Универзитет у Београду – Електротехнички факултет

Катедра за сигнале и системе

ДИПЛОМСКИ РАД

ЈЕДНА РЕАЛИЗАЦИЈА МОБИЛНОГ РОБОТА УПРАВЉАНОГ ГЛАСОМ

Кандидат:

Аница Чанчар, 2016/0265

Ментор:

доц. др Сања Вујновић

Београд, *септембар* 2020. године

Захвалница

Овим путем желим да се захвалим својој породици, дечку и пријатељима на огромној подршци и разумјевању током мог школовања, као и мојој менторки на помоћи при изради овог рада.

Аница Чанчар,

У Београду, 2020. године

Садржај

1 Увод и позадина

1.1 Увод

Роботика је данас дио свакодневне комуникације. Ова област се данас јако брзо развија и представља занимљиво подручје за инжењере. Роботи се у највећој мјери користе у индустрији, али је све већа производња робота у сврху поједностављења свакодневног живота човјека. Стога, неминовно је све веће истраживање и унапријеђење ове области.

Управљање говором, односно коришћење говорних комданди у последње вријеме је постало веома интересантно. Примјену оваквог начина управљања можемо да видимо у доста *IoT*[1] система, највише у области кућне аутоматизације. Са развојом паметних технологија јасна је све већа потреба за реализацијом оваквог начина управљања и у другим системима, као што су системи у аутоиндустрији. С обзиром да је овакав вид управљања лак и примамљив за човјека, у блиској будућности се очекује велика експанзија управљања говором и у другим областима.

Уколико желимо да постигнемо аутономност нашег робота у простору, први корак при његовој изради је детектовање препрека. На тај начин робот има могућност опажања окружења, па се самим тим степен његове интелигенције повећава.

Посебно занимљива тема у области роботике је израда аутономних аутомобила који су способни за самостално учествовање у саобраћају. Посједовање сензора који детектују препреке и избјегавање тих препрека је један од основних корака при изради оваквих робота.

Заинтересованост за претходне теме ме је навела да се у изради овог рада одлучим за реализовање робота који има могућност управљања гласом, као и ручног управљања на принципу даљинског управљача, уз способност детекције препрека и њиховог обилажења.

[1] Internet of Things – Интернет ствари

У овом раду приказани су дизајан и имплементација мобилног робота с освртом на хардвер, софтвер и комуникацију са окружењем у сврху детектовања препрека у реалном времену и њиховог обилажења. За пројектовање овог система коришћене su *Arduino, Bluetooth* и *Android* технологије. Као мозак робота коришћен је *Arduino Uno.* Робот се састоји од доста хардверских компоненти, као што су ултрасонични сензор, *Bluetooth* модул, сервомотор, *DC* мотори са редукторима. Такође у себи садржи доста софтверских компоненти као што су *Android* апликација, која има могућност одабира режима управљања, као и управљања роботом. Корисник може са свог паметног уређаја да управља задајући смјер помоћу даљинског управљања или гласовним командама, а робот такође има могућност детектовања и обилажења препрека.

1.2 Структура пројекта

Овај пројекат се састоји из пет дијелова:

- Први дио представља увод у пројекат
- Други дио говори о *Arduino* платформи
- Трећи дио представља приказ хардверског дијела пројекта
- Четври дио представља приказ софтверског ријешења
- Пети дио су закључак, приједлози за даље унапријеђење и литература

2 Ардуино

2.1 Уопштено о Ардуину

Ардуино је популарна програмибилна плоча која се користи за израду различитих врста пројеката. Састоји се од једноставне хардверске платформе, као и бесплатног уређивача кода који има могућност једноставног компајлирања и слања кода на контролер.

Ардуино нуди *open-source* софтверску платформу која је једноставна и флексибилна за коришћење како за софтвер тако и за хардвер. Ардуино језик представља скуп *C/C++* функција. Хардверски дио је такође *open-source*, па је могуће унаприједити и направити сопствени модул.

Он има могућност опажања околине на тај начин што прима опажаје, који су представљени као улази у Ардуино, од стране сензора. Такође, може управљати окружењем, контролишући моторе, свијетла и остале актуаторе.

Ардуино плоче су релативно јефтине у поређењу са другим платформама микроконтролера. Софтвер може да се покрене на више различитих оперативних система, као што су *Windows, Macintosh OSX,* и *Linux,* док је већина микроконтролера ограничена на *Windows* оперативни систем.

Слика 1. Лого Арудино компаније

2.2 Историја

Пројекат израде Ардуина је започет на *Interaction Design Institute Ivrea(IDII)* од стране студента у Италији 2005. године. Од тада је Ардуино успио да покрене иницијативу за разне врсте пројеката „уради сам“ у електронској индустрији. Хардвер је направљен тако да може лако да се повеже са разним сензорима који представљају улазе и да управља спољашњим уређајима, као што су звучници, мотори и многе друге компоненте. Најбитнија карактеристика Ардуина је једноставност програмирања, па и корисници са мало искуства и знања о тој области могу користити. Ова карактеристика га је учинила једним од најпопуларнијих алата за израду различитих интерактивних објеката.

2.3 Развој микроконтролера

Прије приче о развоју Ардуина, корисно је укратко описати историју микроконтролера. Револуционарни корак у развоју рачунарске индустрије се десио 1960. године са развојем ССД рачунара, који су користили транзисторе да процесирају своје операције и меморију магнетног језгра за њихово чување, умјесто вакуумских цијеви, што је омогућило већу компактност у рачунарском хардверу. Такође, изум интегрисаних кола 1959. године омогућио је стапање кола и транзистора у мале чипове од полупровдничких материјала (сицилијум) као и даљу минимизацију рачунарских компоненти. Следеће јако битно откриће у истој декади је било откриће програмских језика виших нивоа, писаних симболичним језицима. Ово је олакшало писање и читање програмских кодова у односу на до тада кориштени машински језик. *FORTRAN* и *COBAL* су два главна језика која су кориштена у том периоду.

Микропроцесор је једно од највећих открића у историји модерних рачунара. На почетку, микропроцесор је минимизовао све хардверске компоненте *CPU*-а, тако да су оне стале у једно мало интегрисано коло, познатије као микрочип. Микрочип је постао главна покретачка компонента свих микроконтролера, укључујући и Ардуина, који је направљен од микрочипа, улазно/излазног хардвера и меморијског дијела за сензоре. Због малих

димензија микропроцесор је уграђен у велики број електронских уређаја, почевши од калкулатора до персоналног рачунара. Убрзо је развијено још програмских језика као што су *C, C++* и *Java.*

2.4 Развој Ардуина

У развоју микроконтролера, појавили су се типови микроконтролера који су дизајнирани да задовоље потребе хобиста и корисника који имају ограничено техничко знање. Прије изума Ардуина, кориштен је *PIC* микроконтролер који се појавио 1985. године и био једна од најкориштенијих компоненти за електронске ентузијасисте. Разлог због кога је овај микроконтролер био популаран су брзина и једноставност његовог програмирања кроз језике као што су *PBASIC.*

Ардуино тим је формиран у Италији 2005. године и чинили су га *Barragan Massimo, David Cuartielles, Gianluca Marino, Dave Mellis and Nicholas Zambett*. Главни циљ је био развијање електронске прототипске платформе која би поједноставила платформу за ожичавање и направила је приступачном за нестручане кориснике, посебно у креативним пољима.

2.5 Ардуино некад и сад

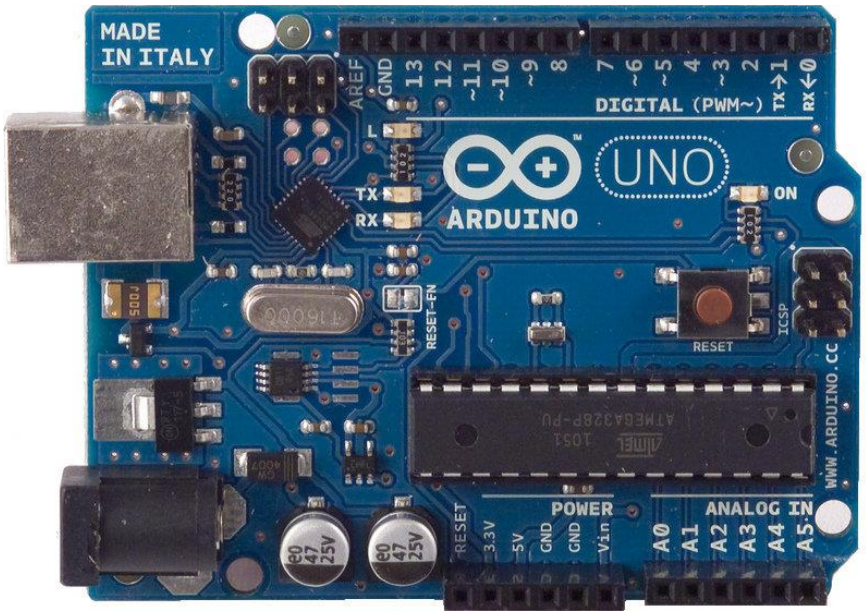
Ардуино је брзо достигао велики успјех. Само за двије године његовог постојања продато је више од 50000 плоча. До 2009. године, Ардуино је имао више од 13 различитих реализација, при чему је свака имала специјализовану апликацију. На примјер, *Arduino Mini* је имао мале димензије, како би се користио у малим интерактивним објектим, *Arduino BT* је направљен за рад са *Bluetooth* модулима и *Arduino Lilypod* за пројекте носивих технологија.

Ардуино платформа постала је прилично популарна међу људима који тек почињу са електроником, и то с добрим разлогом. За разлику од већине претходних програмабилних плочица, Ардуину није потребан посебан комад хардвера (који се назива програмер) да би учитао нови код на плочу - једноставно можете да користите УСБ кабл. Поред тога, Ардуино *IDE* користи поједностављену верзију *C++*-а, што олакшава учење програмирања.

2.6 Ардуино плоче

Ардуино је *open-source* платформа која се користи за израду електронских пројеката. Он се састоји од физичке програмибилне плоче (често се назива микроконтролер) и дијела софтвера или *IDE* (Интегрисано развојно окружење) који се покреће на рачунару и користи се за писање и отпремање кода на плочу.

Као што је већ речено, срце Ардуина је микроконтролер. Ардуино најчешће користи осмобитне микроконтролере које производи компанија *ATMEL*. Најраспрострањенији модел је *Atmega328p* који се користи на основној Ардуино проторипској плочици, приказаној на следећој слици.



Слика 2. Ардуино плочица са *Atmega328p* микроконтролером

Предности Ардуино платформе

- Јефтина
- Могућност покретања на различитим оперативним системима
- Једноставно програмерско окружење
- Надоградив и *open-source* софтвер
- Надоградив и *open-source* хардвер

2.7 Типови и карактеристике Ардуино плоча

Неке од најпознатијих Ардуино плоча и њихове спецификације су приказане у следећој табели.

Ардуино Плоча	**Напон**	**Процесор**	**Меморија**	**Дигитални улази/излази**	**Аналогни улази/излази**
Arduino Uno	5V	16Mhz ATmega328	2KB SRAM, 32KB flash	14	6 улаза, 0 излаза
Arduino Due	3.3V	84MHz AT91SAM3X 8E	96KB SRAM, 512KB flash	54	12 улаза, 2 излаза
Arduino Mega	5V	16MHz ATmega2560	8KB SRAM, 256KB flash	54	16 улаза, 0 излаза
Arduino Leonardo	5V	16MHz ATmega32u4	2.5KB SRAM, 32KB flash	20	12 улаза, 0 излаза

Табела 1. Типови Ардуино плоча

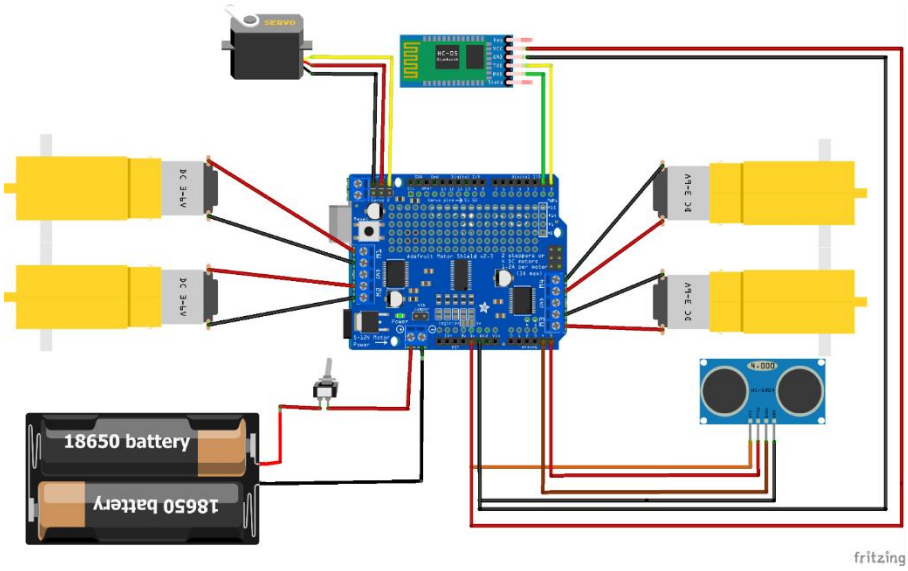
3 Хардверска реализација

3.1 Циљ пројекта

Као што смо већ рекли, идеја овог пројекта је реализација мобилног робота који би имао могућност управљања путем гласовних, као и даљинских команди, а са додатном способношћу детекције препрека и њиховог обилажења. Управљање би се реализовало преко одговарајућег *Bluetooth* модула, користећи мобилни телефон и одговарајућу апликацију. За управљање самим моторима користи се одговарајући драјвер за моторе. Детекција препрека се обезбјеђује помоћу ултрасоничког сензора. Идеја је да овај робот буде довољно интелигентан, како би се поштовао одређене команде, кретао се по простору и вршио детекцију препреке као и њено обилажење.

3.2 Блок дијаграм и опрема

На следећој слици приказан је блок дијаграм хардверске репрезентације пројекта направљен помоћу алата *Fritzing*[2]



Слика 3. Блок дијаграм

[2] *Fritzing* је *open-source* иницијатива за развој аматерског или хоби *CAD* софтвера за дизајн електронског хардвера.

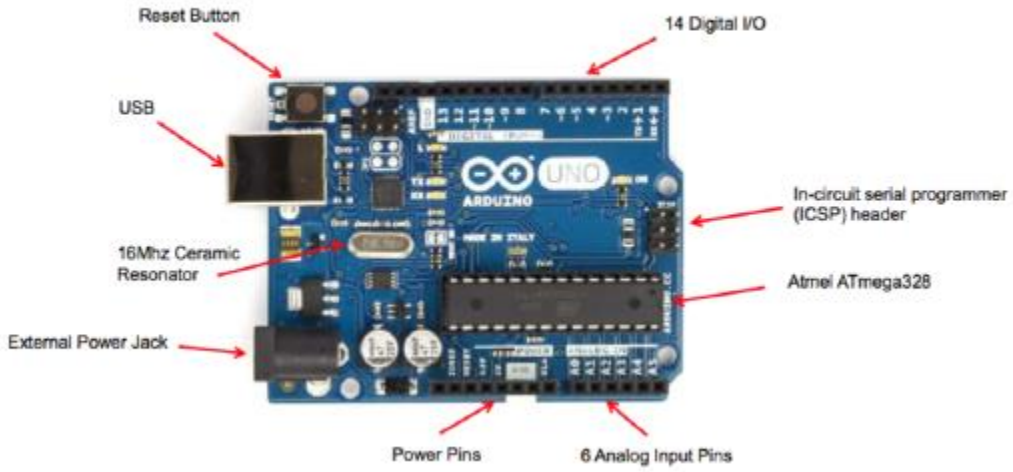
Хардверски уређаји које су коришћене у изради овог рада су:

- Arduino UNO R3
- L293D драјвер за моторе
- Мотори са редукторима 3-6V
- SG-90 Серво мотор
- HC-SR04 Ултрасонични сензор
- HC-05 *Bluetooth* модул

У наставку је објашњена свака од наведених компоненти, дат њен принцип рада, као и улога коју има у изради робота.

3.3 *Arduino UNO R3*

Као мозак овог робота коришћен је *Arduino UNO R3.* На следећој слици су приказани његови дијелови.



