UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Mobilná aplikácia na ovládanie EV3 Lego robota

Bakalárska práca

2016 **Michal Hradečný**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Mobilná aplikácia na ovládanie EV3 Lego robota

Bakalárska práca

Študijný program: Aplikovaná informatika

Študijný odbor: 2511, Aplikovaná informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: RNDr. Peter Borovanksý, PhD.

Konzultant: RNDr. Peter Borovanksý, PhD.

Bratislava 2016 **Michal Hradečný**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Michal Hradečný
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Mobilná aplikácia na ovládanie EV3 Lego robota
Mobile Commander for Lego EV3

Cieľ: Cieľom bakalárskej práce je vytvoriť mobilnú aplikáciu pod operačným systémom Android na prepojenie riadiacej kocky EV3 robota stavebnice LEGO. Na komunikáciu bude použitá knižnica leJOS EV3 a spojenie realizované prostredníctvom Bluetooth alebo WiFi.

Anotácia: Aplikácia umožní užívateľovi ovládať pohyb robota v dvoch rôznych módoch. Prvým spôsobom je dotykom pomocou vizuálnych komponentov zobrazených na obrazovke telefónu. Druhým je využitie polohového G-senzora. Okrem toho bude aplikácia ovládať nemotorické senzory pripojené k robotovi. Mobilná aplikácia používa knižnicu leJOS EV3 umožňujúcej pristupovať k jednotlivým komponentom robota (motory, senzory, lcd, atď). Aplikácia bude konfigurovateľná, podľa aktuálneho zapojenia senzorov do riadiacej jednotky EV3.

Vedúci: RNDr. Peter Borovanský, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 04.10.2015

Dátum schválenia: 28.10.2015

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

Hradečný
študent

vedúci práce

Čestné prehlásenie

Čestne prehlasujem, že som túto bakalársku / diplomovú prácu vypracoval/a samostatne s použitím citovaných zdrojov.

.....

Michal Hradečný

Poďakovanie

Chcem sa poďakovať svojmu školiteľovi RNDr. Petrovi Borovanskému, PhD. za cennú pomoc, rady, konzultácie a čas, ktoré mi venoval počas písania bakalárskej práce.

Abstrakt

Michal Hradečný. Mobilná aplikácia na ovládanie EV3 Lego robota. Bakalárska práca. Univerzita Komenského v Bratislave, fakulta matematiky , fyziky a informatiky. Školiace pracovisko: Katedra aplikovanej informatiky. Vedúci práce: RNDr. Peter Borovanksý, PhD. Bratislava 2016.

K dokumentu je priložené CD so zdrojovým kódom aplikácie a samotný dokument v digitálnej (PDF) forme.

Bakalárska práca sa zaoberá vytvorením androidovej aplikácie , ktorá umožňuje dialkové ovládanie lego EV3 robota pomocou smartfónu. Hlavným cieľom bolo umožňit používateľovi ovládať pohyb robota. Ďaľším cieľom bolo umožniť zapojenie senzorov a zobrazovať ich namerané hodnoty. Nakoniec aplikácia taktiež umožňuje zadávať robotovi sériu príkazov na vykonanie. Dokument je rozdelený na viacero častí. V prvej časti sú obsiahnuté základné informácie o aplikácii, lego EV3 robotovi a jeho technológiách. V druhej časti sa dokument zaoberá návrhom grafickej časti aplikácie a taktiež návrhom dosiahnutia požadovanej funkcionality. Tretia časť podrobne vysvetľuje, ako bola naprogramovaná navrhnutá funkcionalita využitím programovacieho jazyka Java a pripojenej knižnice leJOS. Obsahuje taktiež návod ako nakonfigurovať robota, aby ho bolo schopné ovládať leJOS aplikáciou. Na konci dokumentu je celkové zhrnutie bakalárskej práce a zoznam využitých zdrojov, s pomocou ktorých bola aplikácia vytvorená.

Kľúčové slová: Aplikácia, android, lego EV3, robot, wifi, pohyb, senzory

Abstract

Document contains attached CD with the source code of the application and the document itself in a digital (PDF) form.

The topic of this bachelor thesis is the creation of an android application, which allows remote control of a lego EV3 robot using smartphone. The main goal was to allow the user to control the movement of the robot. The next goal was to allow connection of sensors and to display their measured values. Lastly, the application also allows the user to give the robot a series of commands to carry out. The document is split into several parts. First part contains information about the application and the lego EV3 robot and its technologies. In the second part, the document talks about the proposed graphical layout of our application and the plan to reach the desired funcionality. Third part explains in detail the process of programing the desired funcionality, using programming language Java and its imported class leJOS. It also includes a manual on how to configure the robot to be able to run leJOS applications. At the end of the document is an overall summary of the bachelor thesis and a list of used literature, which was used as a helping hand in creating the application.

Key words: Application, android, lego EV3, robot, wifi, movement, sensors

Obsah

1 Východisková kapitola

1.1 Cieľ bakalárskej práce

Cieľom mojej bakalárskej práce je vytvoriť mobilnú aplikáciu na ovládanie Ev3 Lego robota na operačný systém Android. Primárny rozdiel medzi mojou aplikáciou a aplikáciami spomenutými v kapitole 1.4 je v použitom programovacom jazyku a systéme na ktorom bude bežať. Moja aplikácia bude naprogramovaná v programovacom jazyku Java s pridanou knižnicou leJos a fungovať bude iba na robotovi s nainštalovaným leJos operačným systémom. Dve aplikácie spomenuté v kapitole 1.4 fungujú iba na robotovi so základným Mindstorm operačným systémom, ktorý má robot v sebe pri zakúpení.

Ďalej chcem poskytnúť väčšiu flexibilitu pohybu robota naprogramovaním viacerých módov pohybu. Prvý pomocou dotykového rotačného kolesa na otáčanie robota a ikonami na pohyb dopredu a dozadu. Druhý mód bude využívať accelerometer zabudovaný v smartfónoch na pohyb robota nakláňaním robota.

Ďalším cieľom je využitie aj nemotorických senzorov na poskytnutie rôznych doplnkových funkcií a informácií, ako napríklad využitie senzora na zabránenie nárazu do objektov pred alebo za robotom a zobrazenie vzdialenosti.

Posledným cieľom je naprogramovanie funkcie, v ktorej používateľ bude schopný naplánovať pohyb robota pomocou príkazov na pohyb dopredu, rotáciu, dozadu a následne spustiť reťaz príkazov.

Nastaviteľnosť aplikácie používateľom, ako výber zapojenia motorov a senzorov do rôznych slotov, je taktiež súčasťou aplikácie.

