DIGITALNI BLIZANCI

Seminarski rad

Kolegij: Programiranje mobilnih robota i letjelica

Studenti: Ante Radačić, Matino Brajković, Mate Ćosić Split, 24.01.2023.

Sadržaj

[1 UVOD 3](#_Toc125726894)

[2 Implementacija Arduino dijela aplikacije 4](#_Toc125726895)

[2.1 nRF24L01 modul 4](#_Toc125726896)

[2.2 L298N upravljač motora 5](#_Toc125726897)

[2.3 Arduino mikrokontroler odašiljač (eng. *transmitter*) 7](#_Toc125726898)

[2.4 Arduino mikrokontroler primatelj (eng*. receiver*) 9](#_Toc125726899)

[3 Implementacija Unity dijela aplikacije 11](#_Toc125726900)

[3.1 Unity virtualni prostor 11](#_Toc125726901)

[3.2 Skripta TileScript.cs 11](#_Toc125726902)

[3.3 Skripta SenzorScript.cs 12](#_Toc125726903)

[3.4 Skripta GoToScript.cs 13](#_Toc125726904)

[4 Zaključak 18](#_Toc125726905)

[5 Literatura 19](#_Toc125726906)

# UVOD

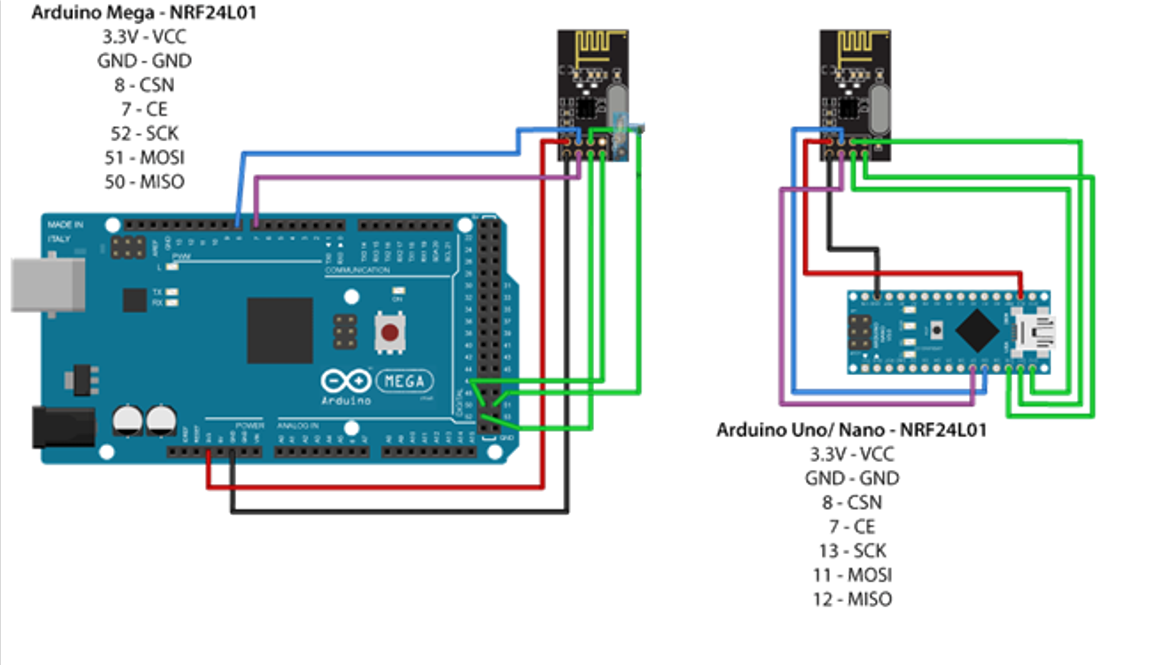
Tema ovog seminarskog rada su dva robota, jedan fizički i jedan virtualni robot. Unutar Unity razvojnog okruženja se autonomno kreće virtualni robot u virtualnom prostoru. Njegove kretnje u fizičkom svijetu prati fizički robot (automobil kojim upravlja Arduino mikrokontroler). Upravljačke podatke potrebne fizičkom robotu da se giba slali bi iz Unity virtualnog svijeta serijskim portom na Arduino mikrokontroler (transmiter) koji ima nRF modul, a zatim bi taj isti Arduino mikrokontroler slao podatke bežično putem radio frekvencija Arduino mikrokontroleru sa nRF modulom koji se nalazi na fizičkom automobilu. Fizički automobil bi trebao imitirati kretnje virtualnog odnosno ponašati se kao njegov blizanac.

