ŠOLSKI CENTER CELJE
SREDNJA ŠOLA ZA KEMIJO, ELEKTROTEHNIKO
IN RAČUNALNIŠTVO

ROBOT NA WI-FI VODENJE

RAZISKOVALNA NALOGA

Elektrotehnika

Avtorja, Lah Sebastjan, E4A
Gutenberger Jan, E4A

Mentor, Kramer Gregor, univ. dipl. inž. el.

Mestna občina Celje, Mladi za Celje
Šentjur, 2021

KAZALO VSEBINE

KAZALO SLIK

POVZETEK

Za raziskovalno nalogo sva si izbrala vodenje robota na Wi-Fi povezavo. Na začetku sva si zastavila hipoteze, na podlagi katerih sva prioritizirala pomembnejše in se kasneje osredotočila na manj pomembne. Začela sva z iskanjem primernega ohišja, nato sva načrtovala in naročila tiskano vezje, nazadnje pa sva se lotila programiranja. Opisano je delovanje programov in način komunikacije med različnima mikrokrmilnikoma. Opisani so uporabljeni mikrokrmilniki in ostali elementi. Opredeljeni so tudi večji problemi, s katerimi sva se soočala, ter kako sva te probleme rešila. Na koncu so potrjene zastavljene hipoteze, predstavljene so ugotovitve in podani tudi programi, ki sva jih napisala.

KLJUČNE BESEDE

- Raspberry Pi 3 B+
- ATmega 1284p
- Wi-Fi

1. UVOD

1.1. Predstavitev raziskovalnega problema

Za ta projekt sva se odločila iz zabave in zaradi pridobivanja znanja. Ugotovila sva, da večino izdelkov na daljinsko vodenje uporablja radijske frekvence ali povezavo Bluetooth. Pri obojnem je glavni problem razdalja, zato naju je zanimalo, ali bi s pomočjo povezave Wi-Fi lahko upravljala robota kjerkoli v hiši brez motenj. Pri radijskih frekvencah in povezavi Bluetooth, stene predstavljajo velik izziv in zelo zmanjšajo domet naprave. Želela sva ugotoviti, ali bodo stene na najinega robota tudi tako močno vplivale. Poleg tega naju je zanimalo tudi, kako bi delovalo vodenje preko povezave Wi-Fi za razliko od standardnih načinov vodenja, kot so radijske frekvence in s povezavo Bluetooth.

1.2. Hipoteze

- Vodenje robota je možno preko povezave Wi-Fi
- GUI (Graphical User Interface) aplikacija na Windows omogoča ogled videa v živo
- Vodenje je mogoče s pomočjo igralne palice
- Robot zazna nevarnost padca in se avtomatsko postavi na varno mesto
- Cena elementov znaša manj kot 200 evrov

1.3. Opis raziskovalnih metod

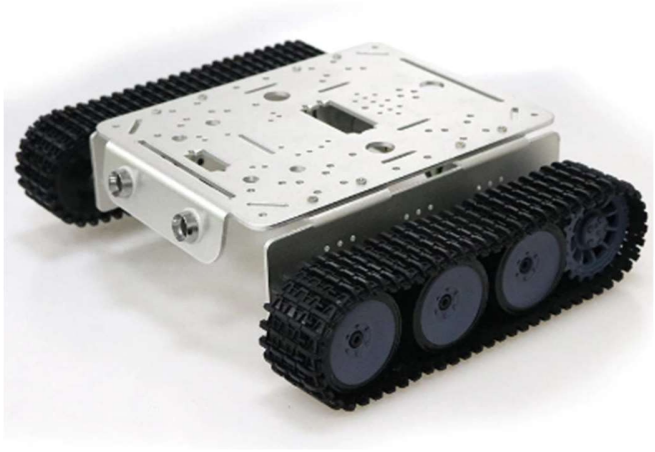
Za vodenje preko povezave Wi-Fi sva se odločila, ko sva razmišljala, na kakšen način naj vodiva robota. Glede na najino predhodno znanje programiranja sva se odločila, da bova kos projektu in ga brez večjih komplikacij izpeljala do konca. Ko sva naletela na težavo, sva se najprej osredotočila na delovanje robota in poskusila odpraviti težavo z opazovanjem in manjšimi spremembami v programu. V primeru, da rešitve nisva našla, sva se osredotočila na internet. S pomočjo komentarjev na objavah, ki so jih napisali ljudje s podobnimi problemi iz različnih projektov, sva lahko našla rešitve ali pa vsaj namige, kako rešiti najine probleme.

2. NAČRTOVANJE

Ideja je nastala s predstavitvijo različnih načinov vodenja. Vodenje preko Wi-Fi povezave se nama je zdelo zanimivo in sva se odločila za preprostega robota, ki bi ga lahko s pomočjo kamere in igralne palice vodila po celem prostoru brez omejitev signala Bluetooth ali klasičnih komunikacij RC. Delo sva razdelila na mehanski in programski del. Osnovno ogrodje sva kupila na spletu in nanj namestila vse potrebne komponente. Ko sva bila zadovoljna s postavitvijo komponent sva se lotila programov. Programi so nama zavzeli večino časa in ko je bil robot zmožen krmiljenja in oddajanja video signala, sva se lotila ohišja.

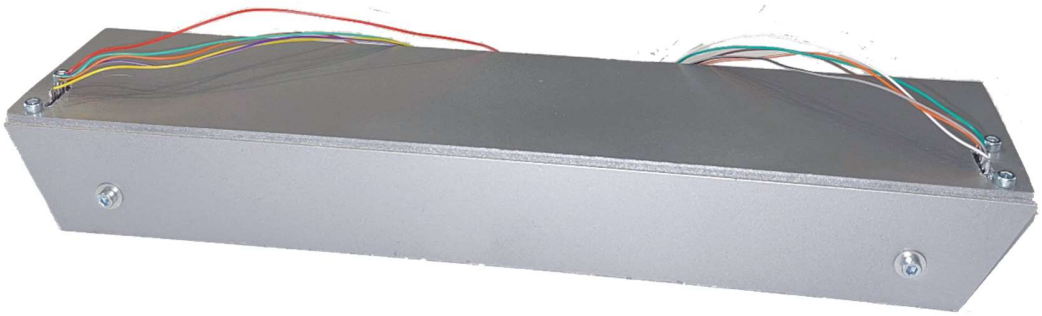
3. OGRODJE

Za ogrodje sva porabila 60 evrov. Poleg ogrodja sva dobila še gosenice, kolesa za gosenice in dva enosmerna DC motorja, kot je razvidno s slike. Ogrodje je dimenzij približno 165 x 205 x 70 mm.



Slika 1 Kupljeno ogrodje

K ogrodju sva dodala par blatnikov, katerih naloga je, da so senzorji čim bolj pred kolesi in čim bolj na zunanji strani. Blatnike sva naredila iz plastike komcel, ki je preprosta za obdelovati in hkrati dovolj trdna. Da sva senzorje lahko namestila na blatnika, sva zvrtala luknje z vrtalnikom in s svedrom, za odprtino za kable pa sva uporabila gravirni stroj. Za lepši učinek, sva jo pobarvala s sprejem srebrne barve.



