

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ŘÍZENÍ ROBOTICKÉHO VOZÍTKA POMOCÍ MOBILNÍHO ZAŘÍZENÍ

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

ALEŠ KAJZAR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ŘÍZENÍ ROBOTICKÉHO VOZÍTKA POMOCÍ MOBILNÍHO ZAŘÍZENÍ

CONTROLLING MOBILE ROBOTIC VEHICLE WITH USING MOBILE DEVICE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

ALEŠ KAJZAR

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. JAN SAMEK, Ph.D.

BRNO 2014

Abstrakt

Cílem této práce je představit robotické vozítko SRV-1 Blackfin a možnosti komunikace vozítka s mobilním zařízením s operačním systémem Android. Praktická práce se skládá z vytvoření Android knihovny a ukázkové aplikace, která tuto knihovnu využívá. Android knihovna zajišťuje bezdrátové spojení s vozítkem a jeho ovládání, ukázková aplikace využívá metod pro zapínání a vypínání laserů, pohyb a nastavení kamery.

Abstract

The goal of this bachelor's thesis is to introduce robotic vehicle SRV-1 Blackfin and possibilities of communication of the vehicle with a mobile device with Android operating system. Practical part of the thesis consists of creating Android library and application, which uses this library. Android library provides a wireless connection with the vehicle and it can control it, application uses methods of the library for turning on and turning off lasers, motion and changing camera settings.

Klíčová slova

Surveyor, Android knihovna, robotické vozítko

Keywords

Surveyor, Android library, robotic vehicle

Citace

Aleš Kajzar: Řízení robotického vozítka pomocí mobilního zařízení, bakalářská práce, Brno, FIT VUT v Brně, 2014

Řízení robotického vozítka pomocí mobilního zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Samka, Ph.D.

.....

Aleš Kajzar
21. května 2014

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Janu Samkovi, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce.

© Aleš Kajzar, 2014.

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1	Úvod	5
1.1	Motivace	5
1.2	Cíl a obsah práce	5
2	Robotické vozítko SRV-1 Blackfin	6
2.1	Hardwarové vybavení	7
2.2	Softwarové vybavení	8
2.3	Řízení robotického vozítka	8
2.3.1	Základní ovládání robota	9
2.3.2	Speciální příkazy	10
2.3.3	Příkazy pro rozpoznávání obrazu	11
2.3.4	Příkazy pro práci s neuronovou sítí	11
2.3.5	Interpret C	12
2.4	Realizované projekty	12
2.4.1	Konzole SRV1Test	12
2.4.2	Projekt SRV1Console	13
2.4.3	Vozítko řízené s využitím platformy Arduino	13
3	Android	14
3.1	Verze systému	14
3.2	Architektura	14
3.3	Struktura aplikace	15
3.3.1	Aktivity	15
3.3.2	Služby	16
3.3.3	Poskytovatelé obsahu	16
3.3.4	Záměry	16
3.4	Práce s funkcemi Android zařízení	16
3.4.1	Konfigurační soubor <code>AndroidManifest.xml</code>	16
3.4.2	Práce se sítí	17
3.4.3	Formulářové prvky	17
3.4.4	Dialogová okna, informační zprávy	17
3.4.5	Vertikální seznam položek	17
3.4.6	Nastavení	17
4	Návrh a implementace	18
4.1	Android knihovna	18
4.1.1	Použití knihovny	19
4.1.2	Struktura knihovny	19

4.1.3	Popis vybraných metod	21
4.2	Ukázková aplikace využívající knihovnu	22
4.2.1	Popis ovládání	22
4.2.2	Popis implementace	24
4.2.3	Obrazovka pro hledání barevných objektů	27
4.2.4	Obrazovka nastavení	27
4.3	Zhodnocení a možnosti rozšíření	28
5	Závěr	29
A	Obsah CD	32
B	Konfigurace robota	33
B.1	Nastavení matchportu	33
B.1.1	Sériové připojení	33
B.2	Instalace firmwaru	34
C	Metody knihovny	36

Seznam obrázků

2.1	Robotické vozítko SRV-1 Blackfin [20].	6
2.2	Stereovizní systém (SVS) [22].	7
2.3	Ovládání robota v prohlížeči [21].	8
2.4	Konzole SRV1Test.	12
2.5	Vozítko řízené s využitím platformy Arduino [23].	13
3.1	Podíl verzí systému na Android zařízeních k 1. květnu 2014 [8].	14
3.2	Android architektura.	15
4.1	Ilustrační schéma konceptu práce.	18
4.2	Část diagramu tříd knihovny.	20
4.3	Obrazovka po připojení k robotickému vozítku.	23
4.4	Obrazovka pro kreslení dráhy pohybu.	23
4.5	Kalibrace.	24
4.6	Obrazovka pro hledání barevných skvrn.	24
4.7	Graf závislosti času ujetí 1 m na nastavené rychlosti.	26
4.8	Graf závislosti času otočení o 360° na nastavené rychlosti.	26
4.9	Obrazovka nastavení ukázkové aplikace.	28
B.1	Ukázka sériového připojení	34

Seznam tabulek

2.1	Vybrané příkazy Wi-Fi protokolu pro základní ovládání.	9
2.2	Vybrané speciální příkazy Wi-Fi protokolu.	11
2.3	Vybrané příkazy Wi-Fi protokolu pro rozpoznávání obrazu.	11
2.4	Vybrané příkazy Wi-Fi protokolu pro práci s neuronovou sítí.	11
2.5	Příkazy Wi-Fi protokolu pro spouštění C interpretu.	12
4.1	Typy informačních zpráv ve výčtu <code>UpdatedStatus</code>	21
C.1	Metody knihovny.	39

Kapitola 1

Úvod

Robotika patří k významným vědním odvětvím, protože se roboti využívají zejména k usnadnění lidské činnosti, automatizaci práce a práci v oblastech, kde může být působení člověka obtížné nebo nebezpečné.

1.1 Motivace

Přestože jde vývoj stále více směrem k autonomii pohybu a rozhodování robota za pomoci technik umělé inteligence, stále bývá potřeba předávat zprávy o operacích, které má robot provádět, případně kdy má ukončit svou činnost. Tyto zprávy jsou většinou zasílány bezdrátově (pomocí Bluetooth, Wi-Fi, XBee, apod.). Zařízením, které tyto zprávy předává, může být například přenosný počítač nebo nějaký sestavený ovládací nástroj, nicméně pro ovládnutí robota může být výhodné použít zařízení, které je zároveň daleko přenosnější než počítač a sofistikovanější než sestavený modul ovladače. Toto zařízení může být mobilní telefon nebo tablet. V této práci se soustředím na komunikaci mezi zařízením s operačním systémem Android a robotickým vozítkem SRV-1 Blackfin.

Se zmíněným robotickým vozítkem je možné komunikovat přes speciální Wi-Fi protokol, který ale neumožňuje snadné zacházení se zprávami zasílanými od robota a je potřeba zprávy dále zpracovávat. Navíc je nutné při tvorbě každé aplikace řešit připojení a odpojení, ošetřovat chybové stavy apod., proto se zdá být výhodné vytvořit knihovnu, která bude tyto problémy eliminovat.

1.2 Cíl a obsah práce

Cílem práce je popsat možnosti bezdrátové komunikace mobilního zařízení s operačním systémem Android a navrhnout a implementovat knihovnu pro OS Android a na základě této knihovny vytvořit aplikaci, která bude knihovnu efektivně využívat.

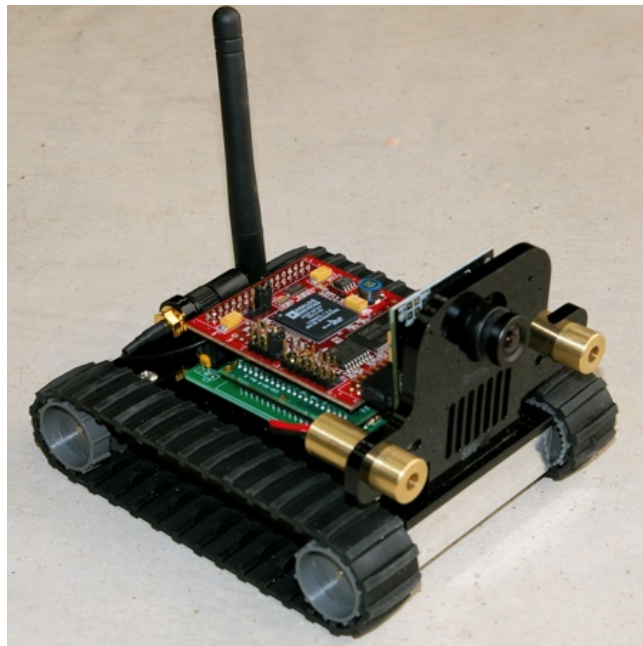
V kapitole 2 je popsáno robotické vozítko SRV-1 Blackfin z pohledu hardwaru i softwaru, jaké funkce vozítko nabízí a jak funguje Wi-Fi protokol robotického vozítka. V kapitole 3 je rozebírána architektura OS Android a je zmíněno několik komponent, které jsou využity v praktické části práce. Kapitola 4 popisuje praktickou část této práce, zmiňuje postup návrhu komunikačního rozhraní, samotný proces implementace, využití knihovny v ukázkové aplikaci a výsledky experimentování.

Kapitola 2

Robotické vozítko SRV-1 Blackfin

Robotické vozítko z dílny firmy Surveyor bylo vytvořeno pro výuku, výzkum a vývoj [20], nejde tedy o zařízení primárně určené pro použití v reálných situacích, ale o zařízení, které lze snadno spustit a učit se na něm chápat obecné principy vývoje programů pro roboty nebo z něj rychle postavit prototyp komplikovanějšího přístroje. První komerční verze byla uvedena již v roce 2006 a od té doby byla s tímto zařízením realizována řada projektů zaměřených na využití různých funkcionalit vozítka.

V této kapitole je podrobně popsáno hardwarové složení robota, volitelné rozšiřující komponenty, předinstalovaný firmware a možnosti komunikace pomocí Wi-Fi protokolu. Krátce je zde zmíněno několik projektů.

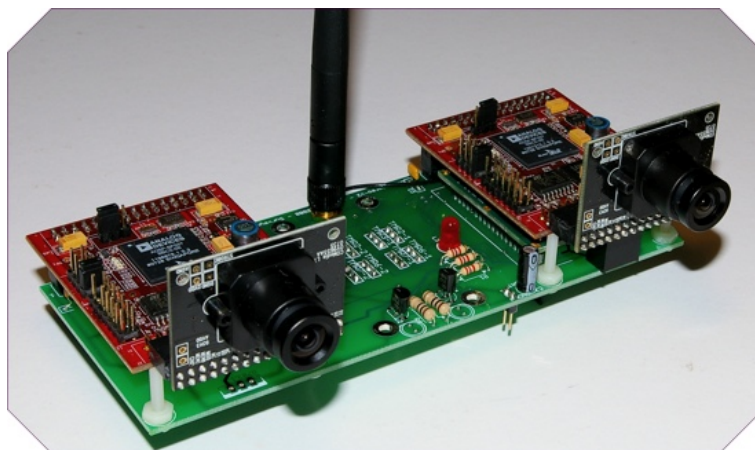


Obrázek 2.1: Robotické vozítko SRV-1 Blackfin [20].