# Implementacija Arduino dijela aplikacije

## nRF24L01 modul

Jako popularan izbor za uspostavu bežične komunikacije između Arduino mikrokontrolera je nRF24L01 modul. Ovaj modul koristi radio frekvencije (2.4 GHz) da bi slao i primao signale i radi sa brzinom prijenosa od 250kbps do 2Mbps. Ako se koristi na otvorenom i sa manjom brzinom prijenosa može doseći udaljenost do 100 metara.

Modul koristi 125 različitih kanala što omogućava uspostavljanje mreže od 125 nezavisnih modula na jednom mjestu. Svaki kanal može imati 6 adresa, odnosno svaki modul može komunicirati sa do 6 drugih modula istodobno. Potrošnja ovog modula je manja od jedne ledice, a iznosi 12mA tijekom slanja. Operativni napon mu ide od 1.9 do 3.6V, ali ostali pinovi toleriraju 5V pa ga je jednostavno povezati sa Arduinom. Modul posjeduje 8 pinova, od kojih su tri za SPI komunikaciju i moraju biti povezani na SPI pinove na Arduinu(Slika 1). Bitno je naglasiti da različiti Arduino mikrokontroleri imaju različite SPI pinove. Pinovi CSN i CE se mogu spojiti na bilo koji digitalni pin na Arduinu i koriste se za postavljanje modula u pripravnost ili aktivni mod te za promjenu između moa za slanje i komandnog moda. Ostali pinovi su za uzemljenje, napajanje i pin za prekidanje.



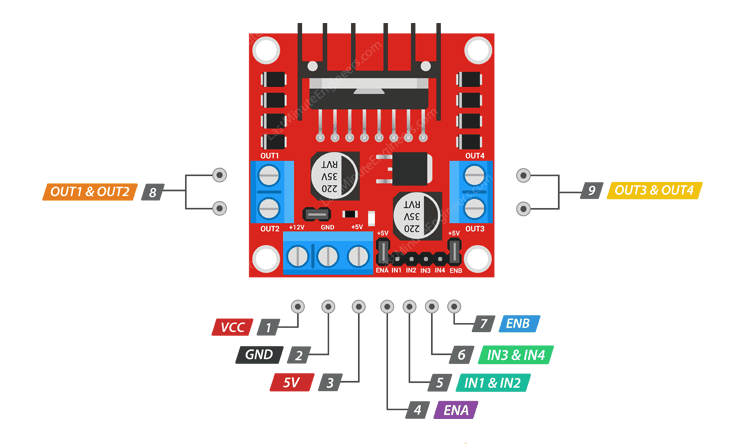
Slika 1 Povezivanje dvaju Arduino mikorkontrolera sa nRFmodulima

## L298N upravljač motora

Za upravljanje motorima na automobilu je korišten L298N upravljač motora koji znatno olakšava upravljanje DC motorima. Kontrolira brzinu i smijer vrtnje motora i omogućava kombiniranje istih. Za kontrolu brzine koristi se PWM (eng. *Pulse Width Modulation*) gdje je brzina motora kontrolirana promjenom ulaznog napona . Prosječni napon se dobije slanjem serije ON-OFF pulseva, a prosječan napon je jednak širini pulseva i označava se kao radni ciklus. Što je veći radni ciklus veća je i prosječni napon usmjeren na motore, što rezultira povečanjem brzine motora i obrnuto. Smijer vrtnje motora se kontrolira promjenom polariteta napona koristeći *H-bridge* tehniku. *H-bridge* čip se sastoji od četiri sklopke i zatvaranjem specifičnih sklopki u isto vrijeme mijenja se polaritet napona, a samim time i smijer toracije motora.

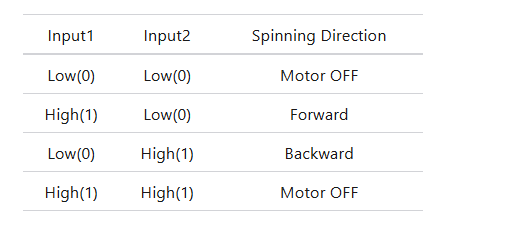
L298N upravljač motora ima jedanaest pinova :

