Arduino programabilni mini robot

Radić, Valentino

SVEUČILIŠTE U SPLITU

POMORSKI FAKULTET U SPLITU

VALENTINO RADIĆ

ARDUINO PROGRAMABILNI MINI ROBOT

ZAVRŠNI RAD

SPLIT, 2017.

SVEUČILIŠTE U SPLITU

POMORSKI FAKULTET U SPLITU

STUDIJ: POMORSKE ELEKTROTEHNIČKE I INFORMATIČKE TEHNOLOGIJE

ARDUINO PROGRAMABILNI MINI ROBOT

ZAVRŠNI RAD

MENTOR:

dr. sc. Joško Šoda

STUDENT:

Valentino Radić

(MB:0171265454)

SPLIT, 2017.

SAŽETAK

Završni rad predstavlja izradu programibilnog robota, temeljenog na razvojnom sučelju Arduino. Predstavit će se i razvojno sučelje Arduino, način rada, mogućnosti te upotrebu u modernoj tehnologiji. Robot ima veliki broj mogućnosti koje su ograničene arhitekturom razvojnog sustava Arduino, te maštom korisnika. Na primjeru samohodnog gusjeničara koji ima mogućnost zaobilaženja prepreka te spajanja bluetooth-om na Android mobilni uređaj radi ručnog upravljanja koristi se Arduino razvojno sučelje.

Ključne riječi: *Arduino,Keyestudio,programiranje,bluetooth,tenk,robot,Andriod*

ABSTRACT

This final thesis will follow the making of an Arduino programmable robot, Arduino itself, the way it works, its capabilities and its use in modern tehnology in detail. The robot has countless possiblilities limited to Arduino itself as its base and user knowledge, but in this example it will stop on a self-propelled tank which has the alibity to overcome obstacles and connect to on Android mobile devices with a goal to control it manually by bluetooth.

Keywords: *Arduino, Keyestudio, programming,bluetooth,tank,robot,Android*

SADRŽAJ

1. UVOD

U današnje „moderno“ vrijeme nekada nepojmljiva tehnologija je postala opće znanje, pa tako i osnovna znanja o robotici i programiranju. Neka od tih su obrađena u ovom radu kroz poglavlja i projekte uz praktični primjer.

Robotika predstavlja jednu od grana inžinjerske znanosti i tehnologija robota, njihov dizajn, proizvodnju i primjenu, te je srodna s elektronikom, mehanikom i područjem razvoja softvera.

Programiranje obuhvaća generiranje smislenog koda na računalu ili nekom drugom uređaju, te učitavanje koda u uređaj koje je u mogućnosti da interpretira kod. Može se nazvati umijećem stvaranja programa tj. sofvera. Upute se pišu na različitim programskim jezicima, u različitim programskim sučeljima, upotrebom određenih sintaksi te koristeći pravila koja vrijede za svaki programski jezik koji se prevodi u strojni jezik (eng. Assembler) svojstven za neko računalo ovisno o njegovoj arhitekturi. Prevođenje se vrši izvršavanjem programa prevodioca (eng. compilera) ili se izravno prevode preko tzv. p_koda.

1.1. ARDUINO

Arduno je tzv. open source hardverska i softverska kompanija, projekt, korisnička zajednica pod jednim imenom koja dizajnira i proizvodi projektne platforme bazirane na mikrokontrolerima koje omogućuju, kako krajnjem korisniku tako i dizajnerima te konstruktorima, stvaranje uređaja koji imaju mogućnost osjeta i kontrole objekata u fizičkom svijetu.

Slika 1. Arduino logo

Projektirala ga je talijanska tvrtka SmartProjects 2005. godine. Rabeći 8-bitne Atmel AVR mikrokontolere je nastala mala i jeftina platforma za povezivanje računala s fizičkim svijetom.

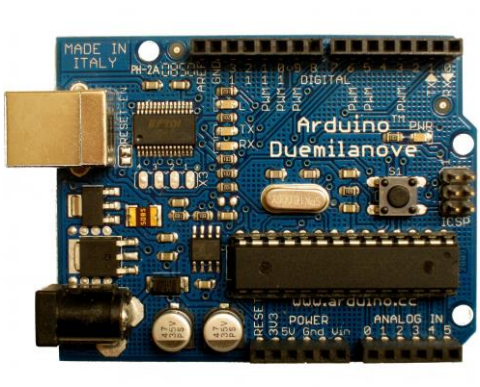
Iako je sustav otvoren, logo (slika 1.) i sklopovlje Arduina je patentirano te se ne može proizvoditi nigdje osim u matičnoj kompaniji. Arduina je moguće nabaviti gotovog, spremnog za upotrebu ili u dijelovima kao tzv. DIY (eng. **D**o **I**t **Y**ourself) projekt.

Arduino tiskane pločice koriste raznorazne mikroprocesore i kontrolere koje su opremljene mnogobrojnim digitalnim te analognim ulazima i izlazima.

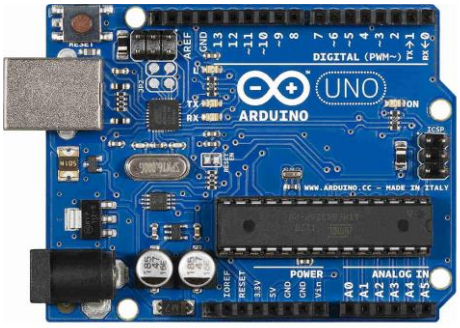
1.1.1. Izvedbe pločica

Službeno postoji 17 verzija Arduino tiskanih pločica: RS232, Diecimila (slika 2.), Duemilanove (slika 3.), Uno (slika 4.; korišten u ovom projektu), NG (slika 5.), Extreme, Bluetooth, Leonardo, Pro, Mega, Nano, LilyPad, Robot, Esplora, Ethernet, Yun, Due.

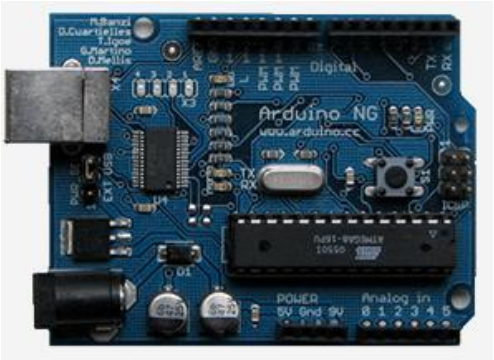
Slika 2. Arduino Diecimila



Slika 3. Arduino Duemilanove

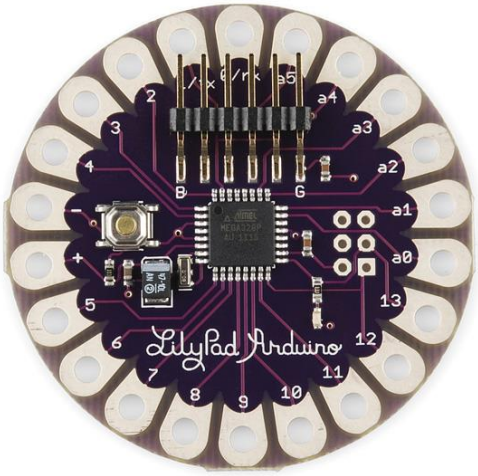


Slika 4. Arduino Uno



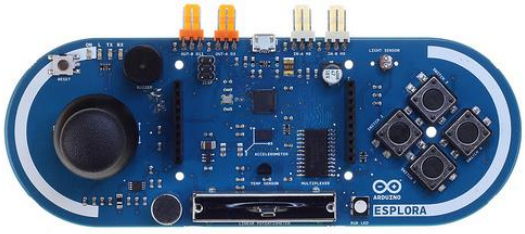
Slika 5. Arduino NG

Iako naizgled slični svaki od modela Arduina odlikuje se svojim karakteristikama, od različitih mikrokontrolera, memorije, brzine do različitih cijena i načina spajanja. U zadnje vrijeme Arduino je posegnuo za modernijim i radikalnijim dizajnom (slike 6,7,8,9) u potrazi za novim korisnicima i proširenjem radnog opsega sklopne pločice kako bi ispratio nove zahtjeve.

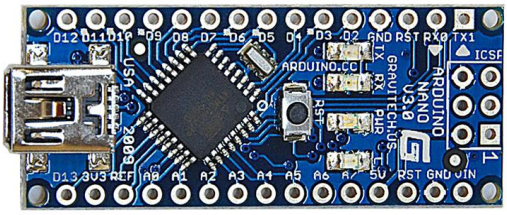


Slika 6. Arduino LilyPad

Slika 7. Arduino Robot



Slika 7. Arduino Esplora



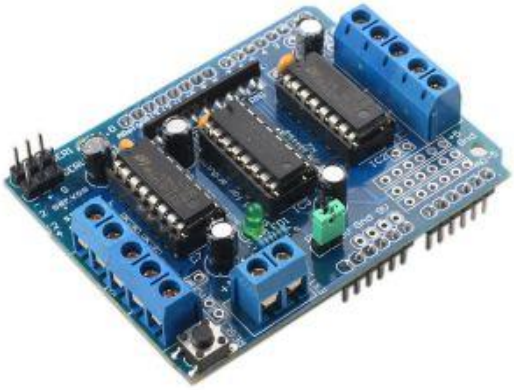
Slika 9. Arduino Nano

1.1.2. Štitovi

Postojeći ulazi i izlazi na Arduinu omogućuju spajanje dodatnih ekspanzijskih sklopnih ploča, takozvanih štitova (eng. shields) koji se priključuju na postojeće kontakte na Arduino sklopnoj pločici. Uz štitove moguće je spojiti i druge elemente i uređaje na Arduino.

Štitovi mogu omogućiti kontrole motora preko Arduina (slika 11.), spajanje GPS-a (slika 13.), Ethernet (slika 10.), spajanje LCD-a (slika 12.), dimnog detektora (slika 14.), dodirne površine, testne pločice (za prototipe), kamere (slika 15.), itd...

Slika 10. Ethernet štit



Slika 11. Motor štit

Slika 12. LCD štit

Slika 13. GSM/GPRS štit

Slika 14. Štit s detektorom dima

Slika 15. Štit s kamerom

1.2. KEYESTUDIO

Keyestudio je proizvođač kreativne robotike i sklopovlja s otvorenim sučeljem, koji je specijaliziran za Arduino kontrolere, štitove, senzore, module, Raspberry PI, zaslone itd.

Slika 16. Keyestudio logo

Osnovani su 2011., sa središtem u gradu Shenzhenu, država Narodna Republika Kina, pod logom bubamare s robotičkim krilima (slika 16.) koja daje naslutiti njihovu svrhu. Keyes Robot ima profesionalni tim posvećen dizajniranju i proizvodnji open source hardvera, posebno za robotske platforme, senzore i štitove kompaktibilne s Arduino razvojnim sučeljem. Kroz zadnjih 6 godina lojalni klijenti i dobar odziv na njihovu paletu proizvoda omogućio je naglo širenje. Danas Keyestudio nudi kompletan nastavni plan i program, sveobuhvatne setove za učenje i video tutorijale dizajnirane da pomažu praktičarima od samog početka do kreiranja projekta.