1.2 EV3 Lego robot

EV3 Lego robot je robotická súprava tretej generácia v takzvanej Lego Mindstorms línii. Súprava obsahuje množstvo súčiastok a senzorov, ktoré umožnujú používateľovi si poskladať robota poďla svojich predstáv, alebo využiť oficiálne lego manuály na poskladanie jedného zo základných robotov. Základné typy robotov sú:
TRACK3R, R3PTAR, SPIK3R, EV3RSTORM, GRIPP3R. Základom každého robota je EV3 kocka so 4 slotmi na zapájanie senzorov a 4 slotmi na zapájanie motorov.
Kocka taktiež obsahuje USB port, čítačku microSD kariet, obrazovku a podsvietené tlačidlá na pohybovanie sa v menu.

Stavebnice LEGO Mindstorms sú určené pre všetkých milovníkov modernej techniky. Mindstorms kombinuje inteligentnú kocku s mikropočítačovým mozgom, početnými senzormi a programovým softvérom s jednoduchým použitím typu "ťahaj a pusť". Umožní zostavovať a programovať roboty, ktoré vás budú počúvať. Veľkou prednosťou LEGO Mindstorms je možnosť jednotlivé modely prestavať v iné modely. Stavebnica LEGO EV3 rozvíja kreativitu, jemnú motoriku i predstavivosť detí a radí sa medzi jednu z najobľúbenejších detských hier.

Robot EV3 má rozsiahle možnosti úpravy a naprogramovania k vykonávaniu širokého spektra činností. Okrem pohybu robota dokáže rozsiahlými úpravami napríklad aj hrať šach,alebo hrať na gitare. Vďaka tomu je robot príťažlivý nielen pre deti, ale aj dospelých záujemcov. Pomocou microSD karty je možné nainštalovať kocke robota nový operačný systém (leJOS) a spúšťať na ňom aplikácie naprogramované v Jave, čo ďalej rozširuje flexibilitu robota.

1.3 Lego senzory podporované aplikáciou

1.3.1 Ultrasonický senzor

Ultrasonický senzor pracuje s vlnami a je často využívaný v aplikáciach na meranie vzdialenosti medzi robotom a okolitými objektami. Senzor vydáva zvukové vlny a spracuváva ich echá na detekovanie a vypočítanie vzdialenosti od jedného alebo viacerých objektov. Najnovší EV3 ultrasonický senzor môže byť okrem iného použitý ako aktívny alebo pasívny sonar. Je navrhnutý na využitie v širokej škále aplikacií, ktoré majú za úlohu skúmať vzdialenosť a detekovať objekty.

funkcie: : skúmanie vzdialenosti do 250 centimetrov (2.5 m)

: +/- 1 centimeter odchýlka v presnosti

: rozpoznáva iné ultrasonické zvuky

Obr. 1.3.1.1 EV3 Ultrasonický senzor

1.3.2 Gyro senzor

Gyroskopické senzory sa používajú ako pômocka pre stabilitu robota. Systém dokáže určiť rotáciu a orientáciu robota na viacerých osiach, čím poskytujú používateľom možnosť počítať stupne odchýlenia a navrhnúť robotov s navigačnými systémami.S využitím prvkov EV3 Gyro senzora sa dajú vytvoriť roboty s veľkou presnosťou pohybu.

funkcie: : počíta rotačný pohyb a zmenu v orientácii

: +/- 3 stupne odchýlka v presnosti

: maximálny výstup 440 stupňov za sekundu

Obr. 1.3.2.1 EV3 Gyro senzor

1.3.3 Farebný senzor (Color sensor)

Farebný senzor je často využívaný študentami a nadšencami na robotické aplikácie, ako napríklad nasledovanie farebnej čiary robotom, alebo detekciu normálneho alebo odrazeného svetla. Tento senzor sa dá využívať na detekovanie absencie farby, zmenami medzi farbami, alebo rozpoznanie jednej zo siedmich farieb (NXT farebný senzor dokáže rozpoznať iba 6 farieb). Taktiež dokáže skúmať intenzitu osvetlenia.

funkcie: : detekovanie jednej zo siedmich farieb

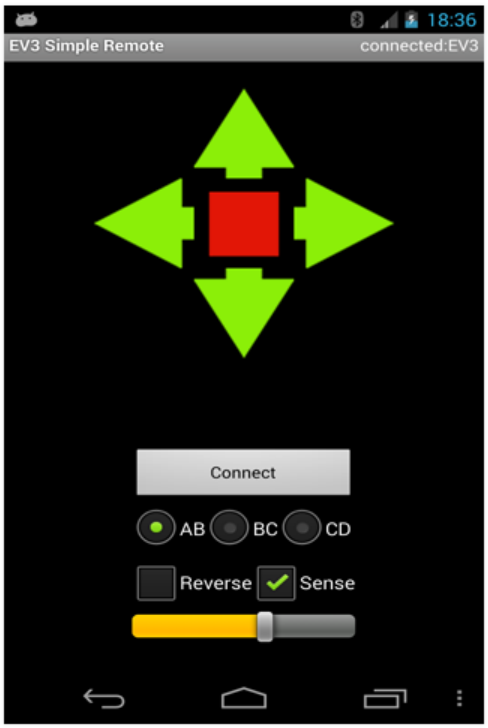
: detekovanie absencie farby

: funguje aj v ambientnom osvetlení

Obr. 1.3.3.1 EV3 Farebný senzor

1.4 Prehľad aplikácií na GOOGLE STORE

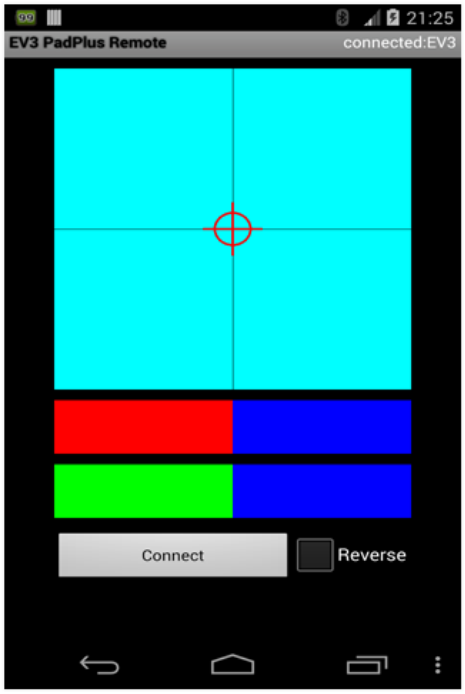
1.4.1 EV3 Simple Remote



Obr. 1.4.1.1 Interface aplikácie EV3 Simple Remote

Simple remote je aplikácia s jednoduchou funkcionalitou, ktorá umožňuje pohyb robota pomocou šípiek na obrazovke dopredu, dozadu a do strán, alebo otáčanie robota na mieste.
Aplikácia predpokladá dvojicu zapojených motorov a flexibilita spočíva iba v možnosti výberu ich rozmiestenia do slotov (AB,BC,CD). Ďalej sa dá nastaviť rýchlosť pohybu a prehodenie ľavého motora na pravý a naopak. Aplikácia nevyužíva zapojenie žiadnych senzorov a spojenie s robotom prebieha cez bluetooth.