* VS – napajanje za H-bridge koji pokreće motore (5-12V)
* VSS – napajanje za logičke sklopove (5 ili 7V)
* GND – uzemljenje
* OUT1 i OUT2 – za spajanje motora A (5-12V DC motor)
* OUT3 i OUT4 – za spajanje motora B (5-12V DC motor)
* IN1 i IN2 – pinovi koji odlučuju smijer okretanja motora A
* IN3 i IN4 – pinovi koji odlučuju smijer okretanja motora B
* ENA – pin koji pali/gasi i kontrolira brzinu motora A
* ENA – pin koji pali/gasi i kontrolira brzinu motora A
* ENB – pin koji pali/gasi i kontrolira brzinu motora B



Slika 2 L298N upravljač motora

Smijer vrtnje motora se kontrolira namještajući *HIGH* (5V) ili *LOW* (uzemljenje) u različitim kombijacijama na ovim ulaznim pinovima (*IN* pinovi).



Slika 3 Logika smijera vrtnje

Namještajući *EN* pinove na *HIGH* pokreće motore, a na *LOW* ih zaustavlja. Koristeći *PWM* možemo kontrolirati brzine motora (skala od 0 do 255). PWM možemo kiristiti ako sklonimo *jumper* sa ovih pinova i spojimo ih na Arduino *PWM* pinove. Ako je *jumper* na mjestu motori se vrte maksimalnom brzinom kada su postavljeni na *HIGH*.

Upravljač motora je preko pinova ENA, IN1, IN2, ENB, IN3, IN4 spojen na Arduino na digitalne pinove 1,2,8,3,7,6. Pinovi 1 i 3 imaju omogućem *PWM*. U našem slučaju ENA, IN1 i IN2 služe za upravljanje sa dva lijeva motora, a pinovi ENB, IN3 i IN4 za upravljanje desna dva motora.

## Arduino mikrokontroler odašiljač (eng. *transmitter*)

Nakon što smo povezali module sa mikrokontrolerima potrebno je napraviti kod za modula za odašiljanje (Slika 4) i za prijemni modul(Slika 5). Prvo je potrebno instalirati biblioteke potrebne za rad, a to su SPI za serijsku komunikaciju i RF24 biblioteka za nRF modul. Slijede kodova za pojedini modul.

