SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. xxxx

**Lokalizacija površinskih vozila korištenjem sustava kamera**

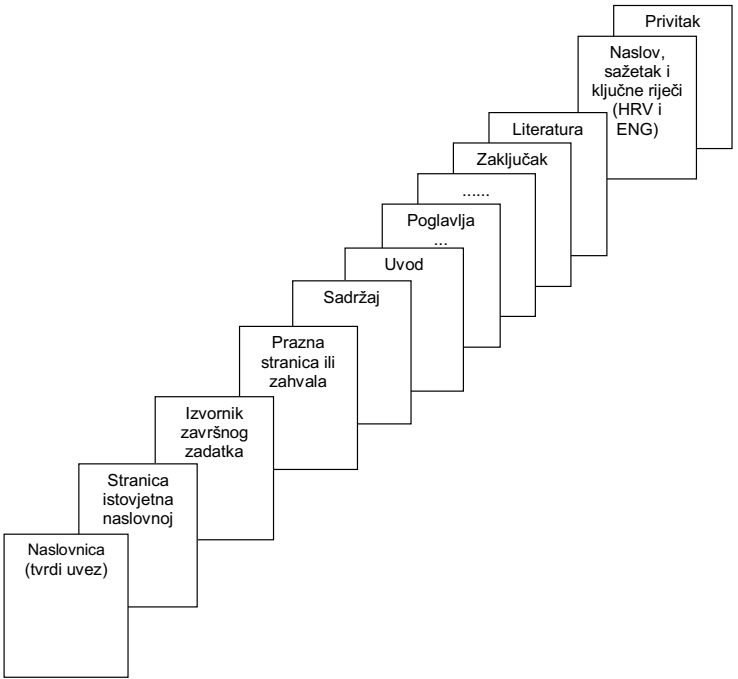
Leonardo Roy Sabolić

Zagreb, lipanj 2025.

Završni rad predaje se u elektroničkoj verziji u pdf obliku.

Student u bazu učitava rad od četvrte stranice (Prazna stranica ili zahvala). Prve tri stranice (naslovnica, stranica istovjetna naslovnici, završni zadatak) bit će automatski povučene iz sustava.

Prilikom oblikovanja rada držite se redoslijeda. Redoslijed stranica okvirno je prikazan na slici.



Student predaje samo pdf verziju rada preko FERweb-a.Ova stranica treba biti:

ILI prazna stranica

ILI stranica sa zahvalom (po želji studenta; zahvala nije obvezna).

Sadržaj

[Uvod 1](#_Toc201166682)

[1. Korišteni alati 2](#_Toc201166683)

[1.1. OpenCv 2](#_Toc201166684)

[1.2. Boost 2](#_Toc201166685)

[1.3. ROS2 2](#_Toc201166686)

[1.4. Arduino 3](#_Toc201166687)

[2. Generalni opis sustava i izmjene 4](#_Toc201166688)

[2.1. Koncept sustava 4](#_Toc201166689)

[2.1.1. Node 4](#_Toc201166690)

[2.1.2. Konektor 5](#_Toc201166691)

[2.2. Izmjene sustava 6](#_Toc201166692)

[2.2.1. Izmijenjeni Node-ovi 6](#_Toc201166693)

[3. Sustav za kašnjenje 7](#_Toc201166694)

[3.1. Princip rada 7](#_Toc201166695)

[3.2. Implementacija 8](#_Toc201166696)

[4. Sustav za praćenje 11](#_Toc201166697)

[4.1. Izbor načina za praćenje 11](#_Toc201166698)

[4.2. Princip rada 12](#_Toc201166699)

[4.3. Implementacija 12](#_Toc201166700)

[4.4. ROS Node 15](#_Toc201166701)

[5. Testiranje sustava 17](#_Toc201166702)

[5.1. Testiranje sustava za kašnjenje 17](#_Toc201166703)