Svatko može imati koristi od robotike u ostvarivanju vlastitih ciljeva i realiziranju kreativnih ideja na učinkovit način.

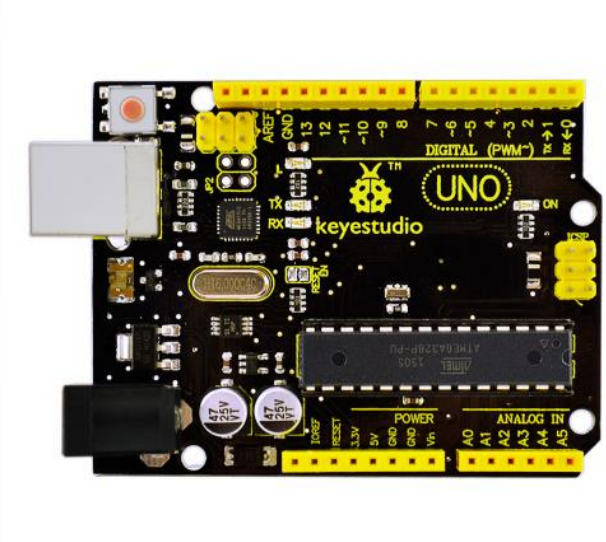
U ovom projektu korišteni su Keyestudio-vi Arduino štitovi i moduli, koji su opisani u daljnjim poglavljima, s ciljem pojednostavljenja sklopovske izvedbe robota.

2. IZRADA

2.1. HARDVER

Kroz ovo poglavlje opisani su sklopovski dijelovi i njihovi parametri potrebni za izradu mini robota.

2.1.1. Keyestudio UNO R3 kontroler



Slika 17. Keyestudio UNO R3 kontroler

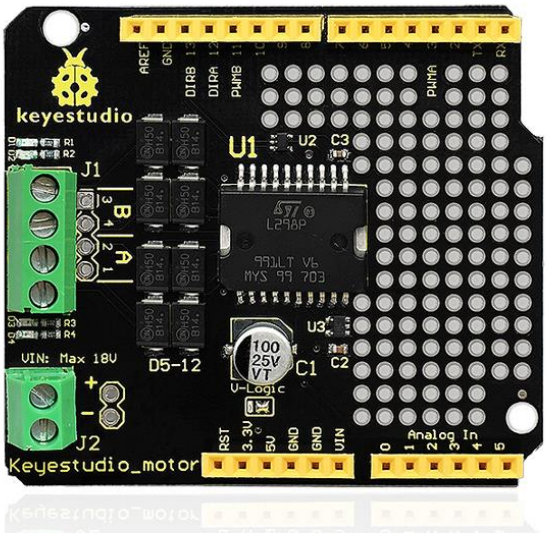
Keyestudio Uno R3 (slika 17.) je mikrokontrolerska pločica bazirana na ATmega328p-pu. Ima 14 digitalnih ulazno/izlaznih kontakta (od kojih 6 može biti korišteno kao PWM izlazi), 6 analognih ulaza, 16 MHz keramički rezonator, USB konekciju, konektor za napajanje, ICSP te reset tipku. Sadrži sve potrebno za potporu mikrokontroleru, jednostavno se poveže preko USB sučelja ili na napajanje za početak.

Uno R3 se razlikuje od svojih prethodnika jer ne koristi FTDI USB-to-serial upravljački program već Atmega 16U2 program kao USB-u-serijski pretvornik. Ostale prednosti ove mikrokontrolerske pločice se mogu dobiti iz njenih specifikacija (tablica 1.).

Tablica 1. Specifikacije UNO R3 mikrokontrolerske pločice

Mikrokontrolerska jezgra	ATmega328P-pu
Radni napon	+5V
Vanjski ulazni napon	+7V ~ +12V
Vanjski ulazni napon (ekstrem)	+6V ≤ Vin ≤ +20V
Digitalni signal U/I	14
Analogni signal ulazi	6
DC U/I struja	20mA
Flash memorija	32KB (ATmega328) od čega 0,5KB koristi bootloader
SRAM	2KB
EEPROM	1KB
Frekvencija takta	16MHz
Dužina	68,6mm
Širina	53,4mm
Visina	25g

2.1.2. Keyestudio L298P štit za motore



Slika 18. Keyestudio L298P štit

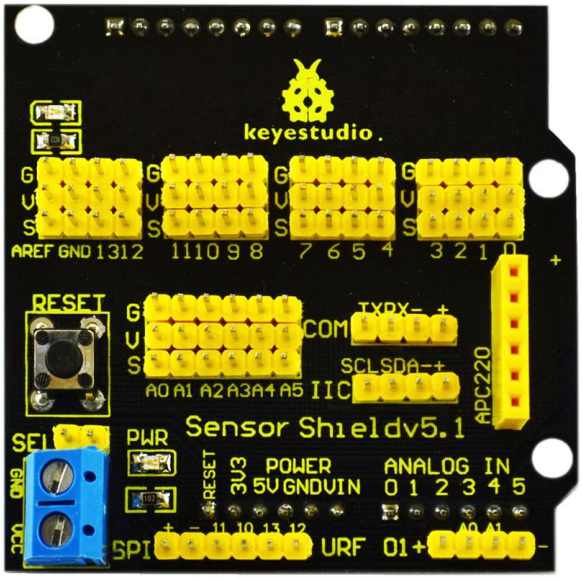
L298P štit za DC motore (slika 18.) usvaja L298P upravljački čip koji je izrađen isključivo za motore velike snage. Može direktno pogoniti 2 DC motora sa strujama

maksimalne jakosti 2A. Izlaz za motore je opremljen s 8 brzih Schottky dioda kao zaštita. Njegov višeslojni dizajn omogućuje izravno priključivanje na Arduino te može biti napajan s istog ili preko terminala na samom upravljaču.

Tablica 2. Specifikacije L298P štita

Napajanje logičkog dijela	5V
Napajanje upravljačkog dijela	VIN: +6,5~12V, PWRIN: +4,8~24V
Radna struja logičkog dijela Iss	≤36mA
Radna struja upravljačkog dijela Io	≤2A
Maksimalna disipacija snage	25W (T=75ºC)
Leveli kontrolnih signala	Visoki: +2,3V≤Vin≤5V
	Niski: -0,3V≤Vin≤1,5V
Radna temperatura	-25 ºC ~ +130 ºC

2.1.3. Keyestudio V5 senzorski štit

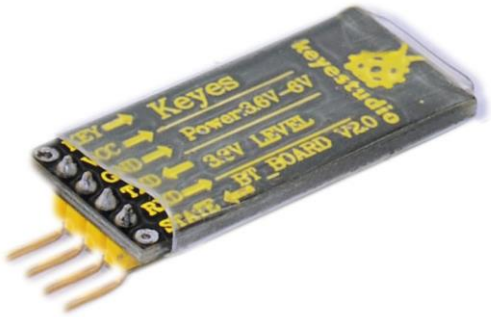


Slika 19. Keyestudio V5 senzorski štit

Senzorski štitovi omogućuju jednostavno spajanje ulaznih i izlaznih uređaja, ne samo senzora, na Arduino. Njihova namjena je, dakle, olakšati spajanje kablova i uređaja na odgovarajuće Arduino kontakte. Iz istog razloga dizajn predstavlja najveću prednost ove vrste štita.

Keyestudio V5 senzorski štit (slika 19) ima 14 digitalnih U/I kontakta u 4 bloka, svaki od blokova ima tri kontakta koja su spojena na uzemljenje G (GND),napajanje V (Vcc +5V) te signal S. Posjeduje 5 analognih izlaza. Moguć je odabir serijske komunikacije (COM) ili „I squared C“ (I2C) komunikacije koje omogućuju spajanje uređaja s kompleksnim komunikacijskim protokolima poput GPS-a, Etherneta ili u ovom slučaju Bluetooh modula.

2.1.4. Keyestudio Bluetooth modul (HC-06)



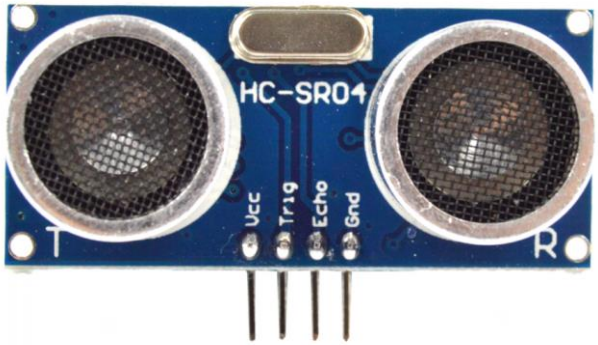
Slika 20. Keyestudio HC-06 bluetooth modul

Ovaj bluetooh modul omogućuje jednostavnu bežičnu serijsku kominikaciju i prijenos podataka. Radi na 2,4 GHz frekvenciji koja je najpopularnija među svim industrijskim, znanstvenim i medicinskim svrhama. Obuhvaćen je Bluetooth 2,1+EDR standardom što mu omogućuje rad s uređajima obuhvaćenim istim standardom (svi mobilni uređaji opremljeni bluetoothom 2,1 ili novijim).

Tablica 3. Specifikacije HC-06 bluetooth modula

Bluetooth protokol	Bluetooth 2.1+EDR standard
USB protokol	USB v1,1/2,0
Radna frekvencija	2,4GHz ISM frekvencijski pojas
Snaga odašiljanja	≤4dBm
Osjetljivost	≤ -84dBm pri 0.1% pogreške
Brzina prijenosa	Asinkrono: 2,1Mbps(Max)/160kbps
	Sinkrono: 1Mbps/1Mbps
Sigurnosne mogućnosti	autentifikacija i enkripcija
Podržana konfiguracija	bluetooth serijski port (glavni i sporedni)
Napajanje	+3,3V DC, 50mA
Radna temperatura	-20°C ~ 55°C

2.1.5. HC-SR04 ultrazvučni senzor



Slika 21. HC-SR04 ultrazvučni senzor

HC-SR04 (slika 21.) ultrazvučni modul omogučuje bezkontaktno mjerenje udaljenosti u rasponu od 2 do 400cm s preciznošću do 3mm. Ultrazvučni modul se sastoji od odašiljača, primatelja i kontrolne ploče.