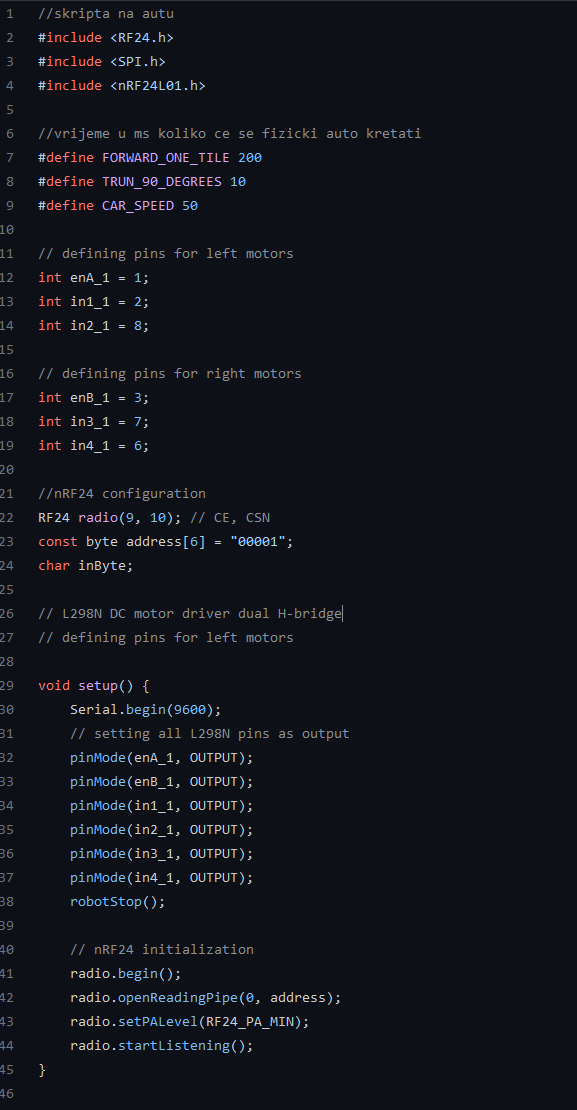


Slika 4 Kod odašiljača

Na početku smo uključili potrebne biblioteke u projekt i kreirali objekt RF24. Argumenti za kreaciju su CSN i CE pinovi prethodno opisani. Zatim kreiramo bajt niz koji će sadržavati adresu modula s kojim komuniciramo, odnosno cijev kroz koju će moduli komunicirati. Dakle obje strane u komunikaciji imaju istu adresu u našem slučaju. U *Setup* funkciji inicijalizirali smo serijsku komunikaciju i radio objekt te koristeći *radio.openWritingPipe(address)* funkciju postavljamo adresu prijemnika kojem ćemo slati podatke. Koristeći *radio.setPALevel(RF24\_PA\_MIN)* funkciju postavljamo pojačivač snage na određeni leve, a u našem slučaju to je na minimum jer udaljenost između modula nije velika. Sljedeće imamo *radio.stopListening()* funkciju koja postavlja modul kao odašiljač. Unutar *loop* funkcije na odašiljaču prvo provjerimo ima li dolaznih podataka na serijskom portu koji dolaze od Unityja sa *Serial.available()* i ako podataka ima čitamo ih i spremamo u *inByte* karakter varijablu sa pozivom funkcije *Serial.read().* Nakon čitanja podatke šaljemo na prijemnik modul (na autiću) pozivom funkcije *radio.write(&inByte, sizeof(inByte))*. Između uzastopnih slanja podataka čeka se jedna sekunda.

## Arduino mikrokontroler primatelj (eng*. receiver*)

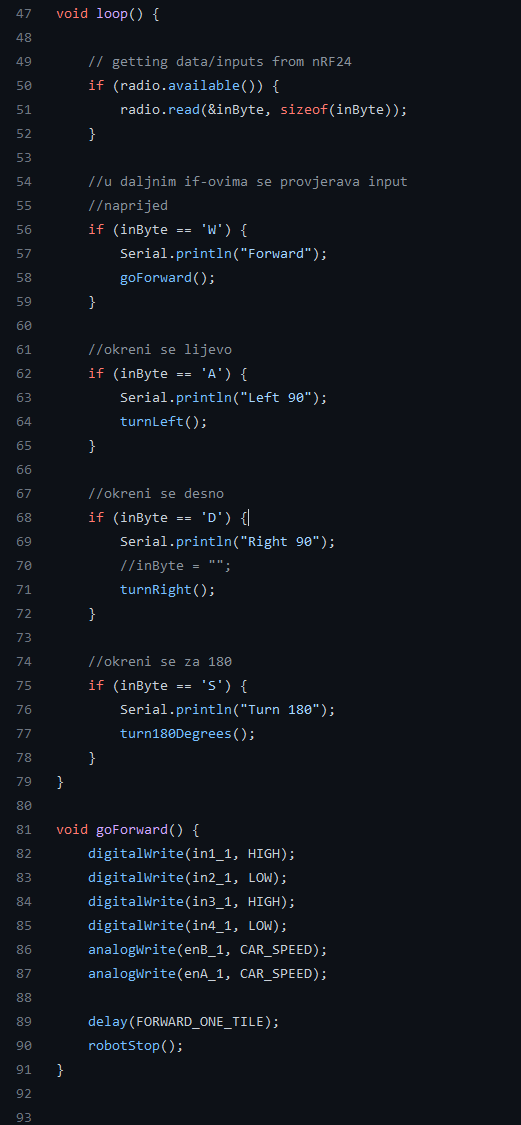
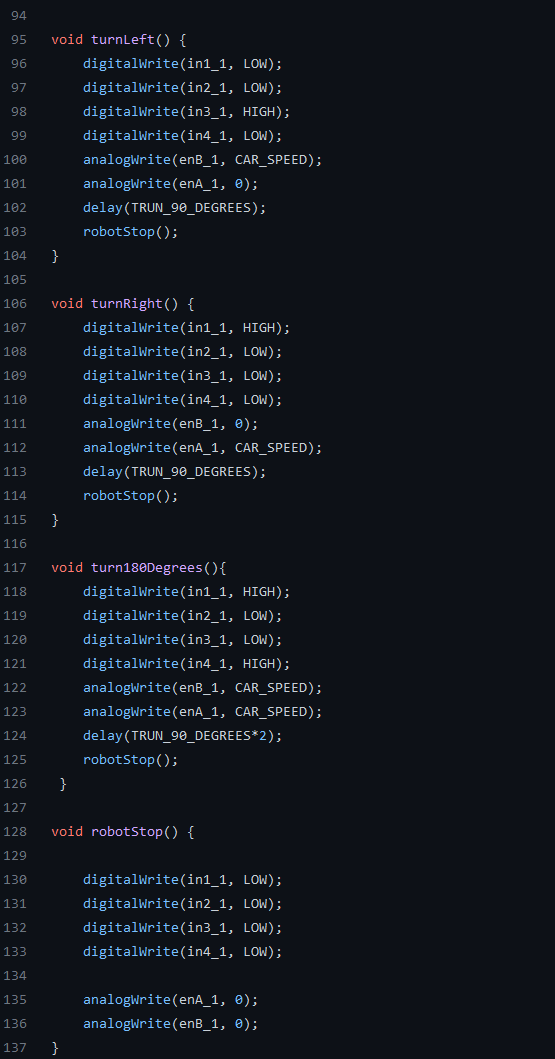
Dio koda od mikrokontrolera primatelja koji je zadužen za uspostavu radio komunikacije je gotovo identičan sa istim dijelom koda od mikrokontrolera koji šalje podatke. Razlika je samo u pozivu funkcije *radio.startListening()* koja postavlja taj Arduino kao primatelja koji zatim provjerava da li ima doalznih podataka preko radio komunikacije funkcijom *radio.available()*, te ako ima pdataka on ih čita funkcijom *radio.read(&inByte, sizeof(inByte))* i sprema ih u karakter varijablu.



