

УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Ненад Лазић

ДАЉИНСКА КОНТРОЛА РОБОТА СА ANDROID УРЕЂАЈА

мастер рад

Београд, 2018.

Ментор:

доц. др Милена Вујошевић Јаничић, доцент
Универзитет у Београду, Математички факултет

Чланови комисије:

проф. др Мирдраг Живковић, редовни професор
Универзитет у Београду, Математички факултет

доц. др Весна Маринковић, доцент
Универзитет у Београду, Математички факултет

Датум одбране: 17. септембар 2018.

Захвалница

Велику захвалност дугујем ментору свог мастер рада проф. др Милени Вујошевић Јаничић. Хвала јој - за савете, указано уверење и стрпљење током изrade мастер рада. Њене сугестије су ми пуно значиле и моје неизмерно поштовање за све што је учинила.

Захваљујем се и мојој породици и девојци за безрезервну подршку током студија и у свему што радим.

Наслов мастер рада: Даљинска контрола робота са Android уређаја

Резиме: Роботика је област која је у експанзији и врло брзо ће све више постајати наша свакодневица. У овом раду описан је један начин прављења малог мобилног робота контролисаног Андроид апликацијом путем интернета. Апликација приhvата команде од корисника преко графичког корисничког интерфејса и сензора жiroskopa. Робот се креће помоћу точкова са електромоторима који се контролишу Росбери Pi рачунаром. Комуникација између апликације и Росберија се остварује разменом порука у JSON формату које се шаљу преко ХТТП протокола. Робот је интелигентан и зна да се креће самостално захваљујући имплементираном БУГ алгоритму и информацијама о окружењу које добија преко сензора дистанце. Дат је прецизан опис шеме и безбедносних смерница за правилно повезивања хардверских компоненти на Росбери. У раду су коришћене Андроид и Росбери Pi платформа због своје удобности и могућности које пружају програмерима. Сваки корак је детаљно описан тако да неко, ко се први пут сусреће са овом материјом, након читања овог рада може сам направити свог робота уз мало труда. Направљени робот је у потпуности функционалан и погодан за даље унапређивање и прилагођавање специфичним захтевима. Следећи корак у развоју робота би могла бити имплементација напреднијег алгоритма за самостално кретања робота као што је Тангентни БУГ алгоритам као и сензора за прецизније позиционирање робота у простору.

Кључне речи: роботика, андроид, росбери пи

Садржај

1 Увод	1
1.1 Допринос рада	2
1.2 Организација рада	2
2 О Андроиду	4
2.1 Историјат Андроида	4
2.2 Развојни модел андроида и заједница отвореног кода	7
2.3 Архитектура Андроида	7
2.4 Где је Андроид нашао своје место у овом раду?	12
3 О Росбери Пи	14
3.1 Историјат Росбери Пи платформе	14
3.2 Подешавање окружења за рад и инсталација софтвера	16
3.3 Физичко повезивање Росберија са платформом робота	19
4 Алгоритам заобилажења препреке	25
4.1 Како раде БАГ1 и БАГ2 алгоритми?	26
4.2 Како робот користи БАГ алгоритам?	29
5 Апликација	33
5.1 Имплементација Андроид апликације	33
5.2 Имплементација Пајтон апликације на Росберију	39
6 Закључак	46
Библиографија	49

Глава 1

Увод

Почетак двадесет првог века је обележио интензиван развој тржишта потрошачке електронике због чега су многе технологије и уређаји постали приступачни великом броју крајњих корисника. За интензиван развој овог тржишта били су потребни мали, јефтини али опет моћни рачунари и већи број талентованих и креативних програмера. Један такав моћан рачунар опште намене, доступан по десетоструко нижој цени од стандардног десктоп или лаптоп рачунара, појавио се 2012. године и назван је Росбери Pi (енг. *Raspberry Pi*). Он је рачунар на ком можете урадити скоро све што можете да урадите и на стандардном рачунару и који се уз мало маште лако може претворити у вашег личног робота. Са јефтиним и моћним хардвером постала је могућа реализација пројекта који су до пре десетак година припадали само истраживачким лабораторијама са огромним буџетима.

Највећи утицај на развој тржишта потрошачке електронике имао је развој преносивих уређаја попут паметних телефона, таблета, повезивих и бежичних уређаја. Подједнако важан фактор је био развој оперативних система попут иОС-а (енг. *iOS*), Блекбери-а (енг. *BlackBerry*), Виндоус-а (енг. *Windows*) и Андроид-а (енг. *Android*). Они су променили животе људи доношењем нових начина перцепције информација, комуникације, едукације и забаве. Најзначајнији оперативни системи се могу поделити у две групе. Прву чине оперативни системи затвореног кода (иОС, Блекбери, Виндоус), а другу оперативни системи отвореног кода (Андроид, Линукс). У целом свету данас је убедљиво најзаступљенији Андроид, а с обзиром на растући број корисника делује да ће тако остати још дugo. Погодне лиценце за комерцијалну употребу, отворен код, велика заједница Андроид програмера и компанија које подржава-

ГЛАВА 1. УВОД

вају Гугл (енг. *Google*) у развоју Андроида, су главни фактори тако велике популарности Андроида. Оперативни системи затвореног кода нису погодни за комерцијалну употребу. Наиме, иако нуде комплете за развој апликација, њихове лиценце намећу многа ограничења. Андроид много тога допушта и због тога се данас користи у разноврсним уређајима попут мобилних телефона, камера, сатова, телевизора па чак и аутомобила.

1.1 Допринос рада

У овом раду искоришћене су могућности које пружају Андроид и Росбери Пи платформа за прављење робота. Робот се покреће захваљујући развојној платформи робота у виду мини-возила која у себи има Росбери Пи рачунар. Робот од функционалности има навигацију, детекцију препреке на путу и алгоритам за заobilажење препреке на путу ка циљу. За контролу робота и његово навођење користи се Андроид апликација на мобилном телефону која очитава сензор жироскоп и у зависности од положаја телефона наводи робота у ком правцу да се креће. Робот има сензор удаљености и када детектује препреку испред себе сам се зауставња, а кориснику путем апликације нуди опцију да сам заobiђе препреку или препусти роботу да коришћењем алгоритма покуша сам да је заobiђе. У алгоритму су искоришћене основне идеје БАГ алгоритма али са одређеним изменама које унапређују овај алгоритам. Са овим функционалностима робот је погодан за даље усавршавање и специјализацију, а могућности примене су широке.

1.2 Организација рада

У поглављу 2 биће описани детаљи о Андроиду. У поглављу 3 биће описани детаљи о Росберију, Росбиану¹ (енг. *Raspberryian*) и самој платформи и начину повезивања хардверских компоненти робота. Поглавље 4 ће вас упознати са алгоритмом који се користи за заobilажење препреке у аутоматском моду крећања ка циљу а у поглављу 5 ће бити описана имплементација софтвера за Росбери, алгоритам за заobilажење препреке, протокол комуникације између робота и апликације као и сама архитектура Андроид апликације за навођење

¹Росбиан је званични оперативни систем за Росбери Пи.

ГЛАВА 1. УВОД

робота. На самом крају рада у оквиру поглавља биће изнети закључци, за-
пажања као и проблеми са којима сам се сусрео током израде овог рада.

Глава 2

О Андроиду

Андроид је оперативни систем компаније Гугл (енг. *Google*), отвореног је кода и доступан је под АПАЧИ лиценцом (енг. *Apache Licence*). АПАЧИ лиценца је допуштајућа лиценца (енг. *permissive licenses*) и она дозвољава копирање и даље ширење производа без било какве наплате као и дистрибуцију изведеног софтверског производа без обавезе да се код учини јавно доступним [6, 7].

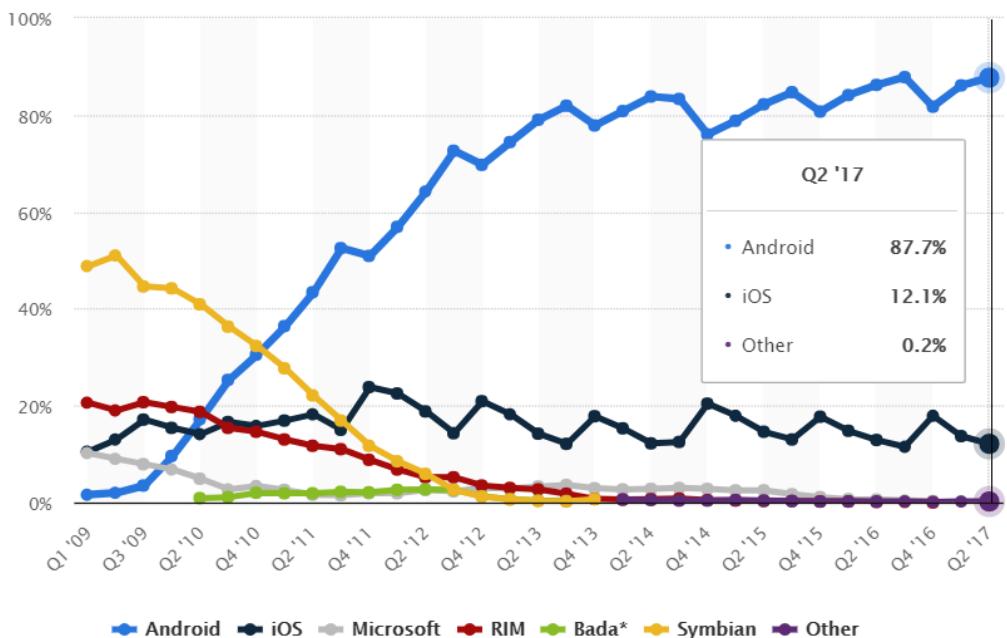
Андроид се састоји од две велике компоненте: Линукс језгра и АОСП пројекта (енг. *AOSP-Android Open Source Project*). Дизајниран је пре свега за мобилне уређаје али се данас може користити и у сатовима, сет-топ бокс уређајима, камерама, аутомобилима и другим уређајима. Број активних Андроид уређаја сваким даном се повећава и у моменту писања овог рада, према статистичким подацима Андроид је заступљен на 87,7% мобилних уређаја у целом свету [8]. На слици 2.1 је приказан график заступљености оперативних система на мобилним уређајима у претходних 8 година.

Постоји велика заједница програмера апликација за Андроид који проширују функционалности, уочавају грешке у АОСП-у и развијају велики број апликација које се могу преузети са Гугл Плеја (енг. *Google Play*). Истраживање спроведено још 2013. године показало је да је Андроид најпопуларнија платформа код 71% програмера [1].

2.1 Историјат Андроида

Почеци развоја Андроида се везују за 2002. годину и компанију Дангер (енг. *Danger*), односно Ендија Рубина, извршног директора ове компаније [5].

ГЛАВА 2. О АНДРОИДУ



Слика 2.1: График раста популарности Андроида у претходних осам година

Компанија Дангер је тада имала иновативан производ, мултифункционални телефон са именом Сајдкик (енг. *Sidekick*), о ком је Енди Рубин држао презентацију на којој су, између осталих, присуствовали и оснивачи компаније Гугл (енг. *Google*) Лари Пејц и Сергеј Брин, за који су се заинтересовали и постали његови корисници. Иако је Сајдкик био иновативан и имао велики потенцијал за даљи развој он ипак није остварио значајан успех а Енди Рубин 2003. године напушта компанију Дангер и оснива компанију Андроид Инц. (енг. *Android Inc.*) која се профилисала ка развоју оперативног система отвореног кода за мобилне уређаје који би се такмичио са већ постојећим Симбијаном (енг. *Symbian*) и Виндоус мобајл (енг. *Windows mobile*).¹ Гугл 17. августа 2005. године купује Андроид Инц. за 50 милиона долара.² Од 2005. до 2007. Гугл је развојни тим, на челу са Ендијем Рубином, интензивно и у тајности ради на развоју Андроида а тек у новембру 2007. прво саопштење о новом оперативном систему издаје група коју чине компаније чија делатност је мобилна телефонија (енг. *OHA-Open Handset Alliance*) попут компанија ХТЦ (енг. *HTC*),

¹Енди Рубин, отац Андроида, се поред оперативних система јако интересивао и за роботику.

²Занимљиво је да је, пре него што је Гугл купио Андроид, Енди Рубин покушао продају Андроида Самсунгу који тада није исказао интересовање.