Slika 2 Blatnik s senzorji

4. NAPAJANJE

Napajalni sistem je sestavljen iz štirih baterij Samsung INR1650-29E tipa Li-ion, dveh stikalnih regulatorjev, ki temeljita na tranzistorjih INRF700 in BMS-a. Baterije so vezane v zaporedni vezavi, z izhodno napetostjo okrog 16.8 V, odvisno od napolnjenosti baterij. Vsaka baterija ima kapacitete 2850 mAh in zdržijo okoli 4 ure. BMS poskrbi za polnjenje in preprečuje, da se baterije ne izpraznijo preveč. Baterije se polnijo preko tokovno zaščitnega regulatorja z napetostjo 20 V in tokovno zaščito 1 A. Regulatorja znižata izhodno napetost motorjev iz približno 16.8 V na 12 V, ter 5 V za napajanje krmilnih sistemov in senzorike.

5. KOMUNIKACIJA IN UPRAVLJANJE

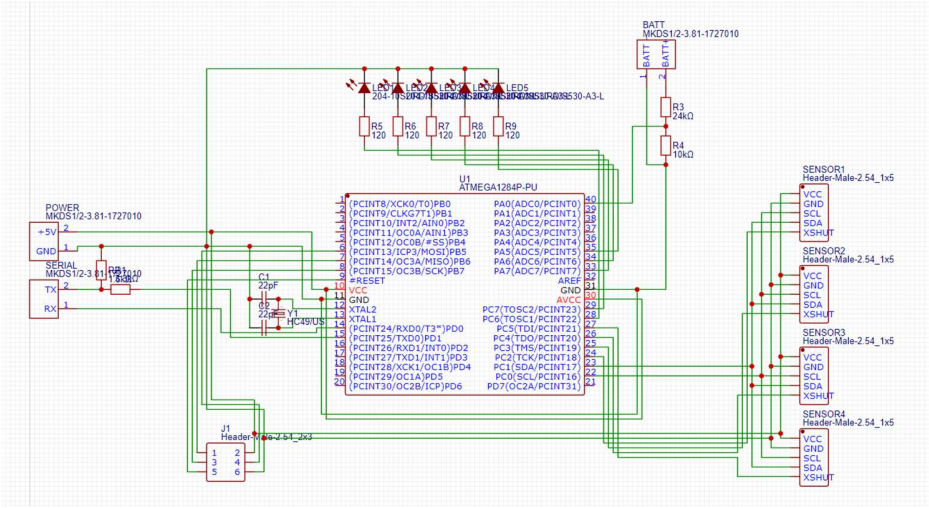
Za komunikacijo med robotom in računalnikom sva se odločila uporabiti povezavo Wi-Fi, kot način krmiljenja robota pa sva uporabljala igralno palico “Joystick Logitech Extreme 3D PRO”. Komunikacija poteka preko dveh portov. Preko enega porta pošija računalnik podatke o položaju kontrolerja v obliki “xxxxx//xxxxx” oziroma “x os//y os”. Raspberry Pi te podatke preračuna in ustrezno nastavi smer in moč motorjev. Na Raspberry Pi je preko serijske komunikacije priključen tudi mikrokrmilnik ATmega 1284p, ki vodi in bere stanje senzorjev, ter hkrati bere tudi nivo baterije. Podatke pošlje na Raspberry Pi v obliki “||xxx/xxx/xxx/xxx/xxx/||” oziroma “||senzor1/senzor2/senzor3/senzor4/baterija||”. Raspberry Pi sprejete podatke uporabi za preprečitev padca, hkrati jih tudi pošlje naprej na računalnik za prikaz stanja baterije.

Raspberry Pi preko polnega mostiča L298N nadzira 2 DC motorja. Največji skupni tok na polnem mostiču je 4 A. Da se ta tok ne preseže, pa poskrbi tokovno omejen regulator, ki je del napajalnega sistema.

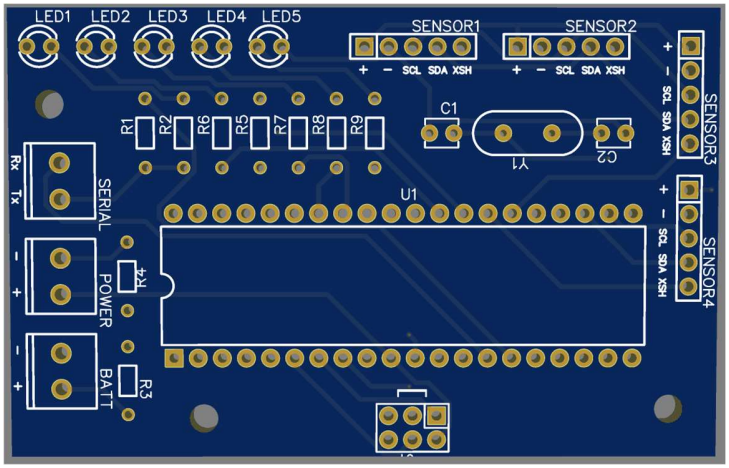
6. SENZORIKA

Na robota sva namestila štiri senzorje za preprečitev padca. Senzorji komunicirajo preko I2C z mikrokontrolerjem ATmega 1284p, za katerega sva v programu EasyEDA narisala vezje. Senzorji in mikrokontroler se napajajo s 5 V enosmerne napetosti. Dodala sva še meritev napetosti baterije z delilnikom napetosti. ATmega 1284p podatke sestavi in jih preko komunikacije UART pošlje na mikroračunalnik Raspberry Pi. Ta podatke uporabi in jih posreduje računalniku za prikaz na programu.

Na PCB sva dodala tudi 5 LED luči, za lažje sledenje programu, še posebej med programiranjem. Ko je bil PCB narisan, sva ga naročila na spletni strani JLCpcb za približno 17 evrov.



Slika 3 Shema vezave senzorjev in napolnjenosti baterije z ATmega1284p



Slika 4 Tiskano vezje vezave senzorjev in napolnjenosti baterije z ATmega1284p

7. PROGRAM

Celotni izdelek ima poudarek na programiranju, saj je mehansko zelo preprost. Celoten program je sestavljen iz treh programov, v katerih so uporabljeni programski jeziki C# in C++ v programskih okoljih .NET Framework, Arduino IDE in Raspbian (Linux za ARM). Program na Raspberry Pi, je zadolžen za upravljanje z motorji in komunikacijo z igralno palico na računalniku z operacijskim sistemom Windows, mikrokontroler ATmega1284p, pa je nadziral senzorje proti padcu in nivo napolnjenosti baterije.

7.1. Programa na robotu

Za vodenje robota uporabljava mikroračunalnik Raspberry Pi 3B+ in mikrokrmilnik ATmega1284p. Mikroračunalnik poskrbi za komunikacijo z igralno palico in upravljanje z motorji L298N. Mikrokontroler poskrbi za zaznavo padca z laserskimi senzorji in meri nivo napolnjenosti baterije. Podatke pošlje na mikroračunalnik preko serijske komunikacije UART. Za zaznavo padca sva se odločila za 4 laserske senzorje oddaljenosti VL53L0. Padec se zazna, ko se oddaljenost senzorjev od tal spremeni na 100 milimetrov. Ko senzorji zaznajo padec, program požene motorje v nasprotno smer nevarnosti. Komunikacija do programa na računalniku poteka preko dveh portov po TCP protokolu. Za video prenos sva uporabila UV4L program za Linux.