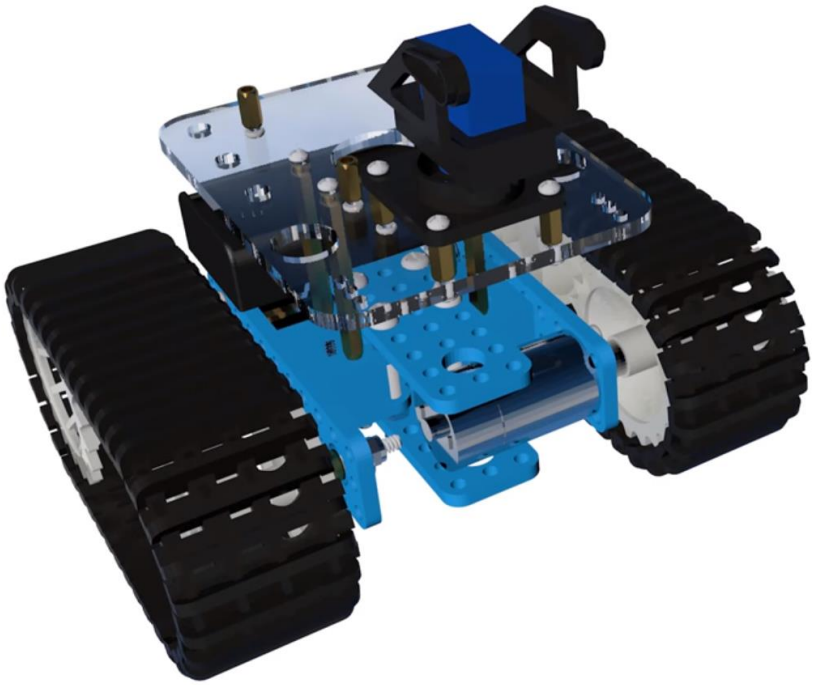
Radi na jednostavnom principu odašiljanja 8 pulsova na 40 kHz kada zaprimi signal od osclilatora i detektira povratni odbijeni puls. Potom nakon što se povratni puls vrati u prijamnik, testna udaljenost se dobiva dijeljenjem proteklog vremena između odašiljanja i zaprimanja pulsa s brzinom zvuka.

Tablica 4. Specifikacije HC-SR04 ultrazvučnog senzora

Radni napon	+5V
Radna struja	15mA
Radna frekvencija	40Hz
Maksimalni domet	4m
Minimalni domet	2cm
Kut mjerenja	15º
Okidni ulazni signal	10µs TTL puls
Dimezije	45*20*15mm

2.1.6. Ostatak hardvera

Ostatak hardvera obuhvaća metalnu karoseriju (slika 22.) sastavljenu na nekoliko razina na koju su ugrađena 2 pogonska motora, parametara iz tablice 5, te pogonjene gusjenice. Također korišten je dodatni servo motor s nosačem na koji je postavljen ultrazvučni senzor s ciljem lakše pokretljivosti tj. detekcije nailazećih prepreka. Naravno za sve to je potrebno napajanje, u ovom slučaju dvije baterije tipa 18650 Efest lithium-manganese, specifikacija iz tablice 6.



Slika 22. Izgled karoserije s ostalim hardverom robota

Tablica 5. Parametri pogonskih motora

Nazivni napon	+6V DC
Struja bez opterećenja	70mA
Brzina vrtnje bez opterećenja	150rpm
Prikladan napon	3V-12V DC (75rpm-300rpm)
Težina	42g
Dužina motora/osovine	42mm/7mm (D profil)
Promjer kučišta/osovine	20mm/4mm

Tablica 6. Specifikacije Efest IMR18650 3000 baterija

Nazivni kapacitet	3000mAh
Nazivni napon	3,7V
Napon na kraju pražnjenja	2,5V
Maximalna struja punjenja	4A
Struja punjenja	2A
Struja pražnjenja	20A
Pulsna struja pražnjenja	35A
Temperatura pri punjenju	0^oC-45^oC
Dimenzije	65,2mm*18,5mm
Certifikati	CE, RoHS, MSDS, UL

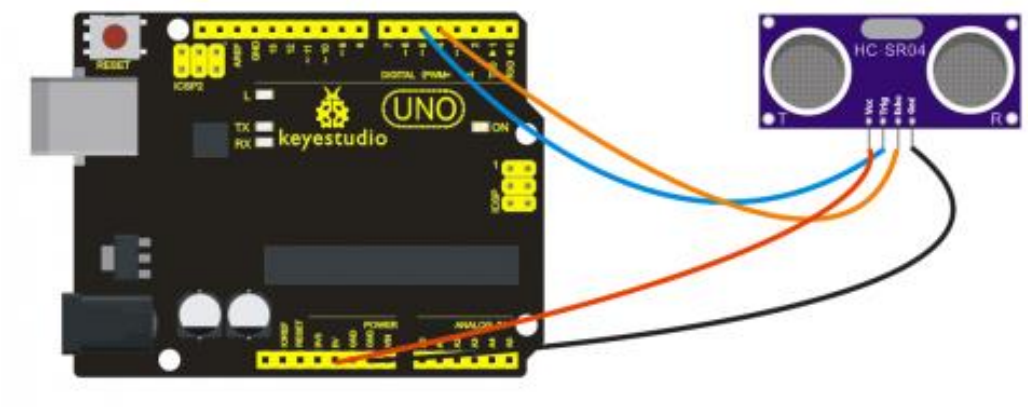
2.2. SOFTVER

Kako bi sve od sklopovlja navedenog kroz prošla poglavlja funkcioniralo kao jedno potreban je upravljački programski kod. Za rad s Arduinom se koristi kombinacija C i C++ programskog jezika s pravilima jedinstvenim za Adruino koji se piše u odgovarajućem Arduino okruženju tj. Arduino IDE, u tzv. offline načinu rada, ili ako postoji stabilna internet veza rabi se tzv. online IDE (Arduino Web Editor) s kojim dolaze i sve prednosti tzv. online rada.

Gotovi programski kod se prevodi u strojni i prenosi na Arduino. Da bi prijenos bio uspješan prethodno treba biti instaliran odgovarajući USB driver na računalo kako bi se ostvarila veza s Arduino sustavom. Ukoliko se koristi Arduino za pogon većih potrošača potrebno je spojiti i dodatno napajanje.

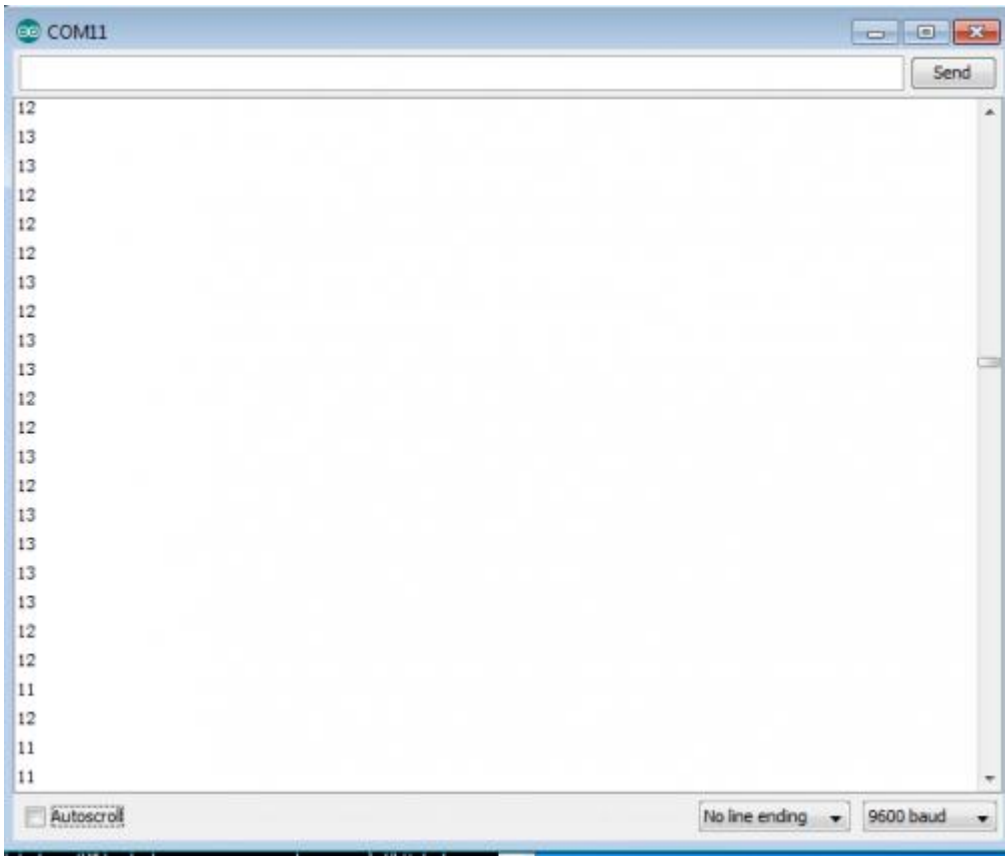
2.2.1. Projekt 1: Testiranje ultrazvučnog senzora

HC-SR04 je ultrazvučni senzor je vrlo dostupni senzor udaljenosti koji je uglavnom korišten za zaobilaženje prepreka u raznoraznim robotičkim projektima. Ukratko, daje Arduinu sposobnost da vidi i može spriječiti sudaranje robota ili pad sa stola. Također, može se koristiti za mjerenje razine tekućine, aplikacijama u kupolama pa čak i kao parkirni senzor. Ovaj jednostavan projekt će koristiti HC-SR04 senzor uz Arduino (slika 23.) i programski kod (prilog 1) kako bi pružio uredan interaktivan prikaz na računalu.



Slika 23. Dijagram spajanja senzora s Arduinom

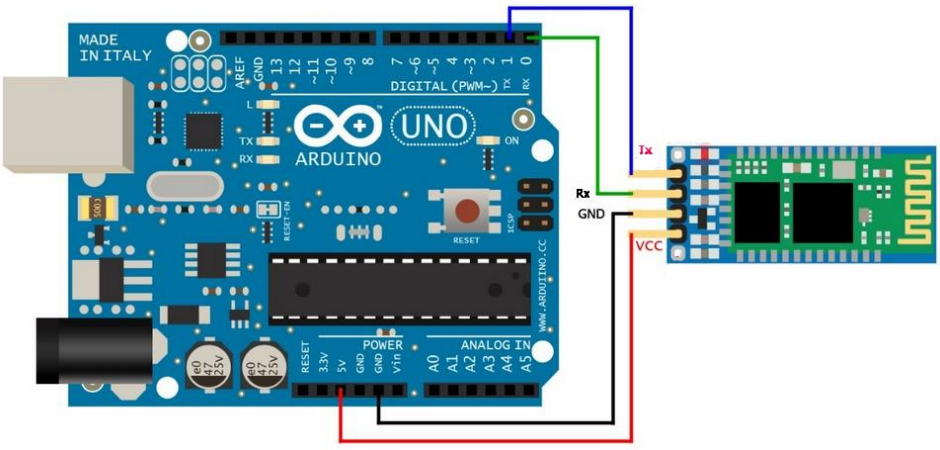
Nakon spajanja i prijenosa, kada ultrazvučni senzor osjeti zapreku unutar osjetnog područja, mjeri udaljenost između samog senzora i prepreke i vrijednost udaljenosti u cm prikazuje na ekranu (slika 24.)