ГЛАВА 2. О АНДРОИДУ

Кодно име:	Верзија	Датум објављивања	АПИ ниво
N/A	1.0	23. септембар 2008.	1
N/A	1.1	09. фебруар 2009.	2
Cupcake	1.5	27. април 2009.	3
Donut	1.6	15. септембар 2009.	4
Eclair	2.0-2.1	26. октобар 2009.	5-7
Froyo	2.2-2.2.3	20. мај 2010.	8
Gingerbread	2.3-2.3.7	6. децембар 2010.	9-10
Honeycomb	3.0-3.2.6	22. фебруар 2011.	11-13
Ice Cream Sandwich	4.0-4.0.4	18. октобар 2011.	15-15
Jelly Bean	4.1-4.3.1	09. јул 2012.	16-18
KitKat	4.4-4.4.4 4.4W- 4.4W.2	31. октобар 2013.	19-20
Lollipop	5.0-5.1.1	12. новембар 2014.	21-22
Marshmallow	6.0-6.0.1	05. октобар 2015.	23
Nougat	7.0-7.1	22. август 2016.	24-25
Oreo	8.0	08. август 2017.	26

Табела 2.1: Верзије Андроида

Сони (енг. *Sony*), Самсунг (енг. *Samsung*), Ти-мобајл (енг. *T-mobile*), Квалком (енг. *Qualcomm*) и Тексас Инструментс (енг. *Texas Instruments*). Група је основана у сврху развоја отворених стандарда за мобилне уређаје. Име пројекта је АОСП (енг. *AOSP-Android Open Source Project*). У септембру 2008. године објављена је прва верзија Андроида и од тада до данас је објављено петнаест верзија Андроида [10]. Верзије Андроида са датумима објављивања, кодним именом и бројем АПИ-ја (енг. *API- Application Programming Interfaces*) можете видети у табели 2.1.

Верзије 1.x и 2.x су намењене телефонима, верзија 3.x таблетима а од верзије 4.x намењене су и мобилним телефонима и таблетима. Кодна имена се дају по абеџедном реду при чему је то прво слово неке посластице.

2.2 Развојни модел андроида и заједница отвореног кода

Нове верзије Андроида се објављују на приближно шест месеци а новине које долазе са новом верзијом јавности постају познате тек непосредно пред објављивање јер се развој нове верзије Андроида диктира од стране компаније Гугл и практично нико ко није у развојном тиму Гугла не може да утиче на нову верзију Андроида. Разлог за то су Гуглови пословни интереси јер на овај начин Гугл контролише дистрибуцију платформе. Као један од начина да би сваки програмер могао допринети предложеном изменом у развоју нових верзија Андроида настало је Цианоген мод (енг. *CyanogenMod*). Он практично представља паралелни развој система али су произвођачи одбили да подрже фирмвер³ попут Цианоген мод-а због потенцијалних некомпатибилности незваничног софтвера на уређајима. Без обзира на Гуглова, помало необичан модел развоја Андроида, чињеница је да је Гугл са Андроидом успео оно што ни један пројекат отвореног кода пре Андроида није успео а то је да створи најраспрострањенији оперативни систем који мобилним уређајима ставља на располагање моћ и преносивост оперативног система Линукс. Девет година након објављивања прве верзије Андроида он је и даље на узлазној путањи и стално се повећава број и разноврсност уређаја који користе Андроид.

2.3 Архитектура Андроида

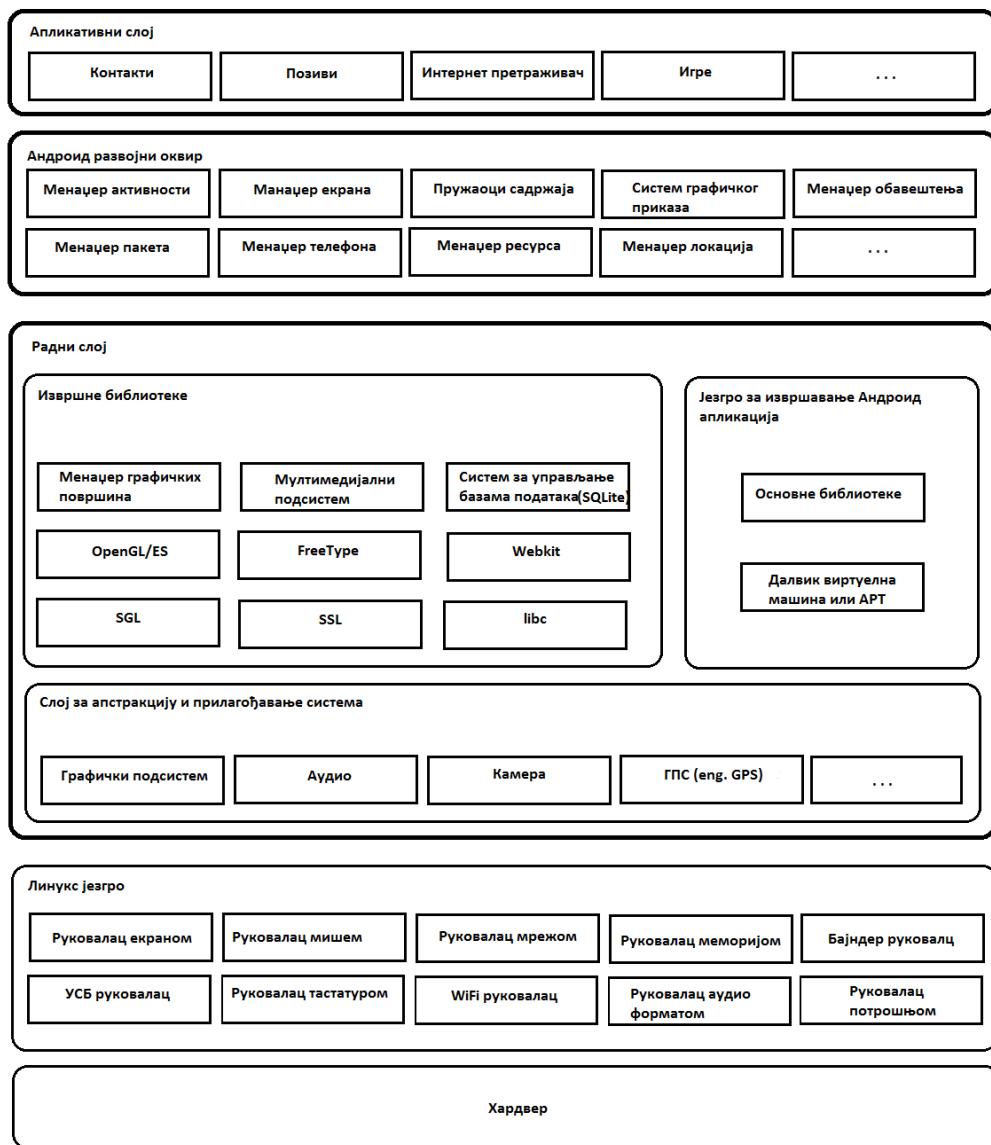
Архитектура Андроид оперативног система је слојевита и састоји се од:

1. Линукс језгра (енг. *Kernel layer*) - које се директно ослања на хардвер
2. Радног слоја (енг. *Runtime layer*) :
 - a) Скупа извршних библиотека (енг. *native*)
 - b) Језгра за извршавање Андроид апликација (*Dalvik, ART*)
 - c) Слоја за апстракцију и прилагођавање система

³Фирмвер је термин који се користи за софтвер смештен на РОМ меморији уређаја и може се заменити флешовањем. Флешовање је краћи назив за поступак који се састоји од прецизно описаног редоследа операција које треба испоштовати за безбедно брисање и поновно писање фирмвера на РОМ меморију.

ГЛАВА 2. О АНДРОИДУ

3. Андроид извршног окружења (енг. *Android Runtime*)
4. Андроид развојног оквира (енг. *Framework layer*)
5. Апликативног слоја (енг. *Application layer*)



Слика 2.2: Архитектура Андроид оперативног система

Сликовит приказ архитектуре Андроида можете видети на слици 2.2.

Линукс језгро

Језгро је део система које је само по себи независна целина и не садржи никакве додатне софтверске алате. Оно пружа основну функционалност оперативног система попут контроле хардверских компоненти и апстракцију свих активних компоненти система у виду процеса, сокета, фајлова итд. Када се систем подиже језгро не мора нужно бити први покренути софтвер али када је једном покренут остаје активан колико и систем. Минимални ресурси потребни за покретање Линукс језгра су:

1. Процесор брзине 100MHz
2. Рам меморија 2MB
3. Ром меморија 4MB

Минимални хардверски захтеви, робусна и стабилна инфраструктура у сталном развоју и отворен изворни код га чине погодним за мобилне уређаје. Важно је нагласити да Андроид не представља дистрибуцију Линукса већ је само „велика апликација“ која се извршава на Линукс језгру. Разлог за то су лиценце и хардверске могућности циљних уређаја. Андроид не поседује неке компоненте које су садржане у Линукс дистрибуцијама попут глибц библиотеке (енг. *glibc*) која је под ГНУ лиценцом (енг. *GNU General Public License*). ГНУ лиценца значи да апликације које користе ову библиотеку морају бити дистрибуиране под истом лиценцом [11]. Други разлог је величина глибц библиотеке па ни због тога није погодна за мобилне уређаје са мање ресурса. Као алтернатива глибцу се користи бионик (енг. *Bionic*) која је флексибилнија по питању лиценце и омогућава дистрибуцију и под затвореним лиценцама.

Линукс језгро је за потребе Андроида проширено апстракцијама попут:

1. Система за руковање системским временом
2. Бајндер (енг. *Binder*) систем за интерпроцесну комуникацију
3. Ешмем (енг. *Ashmem*) систем за дељење меморије
4. Руковаоцима (енг. *Driver*) за УСБ, тастатуру, жироскоп, вибрацију итд.

Радни слој

Радни слој чине слој за апстракцију, скуп извршних библиотека и Андроид виртуелна машина.

Скуп извршних библиотека садржи значајан број библиотека од којих су неке наменски развијене за Андроид. Оне се директно извршавају на циљном уређају (енг. *native code*) и писане су у програмским језицима Џ (енг. *C*) и Џ++ (енг. *C++*). Следи неколико примера ових библиотека и њихов кратак опис.

1. Сурфејс Менаџер (енг. *Surface Manager*) - одговоран да сви прозори буду отворени, из потенцијално различитих апликација, у различито време
2. Опен ГЛ (енг. *Open GL*) и СГЛ (енг. *SGL*) - графичке библиотеке за цртање 2Д и 3Д и могу се комбиновати
3. ОкХТТП (енг. *OkHTTP*), Ретрофит (енг. *Retrofit*), Волеј (енг. *Volley*) - библиотеке за рад са мрежом
4. Веб кит (енг. *WebKit*) - направљен специјално за мале екране и служи за подршку интернет технологијама
5. Ескуел лајт (енг. *SQLite*) - систем за управљање базама података
6. Медија Фрејмворк (енг. *Media Framework*) - кодеци за снимање и репродукцију аудио формата, видео формата и слика

Слој за апстракцију и прилагођавање система служи да пружи униформан програмски интерфејс вишим слојевима са циљем да виши програмски слојеви не морају бринути о различитостима циљних платформи. Дакле сврха овог слоја је да омогући независан развој на вишим нивоима а да се промене услед прилагођавања новим платформама праве само у овом слоју.

Језгро за извршавање Андроид апликација у ствари представља виртуелну машину на којој се извршавају Андроид апликације. Постоје два приступа превођењу апликације у машинске инструкције и извршавања на циљној платформи:

ГЛАВА 2. О АНДРОИДУ

1. Даљвик виртуелна машина - прилагођена је мобилним уређајима и оптимизована за рад са ограниченим меморијом и брзином процесора. Бајт-код добијен превођењем програма написаног у програмском језику Јава Даљвик виртуелна машина оптимизује и преводи у (*.dex) датотеке оптимизоване за извршавање на уређајима са ограниченим ресурсима. Превођење бајт-кода у машински се извршава у рантајму (енг. *JIT Just In Time compilation*). Сваки пут када се покрене нека Андроид апликација прави се нова инстанца Даљвик виртуалне машине на којој се апликација извршава.
2. АРТ (енг. *Android RunTime*) подразумева да се током инсталације апликације бајт код преведе у машински тзв. ЕЛФ (енг. *ELF*) формат који се чува у меморији (енг. *AOT-Ahead Of Time compilation*). Последица је да се са оваквим превођењем добија брже извршавање апликације по цену већег утрошка меморије.

Андроид развојни оквир

Представља слој који апликацијама пружа удобан интерфејс у форми Јава класа. У радном слоју се користе програмски језици C/C++ а у апликацијама Јава тако да је задатак овог слоја да измапира позиве функција у Јави на функције у C/C++. То се ради помоћуJNI механизма (енг. *JNI-Java Native Interface*). На овај начин се сва снага и моћ Линукс језгра и библиотека у радном оквиру у виду АПИ-а пружају на коришћење Андроид апликацијама. Свака верзија Андроида има свој АПИ ниво означен одговарајућим бројем (може видети најаву <https://developer.android.com/index.html>) и који представља јединствени идентификатор скупа функционалности дате верзије Андроида.

Апликативни слој

Налази се на врху Андроид софтверског стека и како само име каже чине га апликације. Андроид долази са скупом основних апликација за електронску пошту, СМС поруке, календар, Веб претраживач, контакте и друге апликације. Корисник може преузету или направити сопствену апликацију и инсталирати на уређају. Апликације које долазе са оперативним системом на уређају немају

ГЛАВА 2. О АНДРОИДУ

посебан статус и скоро свака се може заменити неком другом која има исту функцију⁴.

2.4 Где је Андроид нашао своје место у овом раду?