[5.2. Testiranje sustava za praćenje 19](#_Toc201166704)

[5.3. Testiranje ostatka sustava 23](#_Toc201166705)

[6. Naslov prvog poglavlja 24](#_Toc201166706)

[6.1. Prvo potpoglavlje 24](#_Toc201166707)

[6.2. Stilovi za tekst, naslove i podnaslove 24](#_Toc201166708)

[6.3. Stilovi za nabrajanje 24](#_Toc201166709)

[6.3.1. Stilovi za nabrajanje s točkama i crticama 25](#_Toc201166710)

[6.4. Slike 26](#_Toc201166711)

[6.5. Tablice 28](#_Toc201166712)

[6.6. Matematički izrazi i formule 29](#_Toc201166713)

[6.7. Programski kôd 29](#_Toc201166714)

[Zaključak 31](#_Toc201166715)

[Literatura 32](#_Toc201166716)

[Sažetak 33](#_Toc201166717)

[Summary 34](#_Toc201166718)

[Skraćenice 35](#_Toc201166719)

[Privitak 36](#_Toc201166720)

# Uvod

U robotici se često suočavamo s problemom praćenja pozicije nekog objekta, te postoje razne metode određivanja pozicije to mogu bit unutrašnji ili vanjski senzori od kojih se koriste razne metode.

U ovom radu fokusirat ćemo se na vanjski sustav koji koristi kamere. Tu se već susrećemo s dosta problematike, ovisno o sustavu kamera implementacija našeg praćenja može znatno varirati, udaljenost, vidno polje, svjetlosni uvjeti, brzina i rezolucija kamere, i drugi parametri jako utječu na izbor načina praćenja i kompleksnost sustava za praćenje. Glavni problem kojim se ovaj rad bavi je problematika kod izrade takvih sustava na brz i efikasan način, te mogućnost brze iteracije i testiranja raznih takvih sustava. Ovaj rad se nastavlja na rad[ Weber, A. *Podvodna lokalizacija u eksperimentalnom bazenu korištenjem kamera* [1]], koji se bavi dizajnom samog sučelja programa i funkcionalnostima kalibracije kamere na par načina. Glavni cilj je napraviti sustav koji će omogućiti lagano brzo i precizno postavljanje proizvoljnog sustava kamera, te povezivanje tog sustava s drugim robotskim sustavima, time se omogućuje i rad na nepoznatim lokacijama s nepoznatim sustavom kamera ili lako testiranje i kalibriranje pri izmjeni sustava kamera.

Glavnina rada se bavi nadogradnjama i izmjenama sustava iz navedenog rada[1], povećava se fleksibilnost, dodaju se funkcionalnosti praćenja i sinkronizacije među kamerama, te se testira funkcionalnost cijelog sustava. Sustav bi u konačnici bio korišten za praćenje površinskih vozila ali bi bio primjenjiv i na bilo koji robotski sustav koji koristi neki statičan sustav kamera za navigaciju.

# Korišteni alati

Rad je rađen dijelom na Windows 11 OS-u u C++ programskom jeziku te je za sučelje korištena ImGui biblioteka, više o sučelju je opisano u navedenom radu[1], neke od glavnih biblioteka korištenih su OpenCv i Boost. Drugi dio je rađen na Ubuntu Linux OS-u na ROS2 platformi. Od hardvera je korišten Arduino Uno, logithec c270 kamere i sustav kamera u LABUST-u.

## OpenCv

Biblioteka koja se koristi za obradu slike i to postiže kroz obradu slike u matričnoj reprezentaciji te je time pogodna i za manipulacije matricama. U radu se koristi za dobivanje slike, filtraciju željenih elemenata na slikama, manipulaciju i dobivanje transformacijskih matrica, te ima ugrađenu podršku za praćenje ArUco markera koji se koriste u nekim dijelovima rada. Zbog svega toga ova biblioteka je idealni izbor jer pokriva puno potrebnih funkcionalnost za izvršavanje ovog rada.

