sobě integrovaný 16 bitový procesor a všechen potřebný hardware pro samostatnou funkci Bluetooth. Čip je připojený k hlavnímu procesoru přes dva typy sběrnice. Jednou z nich je SPI, kterou však využívá pouze na případný update firmwaru BlueCore čipu. Druhá sběrnice pracuje na rozhraní UART a slouží ke komunikaci s hlavním procesorem. Komunikace probíhá rychlostí 460Kbit/s. Jednotka komunikuje s ostatními Bluetooth zařízeními pomocí SPP serial port profile, takže se při komunikaci vytvoří virtuální sériový port, po kterém

zařízení plně duplexně komunikují. Jednotka se může také pomocí Bluetooth spojit až se 3mi NXT kostkami přičemž jedna musí být nastavena jako „Master“ a zbylé tři jako „Slave“. Kvůli sériovému typu spojení musí „Master“ jednotka přepínat mezi jednotlivým připojením.

Kostku lze ovládat čtyřmi tlačítky. Zobrazovací systém kostky je Černobílý maticový displej 100 x 64 pixelů o velikosti 40,6 x 26 mm. Jednotka má čtyři šestivodičové očíslované vstupy, které mají digitální i analogové rozhraní, pomocí kterých připojíme periferní senzory. Dále má tři taktéž šestipinové výstupní porty, značené podle abecedy, kterými kostka ovládá servomotory. Výstupní porty obsahují i vstupy, díky kterým jednotka přijímá zpětnovazební údaje např. o úhlovém natočení servomotorů. Jako konektory jsou v obou případech použity šesti žilové RJ12. Jednotka obsahuje také nízkotónový 21mm reproduktor do kterého je zesilován signál čipem od výrobce SUNPLUS. Vzhled NXT inteligentní kostky můžeme vidět na obrázku 2.3. [8] [9]

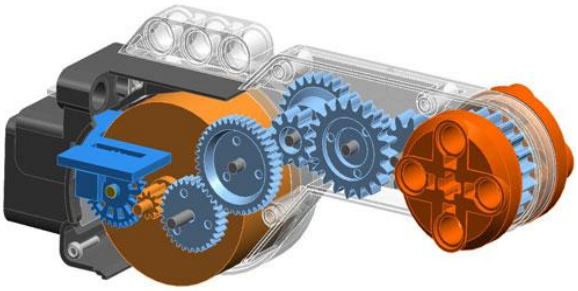
Obrázek 2.3: NXT inteligentní řídicí jednotka

2.3 Akční členy stavebnice

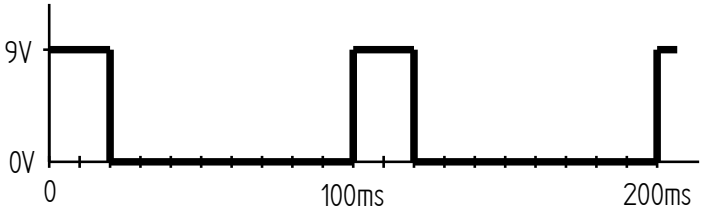
2.3.1 Servomotor

Spolu s řídící jednotkou je dalším nepostradatelným prvkem servomotor. Řadí se do skupiny akčních členů stavebnice. Napájení stejnosměrného motorku uvnitř je 0-9V ten se dokáže točit až 1500 otáčkami za minutu. Díky veliké rychlosti a otáčení a naproti tomu malému točivému momentu je nutné motor převodovat. V Servomotoru se nachází převodovka, která redukuje rychlost na maximálních 160 otáček za minutu. Průřez motoru můžeme vidět na obrázku 2.4. Servomotor obsahuje také rotační inkrementální senzor otáčení, který je schopen

detekovat úhlové natočení ±1º. Díky tomuto senzoru je možné přesně měřit počet otáček, anebo servomotor natočit přesně na požadovaný úhel. Rychlost otáčení motoru je regulována pulzně šířkovou modulací PWM, která je nejpoužívanějším regulačním signálem v takovýchto aplikacích. Perioda PWM je 100ms a rozsah 0-9V. Pokud tedy pošleme do motoru PWM (obr. 2.5) střídy 20:80 motor se bude točit, pokud není zatížen a baterie je v nabitém stavu 20% maximální rychlosti.



Obrázek 2.4: Průřez NXT servomotorem



Obrázek 2.5: Pulzně šířková modulace střídy 20:80, dvě periody

2.4 Senzory

2.4.1 Světelný senzor

Tento světelný senzor (Obrázek 2.6) použijeme, pokud budeme navrhovat robota, který nějakým způsobem musí zpracovávat intenzitu světla okolí nebo například potřebuje detekovat rozdíly mezi výrazně kontrastními barvami. Senzor obsahuje fototranzistor a červenou LED light emitting diodu. Díky těmto komponentám může pracovat ve dvou módech, jako detektor okolní intenzity světla, v tomto případě může být LED zhasnutá. Pokud LED rozsvítíme, můžeme fototranzistorem detekovat intenzitu odraženého světla, tímto dosáhneme možnost detekce mezi kontrastními barvami. Tento druhý způsob se používá zejména

v aplikacích robota, který má za úkol sledovat černou čáru. Signál, který tento senzor přivádí do NXT řídící jednotky je analogový a uvnitř jednotky je převeden na digitální za pomoci 10 bitového A/D Analogově digitálního převodníku.

Obrázek 2.6: Světelný senzor

2.4.2 Zvukový senzor

Zvukový senzor (Obrázek 2.7) měří hladinu akustického tlaku – zvuk. Tento senzor využijeme, pokud má robot za úkol reagovat na určitou hladinu zvuku. Senzor měří akustický tlak v decibelech dB. Rozsah senzoru je do 90dB. Signál ze senzoru je opět přiveden na A/D převodník.

Obrázek 2.7: Zvukový senzor

2.4.3 Dotykový senzor

Další z řad senzorů stavebnice je dotykový senzor (Obrázek 2.8). Použijeme ho, pokud robot potřebuje detekovat přítomnost nějakého předmětu, nebo jako nárazník, který detekuje, když robot narazí na překážku. Tento senzor pracuje asi na nejjednodušším principu ze všech, obsahuje totiž pouze tlačítko, které posílá do řídící jednotky pouze dva stavy. Senzor také

obsahuje ochranný sériový odpor pro případ, že detektor připojíme na výstupní port, aby nedošlo ke zkratu.

Obrázek 2.8: Dotykový senzor

2.4.4 Ultrazvukový senzor

Ultrazvukový senzor (Obrázek 2.9) je jediným detektorem, který obsahuje mikroprocesor. S řídící jednotkou komunikuje pomocí I2C sběrnice. Tento detektor umožňuje robotu „vidět“ okolní předměty a zaměřovat vzdálenosti do 255 *cm* s přesností 3 *cm*. Pracuje na principu echolokace, vyšle ultrazvukový signál o frekvenci 40 *kHz* a měří časový interval, než se odražený signál vrátí zpět, který detekuje speciálním mikrofonem. Z naměřeného zpoždění a známé rychlosti šíření zvuku ve vzduchu senzor stanoví vzdálenost od předmětu. Pracuje výborně při měření vzdálenosti od velké rovné překážky, jako je například stěna. Pokud se v blízkosti vyzařované dráhy ultrazvuku objevuje nějaký rušivý malý předmět, měření je pak nepřesné nebo úplně zkreslené.

Obrázek 2.9: Ultrazvukový senzor

2.4.5 Propojovací kabely

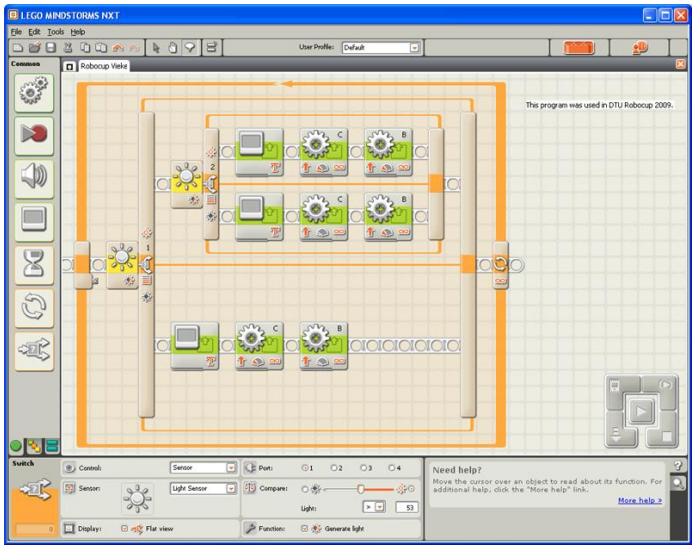
Na propojení senzorů a akčních členů s řídicí jednotkou jsou k dispozici šesti žilové propojovací kabely na koncích osazené konektory RJ12.

2.5 Programování NXT řídicí jednotky robota

Možností jak naprogramovat NXT řídicí jednotku je poměrně mnoho. Mezi nejoblíbenější programovací jazyky patří NXT-G, NXC, lejos NXJ možností je i toolbox vyvinutý pro program Matlab.

2.5.1 NXT-G

Programovací jazyk NXT-G je grafický a vychází z programu LabVIEW společnosti National Instruments. Právě tato firma ve spojení se společností LEGO vyvinula programovací prostředek NXT-G. V programu jednoduše poskládáte funkční bloky a nastavíte potřebné parametry. Tento způsob programování nevyžaduje znalost žádného programovacího jazyku a je intuitivní. Výhodou je, že velice rychle řeší nerozsáhlé projekty. Naopak nevýhodou je implementace složitějších a rozsáhlejších projektů. Tento produkt je dodáván spolu se stavebnicí Lego Mindstorms. Ukázka vzhledu programu je na obrázku 2.10.



Obrázek 2.10: Ukázka programu NXT-G

2.5.2 NXC

Je programovací jazyk odvozený ze známého C. Jazyk se píše ve speciálním freewarovém programu BricxCC. Program napsaný v NXC funguje na stejném firmware jako NXT-G. Jedná se o jazyk velice podobný jazyku C, a proto se tento prostředek těší oblibě u lidí, kteří jsou zvyklí na jazyk C. NXC je programovací jazyk bez grafických prvků, je textový.

2.5.3 LeJOS NXJ

Zřejmě nejoblíbenější způsob programování řídicí jednotky. Je to produkt šířený zdarma společností Sourceforge. Jedná se o programování velmi rozšířeným objektově orientovaným jazykem Java. Pokud chcete programovat v tomto jazyce je nutné do robota nahrát speciální firmware obsahující Java virtual machine, tento firmware můžete kdykoliv smazat a vrátit změnu v standartní firmware Lego Mindstorms. Pro snadné programování LeJOS NXJ obsahuje knihovnu tříd, kterou implementuje API. Další velikou výhodou je možnost zabudování do známých IDE jako je Netbeans nebo Eclipse. API LeJOS NXJ obsahuje velké množství naimplementovaných funkcí, včetně dokumentace pro snadné programování robota. LeJOS NXJ také obsahuje další obslužné programy např. pro čtení dat z flash paměti robota a jiné užitečné nástroje. [4]

2.5.4 Matlab toolbox

Jednou z dalších poměrně úspěšných možností programování robota, je pomocí programu Matlab. Universita RWTH Aachen vyvinula pro Matlab toolbox, kterým je možné ovládat Robota v reálném čase jak pomocí bezdrátového Bluetooth nebo vysokorychlostního USB 2.0. Tento toolbox je velice výhodné použít při modelování systému robota, na které slouží v Matlabu program Simulink. Díky tomuto toolboxu můžete naprogramovat robota vytvořením matematického modelu. [17]

3 Technologie Bluetooth

3.1 Důvod vzniku technologie Bluetooth

Technologie Bluetooth je krátkodosahové radiové spojení, které vzniklo za požadavku levného a malého zařízení schopného bezdrátově komunikovat. Dalším kritériem vývojářů bylo dosáhnout co nejnižší spotřeby. Díky těmto vlastnostem technologie nalezne uplatnění ve velice širokém okruhu jako je komunikace mezi zařízeními stejného druhu i mezi zařízeními různých druhů. Možnost komunikace s periferiemi typu klávesnice, myš, handsfree a mnoho dalších. Využití nalézá i v oblasti průmyslu jako bezdrátový přenos dat z různých měřících zařízení, jako dálková ovládání. Díky své nízké ceně je Bluetooth velice dostupná a v dnešní době hodně rozšířená technologie.