1.4.2 EV3 Pad+ Remote



Obr. 1.4.2.1 Interface aplikácie EV3 Pad+ Remote

EV3 Pad+Remote je aplikácia na ovládanie motorov A a B robota pomocou ťahu terčíka na obrazovke. Motor C sa ovláda pomocou červeného a motor D pomocou zeleného dotykového baru. Pripojenie s robotom prebieha cez bluetooth. Ďalej aplikácia ponúka možnosť zapnutia spiatočného módu, ktorý prehodí pohyb dopredu a dozadu. Aplikácia je celkom jednoduchá a nevyužíva žiadne senzory. Nevýhoda spočíva v slabej flexibilite ovládania motorov, kedže vyššie spomenuté rozmiestnenie ovládacích prvkov pre jednotlivé motory je nemeniteľné.

1.5 Použité technológie

(Informácie získané alebo preložené z Wikipédie)

leJOS :

leJOS je firmvérová náhrada pre Lego Mindstorms programovateľné kocky. Rôzne varianty jeho softvéru podporujú rôzne legové roboty, ako robot NXT, alebo robot EV3. Obsahuje virtuálnu Java mašinu, ktorá umožňuje programovať Lego Mindstorms robotov v programovacom jazyku Java. Často sa využíva na výučbu Javy pre študentov počítačových vied. Ako zaújmavosť, robot "Jitter" bežiaci na leJOSe lietal na medzinárodnej vesmírnej stanici v decembri 2001.

Pojem leJOS zahŕňa taktiež leJOS API, využívané na programovanie a prístup k legovému robotovy a jeho súčiastkam pripojením k Java aplikácii.

Java :

Java je v dnešnej dobe jeden z najpopulárnejších programovacích jazykov. Je vyvíjaný spoločnosťou Oracle. Jeho syntax vychádza z jazykov C a C++. Zdrojové programy sa nekompilujú do strojového kódu, ale do medzistupňa, tzv. „byte-code“, ktorý nie je závislý od konkrétnej platformy. Java umožňuje developerom využívať princíp "write once, run anywhere" (WORA), čo znamená, že skompilovaný Java kód je možné rozbehať na všetkých platformách, ktoré podporujú Javu bez nutnosti rekompilácie.
Java bola vyvinutá Jamesom Goslingom v spoločnosti Sun Microsystems (ktorá bola neskôr odkúpená spoločnosťou Oracle) a vydaná v roku 1995.

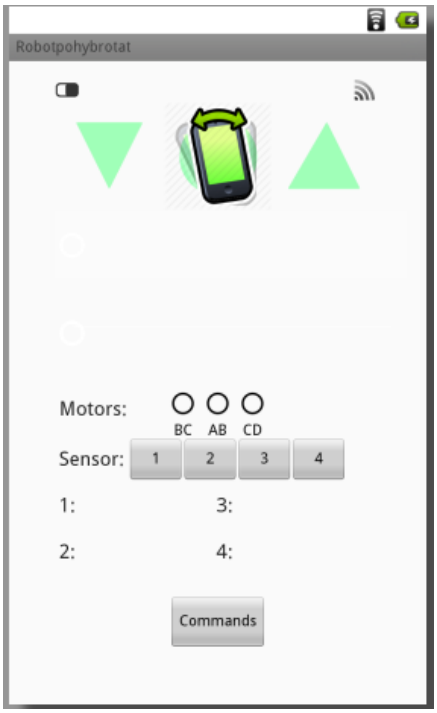
Android:

Android je rozsiahla open source platforma, ktorá vznikla najmä pre mobilné zariadenia (smartphone, PDA, navigácie, tablety). Zahŕňa v sebe operačný systém (založený na jadre Linux), middleware, používateľské rozhranie a aplikácie. Samotná platforma Android dáva k dispozícii nielen operačný systém s používateľským prostredím pre koncových používateľov, ale aj kompletné riešenie nasadenia operačného systému (špecifikácia ovládačov a pod.) pre mobilných operátorov a výrobcov zariadení a v neposlednom rade pre vývojárov aplikácií poskytuje efektívne nástroje pre ich vývoj - Software Development Kit.

2 Návrh

2.1 Grafický návrh aplikácie

Aplikácia bude mať nastavené zobrazenie prvkov na výšku (Portrait mode). V ľavom vrchnom rohu sa nachádza dotyková ikona na prehadzovanie medzi dvoma spôsobmi ovládania robota (ovládanie grafickými komponentami a ovládanie nakláňaním mobilu). V pravom vrchnom rohu je dotyková ikona pripojenia, ktorá spúšta dialóg pripojenie, kde treba zadať správnu ip adresu robota, alebo odpája už pripojeného robota. Pod týmito ikonkami sú grafické ovládacie prvky robota, šípka dopredu, dozadu a rotačné koleso pri grafickom móde ovládania, alebo obrázok mobilu pri nakláňacom móde pohybu. Nižšie sa nachádzajú dva ťahové ovládače rýchlosti pohybu a rotácie robota. Pod nimi sú tri radio buttony na nastavenie zapojenia dvojice motorov a 4 buttony na zapájanie senzorov do jednotlivých portov 1-4. Nižšie sú štyri textové polia zobrazujúce hodnoty získané zo zapojených senzorov. Nakoniec najnižšie je button Commands na spustenie dialógu, kde sa dajú robotovi zadať pohybové príkazy na vykonanie. Návrh je vidieť na obrázku 2.1.1.