Слика 4. *Arduino UNO R3* плоча

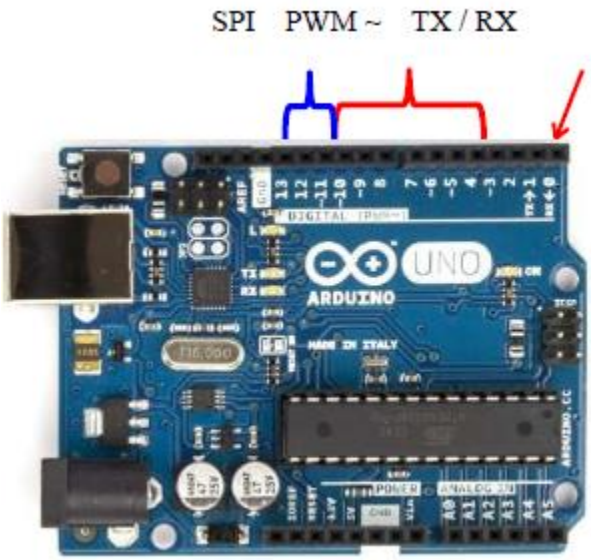
3.3.1 *Arduino UNO* - Напајање

Непосредно испод *USB* прикључка налази се регулатор напона од 5V. Он претвара сваки напон(између 7V и 12V) који долази из утичнице *DC* напајања у константан напон од 5V. Постоје два могућа начина на који овај контролер може да се напаја. Први начин представља спољашњи прикључак за напајање, док други начин представља *USB* напајање. Такође, са слике видимо да постоје конектори који се називају конектори за напајање(*Power Pins)* и служе као извор напајања повезаних уређаја. У наставку је објашњена функција сваког од ових конектора.

Конектор	**Функција**
Vin	Напон са екстерног прикључка напајања
5V	5V излаз са уграђеног чипа регулатора напона
3.3V	3.3V излаз са уграђеног чипа регулатора напона
Gnd	3 конектора за уземљење
IOREF	Користи се да обавјести заштиту који напон да очекује на улазно/излазним конекторима од основног Ардуина
Reset	Ресетује микроконтролер тако да он почиње свој програм испочетка

Табела 2. Функције конектора напајања

3.3.2 *Arduino UNO* – Дигитални улази/излази



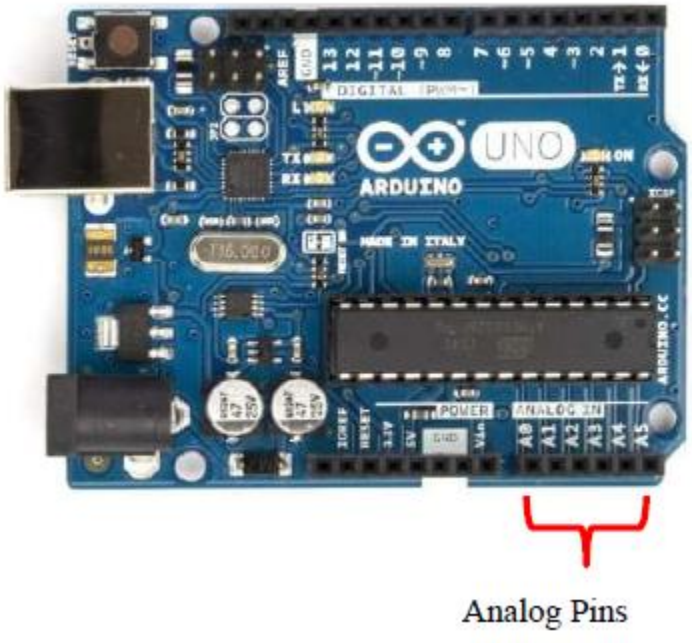
Слика 5. Дигитиални улази/излази Ардуина

- На плочи се налази 14 дигиталних улаза/излаза.
- Напајање износи 40mA по конектору при напону од 5V
    - Ово је довољно за напајање стандардне *LED* диоде, али недовољно за покретање мотора, због тога ако је неопходно управљање моторима морају се користити драјвери за моторе.
- Универзални асинхрони серијски пријемник/предајник
    - Конектор 0 – Пријемник (RX)
    - Конектор 1 – Предајник (TX)

Служе за комуникацију и индиректно представљају пријемни и предајни прикључак за *USB* везу ка рачунару.

- Као 8 – битни *PWM* конектори се могу користити 3, 5, 6, 9, 10, 11
- Као конектори за *SPI* комуникацију могу бити кориштени пинови 10, 11, 12, 13

3.3.3 *Arduino UNO* – Аналогни улази



Слика 5. Аналогни улази Ардуина

- На слици је приказано шест аналогних улаза од A0 – A5
- Могу да мјере напон на конекторима у опсегу од 0 – 5V
- Конвертор има 10 – битну резолуциу и враћа вриједности од 0 до 1023
- Референтни напон за ове улазе се поставља помоћу конектора *Aref*
- Постоји могућност кориштења већине ових конектора као дигиталних улаза/излаза
- *I2C*[3] протокол користи конектор A4 као *SDA(Data)* пин и конектор A5 као *SCL(Clock)* пин.

[3] *I2C* – Протокол који користи двије линије за слање и пријем података. Ово је корисно када желимо да иамо двије плоче које треба да дијеле информације.

3.3.4 *Arduino UNO* – Комуникација

Ова плоча има могућност комуникације са *PC* – јем, другим Ардуино плочама, драјверима и сензорима.

Постоје два типа комуникације: асинхрона и синхрона.

- Асинхрона комуникација(не користи сат)
  - Свака Ардуино плоча има бар један серијски порт који се назива *UART TTL(5V)*
    - Користе се дигитални конектори 0 као пријемник(*RX*) и 1 као предајник(*TX)*
  - На плочи је уграђен *ATmega16U2* који служи као мост између *USB* и главног процесорског серијског порта. Он врши *UART* −> *USB* конверзију па се понаша као виртуелни *COM* пркикључак за рачунар.
  - *ATmega16U2* користи стандардни *USB COM* драјвер, па нису потребне никакве додатне компоненте.
  - *TX / RX LED* – ови трепере док се врши пренос података преко *USB* – а на серијски чип.
- Синхрона комуникација(користи се сат)
  - *Serial Peripheral Interface (SPI)* је протокол за комуникацију микроконтролера са једним или више периферних уређаја, као и за међусобну комуникацију микроконтролера. Код овог типа везе увијек постоји један мастер уређај који контролише остале уређаје. Обично постоје 3 линије заједниче за све уређаје
    - *MasterOut/SlaveIn(MOSI)* – Линија мастера, којом шаље податке осталим уређајима. Код Ардуина за ову линију се користи пин 11.
    - *MasterIn/SlaveOut(MISO)* – Линија којом уређај шаље податке мастеру. Код Ардуина за ову линију се користи пин 12.
    - *Serial Clock(SCK)* – Сат који даје пулсеве, што синхронизује пренос података, генерисане од стране мастера. Код Ардуина за ово служи пин 13.

- *Slave Select(SS)* – Ова линија је специфична за сваки уређај. Овај пин користи мастер како би омогућио коришћење, односно искључивање уређаја. Код Ардуина за ово се користи пин 10.

- *I2C* протокол подразумјева коришћење двије линије за слање и пријем података. Те линије су:
  - *SCL* – Конектор за сат у коме Ардуино плоча шаље импулсе у једнаким интервалима. Код Ардуина као *SCL* конектор се користи A5.
  - *SDA* – Серијски конектор кроз који се шаљу подаци. Код Ардуина као *SDA* конектор се користи A4.

Како се импулси времена мијењају са ниског на високи ниво, један бит информације преноси се са једног уређаја на други.

3.3.5 *Arduino UNO* – Програмирање

Програмирање Ардуина се ради у специјалном програмском окружењу које се назива *Arduino Integrated Development Environment (IDE)*. Спуштање кода се врши путем *USB* прикључка.

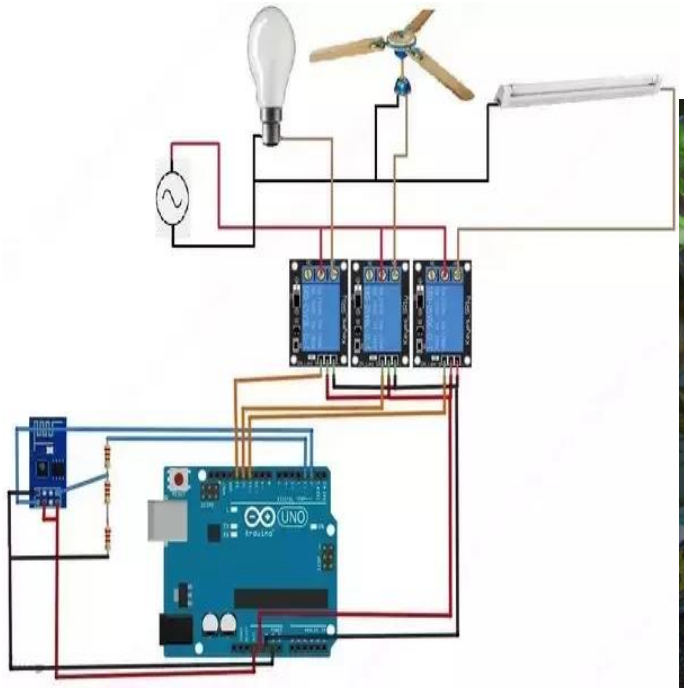
- Меморија *Arduino UNO*
  - 32KB *Flash* меморије – овдје се чувају програмске скице
  - 2KB *SRAM* меморије – овдје се чувају промјењиве док се не рестартује програм
  - 1KB *EEPROM* меморије – овдје се чувају дугорочне промјењиве, константе, серијске промјењиве..

3.3.6 *Arduino UNO* – Примјена

Употреба фамилије ових контролера ће у будућности сигурно да буде већа.

Неке основне примјене овог конторлера су:

- Роботика и системи управљања
- Кућна и индустријска аутоматизација
- Биотехнологија
- Пољопривреда



Слике 6 и 7. Употреба Ардуино Уно у кућној аутоматизацији и пољопривреди

3.4 *HC – SR04* Ултрасонични сензор

Ултрасонични сензори су сензори који конвертују ултрасоничне звучне таласе у електричне сигнале и обрнуто. Ултрасоничним звуком се сматра звук који је ван граница чујности људског уха (20000Hz).

Ултрасонични сензор је уређај који може да мјери удаљеност од других објеката користећи звучне таласе. Он мјери удаљеност тако што шаље звучни талас одређене учестаности и чека да се тај звучни талас одбије. Снимајући претекло вријеме између генерисаног звучног таласа и његовог одбијања омогућава срачунавање удаљености између сензора и објекта.

HC – SR04 ултрасонични сензор је једноставан за употребу и даје добре резултате бесконтактне детекције објеката са високом тачности и стабилним очитавањима. Опсег износи од $2 - 400cm$. Његов рад се не омета сунчевом свијетлошћу или црним материјалима, мада јако мекани материјали попут крпе могу да буду тешки за детекцију.

Састоји се од ултрасоничног пријемног и предајног модула.

Слика 8. Ултрасонични сензор

Знајући да је брзина звука кроз ваздух $344\,\frac{m}{s}$, можемо потребно вријеме да звук оде и да се врати помножимо са овом брзином и добијемо укупни пређени пут звука. С обзиром да смо на овај начин рачунали да путовање звука до објекта и назад, удаљеност коју овако добијамо је дупло већа, па је за коначан резултат неопходно да овај израз подијелимо са два.

Формула за рачунање удаљености објекта помоћу ултрасоничног сензора је:

$$d = \frac{ct}{2} \tag{1}$$

Гдје је:

d – удаљеност објекта

$c = 344\,\frac{m}{s}$ – брзина звука

t – протекло вријеме од слања и пријема звука

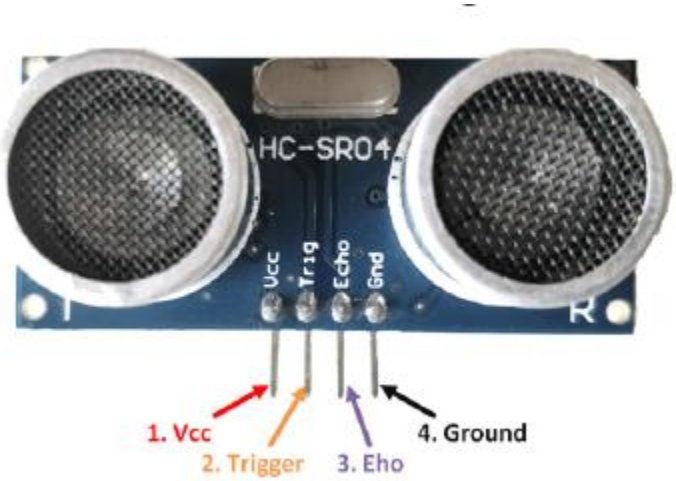
3.4.1 *HC – SR04* Ултрасонични сензор – особине

- Напајање: 5V
- Струја мировања: 2mA
- Радна струја: 15mA
- Ефективни угао: 15°
- Опсег детекције: 2 – 400cm
- Резолуција: 0.3cm
- Угао мјерења: 30°
- Ширина улазних импулса окидача (*trigger)*: $10\mu s$
- Димензије: $45\ mm\ x\ 20\ mm\ x\ 15mm$

Слика 9.

Приказ сензора

3.4.2 *HC – SR04* Ултрасонични сензор – Конектори



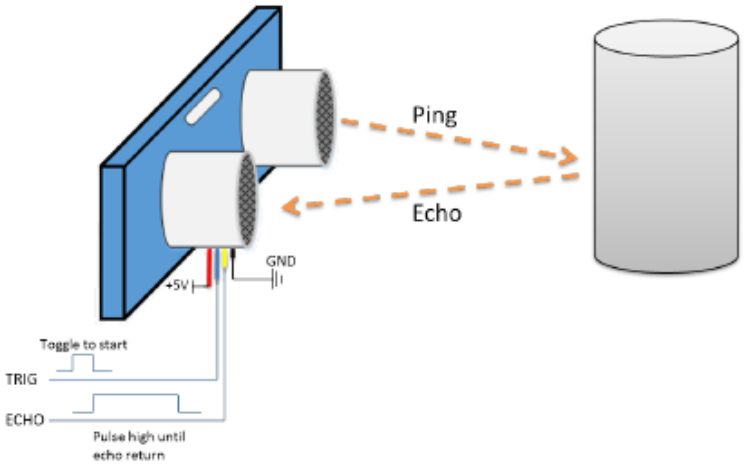
Слика 10. Конектори ултрасоничног сензора

Назив конектора	Фунцкија
VCC	Служи за напајање сензора
Trigger	Овај конектор представља улаз. Он се мора држати на јединици 10μs да би се иницијализовало мјерење слањем ултрасоничног звучног таласа.
Echo	Овај конектор представља излаз. Он ће бити на јединици одређени период времена који је исти као и вријеме које је потребно ултрасоничном звучном таласу да се врати назад до сензора.
Ground	Овај конектор представља конектор за повезивање уземљења

Табела 3. Функција конектора ултрасоничног сензора

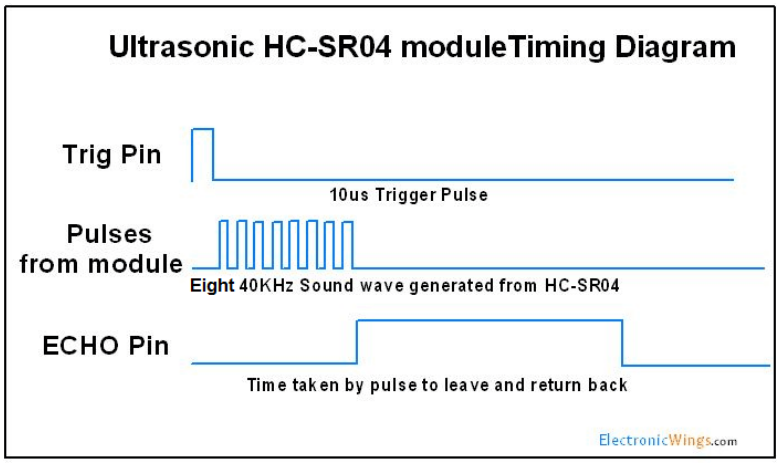
3.4.3 *HC – SR04* Ултрасонични сензор – Принцип рада

Као што смо већ рекли, ултрасонични сензор служи за мјерење удаљености објекта. Ради на истом принципу као радарски систем. Овај тип ултрасоничног сензора шаље ултрасонични талас са учестаношћу 40kHz.



Слика 11. Принцип рада УС сензора

Да би се започело мјерење удаљености, микроконтролер шаље *trigger* сигнал сензору. Тај сигнал траје $10\mu s$. Нако тога ултрасонични сензпр генерише осам акустичних таласних поворки и започиње рачунање времена. Када се прими одбијени звук, тајмер се зауставља. Излаз ултрасоничног таласа је јединични имулс чије је вријеме трајања индетично као протекло вријеме између послатог и примљеног сигнала.