7.2. Program na računalniku

Za celoten program na računalniku sva izbrala programsko okolje .NET Framework v jeziki C# na operacijskem sistemu Windows. Za boljši izgled aplikacije sva uporabila knjižnici NuGet “MahApps.Metro” in “MahApps.Metro.IconsPacks”. Za prikazovanje videoposnetka v programskem oknu GUI sva uporabila “Ozeki SDK”. Za način krmiljenja uporabljava igralno palico “Joystick Logitech Extreme 3D PRO”. Robot ima tudi funkcijo tempomata, ki se nahaja v programskem oknu zgoraj levo. Hitrost tempomata nastavimo v programu. Tempomat lahko izklopimo preko programa ali na tipki na igralni palici. Za informacijo, kateri senzor proti padcu se je sprožil, so v programu štiri ikone. V vsaki ikoni sta dve črki. Prva črka pove ali je senzor spredaj (“F” – front) ali zadaj (“R” – rear), druga črka pa nam pove ali je senzor na desni (“R” – right) ali na levi (“L” – left) strani. Zgoraj desno v programskem oknu sta dve ikoni. Leva pove, ali je računalnik povezan z robotom. Če ni, je v ikoni med dvema računalnikoma prikazan “x”, v nasprotnem primeru, je med njima črta.

Slika 5 Programsko okno GUI

8. ELEMENTI IN STROŠKI

- 4x VL53L0 Laserski senzorji oddaljenosti
- 4X Samsung IRN18650-29E
- Raspberry Pi 3B+
- ATmega 1284p-PU
- polni mostič L298N
- 2x DT25-370
- BMS
- PiCamera v1.3
- Ohišje robota
- Kabli
- Stikalo
- Priključek za polnjenje
- Vijaki
- Matice

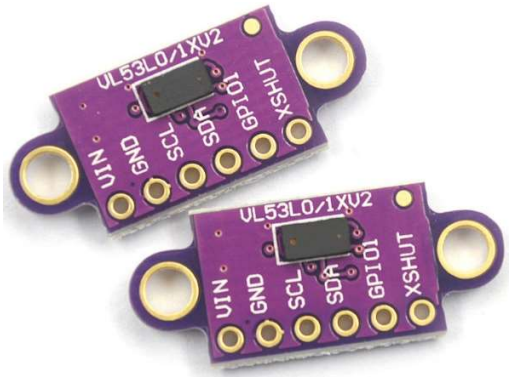
Skupni strošek elementov: 182 evrov.

Laserske senzorje oddaljenosti VL53L0 sva postavila na vse 4 robove robota, za zaznavo padca. Skupna cena senzorjev je bila 25 evrov. Za napajanje robota sva uporabila 4 baterije Samsung IRN18650-29E, v konfiguraciji 4S. Za baterije sva porabila 20 evrov. Dva motorja DT25-370 poganjata gosenice, na katerih se vozi robot. Motorja sva dobila skupaj z ogrodjem, ki naju je stalo 60 evrov, voziva pa ju preko polnega mostiča L298N, s ceno 2 evra. Za vodenje robota poskrbi mikroračunalnik Raspberry Pi 3B+, za katerega sva plačala 40 evrov in mikrokrmilnik ATmega1284p-PU, ki je stal 5 evrov. Za video prenos sva uporabila Raspberry Pi Camera v1.3, s ceno 15 evrov. Za ostale elemente sva porabila približno 15 evrov. Skupna cena elementov za robota je približno 182 evrov, ki potrdi najino hipotezo, da bo cena elementov pod 200 evrov.

9. Opis elementov

9.1. Senzorji oddaljenosti VL53L0

Za laserske senzorje VL53L0sva se odločila zaradi majhne velikosti, natančnosti, velikega območja merjenja in zmožnosti merjenja oddaljenosti pod velikim kotom. Cena enega senzorja ja okoli 6 evrov. Z mikrokrmilnikom komunicira z I2C, zato lahko priklopimo več naprav z manj vodniki, napajajo se s 5 V enosmerne napetosti.



Slika 6 Senzor VL53L0

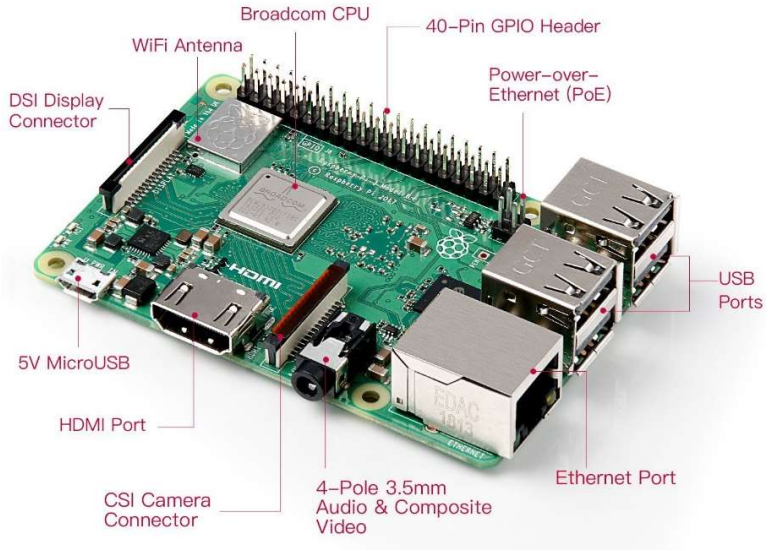
9.2. Baterije Samsung IRN18650-29E

Baterije so Samsung IRN18650-29E, tipa Li-ion in imajo kapaciteto 2850 mAh. Ko so napolnjene, imajo napetost 4.2 V, prazne so pri napetosti 2.5 V. Vezala sva jih zaporedno, saj s tem doseževa dovolj visoko napetost za motorje in približno 4 ure vožnje.

Slika 7 Baterija Samsung IRN18650-29E

9.3. Raspberry Pi 3B+

Raspberry Pi 3B+ je računalnik z 1.2GHz, 4 jedrnim ARM Cortex-A53 procesorjem, in VideoCore IV grafičnim procesorjem, 1 GB RAM-a, ima pa tudi vgrajen Wi-Fi in Bluetooth. Pri projektu sva uporabila njegov 40-pinski I/O konektor, in vgrajeni 15-pinski MIPI serijski konektor za kamero (MIPI CSI-2). Vgrajen ima tudi 100MB ethernt, 4 USB 2.0 priključke, AUDIO vhod in izhod, konektor za ekran, ter DSI (Display Serial Interface). Operacijski sistem je shranjen na mikro SD kartici.