Slika 24. Rezultat projekta 1

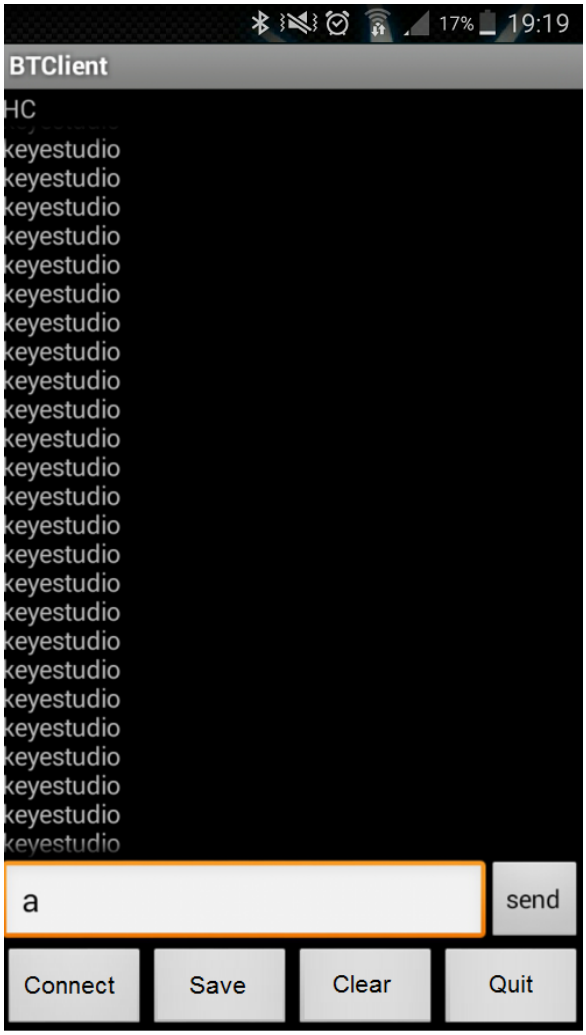
2.2.2. Projekt 2: Testiranje bluetooth modula

U ovom projektu uz bluetooth modul, Arduino (slika 25.) i odgovarajući programski kod (prilog 2) uspostavljena je veza između Arduina i mobilnog uređaja.



Slika 25. Dijagram spajanja bluetooth modula s Arduinom

Nakon uključivanja, kada je LED dioda na bluetooth modulu aktivna (u titrajućem modu), potrebno je upariti bluetooth modul s mobilnim uređajem. Nakon toga u preprogramiranoj aplikaciji za komunikaciju tj. odašiljanje naredbi Android uređaja bluetooth-om, u ovom slučaju „BTClient“, šalje naredbu „a“ na koju će Arduino odgovoriti s „keyestudio“ kao što je prikazano na slici 26.



Slika 26. Rezultat projekta 2

2.2.3. Projekt 3: Samohodni tenk

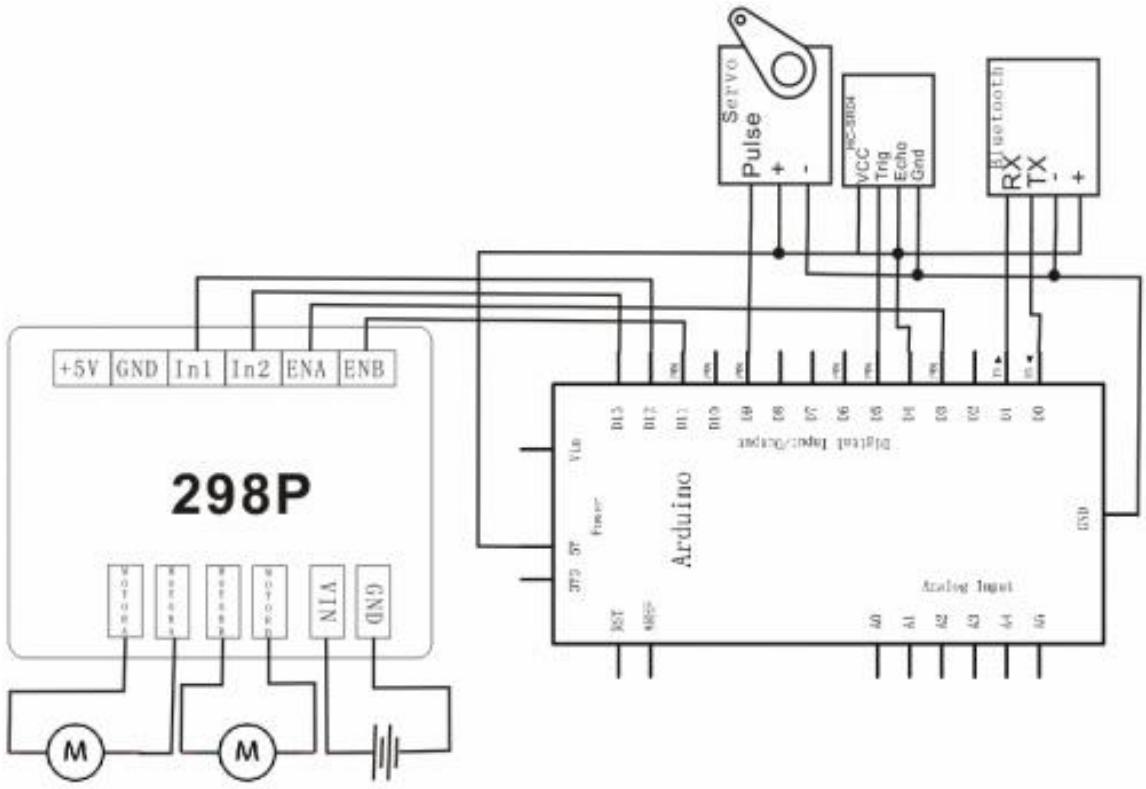
U ovom projektu projektirati će se samohodni tenk baziran na Arduino razvojnom sučelju. Upravljačka ploča je UNO R3, dok su ultrazvučni senzor i servo motor korišteni za detekciju prepreka ispred tenka uz dva pogonska motora za pokretanje cijelog sustava.

Povratni signal s senzora se šalje u UNO koji analizira signal kako bi se, na osnovu koda programa, napravila procjena i izvršilo upravljanje sustava preko motora, te prilagodio smjer kretanja tenka. Kao takav tenk samostalno može zaobilaziti prepreke.

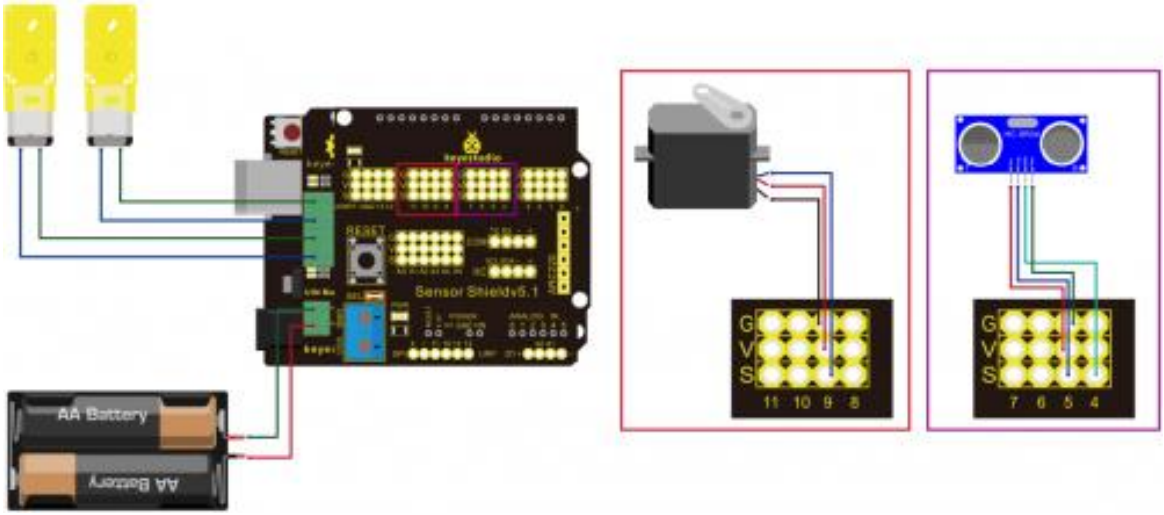
Detaljno; kontroler upućuje signal visoke razine ultrazvučnom senzoru i tada se uključuje vremensko brojilo (eng. Timer). Ultrazvučni senzor šalje signal prema okolini ispred sebe i povratni signal niske razine prosljeđuje se na kontroler. Kada kontroler primi signal na svojem ulazu, vremensko brojilo se zaustavlja. To vrijeme uz brzinu odašiljanja signala (brzina zvuka) daje mogućnost izračuna udaljenosti. Ovisno o očitanoj udaljenost tenk se pomiče sukladno.

Ako je udaljenost od prepreke manja od 10cm, tenk se pomiče unatrag; ako je udaljenost veća ili jednaka 25cm, servo motor okreće ultrazvučni senzor lijevo i desno kako bi utvrdio udaljenosti na obe strane. Ukoliko su udaljenosti s obe strane manje od 10cm tenk se pomiče unatrag. Ako su udaljenosti na lijevo i desno veće ili jednake 10cm tenk se okreće i pomiče u smjeru gdje ima više prostora.

Da bi se ostvarilo isto, navedeno sklopovlje mora biti spojeno po dijagramu spajanja na slikama 27. i 28. te programirano po prilogu 3.



Slika 27. Dijagram spajanja Arduina s štitom za pogonske te servo motore i senzorskim štitom te samim motorima i senzorom



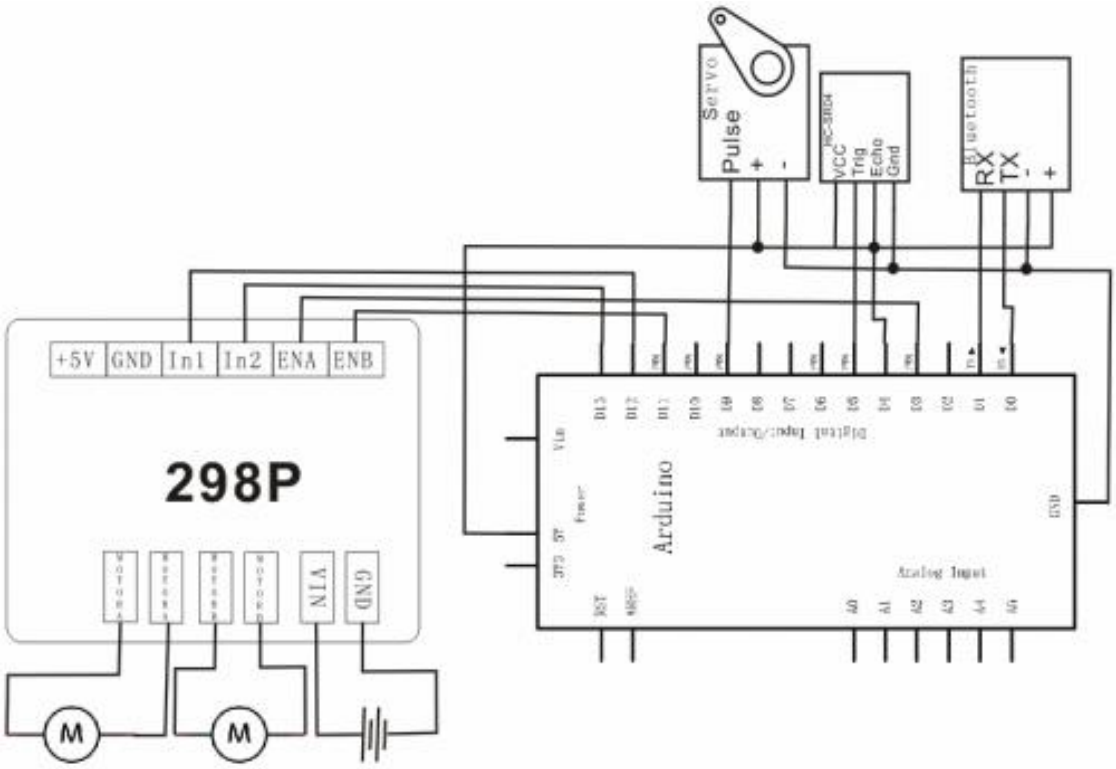
Slika 28. Dijagram spajanja Arduina s štitom za pogonske te servo motore i senzorskim štitom te samim motorima i senzorom (prikaz realnog izgleda sklopovlja)