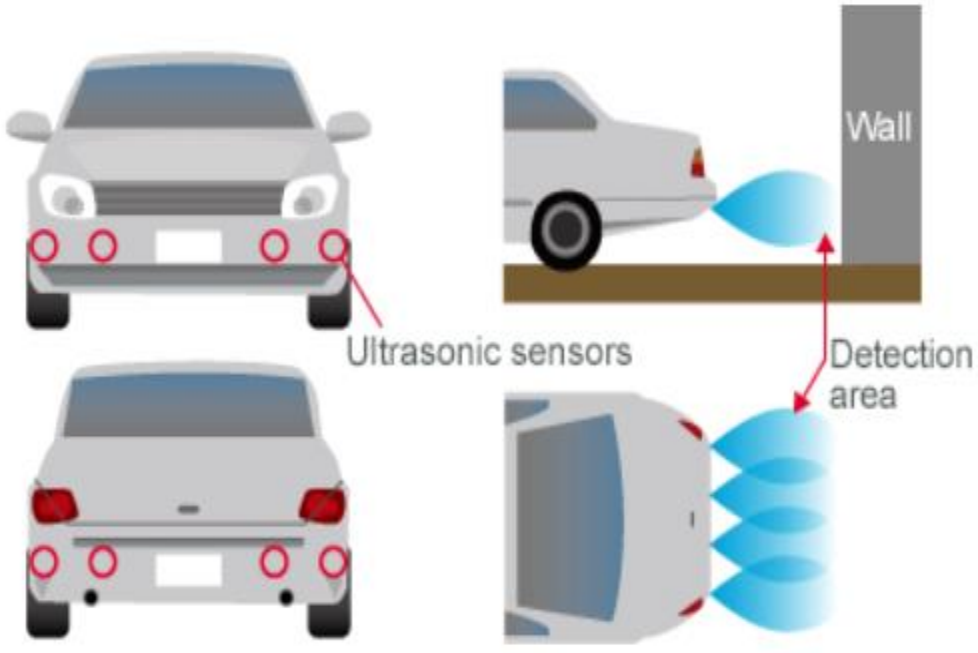


Слика 12. Временски дијаграм УС сензора

3.4.4 *HC – SR04* Ултрасонични сензор – Примјена

Примјене овог сензора су велике, а неке од њих су:

- Овај сензор се користи за мјерење брзине као и смјера између два објекта
- Користи се у бежичном пуњењу
- Медицинска ултрасонографија
- Користи се за откривање предмета и избјегавање препрека помоћу робота
- Мјерење дубине
    - Користећи овај сензор, бунари се могу мјерити преношењем таласа кроз воду
- Аутомобилска индустрија

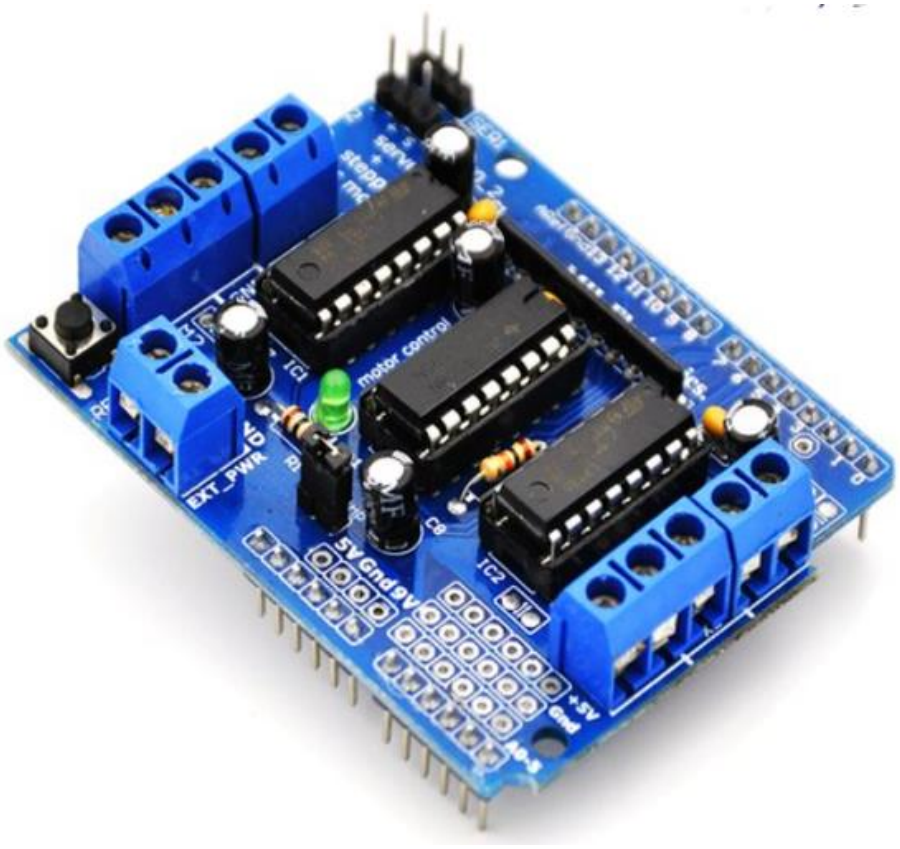


Слика 13. Уптреба УС сензора у аутомобилима

3.5 L293D Мотор драјвер

Драјвер је интерфејс коло између мотора и управљачке јединице и служи за олакшавање покретања мотора. Мотори захтјевају веће струје него што сам Ардуино може да пружи, па да би се омогућило покретање мотора неопходно је коришћење драјвера.

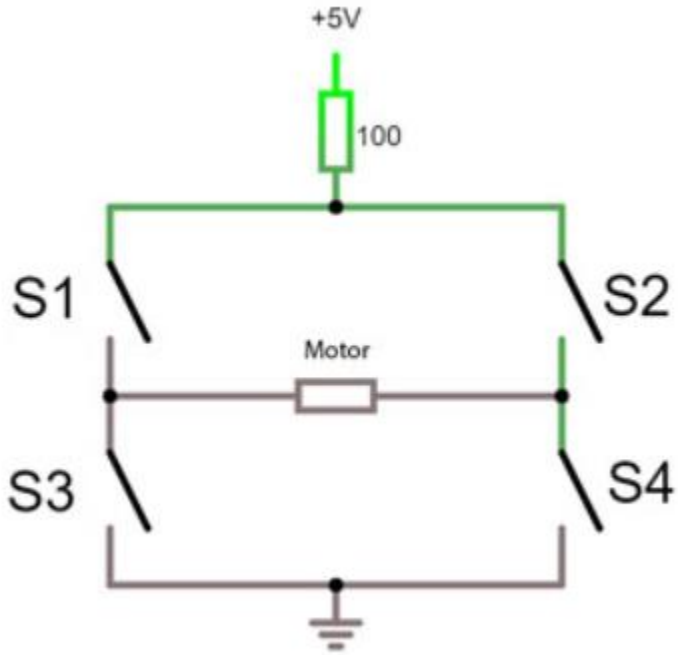
L293D је драјвер заснован на *L293 IC*, који може покренути 4 *DC* мотора и 2 степер или серво мотора истовремено. Сваки канал овог модула има максималну струју 1.2A и не ради ако је напон већи од 25V или мањи од 4.5V. Овај драјвер је познат и као коло типа *H-Bridge*, што омогућава покретање мотора у оба смјера.



Слика 14. L293D мотор драјвер

3.5.1 L293D Мотор драјвер – особине

- Овај драјвер има два прикључка за серво моторе
- Може да се повеже до 4 бидирекциона *DC* мотора
- Могуће повезивање два степер мотора
- У себи садржи четири Х моста
    - Х мост(енг. *H - Bridge)* је електронско коло које омогућује контролу смјера струје кроз једносмјерни електромотор или кроз неки други потрошач. Промјеном смјера струје кроз арматуру једносмјерног мотора, мијења се смјер ротације његове осовине.



Слика 15. Приказ Х моста

- Има отпорнике који омогућавају да мотори буду искључени при укључењу напајања.
- На врху се налази Ардуино ресет дугме
- Има прикључак за екстерно напајање
- Димензије: 69mm x 53mm x 14.3mm
- Компатибилан је са *Mega, UNO* и *Duemilanove*

3.6 *SG – 90* Серво мотор

Слика 16. *SG – 90* серво мотор

3.6.1 Уопштено о серво моторима

Серво мотор је елетрични уређај који може да окреће неки објекат. Ако желимо да ротирамо неки објекат за одређени угао онда користимо серво мотор. Овај серво мотор је направљен од једноставног мотора који се покреће кроз серво механизам. Овакви мотори су нашли примјену у многим областима као што су играчке, хеликоптери, авиони, роботи, машине..

Данас, серво мотори имају огромну индустријску примјену. Серво мотори се често срећу у апликацијама као што су аута на даљински за контролисање смјера кретања и такође, често их можемо наћи у покретачима *CD* и *DVD* – ва. Постоји још много примјена серво мотора у свакодневном животу.

3.6.2 Серво мотор – принцип рада

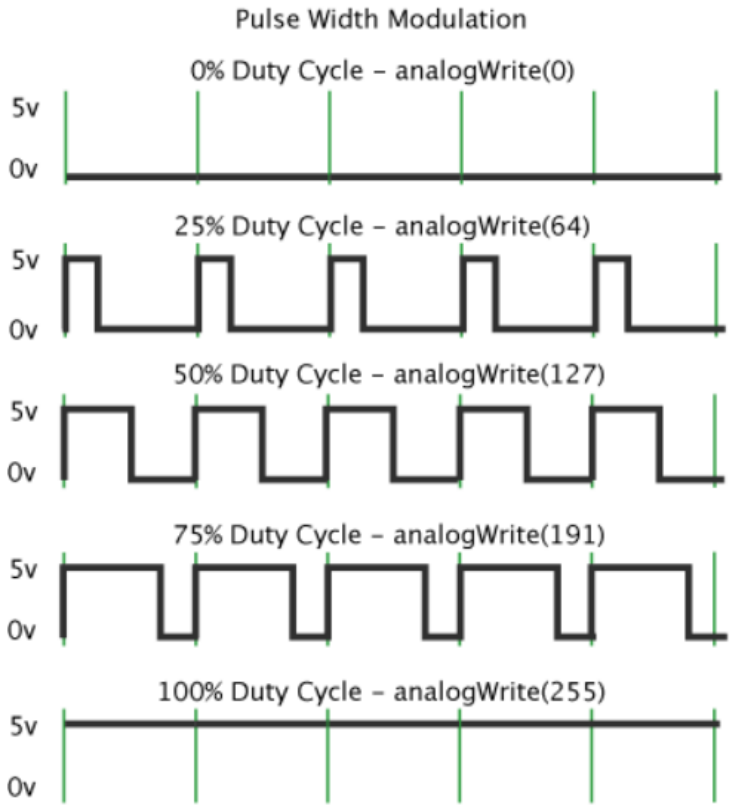
Серво мотор се састоји од потенциометра, склопа зупчаника и управљачког кола.

Прије свега, склоп зупчаника се користи за смањење обртаја у минути и да се повећа моменат мотора. У иницијалној позицији серво мотора, позиција потенциометра је таква да на излазном прикључку потенциометра не постоји електрични сигнал. Када се зада референтни сигнал и грешка расте. Ова грешка представља улаз мотора и он почиње да се окреће. С обзиром да су потенциометар и мотор повезани, са окретањем мотора, доћи ће и до окретања потенциометра, што ће полако довести да се постигне референтни сигнал, грешка ће опадати, а када падне на 0 мотор ће престати да се окреће.

3.6.3 *PWM* – Импулсно ширинска модулација

Прије приче о контроли серво мотора, важно је рећи нешто о импулсно ширинској модулацији.

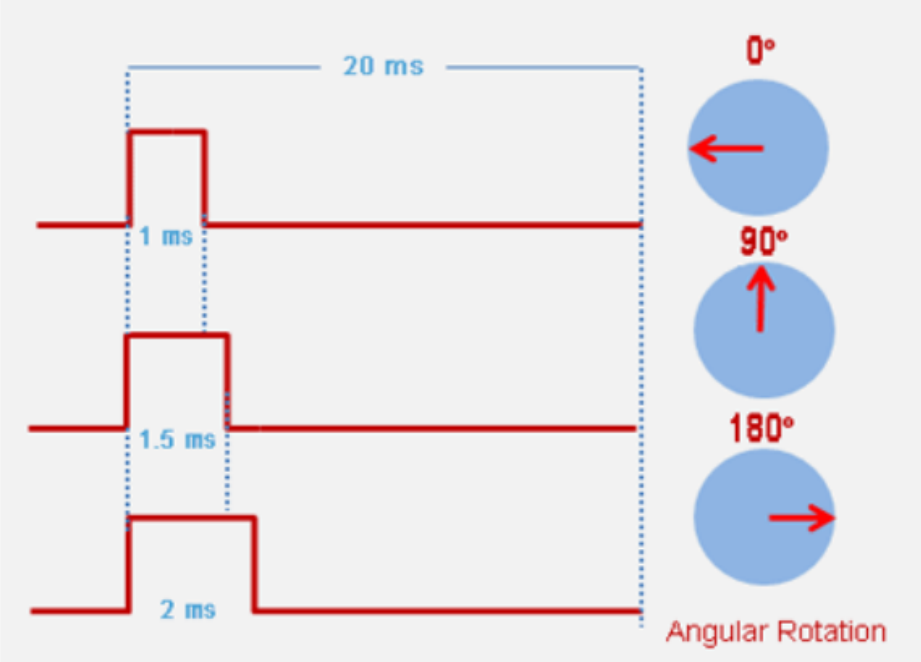
Она представља технику за добијање аналогних резултата дигиталним средствима. Користи се дигитално управљање да се направи правоугаони талас, тј. сигнал који се мијења између 0 и 5V. Овај образац укључивања и искључивања може симулирати напоне између потпуног укључивања (5 волти) и искључења (0 волти) промијеном дијела времена на којем сигнал проводи у стању потпуног укључења у односу на вријеме које сигнал проводи у стању искључења. Вријеме које је импулс укључен се назива ширина импулса. За добијање аналогних вриједности мијења се ширина импулса. Ако би се овај процес укључења-искључења дешавао довољно брзо са диодом, резултат би био стални напон између 0-5V који би контролисао осивјетљеност диоде.



Слика 17. Импулсно ширинска модулација

Главна предност коришћења *PWM* сигнала је то што сигнал остаје дигиталан цијелим путем од процесора до примаоца, тако да *D/A* конверзија није потребна. *PWM* сигнал се сматра дигиталним сигналом јер је у сваком тренутку времена ниво сигнала или на нули или на максимуму. Коришћењем чисто дигиталног сигнала ефекат шума се минимализује, што је велика предност у односу на аналогни пренос и контролу. Шум може да утиче на дигитални сигнал само ако је толико јак да промени логичку нулу на јединицу или обрнуто. Уколико нам је на располагању довољно велика фреквенција *PWM* сигнала, сваку аналогну вредност можемо представити као *PWM*.

3.6.4 Серво мотор – контрола



Слика 18. Контрола серво мотора

Серво мотор се контролише помоћу импулсно ширинске модулације. Он може да се окрене за 90° у оба смјера из своје нулте позиције. Серво мотор очекује пулс на сваких

20ms, а дужина тог импулса одређује колико ће се мотор окренути. Обично импулси трајања 1ms одговарају позицији 0°, 1.5ms позицији од 90°, а 2ms 180° као што је приказано на слици изнад.

3.6.5 Серво мотор – карактеристике и примјена

Серво мотори имају огромну примјену, а неке од области у којима се користе су:

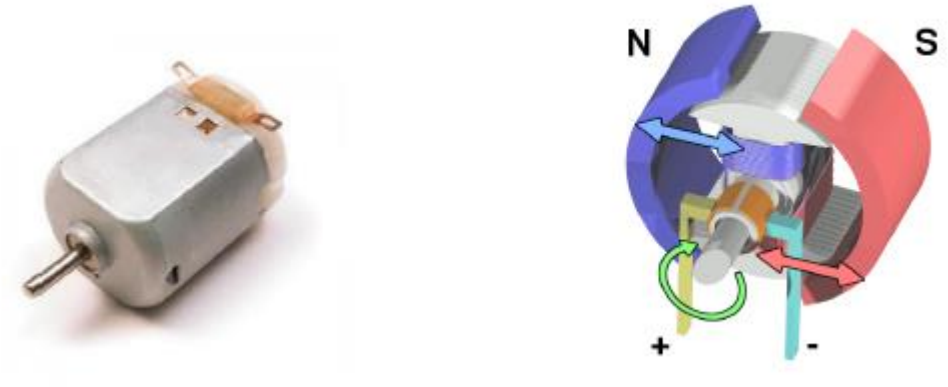
- Роботика
- Камере са аутофокусом
- Бродови
- Авиони
- Аутоматизоване браве
- Индустријски процеси
- Позиционирање антена
- Соларни системи – за помјерање плоча током дана

Карактеристике:

- Они су доста јефтини
- Постоје разне врсте у смисли величине и момента
- Веома су једноставни за контролисање.