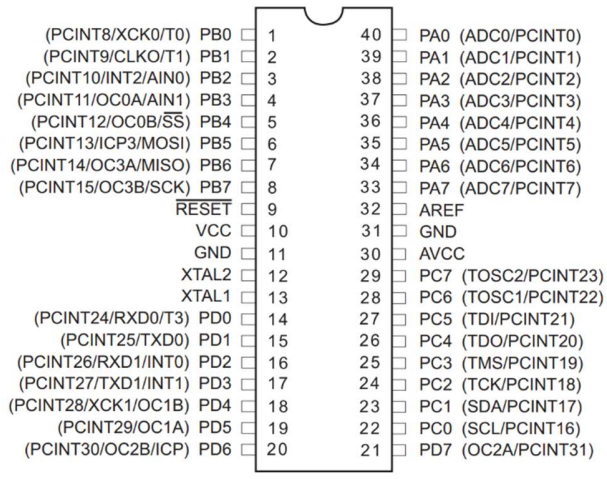


Slika 8 Raspberry Pi 3B+

9.4. ATmega 1284p-PU

ATmega1284P je 8-bitni mikrokrmilnik z nizko porabo energije, ki temelji na AVR arhitekturi. Je 40-pinska naprava s 128 KB Flash, 16 KB SRAM in 4 KB EEPROM-a. Ima dva 8-bitna števca/timerja, dva 16-bitna števca/timerja, 6 pinov PWM in osem 10-bitnih analognih vhodov. Z izvrševanjem navodila v enem urinem ciklu naprave dosežejo prepustnost procesorja, ki se približa milijonu navodil na sekundo (MIPS) na megaherc, kar omogoča optimizacijo porabe energije glede na hitrost obdelave. Deluje na obratovalni napetosti 1.8 - 5.5 V, ter do 20 MHz. Uporabljava ga pri frekvenci 16 MHz. Izbrala sva ga, ker ATmega328P nima dovolj SRAM-a (2 kB) za vse senzorje.

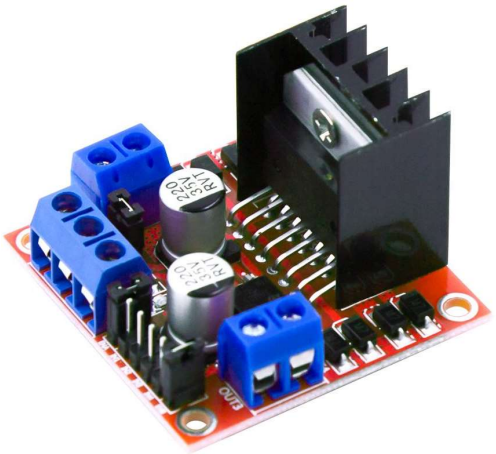
Slika 9 Mikrokrmilnik ATmega 1284p-PU



Slika 10 Pinout mikrokrmilnika ATmega 1284p-PU

9.5. Polni mostič L298N

L298 je visokonapetostni in visokotokovni dvojni polni mostič, zasnovan za sprejemanje standardnih logičnih nivojev TTL napetosti in kot pogon induktivne obremenitve, kot so releji, elektromagneti, enosmerni in koračni motorji. Na voljo sta dva vhoda, ki neodvisno od vhodnih signalov omogočita ali onemogočita napravo. Dodatni napajalni vhod je uporabljen, da lahko logični vhodi delujejo pod nižjo napetostjo (5 V).



Slika 11 Dvojni polni mostič L298N

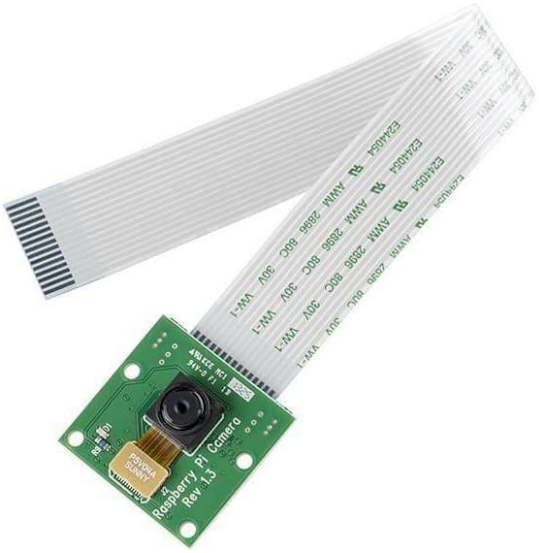
9.6. DC motor DT25-370

DT25-370 je enosmerni motor z operativno napetostjo 12 V in največ 300 obratov/min. Tok brez obremenitve je 0,04 A. Premer osi motorja je 4 milimetra. Njegove dimenzije so 25 x 66,3 x 25 mm.

Slika 12 DC motor DT25-370

9.7. Pi Camera v1.3

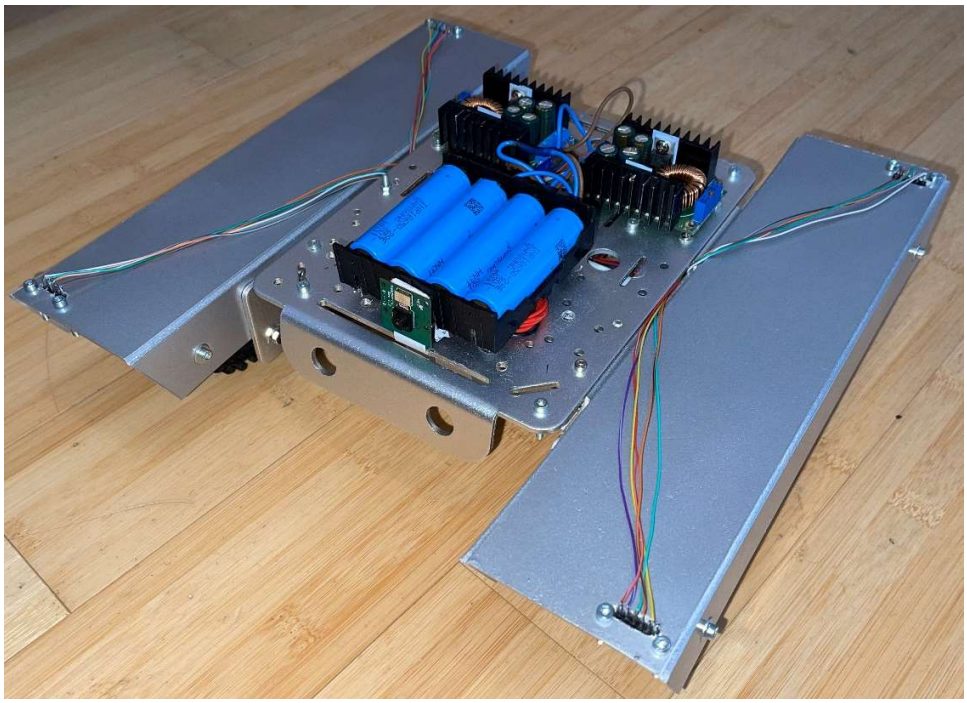
Pi Camera v1.3 uporablja vmesnik MIPI CSI-2, s katerim se poveže na Raspberry Pi. Ima 5 MP kamero, ki lahko zajame slike v različnih formatih in video v RAW h.264 formatu. Je velikosti 25 x 24 x 9 mm in težo 3 grame.



Slika 13 Kamera Pi Camera v1.3

10. KONČNI IZDELEK

Najin cilj, vodenje robota po vsej hiši, je bil dosežen. Robot oddaja video v živo, ki ga lahko spremljava na zaslonu računalnika, ima funkcijo zaznavanja nevarnosti padca. Na zaslonu računalnika nam sporoči, kateri senzor je zaznal nevarnost padca. Naslednja funkcija je preprečevanje padca, s katerim se robot skuša izogniti padcu tako, da se zapelje v nasprotno smer od zaznane nevarnosti padca. Poleg tega sva uspela implementirati tudi tempomat, katerega hitrost lahko nastaviva in izklopiva na računalniku.