2.1 Hardwarové vybavení

Vozítko, které má FIT VUT k dispozici, se sestává ze tří hlavních částí – ze základní desky Blackfin, Wi-Fi modulu a hliníkové kostry.

Na vrchní části vozítka je umístěna základní deska, která obsahuje mimo jiné procesor a kameru s rozlišením od 160x128 do 1280x1024 pixelů. Přímo pod základní deskou je uložen Wi-Fi modul Lantronix Matchport. Všechny tyto části jsou připevněny k hliníkovému tělu, na kterém je umístěna baterie, pásy s motory, Wi-Fi anténa a dva lasery; zmíněné složení není nijak zavazující, k vozítku lze připojit další přídavné komponenty (např. až 4 ultrazvukové senzory), Blackfin desku lze nahradit stereovizním systémem (SVS) se dvěma kamerami (na obrázku 2.2), případně dalšími moduly (viz web¹).



Obrázek 2.2: Stereovizní systém (SVS) [22].

Bližší popis parametrů [20]:

- Procesor: 1000mips 500MHz Analog Devices Blackfin BF537, 32MB SDRAM, 4MB Flash, JTAG.
- Kamera: Omnivision OV9655 1.3 MPix.
- Bezdrátové spojení: Lantronix Matchport 802.11b/g Wi-Fi.
- Dosah spojení: 100m uvnitř budovy, 1000m bez překážek.
- Senzory: 2 lasery pro zaměřování, podpora až 4 ultrazvukových senzorů (volitelný modul).
- Řízení: Pásy s motory.
- Kostra: Hliník.
- Rozměry: Délka 120mm, 100mm šířka, 80mm výška.
- Hmotnost: 350g.
- Napájení: 7.2V 2AH Li-Po baterie, po nabití vydrží čtyři a více hodin.

¹www.surveyor.com

2.2 Softwarové vybavení

Firmware robota je napsán v jazyce C pod licencí GPL. Není nádstavbou žádného operačního systému, program přímo pracuje s hardwarovým vybavením robota a funkce zpřístupňuje přes Wi-Fi protokol, který je popsán v další podkapitole.

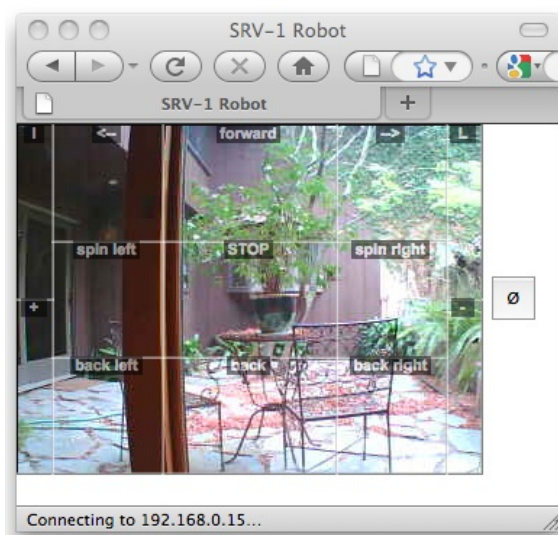
Firmware je možné poměrně jednoduše modifikovat díky tomu, že jsou zveřejněny veškeré zdrojové kódy v otevřené podobě a obsahují bohaté komentáře. Kompilace se provádí speciálním nástrojem „GNU Toolchain for the Blackfin Processor“. Jak přenést upravený (nebo aktualizovaný) firmware do robotického vozítka je popsáno v příloze B.2.

Původní verze firmware, se kterým bylo robotické vozítko uvedeno na trh, se lišila od té současné. Během následného vývoje byly (kromě dalších změn) přidány funkce pro rozpoznávání obrazu, funkce pro práci s neuronovou sítí, funkce pro práci s SVS a bylo opraveno několik chyb. Vývoj byl už pravděpodobně ukončen, poslední aktualizace proběhla v roce 2011, do té doby byly vydávány nové verze téměř každý týden [10]. Hlavními změnami poslední verze jsou aktualizace interpretu PicoC a oprava několika chyb, které s tímto interpretem souvisí.

2.3 Řízení robotického vozítka

Obecně lze robotické vozítko řídit z jakéhokoli zařízení, které obsahuje požadované ovládací prvky (dotykovou obrazovku, tlačítka, joysticky aj.), a které se dokáže připojit buď přímo k robotickému vozítku přes Wi-Fi, nebo k přístupovému bodu (AP), ke kterému se připojí také robotické vozítko. Takovým zařízením může být například PC, mobilní telefon, tablet nebo dálkový ovladač sestavený pro tyto účely. Popis konfigurace připojení robota k přístupovému bodu a další nastavení jsou popsána v příloze B.1.

Vozítko lze ovládat několika způsoby. Po přechodu na URL „<http://ip-adresa-robota:10001/index.html>“ (např. <http://169.254.0.10:10001/index.html>) v běžném prohlížeči se zobrazí přenášený obraz z kamery a pomocí tlačítek je možné ovládat pohyb robota, viz obrázek 2.3. Tuto konzoli lze upravit a zaměnit za vlastní, více informací viz [21].



Obrázek 2.3: Ovládání robota v prohlížeči [21].

Další možností je zasílat příkazy pomocí socketů, kdy se vytvoří spojení s „http://ip-adresa-roboty:10001“. Všechny příkazy, které je možné takto zaslat robotickému vozítku, se sestávají z ASCII znaků, které mohou být následovány osmibitovými čísly nebo ASCII znaky reprezentující čísla v desítkové soustavě, což usnadňuje použití protokolu v jednoduchých konzolách (není nutné vstup, který uživatel píše do textového pole, nijak upravovat a lze jej ihned odeslat). Robot tyto příkazy očekává v nekonečné smyčce. Po každém úspěšně přijatém příkazu robotické vozítko odesílá zpět odpověď, která je ve většině případů shodná s příkazem, liší se pouze prefixem '#'. Odpovědi, které mají proměnnou délku, začínají prefixem „##“. Není-li specifikovaná délka odpovědi, pak je na konci odpovědi znak „návrat vozíku“ a znak nového řádku. Pokud je zaslán příkaz, který nebyl rozpoznán, pak je zpět vrácen řetězec „#?“ [9].

Příkazy protokolu lze rozdělit do pěti kategorií (rozdělení vychází z dokumentace protokolu), jednak jde příkazy, které slouží pro základní ovládání robota, speciální příkazy, příkazy pro rozpoznávání obrazu, příkazy pro práci s neuronovou sítí a příkazy pro provedení kódu ve flash paměti díky integrovanému interpretu PicoC.

2.3.1 Základní ovládání robota

Do této kategorie je zařazeno ovládání pohybu robota, jeho volitelných přídatných zařízení, získávání informací o stavu robota a práce s flash pamětí. Některé příkazy protokolu jsou stručně popsány v tabulce 2.1.

Příkaz	Odpověď	Popis
„Mabc“	„#M“	Přímé ovládání motorů robota. Znaky „a“, „b“ a „c“ jsou osmibitové binární hodnoty, kde „a“ je rychlost levého motoru, „b“ je rychlost pravého motoru a „c“ trvání pohybu. Jsou-li rychlosti záporné, pak jde o opačný pohyb, je-li délka trvání pohybu nulová, pak jde o pohyb bez časového omezení.
„5“	„#5“	Zastavení pohybu motorů.
„Fab“	„#F“	Spuštění tzv. „Failsafe mode“, který nastaví rychlost motorů na hodnoty „a“ a „b“, pokud nepřišel za poslední dvě sekundy žádný příkaz.
„f“	„#f“	Zrušení „Failsafe mode“.
„I“	„##IMJxs0s1s2s3...“	Získání JPEG snímku z kamery.
„l“	„#l“	Zapnutí laserů.
„L“	„#l“	Vypnutí laserů.
„t“	„#l“	Počet milisekund, které uběhly od posledního resetu.
„D“	„##D“	Zjištění stavu baterie. Vrací text „battery voltage okay“ nebo „low battery voltage detected“.

Tabulka 2.1: Vybrané příkazy Wi-Fi protokolu pro základní ovládání.

Pohyb robota

Pohyb robota je možné řídit buď přímo příkazem pro ovládání jednotlivých motorů, kde lze nastavit rychlost (přípustné hodnoty jsou ASCII znaky s číselnou hodnotou 0 až 100) a délku trvání pohybu v milisekundách, nebo pomocí znaků numerické klávesnice. Tyto příkazy ale v tomto textu dále probírány nejsou, protože jejich funkčnost je podmíněna spuštěním prvního příkazu tabulky 2.1, tudíž se pro další použití zdají být nevhodné.

Ovládání připojených zařízení

Velmi důležitou součástí robota je kamera, která po zaslání příkazu „I“ sejme obraz a odešle v odpovědi komprimovaný JPEG. Je-li kamera zaneprázdněna, pak se žádný obraz neodesílá, proto se doporučuje zasílat příkaz do té doby, než je odpověď neprázdná. Rozlišení kamery může být nastaveno od 160x120 do 1280x1024, lze nastavit kvalitu obrázků nebo obraz zrcadlově převrátit.

K hliníkovému tělu jsou navíc připojeny dva lasery, které se dají nastavit do stavu „zapnuto“ a „vypnuto“ a v souvislosti s kamerou je lze využít pro změření vzdálenosti přední části robota od překážky.

Dále k robotovi mohou být připojeny dva servomotory, ultrazvukové senzory pro získávání informací o okolí robota a je možné zapisovat a číst z registrů sběrnice I2C a ovládat tím případně další zařízení. Tímto se ale tato práce nezabývá.

Získání informací o stavu robota

Několik příkazů umožňuje získání různých informací, které robot sbírá nebo uchovává. Mezi tyto příkazy patří příkaz pro získání času od posledního restartu, příkaz pro kontrolu stavu baterie a příkaz pro zjištění nainstalované verze firmwaru.

Práce s flash pamětí

Pro práci s flash pamětí existuje několik příkazů pro zápis a čtení dat. Data mohou být přenesena několika způsoby, jedním způsobem je použití některého z příkazů pro čtení nebo zápis definované délky dat. Další možností je spustit mód textového editoru příkazem „E“, ve kterém je možné přímo upravovat obsah bufferu (lze přesunovat kurzor mezi řádky a jednotlivě je přepisovat nebo mazat). Pro spolehlivější přenos souborů je implementován protokol XMODEM, který v každém paketu s daty odesílá také kontrolní součet.

Přenesená data pak mohou být buď interpretována speciálním příkazem jako C program (interpret PicoC je popsán v části 2.3.5), nebo mohou být použita příslušným příkazem pro přepis firmware.

2.3.2 Speciální příkazy

Speciálními příkazy je možné například zpracovávat GPS vstup, číst výstupy kompasu HMC6352 nebo HMC5843. Tyto moduly bohužel nejsou v současné době na FIT VUT k zapůjčení, proto nelze jejich funkci otestovat. Některé příkazy, které oficiální dokumentace uvádí, jsou v tabulce 2.2.

Příkaz	Popis
\$!	Reset zařízení.
\$g	Zpracování GPS vstupu.
\$C	Čtení dat z digitálního kompasu HMC6352.
\$c	Čtení dat z digitálního kompasu HMC5843.

Tabulka 2.2: Vybrané speciální příkazy Wi-Fi protokolu.