Obr. 2.1.1 Grafický layout aplikácie

2.2 Pohyb robota

Aplikácia bude poskytovať používateľovi tri rôzne spôsoby, ako uviesť robota do pohybu a zadať mu pohybové príkazy. Prvý spôsob využíva grafické prvky na obrazovke na pohyb, ako šípku dopredu, dozadu a rotačné koleso so šípkou, kde ťahanie šípky po 360 stupňovej kruhovej osi zadáva robotovi príkaz na rotáciu. Počet stupňov o koľko sa robot zrotuje sa počíta na základe rozdielu novej a predošlej polohy šípky na osi po skončení ťahu. Druhý spôsob pohybu využíva accelerometer zakomponovaný v smartfóne na určenie pohybu a rýchlosti robota. Kedže grafický layout aplikácie je nastavený na výšku (Portrait mode), x-ová polohová os mobilu bude určovať rotáciu robota, kdežto z-ová polohová os mobilu bude určovať pohyb dopredu a dozadu. Posledná možnosť, ako uviesť robota do pohybu, je zadať mu sériu príkazov z dialógového okna, kde sa dá nastaviť počet stupňov pri rotácii a centimetrov pri pohybe dopredu a dozadu. Stláčaním štyroch buttonov (dopredu| dozadu|doprava|doľava) sa následne príkazy pridávajú za sebou do poľa. Nakoniec sa príkazy dajú vykonať v poradí v akom boli pridávané, alebo zmazať.

2.3 Využitie senzorov

K robotovi sa bude dať pripojiť štvorica senzorov do štyroch portov . Každý port bude mať na obrazovke button, po stlačení ktorého vybehne dialóg so štyrmi možnosťami na výber (tri rôzne druhy senzorov alebo žiaden). Po potvrdení výberu stlačením buttonu OK sa spustí senzor v danom porte (ak je tam pripojený), a začne periodicky odosielať získané hodnoty do textového okna na obrazovke určeného danému portu. Ak je nejaký senzor v danom porte zapnutý a vyberie sa možnosť None (žiaden), daný port sa vypne a prestane vysielať hodnoty. Na získavanie hôdnot z Lego senzorov je potrebné si vytvoriť takzvaný SampleProvider na danom porte a odoberať hodnoty z neho vždy po určitom časovom intervale.

2.4 Nastaviteľnosť

Jedným z hlavných cieľov aplikácie je, aby bola flexibilná a nastaviteľná. Tým pádom si ju každý používateľ vie prispôsobiť podľa vlastných predstáv, či už ju chce využívať iba na pohyb robota, alebo zapájať aj senzory.

Prvý spôsob, ako si vie používateľ upraviť aplikáciu, je kedykoľvek si vybrať jednu z troch možností zapojenia motorov do slotov. Na tento účel je na obrazovke klikateľná skupina RadioButtonov. Aplikácia využíva na pohyb dvojicu motorov, čiže všetky tri možnosti zapojenia motorov predpokladajú zapojenie dvoch motorov do dvoch slotov (obrázok 2.4.1).

Obr. 2.4.1 Zapájanie motorov

Ďalšia možnosť nastavenia aplikácie spočíva v nastavovaní rýchlosti pohybu a rotácie robota. Na obrazovke sa nachádzajú dva grafické prvky, jeden pre rýchlosť pohybu a jeden pre rýchlosť rotácie. Ťahaním týchto prvkov sa zvyšuje alebo znižuje rýchlosť.

Aplikácia taktiež poskytuje používateľovi možnosť zapojenia senzorov do slotov podľa jeho zvolenia. Na obrazovke sú štyri tlačidlá pre štyri rôzne senzorové sloty robota, po ktorých stlačení sa naskytne možnosť vybratia typu zapojeného senzora v danom slote. Senzory sú doplnková funkcionalita aplikácie a aplikácia sa dá používať aj bez senzorov.

Pri vytváraní spojenia medzi robotom a mobilom sa taktiež nastavuje priemer kolies robota a priemer kruhu, ktorý obkolesuje jeho kolesá. Toto nastavenie je dôležité pre presnosť pohybu.

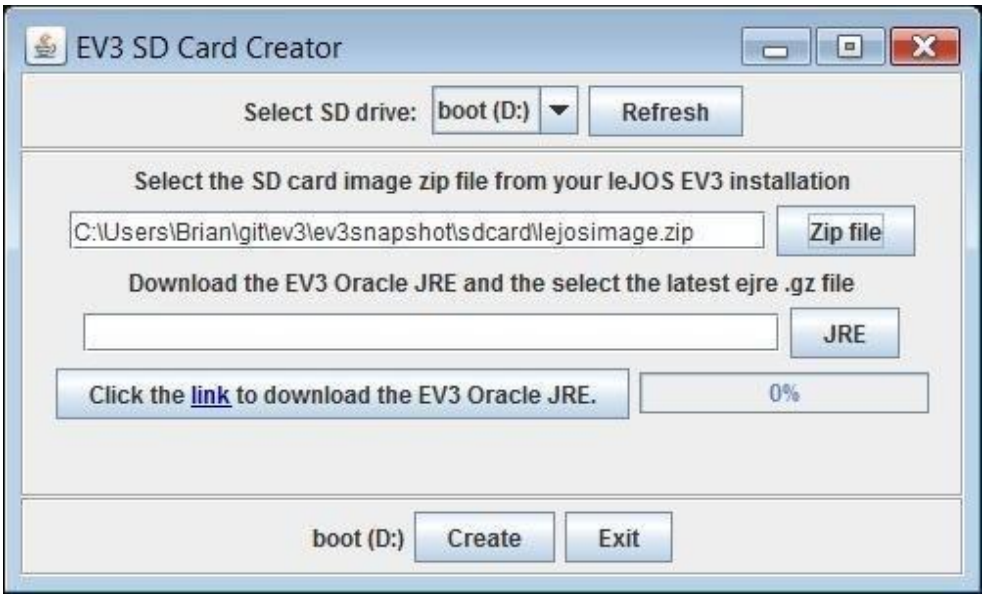
3 Implementácia

3.1 Inštalácia leJOSu

Ev3 robot má v sebe po zakúpení nainštalovaný základný Lego mindstorm operačný systém. Ak na ňom chceme byť schopný spustiť aplikácie naprogramované v Jave, musíme mu najprv nainštalovať leJOS operačný systém (v našom prípade je potrebná minimálne verzia 0.9). Tento proces je bohužiaľ momentálne relatívne obtiažny, čo je zrejme aj dôvod, prečo je zatiaľ málo používaný.

K inštalácii do robota si potrebujeme stiahnuť ideálne najnovšiu verziu leJOS systému z oficiálnej webovej stránky a microSD kartu s minimálne 2 GB pamäte naformátovanú vo FAT32 file systéme. Ešte treba stiahnuť Javu pre LEGO Mindstorm EV3 robota (v mojom prípade verzia 7) .