Slika 14 Končni izdelek

11. RAZPRAVA

Z raziskovalno nalogo sva potrdila vse hipoteze.

- Vodenje robota je možno preko povezave Wi-Fi
- GUI (Graphical User Interface) aplikacija na Windows omogoča ogled videa v živo
- Vodenje je mogoče s pomočjo igralne palice
- Robot zazna nevarnost padca in se avtomatsko postavi na varno mesto
- Cena elementov znaša manj kot 200 evrov

Glavni cilj raziskovalne naloge je bil vožnja robota s pomočjo povezave Wi-Fi po vseh prostorih hiše, brez motenj v signalu. To sva dosegla s komunikacijo med računalnikom, mikrokrmilnikom Arduino in mikroračunalnikom Raspberry Pi.

Uspela sva doseči preprosto programsko okno, v katerem lahko spremljava, ali je robot povezan z računalnikom, stanje baterije na robotu in oddajanje videoposnetka v živo s čim manjšim zamikom.

Kot glavni način krmiljenja robota sva si izbrala igralno palico “Joystick Logitech Extreme 3D PRO”. Ima veliko gumbov in drugih funkcij, ki nama omogočijo dodatne funkcionalnosti ob morebitnem nadaljevanju raziskovanja ali kakšnih nadgradnjah.

Želela sva doseči, da robot zazna nevarnost padca in se poskusi sam rešiti pred njim. Zaznavanje padca sva dosegla z uporabo laserskih senzorjev, poskus preprečitve padca pa sva implementirala tako, da se robot za kratek čas zapelje v nasprotno smer, od koder je zaznal nevarnost padca. Z rezultati sva zadovoljna, saj se tako robot sam postavi v varen položaj.

Kot zadnjo in najmanj pomembno hipotezo sva si zadala, da cena elementov ne bi presegla 200 evrov. To hipotezo lahko potrdiva, saj sva za celotno raziskovanje porabila približno 182 evrov.

Ob programiranju programov sva naletela na kar nekaj manjših problemov in na dva večja. Prvi večji problem je nastal pri zaostanku videoposnetka v živo, ta je znašal približno 10 sekund. Najprej sva zmanjšala resolucijo videoposnetka in s tem dosegla zamik približno 1,5 sekunde. Ker je ta zamik bil še vedno prevelik, sva začela iskati rešitve na spletu. Odločila sva se za uporabo Ozeki SDK (Software Development Kit), ki nama je omogočil približno pol sekunde zaostanka videoposnetka. Na drugi večji problem pa sva naletela pri senzoriki. Senzorji so bili postavljeni preveč proti sredini in so rob zaznali šele, ko se robot sam ni več moral rešiti. Ta problem sva rešila z blatniki, ki so nama omogočili postavitev senzorjev pred gosenice in tudi veliko bolj navzven.

12. ZAKLJUČEK

Ob tej raziskovalni nalogi sva ugotovila, da se način vodenja robota s povezavo Wi-Fi splača le, če je cela hiša pokrita z dobro povezavo Wi-Fi in da se robot uporablja zgolj v notranjosti te hiše. Domet robota temelji na moči signala povezave Wi-Fi. Pomembna je tudi hitrost interneta, ki jo omogoča povezava Wi-Fi. Hitrejši kot je, manj bo zamika pri oddajanju videa v živo. Izdelek vsekakor ni popoln in bi se lahko izboljšalo veliko programskih in mehanskih značilnosti. V nadaljevanju bi lahko raziskala, kako še zmanjšati zamik in hkrati zvišati kakovost videoposnetka. Poleg tega bi robota lahko opremila s kakšno osvetlitvijo in drugimi dodatki.

13. VIRI IN LITERATURA

https://os.mbed.com/questions/2348/How-to-input-global-variable-to-a-thread/, dostop 2.2.2021

https://www.codeproject.com/Questions/652822/how-to-lock-global-variable-in-multi-thread, dostop 2.2.2021

https://makolyte.com/how-to-update-ui-from-another-thread/, dostop 2.2.2021

https://docs.microsoft.com/en-us/dotnet/desktop/wpf/advanced/how-to-make-a-uielement-transparent-or-semi-transparent?view=netframeworkdesktop-4.8, dostop 2.2.2021

https://www.akumulator.si/image.ashx?id=T9%2004%20INR18650%2029E&size=400&fill=1, dostop 5. 2. 2021

https://img.alicdn.com/i1/738263294/O1CN01oYCJjc1aChFtLWjUi_!!738263294.jpg, dostop 5. 2. 2021

https://cdn.shopify.com/s/files/1/0261/8074/7357/products/8_3164767c-bb6d-4078-aac0-adb967253735_2000x.jpg?v=1569274940, dostop 5. 2. 2021

https://cdn.shopify.com/s/files/1/0069/6513/3376/products/AA034_2.jpg?v=1547714206, dostop 5. 2. 2021

https://media.rs-online.com/t_large/F7193932-01.jpg, dostop 5. 2. 2021

https://www.tdgulf.com/wp-content/uploads/2016/06/11868-00a.jpg, dostop 5. 2. 2021

https://www.picclickimg.com/d/l400/pict/324449884482_/Speed-Reduction-Gear-Motor-with-Encoder-Code-Disk.jpg, dostop 5. 2. 2021

https://stackoverflow.com/questions/19013087/how-to-detect-multiple-keys-down-onkeydown-event-in-wpf/19013440, dostop 6. 2. 2021

https://docs.microsoft.com/en-us/dotnet/standard/base-types/how-to-pad-a-number-with-leading-zeros, dostop 6. 2. 2021

https://www.youtube.com/watch?v=fGtNh0ofpCI&ab_channel=TheEngineeringProjects, dostop 6. 2. 2021

https://www.codeproject.com/Articles/5264831/How-to-Send-Inputs-using-Csharp, dostop 6. 2. 2021

https://stackoverflow.com/questions/9301030/how-to-get-and-bind-the-value-of-slider-in-c-sharp-wpf, dostop 7. 2. 2021

https://blog.miguelgrinberg.com/post/how-to-build-and-run-mjpg-streamer-on-the-raspberry-pi, dostop 11. 2. 2021

https://desertbot.io/blog/how-to-stream-the-picamera,

https://stackoverflow.com/questions/43826568/how-can-i-stream-video-from-a-raspberry-pi-to-a-webpage, dostop 11. 2. 2021

https://github.com/ZeBobo5/Vlc.DotNet/issues/261, dostop 11. 2. 2021

https://raspberrypi.stackexchange.com/questions/85143/uv4l-usb-webcam-how-to-change-resolution-fps-etc, dostop 11. 2. 2021

https://raspberrypi.stackexchange.com/questions/50335/how-to-configure-uv4l,

https://www.codeproject.com/Articles/202464/How-to-use-a-WebCam-in-C-with-the-NET-Framework-4, dostop 11. 2. 2021

https://stackoverflow.com/questions/412995/how-to-stream-webcam-in-wpf, dostop 11. 2. 2021

https://www.camera-sdk.com/p_6737-how-to-display-mjpeg-camera-stream-in-c-.html, dostop 11. 2. 2021

14. ZAHVALA

Posebne zahvale za odlično mentorstvo bi rada posvetila mentorju gospodu Gregorju Kramerju, ki nama je pomagal z nasveti in pomoči ter organizaciji pri oblikovanju in vsebini seminarske naloge. Zahvalila bi se tudi profesorju Davorju Zupancu za razne predloge in profesorju Janku Holobarju za nasvete in pomoč pri seminarski nalogi. Seveda pa se morava zahvaliti in pohvaliti tudi delo najine lektorice profesorice Tanje Jelenko, ki je hitro, zelo kakovostno in v zelo majhnem časovnem oknu lektorirala najino seminarsko nalogo.