3.7 *DC* мотори

Мотор једносмјерне струје је ротирајући екетрични уређај који претвара електричну енергију у механичну. Када се на крајеве индуктора, који се налази унутар мотора доведе једносмјерни напон, он производи магнетно поље. Унутар мотора је гвоздена осовина, умотана у калем жице. Ова осовина садржи два фиксна магнета, са обје стране што изазива одбојну и привлачну силу, заузврат, производећи обртни моменат.



Слика 19. Једносмјерни мотор

3.7.1 *DC* мотори – врсте

Постоје следеће врсте једносмјерних мотора:

- Мотори са четкицама
- Мотори без четкица
- Корачни мотори
- Униполарни корачни мотори
- Биполарни корачни мотори
- Серво мотори

3.7.2 *DC* мотори са редуктором

У овом пројекту је кориштен једносмјерни мотор са редуктором за покретање робота. Његов изглед је приказан на следећој слици

Слика 19. Једносмјерни мотор са редуктором

Мотори са редуктором су комбинација мотора и зупчаника. Додатак зупчаника има за циљ смањење брзине, док се повећава излазни моменат.

Најважнији параметри код овог типа мотора су:

- Брзина
- Обртни моменат
- Ефикасност

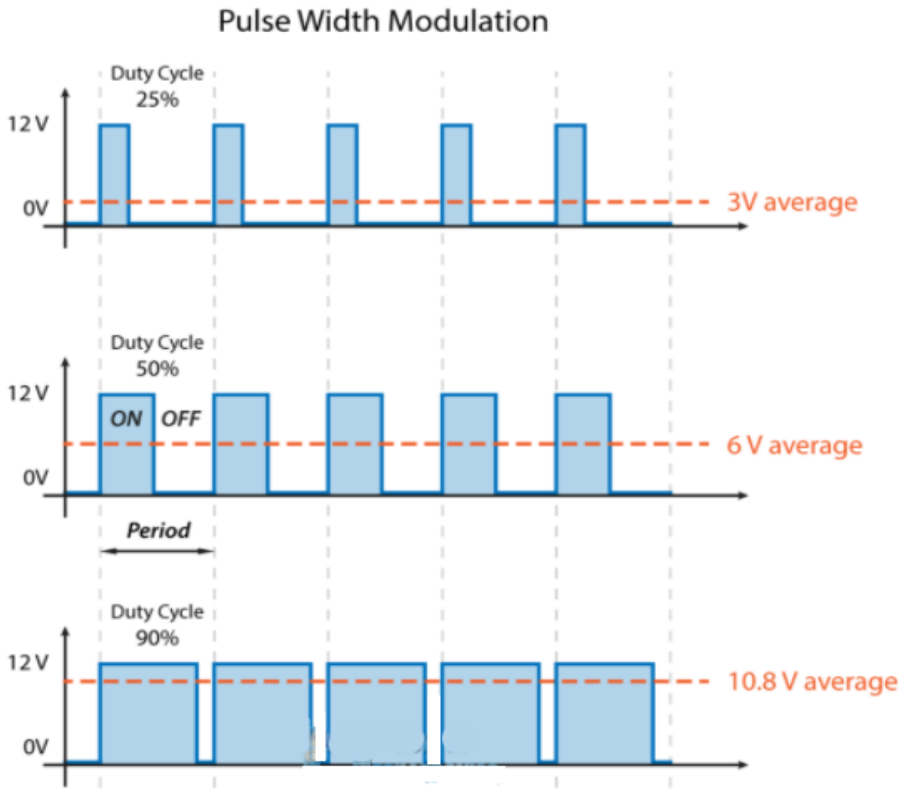
Да би обезбиједили најприкладнији мотор са редуктором за нашу примјену прво морамо израчунати:

- Оптерећење
- Брзину
- Захтјевани обртни моменат за нашу апликацију

3.7.3 Контролисање једносмјерних мотора

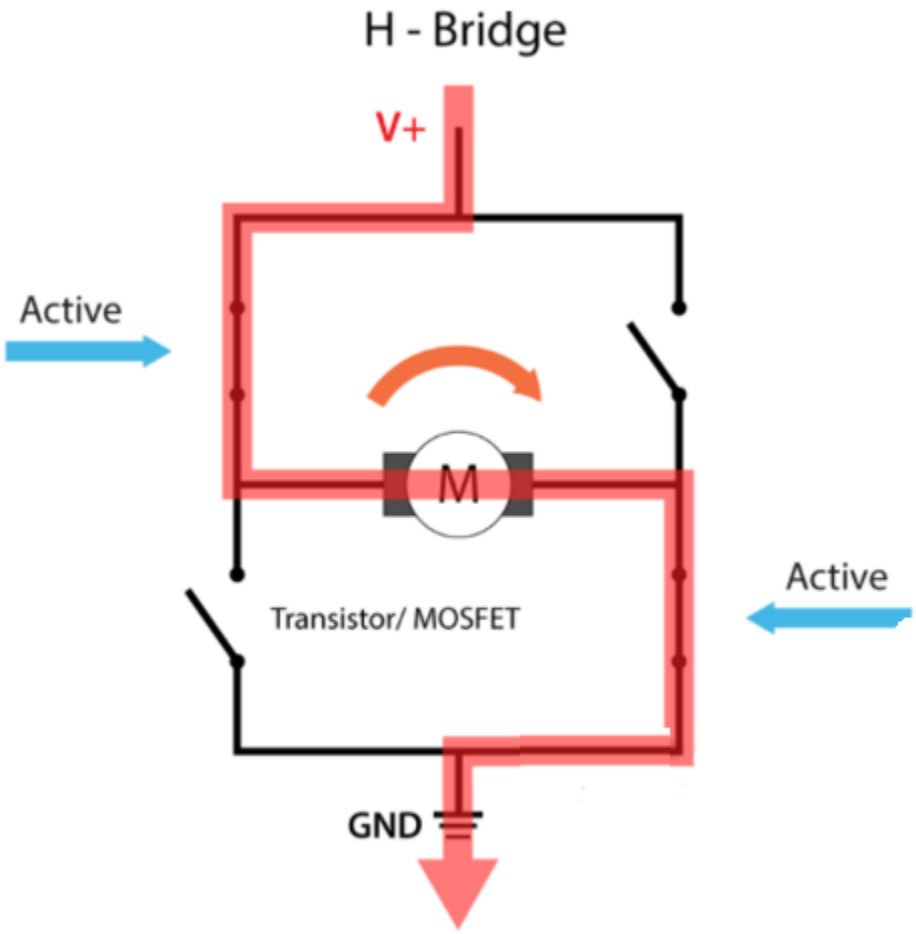
Контролисање брзине једносмјерног мотора је могуће контролом улазног напона мотора и најчешћа се за контролу користи *PWM* сигнал.

PWM, или импулсно ширинска модулација, као што је већ речено, је техника која омгућава да се подеси средња вриједност напона која се постиже довољно брзим укључивањем и искључивањем. Просјечни напон зависи од трајања укључености сигнала у односу на оно када је искључен.



Слика 20. Приказ *PWM* модулације

С друге стране, за контролисање смјера ротације, морамо да окренемо смјер струје кроз мотор и најчешћи начин за то је X мост. То је коло које садржи четири прекидача, транзисторе или мосфет, са мотором који се налази у средини. Активацијом два различита прекидача у исто вријеме, можемо да промјенимо смјер струје, а тиме и смјер окретања мотора.



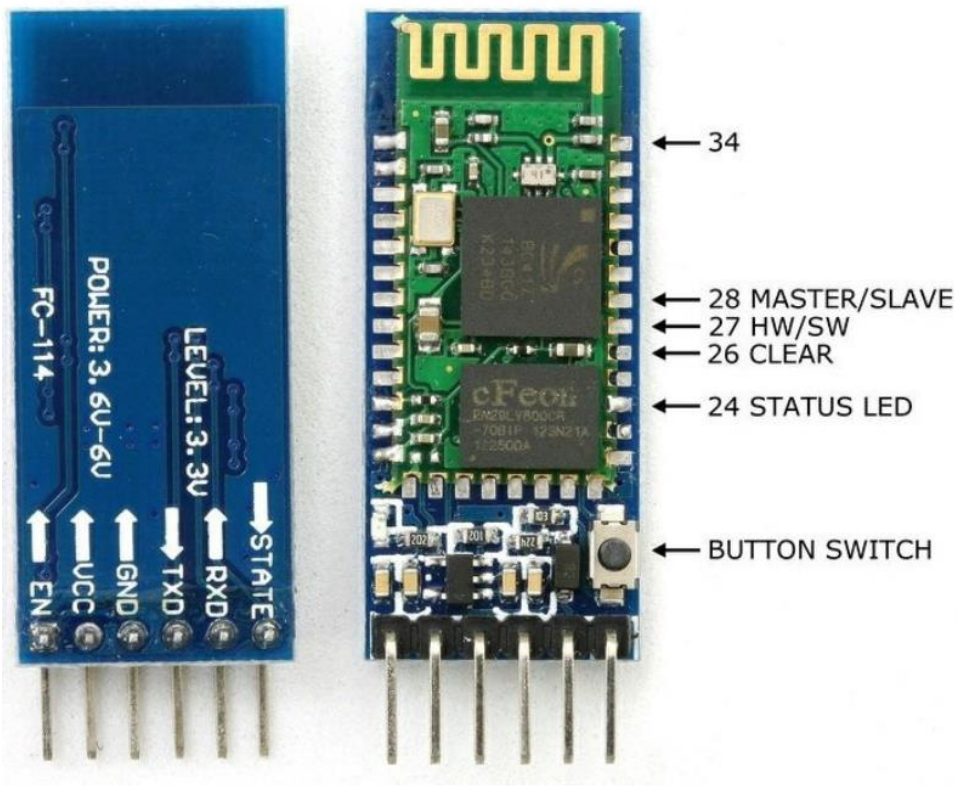
Слика 21. Приказ X моста

Примјеном *PWM* и X моста, омогућава се потпуна контрола једносмјерних мотора. Многи драјвери имају овакве особине, а такав је и претходно објашњен и кориштен у овом пројекту *L293D*.

3.8 HC – 05 Bluetooth модул

Bluetooth је један од сјасних видова бежичне конекције. Примјену је нашао у многим пољима. Он троши јако малу количину енергије. У овом пројекту кориштен је за комуникацију између телефона и робота.

HC – 05 може да комуницира на два начина. Модул комуницира уз помоћ *USART (Universal Synchronous/Asynchronous Receiver/Transmitter)* са брзином преноса 9600bit/s, а подржава и друге брзине преноса. Он може да ради у два режима. Један је *Data Mode,* а други *AT command mode.* Овај модул је *MASTER/SLAVE* модул. Иницијално је у *SLAVE* режиму и то може да се промјени само у *AT command* моду. *Slave* не може иницирати конекцију, али може прихватати друге везе.



Слика 22. *HC-05 Bluetooth* модул

У следећој табели су приказани прикључци и улога сваког од прикључака овог модула.

Конектор	**Функција**
ENABLE	Када је овај конектор на ниском нивоу, модул је искључен, што значи да неће успоставити комуникацију. Када се конектује на 3.3V, модулу се омогућава рад и тада је комуникација могућа
VCC	Напајање у границама од 3.3 – 5V
GND	Уземљење
TXD, RXD	Ова два конектора се понашају као *UART* интерфејс за комуникацију
STATE	Понаша се као статусни индикатор. Када модул није повезан са другим уређајем има низак ниво. Тада лед сијалица константно свјетлуца што означава да модул није конектован. Када се конектује на неки уређај, овај конектор добија висок ниво. У том случају лед свијетло свјетлуца са неким кашњењем, рецимо 2s, што означава да је модул упарен.
BUTTON SWITCH	Користи се да би модул пребацило у *AT command* режим. У овом режиму корисник може промијенити параметре модула, али само ако модул није конектован са другим уређајем.

Табела 4. Конектори *Bluetooth* модула и њихова функција

3.9 Хардверска конструкција робота

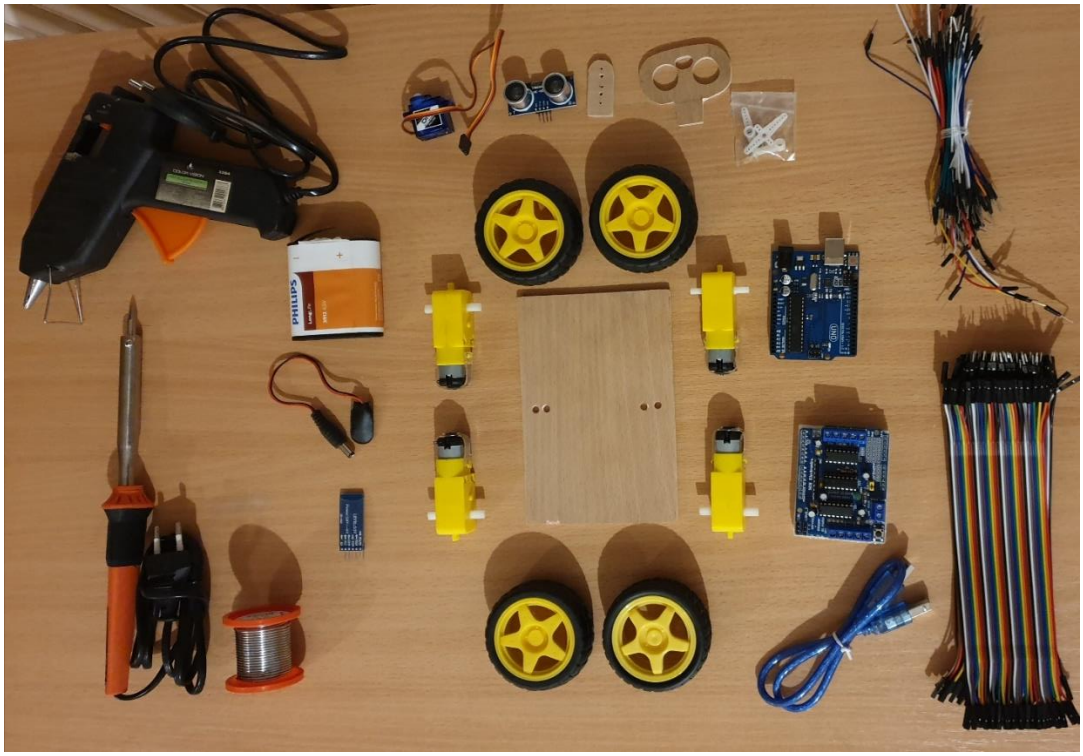
Након набавке свих неопходних компоненти, на ред долази повезивање и склапање хардверске конструкције. У наставку ће по корацима бити објашњен поступак склапања робота.