Ďalej treba nainštalovať leJOS do počítača, kde už na konci inštalácie sa nám poskytne možnosť vyrvoriť si SD kartu s operačným systémom. Tu si treba vybrať našu vloženú (prázdnu !) microSD kartu, lejosimage.zip z priečinka kde sa nám nainštaloval leJOS a stiahnutý Java (ejre.gz) súbor do posledného políčka.

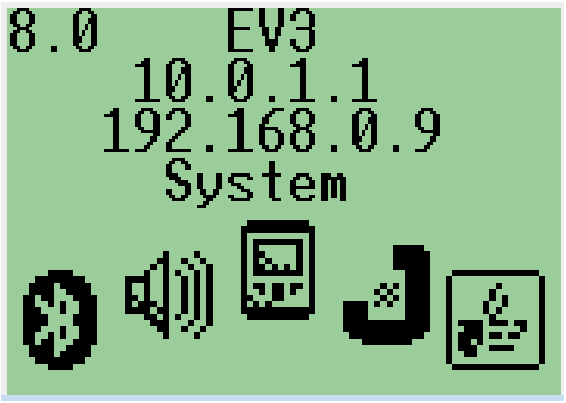


Obr. 3.1.1 Tvorba leJOS microSD karty

Kartu potom treba vložit do vypnutého Ev3 robota, zapnúť ho a inštalácia sa sama spustí. Proces inštalácie môže prvý raz trvať aj viac ako 10 minúť. Po vybratí leJOS microSD karty a zapnutí robota sa znova spustí pôvodný Lego Mindstorm operačný systém. Spomínané dva operačné systémy (leJOS/Mindstorm) sa dajú bez problémov prehadzovať (s/bez karty).
Po opätovnom spustení robota s vloženou leJOS kartou je už inštalácia podstatne rýchlejšia.

Po inštalácii sa zmení vzhľad menu na LCD obrazovke robota. Na obrázku 3.1.2 je vidieť pôvodné Mindstorm menu a na obrázku 3.1.3 Lejos menu po inštalácii.

Obr. 3.1.2 Mindstorm menu



Obr. 3.1.3 Lejos menu

Priebeh celej inštalácie a dôvod možných komplikácií je vysvetlený na zdrojových odkazoch číslo [4] a číslo [6] :

3.2 Pripojenie robota na wifi

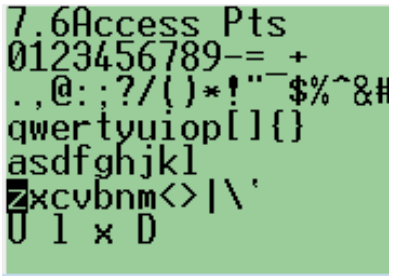
K pripojenie Ev3 robota cez wifi s počítačom alebo mobilným zariadením je potrebné, aby boli obidva objekty pripojené na rovnakej wifi sieti. Robot taktiež sám od seba nepodporuje pripojenie na wifi sieť. Je potrebné k nemu dokúpiť USB wifi adaptér z podporovaných modelov uvedených na lejos stránke: (https://lejosnews.wordpress.com/2015/02/03/comparing-wifi-adapters/).

Ja som sa rozhodol pre model **Edimax EW-7811UN 150Mbps Wireless Nano** (obrázok 3.2.1). Je to kompaktný,výkonný a ľahko dostupný adaptér.

Obr. 3.2.1 Edimax EW-7811UN 150Mbps Wireless Nano

Po zasunutí USB wifi adaptéra a zapnutí robota sa v priečinku wifi po chvíli objavia dostupné wifi siete v dosahu robota. Vybranie akejkoľvek siete spustí ďalšie okno, kde treba zadať heslo pre danú sieť (wifi siete, ktoré požadujú prihlásenie menom aj heslom nie sú podporované).

Obr. 3.2.2 Lejos výber wifi



Obr. 3.2.3 Lejos zadanie wifi hesla

Na prepojenie mobilu a robota na wifi sieti je potrebné v aplikácii stlačit ikonku pripojenia, po ktorej vybehne na obrazovke mobilu dialógové okno (Obrázok 3.2.4). V tomto okne je treba napísať wifi ip adresu robota, ktorá sa nachádza v treťom riadku na jeho LCD obrazovke. Ak spojenie úspešne prebehne, objaví sa na mobile nápis o vydarení pripojenia, inak sa objaví chybná správa. Ako bolo spomenuté, aby spojenie úspešne prebehlo, je potrebné aby boli mobilný telefón aj robot pripojení na rovnakej wifi sieti.

```
@SuppressLint({ "NewApi", "InflateParams" })
private void connectDialog(){

    dialogBuilder = new AlertDialog.Builder(this);

    View dialogView = factory.inflate(R.layout.alert_dialog, null);
    dialogBuilder.setView(dialogView);

    wifi = (EditText) dialogView.findViewById(R.id.Wifi_id);
    wheel = (EditText) dialogView.findViewById(R.id.Wheel);
    diameter = (EditText) dialogView.findViewById(R.id.Distance);

    dialogBuilder.setMessage("Robot parameters");
    dialogBuilder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            HOST = wifi.getText().toString();
            Diam = diameter.getText().toString();
            WheelDiam = wheel.getText().toString();
            if(Diam.length() == 0 || WheelDiam.length() == 0){
                Toast.makeText(MainActivity.this, "Some Diameter missing",Toast.LENGTH_LONG).show();
            }
            else{
            new Control().execute("connect", HOST);
            }
        }
    });

    AlertDialog connectip = dialogBuilder.create();
    connectip.show();
}
```

Obr. 3.2.4 Ukážka zdrojového kódu pre dialóg

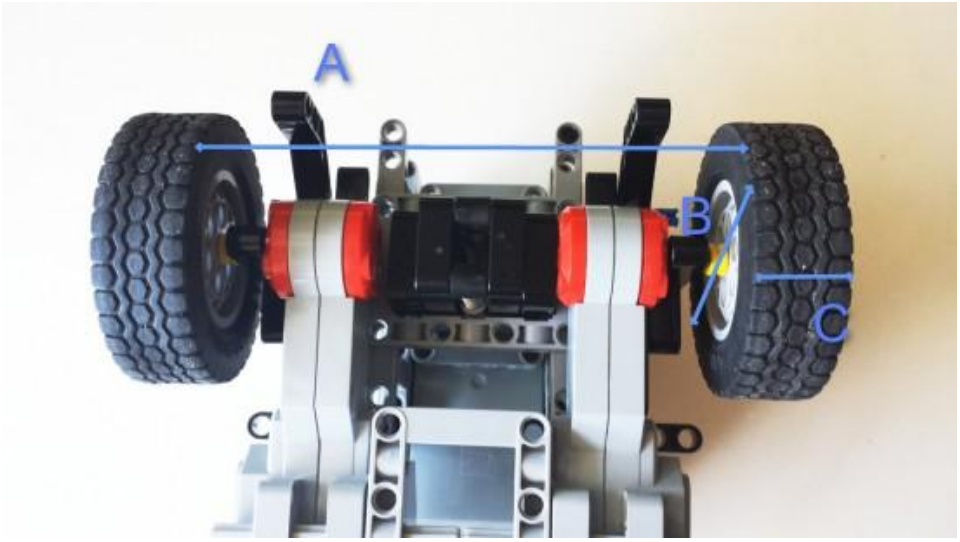
Robot vydáva zvukové signály po úspešnom začatí aj ukončení spojenia. Na zrušenie spojenia medzi robotom a mobilom treba znova kliknúť na ikonku pripojenia v hornom rohu obrazovky mobilu. Dialogové okno taktiež obsahuje dve vstupné políčka pre zadanie diametru (priemeru) kolesa (priamka B na obrázku 3.2.5) a diametru kruhu obkolesujúceho dve kolesá robota (dá sa vypočítať ako velkosť priamky A + priamky C na obrázku 3.2.5).