2.2.4. Projekt 4: Bluetooth-om kontrolirani tenk

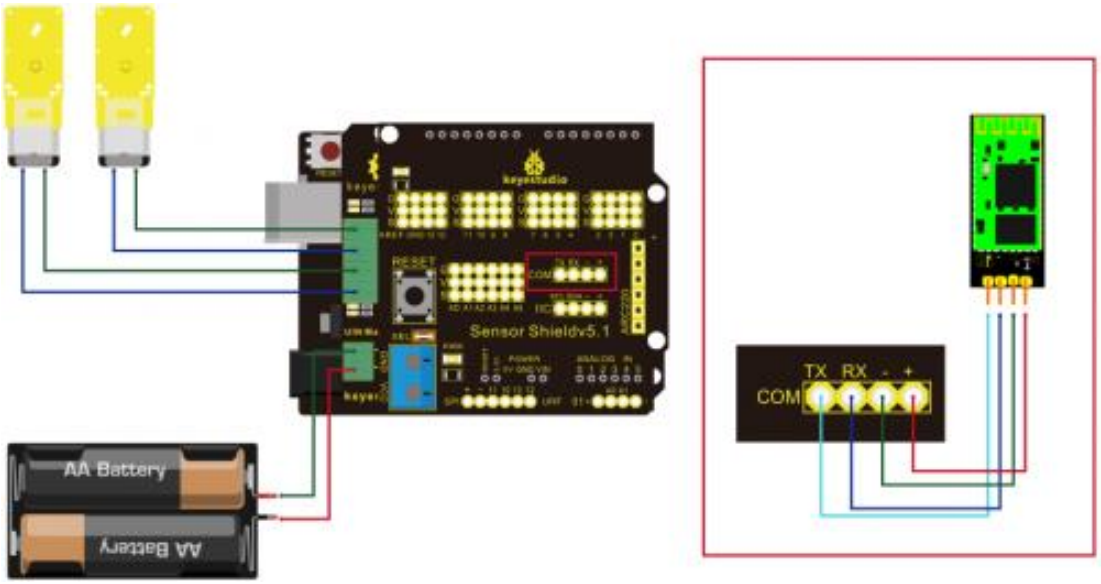
Ovaj projekt je baziran na osnovnim principima bluetooth komunikacije. Bluetooth modul je korišten kako bi zaprimio bluetooth signal s mobilnog uređaja i proslijedio ga UNO-u. UNO analizira signal kako bi procijenio i kontrolirao pokrete motora i prilagodio smjer kretanja tenka. Kao takav, tenk se može upravljati mobilnim uređajem tj. pametnim telefonom (eng. smartphone).

Detaljnije; bluetooth modul je spojen s Arduinom (slika 29. i 30.) te nakon što je uparen komunicira s mobilnim uređajem preko aplikacije „Keyes BT Car“ koja omogućuje odašiljanje naredbi „U“, „D“, „L“, „R“, „S“ na bluetooth modul. Bluetooth modul prosljeđuje naredbe UNO upravljačkoj ploči koja obrađuje podatke i pokreće motore sukladno zaprimljenim naredbama kako je programirano po prilogu 4.

Kada UNO zaprimi naredbu „U“ tenk se pokreće naprijed,kada zaprimi naredbu „D“ ide unatrag, kada zaprimi „L“ ide lijevo, kada zaprimi „R“ ide desno, a kada zaprimi naredbu „S“ staje tj. zaustavlja se.



Slika 29. Dijagram spajanja Arduina s štitom za pogonske motore i senzorskim štitom te samim motorima i bluetooth modulom



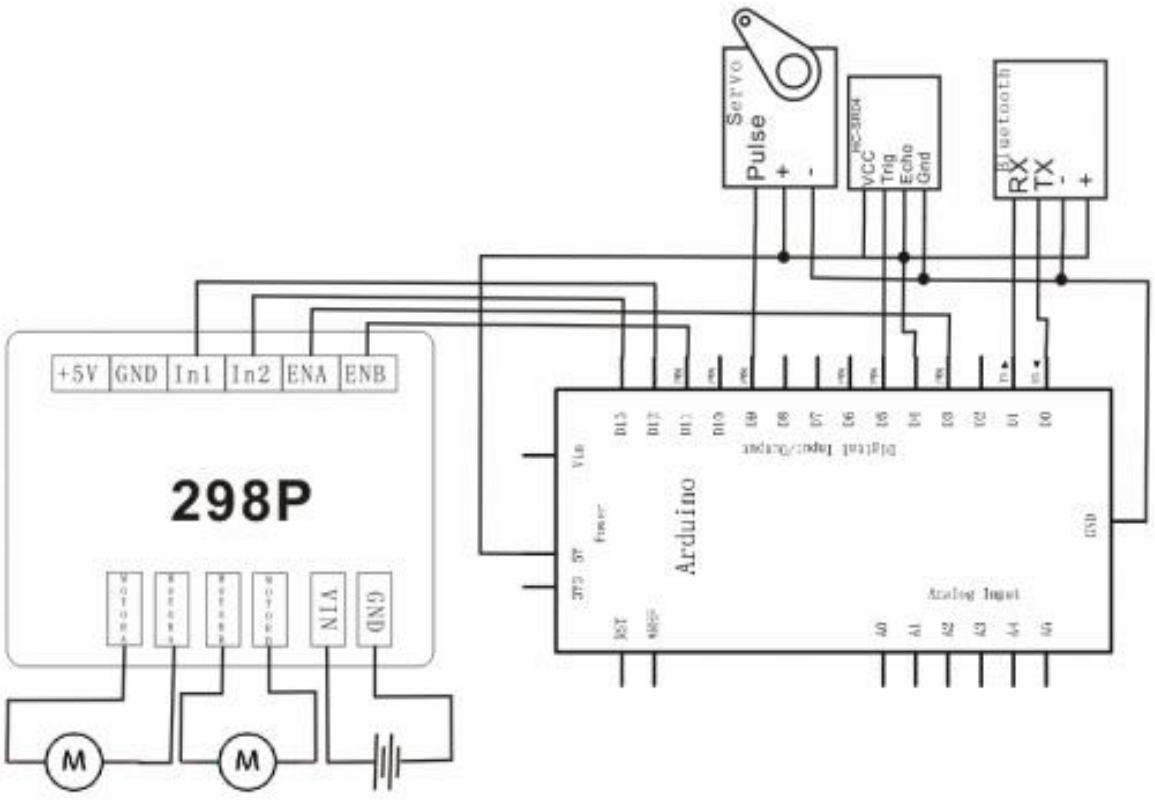
Slika 30. Dijagram spajanja Arduina s štitom za pogonske motore i senzorskim štitom te samim motorima i bluetooth modulom (prikaz realnog izgleda sklopovlja)

2.2.5. Projekt 5: Robot za ultrazvučno mjerenje

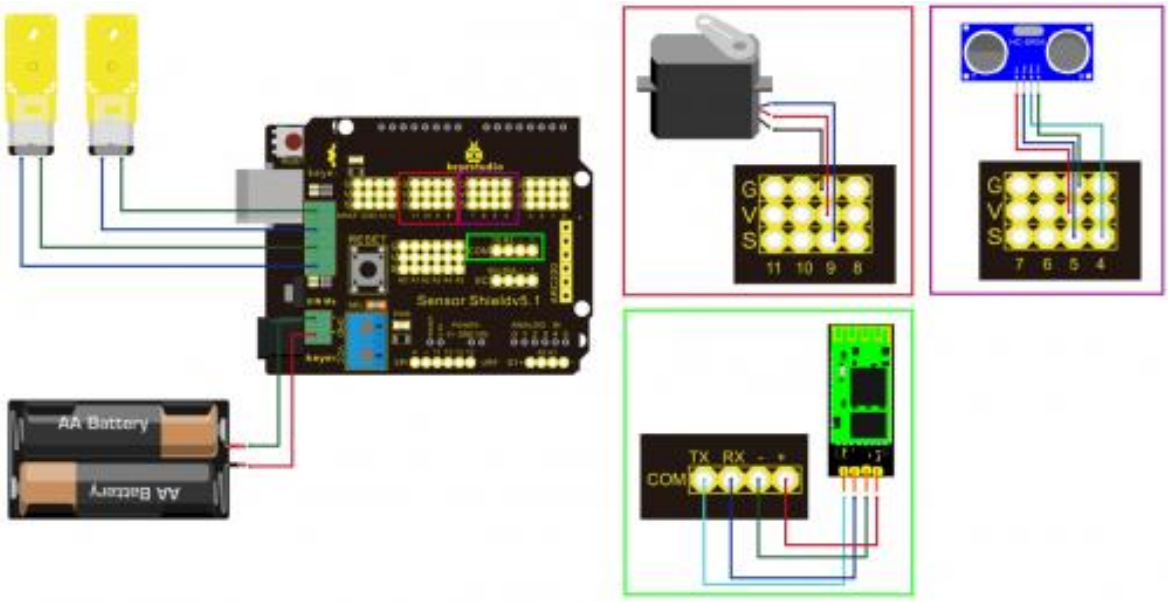
U projektu 3, korišten je ultrazvučni senzor kako bi se ostvarila mogućnost tenka da zaobilazi prepreke. U projektu 4, korišten je bluetooth modul kako bi tenk mogao biti kontroliran preko mobilnog uređaja.

Ovaj projekt je baziran na projektima 3 i 4. U ovom slučaju primjenjujemo ultrazvučni senzor za mjerenje udaljenosti koja će preko bluetooth modula biti poslana na terminal tj. mobilni uređaj tako da se u svakom trenutku vidi koja je udaljenost između tenka i prepreke. Mobilni uređaj, naravno, mora biti uparen s bluetooth modulom te na njemu mora biti instalirana prikladna aplikacija, u ovom slučaju „BTClient“.