Хардверске компоненте коришћене у овом пројекту су:

- *Arduino Uno R3*
- *L293D* драјвер за моторе
- Ултрасонични сензор
- Серво мотор
- *Bluetooth* модул
- Четири мотора са редуктором
- Четири гуме
- Двије батерије *Li Ion 18650*
- Прекидач

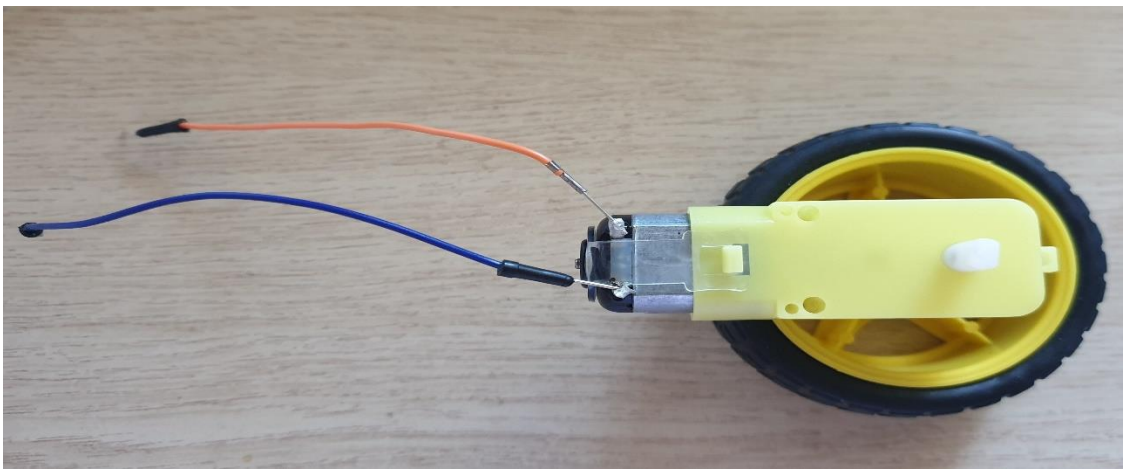
Материјали и алат потребни за израду:

- Лемилица и алат за лемљење
- Конектори за лемљење
- Каблови
- Пиштољ за лијепљење
- Дрвена плочица димензија 10 cm x 15 cm
- Држач за ултрасонични сензор
- Држач за батерије



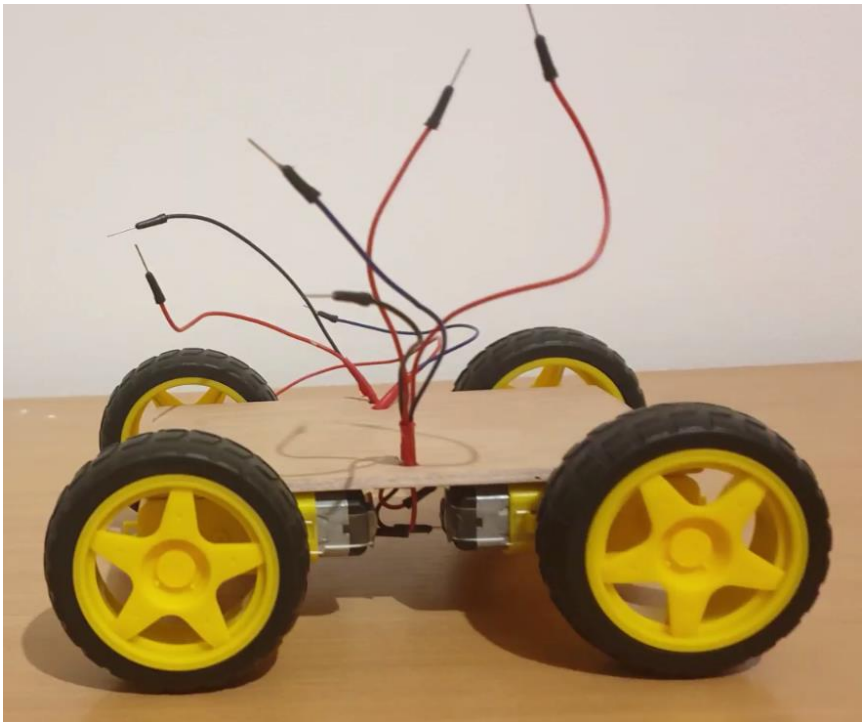
Слика 23. Компоненте и опрема пред почетак склапања

- **Први корак –** лемљење жица на моторе
    - У овом кораку повезујемо црвену жицу на онај крај који сматрамо позитивним и црну жицу на негативни, као што је приказани на слици испод.



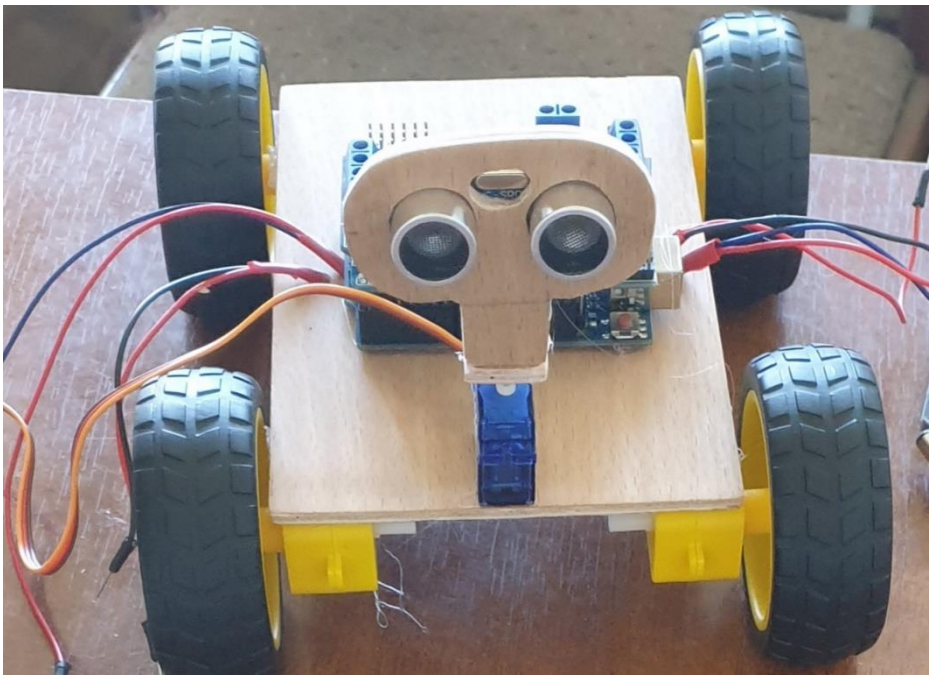
Слика 24. Лемљење жица на моторе

- **Други корак –** монтирање мотора на плочу и стављање гума
    - Мотори су залијепљени топлим лијепком на плочу. Резултат овог корака је изгледао као на слици:



Слика 25. Робот након реалицаије другог корака

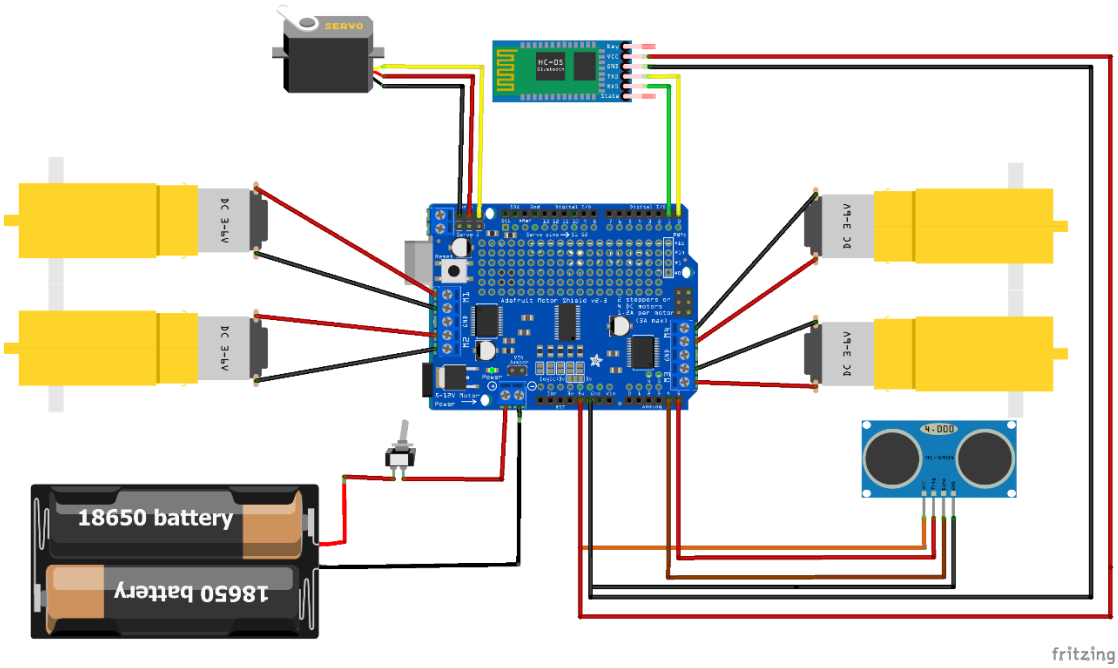
- **Трећи корак –** лемљење неопходних конектора на драјвер
    - С обзиром да су за повезивање свих сензора неопходни прикључци на драјверу који нису обезбјеђени фабрички, било је неопходно залемити додатне конекторе.
- **Четврти корак –** монтирање контролера и драјвера
    - У овом кораку прво је залијепљен контролер на средину шасије, а затим на њега постављен драјвер.
- **Пети корак –** постављање серво мотора и ултрасоничног сензора
    - Прво се топлим лијепком врши поставка серво мотора, затим се на њега уврће држач за ултрасонични сензор и на крају се лијепі ултрасонични сензор за држач.



Слика 26. Изглед робота након петог корака

- **Шести корак –** монтирање држача за *Bluetooth* модул
    - У овом кораку је топлим лијепком залијепљен држач и на њега монтирани каблови за повезивање овог модула
- **Седми корак –** Израда држача за батерије, формирање кола са прекидачем и монтирање на шасију
    - Држач за батерије је направљен од пуњача за батерије. Прво је разлемљено коло које је служило за пуњење батерија, а затим на крајеве залемљене жице за плус и минус.
    - Када је направљен држач, следећи корак је био повезивање са прекидачем и на крају монтирање на шасију.

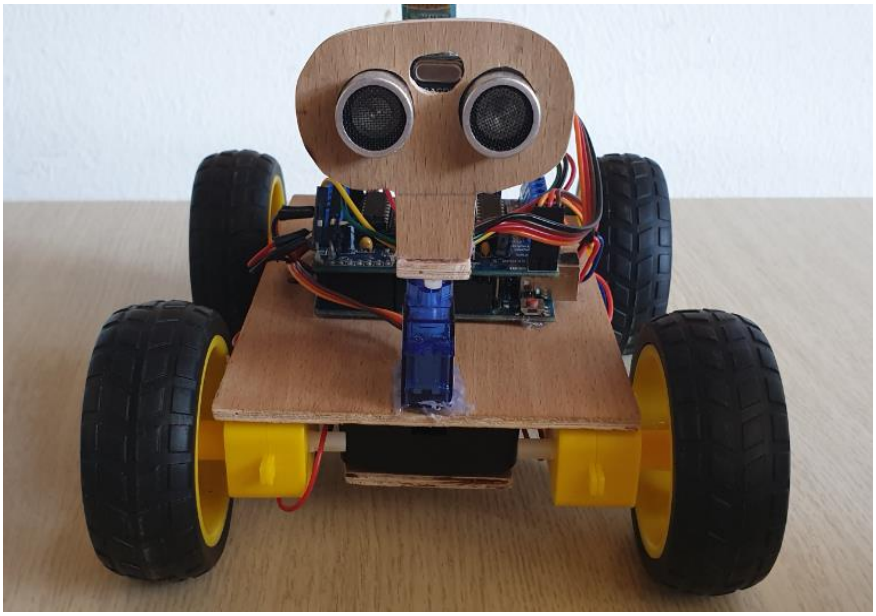
➢ **Осми корак –** Повезивање компоненти

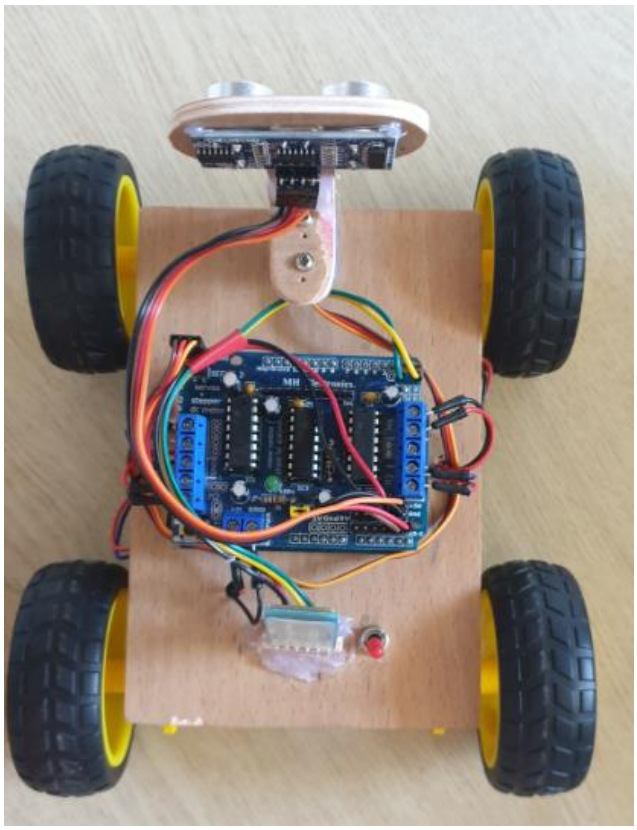


Слика 27. Коло за повезивање компоненти

- Први корак је био повезивање кола са батеријама на конекторе за спољашно напајање које се налази на драјверу. Напајање је заједничко за контролер и за драјвер. Као што је већ речено користе се двије батерије типа *Li Ion 18650,* од 3.7V и имају довољно енергије за покретање цијелог робота.
- Затим је извршено повезивање мотора, тако што су у одговарајуће прикључке драјвера приврнуте жице.
- Послије тога је услиједило конектовање серво мотора и ултрасоничног сензора.
- На самом крају је извршено повезивање конектора за напајање *Bluetooth* модула, а затим су његови *RX* и *TX* респективно залемљени за *TX* и *RX* контролера.

Након свих корака робот изгледа као на следећим сликама





Слика 28. Коначни изглед робота

4 Софтверска реализација

Софтверска реализација се састоји из два дијела. Први дио представља израду Андроид апликације, а други израду Ардуино апликације.

4.1 Принцип рада робота

Овај робот је пројектован тако да има два режима управљања. Први режим је гласовна контрола, а други даљинско управљање. Роботом се управља бежично, помоћу мобилног телефона и *Bluetooth* технологија. Апликација која се назива *RoboControl* омогућава повезивање са роботом и његово управљање у наведеним режимима.

Гласовне команде које су омогућене су редом: „крени“, „напред“, „напријед“, „иди напр(ij)ед, „назад“, „иди назад“, „л(ij)ево“, „десно“, „брже“, „спорије“.

Даљинским управљањем омогућено је кретање напријед, назад, као и окретање робота лијево и десно.

Овај робот има могућност детектовања препрека. У режиму гласовних команди он има алгоритам за обилазак, док у режиму даљинског управљања робот детектује препреку и зауставља се, са онемогућавајући кретање напријед све док је препрека испред њега. Алгоритам за обилазак ради тако што када удаљеност од неког предмета постане довољно мала, робот се зауставља, гледа лијево и десно како би проценио која страна за обилазак препреке је боља. Када је одредио страну, враћа се уназад, поново зауставља, окреће и прати дужину препреке. Када утврди да је прошао препреку, робот се зауставља, окреће и наставља напријед и поново прати дужину препреке са те стране. Када је и ту страну прошао, зауставља се, окреће и враћа на стару путању која му задата. Уколико робот добије другу команду приликом обиласка, он ће имати већи приоритет. Такође, алгоритам обезбјеђује да уколико дужина препреке буде изнад одређеног нивоа, та препрека буде класификована као немогућа за обилазак са овим ресурсима. У ове препреке спада, на примјер зид. У том случају робот се окреће за 180° и зауставља, чекајући следећу команду. Још једна ставар

која је уграђена у алгоритам је и детекција препрека типа „тунела“, гдје робот нема могућност скретања ни лијево ни десно, како би обишао препреку. Тада се робот врати уназад и застави, очекујући следећу команду.

4.2 Израда Андроид апликација

4.2.1 *MIT App Inverter* окружење

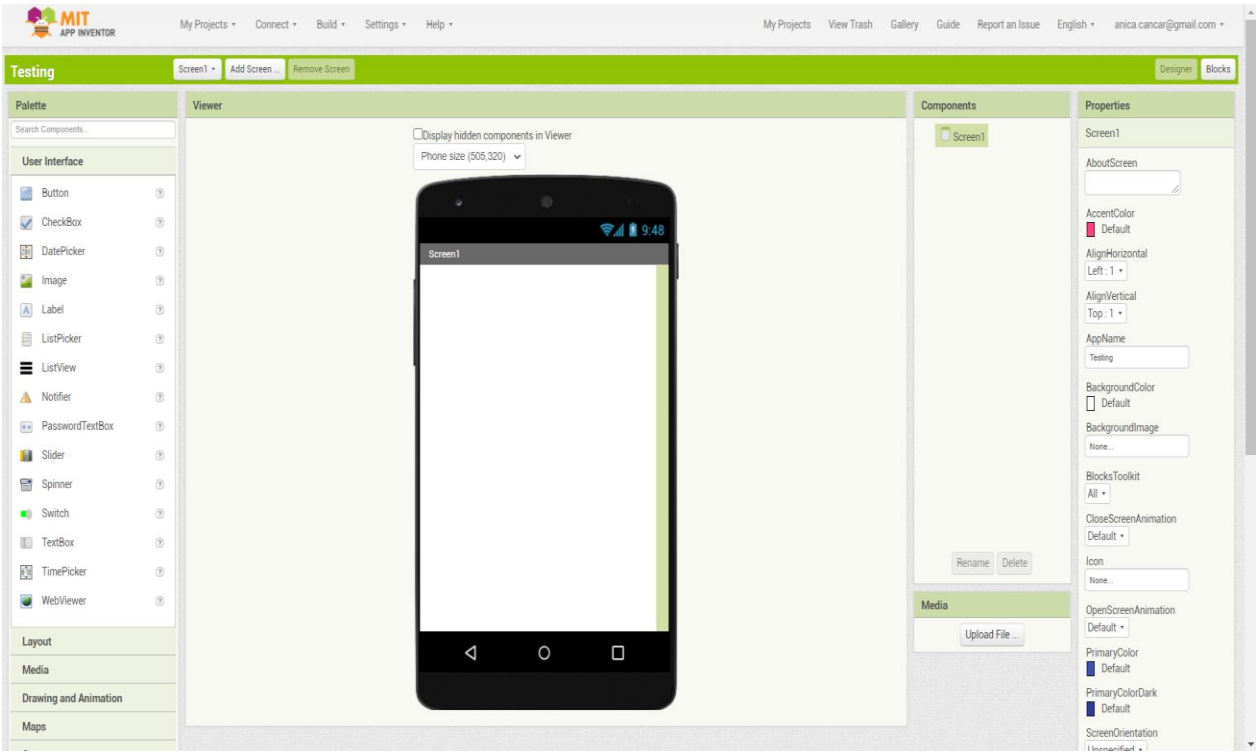
Андроид апликација је израђена помоћу развојног окружења које се назива *MIT App Inventor.* Ово окружење је интегрисано у веб апликације. Првобитно га је обезбиједио Гугл, а тренутно га одржава *Massachusetts Institute of Technology (MIT).* Ово окружење омогућава новајлијама у рачунарском програмирању да креирају софтвер за два оперативна система *Android* и *iOS.* Бесплатно је и представља софтвер отвореног кода.