2.3.3 Příkazy pro rozpoznávání obrazu

Snímky z kamery dokáže zpracovávat robotické vozítko ještě před odesláním přes Wi-Fi. Protokol pro tyto účely nabízí příkazy pro detekci hran, detekci překážek a detekci horizontu. Každý tento příkaz je ve dvou variantách, první varianta detekované objekty pouze graficky zobrazí v přenášených snímcích, druhou variantou je, že se informace o detekovaných objektech přeneše jako souřadnice počátečních a koncových bodů všech odpovídajících úseček.

Protokol umožňuje obraz z kamery upravit pomocí příkazu pro vyrovnání citlivosti, automatické nastavení doby expozice a vyvážení bílé barvy tak, aby byly objekty na snímku co nejlépe rozpoznatelné pro člověka i pro případnou další automatickou analýzu.

Část příkazů pro rozpoznávání obrazu je popsána v tabulce 2.3.

Příkaz	Odpověď	Popis
„g0“, „g1“, ..., „g6“	např. „##g0“	Příkazy pro detekci horizontu, rohů, překážek, zobrazení rozdílů ve snímcích a vyhledání barevné skvrny v obrazu.
„g-“	„#g-“	Zrušení úprav obrazu nastavených příkazy „g0“ až „g6“.
„vax“	„#vax“	Příkaz zapíná nebo vypíná automatické vyrovnávání citlivosti, vyvážení bílé barvy a dobu expozice („x“ je číselný parametr, který určuje, které z těchto funkcí mají být aktivní).

Tabulka 2.3: Vybrané příkazy Wi-Fi protokolu pro rozpoznávání obrazu.

2.3.4 Příkazy pro práci s neuronovou sítí

Firmware robota obsahuje jednoduchou knihovnu pro práci s neuronovými sítěmi, které pro učení využívají zpětnou vazbu.

Příkaz	Popis
np	Uložení nového vzoru.
nd	Zobrazení uloženého vzoru.
ni	Inicializace sítě s náhodnými váhami.
nt	Trénování sítě pomocí uložených vzorů.

Tabulka 2.4: Vybrané příkazy Wi-Fi protokolu pro práci s neuronovou sítí.

2.3.5 Interpret C

Do flash bufferu lze vložit kód napsaný v jazyce C díky integrovanému interpretu PicoC. Interpret není kompletní implementací ISO C, ale obsahuje všechny hlavní struktury jazyka. Navíc je přidáno několik funkcí pro ovládání robota. Spustit interpretaci lze jedním příkazů uvedeným v tabulce 2.5.

Příkaz	Popis
Q	Spustí program uložený v paměti flash.
!	Spustí program v interaktivním režimu.

Tabulka 2.5: Příkazy Wi-Fi protokolu pro spouštění C interpretu.

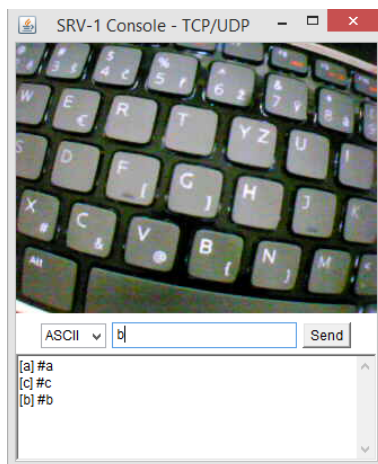
2.4 Realizované projekty

Jak bylo již zmíněno v úvodu, robot je primárně určen pro výuku, proto řada projektů vznikla buď přímo jako práce studentů, nebo pro jejich motivaci a usnadnění práce s robotem tak, aby se mohli soustředit na jimi zkoumaný problém (např. [14]).

Přímo na webových stránkách firmy Surveyor je k dispozici několik ukázkových projektů, které byly s robotickým projektem realizovány a několik odkazů na další projekty. S využitím Wi-Fi protokolu byla vytvořena řada konzolí a knihoven v různých jazycích a implementována rozšíření do některých známých nástrojů pro práci s roboty. Část projektů byla zaměřena také na autonomní pohyb robota ([16], [23]). Něteré z projektů budou zmíněny v této podkapitole.

2.4.1 Konzole SRV1Test

Projekt SRV1Test [13] je napsán v Javě a primárně slouží pro snadné experimentování s protokolem a pro otestování komunikace mezi počítačem a robotickým vozítkem. Po přeložení a spuštění programu a připojení k robotovi se zobrazí okno, ve kterém je obraz přenášený z kamery, a pod ním textové pole, do kterého je možné přímo zadávat příkazy, odesílat je a sledovat, co je vráceno zpět.



Obrázek 2.4: Konzole SRV1Test.

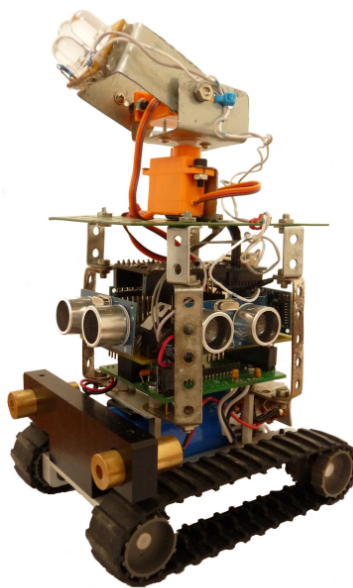
2.4.2 Projekt SRV1Console

Projekt s názvem SRV1Console [18] je aplikace pro mobilní zařízení s operačním systémem Android, ve které je možné se připojit k robotickému vozítku, sledovat přenášený obraz z kamery, zapínat a vypínat lasery a ovládat servomotory. Pokud zařízení obsahuje G-Senzor, pak lze ovládat také pohyb vozítka. Aplikace obsahuje obrazovku s nastavením, lze nastavit ale jenom IP adresu robota.

V aplikaci je bohužel několik chyb a je určená pro starší verze Androidu (kód totiž obsahuje volání metod, které nejsou podporované už od Android API verze 8) a v tuto chvíli už není vyvíjena dále. Přesto byla inspirací pro praktickou část této práce a část kódu je převzata a upravena tak, aby byly zmíněné problémy eliminovány.

2.4.3 Vozítko řízené s využitím platformy Arduino

Toto vozítko bylo na FIT VUT tématem řady bakalářských prací, např. [23], kde se autor zabývá řízením vozítka platformou Arduino. V rámci práce byl implementován obousměrný bezdrátový přenos mezi moduly pomocí XBee modulů a pro řízení byl vytvořen modul ovladače. K vozítku byly připojeny ultrazvukové senzory pro měření vzdálenosti mezi vozítkem a překážkou v jeho okolí, a LED světelná rampa pro osvětlení prostou mimo směr jízdy (vozítko je na obrázku 2.5).



Obrázek 2.5: Vozítko řízené s využitím platformy Arduino [23].

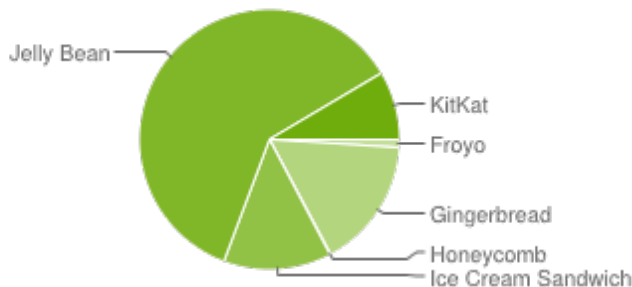
Kapitola 3

Android

Na trhu s mobilními telefony v tuto chvíli převládá operační systém Android [3], který má širokou základnu vývojářů mobilních aplikací i běžných uživatelů. Android OS nabízí aplikacím relativně dobře zpracované API s množstvím balíčků, které zpřístupňují jednotlivé funkce zařízení, což je také důvod, proč se stal tak oblíbeným a společně s iOS zabírá drtivou většinu trhu operačních systémů pro mobilní zařízení.

3.1 Verze systému

Během vývoje bylo dosud vydáno několik verzí systému, viz obrázek 3.1. V době zpracování této práce byla mezi zařízeními s Androidem nejrozšířenější verze s označením Jelly Bean [8], proto byla praktická část práce optimalizována pro Jelly Bean 4.1.



Obrázek 3.1: Podíl verzí systému na Android zařízeních k 1. květnu 2014 [8].

3.2 Architektura

Tento stručný popis architektury vychází z [2]. Architektura Androidu se sestává z pěti vrstev (viz obrázek 3.2).

Základní vrstvou je upravené jádro Linuxu ve verzi 2.6. Toto jádro přímo komunikuje s hardware a obsahuje veškeré ovladače zařízení.

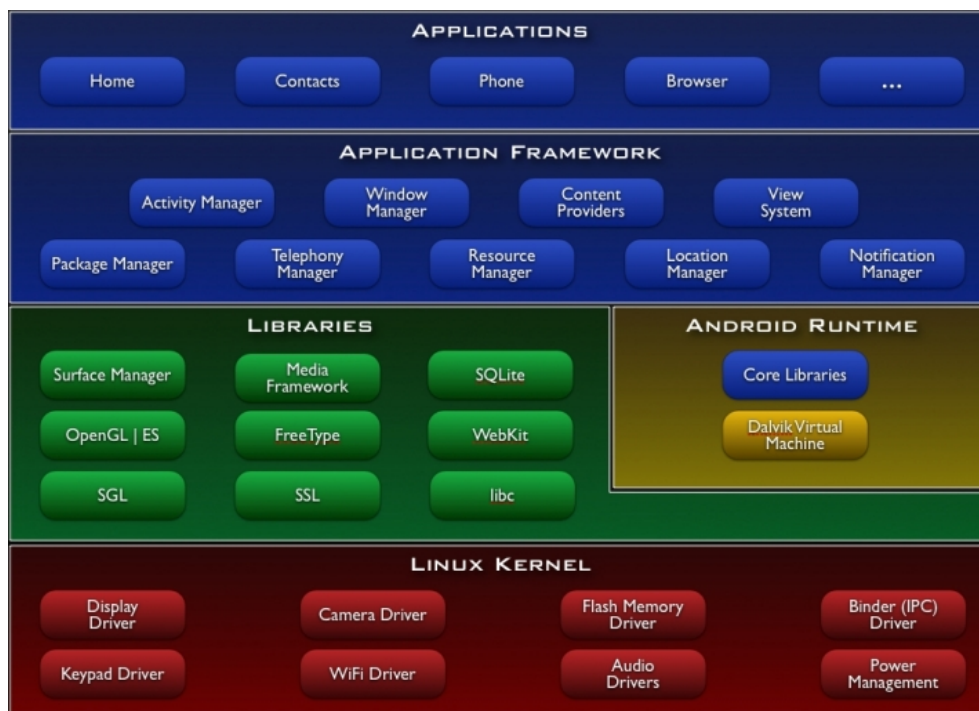
Další vrstvou jsou knihovny, které jsou napsány v C nebo C++. Tato vrstva zahrnuje například SQLite databázi, jádro WebKit nebo OpenGL pro vykreslování 2D a 3D grafiky.

Vrstva Android Runtime se skládá z virtuálního stroje Dalvik a Java knihoven. Dalvik je open-source software typu Java Virtual Machine speciálně vytvořený a optimalizovaný

pro Android zařízení [7].