3.2 Historie Bluetooth

Bluetooth technologie byla vynalezena v roce 1994 inženýry švédské společnosti Ericsoon. V roce 1998 se k projektu připojují další společnosti Intel, IBM, Nokia, a Toshiba a zakládají společnost SIG Special Interest Group. První verze specifikací byla zveřejněna v červenci 1999 a zařízení používající technologií Bluetooth byly přivedeny na trh v roce 2000. S první verzí 1.0 a 1.0b nastalo několik problémů, a proto v roce 2002 byla publikována další verze specifikací 1.1, která chyby předešlé verze odstranila. Další verze 1.2, která byla kompatibilní s předešlou verzí 1.1, nabídla vyšší přenosovou rychlost a to až 721 *Kbit/s* dále upravuje způsob modulace, čímž dosahuje mnohem větší odolnosti vůči rušení, zvyšuje optimalizaci pro hlasové přenosy a další protokolové úpravy. V roce 2007 byla vydána verze 2.0 + EDR. Nová verze rapidně zvýšila přenosovou rychlost EDR (Enhanced Data Rate) až na 3 *Mbity/s* . Nově také zlepšuje párování zařízení a zvyšuje bezpečnost. Další verzí 3.0 + HS (High Speed) byl přidán protokol 802.11, na kterém funguje WiFi a tím byla rychlost zvýšena na 24 *Mbit/s*. Současná verze 4.0 ještě snížila spotřebu a rozměry čipu tak, že bude schopný fungovat až 4 roky napájený standardním monočlánkem do hodinek. Tímto SIG otevřela možnosti zabudovat technologii do dalších zařízení v oblasti zdravotnictví, sportu a fitness zařízení.

3.3 Vlastnosti technologie

Technologie komunikuje v bezlicenčním pásmu 2,4 GHz, konkrétně v rozsahu 2,400 GHz - 2,4835 GHz. Vzdálenost kanálů je 1 MHz, a proto je počet kanálů omezen na 79. Za účelem zvýšení odolnosti proti rušení je pro vysílání použita metoda frekvenčních skoků (FHSS) s rychlostí 1600 skoků za sekundu. Od verze 2.0 EDR používá modulaci PSK (Phase Shift Keying) a to DQPSK pro rychlosti do 2Mb/s a 8DPSK pro rychlosti do 3Mb/s. Bluetooth může vysílat ve třech výkonnostních třídách, které jsou u jednotlivých zařízení voleny s ohledem na spotřebu. Z toho plyne omezení na dosah komunikace. Přehled tříd a jejich vlastností najdete v tabulce 3.1. Mobilní telefony patří do třídy Class 2.

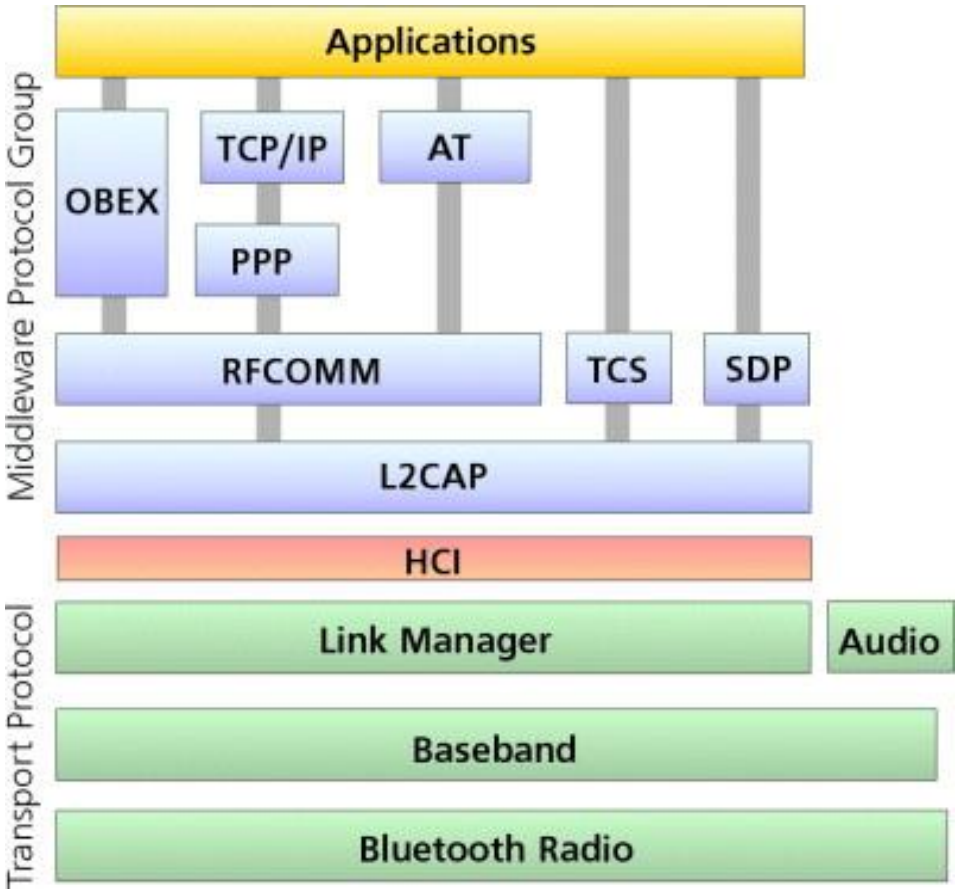
Výkonnostní třída	Vyzařovaný výkon	Maximální dosah
Class 1	100 mW	100 m
Class 2	2.5 mW	10 m
Class 3	1 mW	1 m

Tabulka 3.1: Přehled tříd Bluetooth

3.4 Struktura protokolů

Jako u počítačových sítí je referenční model ISO/OSI má i Bluetooth definované vrstvy a jejich hierarchii tzv. Bluetooth protocol stack (Obrázek 3.1). Nejnižší vrstva Bluetooth radio je obdobou fyzické vrstvy modelu ISO/OSI. Ve vrstvě je definováno uspořádání kanálů a kmitočtové spektrum. Další vrstvou je Baseband, zde se definují formáty paketů, fyzické a logické kanály a režimy pro přenos hlasu a dat. Na další úrovni se nachází vrstva Audio umožňující přenos hlasu. Druhou vrstvou na této úrovni je Protokol správy spojení Link Manager, zabývá se ověřováním spojení, správou buněk a vytvářením popřípadě rušením spojení a jeho konfiguraci. Mezi spodními vrstvami definující transportní protokol a vyššími vrstvami se nachází HCI Host Controller interface, tedy hostitelské řídicí rozhraní. HCI poskytuje standardizované rozhraní mezi hostitelským zařízením a k němu připojeným modulem. První vrstvou z oblasti vrstev vyšších je L2CAP (Link Layer Control and Adaption Protocol) ta provádí multiplexování jednotlivých protokolů vyšších vrstev. RFCOMM je první protokol, který využívá služeb vrstvy L2CAP. Tento protokol emuluje standardní sériové rozhraní a

poskytuje slučitelnost s aplikacemi, které jsou na tomto rozhraní založeny. Protokol TCS (Telephony Control Protocol) je používán pro výměnu signalizace, sestavení a uvolnění hlasových přenosů. Další protokol využívající služby L2CAP je SDP (Service Discovery Protocol) ten zajišťuje aplikaci informace o dostupnosti a konfiguraci služeb na připojeném zařízení. V poslední vrstvě před úrovní aplikace se nachází veliké množství převzatých protokolů např. OBEX používaný pro přenos souborů a dat. TCP/IP pro přístup do sítí a Internetu. PPP protokol pro propojení telefonických sítí a pro ovládání dalších zařízení je zde definován soubor AT příkazů. Mimo jiné je zde taky protokol WAP umožňující datové přenosy po mobilních sítích. [6]



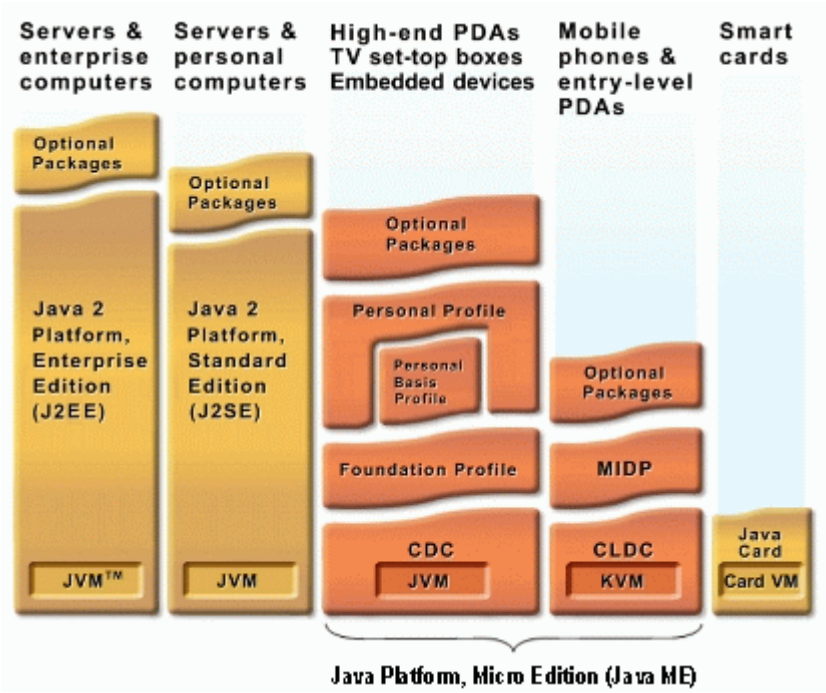
Obrázek 3.1: Struktura vrstev technologie Bluetooth

3.5 Profily Bluetooth

Aby bylo možné připojit k jednomu zařízení jiné, které používá technologii Bluetooth, musí podporovat některé Bluetooth profily. Profily definují možnosti aplikace a popisují, jak je nutné technologii Bluetooth použít pro konkrétní zařízení. Zabraňují možné nekompatibilitě zařízení od různých výrobců. Každý profil musí obsahovat specifikaci závislosti na jiných profilech, doporučený formát uživatelského rozhraní a používané vrstvy protocol stacku. Základním profilem, ze kterého vycházejí všechny ostatní je GAP (Generic Acces Profile) tento profil zejména definuje použití dvou nejnižších vrstev protocol stacku. Mezi hlavní úkoly tohoto profilu patří zavedení definic a metod, jež používají ostatní profily, popis chování zařízení pro objevování a zajištění komunikace. Všechny ostatní profily mohou využít funkce GAP a to také usnadňuje vývojářům definovat nové profily. Další profil SDAP (Service Discovery Aplication Profile) je používán k lokalizaci služeb, které zařízení v dosahu podporují, určuje způsob použití vrstvy SDP, uvedené v 3.4. Další používaný profil je SPP (Serial Port Profile) ten emuluje sériové rozhraní RS-232 pro komunikaci mezi Bluetooth rozhraní, je založený na RFCOMM protokolu.

4 Java ME

Java micro edition zkráceně Jave ME známá také jako J2ME je jedna ze čtyř platform jazyka java. Pokud zanedbáme verzi Java Card, tak je Java ME nejméně obsáhlá platforma viz Obrázek 4.1. J2SE je edice zejména pro osobní počítače. J2EE je platforma používaná na programování serverů a průmyslových počítačových stanic. Jak již název edice Jave ME napovídá, jedná se o odlehčenou verzi jazyka. Tato verze klade důraz na možnost programování menších, výkonově omezených zařízení jako jsou mobilní telefony nebo jednoúčelové embedded systémy. Takové zařízení jsou limitovány různými faktory, jako je nedostatečný výpočetní výkon, nízká zobrazovací schopnost, limitovaná paměť. Aplikace vytvořené v jazyce Java ME jsou tato zařízení schopna provozovat. Absenci balíčků, které Java ME nepodporuje, nahrazuje množství jiných, které jsou navrženy a optimalizovány přesně pro použití s konkrétním zařízením. [15]



Obrázek 4.1: Edice jazyka Java

4.1 Konfigurace Java ME

Java ME byla rozdělena na dvě konfigurace CDC a CLDC podle možností zařízení, na nichž Java ME má fungovat. Konfigurace je navržena tak, aby interpretovala množinu zařízení spadajících do této kategorie.

4.1.1 CDC

Connected Device Configuration je konfigurace zaměřena na nejvýkonnější mobilní zařízení. Používají ji i jiná zařízení jako set-top boxy, navigace. Jejich minimální dostupná ROM paměť je 512 KB a RAM paměť 256 KB. Využívá JVM (Java Virtual Machine) se standardními vlastnostmi, jaké udává JVM SE. Některé konkrétní implementace můžou mít zjednodušené chování kvůli úspoře hardwarových prostředků, ale stále budou odpovídat předepsaným vlastnostem.

4.1.2 CLDC

Connected Limited Device Configuration je konfigurace určena především pro mobilní telefony či PDA s malou pamětí a nízkou výpočetní schopností. Jak lze vidět na Obrázek 4.1 u této konfigurace není potřeba JVM ale mnohem méně složitý virtuální stroj KVM (Kilobyte Virtual Machine), což vede k daleko nižším nárokům na hardware, konkrétně vyžaduje 160 kB paměti ROM a 32 kB RAM. Protože CLDC používá redukovaný virtuální stroj, který neobsahuje plnohodnotný verifikátor byte kódu jako JVM, musí být kód preverifikován při kompilaci. CLDC verze 1.1 nabízí následující balíky:

```
java.io
java.lang
java.util
javax.microedition.io
```

4.2 Profily Java ME

Profily definují rozhraní pro programování aplikací tzv. API (Application Programming Interface). Nejznámější profil konfigurace CLDC je MIDP (Mobile Information Device Profile). Protože jsem aplikaci pro ovládání robota vytvořil na základě konfigurace CLDC a profilu MIDP omezím se při dalším výkladu pouze na profil MIDP.