Окружење користи графички кориснички интерфејс(енг. *GUI)* који је веома сличан програмским језицима *Scratch* и *StarLogo,* што омогућава корисницима да превуку визуелне објекте да би направили апликацију која може да се покрене на паметним уређајима.

Интерфејс овог окружења укључује два главна едитора:

- Уређивач дизајна
  - Дизајнер има могућност превлачења блокова који представљају кориснички интерфејс
- Уређивач блокова
  - У овом дијелу дизајнери могу да изложе логику својих кодова помоћу блокова у боји који се спајају попут дијелова да би описали програм.

Да би олакшало тестирање кода, ово окружење пружа сопствену апликацију која се назива *App Inventor Companion,* коју програмери могу користити за тестирање и прилагођавање понашања своје апликације у реалном времену. Након израде апликације могуће је скинути и користити на паметним уређајима.



Слика 29. *MIT App Inverter* окружење

4.2.2 Израда *RoboControl* апликације

- **Графичка реализација**

Апликација се састоји из једног екрана на коме се налазе сви неопходни елементи.

На почетном екрану се налази:

- Насловна слика
- Дугме за *Bluetooth* конкецију
- Лабела која приказује статус конекције
- Дугме за гласовну контролу
- Дугме за даљинску конторолу

Притиском на дугме за гласовну контролу елементи који су се налазили на екрану, а појављују се следећи елементи:

- Дугме за започињање гласовне команде
- Лабела која приказује команду
- Дугме за повратак за почетни екран

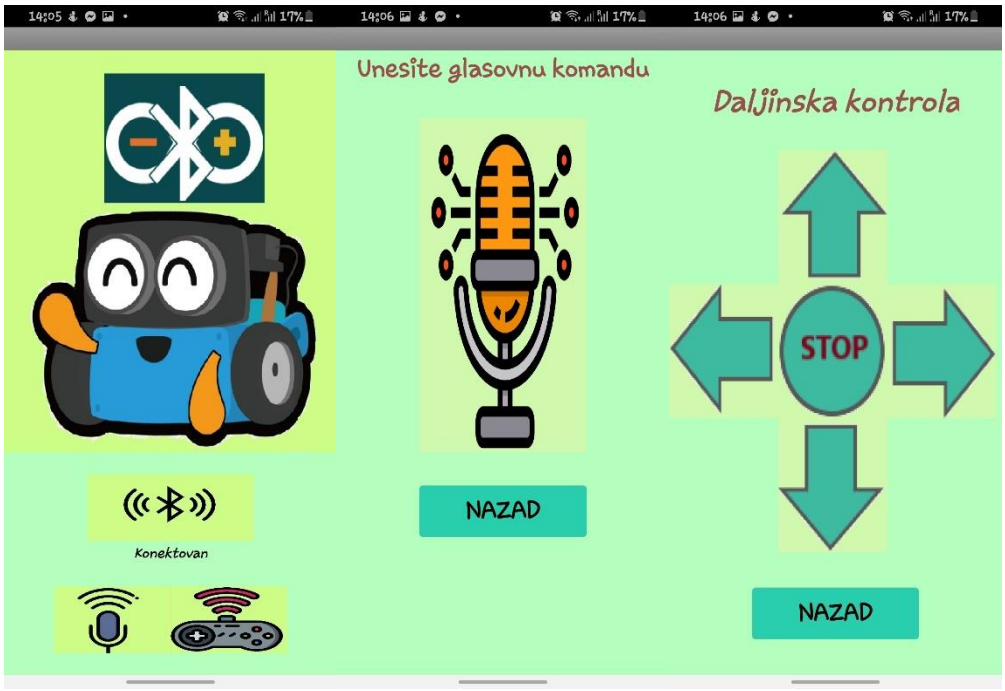
Притиском на дугме за даљинску контролу појављују се следећи елементи:

- Дугме за напријед, назад, лијево, десно, стоп
- Дугме за повратак на претходни екран

Такође програм садржи и неке компоненте које су невидљиве, а то су:

- *Bluetooth client* – служи за успостављање *Bluetooth* конекције
- *Speach recognizer* – служи за конверзију говора у текст
- *Clock*

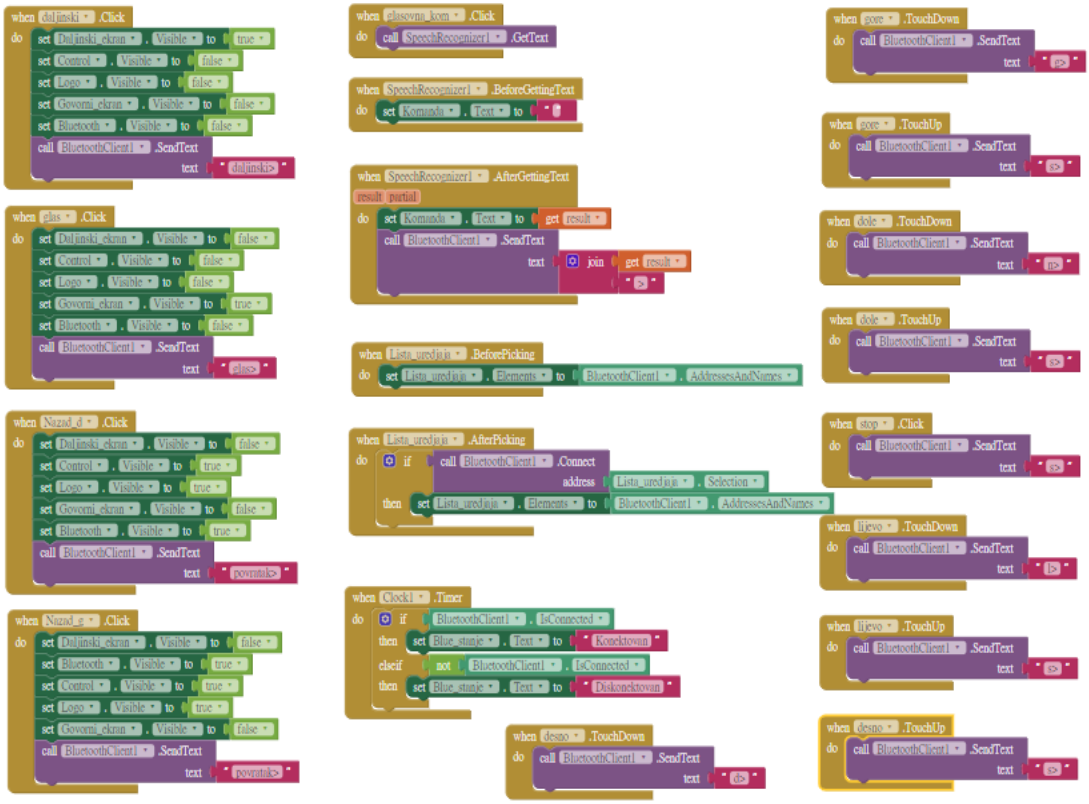
Апликација изгледа као на следећим сликама



Слика 30. *RoboControl* апликација

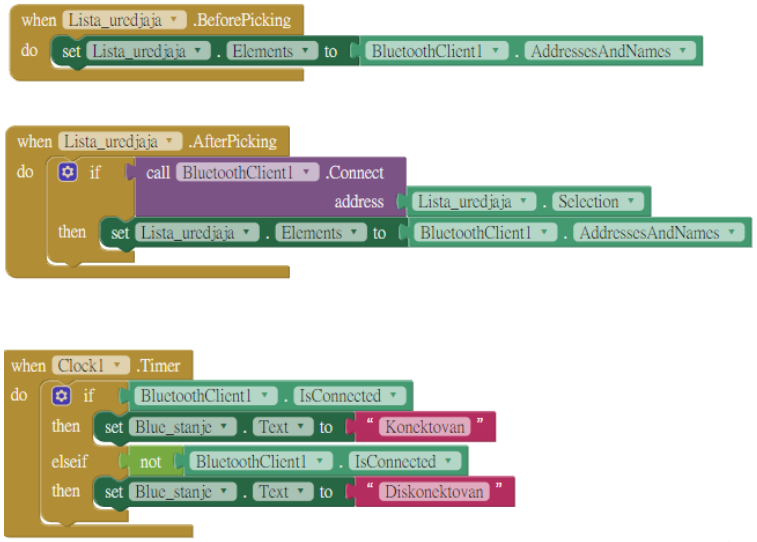
- **Логичка реализација**

Код у *MIT App* окружењу изгледа као на следећој слици:



Слика 31. Код за израду апликације

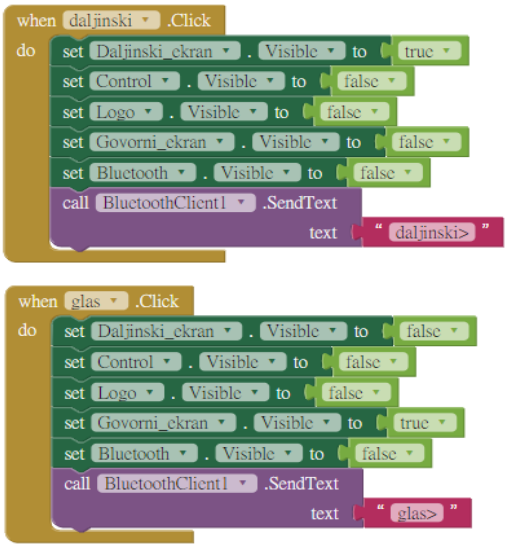
Код за *Bluetooth* реализацију



Слика 32. *Bluetooth* реализација

У програму постоји листа која служи за приказвање свих доступних уређаја за повезивање. Са горње слике видимо да се прије одабира уређаја за конекцију у њу смјесте сви доступни *Bluetooth* уређаји, а након одабира позива се *Bluetooth* клијент и успоставња веза са уређајем. Ако се успјешно конектовао у лабели се исписује „Конектован“, иначе се исписује „Дисконектован“.

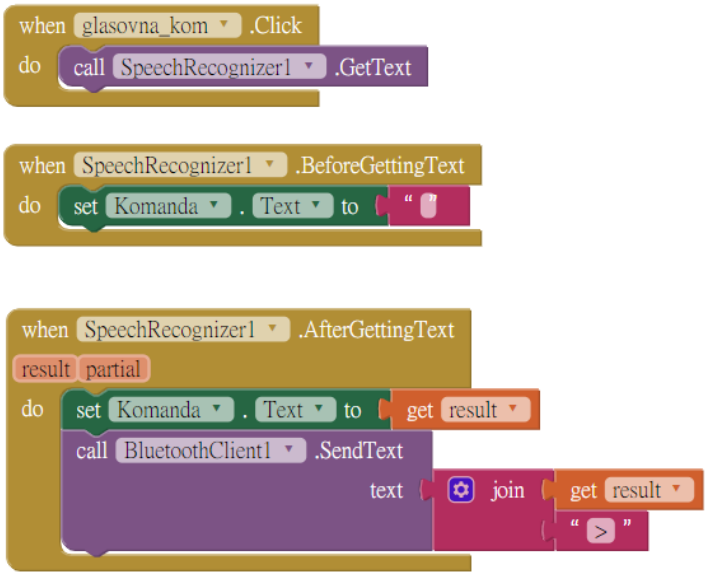
Код за одбирање режима управљања:



Слика 33. Одабирање режима управљања

Код изнад приказује логику када се притисне дугме за даљинску или гласовну контролу. Све компоненте које су неопходне за одређени режим постају видљиве, док остале постају невидљиве. Такође, путем *Bluetooth* конекције се шаље текстуална порука која означава режим рада. Карактер „>“ означава крај стринга.

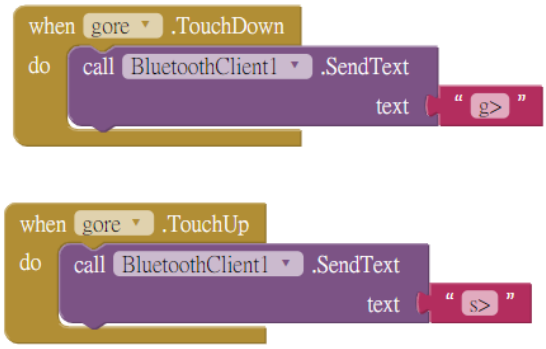
Код за гласовну контролу:



Слика 34. Код за гласовну контролу

Обрада говора започиње притискањем дугмета, када се позива компонента за конверзију говора у текст, а након добијања резултата, путем *Bluetooth* конекције, шаље се текстуална порука контролеру, која означава конвертовани говор.

Код за даљинску контролу:



Слика 35. Код за даљинску конторлу

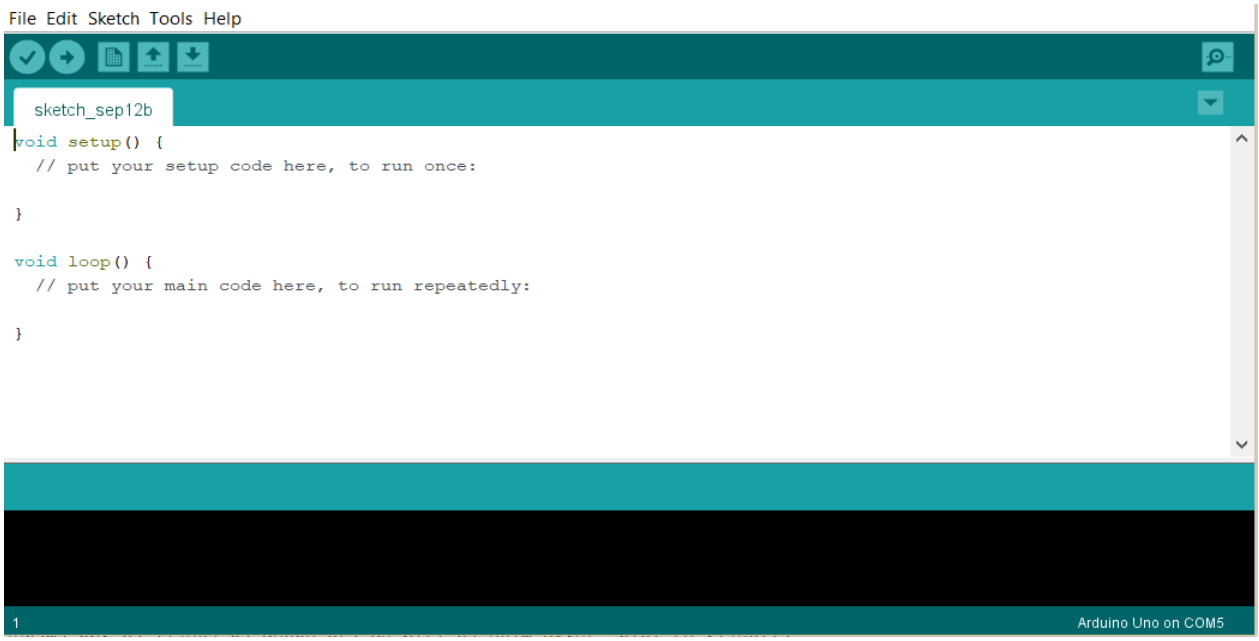
На слици је приказан начин даљинског управљања за кретање унапријед. Овај код говори да докле год је притиснуто дугме, апликација шаље „g>“, односно робот иде напријед, а чим се дугме пусти он шаље „s>“, што означава заустављање. На исти начин се реализују и остале даљинске команде.