15. PRILOGE

PRILOGA 1: PROGRAM NA RAČUNALNIKU

```
public MainWindow()
{
    InitializeComponent();
    thrd = new Thread(video);
    osi = new Os();
    osi.x = 32768;
    osi.y = 32768;
    Thread th = new Thread(sock);
    th.Start();
    Thread thr = new Thread(sockrecieve);
    thr.Start();
    Thread th1 = new Thread(joystick);
    th1.Start();
}
```

Slika 15 Inicializacija okna in večnitnosti

```
private void video()
{
    _connector = new MediaConnector();
    _bitmapSourceProvider = new DrawingImageProvider();
    videoViewer.SetImageProvider(_bitmapSourceProvider);
    _fpsRescaler = new FrameRateHandler(20);

    var config = new OzConf_P_MJPEGClient("http://192.168.1.96:8080/stream/video.mjpeg");
    _mjpegConnection = new MJPEGConnection(config);
    _mjpegConnection.CameraStateChanged += _mjpegConnection_CameraStateChanged;

    _mjpegConnection.Connect();
    _connector.Connect(_mjpegConnection.VideoChannel, _bitmapSourceProvider);
    videoViewer.Start();
}
2 references
private void _mjpegConnection_CameraStateChanged(object sender, CameraStateEventArgs e)
{
    GuiThread(() => HandleState(e.State));
}
1 reference
void GuiThread(Action action)
{
    Dispatcher.BeginInvoke(action);
}
1 reference
public void HandleState(CameraState state)
{
    if (state == CameraState.Connected)
    {
        videoViewer.Start();
    }

    if (state == CameraState.Disconnected)
    {
        _mjpegConnection.CameraStateChanged -= _mjpegConnection_CameraStateChanged;
        _connector.Disconnect(_mjpegConnection.VideoChannel, _bitmapSourceProvider);
        _mjpegConnection.Dispose();
        _mjpegConnection = null;
    }
}
```

Slika 16 Prikaz videa

```
private int RangeConversion(double inputValue, double OldMin = 0, double OldMax = 32767, double NewMax = 10000, double NewMin = 32767)
{
    return Convert.ToInt32((((inputValue - OldMin) * (NewMax - NewMin)) / (OldMax - OldMin)) + NewMin);
}
```

Slika 17 Pretvorba dometa števil

```
    var directInput = new DirectInput();
    var joystickGuid = Guid.Empty;
    foreach (var deviceInstance in directInput.GetDevices(SharpDX.DirectInput.DeviceType.Gamepad,
                DeviceEnumerationFlags.AllDevices))
        joystickGuid = deviceInstance.InstanceGuid;
    if (joystickGuid == Guid.Empty)
        foreach (var deviceInstance in directInput.GetDevices(SharpDX.DirectInput.DeviceType.Joystick,
                DeviceEnumerationFlags.AllDevices))
            joystickGuid = deviceInstance.InstanceGuid;
    if (joystickGuid == Guid.Empty)
    {
        Environment.Exit(1);
    }

var joystick = new Joystick(directInput, joystickGuid);
var allEffects = joystick.GetEffects();
joystick.Properties.BufferSize = 128;
joystick.Acquire();
```

Slika 18 Inicializacija kontrolne palice

```
joystick.Poll();
var datas = joystick.GetBufferedData();

foreach (var state in datas)
{
    if (state.Offset.ToString() == "X")
    {
        osi.x = state.Value;
    }

    if (state.Offset.ToString() == "Y")
    {
        osi.y = state.Value;
    }
}
```

Slika 19 Branje položaja kontrolne palice

```
void sockrecieve()
{
    string server = "192.168.1.96";
    string message = recieve;

    try
    {
        Int32 port = 8082;
        TcpClient client = new TcpClient(server, port);
        Byte[] data = System.Text.Encoding.ASCII.GetBytes(message);
        NetworkStream stream = client.GetStream();
        data = new Byte[256];
        String responseData = String.Empty;
        while (true)
        {
            Int32 bytes = stream.Read(data, 0, data.Length);
            sensvsi = System.Text.Encoding.ASCII.GetString(data, 0, bytes);
            snesicons();
            sensvsi = "";
        }
    }
    catch (Exception e)
    {
        Thread th = new Thread(sockrecieve);
        th.Start();
    }
}
```

Slika 20 Povezovanje in branje TCP porta

```
private void Connected()
{
    this.Dispatcher.Invoke(() =>
    {
        connIcon.Kind = PackIconMaterialKind.LanConnect;
    });
    thrd.Start();
}

1 reference
private void Disconnected()
{
    this.Dispatcher.Invoke(() =>
    {
        connIcon.Kind = PackIconMaterialKind.LanDisconnect;
    });
}
```

Slika 21 Prikaz ikone za povezanost z robotom

```
void sock()
{
    string server = "192.168.1.96";
    string message = nadzor;

    try
    {
        Int32 port = 8081;
        TcpClient client = new TcpClient(server, port);
        //Byte[] data = System.Text.Encoding.ASCII.GetBytes(message);
        NetworkStream stream = client.GetStream();
        Connected();//tukaj se konekta

        while (true)
        {
            Byte[] data = System.Text.Encoding.ASCII.GetBytes(nadzor);
            stream.Write(data, 0, data.Length);
            Thread.Sleep(10);
        }
    }

    catch (Exception e)
    {
        Disconnected();
        //tukaj ni skonectan
        Thread th = new Thread(sock);
        th.Start();
    }
}
```

Slika 22 Povezovanje in pošiljanje komand na TCP port

PRILOGA 2: PROGRAM NA MIKROKONTROLERJU ATMEGA 1284P

```
#include "Adafruit_VL53L0X.h"

#define LOX1_ADDRESS 0x30
#define LOX2_ADDRESS 0x31
#define LOX3_ADDRESS 0x32
#define LOX4_ADDRESS 0x33

#define SHT_LOX1 18
#define SHT_LOX2 19
#define SHT_LOX3 20
#define SHT_LOX4 21

Adafruit_VL53L0X lox1 = Adafruit_VL53L0X();
Adafruit_VL53L0X lox2 = Adafruit_VL53L0X();
Adafruit_VL53L0X lox3 = Adafruit_VL53L0X();
Adafruit_VL53L0X lox4 = Adafruit_VL53L0X();

VL53L0X_RangingMeasurementData_t measure1;
VL53L0X_RangingMeasurementData_t measure2;
VL53L0X_RangingMeasurementData_t measure3;
VL53L0X_RangingMeasurementData_t measure4;

#define L1 22
#define L2 23
#define L3 31
#define L4 30
#define L5 29

bool S1F = true;
bool S2F = true;
bool S3F = true;
bool S4F = true;

typedef struct
{
  int sens1;
  int sens2;
  int sens3;
  int sens4;
} SensorValues;
SensorValues values;
```