## Boost

Vrlo velika biblioteka s raznim alatima koji bi mogli biti potrebni kod developmenta raznih vrsta aplikacija, neke od najčešće korištenih značajki su pametni pokazivači (*eng. smart pointers*), asinkroni I/O u sklopu Boost.Asio, više drevnost (*eng. multy threading*) u sklopu Boost.Thred, Boost.Filesystem i drugi manje poznati dijelovi biblioteke.

Od ovih funkcionalnosti za ovaj rad je najbitniji Boost.Asio, te se također koriste Boost-ovi pametni pokazivači koji su integrirani u C++11 nadalje.

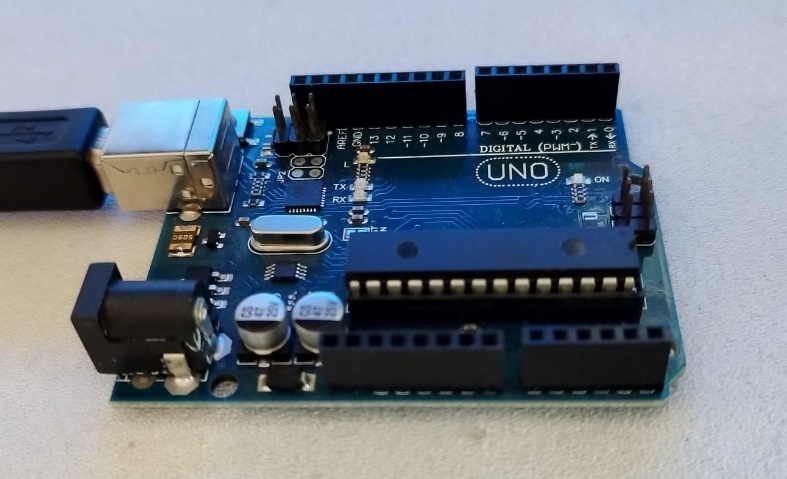
Boost.Asio je biblioteka koja se bavi komunikacijom preko mreže ili bilokakvih ulazno izlaznih jedinica. U radu se koristi pri serijskoj komunikaciji preko USB porta i pri UDP komunikaciji preko mreže.

## ROS2

ROS je platforma koja se koristi u Robotici za laganu komunikaciju raznih sustava potrebnih za upravljanje, percepciju i donošenje odluka u nekom Robotskom sustavu. Cijeli ROS sustav je napravljen tako da se preko pretplatnika (*eng. subscriber*) i izdavač (*eng. publisher*) koji koriste teme (*eng. topic*) razmijenu podataka između ROS node-ova. U sklopu rada se koristi ROS2 Humble, te se preko njega aplikacija koja se razvija povezuje s ostatkom robotskog sustava, ovaj sustav je pogodan jer se lako preko mreže mogu povezati razni sustavi i izmjenjivati podatke. Ovaj sustav podržava rad u Python ili C++ programskim jezicima te je korišten Python radi njegove jednostavnosti.

## Arduino

Arduino je jednostavan programibilan mikro kontroler koji se koristi uglavnom za prikupljanje senzorskih podataka i upravljanje aktuatorima i drugim sklopovima. U radu se koristi Arduino Uno 8bit-ni mikro kontroler izabran radi niske zahtjevnost zadatka koju izvršava, dodatna prednost je niska cijena. Arduino pločica koja je bila korištena je prikazana na slici (Slika 1.1).



Slika 1.1 Arduino Uno

# Generalni opis sustava i izmjene

Sustav je implementiran kao sučelje koje korisniku omogućuje slaganje i spajanje node-ova koji su zasebni procesi za izvršavanje određenih zadatka, te se nadovezivanjem vise takvih node-ova odbija neki željeni efekt. Cijeli sustav je napravljen da radi na sto fleksibilnijem sustavu kamera sto uključuje proizvoljan broj, karakteristike i položaje kamera.