4.3 Израда Ардуино апликације

4.3.1 Ардуино развојно окружење

Ардуино развојно окружење је апликација која је је вишеплатформска (*Widnows, macOS, Linux)* написана на функцијама из *C* и *C++*. У себи садржи уређивач кода са могућностима претраживања и замјене текста, аутоматског увлачења и пружа једноставне механизме комајлирања и отпреме кода једним кликом на Ардуино плочу. Такође садржи подручје за поруке, текстуалну конзолу, траку са алаткама са тастерима за уобичајене функције и менију.

Програми написани у овом окружењу се називају скице. Ове скице имају екстензију .ino. Свака скица има најмање два дијела. То су *setup()* и *loop()* блокови. На само почетку рада контролера се позива функција *setup()* и то само једном када се контролер рестује или укључи на напајање, а функција *loop()* се након тога стално позива са одређеном периодом током рада контролера.



Слика 36. Ардуино развојно окружење

4.3.2 Програмирање Ардуина

Овај код користи три библиотеке. Двије од њих се морају скинути да би се програм могао компајлирати. Прва је *Motor Shield* библиотека коју је издала компанија *Adafruit* која садржи функције за управљање моторима, а друга је *NewPing* која садржи функције за рад са ултрасоничним сензором. Трећа библиотека која се налази већ унутар доступних библиотека је *Servo* библиотека која садржи функције за рад са серво мотором.

На почетку самог кода, прво укључујемо наведене библиотеке у код.

```
#include <AFMotor.h>
#include <NewPing.h>
#include <Servo.h>
```

Следећи корак представља дефинисање конектора и максималне удаљености ултрасоничног сензора.

```
#define ECHO A4
#define TRIG A5
#define MAX_DIST 300
```

Када је завршен овај корак на реду је иницијализација мотора и ултрасоничног сензора.

```
// DC motori
AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
AF_DCMotor motor4(4);

// Servo motor
Servo servo;

// Ultrasonicni senzor
NewPing sonar(TRIG, ECHO, MAX_DIST);
```

Сада прелазимо на setup() функцију.

```
void setup() {
 // Pokretanje protokola i brzina prenosa podataka
Serial.begin(9600);

  // Definisanje servo motora
  servo.attach(10);
  servo.write(90);

  // Zaustavljanje svih motora na pocetku
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
```

```
  //Postavljanje inicijalne brzine
  Speed(150);
  pom = 0;
  prepreka = 0;
  okret = 0;
  komanda = 0;
  mode = "";
  kom = ' ';
  first = 0;
}
```

У овој функцији започињено серијску комуникацију и постављамо да брзина преноса података буде 9600bit/s. Иницијализујемо серво мотор и позиционирамо га на 90°, тако да за ову рализацију робота буде у нултом положају. Заустављамо окретање свих мотора. Иницијална брзина је 150(може максимално 255 због импулсно ширинске модулације).

- **Функције коришћене у овом програму**

У наставку ће бити ријечи о функцијама које су имплементиране у овај код и која је њихова улога.

На самом почетку израде кода било је неопходно направити функције које би означавале кретање робота напријед, назада, лијево, десно и стоп.

Функције напријед и назад су реализоване тако што се сва четири точка окрећу напријед, односно назад.

```
// Kretanje robota unaprijed
void Forward()
{
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
//Kretanje robota unazad
void Backward()
{
  motor1.run(BACKWARD);
```

```
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}
```

Скретање лијево, односно десно је реализовано тако што се два точка окрећу лијево, а друга два десно. Ово окретање траје 350ms, јер на тај се начин робот са овом брзином окрене за око 90°. Заустављање се врши тако што се сви мотори зауставе.

```
//Kretanje robota lijevo
void Left()
{
  Speed(255);
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
  delay(350);
  Speed(150);
  okret = 1;
}

//Kretanje robota desno
void Right()
{
  Speed(255);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
  delay(350);
  Speed(150);
  okret = 1;
}
// Zaustavljanje
void Stop()
{
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
```

```
  motor4.run(RELEASE);
}
```

Након ових функција треба рећи нешто о начину детекције препреке. Постоје двије функције које имају улогу у овом процесу. Прва функција је функција *udaljenost()*. Улога ове фунцкије је да врати процијену удаљености препреке. У њој се користи фунцкија из библиотеке *NewPing* која се назива *sonar.ping_median(iterations).* Ова функција шаље 5 ултрасоничних звукова и као резултат враћа медијану потребног времена. Треба нагласити да је на почетку кориштена функција *sonar.ping_cm(),* која је враћала растојање од предмета, али је за овај проблем давала лошу тачност, па је због тога изабрана првобитна функција.

```
float udaljenost()
{
  float duration = sonar.ping_median(5); // Salje 5 pingova i onda
uzima medijanu, radi bolje tacnosti
  float udaljen = (duration / 2) * 0.0343;
  return udaljen;
}
```

Повратна вриједност ове функције се рачуна по формули (1).

Друга функција која има улогу у детекцији препреке је функција Test(). Ова функција провјерава да ли је удаљеност објекта мања од 30cm и ако јесте враћа јединицу, иначе враћа нулу. Иако је 30cm доста велика вриједност, због несавршности уређаја, показала се као оптимална за заустављање препреке, јер се робот у пракси увијек заустави на мањој удаљености.

```
// Provjera udaljenosti od prepreke
int Test()
{
  if (udaljenost() < 30)
  {
    return 1;
  }
  return 0;
}
```

Следећи корак у изради овог пројекта је био детектовање поруке која се шаље преко *Bluetooth* – а. То је урађено тако што, када се појаве неки бити на улазу, програм крене да чита карактер по карактер, све док не дође до краја поруке што је означено са „>“. Овдје треба напоменути да је прво кориштена функција *Serial.readString()* умјестно *Serial.read(),* која је читала цијелу поруку, не захтјевајући никакву додатну обраду. Проблем са овом функцијом је био тај што је она била превише спора и уносила је кашњење при управљању роботом, посебно при даљинском управљању. Овај дио алгоритма приказан је у наставку.

```
while (Serial.available())
{
  pom = 0;
  prekid = 0;
  okret = 0;
  delay(10);
  char c = Serial.read();
  if (c == '>')
  {
    break;
  }
  q += c;
}
```

Одабиром одговарајућих дугмића на Андроид апликацији, путем *Bluetooth* конекције се шаље информација о режиму рада. У коду се то смјешта у глобалну промјењиву *mode.* Када програм зна у ком је режиму управљања онда он врши процесирање следеће поруке која стиже на одређени начин. Овај дио кода је приказан у наставку.

```
if (p != q && q != "")
{
  change = 1;
  p = q;
}

if (p == "glas" || p == "daljinski")
{
```

```
    mode = p;
    p = "";
    first = 0;
  }

  if (p == "povratak")
  {
    p = "";
    poruka = "";
  }
```

- **Управљање гласом – програмерски дио**

Уколико промјењива *mode* означава да се налазимо у гласовној контроли, онда се прво провјери да ли следећа порука означава команду за убрзавање или успоравање. Уколико стигне команда за убрзавање или успоравање, брзина се повећа, односно смањи за 20, а команда за смјер се не мијења, тј. остаје иста као што је и била прије стизања поруке за брзину. Уколико порука није за промјену брзине, онда се она смјешта у пројењиву *poruka,* која означава команду за кретање робота.

Програм даље ради тако што тежи да испуни ту команду, али прије уласка у извршење команде, прво позива функцију *Test(),* која означава да ли је препрека на путу или не. Уколико се детектује препрека, робот улази у алгоритам за обилазак препреке, а ако не, онда извршава задату команду. Овај дио кода је ради прегледности приказан у додатку.

- **Алгоритам за обилазак**

Алгоритам за обилазак се позива уколико је детектована препрека у гласовном режиму контроле.

Програм тада креће у извршавање функције која се назива *obilazak()*. Робот се прво зауставља, затим се окреће лијево и мјери удаљеност од најближег предмета. Следећи корак је окретање десно и мјерење удаљености с десне стране. Након овога робот враћа ултрасонични сензор у нулти положај и одлучује о даљим корацима. Уколико стигне нова команда, имаће највиши приоритет и она ће бити испоштована, у супротном робот ће провјерити која од страна је боља, тако што гледа која има већу удаљеност од предмета и на

тај начин ће одлучити са које стране ће радити обилазак. Уколико се испостави да су с обје стране препреке, он ће се вратити уназад и зауставити, чекајући следећу команду. Ове функционалности су обезбјеђене функцијом чији код се, ради прегледности, се налази у додатку.

Најбитнију улогу у алгоритму обиласка има функција која се назива *prolazak_prepreke()*. Она се, у зависности да ли се алгоритам одлучио за обилазак с лијеве или десне стране, окреће за одређени број степени и прати препреку. Све док ултрасонични детектује удаљеност мању од 40cm, робот иде напријед. Због сигурности обиласка, остављено је да и након што прође препреку, робот иде још 250ms напријед како би осигурали потпуни пролазак препреке.

У случају појаве нове команде, она ће имати највећи приоритет. У овој функцији се води рачуна и о томе да ли је праћење дужине препреке веће од 4s. Уколико се детектује таква препрека, алгоритам је сматра несавладивом, окреће се за 180° и зауставља, чекајући следећу команду. Овај дио је осмишљен првенствено због детекције зидова у затвореним просторима. Код ове функције је, ради прегледности, се налази у додатку.

Функције *obilazak_s_lijeve()* и *obilazak_s_desne()* имају исте функционалности па ће овдје бити објашњена функција *obilazak_s_lijeve()*.

Након уласка у обилазак с лијеве стране, робот се прво врати 250ms уназад, како би обезбиједио довољан простор за окретање и обилазак. Затим се окреће лијево и позива функцију *prolazak_prepreke()* која је објашњена изнад. Овдје креће рачунање времена које је робот провео у обиласку препреке, што за овај алгоритам представља начин памћења претходне путање. Након завршетка овог дијела алгоритма, робот се окреће десно и иде напријед 500ms, затим поново позива функцију *prolazak_prepreke()* како би обишао и ту страну препреке. Након завршетка тог дијела, робот се окреће десно и иде напријед онолико времена колико је трајао пролазак препреке са горње стране препреке. Ово омогућава да се робот врати на првобитну путању. Након овога он се окреће лијево и наставља са оном командом која је била задата прије уласка у обилазак препреке. Уколико поново наиђе на препреку, поново ће покушати да је обиђе. Код ове функције је, ради прегледности, се налази у додатку.

- **Управљање помоћу даљинског – програмерски дио**

Уколико је промјењива *mode* у даљинском режиму, поступак рада је да се из стринга послате поруке узме само први карактер, што представља команду за овај режим рада.

Свака команда има свој карактер

- Напријед – g
- Назад – n
- Лијево - l
- Десно - d
- Стоп - s

Користи се *swich* блок да би се одредило у ком смјеру ће робот ићи. Уколико робот детектује препреку, све док је она детектована онемогућено је кретање унапријед, док су остале фунцкије могуће како би се ручно та препрека заобишла.

Програмски код је дат у наставку.

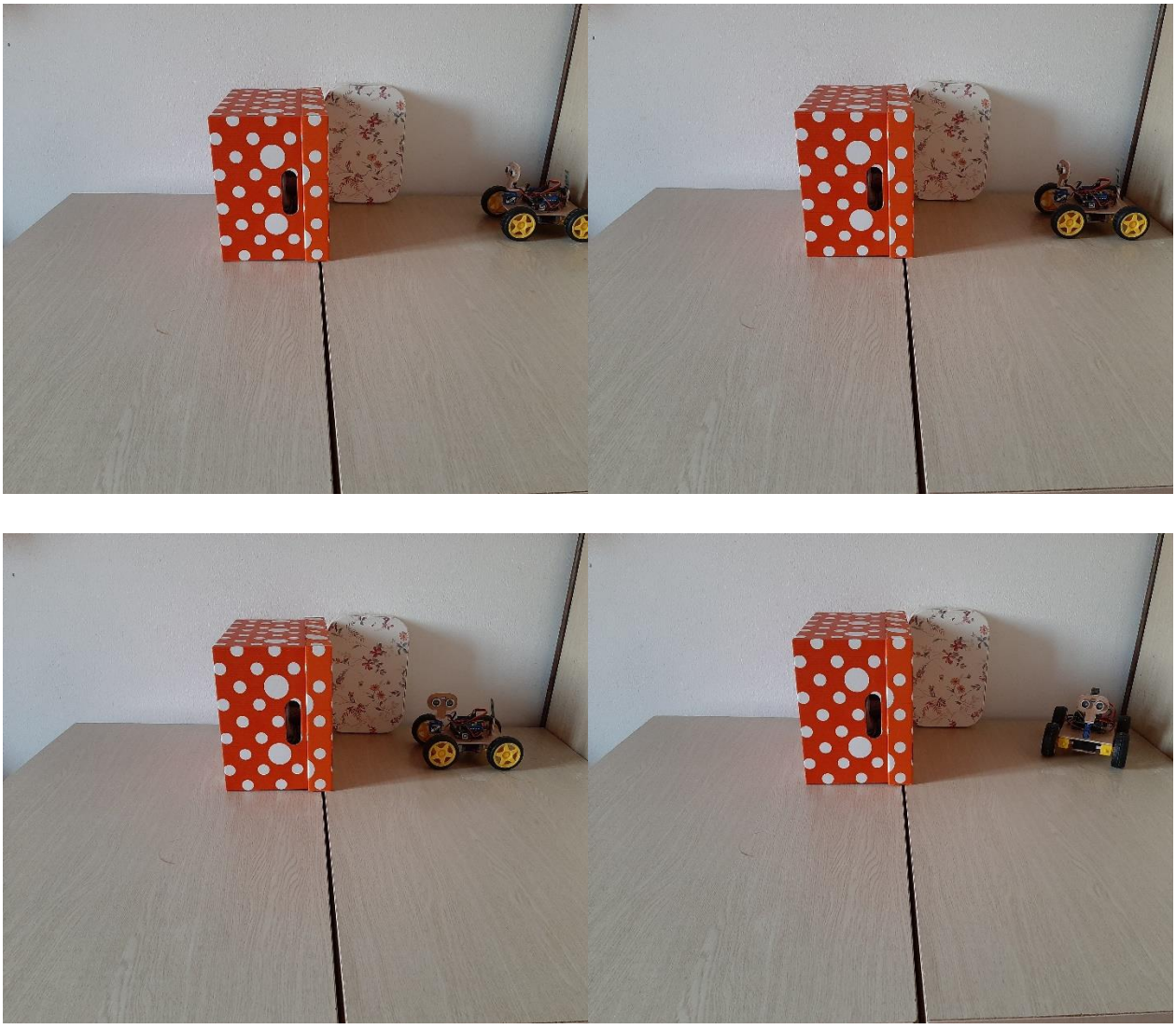
```
    else if (mode == "daljinski")
{
  unsigned int sto;
  unsigned int s;
  //      if (Serial.available() > 0)
  //      {
  Serial.println("p je: " + p);
  if (p.length() > 0)
  {
    kom = p[0];
  } else
  {
    kom = ' ';
  }
  if (Test() == 1 && kom == 'g')
  {
    Stop();
    kom = ' ';
  } else
  {
    Stop();
```