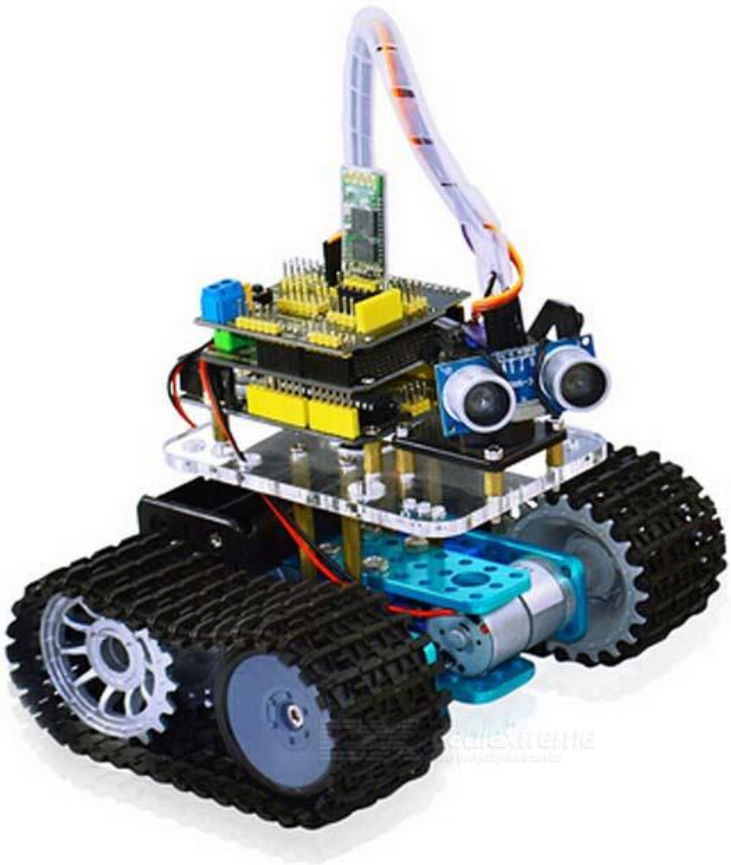
Robot, kao na slici 33, koji se dobije spajanjem sklopovlja po dijagramu spajanja na slikama 31. i 32., čiji je UNO kao kontrolna ploča programiran po prilogu 5, se može koristiti za razna istraživanja terena u nedostupnim prostorima.



Slika 31. Dijagram spajanja Arduina s štitom za pogonske te servo motore i senzorskim štitom te samim motorima, senzorom i bluetooth modulom



Slika 32. Dijagram spajanja Arduina s štitom za pogonske te servo motore i senzorskim štitom te samim motorima, senzorom i bluetooth modulom (prikaz realnog izgleda sklopovlja)



Slika 33. Izgled robota

2.3. PROBLEMI , RJEŠENJA, NADOGRADNJE

Unošenje inovacija i nadogradnji u sustav uvijek sa sobom nosi i neke nedostatke. Počevši od kompliciranosti hardverske izvedbe, cijene, pa do softverske izvedbe, ali sve ovisi o zahtjevima korisnika.

Ovaj projekt može se razvijati u dva smjera, može se rabiti za vojnu primjenu, ukoliko se na njega ugradi kakva robotska ruka, pila ili kupola s topom, a može se rabiti i za primjenu u znanosti (preko algoritama vođenja i inteligentnih algoritama), ukoliko se na njega postavi još kojekakvi štit s npr. kamerom, GPS-om, itd. Uglavnom valja istaći da su mogućnosti skoro neograničene, kada bude potrebno može se promijeniti upravljačka ploča, staviti baterije većeg kapaciteta, jači pogonski motori... kako bi zadovoljili nove zahtjeve korisnika.

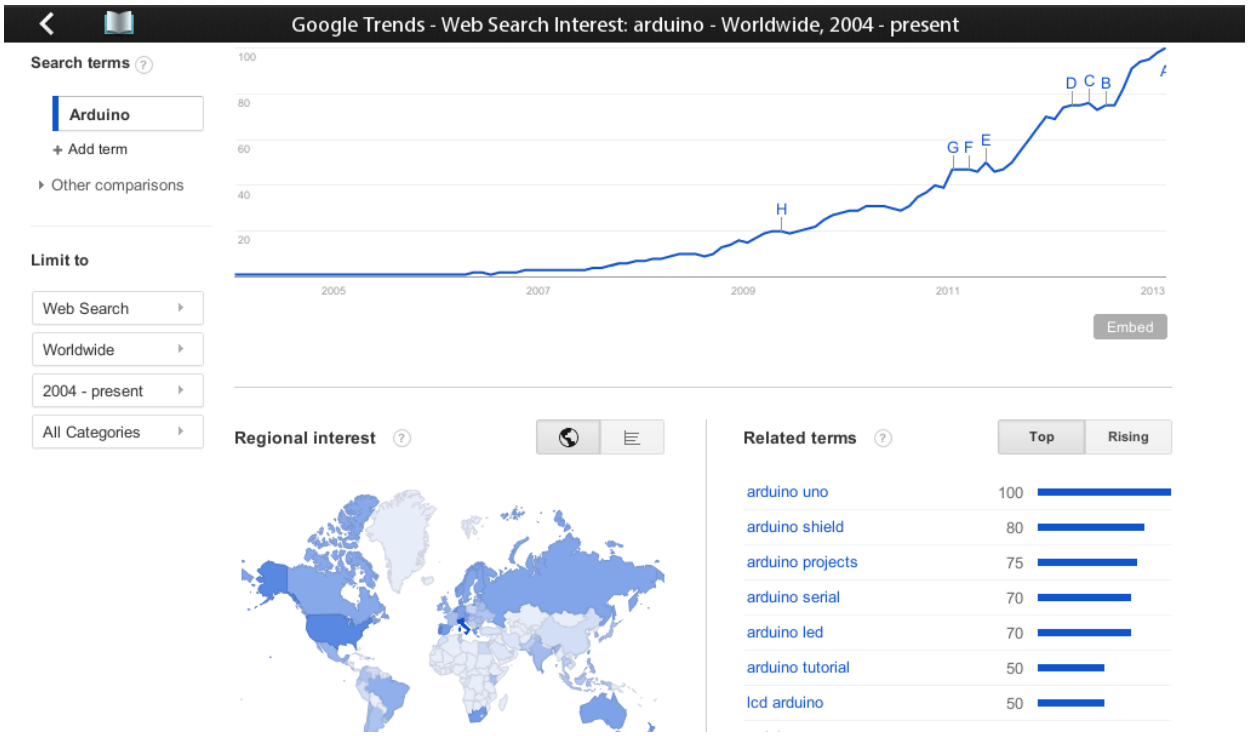
3. RAZVOJ ARDUINA

Kakva je budućnost Arduina?

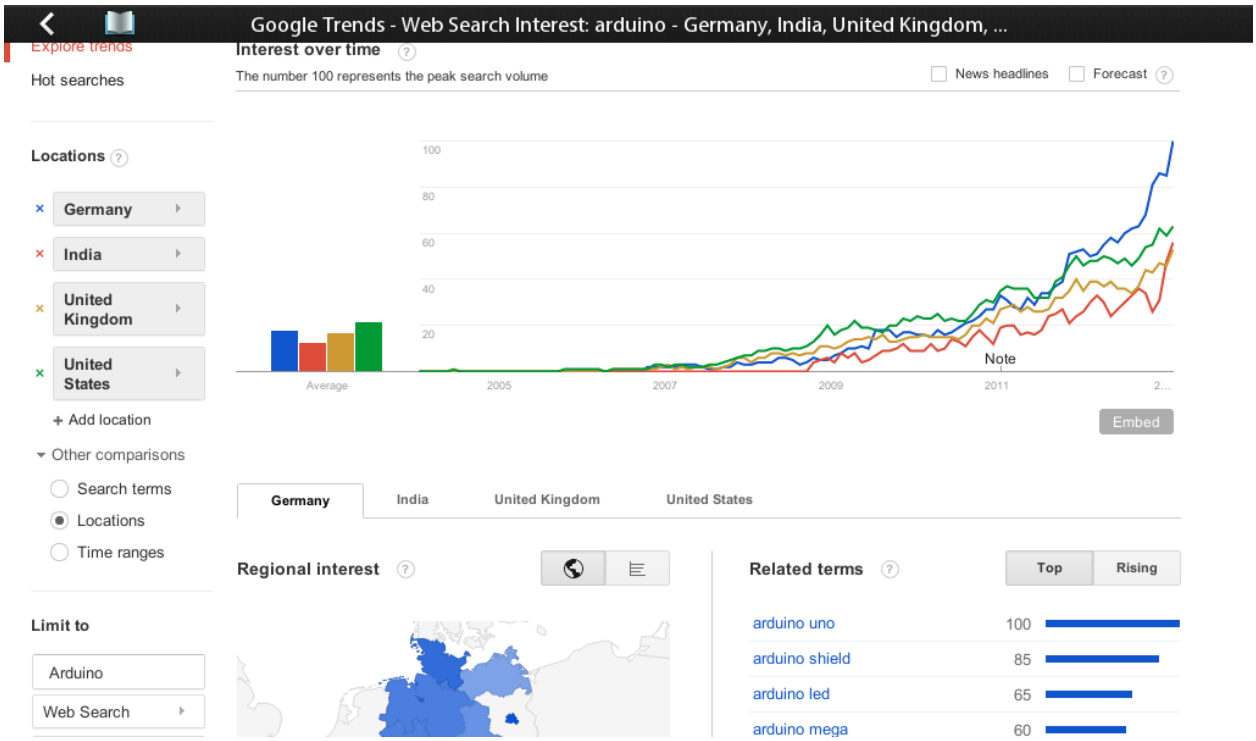
"We're working on the Arduino Tre, which will be a super small Linux machine, kind of like the Raspberry Pi, that can work with a screen and a keyboard. But even simpler than that, because we want people without a lot of experience in technology to be able to just plug it in and use it. Even if they don't understand Linux yet."

Massimo Banzi, 2014

2004, Massimo Banzi je samo trebao odgovarajući učiteljski alat kako bi vlastite studente približo elektronici, danas nije tako lako pretpostaviti Arduinove ciljeve. Tada je Arduino bio inovacija, dok danas postoji mnogo sličnih i diskutabilno boljih ili zanimljivijih drugih razvojnih podloga za razvoj. Nitko ne može predvidjeti budućnost, iz nekih citata osnivatelja tvrtke može se samo nagađati u kojem smjeru ide Arduino. Arduino trenutno zadovoljava potrebe krajnjeg korisnika i tako će i ostati u skorije vrijeme, s bezbroj novih inovacijskih štitova s mnogobrojnim novim mogućnostima, što dokazuju i grafovi Arduina kao Google trenda (slike 34. i 35.).