Slika 5 Kod prijemnika

Unutar skripte se uključuju iste biblioteke potrebne za rad kao i u prethodnoj skripti, te se definiraju vremena u milisekundama koja odrečuju vrijeme kretanja fizičkog automobila. Nakon toga definiraju se pinovi koji kontroliraju smijer vrtnje i brzinu motora koji se u *setup* funkciji postavljaju kao izlazni pinovi.

U *loop* funkciji se provodi ponavljajuće čitanje podataka primljenih od Unity okruženja te ovisno o primljeni podatcima poziva se prikladna funkcija koja izvršava namještanje pinova za kontrolu smijera vrtnje i brzine motora. U svakoj funkciji čeka se određeno vrijeme potrebno da fizički robot izvrši radnju te se robot zaustavlja. Na primjer u funkciji goFOrward postavljaju se pinovi za smijer vrtnje za sva četiri motora na HIGH-LOW kombinaciju što znači da se vrte naprije, a brzina se postavlja na određenu vrijednost. Ove akcije na pinovima se mijenjaju u skladu s potrebnom akcijom.

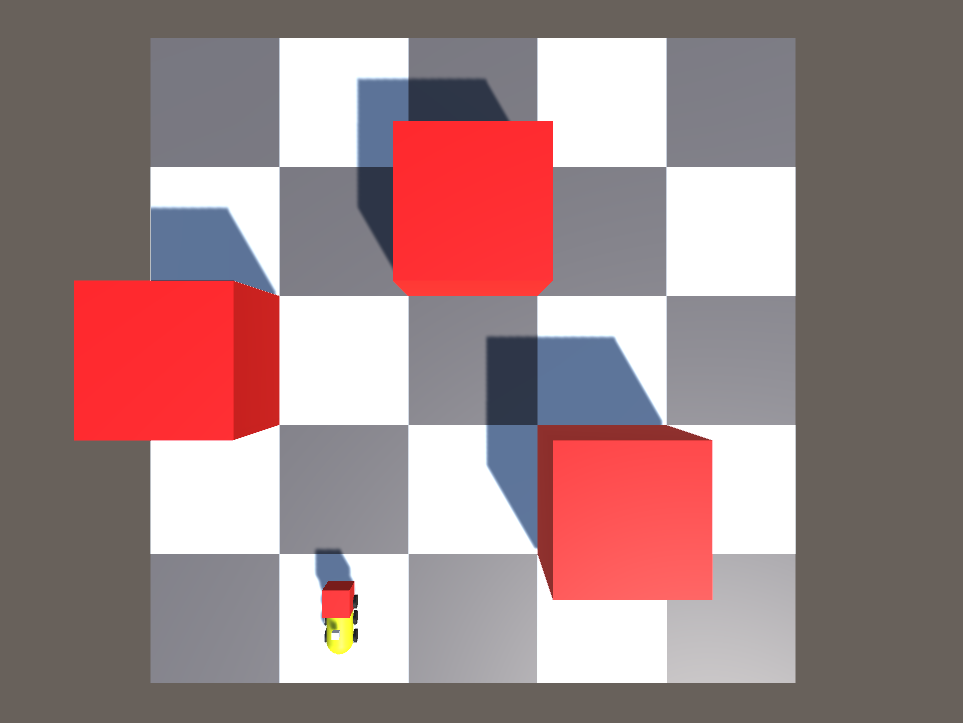


Slika 6 Funkcija loop i funkcija za akciju naprijed Slika 7 Funkcije za ostale akcije