Jednou z posledních vrstev je Application Framework. Tento framework definuje základní strukturu programu a umožňuje zobrazovat jednotlivé obrazovky, sdílet data mezi různými aplikacemi a díky němu lze využívat balíčky, které aplikacím zpřístupňují jednotlivé funkce zařízení (telefonní hovory, získávání GPS souřadnic apod.).

Poslední vrstvou je soubor základních předinstalovaných aplikací zařízení, jako je například SMS klient, webový prohlížeč a seznam kontaktů.



Obrázek 3.2: Android architektura.

3.3 Struktura aplikace

Každá aplikace v Androidu se skládá z komponent. Mezi hlavní patří aktivity, služby, komponenty pro poskytování obsahu, a záměry (angl. intentions). Následující text vychází z [11].

3.3.1 Aktivity

Aktivity jsou části aplikace, které přímo komunikují s uživatelem pomocí uživatelského rozhraní. Zde je možné reagovat na události (např. kliknutí na tlačítko, vložení textu). Aktivity jsou entity systému analogické k oknu klasické webové stránky, takže lze umožnit uživatelům, aby otevírali nové aktivity klepnutím a vraceli se do dříve otevřených aktivit pomocí tlačítka „Zpět“.

Aktivity od svého spuštění až po jejich ukončení prochází několika stavy a jsou volány tyto metody [1]:

- `onCreate()` - Metoda, která je volána po prvním spuštění aktivity.

- `onRestart()` - Metoda, která je volána po zastavení aktivity před tím, než je volána metoda `onStart()`.
- `onStart()` - Metoda, která je volána před tím, než je aktivita zobrazena uživateli. Po ní může následovat volání metody `onResume()` nebo `onPause()` v závislosti na tom, zda se aktivita vrátí do popředí, nebo je skryta na pozadí.
- `onResume()` - Metoda, která je volána po tom, co je aktivita zobrazena uživateli.
- `onPause()` - Metoda volaná po skrytí aktivity na pozadí.
- `onStop()` - Metoda, která se volá, když je aktivita zastavena.
- `onDestroy()` - Metoda volaná při rušení aktivity systémem.

3.3.2 Služby

Služby slouží pro dlouhodobou činnost aplikace na pozadí a jsou nezávislé na aktivitách, to znamená, že mohou být spuštěny i v době, kdy byla řídicí aktivita ukončena. Typicky se využívají pro přehrávání hudby na pozadí nebo ke kontrole dostupných aktualizací v informačním kanálu RSS.

3.3.3 Poskytovatelé obsahu

Poskytovatelé obsahu zajišťují možnost sdílet dat mezi více aplikacemi. Pomocí nich lze např. přistupovat k SQLite databázím ostatních vnitřních aplikací (úložiště SMS, kontakty apod.) [19].

3.3.4 Záměry

Záměry jsou systémové zprávy, které kolují v zařízení a upozorňují aplikace na různé události, počínaje změnami stavu hardwaru (například vložení karty SD) přes příchozí data (například přijetí zprávy SMS) až po události aplikací (například spuštění aktivity z hlavní nabídky zařízení). Záměry jsou velmi podobné zprávám a událostem jiných operačních systémů.

3.4 Práce s funkcemi Android zařízení

V rámci komponent uvedených v předešlé kapitole lze pracovat jak s jednotlivými funkcemi zařízení, na kterém operační systém Android běží, tak s předdefinovanými komponentami uživatelského rozhraní.

Další text bude zaměřen pouze na ty komponenty, které budou následně využity v praktické části této práce.

3.4.1 Konfigurační soubor `AndroidManifest.xml`

Každá aplikace musí mít konfigurační soubor `AndroidManifest.xml` (přesně s tímto názvem), který obsahuje základní informace o aplikaci, které systém Android musí zjistit ještě před tím, než je aplikace spuštěna. Mezi hlavní informace, které jsou takto předávány, patří: jméno aplikace, popis komponent (aktivit, služeb, poskytovatelů obsahu apod.), deklarace minimální verze API, seznam knihoven a povolení k využití různých funkcí zařízení. Je

nutné například povolit práci se sítí (odesílání a přijímání dat), služby pro určení zeměpisné polohy nebo využití digitálního kompasu [5].

3.4.2 Práce se sítí

Android zařízení neumožňuje připojení k ad-hoc Wi-Fi síti, která je vysílána robotickým vozítkem, proto je nutné nejdříve robotické vozítko i Android zařízení nastavit tak, aby se připojilo k přístupovému bodu (AP). Konfigurace vozítka je popsána v příloze B.

Další variantou (pokud to daný telefon nebo tablet umožňuje) je využít předinstalovanou aplikaci „Hotspot Wi-Fi“, ve které lze vytvořit síť, ke které se může vozítko připojit.

3.4.3 Formulářové prvky

Na obrazovkách lze zobrazit několik typů formulářových prvků, které na sobě mohou mít navázané události a tím reagovat na vstupy uživatele. Mezi ty nejčastěji používané patří textová pole, tlačítka, zatrhávací boxy a přepínače, nicméně je možné tyto prvky rozšířit a vytvořit vlastní.

3.4.4 Dialogová okna, informační zprávy

Účelem dialogových oken je obvykle buď upozornit na nějakou informaci, nebo zobrazit velmi jednoduchý formulář, pomocí kterého se po uživateli vyžaduje rozhodnutí o další prováděné akci. Vzhled všech oken je velmi podobný, okno se skládá z názvu, samotného obsahu a tlačítek, které danou akci potvrzují nebo zamítají.

Informační zprávy („toasty“) jsou velmi krátké textové zprávy, které se obvykle zobrazí na spodní části obrazovky po určité době.

3.4.5 Vertikální seznam položek

Seznam položek, které jsou řazeny vertikálně pod sebou je nazván ListView. Jednotlivé položky jsou do seznamu přidávány pomocí komponenty Adapter, která definuje vzhled položky a naplní jí obsahem [12].

Položky seznamu mohou na sobě mít navázané různé události, například se může provést nějaká akce po kliknutí, nebo dlouhém podržení položky.

3.4.6 Nastavení

Velká část aplikací používá různá nastavení pro změnu chování dané aplikace [17]. Například v některých aplikacích lze povolit nebo zakázat zvuky, nastavit URL serveru, na který se má aplikace připojovat apod.

Nastavení je speciální aktivita (`PreferenceActivity`), která používá jako zdroj popisu jednotlivých možností nastavení soubor ve formátu XML. Ve většině případů není potřeba implementovat nic dalšího, framework sám zajistí, aby se po přechodu do aktivity na obrazovce vykreslil seznam polí pro zadání nastavení, a po opuštění aktivity je vše automaticky uloženo.

Kapitola 4

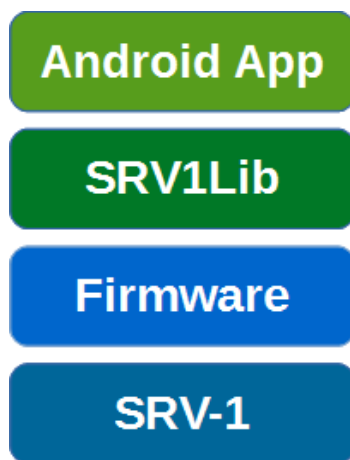
Návrh a implementace

Praktická část této práce zahrnuje popis vytvořené knihovny, která zprostředkovává nejen základní funkce robota, ale také komplikovanější prvky chování a zjišťování stavu robota a jeho okolí, a dále ukázkové aplikace využívající tuto knihovnu.

Idea vytvoření takové knihovny vychází mimo jiné z předpokladu, že část výpočtů lze v budoucnu ve složitějších aplikacích delegovat z robotického vozítka na mobilní zařízení (telefon, tablet), které dokáže výpočty zpracovat lépe díky často výkonnějšímu procesoru a větší operační paměti. Využití mobilního zařízení může mít mnoho výhod oproti notebooku nebo stolnímu počítači: za zmínku například stojí rozdílné hardwarové vybavení (G-Senzor, dotyková obrazovka), které může pomoci k lepší kontrole ovládání.

4.1 Android knihovna

Celý koncept si lze představit jako několik vrstev, které na sebe navazují. Nejnižší postavenou vrstvou je hardware robota s procesorem Blackfin BF537, který přímo zpracovává jednotlivé instrukce. Nad ním stojí firmware, který dokáže ovládat jednotlivé funkcionality a přijímá a zpracovává příkazy zaslané přes Wi-Fi. Další (zde navrhovanou) vrstvou je Android knihovna, jejíž úkolem je snadné připojení k SRV-1 a vytvoření srozumitelných metod pro provádění požadovaných operací. Poslední vrstvou jsou aplikace, které knihovnu využívají. Ilustrační schéma je na obrázku 4.1.



Obrázek 4.1: Ilustrační schéma konceptu práce.

Knihovna je založena na kódu `srv1console` [10], což je aplikace určená pro mobilní telefon s operačním systémem Android, která zpřístupňuje pouze několik základních funkcí robota. Kód byl podstatně upraven a rozšířen tak, aby umožňoval využívat většinu funkcí robota a byl použitelný také v novějších verzích Androidu. Knihovna je kompatibilní s Android API 11 a novějším.

4.1.1 Použití knihovny

Pro použití knihovny v nově vytvořeném projektu je potřeba přidat do konfiguračního souboru `AndroidManifest.xml` oprávnění `android.permission.INTERNET` a do složky `libs` vložit soubor knihovny `srv1lib.jar`. Pak stačí kdekoli, kde má proběhnout připojení k robotickému vozítku, vytvořit instanci třídy `SRV1Lib`. Konstruktor `SRV1Lib(String host, Handler new_handler)` očekává dva parametry, tím prvním je IP adresa robota, druhý parametr je objekt, který je určen pro reakci na došlé zprávy z robotického vozítka. Poté již je možné využít jednotlivé metody knihovny. Příklad takového kódu:

```
Handler interface_handler = new Handler() {
    public void handleMessage(Message msg) { }
}
```

```
SRV1Lib srvLib = new SRV1Lib('192.168.173.15', interface_handler);
srvLib.setLaserOn(true);
```

Knihovna má metody dvojího typu, jedny pouze nastavují činnost robota, druhé aktualizují nějakou informaci. První typ zastupuje např. metoda `setLaserOn(boolean on)`, která jen zapne nebo vypne lasery. Druhý typ zastupuje např. `requestVersion()`. Po zavolání metody `requestVersion()` a přijetí odpovědi se provolá metoda `handleMessage()` objektu předaného parametrem `new_handler` s příslušným stavem (`UpdatedStatus.VERSION_INFO`), který informuje o aktualizaci informace o verzi firmwaru. Tato metoda může obsahovat například následující akci:

```
if(msg.what == UpdatedStatus.VERSION_INFO.getValue())
    Log.d('Verze', srvLib.getVersion());
```