Slika 34. Graf rasta Arduina kao Google trenda u svijetu



Slika 35. Graf rasta Arduina kao Google trenda u Njemačkoj, Indiji, Ujedinjenom Kraljevstvu, Sjedinjenim Američkim Državama

4. ZAKLJUČAK

Arduino je preuzeo jedno područje robotike i mikroprocesora koje je bilo skupo i nedostupno većini te svojim inovacijama i cijenom unio promjene. Postoji još mnogo poslovnih područja koja ne rabe Arduino razvojno sučelje. Jedna od glavnih stvari koje Arduino može učiniti u poslovnom pogledu je smanjiti troškove eksperimetiranja te omogućiti tvrtkama da provode više vremena razvijajuči funkcionalnije proizvode, proizvode koji su lakše nadogradivi. Od medicinskih uređaja preko potrošaćke elektronike do PLC kontrolera, Arduino razvojno sučelje otvara potpuno novi niz mogućnosti. Upravljački sustavi za potrošačku elektroniku koje je razvila open source zajednica će tek izaći na vidjelo. Stvari će biti puno lakše koristiti, modificirati te unaprijediti po vlastitim željama, a moći će se nadzirati i upravljati bežično. Ukratko, budućnost Arduina je svijetla.

Ovaj rad je jednostavan prikaz osnovnih mogućnosti Arduina i njegovih potencijala na području robotike, elektronike, medicine te industrije i samo je dio jedne velike zajednice i njenih mogućnosti koja konstantno raste.

LITERATURA

[1] Arduino službena stranica; Upute; Uvod: https://www.arduino.cc/en/Guide/Introduction

[2] Arduino službena stranica; Upute; Naslovna strana: https://www.arduino.cc/en/Guide/HomePage

[3] Arduino službena stranica; *How to*: https://www.arduino.cc/en/main/howto

[4] Arduino službena stranica; Ploče: https://www.arduino.cc/en/Main/Boards

[5] Arduino forum; tema *Does Arduino have a future?*: https://forum.arduino.cc/index.php?topic=253582.0

[6] Arduino wikispaces; Senzorski štitovi: https://arduino-info.wikispaces.com/SensorShield

[7] Keyestudio službena stranica; O nama: http://www.keyestudio.com/about-us

[8] Keyestudio službena stranica; Keyes mini tenk robot: http://www.keyestudio.com/keyes-mini-tank-robot.html

[9] Keyestudio službena stranica: UNO R3 board: http://www.keyestudio.cc/h-nd-36.html

[10] Keyestudio službena stranica; L298P štit za motore: http://www.keyestudio.cc/h-nd-38.html

[11] Keyestudio službena stranica; V5 senzorski štit: http://www.keyestudio.cc/h-pd-57.html

[12] Keyestudio službena stranica; Bluetooth modul HC-06: http://www.keyestudio.cc/h-nd-129.html

[13] Random er tutorials; Arduino štitovi: https://randomnerdtutorials.com/25-arduino-shields/

[14] Sparkfun; HC-SR04 ultrazvučni senzor; Datasheet: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[15] Quora; *What is the future of Arduino?*: https://www.quora.com/What-is-the-future-of-Arduino

[16] Wikipedia En: Arduino;
https://en.wikipedia.org/wiki/Arduino
[17] Wikipedia Global; Keyestudio mini tank robot:
http://wiki.keyestudio.com/index.php/Ks0071_keyestudio_Mini_Tank_Robot
[18] Wikipedia Hr; Arduino:
https://hr.wikipedia.org/wiki/Arduino
[19] Wikipedia Hr; Robotika:
https://hr.wikipedia.org/wiki/Robotika
[20] Wikipedia Hr; Računalno programiranje:
https://hr.wikipedia.org/wiki/Ra%C4%8Dunalno_programiranje

POPIS SLIKA

POPIS TABLICA

POPIS KRATICA

CE (frl. *Conformité Européenne*)	europska sukladnost
COM (engl. *Component Object Model*)	komponenta modela objekta
DC (engl. D*irect Current*)	istosmjerna struja
DIY (engl. *Do It Yourself*)	napravi sam
EDR (engl. *Enchanced Data Rate*)	poboljšan prijenos podataka
FTDI (engl. *Future Tehnology Devices International*)	međunarodni uređaji buduće tehnologije
GND (engl. *GrouND*)	uzemljenje
GPS (engl. *Global Positioning System*)	globalni sistem pozicioniranja
I2C (engl. *Inter-Integrated Circuit*)	unutar integrirani sklop
ICSP (engl. *In-Circuit Serial Programming*)	serijsko programiranje unutar sustava
IDE (engl. *Integrated Development Enviroment*)	integrirano razvojno okruženje
ISM (engl. *Industrial,scietific and medical*)	industrijska,znanstvena i medicinska
LCD (engl. *Lyquid Crystal Display*)	zaslon s tekućim kristalima
MSDS (engl. *Material Safety Data Sheet*)	podatci o sigurnosti materijala
PLC (engl. *Programmable Logic Controller*)	programabilni logički kontroler
PWM (engl. *Pulse Width Modulation*)	širinsko impulsna modulacija
R3 (engl. *Revision 3*)	treća revizija
RoHS (engl. *Restriction of Hazardous Supstances*)	zabrana štetnih tvari
TTL (engl. *Transistor-Transistor Logic*)	tranzistor-tranzistorski logički sklop
U/I	ulazno/izlazni
UL (engl. *Underwriters Laboratories*)	Underwriters laboratoriji
USB (engl. *Universal Serial Bus*)	univerzalna serijska sabirnica

PRILOZI

PRILOG 1

```
#define echoPin 4 // Echo Pin
#define trigPin 5 // Trigger Pin
#define LEDPin 13 // Onboard LED
int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration, distance; // Duration used to calculate distance

void setup() {
 Serial.begin (9600);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(LEDPin, OUTPUT); // Use LED indicator (if required)
}

void loop() {
/* The following trigPin/echoPin cycle is used to determine the
 distance of the nearest object by bouncing soundwaves off of it. */
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);

 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);

 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);

 //Calculate the distance (in cm) based on the speed of sound.
 distance = duration/58.2;

 if (distance >= maximumRange || distance <= minimumRange){
 /* Send a negative number to computer and Turn LED ON
 to indicate "out of range" */
 Serial.println("-1");
 digitalWrite(LEDPin, HIGH);
 }
 else {
 /* Send the distance to the computer using Serial protocol, and
 turn LED OFF to indicate successful reading. */
 Serial.println(distance);
 digitalWrite(LEDPin, LOW);
 }

 //Delay 50ms before next reading.
 delay(50);
}
```

PRILOG 2

```
int val;
int ledpin=13;
void setup()
{
Serial.begin(9600);
 pinMode(ledpin,OUTPUT);
} void loop()
{ val=Serial.read();
if(val=='a')
 {
digitalWrite(ledpin,HIGH);
delay(250);
digitalWrite(ledpin,LOW);
delay(250);
 Serial.println("keyestudio");
}}
```

PRILOG 3

```
#include <Servo.h>
int pinLB = 12;     // define pin 12
int pinLF = 3;     // define pin 3
int pinRB = 13;    // define pin 13
int pinRF = 11;    // define pin 11

int inputPin = 4;    // define pin for sensor echo
int outputPin =5;    // define pin for sensor trig

int Fspeedd = 0;      // forward speed
int Rspeedd = 0;      // right speed
int Lspeedd = 0;      // left speed
int directionn = 0;   // forward=8 backward=2 left=4 right=6
Servo myservo;        // set myservo
int delay_time = 250; // settling time after steering servo motor moving B
int Fgo = 8;         // Move F
int Rgo = 6;         // move to the R
int Lgo = 4;         // move to the L
int Bgo = 2;         // move B
void setup()
 {
  Serial.begin(9600);     // Define motor output pin
  pinMode(pinLB,OUTPUT); // pin 12
  pinMode(pinLF,OUTPUT); // pin 3 (PWM)
  pinMode(pinRB,OUTPUT); // pin 13
  pinMode(pinRF,OUTPUT); // pin 11 (PWM)
  pinMode(inputPin, INPUT);    // define input pin for sensor
  pinMode(outputPin, OUTPUT);  // define output pin for sensor
  myservo.attach(9);    // Define servo motor output pin to D9 (PWM)
 }
```

```
void advance()     // move forward
   {
   digitalWrite(pinLB,LOW);    // right wheel moves forward
digitalWrite(pinRB, LOW);  // left wheel moves forward
   analogWrite(pinLF,255);
   analogWrite(pinRF,255);
   }
void stopp()        // stop
   {
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinRB,HIGH);
    analogWrite(pinLF,0);
    analogWrite(pinRF,0);
   }
void right()       // turn right (single wheel)
   {
  digitalWrite(pinLB,HIGH);  // wheel on the left moves forward
  digitalWrite(pinRB,LOW); // wheel on the right moves backward
  analogWrite(pinLF, 255);
  analogWrite(pinRF,255);
   }
void left()        // turn left (single wheel)
   {
  digitalWrite(pinLB,LOW);  // wheel on the left moves backward
  digitalWrite(pinRB,HIGH); // wheel on the right moves forward
  analogWrite(pinLF, 255);
  analogWrite(pinRF,255);
   }

void back()        // move backward
   {
    digitalWrite(pinLB,HIGH);  // motor moves to left rear
    digitalWrite(pinRB,HIGH);  // motor moves to right rear
    analogWrite(pinLF,255);
    analogWrite(pinRF,255);
   }
void detection()       // measure 3 angles (0.90.179)
   {
     int delay_time = 250;  // stabilizing time for servo motor after moving backward
     ask_pin_F();           // read the distance ahead
    if(Fspeedd < 10)        // if distance ahead is <10cm
     {
     stopp();              // clear data
     delay(100);
     back();               // move backward for 0.2S
     delay(200);
     }
     if(Fspeedd < 25)        // if distance ahead is <25cm
     {
      stopp();
```

```
    delay(100);            // clear data
    ask_pin_L();           // read distance on the left
    delay(delay_time);     // stabilizing time for servo motor
    ask_pin_R();           // read distance on the right
    delay(delay_time);     // stabilizing time for servo motor

    if(Lspeedd > Rspeedd)  // if distance on the left is >distance on the right
    {
     directionn = Lgo;     // move to the L
    }

    if(Lspeedd <= Rspeedd)  // if distance on the left is <= distance on the right
    {
     directionn = Rgo;     // move to the right
    }
     if (Lspeedd < 10 && Rspeedd < 10) // if distance on left and right are both
                                          <10cm
    {
     directionn = Bgo;     // move backward
    }}
   else                    // if distance ahead is >25cm
   {
    directionn = Fgo;      // move forward
   }}
void ask_pin_F()   // measure the distance ahead
  {
   myservo.write(90);
   digitalWrite(outputPin, LOW);   // ultrasonic sensor transmit low level signal
                                      2μs
   delayMicroseconds(2);
   digitalWrite(outputPin, HIGH);  // ultrasonic sensor transmit high level
                                      signal10μs, at least 10μs
   delayMicroseconds(10);
   digitalWrite(outputPin, LOW);    // keep transmitting low level signal
   float Fdistance = pulseIn(inputPin, HIGH);  // read the time in between
   Fdistance= Fdistance/5.8/10;       // convert time into distance (unit: cm)
   Fspeedd = Fdistance;             // read the distance into Fspeedd
  }
 void ask_pin_L()   // measure distance on the left
  {
   myservo.write(5);
   delay(delay_time);
   digitalWrite(outputPin, LOW);   // ultrasonic sensor transmit low level signal
                                      2μs
   delayMicroseconds(2);
   digitalWrite(outputPin, HIGH);  // ultrasonic sensor transmit high level
                                      signal10μs, at least 10μs
   delayMicroseconds(10);
   digitalWrite(outputPin, LOW);    // keep transmitting low level signal
   float Ldistance = pulseIn(inputPin, HIGH);  // read the time in between
```

```
    Ldistance= Ldistance/5.8/10;       // convert time into distance (unit: cm)
    Lspeedd = Ldistance;              // read the distance into Lspeedd
   }
void ask_pin_R()   //  measure distance on the right
   {
    myservo.write(177);
    delay(delay_time);
    digitalWrite(outputPin, LOW);   // ultrasonic sensor transmit low level signal
                                       2μs
    delayMicroseconds(2);
    digitalWrite(outputPin, HIGH);  // ultrasonic sensor transmit high level
                                       signal10μs, at least 10μs
    delayMicroseconds(10);
    digitalWrite(outputPin, LOW);    // keep transmitting low level signal
    float Rdistance = pulseIn(inputPin, HIGH);  // read the time in between
    Rdistance= Rdistance/5.8/10;       // convert time into distance (unit: cm)
    Rspeedd = Rdistance;              // read the distance into Rspeedd
   }
   void loop()
 {
  myservo.write(90);  // home set the servo motor, ready for next measurement
  detection();       // measure the angle and determine which direction to move
 if(directionn == 2)  // if directionn= 2
 {
  back();
  delay(800);                 //  go backward
  left() ;
  delay(200);              // Move slightly to the left (to prevent stuck in dead end)
 }
 if(directionn == 6)          // if directionn = 6
 {
  back();
  delay(100);
  right();
  delay(600);               // turn right
 }
 if(directionn == 4)          // if directionn = 4
 {
  back();
  delay(600);
  left();
  delay(600);                // turn left
 }
 if(directionn == 8)          // if directionn = 8
 {
  advance();               // move forward
  delay(100);
 } }
```