# Implementacija Unity dijela aplikacije

## Unity virtualni prostor

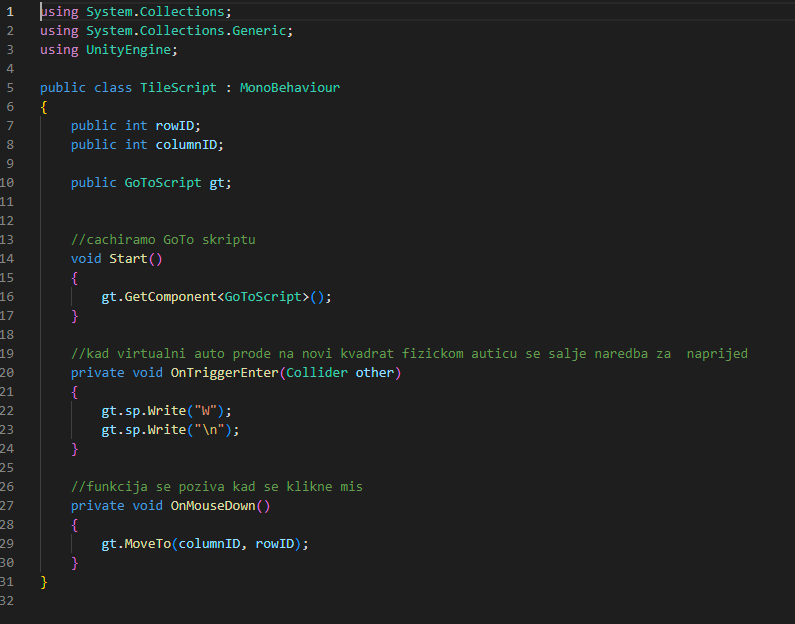
Unity virutalni prostor (Slika 8) sastoji se od mreže kvadrata po kojima se virtualni automobil autonomno giba i nekoliko prepreka koje mora zaobići na putu do odredišta. Automobil ima na sebi senzor kako bi detektirao prepreku. Odredište se automobilu zadaje lijevim klikom miša na kvadrat u koji želimo da on ode.



Slika 8 Virtualni prostor u kojem se giba virtualni automobil

## Skripta TileScript.cs

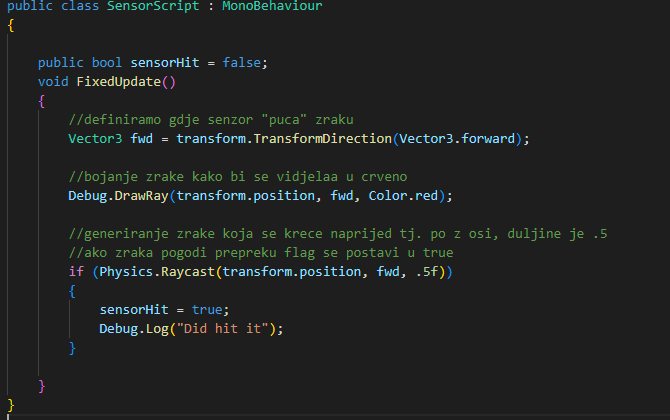
Ova skripta povezuje skriptu u kojoj je isprogramirano autonomno gibanje automobila (*GoToScript.cs*) sa kvadratima na mreži po kojoj se automobil giba. Nakon što spremimo objekt *gt* tipa *GoToScipt* skripte definira se da automobil nakon sudara sa sljedećim kvadratom preko objekta gt i serijskog porta šalje fizičkom automobilu naredbu da ide naprijed (skripta *GoToScript* je povezana sa automobilom). Dakle automobil se zapravo giba kvadrat po kvadrat. Na kraju je definirana funkcija koja određuje da se pozove funkcija *MoveTo* definirana unutar *GoToScript* skripte koja kao argument prima x i z koordinatu odrediđnog kvadrata.