Slika 23 Uporabljenje knjižnice in globalne spremenjivke

```
void setup() {
  pinMode(L1, OUTPUT);
  pinMode(L2, OUTPUT);
  pinMode(L3, OUTPUT);
  pinMode(L4, OUTPUT);
  pinMode(L5, OUTPUT);

  DebugInfo(1);
  pinMode(A0, INPUT);
  pinMode(SHT_LOX1, OUTPUT);
  pinMode(SHT_LOX2, OUTPUT);
  pinMode(SHT_LOX3, OUTPUT);
  pinMode(SHT_LOX4, OUTPUT);

  DebugInfo(2);
  Serial.begin(115200);
  while (! Serial) { delay(1); }

  DebugInfo(3);
  digitalWrite(SHT_LOX1, LOW);
  digitalWrite(SHT_LOX2, LOW);
  digitalWrite(SHT_LOX3, LOW);
  digitalWrite(SHT_LOX4, LOW);

  setAddresses();
}

void loop() {
  measure();
  DebugInfo(13);
  DebugInfo(14);
  String response = "||" + (String)values.sens1 + "//" + (String)values.sens2 + "//" + (String)values.sens3 + "//" + (String)values.sens4 + "//" + (String)map(analogRead(A0),551,1011,0,100) + "||";
  Serial.println(response);

}
```

Slika 24 Inicializacija senzorjev

```
void measure()
{
  DebugInfo(8);
  if(S1F)
    lox1.rangingTest(&measure1, false);
  if(S2F)
    lox2.rangingTest(&measure2, false);
  if(S3F)
    lox3.rangingTest(&measure3, false);
  if(S3F)
    lox4.rangingTest(&measure4, false);

  DebugInfo(9);
  if(S1F)
  {
  if(measure1.RangeStatus != 4)
    values.sens1 = measure1.RangeMilliMeter;
  else
    values.sens1 = 0;
  }
    DebugInfo(10);
  if(S2F)
  {
  if(measure2.RangeStatus != 4)
    values.sens2 = measure2.RangeMilliMeter;
  else
    values.sens2 = 0;
  }
    DebugInfo(11);
  if(S3F)
  {
  if(measure3.RangeStatus != 4)
    values.sens3 = measure3.RangeMilliMeter;
  else
    values.sens3 = 0;
  }
  DebugInfo(12);

  if(S4F)
  {
  if(measure4.RangeStatus != 4)
    values.sens4 = measure4.RangeMilliMeter;
  else
    values.sens4 = 0;
  }

}
```

Slika 25 Merjenje oddaljenosti od tal

```
void setAddresses()
{
  digitalWrite(SHT_LOX1, LOW);
  digitalWrite(SHT_LOX2, LOW);
  delay(10);
  digitalWrite(SHT_LOX1, HIGH);
  digitalWrite(SHT_LOX2, HIGH);
  delay(100);

  DebugInfo(4);
  digitalWrite(SHT_LOX1, HIGH);
  digitalWrite(SHT_LOX2, LOW);
  digitalWrite(SHT_LOX3, LOW);
  digitalWrite(SHT_LOX4, LOW);
  if(!lox1.begin(LOX1_ADDRESS)) {
    S1F = false;
    delay(1000);
  }
  delay(10);

  DebugInfo(5);
  digitalWrite(SHT_LOX2, HIGH);
  digitalWrite(SHT_LOX3, LOW);
  digitalWrite(SHT_LOX4, LOW);
  if(!lox2.begin(LOX2_ADDRESS)) {
    S2F = false;
    delay(1000);
  }
  delay(10);


  DebugInfo(6);
  digitalWrite(SHT_LOX3, HIGH);
  digitalWrite(SHT_LOX4, LOW);
  if(!lox3.begin(LOX3_ADDRESS)) {
    S3F = false;
    delay(1000);
  }
  delay(10);

  DebugInfo(7);
  digitalWrite(SHT_LOX4, HIGH);
  if(!lox4.begin(LOX4_ADDRESS)) {
    S4F = false;
    delay(1000);
  }
  delay(10);
}
```

Slika 26 Nastavitev naslovov senzorjev

```
void DebugInfo(int ledValuesInt)
{
  int ledValues[5];

  if(!S1F || !S2F || !S3F || !S4F)
    ledValuesInt +=16;
  if(!S1F && !S2F && !S3F && !S4F)
    ledValuesInt += 30;

  for (int i = 0; i < 5; i++)
  {
    ledValues[i] = bitRead(ledValuesInt, i);
  }


  digitalWrite(L1, ledValues[0]);
  digitalWrite(L2, ledValues[1]);
  digitalWrite(L3, ledValues[2]);
  digitalWrite(L4, ledValues[3]);
  digitalWrite(L5, ledValues[4]);
}
```

Slika 27 Spreminjanje LED luči na pcb-ju

PRILOGA 3: PROGRAM NA MIKRORAČUNALNIKU RASPBERRY PI

```
#include <unistd.h>
#include <vector>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <sstream>
#include <iostream>
#include <netinet/in.h>
#include <string>
#include <errno.h>
#include <wiringPi.h>
#include <thread>
#include <boost/algorithm/string.hpp>
#include <wiringSerial.h>

#pragma region Defines
#define CPORT 8081
#define SPORT 8082

#define LeftPWM 23 // GPIO 13, PHYS 33
#define LeftFwd 3 // GPIO 17, PHYS 11
#define LeftBck 2 // GPIO 27, PHYS 13 // set back to 2 when testing done // 7 for testing
#define RightPWM 26 // GPIO 12, PHYS 32
#define RightFwd 13 // GPIO 9, PHYS 21 // set back to 13 when testing done // 2 for testing
#define RightBck 6 // GPIO 25, PHYS 22
#pragma endregion

using namespace std;

#pragma region Structs
struct SensorValues
{
  int Sens1;
  int Sens2;
  int Sens3;
  int Sens4;
  int Batt;
};

struct MotValues{
    int left, right;
};

struct SplitRaw{
    int speed, turn;
};
#pragma endregion
```

Slika 28 Knjižnice in structi

```
#pragma region GlobalVars
unsigned int prevTime;

int s_server_fd, s_new_socket, s_valread;
struct sockaddr_in s_address;
int s_opt = 1;
int s_addrlen = sizeof(s_address);
char s_buffer[1024] = {0};

int server_fd, new_socket, valread;
struct sockaddr_in address;
int opt = 1;
int addrlen = sizeof(address);
char buffer[1024] = {0};
SensorValues cs;

string ArduinoSend = "";

char recvMotVal[12];
#pragma endregion
```

Slika 29 Globalne spremenjivke

```
void sockInitialize()
{
    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port 8080
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                                                &opt, sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons( CPORT );

    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr *)&address,
                                    sizeof(address))<0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
}

void sockListenAccept()
{
    cout << "Socket is in listen mode" << endl;
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                                (socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    cout << "Socket accepted" << endl;
}

void sockRead()
{
    char buffer[12];
    int recvCount=0;
    recvCount=recv(new_socket, recvMotVal, sizeof(recvMotVal), 0 );

}
```