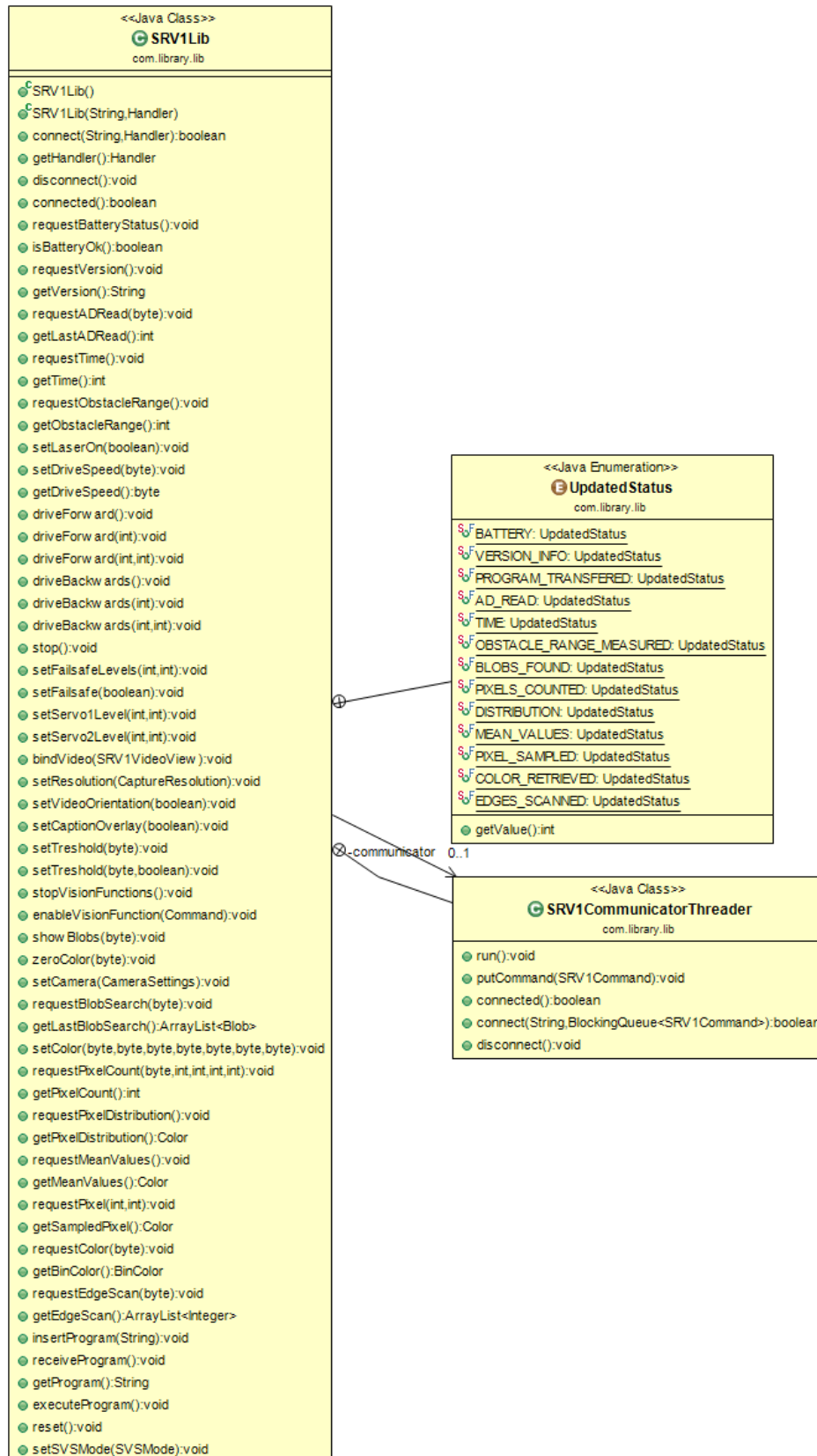
V tomto příkladu je pro získání řetězce s posledním přijatým názvem verze použita metoda `getVersion()`. Řetězec je pak zobrazen jako zpráva v konzoli. Všechny druhy informačních zpráv jsou uvedeny v tabulce 4.1.

4.1.2 Struktura knihovny

Metody knihovny zpřístupňují většinu funkcí robota. Ty, které je možné využít, jsou zobrazeny na obrázku 4.2, jejich kompletní a detailní popis v angličtině je v příloze C.

Každá metoda pracuje tak, že vytváří instanci jedné ze tříd, která rozšiřuje abstraktní třídu `SRV1Command`. Využívá se pro informování, zda je nutné příkaz opakovat při chybě, a také pro zpracování došlých dat.

Tato instance je poté předána vláknu `SRV1CommunicatorThreader`, které zašle konkrétní příkazy robotickému vozítku, poté přečte vrácená data a znovu je předá zmíněné instanci rozšířené abstraktní třídy `SRV1Command`. V některých případech (např. při chybné odpovědi) je pak zaslán příkaz robotickému vozítku znovu, nebo je informován objekt typu `Handler` o proběhlé události (např. že byl přečten čas od spuštění robotického vozítka) s příslušným stavem výčtu `UpdatedStatus`.



Obrázek 4.2: Část diagramu tříd knihovny.

Stav	Význam
BATTERY	Byla aktualizována informace o úrovni nabití baterie.
VERSION_INFO	Byla získána informace o verzi nainstalovaného firmware.
PROGRAM_TRANSFERED	Program byl přenesen.
AD_READ	Byla přečtena hodnota A/D převodníku.
TIME	Byl získán čas od posledního resetu robotického vozítka.
OBSTACLE_RANGE_MEASURED	Byla změřena vzdálenost od překážky.
BLOBS_FOUND	Byla vyhledána pole s definovanou barvou.
PIXELS_COUNTED	Byl sečten počet pixelů dané barvy.
DISTRIBUTION	Bylo spočítáno rozložení barev.
MEAN_VALUES	Byly spočítány střední hodnoty Y, U a V.
PIXEL_SAMPLED	Byly získány informace o barvě pixelu.
COLOR_RETRIEVED	Byla získána uložená barva.
EDGES_SCANNED	Byl aktualizován seznam detekovaných rohů.
CONNECTED	Připojení k robotickému vozítku proběhlo v pořádku.
DISCONNECTED	Robot byl odpojen.

Tabulka 4.1: Typy informačních zpráv ve výčtu `UpdatedStatus`.

4.1.3 Popis vybraných metod

V níže uvedeném textu budou popsány převážně ty metody, které jsou použity také v ukázkové aplikaci využívající vytvořenou knihovnu.

Ovládání pohybu

Pro ovládání pohybu vpřed slouží primárně `driveForward(int percent, int duration)`, `driveBackwards(int percent, int duration)` a `stop()`. První dvě metody očekávají dva nepovinné parametry, první z nich vyjadřuje, jak rychle se má robot otáčet (možné hodnoty jsou od -100 do 100, v případě `driveForwards(int percent, int duration)` jde u záporných hodnot o otáčení vlevo, u kladných o otáčení vpravo). Maximální rychlost otáčení (které odpovídají hodnotám -100 a 100) je dosažena při protichůdném pohybu pásů. Druhý parametr určuje dobu trvání pohybu v milisekundách. Výchozí hodnota je 0, což značí, že není vložen žádný časový limit a robot se bude pohybovat daným způsobem do doby, než bude zastaven metodou `stop()`.

Metoda `setDriveSpeed(byte speed)` nastavuje rychlost pohybu v rozmezí od 0 do 100. Rychlost přesunu robota na jiné místo je závislá na povrchu a jeho sklonu, parametr `speed` je tedy relativní. Metoda `getDriveSpeed()` zjišťuje poslední nastavenou rychlost pohybu.

Ovládání laserů

Laser je ovládán metodou `setLaserOn(boolean on)`, parametr označuje, zda mají být lasery zapnuty (`true`) nebo vypnuty (`false`). Aktuální stav laserů se zjišťuje metodou `isLaserOn()`.

Zjišťování stavu robota

Pro získání informací o stavu robota slouží několik metod. Stav baterie je zkontrolován na základě metody `requestBatteryStatus()`, výstup je možné číst metodou `isBatteryOk()`, `requestTime()` získá čas od posledního resetu robota, výstup se čte metodou `getTime()`. Metody `requestVersion()` a `getVersion()` slouží k získání verze nainstalovaného firmwaru.

Nastavení kamery

Pro zobrazení obrazových informací z kamery byla vytvořena komponenta `SRV1VideoView`, do které se začne vkládat obsah po zavolání metody `bindVideo(SRV1VideoView video)`.

Samotný obraz je možné různě upravit už před odesláním do mobilního zařízení, například kvalitu obrazu lze nastavit metodou `setCameraQuality(byte quality)`, kde parametr `quality` je ASCII znak v rozmezí 1 až 8. Rozlišení kamery se nastavuje metodou `setResolution(CaptureResolution resolution)`, a dále je možné obraz zrcadlově otočit metodou `setVideoOrientation(boolean flipped)`.

Rozpoznávání obrazu

Jednou z klíčových vlastností robotického robota je, že obsahuje už v jeho základní implementaci příkazy pro rozpoznávání obrazu, které jsou popsány v kapitole 2.3.3. Z těchto příkazů vychází mnoho metod knihovny `SRV1Lib`.

Metody lze rozdělit do dvou skupin, jedny získávají pozici nalezených objektů, a jedny tyto objekty pouze zobrazují. Do té první skupiny patří třeba `requestBlobSearch(byte color_bin)`, `requestEdgeScan(byte columns)` a `requestMeanValues()`, do druhé skupiny patří mimo jiné i metoda `enableVisionFunction(Command enabled_function)`, která povoluje zobrazení detekce rohů, horizontu nebo překážek.

Vložení programu v jazyce C

Knihovna umožňuje vložit do řetězce program v jazyce C, který je bezdrátově odeslán robotickému vozítku, který jej interpretuje pomocí interpretu `PicoC`. Slouží k tomu metoda `insertProgram(String program)`, která očekává jeden parametr, což je daný program. Jakmile je odeslán, je možné zkontrolovat jeho správné zaslání metodami `receiveProgram()` a `getProgram()`. Metoda `executeProgram()` program spustí.

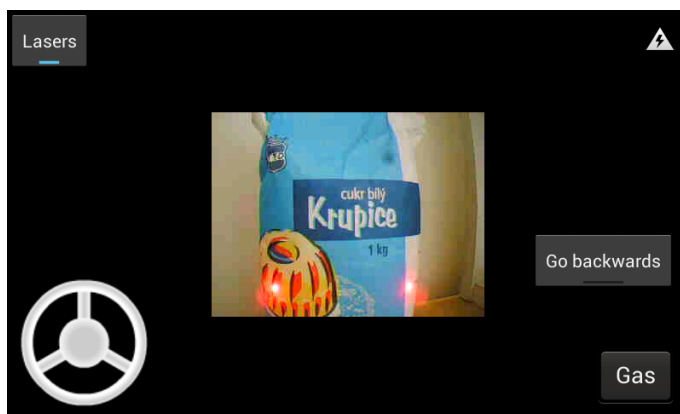
4.2 Ukázková aplikace využívající knihovnu

Aplikace demonstruje využití implementované knihovny. Hlavním přínosem je, že v aplikacích není nutné řešit odesílání konkrétních příkazů a je možné se soustředit na implementaci požadovaného chování.