4.2.1 MIDP

Profil rozšiřuje konfiguraci CLDC o některé další užitečné funkce. Skupina vývojářů JSP (Java Community Process) definuje každou specifikaci jménem JSP (Java Specification Request) a unikátním číslem. První verze MIDP 1.0 je JSR-37. MIDP 1.0 obohacuje CLDC konfiguraci o úložiště dat v paměti zařízení, podporu pro řízení životního cyklu aplikace, uživatelské rozhraní a možnosti připojení k síti, to interpretují následující balíčky:

```
javax.microedition.rms
javax.microedition.midlet
javax.microedition.io
javax.microedition.lcdui
```

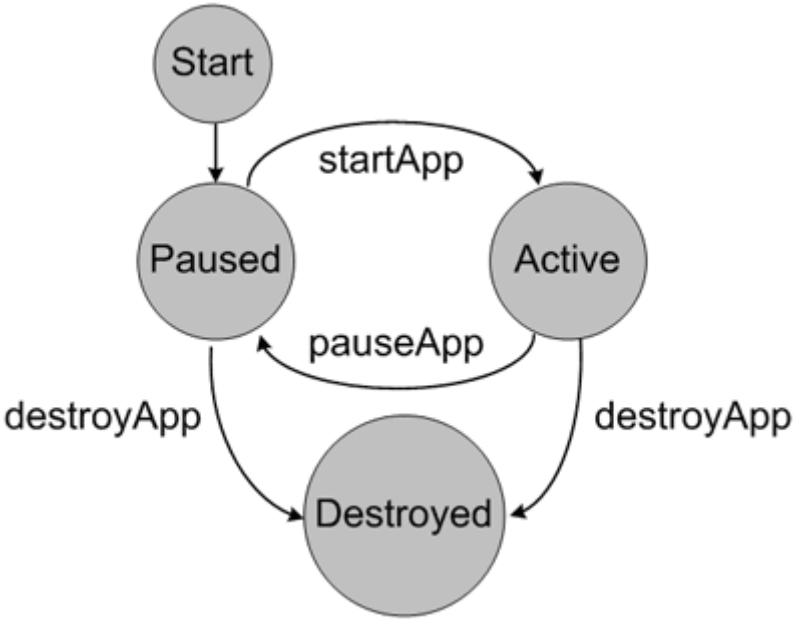
Profil s sebou nese i jistá omezení pro hardwarovou konfiguraci. Zařízení musí mít display alespoň 96 x 54 pixelů, 32 kB paměti pro běh Javy, 8 kB stálé paměti pro data aplikace, 128kB stálé paměti pro MIDP, vstup klávesnice nebo dotykový display a přítomnost obousměrné bezdrátové síťe. Druhá verze JSR-118 tedy MIDP 2.0 přidává další funkcionality a zvyšuje pouze požadavky na velikost paměti. Přehrávání multimedií, API pro tvorbu 2D grafických her, zabezpečení připojení a práci s desetinnými čísly. Přidává tedy nové zdrojové balíky:

```
javax.microedition.media
javax.microedition.lcdui.game
javax.microedition.pki
```

Aplikace vybudovaná na profilu MIDP se nazývá MIDlet, vstupní bod této aplikace je třída

```
javax.microedition.midlet.MIDlet
```

tato třída definuje abstraktní metody, které jsou volány v průběhu životního cyklu samotné aplikace (Obrázek 4.2). Taková aplikace je řízena AMS (Amplication Management Software) vestavěného do zařízení. Aplikace reaguje na pozastavení např. příchozím hovorem, ten je na obrázku 4.2 znázorněn jako stav „Paused“, následně musí být schopna obnovit činnost do stavu „Active“.



Obrázek 4.2: Životní cyklus MIDletu

Pro přenositelnost mezi různými zařízeními musí MIDlet splňovat určité požadavky. Aplikace musí být zabalena do souboru s příponou *jar*, hlavní třída musí být potomkem třídy *javax.microedition.midlet.MIDlet*, soubor *.jar* musí být preverifikován z důvodu uvedeném v 4.1.2. V některých případech musí být i soubor podepsán mobilním operátorem. Návod jak vytvořit „Hello word“ MIDlet aplikaci je na internetu velké množství např. na webu [7] je k dispozici seriál o programování Jave ME MIDP aplikací od úplného začátku až po pokročilé úlohy. [5]

4.3 Java ME a Bluetooth

Pro práci s technologií bluetooth na zařízeních podporujících java ME je definovaná specifikace JSR-82, ta zjednodušuje specifikaci složitého zásobníku bluetooth množinou aplikačních rozhraní. JSR-82 definuje dvě základní API pro přístup k technologii bluetooth, a to OBEX (Object exchange) API a základní bluetooth API tomu odpovídají tyto dvě třídy:

```
javax.bluetooth
javax.obex
```

Aplikační rozhraní od JSR-82 požaduje po bluuetooth zásobníku zařízení podporu profilu GAP, SDP a SPP o těchto profilech bylo již psáno v kapitole 3.5. Základní kroky při použití aplikačního rozhraní definujícího obsluhu Bluetooth:

- inicializace zásobníku bluetooth
- správa zařízení
- vyhledání zařízení
- hledání služeb na dostupných zařízeních
- navázání komunikace se zařízením
- použítí funkcí pro komunikaci na určeném protokolu
- uzavření spojení

Prakticky při zavedení aplikačního rozhraní Bluetooth API třída implementuje abstraktní metody, které reagují na události z výše zmíněného seznamu. Metody jsou poté přepisovány a následně použity např. k vytvoření seznamu nalezených zařízení. Pokud chceme použít komunikaci technologií bluetooth v java ME aplikaci na konkrétním zařízení, musíme nejdřív zjistit, zda naše zařízení podporuje API z JSR-82, což není u mnoha zařízení samozřejmostí. [1] [5]

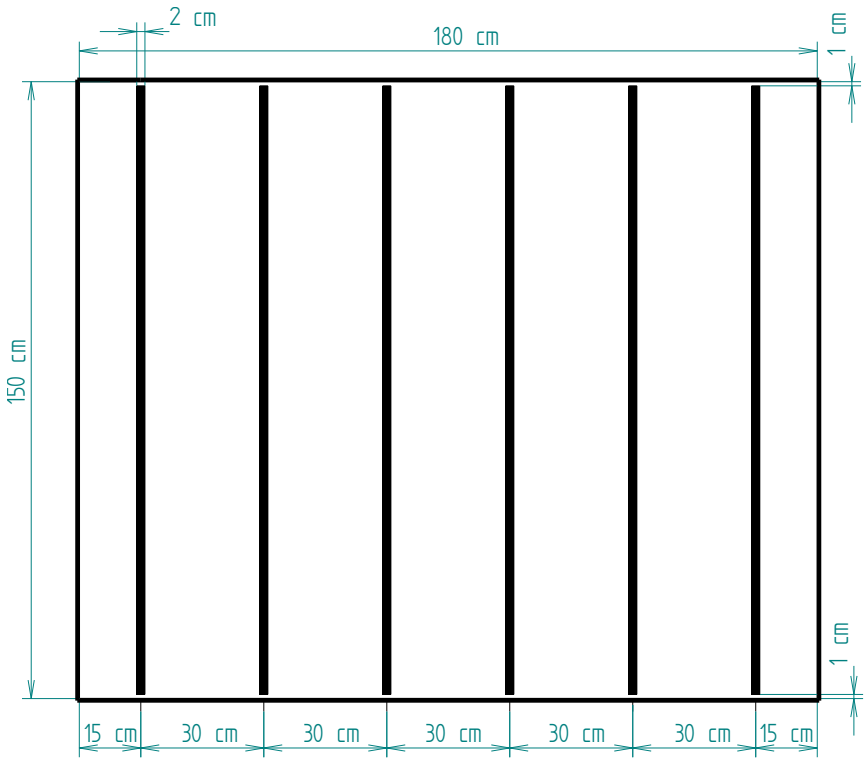
Jednoduchý příklad, jak navrhnout aplikaci schopnou vyhledat zařízení a navázat komunikaci lze nalézt na webu [11].

5 Návrh soutěžní úlohy robotický fotbal

Kapitola se věnuje popisu pravidlům soutěžní úlohy, stanovení požadavků pro mechanickou konstrukci robota a popis jednotlivých kroků při stavbě robota.

5.1 Pravidla a hřiště

Úkolem bylo navrhnout robotický fotbal, který bude robotickou obdobou klasického stolního fotbalu. V soutěži proti sobě budou soupeřit dva týmy. Tým se má skládat ze třech nezávislých robotů a každý uživatel bude ovládat mobilním telefonem jednoho robota. Soutěž bude probíhat na hřišti o velikosti 1,8 *m* na 1,5 *m* ohraničené mantinely výšky 25 *cm*. Míček bude použit standartní ze stavebnice Lego Mindstorms. Tento míček má průměr 5 *cm*. Na povrchu hřiště (Obrázek 5.1) jsou rovnoměrně rozmístěny černé čáry, po kterých se může robot pohybovat. Robot má k dispozici tedy pruh šířky 30 *cm*, který nesmí nijak přesáhnout, aby nedošlo ke kontaktu s jiným robotem. Toto omezení je nutné respektovat při konstrukci robota. Robot musí být schopen manipulovat s míčkem v každém místě jeho prostoru.



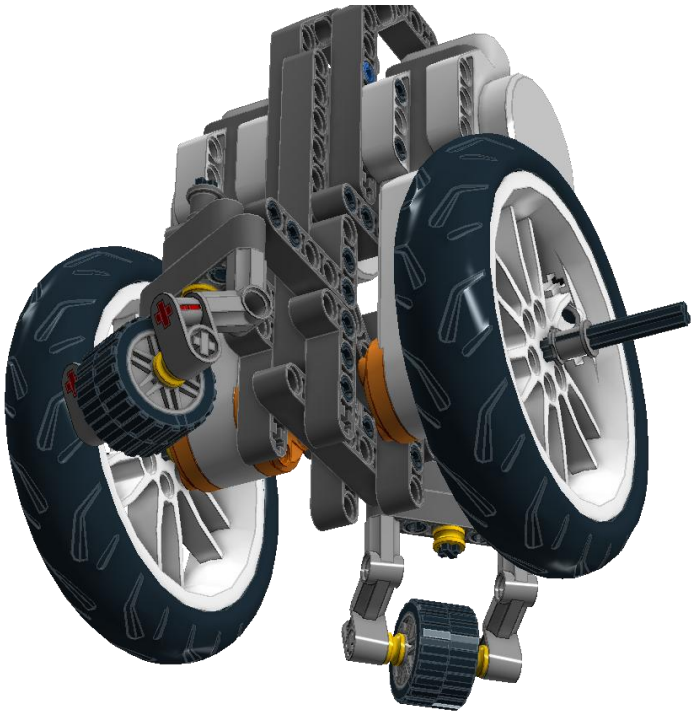
Obrázek 5.1: Povrch hřiště pro robotický fotbal

5.2 Konstrukce robota

Robot musí splňovat konstrukci vhodnou pro soutěžní úlohu. Z výše zadaných pravidel vyplývá, že robot má sledovat černou linii. Dále musí být schopen manipulovat s míčkem na svém prostoru, tedy musí být otočný. Důležitým parametrem konstrukce je ale především schopnost odpálit míček. Na základě těchto požadavků bude tedy robot navrhnut.