Видели смо да је Андроид оперативни систем са великим могућностима проширивости и примене у разним областима. Такође смо видели да се користи за разноврсне уређаје попут камера, таблета, аутомобила итд. У овом мастер раду искоришћен је Андроид за контролу робота развијеног на Росбери Пи (енг. *Raspberry Pi*) платформи о којој ће бити речи у наредном поглављу. Искористићемо мобилни телефон са „Marshmallow” верзијом Андроида и API који ова верзија пружа. За очитавање улаза који задаје корисник искористићемо занимљив сензор, жироскоп односно акцелерометар, који поседују скоро сви Андроид телефони новијег датума. Акцелерометар је уређај који се може искористити у разне сврхе попут мерења убрзања, пређеног пута, навођење пројектила итд. Рад овог уређаја се заснива на мерењу сила инерције које делују на уређај у току кретања. У средишту кућишта сензора се налази „велика” и „тешка” маса, величине делића милиметра, разапета између неколико опруга са ограниченим могућношћу љуљања у простору. Та маса са статичним деловима кућишта формира кондензаторе⁵ чији капацитети се мењају у зависности од положаја те масе на коју, током померања уређаја, делују сile инерције. Очитавање вредности сензора се заправо своди на очитавање вредности капацитивности тих кондензатора. Жироскоп служи за мерење промене угла и може се реализовати коришћењем претходно поменутог сензора.

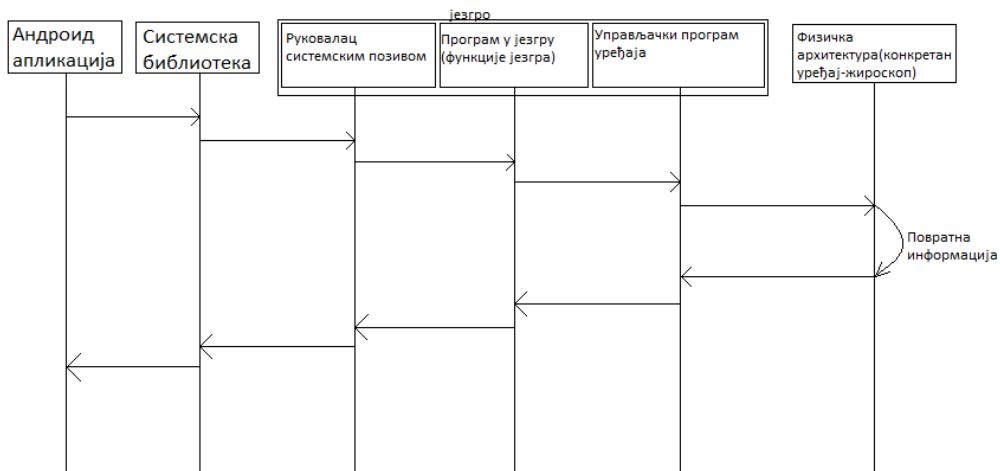
Апликација за навођење робота користи управљачки програм из већ поменутог Линукс језгра како би комуницирала са жироскопом који је хардверски уређај. Адресна магистрала омогућава процесору да приступа уређајима који су повезани на њега и те адресе које су доступне процесору се називају Физичке јер представљају адресе стварних физичких компоненти. Физичка адресна мапа представља стварну локацију сваке компоненте повезане са процесором и њу одређује произвођач уређаја приликом дизајна штампане плоче

⁴Постоје изузети попут апликације за подешавање система која не може бити замењена.

⁵Кондензатор је компонента који може да сачува енергију у облику електричног поља између две електроде раздвојене изолатором.

ГЛАВА 2. О АНДРОИДУ

и повезивања процесора са компонентама. Поред физичког адресног простора постоји и виртуелни у ком се апликација извршава и омогућава да апликације међусобно имају изоловане адресне просторе а језгром има свој физички адресни простор путем кога комуницира са хардверским компонентама. Током извршавања процеса, виртуелни адресе се пресликавају у стварне, физичке адресе. Један пример је РАМ меморија (енг. *RAM-random access memory*). Сваки процес има свој део меморије (виртуални адресни простор) који се захваљујући јединици за управљање меморијом (енг. *MMU-Memory Management Unit*) преводи у физичке адресе на самој хардверској компоненти. Графички приказ пропагације позива од Андроид апликације до самог жироскопа као хардверске компоненте можете видети на слици 2.3.



Слика 2.3: Пропагација позива од апликације до очитавања параметара жироскопа

Апликација користи одговарајућу системску библиотеку и позива њене функције које врше обраду и системске позиве преко руковаоца системским позивима који даље врши обраде и позива потребне функције језгра. Језгром позива управљачки програм задужен за сензор жироскоп који очитава његов статус и враћа га назад до апликације. На овај начин апликација добија сазнање у ком положају се уређај налази на основу три параметра који се враћају као резултат очитавања. Ти подаци са жироскопа се користе за навођење робота у ком правцу да се креће о ком ће бити више речи у наредним поглавњима.

Глава 3

О Росбери Pi

Росбери Pi (енг. *Raspberry Pi*) је рачунар који се може најкраће описати са три речи: мали, моћан и јефтин. Ове три речи су уједно и разлог његове велике популарности и примене у различитим областима Интернет Ствари (енг. *IoT Internet of Things*), Паметна кућа (енг. *SmartHouse*), едукације и другим. На тржишту се појавио у фебруару 2012. године, а настао је у истоименој фондацији.

Име Росбери Pi потиче из старе традиције да се компанијама дају имена по воћу. Постоји доста примера попут: енг. *Apricot Computers* - име добијено по кајсији, енг. *Apple* - име добијено по јабуци, енг. *Acorn* - име добијено по једној врсти ораха, енг. *BlackBerry* - име добијено по купини. Росбери Pi је тако добио име по малини, а део Pi потиче од скраћенице **Пајтон интерпретер** (енг. **Python interpreter**).

3.1 Историјат Росбери Pi платформе

Идеја за настанак Росбери Pi платформе се родила почетком прве деценије 21. века на Кембриџу (енг. *University of Cambridge*) због тенденције смањивања броја кандидата заинтересованих за студије Информатике. Још један, можда и важнији, мотив за ову идеју на Кембриџу су видели и у слабом предзнању програмирања оних који се одлуче да студирају. Разлог за то су видели у непостајању платформе доволно занимљиве а опет једноставне и јефтине да заинтригира младе људе и подстакне их на учење и сопствени развој а самим тим и обезбеди боље предзнање будућих студената.

На Кембриџу су до овог закључка дошли кроз анализу претходних генера-

ГЛАВА 3. О РОСБЕРИ ПИ

ција које су имале више практичног програмерског знања али и лако програмабилне кућне рачунаре (попут енг. *ZX Spectrum, Commodore 64, Amiga*) које је било потребно програмирати за било какву персонализацију и озбиљнији рад на њима. Генерације са почетка 2000-их су имале лажан привид знања о рачунарским наукама поистовећујући га са корисничким истукством. Због тога се на почетку 21. века свет суочава са недостатком квалификованых ИТ стручњака. Потражња за квалитетним ИТ стручњацима у свету је велика а чак и већа потражња је вероватна због повећања нашег апетита и очекивања од технологије.

У настојању да се ситуација преокрене Ебен Уpton (енг. *Eben Upton* тадашњи члан Кембриџ рачунарског одељења), Роб Мулинс (енг. *Rob Mullins*), Џек Ланг (енг. *Jack Lang*), Алан Мајкрофт (енг. *Alan Mycroft*) са Кембриџа, Пит Ломас (енг. *Pete Lomas*) и Дејвид Бребен (енг. *David Braben*) су се удружили 2006. и основали фондацију Росбери Pi [4]. Ова организација је као примарни циљ имала креирање јефтиног, програмабилног рачунара и његово пласирање у што већи број руку. Као резултат заједничког рада, 29. фебруара 2012. је настао Росбери, уређај који нас враћа на рачунарске основе са циљем промоције рачунарства и програмирања код младих. За 6 месеци од момента када се први Росбери појавио продато је 500000 примерака овог рачунара који је поседовао 32-битни Бродкомов¹ чип BCM2835 (енг. *Broadcom*) са АРМ архитектуром (енг. *ARM-Advanced RISC Machines*), брзином од 700MHz и 256MB меморије [4].

Од тада до данас се појавило више генерација Росбери рачунара а њихове разлике се огледају пре свега у брзини процесора, количини меморије, броја ГПИО (енг. *GPIO General-purpose Input/Output*) пинова који служе за повезивање и контролисање додатних хардверских компоненти. Прва генерација Росберија је изашла у фебруару 2012. са основним моделом А и моделом Б који је имао више ресурса. Модели А+ и Б+ су се појавили годину дана касније а Росбери 2 модел Б у фебруару 2015. Најновији модел Росбери 3 модел Б који је нама и најзанимљивији, јер је рад урађен управо на овом моделу, појавио се у фебруару 2016. године. У време писања овог рада цена овог рачунара у Србији је око 6000 динара. Хардверске карактеристике овог рачунара можете видети у табели 3.1.

¹Бродком је један од највећих производа чипова у свету.

ГЛАВА 3. О РОСБЕРИ ПИ

Табела 3.1: Росбери 3 модел Б конфигурација

Процесор	1.2GHz 64-bit quad-core ARM Cortex-A53 32kb cache L1 512kb L2
Графички процесор	Broadcom VideoCore IV, ОпенGL ES 2.0, 1080p30 h.264/MPEG-4 AVC BluRay quality playback
РАМ меморија	1GB DDR2
Складиште података	MicroSD
Мрежа	Ethernet RJ45
WiFi	802.11 b/g/n
Bluetooth	4.1 Low Energy
Видео излази	3.5mm конектор
Аудио излази	3.5mm конектор
HDMI	Full HDMI port
USB	4 port
GPIO	40
Камера интерфејс (CSI)	да
Екран интерфејс (DSI)	да
Напајање	2.5A @ 5V
Произведено	UK

3.2 Подешавање окружења за рад и инсталација софтвера

За почетак рада са Росберијем поред самог Росберија је потребно напајање и SD картица. Напајање треба да има прецизно регулисан напон од 5V са једносмерном струјом од 1A до 2,5A. Није могуће користити USB везу са рачунара за напајање Росберија јер оно даје максимално 500mA. Уколико нема додатних компоненти повезаних на Росбери и ако није оптерећен он ће можда радити и на 500mA али ће се с времена на време изненадно ресетовати што може довести до уништавања USB порта на рачунару уколико није заштићен од преоптерећења. Због овога је важно да Росбери има посебно напајање од 5V и струјом од 1A. С обзиром да Росбери нема хард диск али има слот за SD картицу потребна је и SD картица. На картици се налази оперативни систем са драјверима и простор за смештање фајлова током рада Росберија. Минимални капацитет картице је 4GB.

За рад са Росберијем је довољна интернет конекција и рачунар са ког се он

ГЛАВА 3. О РОСБЕРИ ПИ

може контролисати коришћењем SSH² протокола. Монитор, тастатура и миш ипак у многоме могу олакшати посао приликом неких подешавања Росберија.

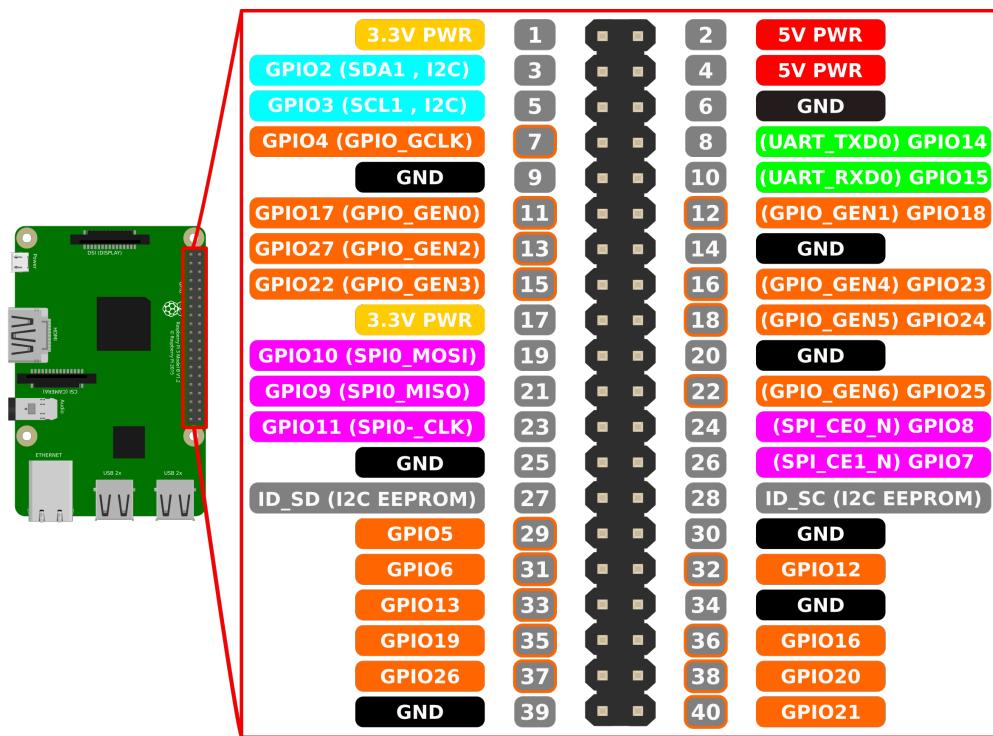
У случају коришћења монитора најбоље је користити монитор са HDMI улазом јер Росбери има HDMI излаз. Иако је могуће користити и монитор са VGA улазом помоћу HDMI на VGA конвертера потребно је бити пажљив са одабиром конвертера. Конвертер треба да има сопствено напајање јер би у супротном чип за конверзију узимао напајање са Росберија, што би после неког времена направило штету и у најбољем случају прегорела би само заштитна диода а у горем могла би настати штета на самом чипу. Када се користи монитор потребно је прво укључити монитор и сачекати поруку „No HDMI signal“ и тек онда покренути Росбери. Што се тиче тастатуре и миша не постоје неки специјални захтеви. Најбоље је одабрати уређаје са каблом и USB конектором како би се избегли потенцијални проблеми са драјверима у случају бежичних уређаја.