Presnosť zadaných parametrov pripojeného robota je dôležitá pre presnosť pohybu pri pohybe robota grafickými komponentami alebo zadávaním série príkazov. Tieto parametre sa zadávajú aj pri vytváraní takzvaného pilota pre EV3 robota, ktorý zjednodušuje programovanie pohybu robota. príklad :
pilot = (RemoteRequestPilot) ev3.createPilot(6, 13,"B", "C");

Kde prvé dva parametre za zátvorkou po "createPilot" predstavujú priemer kolesa a následne priemer medzi dvoma kolesami robota. Podmienka je, že obidve hodnoty musia byť rovnakého typu (cm,mm,m). parametre "B" a "C" určujú zapojenie dvoch motorov robota, v tomto prípade do portov B a C.

Vytvorený pilot následne využíva zadané parametre, aby pripojeným motorom po zadaní príkazov ako pilot.rotate(50) alebo pilot.travel(50) správne určil počet rotácií, a teda robot sa zrotoval o 50 stupňov, respektíve pohol sa o 50 centimetrov dopredu, ak boli priemery zadané pri vytváraní pilota odmerané v centrimetroch.

Viac informácií o spôsobe výpočtu rotácií podľa zadaných priemerov je možné nájsť na zdrojoch v odkaze číslo [5] .



Obr. 3.2.5 Osi priemerov EV3 robota

3.3 Pohyb pomocou grafických komponentov

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

    if (action == MotionEvent.ACTION_UP){
        if(switch_bool==false){
        new Control().execute("stop");
        }
    }
    else if (action == MotionEvent.ACTION_DOWN) {
        if (v == forward)
            new Control().execute("forward");
        else if (v == backward)
            new Control().execute("backward");
        else if (v.getId() == R.id.connect) {
            if (ev3 == null) {
                connectDialog();
            }
            else {
                new Control().execute("disconnect");
            }
        }
```

Obr. 3.3.1 onTouch ukážka kódu

Pohyb pomocou grafických komponentov využíva rozmiestnenie prvkov typu **ImageButton** na obrazovke, ktoré majú sprístupnenú dotykovú funkcionalitu vďaka metóde **setOnTouchListener(this)**. Následne sa im vo funkcii **onTouch** nastavuje funkcionalita, ktorá sa má vykonať po ich stlačení. V našom prípade sa po stlačení šípky dopredu alebo dozadu zavolá asynchrónny task s určitým príkazom (forward, backward) ako vidieť v ukážke kódu (obrázok 3.3.1).

Kým je jeden z buttonov stlačený, robot vykonáva pohyb dopredu alebo dozadu pomocou funkcie pilot.forward() alebo pilot.backward(). Po zdvihnutí prstu z buttonu sa zavolá nový asynchrónny task s príkazom "stop" pre robota a robot sa zastaví. Rotácia pomocou rotačného kolesa je vyriešená využitím **setRotationListener(this)** a metódy **onAngleChanged(int angle),** ktoré po zrotovaní šípky v strede kolesa zavolajú asynchrónny task s príkazom rotate a druhým príkazom stupňom rotácie v akom šípka skončila po potiahnutí.Následne sa robot zrotuje o nový stupeň rotácie šípky - predošlý stupeň rotácie šípky.Nakoniec sa do pomocnej premennej, ktorá si pamätá predošlý stupeň rotácie šípky, nastaví nová hodnota :

```
try{
pilot.rotate(angle - newAngle);
}
catch(Exception e){
}
angle = newAngle;
```

3.4 Pohyb nakláňaním mobilu

Pohyb nakláňaním mobilu využíva senzor typu acceletometer zabudovaný v každom modernom smartfóne na zisťovanie naklonenia mobilu. Zmeny naklonenia mobilu sa odchytávajú vo funkcii **onSensorChanged** (obrázok 3.4.1), kde sa dajú z accelerometra zistiť naklonenia na x-ových , y-ových a z-ových osiach. Predtým však treba taktiež príslušnej triede androidovej aktivity implementovať **SensorEventListener,** ktorý nám umožňuje odchytávať zmeny senzorov v metóde **onSensorChanged.**

```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {

    if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {

        x = sensorEvent.values[0];
        z = sensorEvent.values[2];
```

Obr. 3.4.1 Ukážka funkcie **onCheckedChanged**

Tieto odychytené hodnoty sa ukladajú do premenných x a z , kedže aplikácia funguje v portrétovom móde , teda móde na výšku obrazovky. Tým pádom nám x-ová os určuje rotáciu robota a z-ová os pohyb dopredu a dozadu. Y-ovú os môžme zanedbať.

Po spustení módu pohybu nakláňaním mobilu kliknutím na jeho ikonku na obrazovke, sa vytvorí nový asynchrónny task s príponou accelerometer , v ktorom sa sledovaním priebežne sa meniacich hodnôt premenných x a z nastavuje pohyb robota a jeho rýchlosť. Rýchlosť sa určuje násobkom z-ovej osi s konštantou SPEED_FACTOR, ktorá predstavuje jednu desatinu maximálnej rýchlosti robota vo funkcii **setTravelSpeed**:

pilot.setTravelSpeed(Math.abs((int)(z * SPEED_FACTOR)));

3.5 Zapojenie motorov

Aplikácia umožňuje používateľovi zvoliť si zapojenie motorov do slotov podľa vlastnej preferencie. Podmienkou je však zapojenie dvoch motorov (každý na jedno koleso). Na obrazovke je grafický prvok typu **RadioGroup,** ktorý v sebe obsahuje tri možnosti výberu typu RadioButton pre tri rôzne možnosti zapojenia motorov do slotov (AB,BC,CD). Tento prvok typu RadioGroup ma následne nastavenú funkciu **setOnCheckedChangeListener((OnCheckedChangeListener)this)**, ktorá umožnuje po vybratí jednej z možností typu RadioButton vykonať nejaký príkaz.

Tieto príkazy sa vykonávajú z funkcie **onCheckedChanged** (Obrázok 3.5.1).