## Koncept sustava

Sustav je koncipiran kao Node/Connection sistem, svaki Node ima konektore koji su ili ulazni ili izlazni te svaki konektor prima ili šalje određeni tip poruke te se dodavanjem i spajanjem Node-ova prenose obrađeni podaci medu njima i postiže se željeni efekt.

### Node

Svaki Node ima istih par funkcija u kodu pomoću kojih se definiraju konektori i njihovi tipovi, u funkcijama getInMessageTypes() i getOutMessageTypes() se definiraju ulazni i izlazni konektori. Funkcije drawNodeWork() i drawNodeParams() se izvršavaju svako iscrtavanje programa i služe za prikaz elemenata Node-a jedan je namjenjen za stalne elemente a jedan za promjenjive koji ovise o nekakvom procesiranju. Funkcija recieve() se pokreće svaki put kada je primljena poruka te se iz primljene poruke ona može po nekom svojstvu odrediti kako dalje procesirati poruku.

Opisi Node-ova:

Većina Node-ova je opisano u radu[1].

* NodeSourceStream
  + Služi za spajanje na kamere i definiranje parametara, također služi za intrinzičnu kalibraciju kamere, više o tom je napisano u navedenom radu[1].
* Node-color-treshold
  + Služi za filtraciju određene boje iz slike, detaljniji opis je u navedenom radu[1].
* Node-inflate
  + Služi za popravljanje rubova nakon filtracije kako bi se dobila stabilnija filtrirana slika, detaljnije opisan u navedenom radu[1].
* Node-background-subtraction
  + Služi za micanje pozadine kako bi se izdvojio objekt, detaljnije opisan u navedenom radu[1].
* Blob-creator
  + Služi za kreiranje blobova (određeni tip podatka) iz filtrirane slike, detaljnije opisano u navedenom radu[1].
* Blob-grouper
  + Služi za automatsku ekstrinzičnu kalibraciju pomoću detektiranog filtriranog bloba, detaljnije objašnjeno u navedenom radu[1].
* Node-manual-extrinsics
  + Služi za ekstrinzičnu kalibraciju putem izbora točaka koje izabire korisnik, i za definiranje nekog proizvoljnog koordinatnog sustava i izračun transformacijske matrice, detaljnije objašnjeno u navedenom radu[1].
* Node-delay-measurment
  + Služi za mjerenja kasnjenja jedne ili vise kamera pomocu spajanja na vanjski mikrokontroler, ovaj Node se detaljnije opisuje u nastavku ovog rada.
* Node-aruco-tracking
  + Sluzi za pracenje ArUco markera u odabranom kordinatnom sustavu, radi na proizvoljnom broju kamera i šalje podatke o markerima na željenu ip addresu : port, ovaj Node se detaljnije opisuje u nastavku ovog rada.

### Konektor

Konektori su vizualno prikazani na Node-ovima s lijeve i desne strane, s lijeve su ulazni konektori a s desne izlazni, pored svakog konektora piše tip poruke koji prima ili šalje, konektori se vizualno povezuju plavim linijama.

U implementaciji se komunikacija postiže dijeljenim pokazivaćima (*eng. shared pointer*) te se koristi funkcija send() ili sendAll() koje šalju poruku na neki konektor s željenim id-em ili na sve konektore.

Postoji problematika u ovakvoj implementaciji a to je da je dosta teško imati više izlaznih konektora istog tipa jer nema neki način za odrediti koji je koji konektor, moguće je dobiti dodijeljeni id svakog konektora ali on nam ne pomaže pri razlikovanju konektora pri inicijalizaciji node-a, ova situacija se trenutno nigdje ne koristi ali bi bilo korisno imati mogućnost dodijeliti ime konektoru pri inicijalizaciji radi lakšeg dodjeljivanja željenih poruka. Primjer bi bio recimo node za filtriranje slike, umjesto da koristimo 3 takva jedan bi bio dovoljan za filtraciju svih ulaznih kamera te je to trenutno nemoguće, ali bi u budućoj implementaciji to bilo dobro popraviti.