4.2.1 Popis ovládání

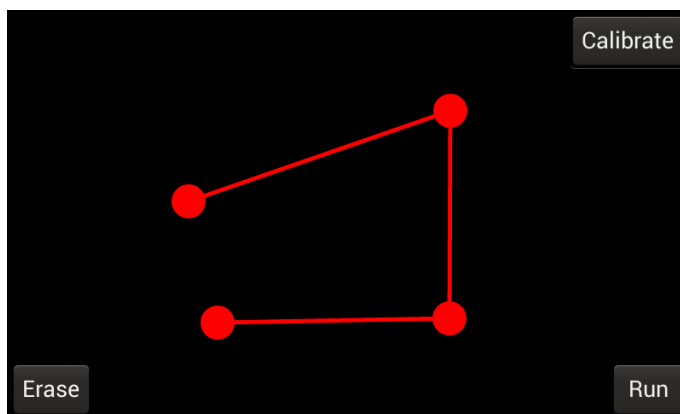
Po spuštění aplikace se zobrazí obrazovka s menu, ze které je možné se buď připojit k SRV-1 a vozítko přímo řídit (položka menu „Drive“), spustit obrazovku pro kreslení dráhy („Draw way“), spustit obrazovku pro hledání objektů dané barvy („Blob search“), nebo přejít do nastavení.

Po vybrání první volby menu a úspěšném připojení se zobrazuje video přenášené z SRV-1, lze pohybovat s robotem a ovládat lasery (viz obrázek 4.3). Směr pohybu se řídí komponentou volantu v levé části obrazovky a přepínacím tlačítkem s nápisem „Go backwards“. V pravé části tlačítko „Gas“ uvádí robota do pohybu. Je-li detekována nízká úroveň nabití baterie, pak se ve vrchní části obrazovky zobrazí obrázek, který o tomto stavu uživatele informuje.



Obrázek 4.3: Obrazovka po připojení k robotickému vozítku.

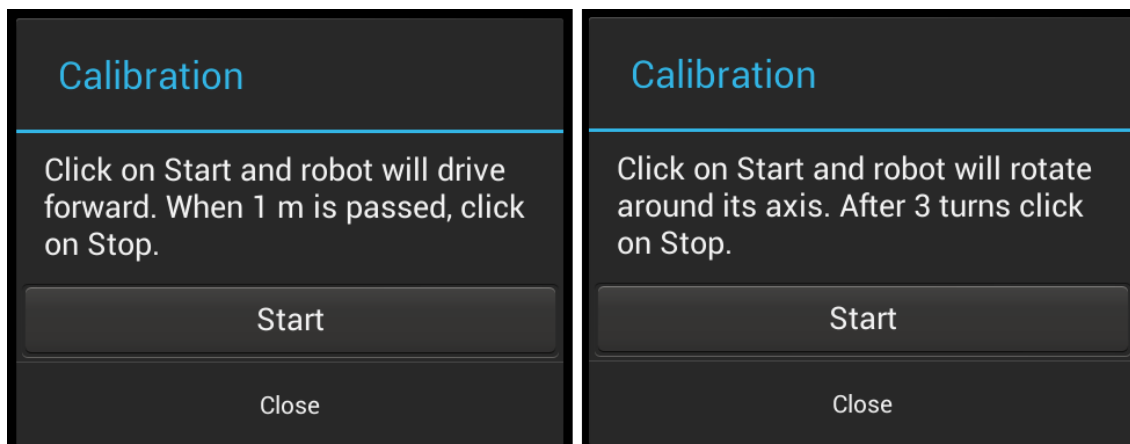
Aplikace byla v pozdější fázi vývoje rozšířena o obrazovku pro kreslení dráhy pohybu robota (viz obrázek 4.4), na které je možné dotykem prstu přidávat body, kterými má robot projet. Po zadání několika bodů a stisknutí tlačítka s nápisem „Run“ robot ujede požadovanou dráhu. Mezi prvním a druhým nakresleným bodem jede robot přímo, až poté se začne otáčet (existuje-li další bod), protože není možné bez přidavných komponent (např. kompasu) sledovat natočení robota vůči mobilnímu zařízení. Tlačítko „Erase“ slouží pro smazání nakreslené dráhy.



Obrázek 4.4: Obrazovka pro kreslení dráhy pohybu.

Tlačítko „Calibrate“ v pravém horním rohu obrazovky slouží pro nastavení, jaký časový úsek je nutný pro to, aby robotické vozítko ujelo dráhu jednoho metru a otočilo se o 360° (popsáno v podkapitole 4.2.2). Po stisknutí tlačítka se zobrazí dialogové okno, které je na obrázku 4.5 vlevo. Stisknutí tlačítka „Start“ uvede vozítko do pohybu. Tlačítko je nahrazeno tlačítkem „Stop“, které robota zastaví, zobrazí informační zprávu („toast“) o naměřené

hodnotě, uloží informaci do nastavení a spustí se další dialogové okno (na obrázku 4.5 vpravo), které funguje obdobně pro zjištění délky pohybu jednoho otočení vozítka.



Obrázek 4.5: Kalibrace.

Poslední implementovanou obrazovkou je obrazovka pro hledání barevných objektů, viz obrázek 4.6. Po dotyku prstem se v pravém horním rohu obrazovky změní barva čtverce na takovou, jaká byla detekována v místě dotyku. Všude tam, kde je v obrazu robotem nalezena skvrna stejné barvy, která je větší, než 5 pixelů, je označena průhledným obdélníkem se žlutým okrajem.



Obrázek 4.6: Obrazovka pro hledání barevných skvrn.

4.2.2 Popis implementace

Při implementaci bylo použito kromě standardních komponent Android API také několik komponent vytvořených speciálně pro tuto aplikaci. Ty jsou popsány v následujících podkapitolách.

Na začátku každé aktivity, ve které má aplikace komunikovat, se aplikace pokusí o připojení k robotickému vozítku. Selže-li tento pokus, je o tom uživatel informován informační zprávou a aplikace se vrátí do hlavního menu. Pokud je aplikace pozastavena (např. je přesunuta na pozadí a je spuštěna aplikace jiná), pak se spojení s robotem dočasně ukončí.

Obrazovka pro přímé ovládání pohybu

Na obrazovce pro přímé ovládání pohybu robota se nachází volant (`WheelView`), který slouží pro pohodlnější určování směru. Třída `WheelView` rozšiřuje standardní komponentu `ImageView` o možnost otáčení obrázku dotykem a o metodu `getValue()`, která vrací celočíselný údaj o natočení. Protože knihovní metody pro pohyb očekávají parametr `percent` jako číslo v rozmezí -100 do 100, byla komponenta volantu vyvinuta tak, aby vracela hodnoty v rozmezí stanoveném atributem `maxValue`.

Pohyb robota je vyřešen tak, že po stisknutí tlačítka s nápisem „Gas“ je zavolána knihovní funkce `driveForward(int percent)` nebo `driveBackwards(int percent)` (záleží, zda je přepínač „Go backwards“ v poloze „zapnuto“ nebo „vypnuto“). Po uvolnění stisku je robot zastaven metodou `stop()`.

Indikátor stavu baterie se dotazuje na stav každou minutu, je-li indikována nízká úroveň nabití, je zobrazen obrázek, který o tomto stavu informuje.

Obrazovka pro kreslení dráhy

Při kreslení se jednotlivé body ukládají do datového typu `ArrayList<CirclePoint>`, což je uspořádaný seznam instancí tříd `CirclePoint`, které obsahují informaci o souřadnicích bodu. Po stisknutí tlačítka „Run“ je do řetězce sestaven program v jazyce C. Řetězec je pak vložen do metody `insertProgram(String program)`, po odeslání je spuštěn metodou `executeProgram()`.

Bylo zamýšleno, že plocha pro kreslení bude vytvořena tak, aby ve všech směrech robot maximálně ujel přibližně 1 m (druhá vzdálenost záleží vždy na rozlišení obrazovky daného zařízení). Vzhledem k tomu, že knihovna ani samotné robotické vozítko neumožňují zadat vzdálenost, kterou robot má urazit, ale pouze čas, po který se má robot pohybovat, bylo nutné zjistit vztah mezi rychlostí robota a dobou, za kterou robot ujede dráhu jednoho metru v přímém směru, a dále dobou, za kterou se robot na místě otočí o 360°.

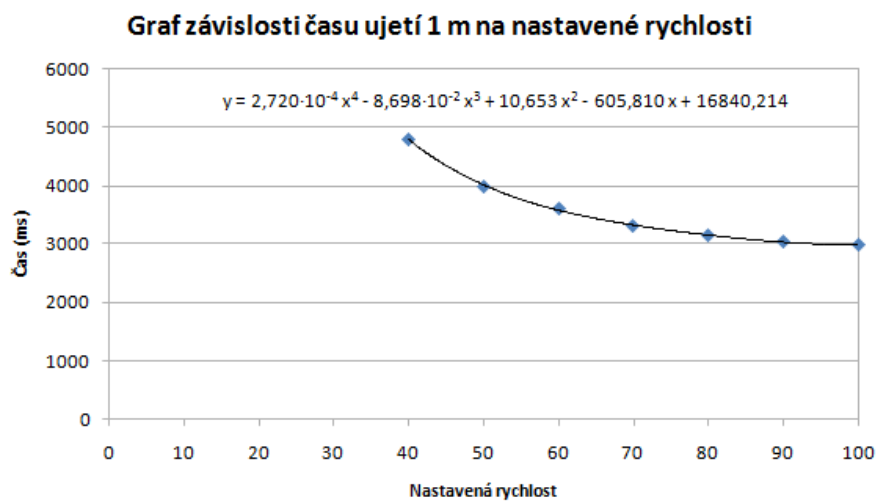
Robot pro pohyb používá pásy s motory, které zapříčiňují to, že na drsném povrchu vozítko potřebuje jinou dobu pro uražení dané vzdálenosti (nebo pro otočení se o daný úhel), než na povrchu, který je hladký. Měření probíhalo na poměrně hladkém povrchu, takže pohyb nemusí přesně kopírovat nakreslenou dráhu, proto bylo vytvořeno tlačítko „Calibrate“, které tento problém řeší.

Pro účely měření byla využita metoda `setDriveSpeed(byte speed)`, která očekává hodnoty od 0 do 100. Robot se při experimentech začal pohybovat v přímém směru až při hodnotě 30, otáčet se začal při hodnotách nad 40. Bylo provedeno několik měření, kdy pro každou uvažovanou hodnotu rychlosti bylo měření opakováno třikrát. Hodnoty měření jsou samozřejmě závislé na povrchu, na kterém Graf na obr. 4.7 zobrazuje průměrné hodnoty času v milisekundách a polynomiální regresi vypočítanou metodou nejmenších čtverců (byl zvolen polynom čtvrtého řádu).

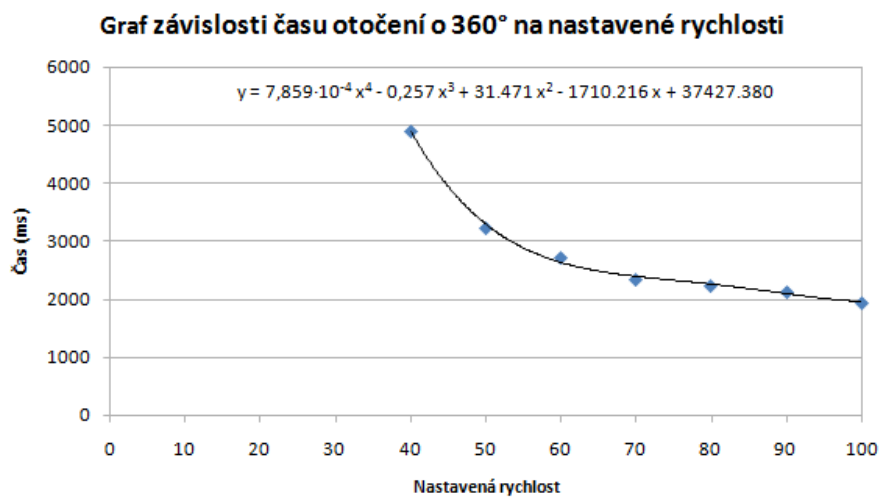
Funkce $y(x)$ je pak použita ve vztahu pro výpočet doby v sekundách (*duration*) potřebné pro ujetí vzdálenosti mezi body označenými na displeji zařízení. Proměnná *pixels* značí počet pixelů mezi body na obrazovce, konstanta `MAX_WIDTH` je maximální šíře obrazovky v jednom směru.

$$duration = \frac{pixels}{MAX_WIDTH} \cdot y(x) \quad (4.1)$$

Podobně byl získán vztah pro výpočet doby takové, aby se robot otočil přibližně o n stupňů.