Због заштите самог уређаја приликом повезивања додатних компоненти путем ГПИО пинова потребно је нагласити да ГПИО пинови, на које се повезују компоненте које примају сигнал од Росберија, као излаз дају 3,3V што је ниво логичке јединице. Они могу безбедно провести струју до 16mA па је због тога потребно бити опрезан приликом повезивања додатних компоненти како не би дошло до прегоревања Росберија. Приликом додавања додатних хардверских компоненти потребно је пажљиво прорачунати коло и ако је потребно додати отпорнике одговарајуће отпорности, везане у редној вези са компонентом, како би се Росбери заштитио од прегоревања. За прорачунавање кола довољно је искористити Омов закон који каже да је струја која пролази кроз проводник изменју две тачке директно пропорционална напону на истим тачкама тог проводника а обрнуто пропорционална његовом електричном отпору

$$I = V/R$$

Након прорачуна кола битно је бити пажљив и на који пин на Росберију поvezуете то коло јер немају сви пинови исту сврху. На слици 3.1 је дат приказ распореда ГПИО пинова за Росбери Пи 3 и њихов назив.

²SSH-Secure Socket Shell је мрежни протокол који обезбеђује сигурну комуникацију са удаљеним рачунаром.



Слика 3.1: Називи и распоред ГПИО пинова на Росбери Пи 3

Инсталација оперативног система

Росбиан (енг. *Raspbian*) је званични подржани оперативни систем фондације. Може се инсталирати помоћу апликације за инсталацију оперативног система NOOBS или преузимањем слике (енг. *image*) и инсталирањем на картицу. За први сусрет корисника са Росберијем препоручује се коришћење NOOBS-а. NOOBS је једноставан софтвер за инсталацију оперативног система који се може преузети са сајта организације. Све што је потребно је SD картица, читач SD картице и преузета архива NOOBS-а са сајта³. Картицу је потребно форматирати и преузету архиву екстрактовати а затим ископирати на SD картицу. Након тога је довољно убацити SD картицу у Росбери и повезати га са монитором, тастатуром, мишем и укључити напајање након чега ће се Росбери бутовати и NOOBS инсталер ће вам понудити опције за инсталацију неколико оперативних система прилагођених за Росбери. Након одабраног жељеног оперативног система (у мастер раду је коришћен Росбиан оперативни систем) и

³Уколико немате читач SD картице могуће је са Гуглове продавнице (енг. *Google Play Store*) преузети бесплатну апликацију (енг. *Raspi Card Imager*) на Андроид уређај који поседује SD читач картица и помоћу ове апликације инсталирати NOOBS на картицу.

ГЛАВА 3. О РОСБЕРИ ПИ

пар кликова добија се SD картица са инсталираним оперативним системом.

Алтернативни начин, који је и бржи, је преузимањем већ спремљене слике оперативног система са сајта организације. Следи алгоритам за инсталацију оперативног система путем већ спремљене слике оперативног система:

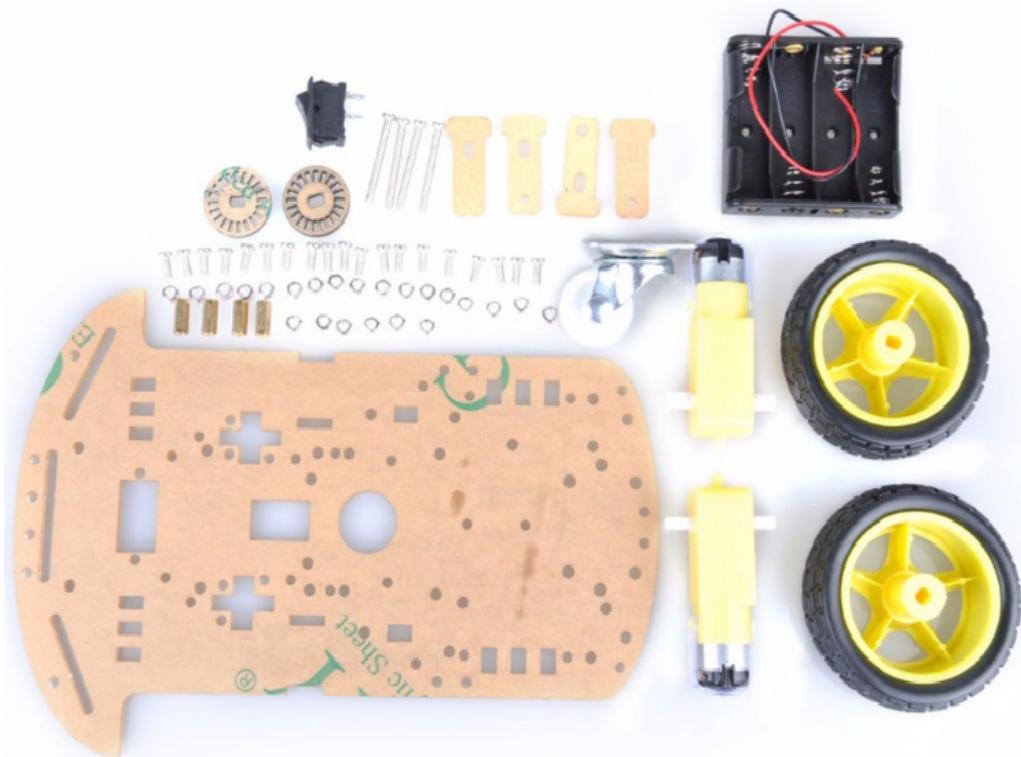
1. Са сајта орханизације преузети архивирану слику оперативног система
<https://www.raspberrypi.org/downloads/raspbian/>
2. Екстрактовати преузету архиву
3. Преузети и инсталирати Etcher <https://etcher.io/> - софтвер за снимање слике оперативног система на SD картицу или USB
4. Покренути Etcher и одабрати екстрактовану архиву (слику оперативног система) и одабрати SD картицу на коју желите да је сачувате
5. Убацити SD картицу у Росбери и повезати га са монитором, тастатуром, мишем и укључити напајање након чега ће се Росбери бутовати

Када се Росбери бутује нису неопходна никаква додатна подешавања да би почели са радом. Све што је потребно је да се повежете на бежичну везу (енг. *WiFi*) и забава са овим сјајним рачунаром може да почне.

3.3 Физичко повезивање Росберија са платформом робота

Платформа робота коју ће контролисати Росбери је димензија $218mm \times 110mm$. Креће се помоћу три точкића при чему су два погонска и имају своје моторе, по један за сваки од погонских точкова и једног помоћног точка који служи за одржавање равнотеже робота. Платформа је поручена из Кине путем сајта Али Експрес (енг. *AliExpress*) и њена цена је око 1300 динара. У овом комплету добијате шасију возила, два мотора, носач за батерије (4 алкалне батерије од по 1,5 V), прекидач за напајање, два точкића са гумама димензија 6,5cm x 2,7cm и један мањи ротирајући точкић. Изглед платформе можете видети на слици 3.2.

Мотори су димензија $65mm \times 18.5mm \times 22mm$ и сваки од мотора је директно повезан са својим точком. У питању су ДЦ електромотори који служе



Слика 3.2: Платформа робота коју контролише Росбери

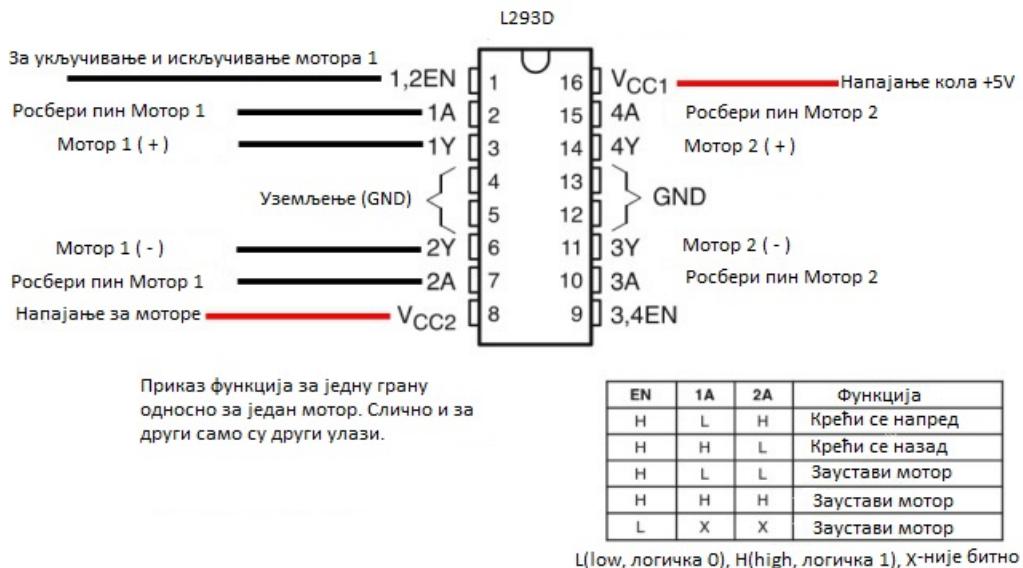
да претворе електричну енергију у механичку и углавном раде на принципу електромагнетне индукције. Ови мотори се побуђују једносмерном струјом, углавном се праве за мање снаге и предност им је лакше управљање брзином, моментом и смером обртања. Радни напон им је од 3-12 V (препоручени радни напон 6-8 V) а струја оптерећења 70mA (до максималних 250mA).

Од хардвера је још потребно посебно напајање за Росбери⁴, развојна плоча за лакше повезивање компоненти, проводници, електрично коло L293D које је заправо мотор контролер и сензор удаљености који ће служити за аутоматско препознавање препреке на путу у режиму аутономног кретања робота ка неком циљу. За напајање је искоришћена преносива батерија за мобилни телефон, капацитета 4400mA, напоном од 5V и струјом од 1,5A. Чип L293D је веома користан чип и он је практично посредник у комуникацији између Росберија и самих мотора. Овај чип може контролисати два мотора независно. Неопходан

⁴Потребно је да Росбери има одвојено напајање како не би долазило до пада напона када се мотори укључе јер би иначе ометали рад Росберија и чак повремено, у случају већег пада напона, и до искључивања уређаја.

ГЛАВА 3. О РОСБЕРИ ПИ

је како би се избегло да Росберијем директно контролишемо моторе јер мотори могу да повуку већу струју и изазову прегоревање самог Росберија. Овај чип дакле контролише струју и напон за напајање мотора како би Росбери могао неометано да функционише. На овај чип се доводи напајање а онда са њега води на моторе. Излази са Росберијевих ГПИО пинова се доводе као логички улази овог кола и од њихових вредности зависи када је који мотор укључен, у ком смеру се окреће и којом брзином односно када коло проводи или не проводи струју и на тај начин напаја моторе. Назив и кратак опис сваког пина за коло L293D се налази у табели 3.2.



Слика 3.3: Мотор контролер L293D

Као сензор удаљености искоришћено је коло HC-SR04 за прецизно мерење растојања користећи ултразвучно мерење. Димензије кола су $16\text{mm} \times 50\text{mm}$ и може да мери удаљеност на раздаљини од 2cm до 450cm са малом грешком од 0.3cm. За што прецизније мерење битно је сензор поставити тако да сензор гледа хоризонтално са углом у односу на хоризонталу не већим од 15 степени. Радни напон је 5V а радна струја 15mA. Коло има 4 пина. Један служи за напајање, један за уземљење, један је за окидање (слanje ултразвучног сигнала) и један пин је ехо (повратна информација). Сензор ради тако што еmitује ултразвучне импулсе⁵ и мери време од тог тренутка до тренутка када добије

⁵Звучни импулси фреквенције изнад горње границе чујности за нормално људско ухо, преко 20kHz.