Slika 30 Ustvarjanje in branje TCP porta

```
SplitRaw SplitValues(string input)
{
    std::vector<std::string> result;
    boost::split(result, input, boost::is_any_of("//"), boost::token_compress_on);
    //cout << result[0] << endl;
    stringstream turn(result[0]);
    stringstream speed(result[1]);
    int turni = 0;
    turn >> turni;
    int speedi = 0;
    speed >> speedi;
    SplitRaw a;
    a.turn = turni;
    a.speed = speedi;
    return a;
}

int RangeConversion(int inputValue, int OldMin = -00000, int OldMax = 65535, int NewMax = -1024, int NewMin = 1024)
{
    return (((inputValue - OldMin) * (NewMax - NewMin)) / (OldMax - OldMin)) + NewMin;
}
```

Slika 31 Branje pridobljenih podatkov igralne palice

```
MotValues CalcMotors(SplitRaw input)
{
    int speed = RangeConversion(input.speed);
    int turn = RangeConversion(input.turn);
    int hturn = turn / 2;
    MotValues a;

    //cout << hturn << endl;
    int lspeed = speed;
    int rspeed = speed;

    lspeed -= hturn;
    rspeed += hturn;

    for(int i = 0; i < 2; i++)
    {
        if(lspeed > 1024)
        {
            int cnd = lspeed - 1024;
            lspeed -= cnd;
            rspeed -= cnd;
        }
        else if (lspeed < -1024)
        {
            int cnd = lspeed + 1024;
            lspeed -= cnd;
            rspeed -= cnd;
        }

        if(rspeed > 1024)
        {
            int cnd = rspeed - 1024;
            lspeed -= cnd;
            rspeed -= cnd;
        }
        else if (rspeed < -1024)
        {
            int cnd = rspeed + 1024;
            lspeed -= cnd;
            rspeed -= cnd;
        }
    }

    a.left = lspeed;
    a.right = rspeed;
    //cout << "LEFT: " << a.left << "     RIGHT: " << a.right << endl;

    return a;
}
```

Slika 32 Preračunavanje hitrosti motorjev

```
void KickBack()
{
    digitalWrite(LeftFwd, 0);
    digitalWrite(LeftBck, 1);
    digitalWrite(RightFwd, 0);
    digitalWrite(RightBck, 1);

    pwmWrite(LeftPWM, 1024);
    pwmWrite(RightPWM, 1024);
}

void KickForward()
{
    digitalWrite(LeftFwd, 1);
    digitalWrite(LeftBck, 0);
    digitalWrite(RightFwd, 1);
    digitalWrite(RightBck, 0);

    pwmWrite(LeftPWM, 1024);
    pwmWrite(RightPWM, 1024);
}
```

Slika 33 Vožnja v nasprotno smer padca

```
void SetMotors(string input)
{

    SplitRaw sv = SplitValues(input);
    MotValues calcValues = CalcMotors(sv);
    if(calcValues.left > 0){
        digitalWrite(LeftFwd, 1);
        digitalWrite(LeftBck, 0);
    }
    else if(calcValues.left < 0){
        digitalWrite(LeftFwd, 0);
        digitalWrite(LeftBck, 1);
    }
    if(calcValues.left == 0){
        digitalWrite(LeftFwd, 0);
        digitalWrite(LeftBck, 0);
    }

    if(calcValues.right > 0){
        digitalWrite(RightFwd, 1);
        digitalWrite(RightBck, 0);
    }
    else if(calcValues.right < 0){
        digitalWrite(RightFwd, 0);
        digitalWrite(RightBck, 1);
    }
    if(calcValues.right == 0){
        digitalWrite(RightFwd, 0);
        digitalWrite(RightBck, 0);
    }

    if(abs(calcValues.left) < 100)
    {
        calcValues.left = 0;
    }

    if(abs(calcValues.right) < 100)
    {
        calcValues.right = 0;
    }
    //cout << cs.Sens1 <<endl;
    if(cs.Sens1 > 100 || cs.Sens2 > 100)
    {
        KickBack();
    }
    else if(cs.Sens3 > 100 || cs.Sens4 > 100)
    {
        KickForward();
    }
    else{
        pwmWrite(LeftPWM, abs(calcValues.left));
        pwmWrite(RightPWM, abs(calcValues.right));
    }
```

Slika 34 Nastavljanje motorjev

```
void GPIOinit()
{
    wiringPiSetup();

    pinMode(RightFwd, OUTPUT);
    pinMode(RightBck, OUTPUT);
    pinMode(LeftFwd, OUTPUT);
    pinMode(LeftBck, OUTPUT);
    pinMode(LeftPWM, PWM_OUTPUT);
    pinMode(RightPWM, PWM_OUTPUT);
    delay(1);
    cout << "GPIO INIT SUCESS" << endl;
}
```

Slika 35 Inicializacija GPIO pinov

```
SensorValues DecodeValues(string recv)
{
  SensorValues s;
  vector<string> result;
  boost::split(result, recv, boost::is_any_of("//"), boost::token_compress_on);
  //cout << result[0] << endl;
  stringstream s1(result[0]);
  stringstream s2(result[1]);
  stringstream s3(result[2]);
  stringstream s4(result[3]);
  stringstream bt(result[4]);

  int s1i = 0;
  int s2i = 0;
  int s3i = 0;
  int s4i = 0;
  int bti = 0;

  s1 >> s1i;
  s2 >> s2i;
  s3 >> s3i;
  s4 >> s4i;
  bt >> bti;


  s.Sens1 = s1i;
  s.Sens2 = s2i;
  s.Sens3 = s3i;
  s.Sens4 = s4i;
  s.Batt = bti;

  //cout << s1i << endl;

  return s;
}
```

Slika 36 Dekodiranje vrednosti senzorjev

```
void Sensors()
{
  //thread thread_obj(SensorThread);
  SensorSockInit();
  SensorSockListenAccept();
  int fd;
  if((fd=serialOpen("/dev/ttyS0",115200))<0){

    fprintf(stderr,"Unable to open serial device: %s\n",strerror(errno));
    while(true){}
  }
  string recv;
  bool StringBeg = false;
  int Keys = 0;
  int CharLoc = 0;
  while(true)
  {
    char rch = serialGetchar(fd);
    if(rch=='|')
    {
      Keys ++;

      if(Keys == 2 && !StringBeg)
      {
       Keys = 0;
       StringBeg = true;
      }
      else if(Keys == 2 && StringBeg)
      {
       Keys = 0;
       CharLoc=0;
       StringBeg = false;
       //cout << recv << endl;
       cs = DecodeValues(recv);
       SensorThread(recv);
       recv = "";
      }
    }
    else
    {

      recv += rch;
      CharLoc++;
    }
  }
}
```

Slika 37 Povezovanje z ATmega 1284p in branje senzorjev

```
int main(int argc, char const *argv[])
{
    GPIOinit();
    pwmWrite(LeftPWM, 0);
    pwmWrite(RightPWM, 0);
    thread thread_obj(Sensors);

    sockInitialize();
    sockListenAccept();


    while(1){
        sockRead();
        //cout << recvMotVal << endl;
        if(recvMotVal == "0")
        {
            sockListenAccept();
        }
        else
        {
            SetMotors(recvMotVal);
        }
    }
    return 0;
}
```

Slika 38 Inicializacija programa

PRILOGA 4: IZJAVA

Mentor Kramer Gregor v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom Robot na Wi-Fi vodenje, katere avtorja sta Sebastjan Lah in Jan Gutenberger:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 6.5.2021

žig šole

Podpis mentorja

Podpis odgovorne osebe

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.