5.2.1 Pohon robota

Pohon může být řešen různými způsoby. Například společný servomotor pro rovný směr pohybu a další akční člen pro změnu pohybu a natočení. Vhodný způsob řešení se však osvědčil případ dva nezávislé servomotory pro přímý směr, čímž snadno dosáhneme změny směru, pokud budou mít pohonné jednotky rozlišnou rychlost otáčení. Jak se i později ukázalo, tak tento způsob je výhodný pro sledování čáry. Praktická realizace (Obrázek 5.2) byla dva servomotory k sobě připevněné. Motory byly osazeny velkými koly a ke konstrukci byly přidány



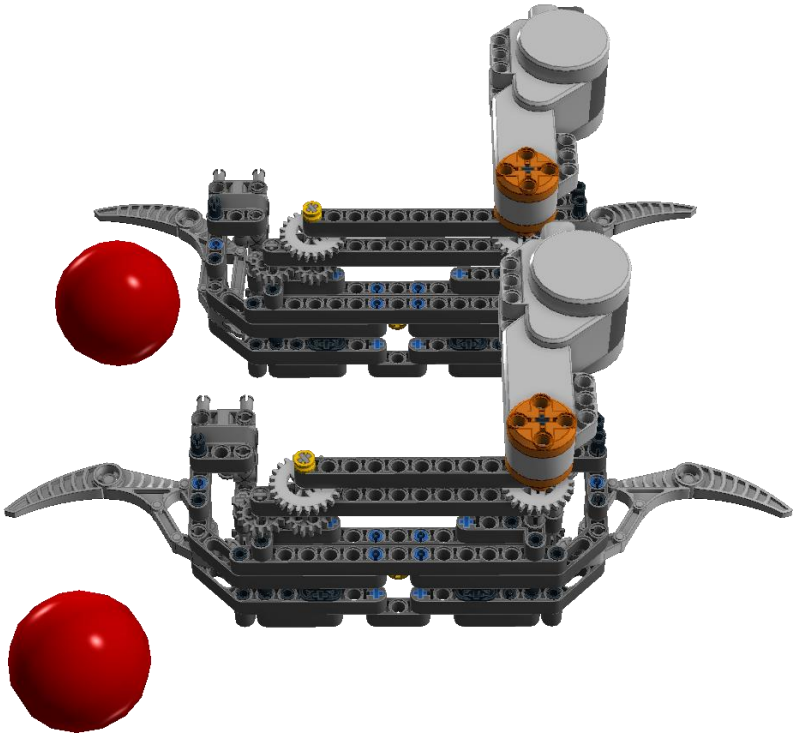
Obrázek 5.2: Konstrukční provedení pohonu

dvě malé točny s podpůrnými otočnými kolečky, točny se natáčejí podle směru robota samovolně. Díky tomuto mechanizmu se robot dokáže otáčet okolo své osy. Mírnou nevýhodou je doba ustálení točen, při které má robot snahu měnit směr.

5.2.2 Odpalovací mechanizmus

Dalším problémem konstrukce bylo vytvoření funkčního odpalovacího mechanizmu. Tento problém byl navíc omezen mechanickými prostředky, protože dva výstupy byly již použity pro pohon. Schopnost robota odpálit míček by měla být co nejlepší, proto není vhodné, aby měl odpalování jen na jedné straně.

Prakticky byl problém vyřešen dvojicí klapek umístěných na obou stranách robota. Klapky mají svůj doraz a jsou nataženy gumou ze stavebnice Lego Mindstorms. Ve výchozí poloze jsou zajištěné a připravené k odpálení. Servomotor klapky současně odjišťuje a následně je natáhne opět do výchozí polohy. Na obrázku 5.3 můžete vidět mechanizmus před a po odpálení míčku.



Obrázek 5.3: Odpalovací mechanizmus před a po odpálení míčku

Servomotor ovládá obě klapky současně pomocí řetězu, který je ve skutečnosti napnutý mezi ozubenými koly umístěnými pod servomotorem. Na straně kde motor není přímo na hřídeli, musí být ještě převodem ozubených kol změněn smysl otáčení.

5.2.3 Konstrukce pro sledování čáry

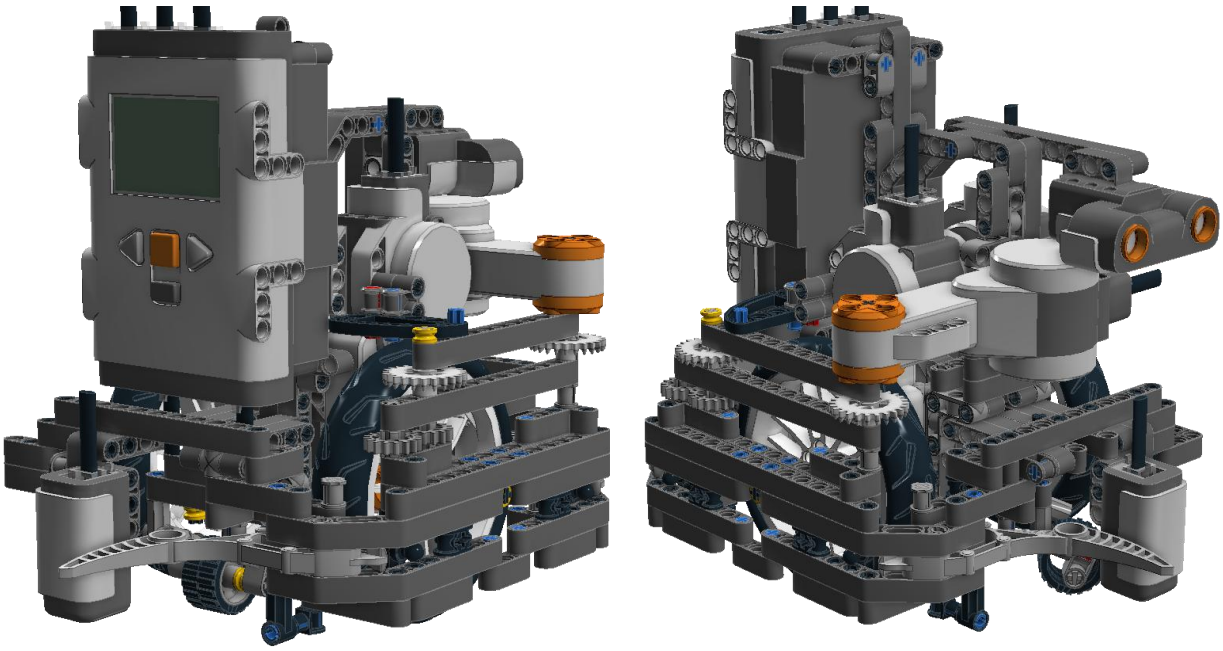
Aby byl robot schopný jezdit po vyznačených liniích, musí řídicí jednotce nějakým způsobem podávat informace o své poloze vůči černé čáře na povrchu hřiště. K tomuto účelu poslouží světelný senzor. Tento senzor je schopný distribuovat jednotce intenzitu světla, které se odrazilo od hřiště. Tímto senzorem dokážeme vyhodnotit rozdíl mezi černou a bílou. Senzor tedy musí směřovat svým světlem kolmo na povrch hřiště a musí být umístěn co možná nejblíže této ploše. Jak bylo při implementování algoritmu pro robota zjištěno, musí být tyto senzory dva, na obou stranách robota. Více o sledování čáry je uvedeno v kapitole 7.2. Pro správnou funkčnost byl tedy robot vybaven dvěma světelnými senzory, každý na jedné straně.

5.2.4 Omezení pohybu robota

Robot se smí pohybovat po vyznačené linii pouze mezi mantinely, abychom předešli nárazu je vybaven ultrazvukovým senzorem polohy. Tento senzor měří vzdálenost od stěn hřiště a předává tuto informaci řídicí jednotce robota. Jak se později ukázalo tak tento senzor není možné synchronizovat s dalšími na ostatních robotech a dochází k rušení, více v kapitole 7.4.

5.2.5 Sestavení robota

Nakonec byly jednotlivé komponenty sestaveny v celek pomocí vhodných konstrukčních prvků stavebnice. Všechny senzory a akční členy byly propojeny kabely s řídicí jednotkou, která byla umístěna na dostupné místo. Celková stavba odpovídá daným rozměrům a pravidlům hřiště, takže nebude docházet ke střetům s ostatními roboty. Obrázek 5.4 znázorňuje kompletně sestaveného robota. Dokumentace ke stavbě robota byla vytvořena v programu Lego Digital Designer dodávaný společností Lego [14]. Výstupem tohoto programu je podrobným postup pro sestavení robota. Tento návod je možné nalézt v příloze.



Obrázek 5.4: Konečná podoba robota: pohled zleva a zprava

6 Aplikace pro ovládání robota

Tato kapitola se věnuje popisu návrhu mobilní aplikace pro ovládání robota. Aplikace má být implementována v konfiguraci schopné fungovat na cenově méně nákladných mobilních telefonech, což její malá složitost dovoluje. Tyto přístroje podporují aplikace napsané v jazyce Java ME. Jak bylo popsáno v kapitole 4, tato odlehčená platforma Java ME nabízí pro mobilní telefony dvě konfigurace a profily. Aplikace bude vytvořena v konfiguraci CLDC a za použití nejpoužívanějšího profilu MIDP.

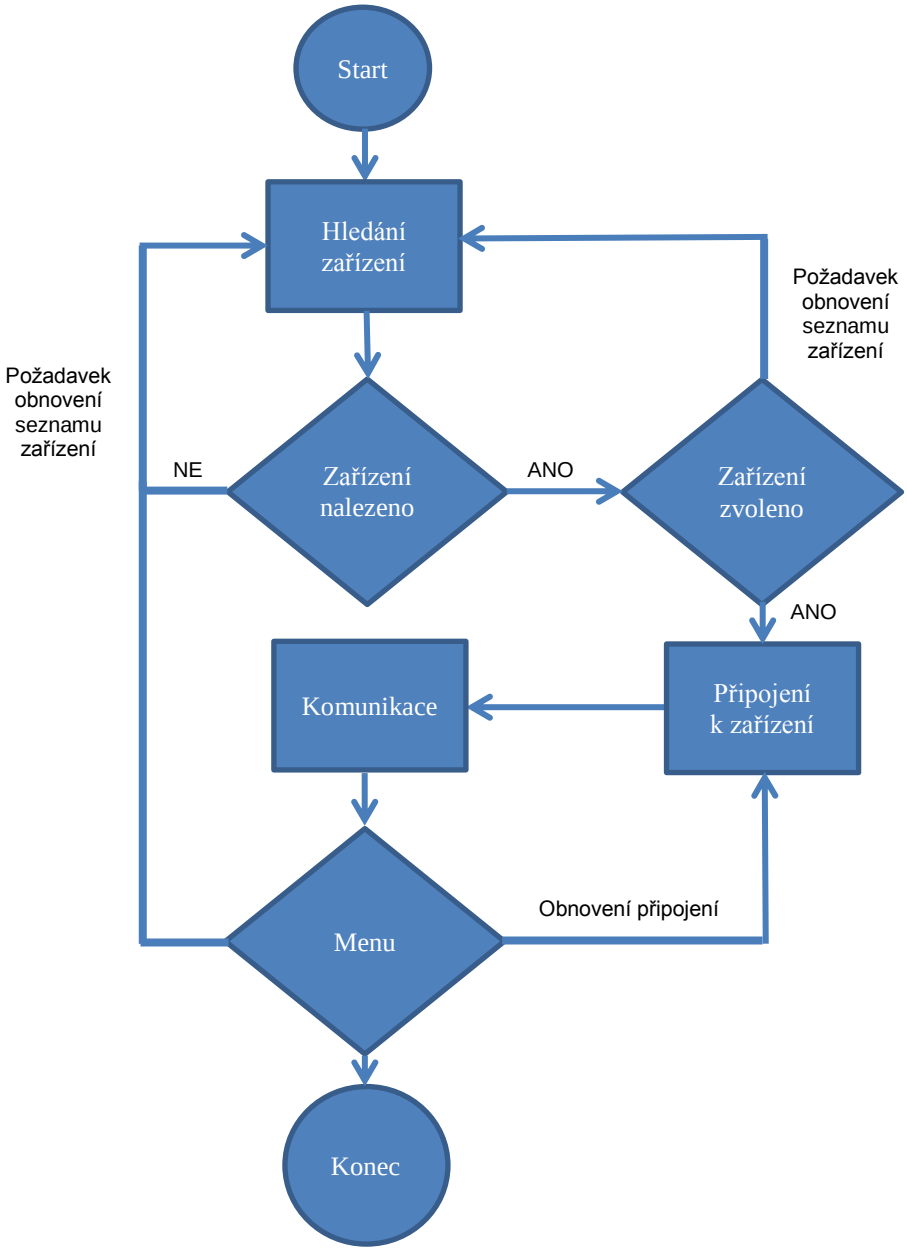
6.1 Příprava vývojového prostředí pro návrh aplikace

Jako vývojové prostředí pro navržení, vybudovaní a odladění aplikace byl použit IDE (Integrated Development Environment) NetBeans, ten je možné získat z webových stránek *http://netbeans.org/downloads*. Pro nainstalovaní tohoto prostředí je ještě nutné mít nainstalovaný JDK (Java Development Kit), ten je možné získat ze stejných webových stránek. Pokud máme nainstalovaný program Netbeans bez přídavného zásuvného modulu pro Java ME je ještě nutné tento plug-in přidat pomocí Plugin Manageru. V nabídce „*Tools*“ vybereme „*Plugins*“ kde zvolíme sekci „*Available Plugins*“, zobrazí se nám seznam dostupných zásuvných modulů. Vybereme položku „*Mobility*“ z kategorie „*Java ME*“ a zvolíme „*Install*“ pak potvrdíme a počkáme, než se nám modul stáhne z internetu a následně nainstalujeme. Pro překlad a testovaní aplikace musíme mít ještě nainstalovaný Sun Java Wireless Toolkit for CLDC, který lze stáhnout z webových stránek společnosti Oracle *http://www.oracle.com/* v sekci *Downloads*. Tento program se pak automaticky zavolá při spuštění vytvořené Java ME aplikace v IDE a umožnuje vyzkoušet aplikaci na emulátoru mobilního telefonu.