```
public void onCheckedChanged(RadioGroup group, int checkedId) {

       int cid = radgrp.getCheckedRadioButtonId();

       switch(cid){
         case R.id.radio0:
              if(ev3!=null){
              try{
pilot      =      (RemoteRequestPilot)ev3.createPilot(Float.parseFloat(WheelDiam),
Float.parseFloat(Diam),"B", "C");
              }
              catch(Exception e){ }
              }
             break;
```

Obr. 3.5.1 Ukážka funkcie **onCheckedChanged**

Funkcia využíva identifikačné čísla vnorených RadioButtonov v skupine RadioGroup na určenie výberu. Pomocou funkcie switch sa následne odchytí vybraté identifikačné číslo RadioButtonu a vytvorí sa nový pilot pre robota so zapojením motorov zodpovedajucého výberu používateľa, a diametrami kolies uloženými pri vytvorení spojenia medzi robotom a mobilom. Ak nie je pripojený žiaden robot, funkcia nič nerobí.

3.6 Zadávanie príkazov robotovi

Na spodku obrazovky aplikácie sa nachádza tlačidlo s nápisom Commands, ktoré spúšťa nové dialógové okno so štyrmi prvkami typu **Button (**pre pohyb dopredu, dozadu, doprava a doľava**)** a štyrmi prvkami typu **SeekBar** (pre každý Button jeden). Pri každom novom kliknutí na Commands a spustení dialógového okna sa vytvorí nové pole Stringov.
commands = new ArrayList<String>();

Každý SeekBar má priradený textové okno, ktoré zobrazuje jeho momentálnu nastavenú hodnotu. Toto nastavovanie pre všetky SeekBary (aj pre nastavovanie rýchlosti motora a rotácie z hlavného okna aplikácie), sa deje vo funkcii **public void onProgressChanged** (obrázok 3.6.1)**.**

```
@Override
public void onProgressChanged(SeekBar seekBar, int progress,boolean fromUser) {
    if (seekBar == speed) speedValue.setText("Travel speed: " + progress);
    else if (seekBar == rotationSpeed) rotationSpeedValue.setText("Rotation speed: " + progress);
    else if (seekBar == left) leftValue.setText("Left(degrees):" + progress);
    else if (seekBar == right) rightValue.setText("Right(degrees):" + progress);
    else if (seekBar == forw) forwardValue.setText("Forward(cm):" + progress);
    else if (seekBar == back) backwardValue.setText("Backward(cm):" + progress);
}
```

Obr. 3.6.1 Ukážka funkcie **onProgressChanged**

Táto funkcia vždy pri zmene hodnoty na niektorom zo SeekBarov nastaví priradenému textovému oknu novú hodnotu zodpovedajúcu aktuálnej hodnote na SeekBare. Stláčanie Buttonov (Add Left, Right, Forward, Backward) ukladá hodnoty uložené v textových oknách SeekBarov do vytvoreného poľa Stringov commands aj s typom pohybu. Odchytávanie dotykov Buttonov sa deje vo funkcii **onTouch,** príklad :

```
else if (v.getId() == R.id.buttonForward) {
    commands.add(forwardValue.getText().toString());
}
```

Po stlačení potvrdzujucého tlačidla OK sa asynchrónne vykonajú všetky príkazy pohybu pre robota v takom poradí, v akom boli uložené do poľa. Tlačidlo Cancel zruší dialógové okno a zrušia sa uložené príkazy v poli.

3.7 Zapojenie senzorov

Na obrazovke sa nachádzajú štyri prvky typu **Button** pre štyri sloty robota na zapojenie senzorov (1-4), ktoré umožňujú používateľovi spustiť senzor v príslušnom vybranom slote a zobrazovať jeho hodnoty na obrazovke. Stlačením jedného z týchto prvkov sa spustí nový **AlertDialog,** kde si používateľ môže vybrať jednu zo štyroch možných typov senzora v danom slote (Ultra , Color, Gyro , None). Tento výber je zrealizovaný pridaním funkcionality **setSingleChoiceItems** a metódy switch v podliehajúcej funkcii **onClick.** (obrázok 3.7.1)

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Choose sensor in port 1").setSingleChoiceItems(sensors, -1, new DialogInterface.OnClickListener(){
    public void onClick(DialogInterface dialog, int which) {
        switch(which){
            case 0:
                selection= sensors[which];
                break;
            case 1:
                selection= sensors[which];
                break;
            case 2:
                selection= sensors[which];
                break;
            case 3:
                selection= sensors[which];
                break;
        }
```

Obr. 3.7.1 Ukážka funkcionality **setSingleChoiceItems**

Táto funkcionalita umožňuje používateľovi vybrať jednu možnosť z poľa štyroch Stringov s názvom **senzors** podľa ich poradového čísla v poli (od 0 po 3) a následne uložiť tento výber do pomocného Stringu s názvom **selection,** s ktorým aplikácia vie zaobchádzať aj v iných metódach.

Na prístup k hodnotám senzorov pripojených k robotovi je vytvorený prvok typu **RemoteRequestSampleProvider,** ktorý poskytuje možnosti na výber slotu, v ktorom je senzor zapojený , typ zapojeného senzora a taktiež jeho mód fungovania. (obrázok 3.7.2)

```
ultraSP = (RemoteRequestSampleProvider) ev3.createSampleProvider("S4",
        "lejos.hardware.sensor.EV3UltrasonicSensor", "Distance");
ultraSample = new float[ultraSP.sampleSize()];

colorSP = (RemoteRequestSampleProvider) ev3.createSampleProvider("S3",
        "lejos.hardware.sensor.EV3ColorSensor", "ColorID");
colorSample = new float[colorSP.sampleSize()];

gyroSP = (RemoteRequestSampleProvider) ev3.createSampleProvider("S1",
        "lejos.hardware.sensor.EV3GyroSensor", "Angle");
gyroSample = new float[gyroSP.sampleSize()];
```

Obr. 3.7.2 Ukážka vytvárania senzorov

V zátvorke za vytvorením SampleProvidera pomocou funkcie **createSampleProvider** vidíme tri nastavenia. Prvé nastavenie "S4" určuje slot zapojenie (označujú sa ako S1 až S4). Druhé nastavenie určuje typ senzora referenciou na jeho triedu z leJOS knižnice. Tretie a posledné nastavenie určuje jeho mód fungovania, keďže každý senzor má niekoľko rôznych módov (napr. "Distance" pri ultrasonickom senzore).