ГЛАВА 3. О РОСБЕРИ ПИ

Табела 3.2: Називи и опис функција пинова на чипу L293D

Пин	Опис његове функције	Назив
1	Укључивање контроле над мотором 1 (Мотор реагује на сигнале за смер кретања са пинова 1 2)	Enable 1,2
2	Улаз 1 за мотор 1	Input1
3	Излаз 1 за мотор 1	Output1
4	Уземљење (0V)	Ground
5	Уземљење (0V)	Ground
6	Излаз 2 за мотор 1	Output2
7	Улаз 2 за мотор 1	Input2
8	Напон напајања за моторе 9-12V (до максималних 36V)	Vcc2
9	Укључивање контроле над мотором 2 (Мотор реагује на сигнале за смер кретања са пинова 3 и 4)	Enable 3,4
10	Улаз 1 за мотор 2	Input3
11	Излаз 1 за мотор 2	Output3
12	Уземљење (0V)	Ground
13	Уземљење (0V)	Ground
14	Излаз 2 за мотор 2	Output4
15	Улаз 2 за мотор 2	Input4
16	Напон напајања за чип 5V (до максималних 36V)	Vcc1

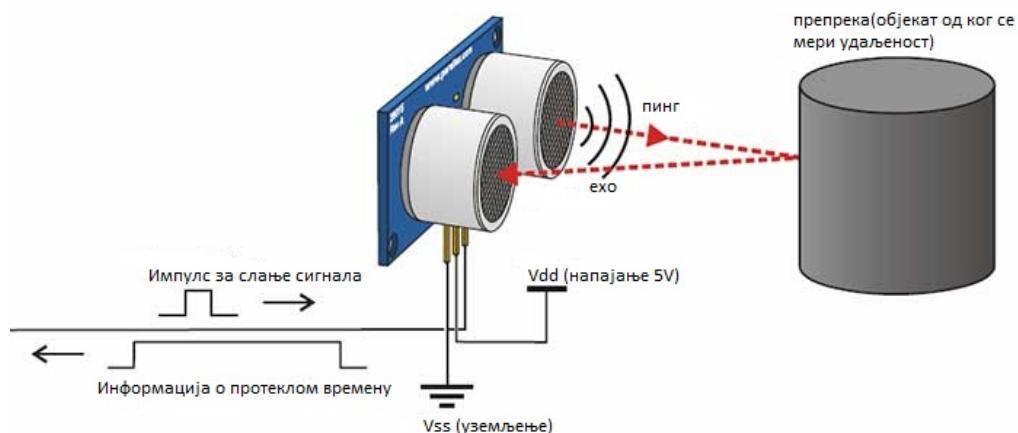
повратну информацију односно до тренутка када сензор прими импулсе који су се одбили од потенцијално постојеће препреке. Формула за рачунање растојања до препреке (уколико постоји) је:

$$D = V * t$$

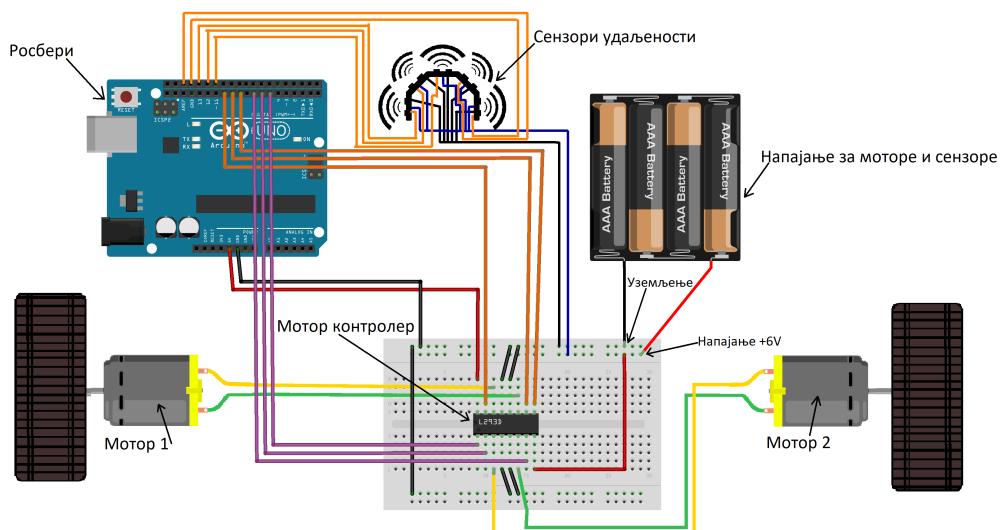
При чему је D - растојање, V - брзина ултразвучног сигнала кроз ваздух и t- протекло време од слања ултразвучног сигнала са сензора до детектовања његовог еха на сензору. Изглед овог кола (сензора) и илустрацију начина рада можете видети на слици 3.4.

Све горе описане компоненте (Росбери, развојну платформу робота, моторе, сензоре, чип L293D, батерије) је потребно повезати у једну целину која ће чинити робота. Шему за повезивање ових компоненти можете видети на слици 3.5.

На робота је постављено три сензора удаљености тако да је један постављен у правцу кретања робота и по један лево и десно под угловима од -90 и +90 степени у односу на правац кретања робота. Користи се више сензора због бољег решавања колизија у режиму аутономног кретања робота.



Слика 3.4: Принцип рада и изглед сензора удаљености



Слика 3.5: Шема за повезивање Росберија са платформом и сензорима

Шема илуструје оно што је до сада објашњено и дат један од могућих начина за повезивање Росберија са платформом робота и сензорима онако како је то урађено на роботу. Сада када је потребан хардвер повезан у функционалну целину долази ред на софтверски део и имплементацију која ће се састојати из три битне целине:

- Андроид апликације за телефон - служи као кориснички интерфејс за руковање роботом
- Пајтон (енг. *Python*) апликације за Росбери - служи да управља роботом

ГЛАВА 3. О РОСБЕРИ ПИ

односно да прима наредбе од корисника (Андроид апликације) и да их преводи у одговарајуће реакције хардвера

- Имплементација алгоритма за аутономно кретање - служи да робота учини „интелигентним”

У наредна два поглавља је дат опис алгоритма, његов начин рада и опис имплементације а након тога и комплетан систем са апликацијама и алгоритмом.

Глава 4

Алгоритам Заобилажења препреке

Поред тога што се робот може наводити ручно, помоћу Андроид апликације, лепота и сврха постојања роботике се огледа управо у томе да робот може сам да обавља неки задатак без помоћи човека. Савремени роботи решавају реалне проблеме у динамичном и отвореном окружењу. Због тога се појављују нови изазови у областима алгоритама контроле робота и планирања покрета. Ови изазови се јављају услед повећане потребе за аутономијом и флексибилношћу робота. Један од основних задатака роботике је „научити” робота да се самостално креће без асистирања човека. У овом поглављу биће описан БАГ (енг. *BUG*) алгоритам који се бави проблемом планирања путање кретања робота. Након тога биће речи о томе како је овај алгоритам адаптиран за потребе овог рада.

БАГ алгоритам служи за планирање путање робота од стартне до циљне тачке у дводимензионалном простору без информација о изгледу мале терена¹ и могућим препрекама. С обзиром да се роботи не морају користити искључиво у фабрикама, кућама или негде другде где постоји усталјена мапа терена, овај проблем кретања робота у „непознато” је утолико занимљивији и шире применљив. БАГ алгоритам је један од првих алгоритама базираних на контактним сензорима или сензорима дистанце и гарантује проналажење решења у коначном броју корака уколико је циљ достижен.

Најбитније три верзије овог алгоритма: БАГ1, БАГ2 и Тангентни БАГ алгоритам (енг. *Tangent BUG algorithm*)². БАГ1 и БАГ2 алгоритми претпостављају

¹Под мапом терена се мисли на слику или поглед из ваздуха на област кроз коју се робот креће и чијом анализом се могу применити неки други и ефикаснији алгоритми за проналажење оптималне путање.

²Постоји још неколико верзија које користе основне идеје БАГ алгоритма са додатком

ГЛАВА 4. АЛГОРИТАМ ЗА ОБИЛАЖЕЊА ПРЕПРЕКЕ

поседовање контактних сензора док Тангентни БАГ алгоритам претпоставља поседовање сензора који могу прецизно мерити дистанцу до неког објекта у свих 360 степени. Без обзира на врсту сензора, све три верзије карактеришу две особине: кретање ка циљу правом линијом и заобилажење препреке праћењем њене контуре [12].

Постоје и други напреднији алгоритми за овај проблем али сваки од тих напреднијих алгоритама почива на претпоставкама попут поседовања:

- Снажних сензора (домет сензора је велики)
- Доста рачунарске снаге за сложена израчунавања
- Доста меморије за чување мапе
- Прецизних GPRS уређаја
- Слике из ваздуха (са висине која може да посматра већи простор)

У наставку рада ће бити демонстрирано да није неопходна скупа опрема и снажни рачунари за аутономно кретање робота. Довољан је један јефтин сензор дистанце и Росбери на ком ће се извршавати БАГ алгоритам да би се овај проблем решио.

4.1 Како раде БАГ1 и БАГ2 алгоритми?

За самостално кретање робота у овом раду су искоришћене основне идеје БАГ1 и БАГ2 алгоритма. Прецизније, БАГ2 алгоритам је модификација БАГ1 алгоритма и он је прилагођен за потребе овог рада. Пре самог описа алгоритама битно је нагласити да су они засновани на одређеним претпоставкама. Неке од њих у реалним применама могу бити проблематичне. Претпоставке које се користе у алгоритму су:

- Робот је тачка са савршеним позиционирањем (кретање односно положај робота се може пратити савршено прецизно)
- Робот може отворити препреку уколико је додирне
- Робот може прецизно мерити растојање између било које две тачке $d(x, y)$

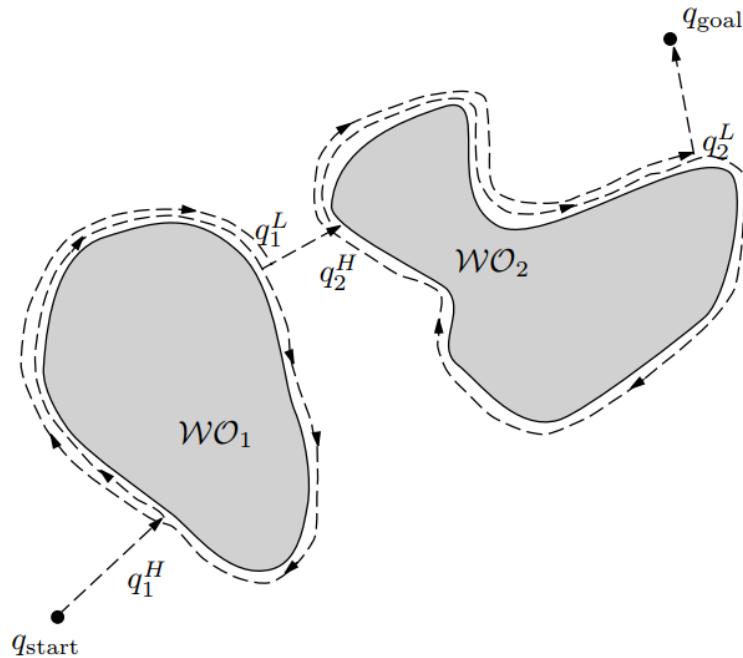
разних хеуристика за проналажење што бољег решења у задатим условима.

- Простор W у ком се робот креће је ограничен

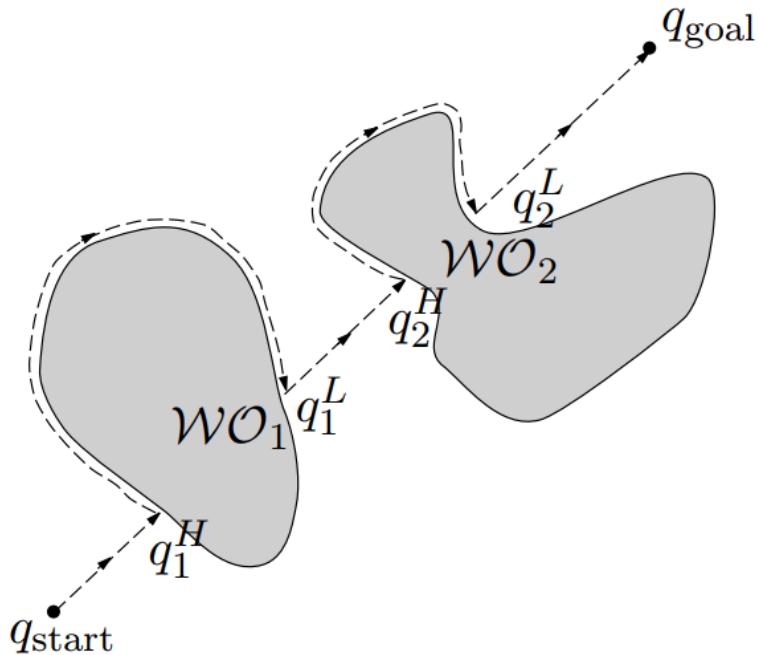
Тачку са које робот креће означићемо са q_{start} а тачку која представља циљ, који робот треба да достигне, означићемо са q_{goal} . У алгоритму се користе појмови тачке поготка, тачке напуштања и m -линије. Тачка поготка је тачка у којој је детектована колизија са другим објектом током кретања робота ка циљу. То је тачка у којој се робот сударио са препреком и означава се са q_i^H . Индекс i означава редни број препреке са којом је робот дошао у колизију током кретања а ознаке H (енг. *Hit*) и L (енг. *Leave*) говоре о томе да ли је тачка са индексом i тачка поготка или тачка напуштања. Тачка напуштања је тачка у којој се робот одваја од контуре препреке након њеног заобилажења и означава се са q_i^L . M -линија, у алгоритму БАГ1, је линија која спаја тачке q_i^L и q_{goal} . Она се користи током праволинијског кретања робота ка циљу.