Pokud jsme podle těchto kroků nainstalovali do IDE zásuvný modul máme možnost vytvářet aplikace pro mobilní telefony Java ME kompatibilní. Při vytvoření nového projektu máme teď rozšířenou nabídku kategorií aplikací o Java ME. Pokud vybereme tuto kategorii, můžeme vytvořit nový projekt „*Mobile Aplication*“. Teď už je vše připraveno pro samotný návrh aplikace. Po vytvoření a spuštění aplikace se nám v adresáři projektu ve složce „*/dist/*“ vytvoří soubor s koncovkou *jar*, který pak snadno přeneseme např. přes Bluetooth do mobilního zařízení. [15]

6.2 Návrh aplikace „Robot Control“

Tato aplikace ponese název *„Robot Control“*, bude využita k ovládání Robota sestaveného ze stavebnice Lego Mindstorms, pomocí technologie Bluetooth. Aplikace používá API pro práci se zařízeními technologie Bluetooth ze specifikace JSR-82 viz kapitola 4.3. Díky této API je schopna vyhledat zařízení, služby a následně se připojit na zařízení poskytující službu SPP. Tato služba je jediná, kterou Bluecore čip v řídicí jednotce robota podporuje. Aplikace po vytvoření spojení reaguje na stisk a uvolnění tlačítka odesláním příslušného kódu jednotce robota. Na následujícím obrázku 6.1 je znázorněn životní cyklus aplikace. [10]



Obrázek 6.1: Životní cyklus aplikace

Strukturu tvoří tři třídy *Midlet.java*, *Bluetooth.java* a *Control.java*. Aplikace také obsahuje jeden obrázek *keys.png*, pro názornější ukázku rozložení kláves, při ovládání robota. Tento obrázek se zobrazí ihned po propojení se zařízením.

6.2.1 Třída Midlet

Tato třída je hlavní součástí celé aplikace – Midletu. Dědí vlastností ze třídy *MIDlet* a implementuje rozhraní *CommandListener*. Třídu *MIDlet* musí dědit každá aplikace vytvořená v tomto profilu. Zděděné metody, které představují životní cyklus MIDletu (Obrázek 4.2) musí každá aplikace přepisovat. Metoda *startApp()* je použita pro popisek na úvodní obrazovce po startu aplikace. Popisek informuje, k čemu aplikace slouží. V metodě *pauseApp()* není nutné nic vykonávat a metoda *destroyApp()* se zavolá v případě, že uživatel požádá o ukončení aplikace. Metoda *commandAction* je z rohraní *CommandListener*, a v jejím těle se reaguje na příkaz vyvolaný stiskem tlačítka, zavoláním příslušné metody. Tato metoda má parametry *commnand* a *displayable*. První parametr nám říká, o který příkaz se jedná a druhý, v jakém zobrazení byl příkaz zavolán. Další metoda byla vytvořena pro vyhledání zařízení *startInquiry()*, je zavolána pokud uživatel spustí aplikaci a zvolí tlačítko *„Search“* (Obrázek 6.2), nebo pokud nebylo nalezeno potřebné zařízení, může uživatel tlačítkem *„Refresh“* požádat o obnovení seznamu. Tato metoda využívá služeb další třídy *Bluetooth.java* a zavolá její metodu pro nalezení zařízení v nově spuštěném vlákně, aby nedošlo k „zamrznutí“ aplikace. Pokud proběhne hledání bez chyb, zavolá další metodu *selectList()*, která zveřejní seznam nalezených zařízení na display mobilu. Pokud uživatel zvolí zařízení tlačítkem *„Connect“* metoda *commandAction* zareaguje zavoláním další metody *OpenConnection()*, kde je v nově spuštěném vlákně opět využita metoda třídy *Bluetooth.java* pro navázání spojení a následně zavolána pro zobrazení třída *Control.java*.

6.2.2 Třída Bluetooth

Jak je z názvu třídy patrné je určena pro práci s technologií bluetooth. Implementuje rozhraní *DiscoveryListener* z knihovny *javax.Bluetooth*. Z tohoto rozhraní vychází nutnost přepsat některé abstraktní metody využívané pro hledání dostupných zařízení. První

metodou, která však není definována rozhraním je *searchDevices(),* ta je zavolána ze třídy *Midlet.java* při požadavku o nalezení zařízení. Pro nalezení zařízení je incializován *DiscoveryAgent* požadavkem *startInquiry*, poté vlákno čeká na odblokování přepsanou abstraktní metodou *inquiryCompleted().* Dále je *DiscoveryAgent* zavolán pro *searchServices* s parametry nalezeného zařízení a profilu, ten je v tomto případě SPP. Vlákno je znovu zablokováno a čeká na odblokování další přepsanou metodou *serviceSearchCompleted*. V době kdy je vlákno zablokované metoda rozhraní *servicesDiscovered* uloží jednotlivá jména zařízení a jejich URL (Uniform Resource Locator) potřebné pro navázání spojení se zařízeními. Pokud je celý přenos dokončen informace o nalezených přístrojích jsou předány třídě *Midlet.java* pro zobrazení. Pokud je zavolána metoda *connect()* je navázáno spojení pomocí URL vybraného zařízení. Třída dále obsahuje metodu *disconnect()* pro uzavření připojení a *send()* pro převedení kódu klávesy přijaté od třídy *Control.java,* na pole typu *byte* a přidána informace o délce a následně zaslána otevřeným výstupním streamem.

6.2.3 Třída Control

Poslední třída je zavolána po připojení k zařízení a slouží k reagování na stisk tlačítka a dědí od třídy *Canvas*. Zděděná metoda *keyPressed()* s parametrem kódu tlačítka definovaného třídou *Canvas* reaguje na stisknutí tlačítka a zavolá v nově vytvořeném vlákně metodu *send()* z třídy Bluetooth.java a předá jí kód tlačítka. Tyto kódy jsou typu *Integer*. Tabulku kódů naleznete v kapitole 7.5. Další metoda *keyReleased()* je stejného typu ale reaguje na uvolnění tlačítka zasláním speciálního kódu stejným postupem jako metoda *keyPressed().* Jedinou výjimku tvoří tlačítko „5“, na které metoda nereaguje z důvodu využití tohoto tlačítka pro speciální účel, který je popsán v kapitole 7.5.1. Třída také metodou *paint()* zobrazí na celý display obrázek keys.png, který znázorňuje rozložení kláves pro ovládání.

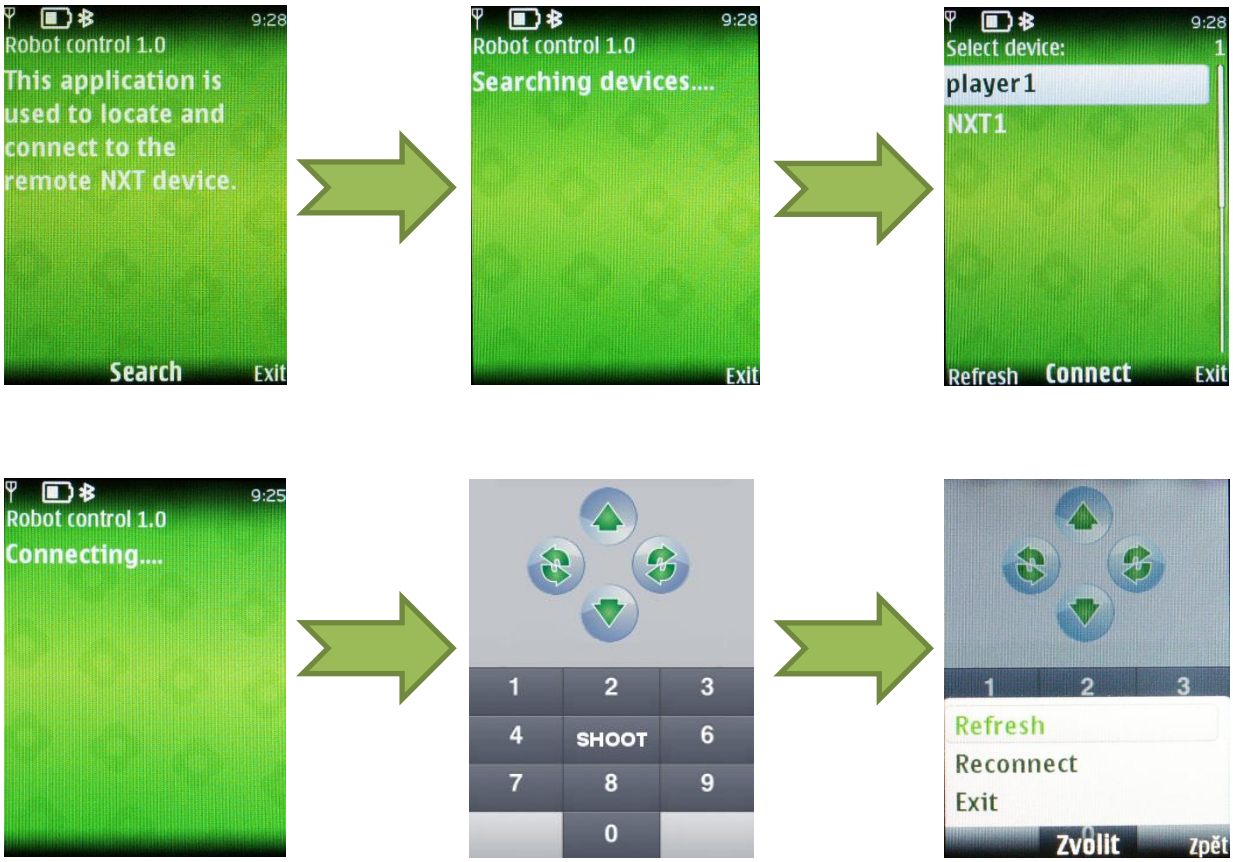
6.3 Využití aplikace

Tato aplikace, ač se zdá být speciálně vytvořená pro jeden účel, může nalézt uplatnění ve všech aplikacích ovládání Robota Lego Mindstorms. Její výhodou je veliká jednoduchost, která je obvykle při těchto případech žádoucí. Další výhodou je snadný způsob

navázání komunikace na straně robota a následná jednoduchost přijetí kódů vyslaného mobilním zařízením.

6.4 Závěr

Podařilo se vytvořit funkční aplikaci postavenou na konfiguraci CLDC a profilu MIDP. Aplikaci lze spustit na mobilním telefonu s technologií Bluetooth a podporujícím Java micro edition. Mobilní zařízení musí taky podporovat použití specifikace JSR-82 obsahující API pro správu Bluetooth zařízení. Pokud zařízení podporuje Java ME a má zabudováno Bluetooth, nemusí vždy podporovat i JSR-82. Pro zjištění podpory specifikace budeme muset navštívit oficiální stránky výrobce zařízení a podívat se do detailních specifikací konkrétního mobilního telefonu. Celá aplikace byla odladěna na mobilním telefonu Nokia 3720c-2 viz Obrázek 6.2. Funkčnost byla vyzkoušena ještě na třech různých mobilních telefonech od firmy Nokia.



Obrázek 6.2: Ukázka grafického rozhraní aplikace

7 Návrh programu a nastavení robota

Tato kapitola je věnována způsobu naprogramování a implementaci algoritmu pro robota. Podrobně rozebírá problém sledování čáry, otáčení robota, blokování pohybu na koncích hřiště a popisuje možná řešení. Kapitola obsahuje také popis nastavení regulátoru, způsob ladění robota a logování dat na display a do souboru, což je výhodné pro vývoj řízení a diagnostiku robota. Na závěr pak popisuje uvedení robota do provozu.

7.1 Příprava prostředí pro návrh algoritmu

Řídicí jednotku robota je možné, jak bylo uvedeno v kapitole 2, programovat v různých programovacích jazycích. Pro výborné možnosti a snadné vyvíjení byl vybrán programovací jazyk NXJ, neboť umožňuje nainstalovat plugin do IDE Netbeans. Vývoj algoritmu je tak nejsnazší a robot se snadno naprogramuje přímo z prostředí IDE. Pro naprogramování robota stačí mít řídicí jednotku propojenou s počítačem pomocí USB kabelu a v jednotce mít nainstalovaný firmware *leJOS NXJ*. Podrobný návod, jak nainstalovat tento firmware do jednotky, jak zabudovat zásuvný modul do Netbeans a jak programovat v API *leJOS NXJ* je obsahem bakalářské práce Dana Martince [4]. Po nainstalování softwaru *leJOS NXJ* máme k dispozici i užitečné programy pro správu robota. [13]

7.2 Sledování čáry

Problém sledování čáry je jednou z klíčových úloh celého algoritmu. Je nutné, aby robot, jenž má vyhrazený jen určitý prostor, následoval přidělenou čáru na hřišti. Pro sledování používá světelný senzor popsaný v kapitole 2.4.1. Prvním pokusem o vyřešení byl umístěn jeden světelný senzor na ose otáčení robota, tedy mezi hnacími koly. Tento případ byl obtížnější pro regulaci, neboť reakce světelného senzoru je mírně zpožděna a není tak citlivý na malé změny směru pohybu robota. Takové řešení vyžaduje poměrně veliký vzorkovací kmitočet. Dalším důvodem pro nalezení jiného řešení byla nespolehlivost použití tachometru otáčení kol pro vrácení při natočení robota zpět na směr pohybu po čáře. Další návrh, který vyřešil oba problémy byl použití dvou světelných senzorů po stranách robota viz Obrázek 5.4. Tedy robot má nyní

světelný senzor vždy před sebou ve směru pohybu, je přepínáno mezi senzory v závislosti na směru jízdy. Díky této konstrukci má světelný senzor zvětšenou citlivost na změnu pohybu robota a je snazší jej regulovat. Senzor podává informace o intenzitě odraženého světla. Tato intenzita je odlišná při odrazu z černé a bílé barvy, díky tomu je robot schopný detekovat, když se nachází na hraně mezi černou a bílou. Hodnota odražená na tomto rozhraní je pro nás referenční hodnota regulace. Regulovat budeme pohon levé a pravé strany robota pomocí PWM.