## Izmjene sustava

Sustav je u svojem početnom stanju bio dosta nefleksibilan s nekim stvarima kao što su rezolucija, način kalibracije i imao je neke manje probleme. Zbog toga je za svaki node dodana mogućnost korištenja bilo koje rezolucije, te korištenje bilo koje kalibracijske ploče (crno bijela šahovnica).

### Izmijenjeni Node-ovi

Najviše izmijenjen node je NodeSourceStreme gdje je dodana mogućnost upis veličine kalibracijske ploče, veličine kockica na ploči, dodana je mogućnost izbora rezolucije, te je dodan ulazni parametar tipa int(cijeli broj) koj služi za primanje podatka o kašnjenju kamere.

U Node-manual-extrinsics je dodana rezolucija te je to inkorporirano u kalkulacije extrinzičnih matrica, postoji jedan problem koji nastaje zbog logike izbora parova točaka a to je da to sučelje računa s tim da je rezolucija prozora za izbor točaka jednaka rezoluciji kamera, te da je udaljenost među njima 8px. Problem s tim je što nije lagano birati točke na većim rezolucijama bar na monitoru slične rezolucije.

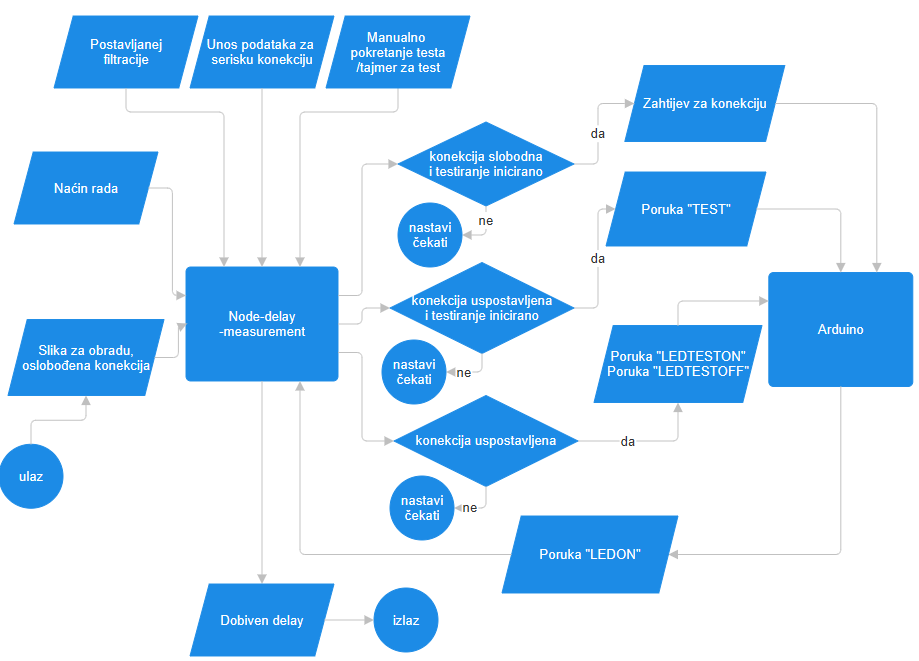
# Sustav za kašnjenje

Sustav za kašnjenje je napravljen kao Node-delay-measurmant, ulazni konektor prima podatak sliku i cijeli broj koji služi za sinkronizaciju testova među više instanci node-a, na izlazu vrača cijeli broj kašnjenje u milisekundama (ms) i filtriranu sliku.