Obrázek 4.7: Graf závislosti času ujetí 1 m na nastavené rychlosti.



Obrázek 4.8: Graf závislosti času otočení o 360° na nastavené rychlosti.

$$duration = \frac{n}{360} \cdot y(x) \quad (4.2)$$

Metody pro výpočet doby pohybu na základě vztahů uvedených výše byly implementovány pouze do ukázkové aplikace, protože robot může být zvnějšku modifikován a knihovna by pak nebyla univerzální.

Sestavování programu pak probíhá v cyklu, který má dva kroky. Nejdříve je spočítána potřebná doba přímého pohybu mezi dvěma po sobě jdoucími body, a existuje-li další bod, pak je spočítána také doba pohybu potřebná pro natočení robota. Pro pohyb jsou využity funkce `motors(byte left, byte right)` a `delay(byte time)` tak, aby se dohromady chovaly stejně, jako metoda `setDriveSpeed(byte speed)`.

4.2.3 Obrazovka pro hledání barevných objektů

Po dotyku prstem se z plochy, která zobrazuje přenášený obraz z kamery (`SRV1VideoView`) metodou `getFrame()` získá právě zobrazovaný obraz, a z něj je standardní metodou Android API `getPixel(int x, int y)` získána barva pixelu, který je na dané pozici. Tato barva je ve formátu RGBA, metoda knihovny pro komunikaci s vozítkem očekává barvu ve formátu YUV, proto je nutné jí převést (převod popsán v [6]).

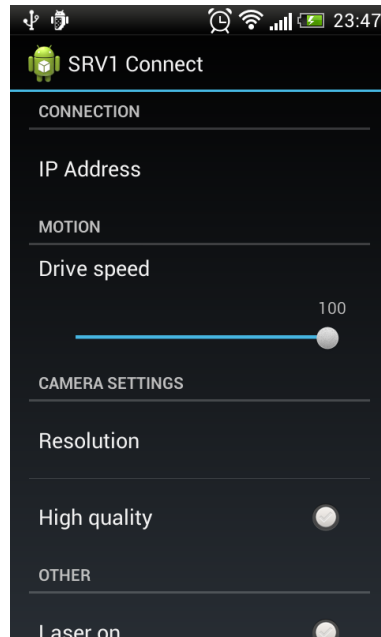
Po převodu je použita metoda `setColor` pro přenesení barvy vozítku. Aby byl pro robota objekt rozpoznatelný, je potřeba přidat jistou toleranci barvy, která byla stanovena tak, že se každá složka barvy (Y, U a V) může lišit maximálně o hodnotu 15 (kladnou i zápornou). Poté jsou skvrny zobrazeny metodou `showBlobs`.

4.2.4 Obrazovka nastavení

Nastavení jsou rozdělena do čtyř kategorií: nastavení spojení, nastavení pohybu, nastavení kamery a ostatní. Lze nastavit tyto parametry:

- IP adresa SRV-1,
- rychlost pohybu,
- rozlišení kamery,
- kvalita obrazu,
- výchozí nastavení laserů po připojení,
- kalibrace (výchozí nebo nastavená).

Pro nastavení je použita standardní komponenta Android API `PreferenceActivity`, díky které stačí jednotlivá nastavení definovat ve speciálním souboru ve formátu XML. Vzhledem k tomu, že Android API neobsahuje vhodnou komponentu pro rychlost pohybu, byla využita komponenta `SeekBar` Kirka Baucoma, která umožňuje nastavit hodnotu v dané škále [4].



Obrázek 4.9: Obrazovka nastavení ukázkové aplikace.

4.3 Zhodnocení a možnosti rozšíření

Vytvořená knihovna splňuje předem dané požadavky a zpřístupňuje většinu funkcí robota. Aplikace byla testována na několika virtuálních zařízeních s Androidem 3.0, 3.1 a 4.0. Nebyly zaznamenány žádné problémy, rozdíl byl pouze ve vzhledu formulářových prvků (což je standardní chování všech aplikací). Některé funkce se do knihovny implementovat nepodařilo, protože buď nebyly během práce k dispozici přídatné komponenty pro SRV-1, aby mohly být metody otestovány, nebo funkce robota nefungovaly korektně. Knihovnu by bylo možné rozšířit o další metody po zapůjčení příslušných komponent, případně vytvořit složitější metody pro rozpoznávání obrazu.

Na základě knihovny byla vytvořena ukázková aplikace, která ukazuje, jakým způsobem je možné knihovnu použít. Aplikaci by bylo možné rozšířit tak, aby využívala více metod knihovny, zejména další metody pro rozpoznávání obrazu. Obrazovka pro kreslení dráhy by mohla po připojení digitálního kompasu (např. HMC6352) k robotickému vozítku zohledňovat polohu robota vůči zařízení.

Kapitola 5

Závěr

Úkolem této práce bylo seznámit se s robotickým vozítkem SRV-1 a popsat možnosti bezdrátové komunikace se zařízením s operačním systémem Android. Po nastudování tvorby Android aplikací a možností robotického vozítka včetně Wi-Fi protokolu byla navržena knihovna, která usnadňuje komunikaci s tímto zařízením a umožňuje vývojáři, který knihovnu použije, se lépe soustředit na algoritmy chování robota místo řešení připojení a odesílání příkazů.

Po implementaci knihovny a popsání jejích metod byla vytvořena ukázková aplikace, která knihovnu využívá, a na základě této aplikace byla ještě dále upravována knihovna pro praktické potřeby programátora. Aplikace demonstruje využití komponenty pro přenos videa a metod pro řízení pohybu.

Literatura

- [1] Activity [online].
URL <http://bit.ly/1n9BMAR>
- [2] Android Architecture - The Key Concepts of Android OS. [online]. Únor 2012.
URL <http://www.android-app-market.com/android-architecture.html>
- [3] Android now powers almost 60 % of mobile devices - about 3 times the amount of iOS devices. [online]. Květen 2013.
URL <http://bit.ly/1kir0Ju>
- [4] Android SeekBar preference v2. [online]. Srpen 2013.
URL <http://bit.ly/1oMm5f7>
- [5] AndroidManifest.xml [online].
URL <http://bit.ly/1msJ804>
- [6] Converting Between YUV and RGB. [online]. Duben 2004.
URL <http://bit.ly/1hGzx0o>
- [7] Dalvik (software) [online]. Březen 2014.
URL <http://bit.ly/1i89T7w>
- [8] Dashboards. [online]. Květen 2014.
URL <http://bit.ly/18oBxX9>
- [9] Definition of the SRV-1 Control Protocol (Blackfin Version). [online]. Duben 2010.
URL http://www.surveyor.com/SRV_protocol.html
- [10] Firmware source code for Surveyor SRV-1 / SVS Blackfin cameras and robots [online]. Únor 2011.
URL <http://bit.ly/1i7QDqF>
- [11] Grant, A.: *Android 4 Průvodce programováním mobilních aplikací*. Computer Press, 2013, ISBN 978-80-251-3782-6.
- [12] ListView [online]. Duben 2014.
URL <http://bit.ly/11TL1AH>
- [13] New Java-based test application for SRV-1 Blackfin [online]. Červen 2008.
URL <http://bit.ly/1j1xMuq>
- [14] Reddy, S.: On Developing a Mesh Network of Robots for Hands-On Undergraduate Education. In *Proceedings of the 39th IEEE international conference on Frontiers in education conference*, IEEE Press, Říjen 2009, str. 1620.

- [15] Restoring SRV-1 Blackfin flash firmware. [online]. 2008.
URL <http://bit.ly/1g95xIT>
- [16] Sales, D.: Vision-Based Autonomous Navigation System Using ANN and FSM Control. In *Robotics Symposium and Intelligent Robotic Meeting (LARS)*, IEEE, Říjen 2010, s. 85–90.
- [17] Settings [online]. Duben 2014.
URL <http://bit.ly/1bNGjVz>
- [18] srv1console [online]. Duben 2014.
URL <https://code.google.com/p/srv1console/>
- [19] Struktura aplikací [online]. Listopad 2012.
URL <http://bit.ly/1nMOBjT>
- [20] Surveyor SRV-1 Blackfin Robot. [online]. Červen 2010.
URL http://www.surveyor.com/SRV_info.html
- [21] Surveyor SRV-1 Blackfin Setup. [online]. Duben 2010.
URL <http://bit.ly/1qvLMXj>
- [22] Surveyor Stereo Vision System. [online]. Prosinec 2009.
URL <http://bit.ly/1j62por>
- [23] Čechmánek, M.: *Robotické vozítko řízené s využitím platformy Arduino*. Diplomová práce, FIT VUT v Brně, 2013.

Dodatek A

Obsah CD

Na přiloženém CD disku jsou uloženy všechny zdrojové soubory, dokumentace k Android knihovně a manuál pro přeložení ukázkové aplikace. Přesný popis adresářové struktury je uveden níže:

- **tex/** Zdrojové L^AT_EX soubory pro sestavení PDF souboru této práce.
- **tex/fig/** Obrázky a fotografie použité v této práci.
- **doc/** Dokumentace knihovny (HTML).
- **pdf/** PDF dokument této práce.
- **manual/** PDF dokument s popisem, jak přeložit a spustit ukázkovou aplikaci.
- **src/** Zdrojové soubory mobilní aplikace.

Dodatek B

Konfigurace robota

Základním předpokladem pro práci s robotem je kromě správného připojení jednotlivých desek také propojení s jiným zařízením přes Wi-Fi a nainstalování posledního firmwaru do Blackfin desky.