На почетку алгоритма се узима да је тачка $q_0^L = q_{start}$ и бројач препрека $i = 0$. Робот одређује m -линију од тачке q_0^L до q_{goal} и започиње кретање по њој ка циљу. Уколико дође до циља алгоритам се завршава јер је решење пронађено. Уколико је током кретања m -линијом дошло до колизије (робот је детектовао препреку на свом путу) робот увећава бројач препрека $i = i + 1$ и тачку у којој је дошло до колизије означава са q_i^H и памти њене координате. Затим кружжи око препреке, пратећи њене контуре, све док се не врати у тачку q_i^H . Тада има доволјно знања о препреци да може да одреди тачку на контури препреке која је најближа циљу (тачки q_{goal}). Та тачка је тачка напуштања за i -ту препреку и биће означена са q_i^L . Робот се креће до тачке q_i^L и ажурира m -линију и наставља кретање. Уколико новодобијена линија пресецда тренутну препреку закључак је да не постоји пут до циља односно робот се налази унутар затворене препреке [12]. Илустрацију начина рада алгоритма БАГ1 можете видети на слици 4.1.

БАГ2 алгоритам представља модификацију алгоритма БАГ1. Основне разлике су то што се m -линија код БАГ2 алгоритма не мења током извршавања алгоритма. Она је фиксна и повезује тачке q_{start} и q_{goal} . Још једна разлика је у томе што када робот нађе на препреку (тачка поготка) не обилази целу њену контуру. У тачки поготка се одлучује са које стране ће обилазити препреку и започиње праћење контуре препреке све док се поново не нађе на m -линији (тачка напуштања). У тачки напуштања се одваја од препреке и наставља кретање ка циљу по m -линији. Илустрацију начина рада алгоритма БАГ2 можете видети на слици 4.2. Ако се робот два пута нађе у истој тачки поготка q_i^H тада закључује да не постоји пут до циља јер се налази у затвореној препреци (нема



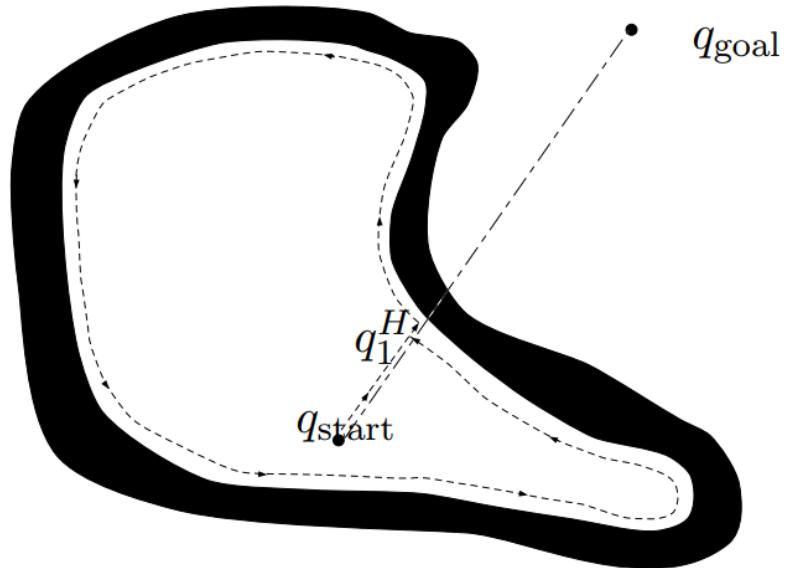
Слика 4.1: Начин рада алгоритма БАГ1



Слика 4.2: Начин рада алгоритма БАГ2

излаз из затворене препреке). Овај случај можете видети на слици 4.3.

Ова два алгоритма илуструју два основна приступа за проблеме претра-



Слика 4.3: Детекција случаја када не постоји пут до циља

живљања. Алгоритам БАГ1 проналази оптималну тачку напуштања за сваку препреку на коју нађе али по цену да ради иссрпну претрагу. Алгоритам БАГ2 користи похлепни приступ јер гледа само локалну добит односно бира прву опцију која му у том тренутку обећава проналажење решења.

4.2 Како робот користи БАГ алгоритам?

Током навођења робота помоћу Андрид апликације могу се десити колизије. Робот је „паметан” и неће дозволити човеку да га води у „опасност”. Када детектује препреку на путу којим га води човек робот се сам зауставља. Он нуди човеку могућност да га човек води другачијим путем или да му препусти да се сам снађе и превазиђе проблем. Уколико човек препусти контролу роботу након обилажења препреке, уколико је то могуће, робот враћа контролу човеку. Имплементиран алгоритм се у великој мери ослања на алгоритам БАГ2 јер користи његову идеју приликом заobilажења препреке. Дакле, не ради иссрпну претрагу као БАГ1 већ користи похлепни приступ па се може рећи да је сличнији БАГ2 алгоритму. Зависно од примене робота и један и други алгоритам се могу применити. У овом раду се не проналази оптимално решење већ оно које ће у реалном раду робота дати брже решење односно оно које ће робота брже

ГЛАВА 4. АЛГОРИТАМ ЗА ОБИЛАЖЕЊА ПРЕПРЕКЕ

довести до циља. Коришћење БАГ1 алгоритма има смисла када робот треба да научи неки пут и да га после поново користи али када је основни циљ стизање до неке тачке у „непознатом” оправданије је користити БАГ2 алгоритам.

Дакле, у основи алгоритма који користи робот за аутономно кретање се налази БАГ2 алгоритам. Овом алгоритму је, за разлику од БАГ2 који користи само контактни сензор, на располагање дат сензор дистанце или тачније неколико сензора дистанце. Ови сензори могу да детектују препреку на неколико метара. Захваљујући њима робот, у тренутку када му се препусти контрола да сам заобиђе препреку, не бира баш на потпуно случајан начин страну са које ће заобићи препреку већ је мало анализира помоћу сензора. Тачније робот може да „погледа” лево и десно пре него крене и да процени где му је отворенији пут. Више сензора је постављено под различитим угловима у односу на правац кретања из разлога да робот не мора да физички прави заокрете како би направио ово процену. У наставку следи опис како је алгоритам практично имплементиран.

За рад алгоритма је потребно успоставити неки координатни систем у ком се робот налази. Правац у ком је усмерен робот одређује y осу координатног система а нормала на тај правац, кроз тачку у којој се налази робот, одређује x осу. Смер у ком робот „гледа” одређује позитивни део y осе, иза робота је негативни део y осе, десно позитивни део x осе а лево негативан део x осе. Овај координатни систем се успоставља на почетку рада алгоритма и до краја се не мења али ће робот током кретања мењати своје координате у овом систему.

Робот током кретања, помоћу сензора постављеног са предње стране, не-престано проверава да ли му се на путу налази препрека. Када детектује да му је препрека на путу то јавља човеку (апликацији за навођење) и нуди опцију да га води другим путем или да му препусти да заобиђе препреку и да му након тога врати контролу. Када му човек преда контролу он рачуна t - линију³ и помоћу сензора дистанце постављених под угловима од -90 , -45 , $+45$ и $+90$ степени, проверава да ли му је у неком смеру слободан пут. Ако јесте, критеријуми за одабир новог правца кретања v робота су:

- што мањи угао између t - линије и правца v
- непостојање или што веће растојање до нове колизије у правцу v у дometу сензора дистанце

³ t - линија ће заправо бити y оса координатног система који се поставља у том тренутку.

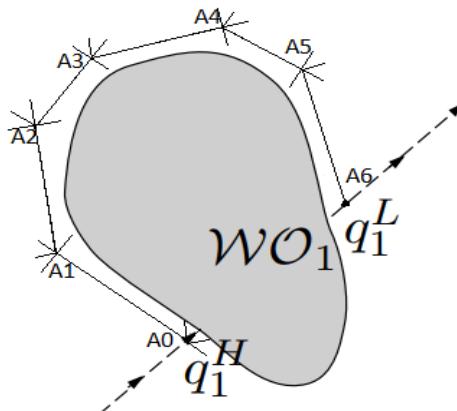
ГЛАВА 4. АЛГОРИТАМ ЗАОБИЛАЖЕЊА ПРЕПРЕКЕ

- оба сензора дистанце (парови $(-90, -45)$ и $(+45, +90)$) са стране робота на којој се налази правац v сигнализирају слободан пут

Уколико није добио ни један позитиван одговор, робот на случајан начин бира страну и врши постепен заокрет у одабрану страну радећи исте провере све док не добије позитиван одговор са неког од сензора. На основу добијеног позитивног одговора, или више њих, и положаја сензора са кога је добијен тај одговор робот бира најбољи по горе наведеним критеријумима.

Када је нови правац кретања робота одабран потребно је сачувати информацију о томе да ли кретањем у одабраном правцу v робот иде лево или десно у односу на t -линију тј да ли обилажење препреке иде са леве или са десне стране. Ова информација је битна за праћење контуре препреке.

Робот започиње кретање по одабраном правцу непрестано проверавајући сензор са предње стране и два сензора десно ако препреку обилази са леве стране или два сензора лево ако препреку обилази са десне стране.



Слика 4.4: Начин рада БАГ2 алгоритма на роботу

За сензоре са бочне стране робота (под угловима од 45 и 90 степени) у дањем тексту користи се ознака S . Робот се креће у одабрном правцу све док неки од сензора из S не пријави слободан пут и тада робот поново мења правац. Робот мери пређени пут од претходне тачке промене правца A_i и бележи нове координате у којој је променио правац. На тај начин се прати контура препреке а обилажење је готово када се робот поново нађе на t линији односно када му је координата поново постане 0. Илустрацију начина на који робот заobilази препреку можете видети на слици 4.4.

ГЛАВА 4. АЛГОРИТАМ ЗА ОБИЛАЖЕЊА ПРЕПРЕКЕ

Када је робот обишао препреку наставља да се креће напред све док човек поново не преуме контролу.

Глава 5

Апликација

Софтверско решење које се користи за навођење и контролу робота се за-
снива на клијент-сервер архитектури и састоји се од:

- Андроид апликације
- Пајтон сервера

Клијентску страну представља Андроид апликација која се извршава на
мобилном телефону или таблету. Служи да улазне информације добијене од
корисника проследи Пајтон серверу и да повратну информацију од робота при-
каже кориснику.

Серверску страну представља Пајтон сервер на Росберију и састоји се од
три битна модула:

- Модул за комуникацију са апликацијом
- Модул за контролу платформе робота
- Модул за аутономно кретање

Следи детаљнији опис сваког дела софтвера.

5.1 Имплементација Андроид апликације

Андроид апликације могу бити писане у програмским језицима Котлин (енг. *Kotlin*),
Јава (енг. *Java*) или Ц++ (енг. *C++*). За навођење робота је направљена Ан-
дроид апликација коришћењем програмског језика Јава и развојног окружења

ГЛАВА 5. АПЛИКАЦИЈА

СДК (енг. *SDK Software Development Kit*) за Андроид. Апликације направљене на овај начин се називају и нативне (енг. *native*) апликације и програмерима дају потпуну слободу и приступ могућностима и сензорима уређаја.

Основни градивни елементи сваке Андроид апликације су следеће компоненте:

- Активности (енг. *Activity*)
- Сервиси (енг. *Services*)
- Пружаоци садржаја (енг. *Content providers*)
- Пријемници емитованих обавештења (енг. *Broadcast receivers*)

Активности су компоненте чија је сврха приказивање корисничког интерфејса и интеракција са корисником. Већина апликација се састоји из једне или више активности чији интерфејс се обично дели у фрагменте који представљају модуларне делове активности. Апликација не мора нужно имати активности уколико је реч о апликацији која ради у позадини и није видљива кориснику. Апликације које користе активности их имплементирају као поткласе класе „*Activity*”.

Сервиси су компоненте које немају кориснички интерфејс и служе за обављање послова у позадини апликације. Сервиси имају свој API и пружају услуге активностима или другим сервисима. Користе се углавном за дуготрајне операције попут комуникације преко мреже или за одржавања апликације у стању извршавања иако корисник можда тренутно користи неку другу апликацију. У апликацијама се имплементирају као поткласа класе „*Services*”.

Пружалац садржаја је компонента која управља скупом података апликације који се чувају у фајл систему, бази података, на вебу или било којој другој локацији којој апликација има приступ. Друге апликације могу да читају и мењају те податке уколико им то пружалац садржаја дозволи. У апликацијама се имплементира као поткласа класе „*ContentProvider*” и мора да имплементира скуп API-а који ће друге апликације подразумевати.