Moguće je testiranje kašnjenja bez sinkronizacije ali se može javiti problem kada jedna od instanci node-a dobi konekciju na Arduino ali se usred inicijalizacije dogodi zahtjev konekcije od drugog te se onda Arduino zamrzne, zato je dodana sinkronizacija a time se dodatno izbjegava zagušenje serijskog kanal s zahtjevima za spajanje. Ovaj problem se ne javlja kod korištenja zasebnog Arduina za svaku instancu node-a.

## Princip rada

Slika 3.1. Dijagram rada



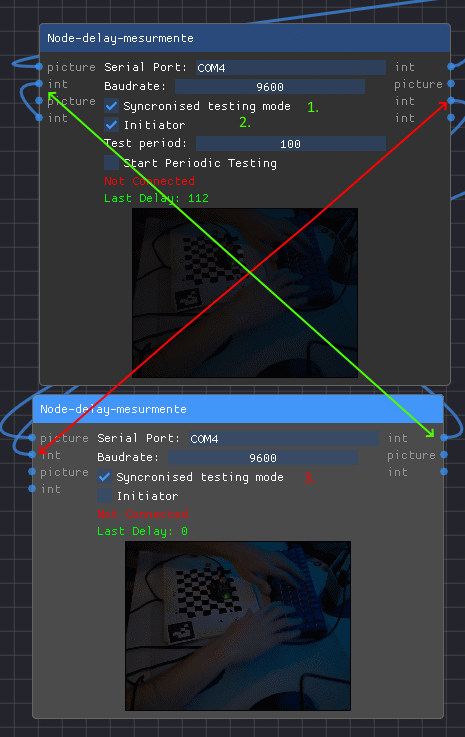
Node prima sliku od kamere, te se pokretanjem testiranja šalje poruka (TEST) vanjskom mikro kontroleru koji povratno pali lampicu i vraća poruku (LEDON), te onda Node prati vrijeme od dobivanja poruke od detekcije željene značajke na slici, te se prosljeđuje dobivena razlika u vremenu na izlazni konektor. Kod više testova s istim mikro kontrolerom se onda dobivena vrijednost prosljeđuje slijedećem Node-u te on onda pokreće testiranje i ponavljaju se već navedeni koraci.

Poruke LEDTESTON i LEDTESTOFF su poruke koje se šalju kod postavljanja filtera za detekciju izabrane značajke te se odvije radi o paljenju i gašenju led diode, ideja je da se filter postavi s upaljenom led diodom te se onda unaprijed vidi kvaliteta filtera i detekcije.