Slika 9 Kod TileScript.cs skripte

## Skripta SenzorScript.cs

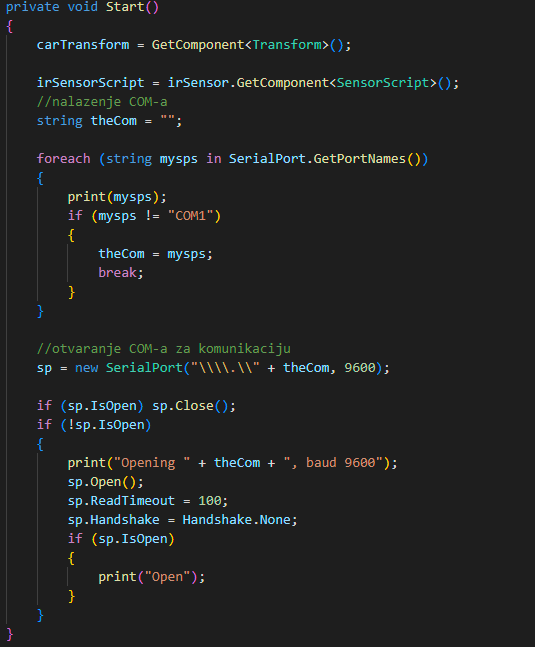
Unutar skripte koja definira ponašanje senzora na virtualnom automobila definira se *RayCast* zraka koja se ispaljuje sa prednje strane automobila pravo. Zraka je potavljena na određenu duljinu i ako pogodi prepreku to znači da je potrebno mijenjati smijer robota i postavlja se zastavica, koja se odnosi na to da li je automobil naišao na prepreku, u stanje istine što će poslužiti automobilu prilikom definiranja njegovog gibanja unutar *GoToScript* skripte. Zraka se uvijek šalje naprijed odnosno po z osi jer se rotira sa automobilom i njegov koordinatni sustav.



Slika 10 Kod SenzoScript.cs skripte

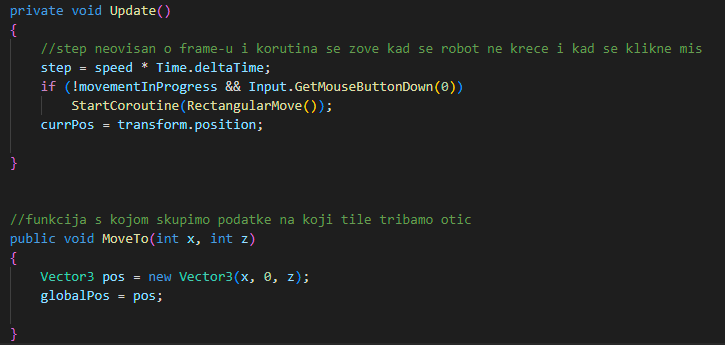
## Skripta GoToScript.cs

GoToSkript skripta je glavna skripta unutar koje se pored definiranja kretanja automobila do odredišta odvija i serijska komunikacija sa Arduino mikrokontrolerom koji će slati podatke fizičkom automobilu (*transmitter*). Šalju se karakteri koji će sukladno odraditi akcije gibanja na virtualnom i fizičkom automobilu. Unutar *start* funkcije postavljaju se komponente potrebne za upravljanje virtualnim automobilom. Dohvati se prethodno opisana skripta senzora kako bi se mogla detektirati prepreka i dohvati se *transform* komponenta objekta automobila unutar scene kako bi se mogao dohvatiti njegoc položaj. Nakon toga pretraže se svi serijski potrovi kako bi se pronašao odgovarajući te se odgovarajući otvori za komunikaciju sa Arduinom sa propusnom širinom od 9600 bitova u sekundi.



Slika 11 Start funkcija

*Update* funkcija namiješta step koji će se u sljedećim funkcijama koristiti za gibanje automobila. Step je neovisan o *frame-u* i kretat će se uvijek jednako. Korutina RectangularMove se poziva kada se robot ne kreće (kada je zavrio prethodno gibanje) i kada se krlikne mišem na kvadrat. MoveTo je funkcija koja se poziva iz skripte zakačene na kvadrat objekt i ona dohvaća x i z koordinate odredišnog kvadrata na koji se klikne.