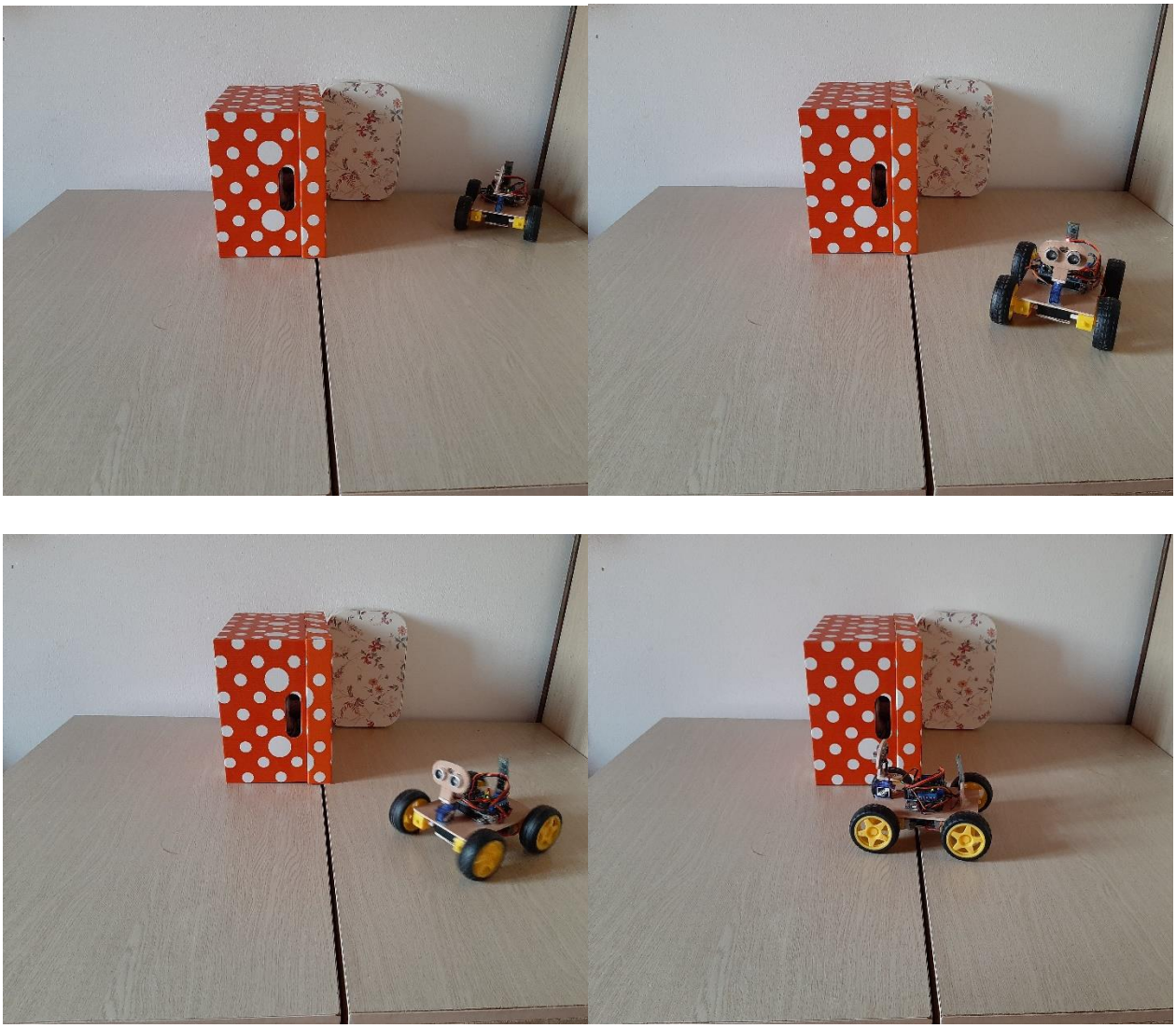
```
      switch (kom) {
        case 'g':
          Forward();
          break;
        case 'n':
          Backward();
          break;
        case 'l':
          motor1.run(BACKWARD);
          motor2.run(BACKWARD);
          motor3.run(FORWARD);
          motor4.run(FORWARD);
          break;
        case 'd':
          motor1.run(FORWARD);
          motor2.run(FORWARD);
          motor3.run(BACKWARD);
          motor4.run(BACKWARD);
          break;
        default:
          Stop();
          break;
      }
    }
  }
  //      }
  change = 0;
}
```

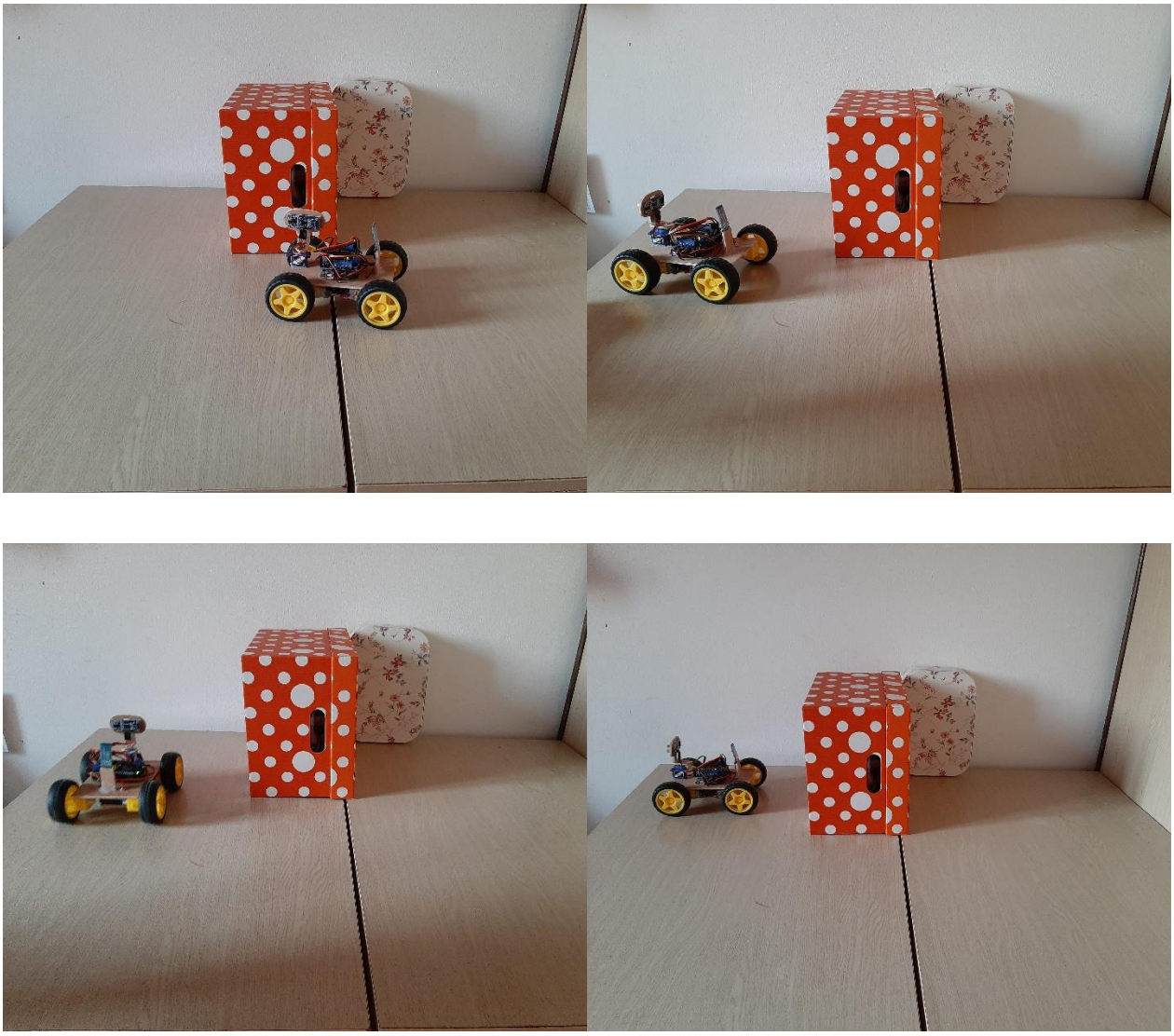
5 Приказ рада робота

У овом одјељку ће бити приказан сликовити приказ рада робота у различитим облицима препрека.

- Приказ робота при обиласку препреке







Слика 37. Приказ алгоритма обиласка препреке

На сликама се по корацима види приказ алгоритма који је описан изнад.

- Приказ рада робота при детекцији зида

Слика 38. Приказ детекције зида

Робот након што увиди да је прошло 3s почетка препреке, окреће се за 180° и зауставља.

- Приказ рада робота при детекцији препреке типа тунела

Слика 39. Приказ рада робота при препреци типа тунела

Као што се види са слика, робот се након што увиди да не може обићи препреку на стандардан начин, враћа мало уназад и зауставља, очекујући следећу команду.

6 Закључак и унапријеђења

У савременом свијету технологија, роботика представља неизоставну област. У мањој или већој мјери се све више јавља у свакодневном животу човјека. Роботе данас можемо видјети у многим сферама, које нису нужно индустријске или научне. Свакако, у будућности можемо очекивати још већу потребу за роботима, а самим тим и развој ове области.

Циљ овог пројекта је био потупуна хардверска и софтверска реализација мобилног робота, који има могућност управљања гласом или даљинским. При изради робота кориштено је више различитих хардверских компоненти, које су обухватиле различите сензоре, моторе и драјвере за моторе. Њихов квалитет је у великој мјери утицао на рад и прецизност овог робота. За израду управљања кориштени су софтверски алати *MIT App Inventer* и *Arduino IDE.*

Прво је извршено неопходно истраживање о компонентама, њиховом раду и улози у функционисању робота. Након тога је услиједио процес склапања конструкције робота и повезивање компоненти. На самом крају је направљена апликација за управљање роботом и пргамирање контролера и извршено тестирање робота.

Робот има два могућа типа управљања, гласовно и даљинско управљање. У случају гласовног управљања робот испоштује команде, а у случају наиласка на препреку обезбјеђује њено заобилажење. У тестирању робота, показало се да успјешност рада робота зависи доста од окружења. Уколико је окружење попуњено са доста компоненти, постоји могућност да робот детектује неки други објекат или да има сметње. Када је прилагођено окружење, робот је радио доста добро. Када детектује препреку, крене да је обилази. Робот има одређени степен самосталности, јер сам процијењује дужину препреке и одређује тренутке проласка препреке. Уколико утврди да препрека спада под, за његове перформансе, несавладиву, зауставља се, окреће и чека следећу команду. Други режим управљања је замишљен као ручни начин управљања. Робот може да се креће напријед, назад, лијево и десно. Обезбјеђена је детекција препреке и заустављање робота у случају детекције, како би се обезбиједила заштита робота.

Може се рећи да је оваква реализација робота доста добра, али да постоји много материјала за унапријеђење. Неке од могућих унапријеђења су

- Уградња *IR* сензора на задњи дио робота, како би обезбиједили детекцију препрека при кретању уназад
- Уградња додатног ултрасоничног сензора, како би обезбиједили бољи алгоритам за обилазак препрека и ријешили проблем несавладиве препреке
    - Идеја је да постоје два ултрасонична сензора гдје би један пратио дужину препреке, а други пратио пут испред да не дође до ударца
- Надоградња софтвера, рецимо да се обезбиједе још неки режими рада, као што су режим праћења линије, за шта би били неопходни додатни сензори и самостално кретање робота у простору уз могућност детекције и избјегавања препрека.
- Набављање прецизнијих сензора како би се побољшала тачност

На крају, треба рећи да су сви циљеви овог пројекта испуњени. Робот одлично ради, има способност гласовне и даљинске контроле, детекције препрека и њиховог обилажења, али као што је речено у претходном пасусу, постоји још много начина за унапријеђење.

Додатак:

Програмски код режима управљања гласом

```
  if (mode == "glas")
  {
    komanda = 1;
    por = p;
    if (por != "брже" && por != "sporije" && por != "" && pom == 0)
    {
      poruka = por;
      first = 1;
    } else
    {
      if (por == "брже")
      {
        Speed(brzina + 20);
      } else if (por == "sporije")
      {
        Speed(brzina - 20);
      }
    }
    prepreka = Test();
    if (prekid == 1)
    {
      poruka = "stop";
    } else if (prekid == -1)
    {
      int n = pom_poruka.length();
      poruka = pom_poruka.substring(0, n - 1);
      pom = 1;
    }

    if (poruka == "nazad" || poruka == "idi nazad" || poruka ==
"vrati se")
    {
      Backward();
    }

    if (prepreka == 1 && first == 1 && prekid == 0 && pom == 0)
    {
      obilazak();
```

```
    } else
    {
      pret_poruka = poruka;
      //      pom = 0;
      if (poruka == "kreni" || poruka == "naprijed" || poruka ==
"idi naprijed" || poruka == "napred" || poruka == "idi napred")
      {
        Forward();

      } else if (poruka == "lijevo" || poruka == "idi lijevo" ||
poruka == "levo" || poruka == "idi levo")
      {
        if (okret == 0)
        {
          Left();
        } else
        {
          Stop();
        }
      } else if (poruka == "desno" || poruka == "idi desno")
      {
        if (okret == 0)
        {
          Right();
        } else
        {
          Stop();
        }
      } else if (poruka == "stop" || poruka == "stani" || poruka ==
"zaustavi se"  || poruka == "")
      {
        Stop();
        poruka = "";
      }
      komanda = 0;
    }
  }
```

Функција за обилазак препреке:

```
void obilazak()
{
  Stop();
```

```
  delay(10);
  servo.write(180);
  delay(1000);
  float left = udaljenost(); //lijevo
  if (left == 0) left = 300;
  servo.write(0);
  delay(1000);
  float right = udaljenost(); //desno
  if (right == 0) right = 300;
  servo.write(90);
  delay(500);
  // Ako se s obje strane nalaze prepreke
  if (Serial.available() > 0)
  {
    pom_poruka = Serial.readString();
    int n = pom_poruka.length();
    poruka = pom_poruka.substring(0, n - 1);
    pom = 1;
    Serial.println("Poruka u obilazak petlji: " + poruka);
  } else
  {
    pom = 0;
    if (left < 50 && right < 50)
    {
      Backward();
      delay(300);
      Stop();
      poruka = "";
      pom = 1; // Da bi se robot zaustavio ako vidi da ne moze ni
lijevo ni desno, ceka sledecu komandu!
    } else if (left > right)
    {
      obilazak_s_lijeve();
    } else
    {
      obilazak_s_desne();
    }
  }
}
```

Функција за пролазак препреке:

```
void prolazak_prepreke(int stepeni)
```

```
{
  // Okretanje serva u odredjenu stranu
  prekid = 0;
  Stop();
  delay(500);
  servo.write(90 + stepeni);
  Speed(127);
  delay(500);
  unsigned long start = millis();
  while (udaljenost() < 40)
  {
    if (Serial.available() > 0)
    {
      novaKomanda();
      prekid = -1; // Neka ovo oznacava da je nova komanda stigla
      Serial.println("Usao u citanje u whilu u pp: " + poruka);
      break;
    } else
    {
      Forward();
      if (millis() - start >= 3000)
      {
        prekid = 1;
        break;
      }
    }
  }
  if (prekid == 1)
  {
    Stop();
    poruka = "";
    Speed(150);
    servo.write(90);
  } else if (prekid == -1)
  {
    servo.write(90);
    Speed(150);
    Serial.println("prekid je -1: " + poruka);
  } else
  {
    Forward();
    delay(250);
    servo.write(90);
```

```
    Speed(150);
    delay(100);
  }
}
```

Функција за обилазак препреке са лијеве стране:

```
void obilazak_s_lijeve()
{
  if (Serial.available() > 0)
  {
    novaKomanda();
    Serial.println("Usao u obilazak sa lijeve u nk: " + poruka);
  } else
  {
    Backward();
    delay(250);
    Stop();
    delay(1000);
    Speed(150); // Brzina na 200
    Left();
    unsigned long start_time_l = millis();
    prolazak_prepreke(-90);
    unsigned long pass_time_l = millis();
    if (prekid == 1)
    {
      Left();
    } else if (prekid == -1)
    {
      pom = 1;
    } else
    {
      if (Serial.available() > 0)
      {
        novaKomanda();
        prekid = -1;
      }
      Stop();
      delay(1000);
      Right();
      Forward();
      delay(500);
      Stop();
      prolazak_prepreke(-90);
```

```
      if (prekid == -1)
      {
        pom = 1;
      } else
      {
        Stop();
        delay(1000);
        if (Serial.available() > 0)
        {
          novaKomanda();
          prekid = -1;
        } else
        {
          Right();
          Speed(127);
          unsigned int start = millis();
          while (millis() - start < pass_time_l - start_time_l -
1350)
          {
            if (Serial.available() > 0)
            {
              novaKomanda();
              prekid = -1;
              break;
            }
            Forward();
          }
          //          delay();
          if (prekid != -1)
          {
            Stop();
            Speed(150);
            delay(1000);
            Left();
            Stop();
            delay(500);
          }
        }
      }
    }
  }
}
```

Литература

[1] John-David Warren, Josh Adams, and Harald Molle, "*Arduino Robotics*", 2011, New York

[2] MichaelMargolis, „*Make An Arduino-Controlled Robot*", 2013, Sebastopol

[3] Abhishek Thakur, Ashish Sharma, Biplov Ghosh, Nitin Rawani, Harpreet Kaur Channi, "*Designing and Modeling of Remote Control Car Using Arduino",* Department of Electrical Engineering University Institute of Engineering, Chandigarh University, Gharuan, SAS Nagar, Punjab, India, IJSART - Volume 3 Issue 11 – NOVEMBER 2017

[4] Сајт *ElProCus*
https://www.elprocus.com/atmega328-arduino-uno-board-working-and-its-applications/

[5] Сајт *Arduino*
https://www.arduino.cc
https://playground.arduino.cc/Code/NewPing
https://www.arduino.cc/en/Reference/Servo

[6] Сајт *Adafruit*
https://www.adafruit.com

[7] Сајт *Wikipedia*
https://en.wikipedia.org/wiki/Arduino