Пријемник емитованих обавештења је компонента апликације која омогућава да се укључи „хватање” догађаја (енг. *event*) на систему. Другим речима, ова компонента апликације каже оперативном систему да га обавести када се одређени догађај појави. Апликација не мора бити у стању извршавања да

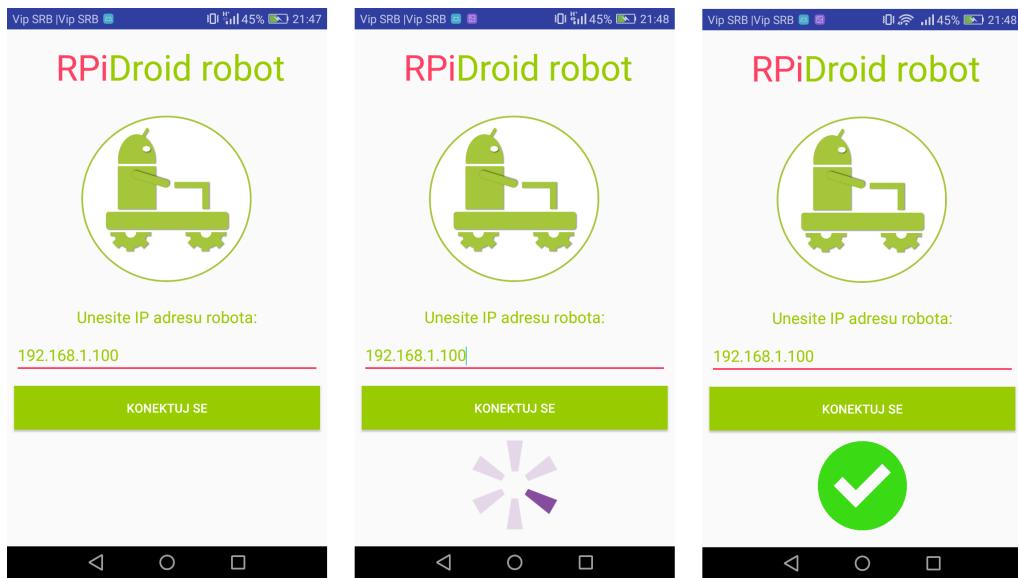
ГЛАВА 5. АПЛИКАЦИЈА

би одговорила на догађај већ се њено покретање може иницирати догађајем. Имплементира се као поткласа класе „BroadcastReceiver”.

Још једна неизоставна ствар коју свака Андроид апликацији мора имати је манифест фајл (енг. *AndroidManifest.xml*). Све компоненте од којих се апликација састоји морају бити наведене у овом фајлу. Такође је потребно навести и дозволе које су потребне апликацији за исправан рад. То може бити приступ камери, именику, жироскопу или било шта друго што постоји на уређају а није подразумевани ресурс апликације. Обавезно је и навођење минималног и циљаног АПИ-а односно верзије Андроида за коју је апликација написана.

Апликација за навођење састоји се од две активности и једног сервиса. Због јаснијег описа прво ће бити приказан и описан кориснички интерфејс и случајеви употребе па тек онда технички детаљи имплементације.

Прва активност служи за унос ИП адресе робота и успостављање везе са роботом. Изглед корисничког интерфејса ове активности можете видети на слици 5.1.



Слика 5.1: Успостављање везе између апликације за навођење и робота

Када робот одговори да је спреман да прихвата наредбе, веза је успостављена и покреће се друга активност која служи као интерфејс за управљање роботом. Она је централна компонента у апликацији. Из ње се очитава сензор жироскопа коришћењем системских сервиса намењених за то и врши комуникација са роботом коришћењем сервиса направљеног за потребе апликације.

ГЛАВА 5. АПЛИКАЦИЈА

Ова активност има неколико ствари које приказује. Једна од информација је стања сензора жироскопа у виду стрелица које показују на коју страну је нагнут уређај на ком се извршава апликација. Ове стрелице означавају правац у ком наводимо робота да се креће. У овој активности се налазе и два контролна дугмета. Једно служи за укључивање аутономног режима кретања а друго је команда којом се прекида сваки посао и робот се зауставља. Аутономни режим подразумева кретање робота право напред а уколико нађе на препреку сам треба да се снађе и заобиђе је. Из аутономног режима се излази притиском на дугме за заустављање. Изглед корисничком интерфејса активности за навођење робота можете видети на слици 5.2.



Слика 5.2: Кориснички интерфејс активности за навођење робота

Ова активност такође приказује одговоре робота на корисникove команде односно на команде за навођење које му шаље апликација. Тако ће робот у одговору, уколико нађе на препреку, јавити кориснику ту информацију и сачекати нову команду. Уколико нема препреке он ће извршити задату команду односно наставиће кретање у задатом смеру и јавити то кориснику. У зависности од одговора робота, ова активност ће променом корисничком интерфејса и доступних опција јасно ставити до знања који су дозвољени кораци које корисник може предузети. Изглед интерфејса можете видети на слици 5.3.

Поред поменутих активности апликација има и један сервис. Он апликацији омогућава комуникацију са роботом преко интернета слањем порука у складу са успостављеним протоколом комуникације. Овај сервис ради у позадини (енг. *background process*) и активан је док год се апликација налази у меморији. За свако слање поруке роботу покреће нову нит (енг. *thread*) из ког коришћењем ХТТП (енг. *HTTP - Hypertext Transfer Protocol*) протокола шаље ПОСТ (енг. *POST*) захтев серверу (роботу). Слање ХТТП захтева се ради из

ГЛАВА 5. АПЛИКАЦИЈА



Слика 5.3: Обавештења робота која се приказују у апликацији

нити због тога што операције са мрежом могу трајати дugo а то може довести до кочења и рушења апликације. У телу захтева се у JCOH (енг. *JSON - JavaScript Object Notation*) формату преносе информације релевантне за сервер. Следи један пример поруке која се шаље као команда да робот скрене у десно и добијеног одговора за успешно извршену команду:

```
1 {  
2   "message_type": "RPi_MSG_DIRECTION",  
3   "value": "GO_RIGHT"  
4 }
```

Код 5.1: Пример захтева који се шаље роботу за скретање у десно

```
1 {  
2   "message_type": "RPi_MSG_DIRECTION",  
3   "success": "YES"  
4 }
```

Код 5.2: Пример одговора који се добија од робота за успешно извршену команду

Скуп порука које апликација може послати роботу и које може добити као одговор од робота је дефинисан протоколом комуникације. Прво поље у поруци одређује тип поруке а оно одређује начин на који се интерпретира вредност из другог поља поруке (уколико оно постоји за тај тип поруке). Дозвољене вредности за тип поруке и њихово значење су:

- RPi_MSG_DIRECTION - слање команде у ком правцу да се креће робот, вредности у другом пољу означава правац кретања у односу на тренутни положај:

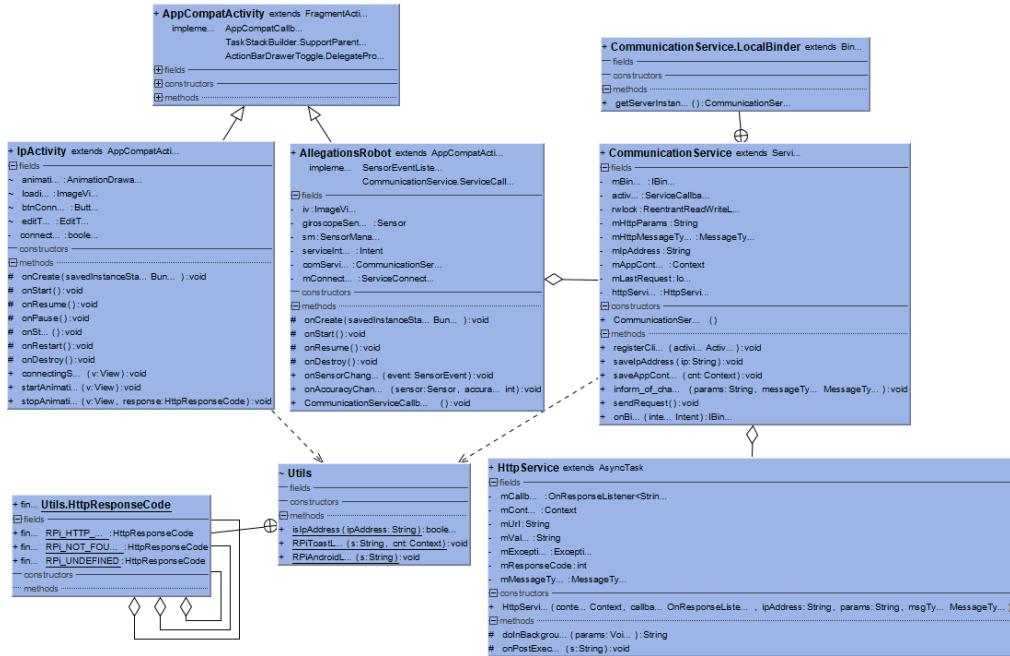
ГЛАВА 5. АПЛИКАЦИЈА

- 1 - напред
- 2 - напред полулево
- 3 - напред полудесно
- 4 - лево
- 5 - десно
- 6 - назад
- 7 - назад полулево
- 8 - назад полудесно
- 0 - заустави се

- RPi_MSG_CONNECTING - успостављање комуникације са роботом и његова иницијализација (прва порука у комуникацији)
- RPi_MSG_DISCONNECTING - прекидање комуникације са роботом и његова деиницијализација (задња порука у комуникацији)
- RPi_MSG_AUTO_MODE - укључивање и искључивање аутономног режима кретања
 - 0 - укључи аутономни режим кретања
 - 1 - искључи аутономни режим кретања
- RPi_MSG_RESPONSE_OK - одговор од робота, команда извршена успешно
- RPi_MSG_RESPONSE_FAILURE - одговор од робота, команда није извршена успешно
- RPi_MSG_RESPONSE_OBSTACLE - одговор од робота, нашао на препреку
- RPi_MSG_UNDEFINED - грешка у комуникацији

На слици 5.6 приказан је дијаграм класа апликације. Овај дијаграм приказује структуру апликације и интеракције међу компонентама апликације.

ГЛАВА 5. АПЛИКАЦИЈА



Слика 5.4: Дијаграм класа Андроид апликације

5.2 Имплементација Пајтон апликације на Росберију

Софтвер који се извршава на Росберију и намењен је контроли платформе робота се може посматрати као апликација која се састоји из три модула са јасно подељеним одговорностима. Ови модули међусобно комуницирају преко API-а која сваки модул нуди осталим модулима. Сликовит приказ модула и њихове међусобне комуникације можете видети на слици 5.5. Следи детаљнији опис сваког модула појединачно.

Модул за комуникацију

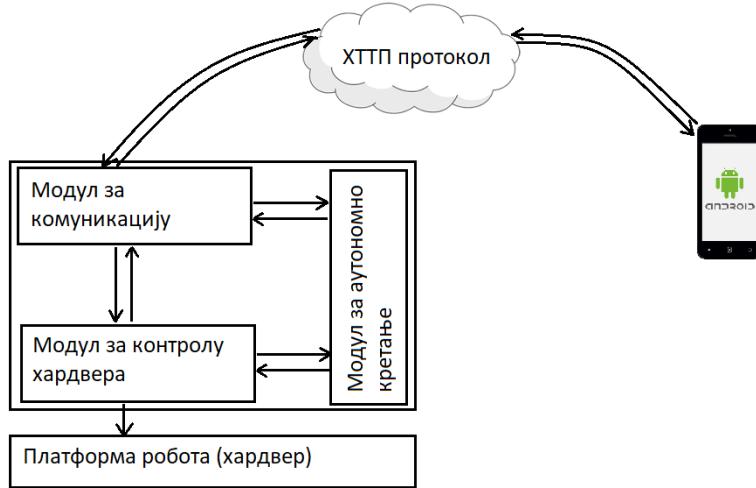
Модул за комуникацију се први покреће и он мора бити активан од момента укључивања до момента искључивања робота. Овај модул се покреће чим се Росбери бутује. То се постиже тако што се у фајл

```

1 /etc/rc.local
2 sudo python3 /home/pi/rpidroid_robot/communication_module.py

```

дода наредба за покретање модула за комуникацију.



Слика 5.5: Сликовит приказ модула апликације на Росберију и њихове сарадње

Модул за комуникацију је имплементиран као флакс (енг. *flask*) сервер. Флакс је лагани (енг. *lightweight*) веб фрејмворк написан у Пајтону. Не захтева посебне алате и библиотеке па је због тога јако погодан за уређаје са мањом количином рачунарске снаге. Нема функционалности попут слоја абстракције базе података, валидацију формата и многе друге функционалности које се обично очекују од веб сервера али упркос томе флакс се може лако проширити додавањем екstenзија и прилагодити сопственим потребама. У мастер раду је дефинисана функција ФлаксАпликација (енг. *FlaskApplication*) која инстанцира објекат класе Флакс и она се покреће у посебној нити. У њој се врши:

- Подешавање ruta
- Парсирање тела захтева
- Ажурирање контекста односно стања робота
- Позивање функција из АПИ-а модула за контролу хардвера
- Покретање модула за контролу аутономног кретања робота
- Слање одговора и обавештења Андроид апликацији

ГЛАВА 5. АПЛИКАЦИЈА

Модул за контролу платформе робота

Модул за контролу платформе робота се покреће када модул за комуникацију добије захтев за повезивање са роботом од Андроид апликације, односно корисника. У мастер раду је имплементиран кроз функцију КонтролаГПИО (енг. *ControlGPIO*) и покреће се у посебној нити. На почетку се врши неопходна иницијализација стања свих хардверских компоненти. Комуникација са модулом за комуникацију се одвија преко објекта контекстРПи (енг. *ContextRPi*) класе Контекст (енг. *Context*) која садржи информације о тренутном стању робота и новим командама које су стигле. Овде се такође уписују и одговори које треба вратити кориснику. Читање и ажурирање информација се штити мутексима. Овај модул има неколико задатка:

- Очитавање наредне команде од модула за комуникацију кроз објекат контекстРПи
- Покретање и заустављање мотора
- Очитавање дистанце са сензора
- Ажурирање стања робота кроз објекат контекстРПи

Контрола се врши кроз контролу ГПИО пинова на Росберију. За контролу пинова потребно је импортовати ГПИО библиотеку која долази са Росбиан оперативним системом.

```
1 import RPi.GPIO as GPIO
```

Ова библиотека пружа удобан АПИ како за слање излаза тако и за очитавање улаза са ГПИО пинова. У наставку је команда којом се укључује мотор који покреће леви точак робота.

```
1 GPIO.output(21,GPIO.HIGH)
```

Овом командом се поставља логичка јединица на пин 21 која ће контролеру мотора пренети ту информацију а он ће провести струју до мотора и мотор ће се покренути. За контролу мотора су коришћени пинови 2, 3, 4, 16, 20 и 21. Прва три су коришћена за десни точак а друга три за леви. Пинови 4 и 21 служе за укључивање и искључивање мотора а пинови 2 и 3 односно 16 и 20 служе да одреде смер у ком се мотор, односно точак окреће. Тако, на пример, ако се пинови поставе на следећи начин, робот ће се кретати напред.