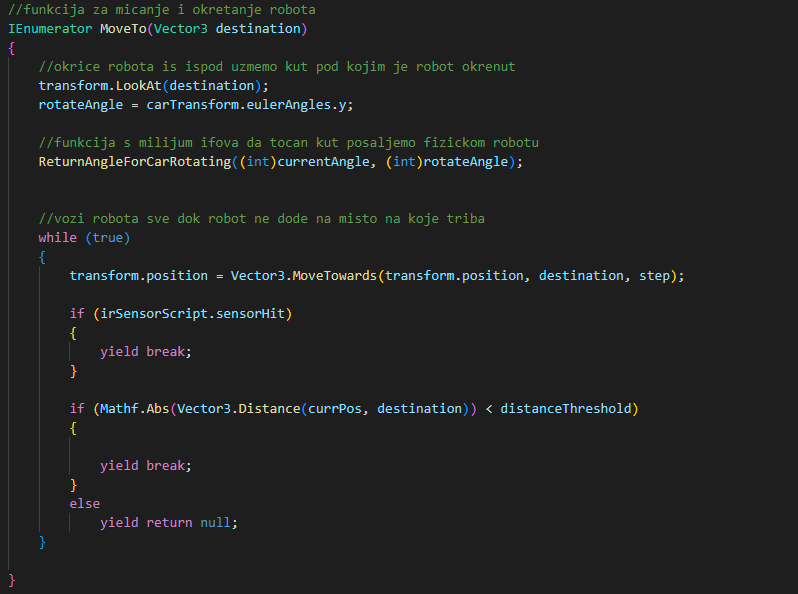
Slika 12 Update i MoveTo funkcije

Funkcija *RectangularMove* rastavlja vektor gibanja automobila na x i z os. Prvo se odvija gibanje na x osi a zatim po z osi. Dakle prvo odredi *target* vektor za x os i proslijedi ga MoveTo korutini, a kada se završi gibanje isto se ponovi za z os. Ova funkcija isto tako vodi brigu o zaobilaženju prepreke koristeći već spomenutu zastavicu *senzorHit.* Zaobilaženje se odvija tako da automobilu proslijedimo sljedeći slobodni kvadrat da se giba na njega i zatim nastavi do cilja. Zaoblijaženje prepreke je isto tako rastavljeno na slučaj ako se nalazimo u odgovarajućek redku ili stupcu. Posljednja provjera uvjeta ograničava automobil da prilikom zaobilaženja prepreka ne izađe sa mreže kvadrata.



Slika 13 Kod RectangularMove funkcije

Korutina *MoveTo* se poziva uvijek kada funkcija *RectangularMove* odredi *target* vektor na pojedinoj osi. Ona okreće robota koristeći *target* vektor kao argument funkcije LookAt. Nakon ove akcije uzima se kut za koji će u usporedbi na predhodni kut (koji se postavlja svaki put akda se pronađe novi target vektor) okretati fizički automobi i na osnovu kojega će se slati naredba fizičkom automobilu. Funkcija *MoveTowards* koristi prethodno definirani step kako bi prebacila objekt automobila do odredišta. Korutina *MoveTo* se prekida ako automobil naiđe na prepreku ili je udaljenost na koju kliknemo premala.



Slika 14 MoveTo funkcija

I za kraj ostaje funkcija *ReturnAngleForCarRotating* koja kako joj i samo ime govori određuje kut za koji se fizički automobil treba okrenuti. Fizički kut ovisi o trenutnom kutu kako je postaljen automobil prije kretanja i kutu koji se dobije akd se izraluna novi target vektor i okrene se automobil prema njemu. Na slici 15 je vidljiv dio slučajeva postavljanja odgovarajućeg kuta u skladu sa svim navedenim ujvetima. Kad je isti kut u kojem treba biti usmjeren fizički automobil ne šalje se ništa jer bi se automobil okrenuo bespotrebno.



Slika 15 Dio koda funkcije ReturnAngleForCarRotating

Na dnu funkcije se nalazi *switch case* koji će zapravo preko serijskog porta i ogovarajućeg kuta za rotaciju fizičkog automobila slati (odnosno pisati sa *sp.Write()* na serijski port) naredbu za okretanje Arduino odašiljaču. Šalje se i „\n“ kontrolni znak kako fizički robot ne bi stalno izvodio istu naredbu.



Slika 16 Slanje upravljačkih naredbi za rotaciju

# Zaključak

Komunikacija između Unityja i Arduino mikrokontrolera zaslužnog za odašiljanje uspječno je realizirana preko serijskog porta. Isto tako komunikacija radio valovima između mikrokontrolera na fizičkom automobilu i mokrokontrolera spojenog na računalo je uspješno realizirana preko dvaju nRF modula i pripadnih biblioteka. Virtualni automobil se nakon korisnikovog klika unutar Unity okruženja samostalno prebaci do odredišta izbjegavajući prepreke, a fizički robot ga prati ukoliko nema greške u komunikaciji. Virtualni i fizički automobil se ponašaju kao „digitalni bliznaci“.

# Literatura