PRILOG 4

```
int pinLB = 12;     // define pin 12
int pinLF = 3;     // define pin 3
int pinRB = 13;    // define pin 13
int pinRF = 11;    // define pin 11
int val;
void setup()
 {
  Serial.begin(9600);     // define pin for motor output
  pinMode(pinLB,OUTPUT); // pin 12
  pinMode(pinLF,OUTPUT); // pin 3 (PWM)
  pinMode(pinRB,OUTPUT); // pin 13
  pinMode(pinRF,OUTPUT); // pin 11 (PWM)
 }
void advance()     // move forward
    {
    digitalWrite(pinLB,LOW);    // right wheel moves forward
    digitalWrite(pinRB, LOW);  // left wheel moves forward
    analogWrite(pinLF,255);
    analogWrite(pinRF,255);
    }
void stopp()        // stop
    {
     digitalWrite(pinLB,HIGH);
     digitalWrite(pinRB,HIGH);
     analogWrite(pinLF,0);
     analogWrite(pinRF,0);
    }
void right()       // turn right (single wheel)
    {
   digitalWrite(pinLB,HIGH);  // left wheel moves forward
   digitalWrite(pinRB,LOW); // right wheel moves backward
   analogWrite(pinLF, 255);
   analogWrite(pinRF,255);
    }
void left()        // turn left (single wheel)
   {
   digitalWrite(pinLB,LOW);  // left wheel moves forward
   digitalWrite(pinRB,HIGH); // right wheel moves backward
   analogWrite(pinLF, 255);
   analogWrite(pinRF,255);
    }
  void back()          // move backward
    {
     digitalWrite(pinLB,HIGH);  // motor moves to left rear
     digitalWrite(pinRB,HIGH);  // motor moves to right rear
     analogWrite(pinLF,255);
     analogWrite(pinRF,255);
    }
void loop()
```

```
    { val=Serial.read();
    if(val=='U')advance();
    if(val=='D')back();
    if(val=='R')left() ;
    if(val=='L')right();
if(val=='S')stopp();
}
```

PRILOG 5

```
#include <Servo.h>
int pinLB = 12;     // define pin 12
int pinLF = 3;     // define pin 3
int pinRB = 13;    // define pin 13
int pinRF = 11;    // define pin 11
int inputPin = 4;    // define pin for sensor echo
int outputPin =5;    // define pin for sensor trig
int Fspeedd = 0;      // forward speed
int Rspeedd = 0;      // right speed
int Lspeedd = 0;      // left speed
int directionn = 0;   // forward=8 backward=2 left=4 right=6
Servo myservo;        // set myservo
int delay_time = 250; // settling time after steering servo motor moving B
int Fgo = 8;         // Move F
int Rgo = 6;         // move to the R
int Lgo = 4;         // move to the L
int Bgo = 2;         // move B
void setup()
 {
  Serial.begin(9600);     // Define motor output pin
  pinMode(pinLB,OUTPUT); // pin 12
  pinMode(pinLF,OUTPUT); // pin 3 (PWM)
  pinMode(pinRB,OUTPUT); // pin 13
  pinMode(pinRF,OUTPUT); // pin 11 (PWM)
  pinMode(inputPin, INPUT);    // define input pin for sensor
  pinMode(outputPin, OUTPUT);  // define output pin for sensor
  myservo.attach(9);    // Define servo motor output pin to D9 (PWM)
 }
void advance()     // move forward
    {
    digitalWrite(pinLB,LOW);    // right wheel moves forward
digitalWrite(pinRB, LOW);  // left wheel moves forward
    analogWrite(pinLF,255);
    analogWrite(pinRF,255);
    }
void stopp()        // stop
    {
     digitalWrite(pinLB,HIGH);
     digitalWrite(pinRB,HIGH);
     analogWrite(pinLF,0);
     analogWrite(pinRF,0);
```

```
  }
void right()       // turn right (single wheel)
  {
  digitalWrite(pinLB,HIGH);  // wheel on the left moves forward
  digitalWrite(pinRB,LOW); // wheel on the right moves backward
  analogWrite(pinLF, 255);
  analogWrite(pinRF,255);
  }
void left()        // turn left (single wheel)
  {
  digitalWrite(pinLB,LOW);  // wheel on the left moves backward
  digitalWrite(pinRB,HIGH); // wheel on the right moves forward
  analogWrite(pinLF, 255);
  analogWrite(pinRF,255);
  }

void back()        // move backward
  {
   digitalWrite(pinLB,HIGH);  // motor moves to left rear
   digitalWrite(pinRB,HIGH);  // motor moves to right rear
   analogWrite(pinLF,255);
   analogWrite(pinRF,255);
  }
void detection()       // measure 3 angles (0.90.179)
  {
   int delay_time = 250;  // stabilizing time for servo motor after moving backward
   ask_pin_F();          // read the distance ahead
   if(Fspeedd < 10)        // if distance ahead is <10cm
   {
   stopp();              // clear data
   delay(100);
   back();               // move backward for 0.2S
   delay(200);
   }
   if(Fspeedd < 25)        // if distance ahead is <25cm
   {
    stopp();
    delay(100);           // clear data
    ask_pin_L();          // read distance on the left
    delay(delay_time);     // stabilizing time for servo motor
    ask_pin_R();          // read distance on the right
    delay(delay_time);     // stabilizing time for servo motor

    if(Lspeedd > Rspeedd)   // if distance on the left is >distance on the right
    {
     directionn = Lgo;     // move to the L
    }
     if(Lspeedd <= Rspeedd)   // if distance on the left is <= distance on the right
    {
     directionn = Rgo;     // move to the right
```

```
      }
      if (Lspeedd < 10 && Rspeedd < 10)   // if distance on left and right are both
                                              <10cm
      {
       directionn = Bgo;      // move backward
      } }
     else                    // if distance ahead is >25cm
     {
      directionn = Fgo;       // move forward
     }}
void ask_pin_F()   // measure the distance ahead
   {
     myservo.write(90);
     digitalWrite(outputPin, LOW);   // ultrasonic sensor transmit low level signal
                                        2μs
     delayMicroseconds(2);
     digitalWrite(outputPin, HIGH);  // ultrasonic sensor transmit high level
                                        signal10μs, at least 10μs
     delayMicroseconds(10);
     digitalWrite(outputPin, LOW);    // keep transmitting low level signal
     float Fdistance = pulseIn(inputPin, HIGH);  // read the time in between
     Fdistance= Fdistance/5.8/10;       // convert time into distance (unit: cm)
     Fspeedd = Fdistance;              // read the distance into Fspeedd
  Serial.print("Fspeedd = ");
  Serial.print(Fspeedd );
  Serial.println("  cm");
   }
 void ask_pin_L()   // measure distance on the left
   {
     myservo.write(5);
     delay(delay_time);
     digitalWrite(outputPin, LOW);   // ultrasonic sensor transmit low level signal
                                        2μs
     delayMicroseconds(2);
     digitalWrite(outputPin, HIGH);  // ultrasonic sensor transmit high level
                                        signal10μs, at least 10μs
     delayMicroseconds(10);
     digitalWrite(outputPin, LOW);    // keep transmitting low level signal
     float Ldistance = pulseIn(inputPin, HIGH);  // read the time in between
     Ldistance= Ldistance/5.8/10;       // convert time into distance (unit: cm)
     Lspeedd = Ldistance;              // read the distance into Lspeedd
  Serial.print("Lspeedd = ");
  Serial.print(Lspeedd );
     Serial.print("  cm  ");
   }
void ask_pin_R()   //  measure distance on the right
   {
     myservo.write(177);
     delay(delay_time);
     digitalWrite(outputPin, LOW); //ultrasonic sensor transmit low level signal 2μs
```

```
    delayMicroseconds(2);
    digitalWrite(outputPin, HIGH);  // ultrasonic sensor transmit high level
                                       signal10μs, at least 10μs
    delayMicroseconds(10);
    digitalWrite(outputPin, LOW);    // keep transmitting low level signal
    float Rdistance = pulseIn(inputPin, HIGH);  // read the time in between
    Rdistance= Rdistance/5.8/10;       // convert time into distance (unit: cm)
    Rspeedd = Rdistance;             // read the distance into Rspeedd
  Serial.print(" Rspeedd = ");
  Serial.print(Rspeedd );
    Serial.println("  cm");
   }
void loop()
 {
   myservo.write(90);  // home set the servo motor, ready for next measurement
   detection();       // measure the angle and determine which direction to move
  if(directionn == 2)  // if directionn= 2
  {
    back();
    delay(800);                 //  go backward
    left() ;
    delay(200);             // Move slightly to the left (to prevent stuck in dead end)
  }
  if(directionn == 6)          // if directionn = 6
  { back();
    delay(100);
    right();
    delay(600);               // turn right
  }
  if(directionn == 4)         // if directionn = 4
  { back();
    delay(600);
    left();
    delay(600);               // turn left
  }
  if(directionn == 8)         // if directionn = 8
  { advance();       // move forward
   delay(100);
  }}
```