ГЛАВА 5. АПЛИКАЦИЈА

```
1 GPIO.output(16,GPIO.HIGH)
2 GPIO.output(20,GPIO.LOW)
3 GPIO.output(2,GPIO.HIGH)
4 GPIO.output(3,GPIO.LOW)
5 GPIO.output(4,GPIO.HIGH)
6 GPIO.output(21,GPIO.HIGH)
```

Ако би заменили вредности за пинове 16 и 20 односно 2 и 3 робот би се кретао назад. Ово је заиста лак начин за контролу али у пракси ово не ради како бисмо очекивали. Проблем је у томе што хардверске компоненте као што су ДЦ мотори немају идеално исте карактеристике па се у случају из претходног примера, уместо очекиваног понашања праволинијског кретања напред, дешава да робот благо скреће у лево или у десно. Разлог за то је мала разлика у брзини обратаја мотора која се манифестије оваквим понашањем. Решење за овај проблем је додавање радног такта за сваки од мотора. Под радним тактом се мисли на то да мотор није константно укључен већ да је део времена укључен а део времена искључен што ће се одразити на брзину обратаја точка. На овај начин се експерименталним мерењима долази до вредности тих малих пауза у раду мотора којима се поправљају разлике у самим моторима.

Још један битан задатак овог модула је очитавање вредности са сензора удаљености. Овај задатак се периодично понавља у једнаким временским интервалима. Овај сензор емитује ултразвучни импулс када му се сигнализира да то треба да уради и поставља логичку јединицу на свој излазни пин који је повезан са Росберијевим ГПИО пином. Дистанцу до најближе препреке меримо тако што помножимо брзину простирања ултразвучних таласа са временом путовања таласа. Време се мери од момента емитовања сигнала до момента пријема одбијених сигнална од препреке. Брзина простирања ултразвучних таласа је позната и познато је време које је потребно поделити са два, јер је то време за које сигнал стигне до препреке и врати се до робота.

За окидање слања сигнала са сензора и пријем повратне информације, за један од сензора, коришћени су пинови 7 и 11. Очитавање се врши на следећи начин:

```
1 PIN_TRIGGER = 7
2 PIN_ECHO = 11
3
4 GPIO.setup(PIN_TRIGGER, GPIO.OUT)
5 GPIO.setup(PIN_ECHO, GPIO.IN)
6
```

ГЛАВА 5. АПЛИКАЦИЈА

```
7 GPIO.output(PIN_TRIGGER, GPIO.LOW)
8
9 print "Racunanje razdaljine"
10 GPIO.output(PIN_TRIGGER, GPIO.HIGH) #signalizacija senzoru da emituje
11     signal
12 time.sleep(0.00001)
13 GPIO.output(PIN_TRIGGER, GPIO.LOW)
14
15 while GPIO.input(PIN_ECHO)==0:
16     pulse_start_time = time.time() #vreme slanja signala
17 while GPIO.input(PIN_ECHO)==1:
18     pulse_end_time = time.time() #vreme dobijanja odbijenog signala
19
20 pulse_duration = pulse_end_time - pulse_start_time #razlike vremena
21
22 distance = round(pulse_duration * 17150, 2) #racunanje distrance
23
24 print "Razdaljina:",distance/2,"cm"
```

Модул за аутономно кретање

Модул за аутономно кретање омогућава роботу аутономно кретање тако што имплементира БУГ алгоритам описан у поглављу 4.

Када је овај режим укључен робот се креће напред све док не нађе на препеку. Тада се тренутна позиција памти као нулта тачка и памти се тренутни правац и смер робота.¹ На основу експериментално измерених вредности пређеног пута у јединици времена, у сваком кораку се ажурира позиција робота у том координатном систему и угао под којим гледа у односу на онај када је нашао на препеку. Бира страну са које ће заобићи препеку тако што провери вредности са свих сензора и анализом вредности утврди који сензори јављају отворенији пут. Након тога, у зависности од одабране стране, се репозиционира, ажурира контекст и поново покушава кретање напред, ако је могуће.

Централни део имплементације се налази у следећем исечку кода:

```
1 flagGo=False
2 obstacleDetected=False
3
4 while(True):
5     targetReached=checkTarget() #proverava da li je cilj dostignut
6     if(targetReached==True): #ako jeste prekida algoritam
7         break
8
```

¹То је координатни систем који се успоставља у тренутку наиласка на препеку

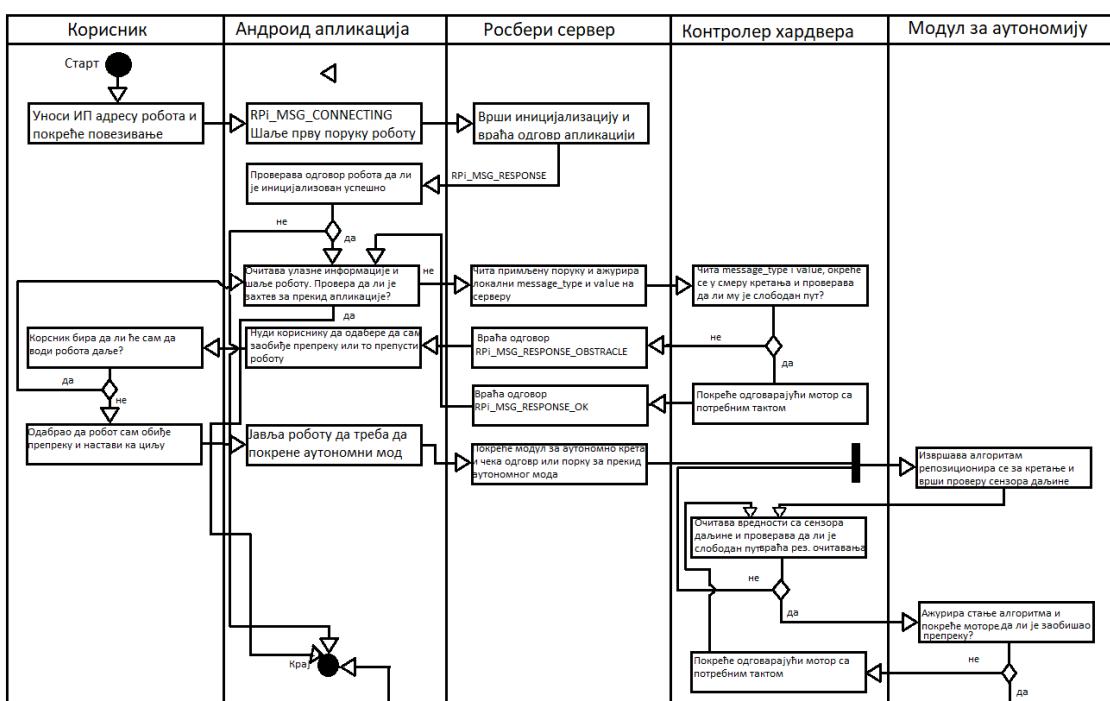
ГЛАВА 5. АПЛИКАЦИЈА

```
9  obstracleDetected=goAhead() #Pokusava kretnju napred ako je moguca
10
11 while(!obstracleDetected): #ide napred sve dok moze
12     flagGo=True
13     obstracleDetected=goAhead()
14
15 side=chooseTheSide(flagGo) #kada je naisao na prepreku proverava sve
16     senzore i bira stranu
17
18 if(side==left):
19     flagGo=False
20     turnLeft5Degrees() #pravi zaokret u levo
21 else if(side==right):
22     flagGo=False
23     turnRight5Degrees() #pravi zaokret u desno
24
25 updateContext() #azurira informacije u kontekstu
```

Имплементације осталих функција које се позивају се ослањају на већ описане функционалности модула за контролу хардвера са додатком математичких прорачуна. Целокупан код апликације се налази на ГитХуб-у на адреси <https://github.com/nenadlazic/RPiRobot.git>.

Слика 5.6 приказује дијаграм случајева употребе.

ГЛАВА 5. АПЛИКАЦИЈА



Слика 5.6: Дијаграм случајева употребе

Глава 6

Закључак

На самом kraју друге деценије 21. века сведоци смо све веће присутности роботике и аутоматизације. Њена присутност и утицај у будућности ће бити све већи и у све ширем спектру примена.

У овом раду детаљно је приказан развој мобилног робота, корак по корак. Финални изглед робота може се видети на слици 6.1. Количина коришћених софтверских алата и хардверских компоненти говори о томе да је развој једног робота комплексан процес. Направљени робот је потпуно функционалан. Робот се у сваком моменту може контролисати Андроид апликацијом ручним навођењем, укључивањем и искључивањем аутономног режима, а омогућено је и добијање повратних информација од робота. Робот има функционалну детекцију препреке на путу, заустављање и јављање апликацији. Алгоритам за аутономно кретање је такође имплементиран и функционалан. Алгоритам ради са прихватљивом прецизношћу али са нешто мањом од очекиване због ограничene количине информација о простору и изгледу препреке која се може добити са коришћеним сензорима.

Развојем Андроид апликације за навођење потврђено је да у оквиру Андроида постоје уgraђене библиотеке које имају подршку за најбитније функционалности у овом контексту као и добро документован АПИ. Писање софтвера за Росбиан у програмском језику Пајтон се такође испоставило веома удобно, делом захваљујући већ развијеним библиотекама а делом и због карактеристика самог језика Пајтон. Обе платформе, Андроид и Росбиан, су јако захвалне за рад и у потпуности су испуниле очекивања.

Сам робот има неколико места за надоградњу. Један од могућих правца даљег рада је имплементација напреднијих алгоритама за аутономно заобија-

ГЛАВА 6. ЗАКЉУЧАК

жење препрека као што је Тангентни БУГ алгоритам који захтева додавање прецизнијих сензора који би могли да пруже информацију о близини препрека у свих 360 степени. Још једно побољшање којим би се могао унапредити алгоритам је додавање сензора за мерење пређеног пута како би се омогућило прецизније мерење позиције робота.



Слика 6.1: Изглед робота након завршетка рада и склапања

Библиографија

- [1] Karim Yaghmour. *Embedded Android*. O'REILLY 978-1-449-30829-2, 2013.
- [2] Ian F. Darwin. *Android cookbook*. O'REILLY 978-86-7555-383-0, 2013.
- [3] Erik Hellman. *Android programming-pushing the limits*. Wiley 978-1118717370, 2013.
- [4] Bert Van Dam. *Raspberry Pi*. EHO 978-86-915999-9-7, 2014.
- [5] Иштван Пап, Немања Лукић. *Пројектовање и архитектура софтверских система: системи засновани на Андроиду*. Факултет техничких наука 978-86-7892-773-7 , 2015.
- [6] The Apache Software License (ASL)
<http://www.apache.org/licenses/LICENSE-2.0>
- [7] Android Open Source Project License
<https://source.android.com/setup/licenses>
- [8] Mobile Operating System Market Share Worldwide
<https://www.statista.com/statistics/266136>
- [9] Android history
<https://developer.android.com/about/dashboards/index.html>
- [10] Raspberry Pi organization site
<https://www.raspberrypi.org/>
- [11] GNU licence
<https://www.gnu.org/licenses/fdl-1.3.en.html>

БИБЛИОГРАФИЈА

- [12] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, Sebastian Thrun *Principles of Robot Motion*. MIT 978-02-6233-251-4 , 2005.

Биографија аутора

Ненад Лазић Рођен сам у Љубовији 13. фебруара 1993. године. Основну школу Доситеј Обрадовић сам завршио у Љубовији 2008. године након чега сам уписао Техничку школу у Ваљеву, смер електротехничар рачунара. Током средње школе заволео сам програмирање и математику због чега сам одлучио да мој избор у наставку школовања буде Математички факултет у Београду. На Математички факултет сам се уписао 2012. године на смеру Информатика. Основне академске студије сам завршио 2015. године након којих сам уписао мастер студије.