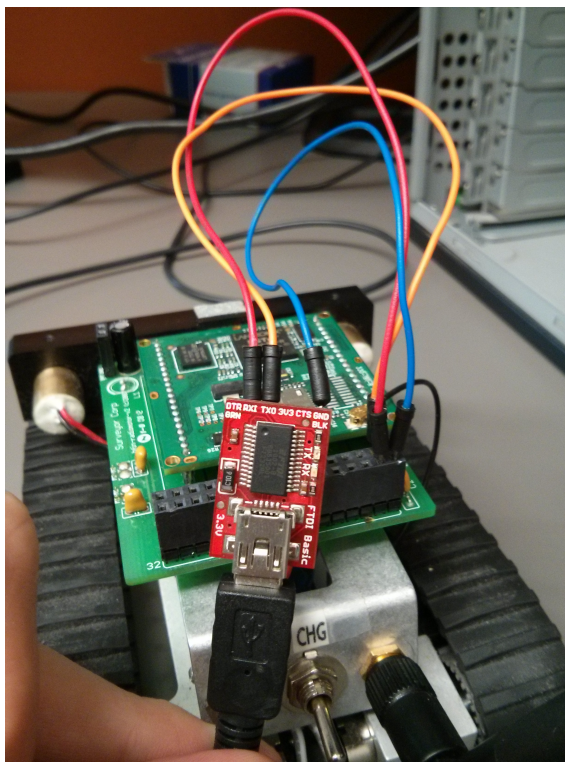
B.1 Nastavení matchportu

Obecně lze tento Matchport nastavit do dvou módů, buď je robot připojen do existující sítě Wi-Fi, nebo vysílá vlastní Ad-Hoc Wi-Fi. Ve výchozím nastavení je zvolena druhá možnost a identifikátor sítě je nastaven na „SRV1“. Pokud je možné se takto připojit, pak lze Matchport dále konfigurovat pomocí telnetu na IP 169.254.0.10 a portu 9999. S připojením ovšem bývají problémy, a to jak při použití operačního systému Windows, tak při použití systému Linux, proto je lepší nastavit matchport do módu Infrastructure, což znamená, že se bude připojovat k jiné Wi-Fi síti. V takovém případě je nutné provést následující kroky.

1. Zvolit volbu 0.
2. Nastavit síť na Wireless Only.
3. Nastavit statickou IP adresu robota (např. 192.168.173.15).
4. Zbylá nastavení přeskočit.
5. Zvolit volbu 4, nastavení WLAN.
6. Nastavit topologii sítě na Infrastructure.
7. Nyní nastavit parametry existující Wi-Fi sítě – SSID, typ zabezpečení, heslo a šifrování. Zbylá nastavení je možné opět přeskočit.
8. Nakonec volbou 9 uložit a ukončit spojení, po restartu bude robot připojen k síti, což je možné ověřit např. programem ping.

B.1.1 Sériové připojení

Pokud není možné se připojit k Ad-Hoc Wi-Fi, nebo robot není správně připojen, je nutné použít zapojení zobrazené na obrázku [B.1](#) a USB připojit k počítači. Poté se dá komunikovat např. přes Putty. Typ spojení je potřeba nastavit na Serial, rychlost spojení na 9600 a název COM portu se dá zjistit v programu Správce zařízení. Po kliku na Open, zapnutí robota a po trojitém rychlém stisknutí klávesy 'x' se dá nastavení měnit tak, jak je popsáno výše.



Obrázek B.1: Ukázka sériového připojení

B.2 Instalace firmwaru

Nepracuje-li firmware správně, nebo je vydána novější verze, je vhodné využít tento postup (převzato z [15] a vyzkoušeno v systému Windows 8.1):

1. Nejdříve je potřeba stáhnout a nainstalovat program ldr, který je dostupný pro Windows i Linux a stáhnout 115kbps verzi SRV-1 firmware.
2. Po připojení k Matchportu a spuštění konfiguračního menu (popsáno v kapitole B.1) se musí změnit přenosová rychlost na 115200. To se provádí přes volbu 1, klávesou enter se přeskočí všechny možnosti nastavení kromě Baudrate, což se nastaví na 115200 a Flow control, což musí být nastaveno na 0. Nakonec se volbou 9 nastavení uloží a ukončí se spojení.
3. Po restartu robota se z vrchní desky odstraní jumper umístěný mezi piny 7 a 8.
4. V příkazovém řádku se přejde do složky, kde je uložen program ldr a spustí se příkaz `ldr -l -v srv1.ldr.recovery 169.254.0.10:10001` (IP adresa se může lišit v závislosti na aktuálním nastavení matchportu).
5. Na chvíli se pomocí jumperu spojí piny 1 a 2.
6. Poté se jumper vrátí mezi piny 7 a 8.
7. Pomocí programu Tera Term se počítač připojí k 169.254.0.10:10001 a po napsání znaku 'V' by měla být vypsána verze právě nahraného firmwaru. Nyní se protokolem

XMODEM přeneše poslední verze firmwaru (k tomu se dá dostat postupným stisknutím kláves Alt, F, T, X, S). Na konci přenosu se ještě do terminálu napíše „zZ“ a potvrdí klávesou Enter. Nyní by mělo být v terminálu vypsáno něco podobného tomuto: „##zZ boot image write count: 131072“.

8. Nakonec se znovu nastaví matchport. Nejdříve se stiskne klávesa '5', zde se nastaví CPU performance na FF, clk na 81, MTU size na 1024. Dále se bude měnit nastavení kanálu 1. Baudrate se nastaví na 1, divisor na 2, flow control na 2, FlushMode na 80, Pack Cntrl na C0, InterCh Time na 3, zbylé volby se přeskočí a volbou 9 se nastavení uloží a ukončí se spojení.

Dodatek C

Metody knihovny

Níže je seznam s anglickým popisem metod knihovny.

Modifier, type and method	Description
<code>void bindVideo(SRV1VideoView video)</code> <code>boolean connected()</code> <code>void disconnect()</code> <code>void driveBackwards()</code> <code>void driveBackwards(int percent)</code> <code>void driveBackwards(int percent, int duration)</code> <code>void driveForward()</code> <code>void driveForward(int percent)</code> <code>void driveForward(int percent, int duration)</code> <code>void enableVisionFunction(Command enabled_function)</code> <code>void executeProgram()</code> <code>BinColor getBinColor()</code> <code>byte getDriveSpeed()</code> <code>ArrayList<java.lang.Integer> getEdgeScan()</code> <code>Handler getHandler()</code> <code>int getLastADRead()</code> <code>ArrayList<Blob> getLastBlobSearch()</code> <code>Color getMeanValues()</code> <code>int getObstacleRange()</code> <code>int getPixelCount()</code>	<p>Connects to SRVs.</p> <p>Finds out if the mobile device is connected to SRV1.</p> <p>Safely disconnects from SRV1.</p> <p>Drive straight backwards.</p> <p>Drive backwards</p> <p>Drive backwards</p> <p>Drive straight forward.</p> <p>Drive forward -180° to 180°</p> <p>Drive forward -100% to 100% for duration*10 ms</p> <p>Enables vision commands (color segmentation, edge detection).</p> <p>Executes program in flash buffer.</p> <p>Gets last bin color retrieved.</p> <p>Gets drive speed</p> <p>Gets last edge scan.</p> <p>Get handler.</p> <p>Gets last read from AD7998.</p> <p>Gets last read of blobs.</p> <p>Gets last computed pixel distribution.</p> <p>Gets range to nearest obstacle.</p> <p>Returns the number of pixels matching color bin #c.</p>

<code>Color getPixelDistribution()</code>	Gets last computed pixel distribution.
<code>String getProgram()</code>	Gets C program saved in flash buffer.
<code>Color getSampledPixel()</code>	Gets last sampled pixel.
<code>int getTime()</code>	Gets time from restart.
<code>String getVersion()</code>	Returns version.
<code>void insertProgram(String program)</code>	Inserts C program to flash buffer.
<code>boolean isBatteryOk()</code>	Returns battery status info.
<code>void receiveProgram()</code>	Receives C program saved in flash buffer.
<code>void requestADRead(byte channel)</code>	Reads specified AD7998 A/D channel.
<code>void requestBatteryStatus()</code>	Refreshes battery status info.
<code>void requestBlobSearch(byte color_bin)</code>	Searches for blobs matching the colors saved by <code>setColor</code> method.
<code>void requestColor(byte color_bin)</code>	Retrieve stored color info from color bin #c.
<code>void requestEdgeScan(byte columns)</code>	Scans for edge pixels in (1-9) columns using <code>edge_thresh</code> set by <code>setThreshold</code> method.
<code>void requestMeanValues()</code>	Computes mean values for Y, U and V over the entire image.
<code>void requestObstacleRange()</code>	Measures range to nearest obstacle using laser pointers.
<code>void requestPixel(int x, int y)</code>	Samples a single pixel defined by coordinates.
<code>void requestPixelCount(byte color_bin, int x1, int x2, int y1, int y2)</code>	Counts the number of pixels matching color bin #c in the range x1, x2, y1 and y2.
<code>void requestPixelDistribution()</code>	Computes and lists the distribution of Y, U and V pixels over the entire range of possible values.
<code>void requestTime()</code>	Outputs time in ms since reset.
<code>void requestVersion()</code>	Refreshes version info.
<code>void reset()</code>	Resets blackfin.
<code>protected void setADRead(int adRead)</code>	Sets last read from AD7998
<code>protected void setBatteryOk(boolean isBatteryOk)</code>	Sets info about battery.

<code>protected void setBinColor(BinColor binColor)</code>	Sets bin color.
<code>protected void setBlobList(ArrayList<Blob> blobList)</code>	Sets last read of blobs.
<code>void setCamera(CameraSettings settings)</code>	Enables/disables automatic gain, white balance and exposure camera functions.
<code>void setCameraQuality(byte quality)</code>	Sets JPEG quality between 1-8.
<code>void setCaptionOverlay(boolean on)</code>	Sets caption overlay.
<code>void setColor(byte color_bin, byte y_min, byte y_max, byte u_min, byte u_max, byte v_min, byte v_max)</code>	Sets color to bin # in SRV1.
<code>void setDriveSpeed(byte speed)</code>	Sets drive speed
<code>protected void setEdgeScan(ArrayList<java.lang.Integer> edgeScan)</code>	Sets last edge scan.
<code>void setFailsafe(boolean failsafe)</code>	Set failsafe to on or off
<code>void setFailsafeLevels(int leftServo, int rightServo)</code>	Sets motor/servo levels in case no command is received via the radio link within 2 seconds.
<code>void setLaserOn(boolean on)</code>	Turns on/off lasers.
<code>protected void setMeanValues(Color meanValues)</code>	Sets pixel distribution
<code>protected void setObstacleRange(int obstacleRange)</code>	Sets range to nearest obstacle.
<code>protected void setPixelCount(int pixelCount)</code>	Sets pixel count.
<code>protected void setPixelDistribution(Color pixelDistribution)</code>	Sets pixel distribution
<code>protected void setProgram(String program)</code>	Saves program received by receiveProgram().
<code>void setResolution(CaptureResolution resolution)</code>	Sets capture resolution.
<code>protected void setSampledPixel(Color sampledPixel)</code>	Sets sampled pixel.
<code>void setServo1Level(int leftServo, int rightServo)</code>	Sets servo 1 level
<code>void setServo2Level(int leftServo, int rightServo)</code>	Sets servo 2 level
<code>void setSVSMODE(SVSMODE mode)</code>	Sets Blackfin to the slave or master status.
<code>protected void setTime(int time)</code>	Sets last read from AD7998
<code>void setTreshold(byte range)</code>	Changes threshold in edge detection.
<code>void setTreshold(byte range, boolean highPrecision)</code>	Changes threshold in edge detection.
<code>protected void setVersion(String version)</code>	Sets info about version.
<code>void setVideoOrientation(boolean flipped)</code>	Flips video orientation.
<code>void showBlobs(byte color_bin)</code>	Graphically overlay blob search results for specified color saved by setColor method.
<code>void stop()</code>	Stops motor.
<code>void stopVisionFunctions()</code>	Stops all vision commands.

<code>void zeroColor(byte color_bin)</code>	Zeros out all or specified color bin.
---	---------------------------------------

Tabulka C.1: Metody knihovny.