7.2.1 Výběr regulátoru a jeho popis

Aby výsledný efekt sledování čáry byl co nejlepší, je nutné zvolit vhodný regulátor. Abychom zjistili, o jaký systém se zhruba jedná, vyšleme do pohonu jednotkový skok. V praxi to pro robota znamená, že nastavíme funkcí *setPower()*, která je v rozsahu 0 až 100 na jedné straně PWM o jednotku menší např. 50 a druhé straně o jednotku větší, tedy 60.

```
Motor.B.setPower(50);
Motor.C.setPower(60);
Motor.C.forward();
Motor.B.forward();
```

Po zapnutí vidíme, že robot začne odjíždět od čáry. Z toho lze jednoznačně posoudit, že se jedná o astatický systém, tedy systém má jeden pól umístěný v nule. Tato důležitá vlastnost nám dovoluje pro dobrý výsledek použít regulátor bez integrační složky. Máme tedy na výběr, lze použít regulátor proporcionální, nebo proporcionálně derivační. Regulátor proporcionální bychom použili, kdyby nám nevadilo mírné kmitání okolo čáry, ale pokud chceme, aby se robot pohyboval bez kmitů, využijeme regulátor proporcionálně derivační zkráceně PD. Rovnice PD regulátoru:

$$u(t) = K_p e(t) + K_d \dot{e}(t)$$

Kde $u(t)$ je akční zásah, který je nutné vykonat, K_p je proporcionální konstanta, K_d je derivační konstanta a $e(t)$ je regulační odchylka, kterou lze vyjádřit jako rozdíl mezi výstupní hodnotou $y(t)$ a hodnotou referenční.

$$e(t) = y(t) - r(t)$$

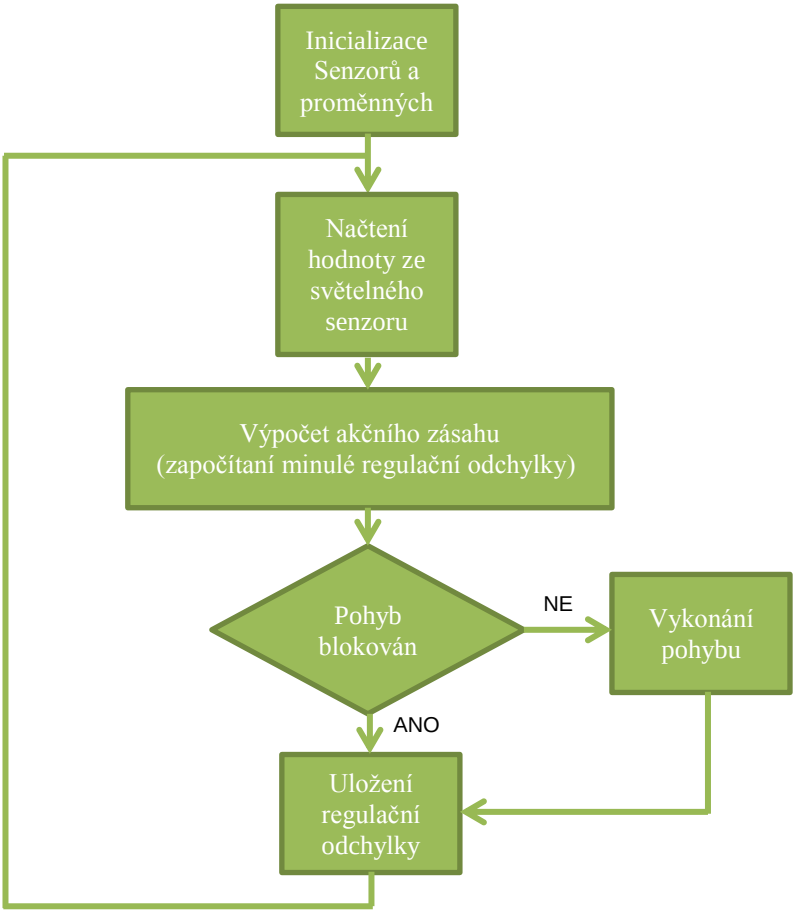
Derivaci regulační odchylky $\dot{e}(t)$ pak lze jednoduše vyjádřit jako rozdíl mezi odchylkou aktuální a odchylkou předchozí. Tímto aproximujeme derivační složku, neboť doba mezi vzorky není v reálném případě nekonečně krátká.

$$\dot{e}(t) = e(t) - e_{-1}(t)$$

O nastíněné kroky a celkově o zajištění funkce sledování čáry slouží naimplementovaná třída *LineFollower.java*. [16]

7.2.2 Třída LineFollower

LineFollover.java implementuje rozhraní *Runnable*. Metoda *run()* je přepsána cyklem, který nikdy nekončí a v jeho těle jsou vykonávány jednotlivé kroky regulace. Na obrázku 7.1 je znázorněn zjednodušený vývojový diagram tohoto cyklu.



Obrázek 7.1: Vývojový diagram sledování čáry

Ve skutečnosti se v cyklu ještě testuje směr pohybu a následně je načtena hodnota ze správného světelného senzoru a pohyb vykonán požadovaným směrem. Podmínka pro zablokování pohybu je splněna, dokud je robot mimo čáru, tedy otáčí se. Tuto proměnnou nastavuje třída *RobotRotating.java*. Dále je zde metoda pro zastavení pohybu robota uživatelem *LineFollowingStop()*. Další metody jsou určené pro nastavení popř. přečtení některých konstant regulace. Jedná se o konstanty regulátoru proporcionální, derivační, dále pak referenční hodnotu regulace a rychlost jízdy robota. Nastavení těchto hodnot je uvedeno v kapitole 7.6.

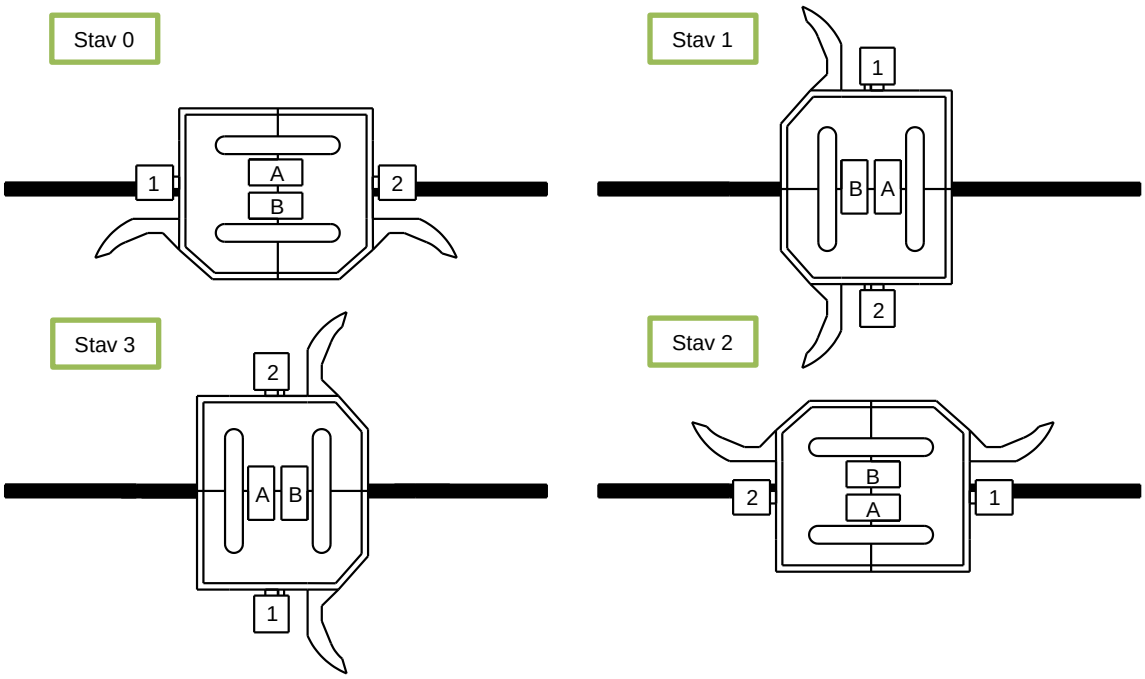
7.3 Otáčení robota

Mezi nejdůležitější vlastnosti robota patří schopnost se otáčet kolem své osy. Samotná jízda z jedné strany hřiště na druhé nestačí. Robot musí být schopen zasáhnout míček v celém prostoru pro něho vyhrazeném, aby byla zachována plynulost hry. Pokud se robot natočí kolem své osy a uživatel bude chtít pokračovat po čáře na jiné místo, dochází k problému, který je nutné vyřešit. K natočení robota dojde, pokud se budou servomotory na pohon točit opačným směrem. Při pokynu pokračovat po čáře se však robot musí nejprve vrátit do polohy, kde jsou světelné senzory schopny robota zregulovat zpět na čáru. První návrh, jak tento problém vyřešit, bylo použití inkrementálního snímače otáčení, kdy při rotování robota budeme zaznamenávat počet obtočení kol a snadno pak stanovíme potřebné natočení do výchozí polohy. Tento první způsob se v praxi jevil jako velmi nespolehlivý a docházelo k velkým chybám při návratu do výchozí polohy. Další návrh řešení vznikl s vyřešením problému sledování čáry viz 7.2. Dva nové senzory světla na krajích robota (Obrázek 5.4), dovolují přesně detekovat při otáčení robota průchody přes čáru. Tento způsob je naprosto robustní a v praxi se osvědčil. O správnou polohu a funkci natočení robota se stará třída *RobotRotating.java*.

7.3.1 Třída RobotRotating

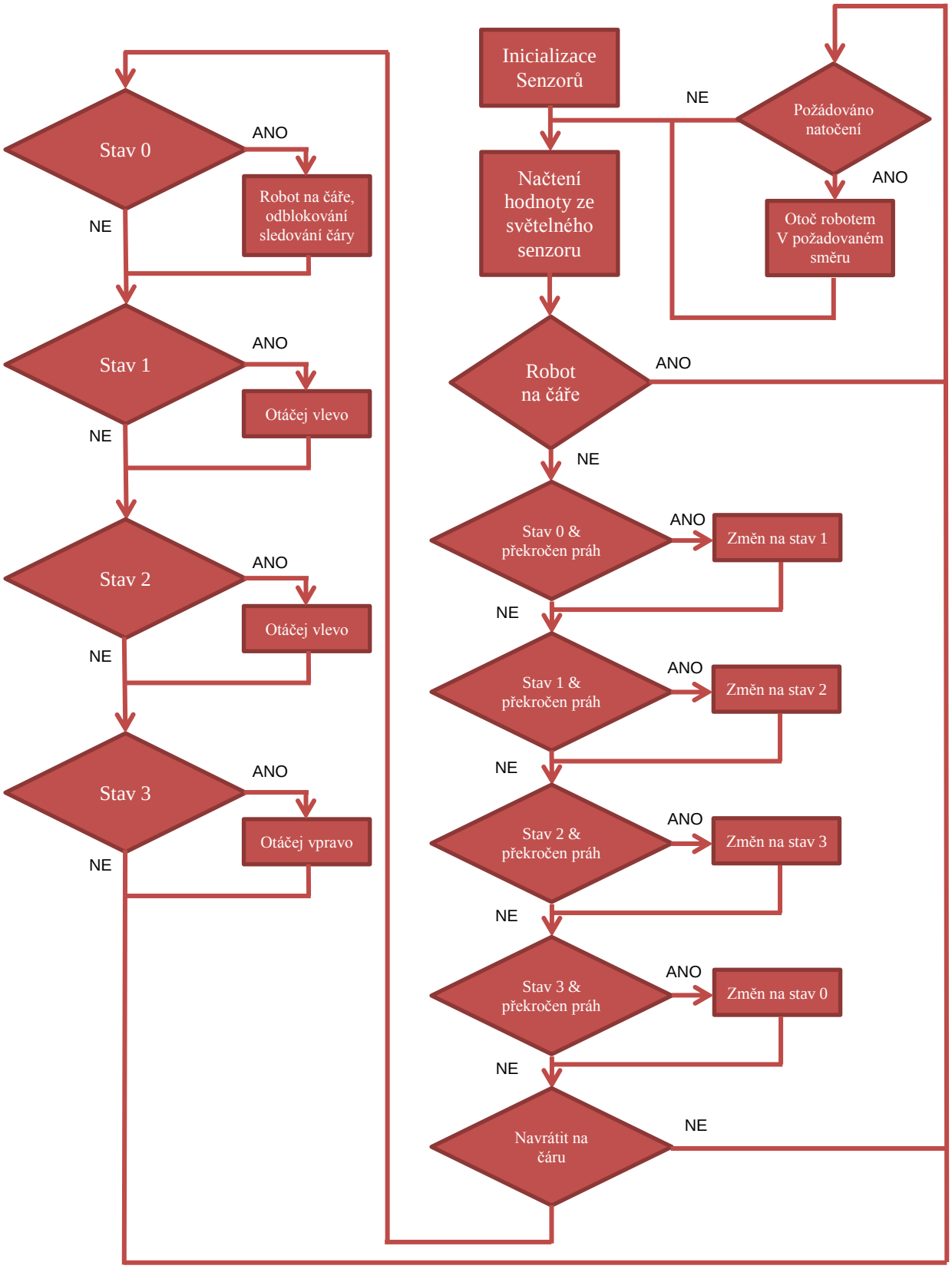
Jako třída *LineFollower.java* i tato implementuje rozhraní *Runnable*. Pro správnou funkčnost musí být cyklus kontrolující a zajišťující natočení robota v běhu paralelně s cyklem pro sledování čáry. Tedy metoda *run()* je přepsána tímto cyklem. Pro detekci, v jaké poloze se robot nachází, jsou využity oba senzory. Byly vytvořeny 4 stavy interpretující rozsah míst, kde se