Po zapnutí senzora a vytvorení SampleProvidera treba taktiež vytvoriť premennú typu float [], ktorá uchováva hodnoty zo senzorov (premenné s prívlastkom Sample v obrázku 3.7.2). Do týchto premenných sa pomocu funkcie **fetchSample** ukladajú hodnoty z vytvorených SampleProviderov (obrázok 3.7.3). Nakoniec sa tieto odchytené hodnoty zobrazujú v textových oknách na obrazovke v určitých časových intervaloch využitím vlákien (thread).

```
ultraSP.fetchSample(ultraSample, 0);
colorSP.fetchSample(colorSample, 0);
gyroSP.fetchSample(gyroSample, 0);

int colorId = (int)colorSample[0];
colorName = "";
switch(colorId){
    case Color.NONE: colorName = "NONE"; break;
    case Color.BLACK: colorName = "BLACK"; break;
    case Color.WHITE: colorName = "WHITE"; break;
    case Color.BLUE: colorName = "BLUE"; break;
    case Color.GREEN: colorName = "GREEN"; break;
    case Color.YELLOW: colorName = "YELLOW"; break;
    case Color.RED: colorName = "RED"; break;
    case Color.BROWN: colorName = "BROWN"; break;
}
```

Obr. 3.7.3 Získavanie hodnôt senzorov

3.8 Dialógové okná

Aplikácia vo viacerých prípadoch na získanie príkazov od používateľa po stlačení tlačidiel (napr. na pripojenie robota k aplikácii) využíva vytváranie dialógových okien.

Tieto okná majú nastavený vlastný layout využitím prvku typu **LayoutInflater,** ktorý je inicializovaný v metóde **onCreate** hlavnej aktivity:

factory = LayoutInflater.from(this);

Tento prvok nám umožňuje využiť rozšírenú funkcionalitu xml súborov oproti bežným možnostiam prvkov typu **AlertDialog (**setPositiveButton , setMessage a podobne**)** a vytvoriť im upravený vzhľad s ľubovolnými prvkami. V metódach príslušných dialógov sa následne vytvorí nový **View,** ktorému sa pomocou premmenej **factory (LayoutInflater)** a metódy **inflate** nastaví layout (obrázok 3.8.1).

Následne sa v týchto metódach inicializujú grafické prvky z novo nastaveného layoutu, s ktorými môžeme pracovať a odchytávať ťuknutia na dotykové prvky v metóde **onTouch** hlavnej aplikácie.

```
dialogBuilder = new AlertDialog.Builder(this);

View dialogView = factory.inflate(R.layout.alert_dialog, null);
dialogBuilder.setView(dialogView);

wifi = (EditText) dialogView.findViewById(R.id.Wifi_id);
wheel = (EditText) dialogView.findViewById(R.id.Wheel);
diameter = (EditText) dialogView.findViewById(R.id.Distance);
```

Obr. 3.8.1 Layout vnútri dialógu

Po všetkých nastaveniach a naprogramovaní funkcionality dialógového okno treba toto okno nakoniec vytvoriť a zobraziť ho na obrazovke mobilu dvojicou príkazov:

```
AlertDialog connectip = dialogBuilder.create();
connectip.show();
```

4 Záver

V tejto kapitole je obsiahnuté celkové zhrnutie výsledkov bakalárskej práce. Podarilo sa úspešne implementovať dva rôzne spôsoby pohybu robota grafickými komponentami, alebo nakláňaním mobilu. Medzi týmito dvoma spôsobmi pohybu sa dá kedykoľvek prepínať pomocou ikony určenej na zmenu módu pohybu.
Pomocou grafických prvkov na obrazovke sa dá robotovi taktiež nastavovať rýchlosť jeho pohybu a rotácie.

Komunikácia robota s mobilom prebieha bezdrôtovo cez wifi sieť , kde obidve zariadenia musia byť pripojené na rovnakej wifi sieti a robot musí mať v sebe vsunutý USB wifi adaptér.

Aplikácia taktiež poskytuje možnosť zapojenia a merania hodnôt z troch rôznych typov legových senzorov (Ultrasonic , Gyro , Color) do ľubovolného slotu.
Podarilo sa taktiež vytvoriť konzolu na zadávanie série príkazov robotovi klikaním na tlačidlá určené príslušným druhom pohybu (dopredu,dozadu,doprava,doľava) a následne ich vykonať.

Aplikácia beží na leJOS operačnom systéme a vyžaduje si vytvorenie inštalačnej microSD karty pre robota. Ako taká je určená najmä ľudom, ktorí sa zaujímajú o, alebo majú skúsenosti s leJOS operačným systémom. Po prekonaní tohto vstupného zádrheľu je však jednoduchá na používanie aj pre deti.

5 Zdroje

1. Lukáš Dilík, Využitie G-senzora v mobilnom telefóne na ovládanie pohybu robota prostredníctvom Bluetooth (Bakalárska práca, FMFI UK, 2012)

2. Ukážka androidových aplikácii na leJOS EV3 robota
   (https://lejosnews.wordpress.com/tag/android, 3.2.2016) (Online)

3. Dokumentácia leJOS EV3 knižnice
   (http://www.lejos.org/ev3/docs/, 3.2.2016) (Online)

4. Tvorba leJOS microSD inštalačnej karty
   (http://sourceforge.net/p/lejos/wiki/Creating%20a%20bootable%20SD%20card/,3.2.2016) (Online)

5. Draw a Square: Using Motors in leJOS EV3
   (http://thetechnicgear.com/2014/05/tutorial-using-motors-draw-square-mindstorms-ev3-lejos/, 3.2.2016) (Online)

6. HOWTO Install leJOS 0.8.1 on Mindstorms EV3
   (https://www.youtube.com/watch?v=NyoF0Ws6SkY, 3.2.2016) (Online)

7. Framework senzorov
   (https://sourceforge.net/p/lejos/wiki/Sensor%20Framework, 9.5.2016) (Online)

8. Sensors for LEGO Mindstorms EV3
   (http://www.intorobotics.com/sensors-lego-mindstorms-ev3-features-comparison/, 9.5.2016) (Online)

6 Príloha

Priložené CD so zdrojovým kódom aplikácie a digitálnou formou dokumentu (PDF).