robot právě nachází, tyto stavy jsou vyobrazeny na obrázku 7.2. Světelné senzory jsou označeny jako 1 a 2. Servomotory pohonu jsou na obrázku označeny jako A a B.



Obrázek 7.2: Rozdělení stavů při otáčení

Obrázek 7.3 znázorňuje zjednodušený vývojový diagram cyklu, pro jeden směr otáčení. Třída obsahuje ještě další metody nutné pro zajištění správné funkce. Metoda *stopRotating()* zastaví v cyklu veškeré testovaní polohy a akční zásahy. *stateOnLine()* je zavolána z cyklu pokud se robot vrátí na „Stav 0“ ta zavolá metodu *stopRotating()* a také odblokuje ve třídě *LineFollower.java* vykonání akčního zásahu, díky čemuž může robot opět sledovat čáru. Další důležitá metoda getFirstState() je zavolána vždy při požadavku otočení robota. Ta nejprve detekuje, jestli byl robot před zavoláním této metody na čáře a pokud ano, pokusí se zjistit, v jakém stavu se nachází po zabrzdění. Bylo testováním zjištěno, že při zastavení robota oba servomotory současně nezareagují, proto metoda zjistí jaký je výchozí stav. Další dvě metody *rotateRight()* a *rotateLeft()* dodrží popsaný postup, zablokují sledování čáry a nastaví v cyklu požadovaný směr otočení. Poslední metody jsou nastavovací a čtecí, kde zejména hraje roli nastavení prahu, při kterém se překlopí na jiný stav.



Obrázek 7.3: Vývojový diagram otáčení robota pro jeden směr

7.4 Blokování pohybu robota

Další užitečnou vlastností robota je zastavení na konci jeho dráhy před boky hřiště. Toto ošetření není až tak úplně důležité. Problém nastal při návrhu jak toto blokování realizovat. Opět první návrh spočíval v kontrolování počtu otáček tachometrem otáčení servomotoru, ale jako v předchozím případě byl tento způsob nespolehlivý až dokonce silně poruchový. Proto byl návrh převeden na užití nového senzoru polohy, a sice na ultrazvukový dálkoměr. Dálkoměr si pomocí principu sonaru určí vzdálenost od mantinelu a program pak vyhodnotí, kdy bude nutné zastavit. Jednoduchost tohoto použití a funkčnost byla jednoznačně lepší. Veliký problém však v tomto případě nastane, pokud bude odměřovat vzdálenost od boků hřiště šest robotů najednou. Synchronizovat jejich vysílání se zdá prakticky nemožné, a proto je tento způsob použitelný pouze v případě jednoho hrajícího robota. Balík programu obsahuje třídu na obsluhu tohoto způsobu lokace *DistanceChecker.java* pro případ použití jednoho robota. Třída běží jako samostatné vlákno a zasahuje do pohybu robota zablokováním akčního zásahu třídě *LineFollower.java*. Tuto třídu však není pro soutěžní úlohu, kde bude hrát šest robotů, možné použít proto jí nebudeme dále rozebírat.

7.5 Hlavní třída programu robota

Hlavní třída, která je spuštěna po zapnutí robota se jmenuje *FootballPlayer.java*. Má za úkol zejména počkat na příchozí připojení od mobilního telefonu a následně reagovat otevřením streamu. Dále musí spustit vlákna potřebných tříd *LineFollower.java* a RobotRotating.java, aby mohl robot fungovat. Pokud se povede otevřít komunikace, převede vstupní stream na *Integer* dle struktury bytů odeslaných z mobilního telefonu. Podle toho, jaký číselný kód přijme, zareaguje příslušnou akcí zavoláním metody některé třídy z balíčku. Číselné kódy odpovídají kódům tlačítek z dokumentace třídy *Canvas* z balíčku *javax.microedition* použité při vývoji aplikace do mobilního telefonu, jsou ale přidány některé speciální navíc. Tyto konstanty jsou uloženy ve třídě *Keys.java*, jednotlivé zkratky konstant a jejich odpovídající číselné kódy typu *Integer* znázorňuje Tabulka 7.1, dále zde je i naznačeno konkrétní využití klávesnice. [10]

Název tlačítka	Odpovídající kód typu Integer	Funkce tlačítka
KEY_NUM_0	*48*	
KEY_NUM_1	*49*	
KEY_NUM_2	*50*	
KEY_NUM_3	*51*	
KEY_NUM_4	*52*	
KEY_NUM_5	*53*	*Odpálení míčku*
KEY_NUM_6	*54*	
KEY_NUM_7	*55*	
KEY_NUM_8	*56*	
KEY_NUM_9	*57*	
KEY_POUND	*35*	
KEY_STAR	*42*	
KEY_LEFT	*-3*	*Otáčení robota doleva*
KEY_RIGHT	*-4*	*Otáčení robota doprava*
KEY_UP	*-1*	*Pohyb po čáře dozadu*
KEY_DOWN	*-2*	*Pohyb po čáře dopředu*
KEY_RELEASED	*99*	*Signalizace uvolnění tlačítka*

Tabulka 7.1: Přiřazení kódu tlačítkům klávesnice mobilu a jejich funkce

7.5.1 Třída FotballPlayer

Tato hlavní třída rovněž jako ostatní implementuje rozhraní *Runnable* a hned po startu je přepsaná metoda *run()* spuštěna, neboť zajišťuje neustálou obsluhu tlačítka „exit“ pro ukončení aplikace. Tento způsob vypnutí je jediný možný, protože hlavní vlákno je pozastaveno čekáním na příchozí bluetooth připojení. Komunikace s mobilním telefonem pomocí Bluetooth je zprostředkováno pomocí proměnné typu *BTConnection* z balíku *lejo.nxt.comm*. Této proměnné nastavíme *waitForConnection()* zde se hlavní smyčka programu zastaví a čeká se na připojení klienta (mobilního telefonu). Po připojení se tato proměnná zavolá s metodou *openDataInputStream()*. V případě, že přijmem streamem data, převedeme je na typ *Integer* a ty odpovídají, jak již bylo zmíněno hodnotám tlačítek (Tabulka 7.1). Pro vykonání příslušné metody na základě kódu tlačítka slouží *switch* uzavřený do cyklu, který běží stále, dokud jsou přijímána data. Pokud například ukončíme aplikaci v mobilním telefonu, tento cyklus se zastaví a

dojde k vypnutí také Robota. Pokud je přijat kód, vykonává se patřičná akce, dokud není přijat kód odpovídající uvolnění tlačítka (KEY_RELEASED), tím se obvykle zastaví veškeré akce. Jedinou výjimkou, jak již bylo v kapitole 6.2.3 popsáno, je tlačítko KEY_NUM_5 využívané pro odpalování míčku. Pokud chceme odpálit míček, provede se v nově spuštěném vlákně v metodě *shoot()* otočení servomotoru odpalovacího mechanizmu o 360 stupňů. Tedy mechanizmus se odpálí a znovu nabije. Pokud bychom však i tlačítko „5“ měli opatřené odesíláním informace o uvolnění, došlo by např. při odpálení za jízdy k zastavení robota.

Třídě *FootballPlayer.java* je tedy využívána pro spojení všech obslužných tříd programu robota a stará se o vykonání potřebných funkcí požadovaných uživatelem.

7.6 Uvedení robota do provozu

Aby robot správně fungoval je zapotřebí mu nejprve nastavit veškeré konstanty nezbytné pro jeho provoz (Tabulka 7.2).

Název konstanty	Využití	Třída konstanty
Kp	*Derivační konstanta regulátoru*	*LineFollower.java*
Kd	*Proporcionální konstanta regulátoru*	*LineFollower.java*
offset	*Referenční hodnota regulátoru*	*LineFollower.java*
speed	*Rychlost pohybu robota při sledování čáry*	*RobotRotating.java*
r_speed	*Rychlost otáčení robota*	*RobotRotating.java*
treshold	*Mez intenzity světla kdy překlopení robot do jiného stavu při otáčení*	*RobotRotating.java*

Tabulka 7.2: Konstanty programu potřebné nadefinovat

První konstanty, které je nutné experimentálně určit, jsou referenční hodnota regulátoru a mez intenzity světla, která umožňuje detekovat průchod čarou při otáčení robota. Dalším krokem pak bude nastavení konstant regulátoru, regulace je samozřejmě závislá na rychlosti jízdy robota, proto je nutné najít konstanty pro rozumnou rychlost robota.

7.6.1 Logování dat do souboru

Při určování konstant a ověření, zda jsme vše navrhnuli správně, poslouží třída *DataLog.java*, ta zapisuje potřebná data do souboru. Třída je v případě potřeby zavolána z hlavní smyčky programu a spuštěna jako samostatné vlákno. Z hlavní smyčky nastavíme i jméno souboru kam se mají data ukládat. Po spuštění se otevře výstupní stream do souboru a data se zapisují v nekonečném cyklu v posloupnosti, jak si uživatel nadefinuje. Pro ukončení zápisu slouží pak metoda *closeFile()*. Tato metoda musí být před skončení programu zavolána, jinak se soubor neuloží do paměti NXT inteligentní jednotky.

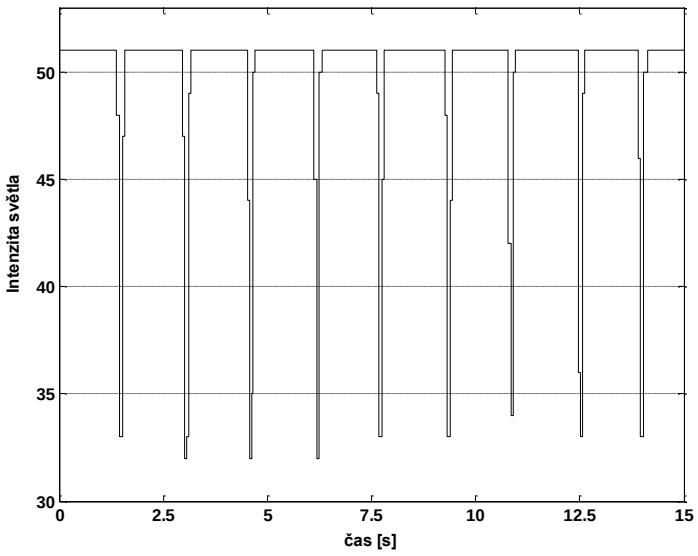
Pro přenesení souboru do PC poslouží program *NXJ Control Center* ze softwaru leJOS NXJ [13]. Tento program nalezneme obvykle ve složce *Program Files\leJOS NXJ\bin\nxjcontrol.exe*. Program spustíme a připojíme NXT jednotku přes USB k počítači. V grafickém rozhraní programu zvolíme *„USB“* a *„LCP“* a klikneme na tlačítko *„Search“*. Načte se nám seznam, kde vybereme naše zařízení a zvolíme tlačítko *„Connect“*. Pro práci se soubory uloženými v NXT jednotce zvolíme záložku *„Files“*. Zde máme seznam souborů v paměti. Zvolíme soubor, do kterého jsme logovali data a stiskneme *„Download file“*, soubor uložíme do PC. Pro zpracování dat pak využijeme buď program Microsoft Excel a nebo Matlab, kam jednoduše soubor importujeme.

7.6.2 Nastavení referenční hodnoty regulátoru

Pro zjištění konstanty potřebujeme vypisovat na display hodnoty ze světelných senzorů a k tomu nám poslouží třída *DisplayWritter.java*, která je opět *Runnable* a po zavolání běží jako samostatné vlákno. Tato třída každých 200ms obnovuje display a zobrazí potřebné informace. Třídu tedy využijeme na zobrazení hodnot přečtených ze světelných senzorů. Z hlavní třídy rozběhneme její vlákno zavoláním *DisplayWritter.start ()*. Nahrajeme program do robota a spustíme. Hned po startu display robota zobrazuje námi nadefinované informace. Přiložíme robota na čáru, tak aby světlo vyzařující světelný senzor dopadalo na rozhraní černé a bílé barvy a odečteme hodnotu z displeje. Tímto krokem jsme získali referenční hodnotu regulátoru, jinými slovy, pokud tuto hodnotu nastavíme regulátoru, jako referenci bude se snažit regulovat pohyb tak, aby robot sledoval přesně toto rozhraní černé čáry a bílého hřiště. Tato hodnota je závislá na okolním osvětlení, ale pokusem bylo zjištěno, že *offset=45* vyhovuje většině prostředí a chyba vzniklá okolním osvětlením je zanedbatelná.

7.6.3 Nastavení prahu překlopení při otáčení robota

Pro určení další konstanty využijeme třídu pro zápis do souboru *DataLog.java.* Nejprve však musíme stanovit, jakou rychlostí se bude robot otáčet. Zkoušením ovladatelnosti bylo zjištěno, že vhodná rychlost bude *r_speed=20.* Experimentálně určíme konstantu *treshoold* tak, že budeme zapisovat do souboru načtené hodnoty ze světelných senzorů při otáčení robota. Abychom určili konstantu, která spolehlivě detekuje průchod čarou, naměříme několik period otočení robota. Získaný průběh můžeme vidět na obrázku 7.4. Světelný senzor nám dává informaci o velikosti odraženého světla, ta je nejvyšší pokud se odráží z bílé barvy, naopak minima jsou při odrazu z černé linie.



Obrázek 7.4 Intenzita světla při otáčení robota

Z průběhu je patrné, že pokud zvolíme konstantu *treshold=35* docílíme spolehlivého detekování průchodů čarou.

7.6.4 Nastavení regulátoru

Nastavení konstant regulátoru je nejčastější problém regulační techniky. Existuje opravdu mnoho metod nastavení. Jedna z velice často používaných metod, ať už pro ruční doladění nebo kompletní nastavení je metoda cyklické optimalizace konstant. Tato metoda je zřejmě nejjednodušší, avšak časově náročná, protože vyžaduje cyklus měnění konstant a testování. Pro nastavení našeho PD regulátoru tuto metodu využijeme.

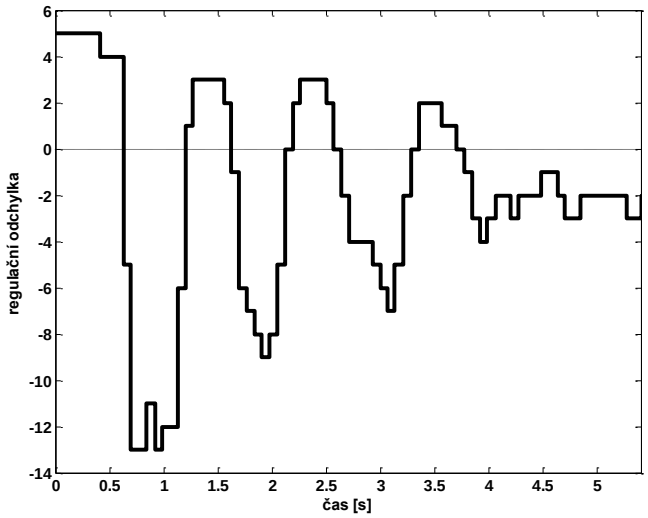
Pro snadné měnění konstant použijeme třídu *ConstantChanger.java*, která se spustí jako samostatné vlákno. Tato třída spolupracuje s *LineFollower.java*. Pokud zavoláme metodu pro nastavení konstanty, např. *setKpchange()* s parametrem zda inkrementovat nebo snižovat, pokusí se měnit konstantu cyklicky po 50 *ms*. Metoda stopCount() potom zastaví cyklus nastavování konstanty. Tedy metodu pro změnu zavoláme po stisku některého z volných tlačítek (Tabulka 7.1) a *stopCount()* potom po přijetí informace o uvolnění tlačítka.

Před samotným nastavováním je nutné nadefinovat rychlost pohybu robota *speed* dle požadavků na dynamiku soutěžní úlohy libovolně. Pro urychlení procesu optimalizace je výhodné řídit se několika empirickými pravidly (Tabulka 7.3).

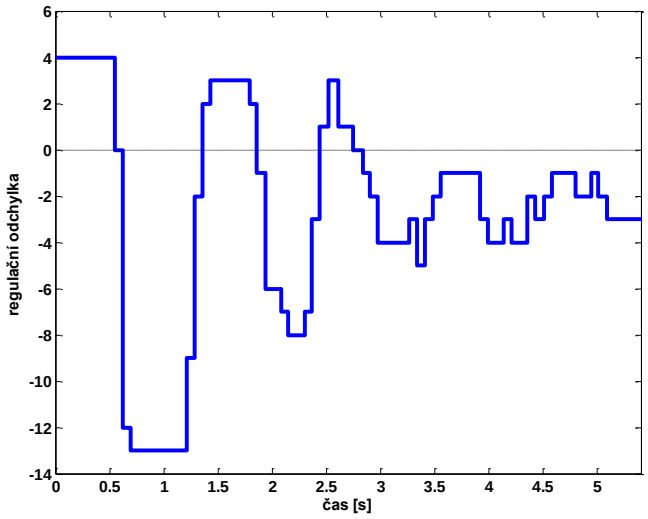
Robot kmitá	↑*Kd* ↓*Kp*
Regulace je pomalá	↑*Kd* ↑*Kp*
Robot se ustaluje až po větším zakmitání	↑*Kd*
Robot se neustaluje - rychle kmitá	↓*Kp*

Tabulka 7.3: Pravidla nastavení regulátoru

Pokud dosáhneme optimalizací uspokojivého výsledku, musíme si uvědomit, že nalezené konstanty platí pouze pro nastavenou rychlost pohybu. Při zvýšení rychlosti je nutné regulátor doladit. Na následujícím Obrázek 7.5 jsou vidět průběhy regulace při startu robota mimo čáru pro dvoje nastavení regulátoru. Tyto průběhy odpovídají rychlosti robota *speed=40.*



a) Kp=7, Kd=45



b) Kp=5 Kd=55

Obrázek 7.5: Odezva robota při regulaci

Kladné hodnoty regulační odchylky odpovídají bílému povrchu. Na rozhraní bílé a černé je podle nastavené referenční hodnoty regulační odchylka rovna nule. Změna odchylky je mnohem větší při poloze robota blíže ke středu černé linie. Z průběhů vidíme, že se robot přesně neustálí na nulové odchylce, ale pokud hodnota kolísá v záporných hodnotách, jedná se ve skutečnosti o velmi malé změny směru pohybu. Proto pokud hodnota klesne pod nulu, můžeme říci, že se robot ustálil a sleduje přesně černou linii. Z naměřených průběhů můžeme tedy stanovit, že rychleji se ustálí regulátor nastavený podle případu na obrázku 7.5b. Doba ustálení je v tomto případě přibližně 2.5 *s*.

7.7 Změna jména robota

Před samotným zahájením hry je nutné, aby každý robot měl svoje unikátní jméno, jinak by docházelo ke kolizi při výběru na mobilním zařízení z dostupných robotů. Pro změnu jméno z implicitního „NXT“ na jiné např. „Player 1“ využijeme program NXJ Control Center, jak tento program zapnout, bylo popsáno v kapitole 7.6.1. Připojíme NXT řídicí jednotku přes USB a zapneme program popsaným způsobem. Vybereme záložku *„Miscellaneous“*. V této záložce máme formulář s názvem *„Change Friendly Name“*, zde napíšeme nové jméno jednotky a potvrdíme tlačítkem *„Set Name“*. V tuto chvíli má jednotka jiné jméno, pokud zařízení odpojíme tlačítkem *„Disconnect“* a poté znovu připojíme, nahlásí se pod novým jménem.

8 Návrh webových stránek

Webové stránky budou sloužit jako prezentace dalšího projektu na již existujícím webu *http://support.dce.felk.cvut.cz/roboti/* předmětu ROBOTI na katedře řídicí techniky. Navrhnuté stránky jsou tedy pouze modulem, který se na tento web přidá do sekce „projekty“. Zdrojové kódy navržených webových stránek jsou uloženy v příloze na CD.

9 Závěr

Soutěžní úloha robotický fotbal byla úspěšně navržena. Jedná se samozřejmě o jednu z mnoha možností jak tuto úlohu řešit. Ze zadání nejprve nebylo úplně jasné, co všechno bude práce obnášet. Postupem času bylo zjištěno, že dostupné aplikace pro mobilní telefony, které by mohly robota ze stavebnice Lego Mindstorms ovládat, nejsou příliš vhodné např. kvůli neschopnosti se připojit k jinému robotu než se jménem „NXT“. Proto byla navrhnuta aplikace, která přesně vyhovuje požadavkům této soutěžní úlohy. Aplikace nejprve zveřejní seznam dostupných zařízení a až po výběru správného začne komunikovat. Poměrně složitým úkolem byl návrh konstrukce robota. Kombinace různých pohybových požadavků bylo nutné vyřešit s dostupnými díly stavebnice. Výsledkem práce je robot schopný hrát podle definovaných pravidel obdobu stolního fotbalu.

Ukázku pohybových možností a metodu odpalování míčku je možné vidět na videu přiloženém na CD nebo na adrese *http://www.youtube.com/watch?v=n8CK_z64zQM*.
Toto video bylo natočeno na provizorním hřišti.

Literatura

[1] Valenta, Tomáš. *Bakalářská práce: Využití Bluetooth přenosu v J2ME.* Praha : Katedra počítačů, České vysoké učení technické, 2006.

[2] Trojánek, Pavel. *Bakalářská práce: Využití robota LEGO MINDSTORMS při výuce.* Praha : Katedra řídicí techniky, České vysoké učení technické, 2009.

[3] Tomáš, Bělík. *Bakalářská práce: Využití robota Lego Mindstorms Návrh a realizace speciálních úloh.* Praha : Katedra řídicí techniky, České vysoké učení technické, 2010.

[4] Martinec, Dan. *Bakalářská práce: Využití robota Lego Mindstorms při výuce předmětu A3B99RO Roboti.* Praha : Katedra řídicí techniky, České vysoké učení technické, 2010.

[5] Ing. Zuzaňák, Jiří. *Platforma J2ME a mobilní Bluetooth.* Brno : Ústav počítačové grafiky a multimédií, Fakulta informačních technologií, Vysoké Učení technické.

[6] Ing. Svoboda, Jiří. Principy a perspektivy technologie Bluetooth. *Sdělovací technika 8/2004.*

[7] Programování J2ME. *Interval.cz.* [Online] http://interval.cz/programovani/j2me/.

[8] LEGO MINDSTORMS NXT Hardware Developer kit . *LEGO.com.* [Online] 2006. http://mindstorms.lego.com/en-us/support/files/default.aspx.

[9] LEGO MINDSTORMS NXT Bluetooth Developer kit. *LEGO.com.* [Online] 2006. http://mindstorms.lego.com/en-us/support/files/default.aspx.

[10] *Java ME Technical Documentation.* [Online] http://download.oracle.com/javame/.

[11] Codeguru. *Simple Bluetooth Communication in J2ME.* [Online] http://www.codeguru.com/java/article.php/c13147.

[12] BlueTomorrow.com. *About Bluetooth.* [Online] http://www.bluetomorrow.com.

[13] leJOS Java for LEGO Mindstorms. *Software a Dokumentace leJOS NXJ API.* [Online] http://lejos.sourceforge.net/.

[14] LEGO Digital Designer . [Online] http://ldd.lego.com/.

[15] ORACLE. *Hardware and software.* [Online] http://www.oracle.com.

[16] Sluka, J. PID Controller For Lego Mindstorms Robots. [Online] http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html.

[17] RWTH - Mindstorms NXT Toolbox for MATLAB [Online] http://www.mindstorms.rwth-aachen.de

A Příloha

Obsah přiloženého CD

Na přiložené CD je následující struktura adresářů:

/DOKUMENTY/	obsahuje elektronickou verzi bakalářské práce ve formátu PDF
/NXJ/	obsahuje balík programu do robota včetně zdrojových kódů
/J2ME/	obsahuje aplikaci do mobilního telefonu včetně zdrojových kódů
/LEGO/	obsahuje soubor programu LEGO Digital Designer s modelem robota
/WEB/	obsahuje webové stránky prezentující soutěžní úlohu
/VIDEO/	obsahuje videa prezentující funkčnost robota