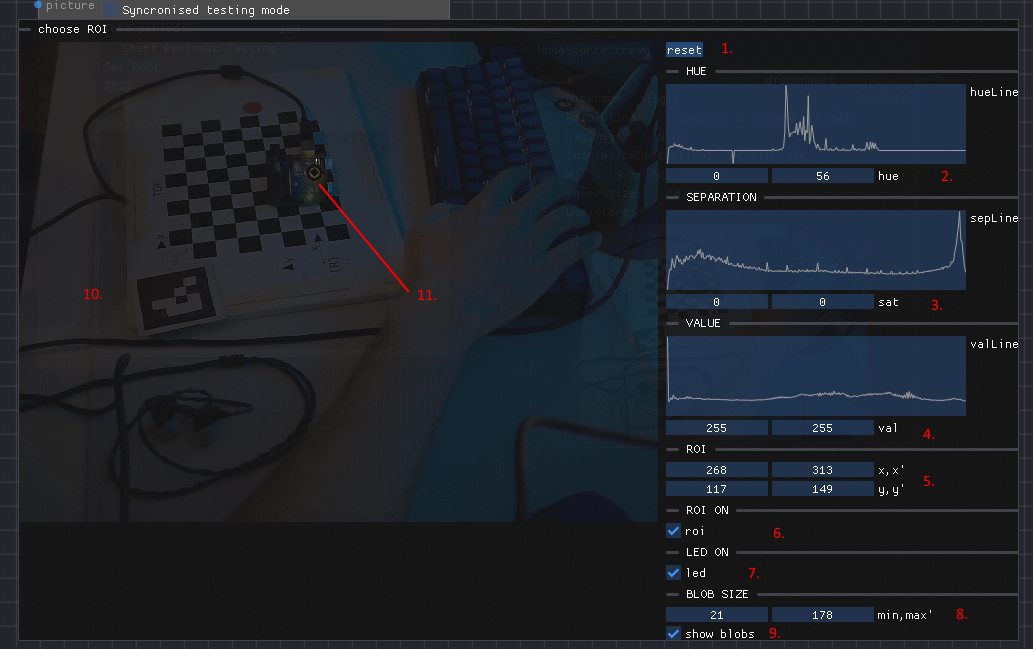
## Implementacija

Slika 3.2 Primjer Node-delay-measurment

Node prima sliku i int na ulazu (1. ulazni konektori) te se onda slika filtrira (12. prikaz filtrirane slike). Node prima parametre za serijsku komunikaciju (3. i 4.) s Arduinom, te se onda pri pokretanju testa manualno, na tajmeru ili putem drugog Node-a pokuša uspostaviti konekcija s Arduinom (10. indikator konekcije) pri uspješnoj konekciji se tada pokreće testiranje, te se rezultat testa prikazuje (11.) i šalje na izlaz (2.). Na izlazu se također šalje i filtrirana slika. Gumbi (8. i 9.) služe za manualno spajanje i testiranje, a za automatsko se postavlja periodičko testiranje (6. i 7.) s periodom u sekundama ili ako se koristi sinkronizirano testiranje onda se Node postavlja u taj način rada (5.) . Pritiskom na filtrirani prikaz (12.) se otvara sučelje za postavljanje filtera. Svi brojevi se odnose na sliku (Slika 3.2)

Pri korištenju sinkroniziranog načina rad je bitno povezati sve Nodove koji koriste isti Arduino u loop kao što je prikazano s zelenim i crvenim strelicama, još je bitno imati inicijatora (2.) koji će pokretati testiranje periodički, bitno je isto sve node-ove staviti u sinkronizirani način rada (1. i 3.). Ova implementacija nije baš naj ljepša vizualno ali izbjegava nepotrebno gušenje serijskog porta. Bolja solucija bi bila imati jedan node s više ulaza za sliku i više filtera i testova za svaku sliku, te bi tako mogli sve istovremeno testirati, no problem je kod izlaza jer trenutna implementacija nema laki način za određivanje koji izlazni port će se dodijeliti kome. Svi brojevi se odnose na sliku (Slika 3.3)

Slika 3.3 Testiranja u sinkroniziranom načinu rada



Slika 3.4 Sučelje za filtraciju slike

Za filtraciju je korišten hsv filter u kombinaciji s roi filterom. Hsv filter se može u sučelju podesiti na (2., 3. i 4.) ili se pritiskom desnog klik na mišu i povlačenjem pokazivača preko željenih stvari za detekciju u (10.) prikazu slike automatski postavljaju vrijednosti za hsv filter. Roi filter se postavlja na (5. i 6.), postavljaju se maksimalne i minimalne vrijednosti pixela koji ulaze u filtraciju te je to vizualizirano tamnijim kvadratom izvan filtera. Postavljanje detekcije je moguće pomoću dijela (7., 8. i 9.) gdje se postavlja maksimalna i minimalna veličina značajke detekcije to je vizualizirano s dva crna kruga (11.) manji predstavlja minimalnu površinu detekcije a veći maksimalnu. Kod filtracije nije potrebno maknuti sve ostale detekcije već je potrebno postaviti takav filter da je broj detekcija stalan, ili ne naglo varirajući, to ubiti znači da nam ne smije treperiti nijedna od detekcija što će biti očito pri postavljanju blob size-a (8.), sustav mjeri promjenu detekcija te tako izbjegava slučajeve gdje se zbog vanjskih uvjeta promijeni okolina željene značajke. Svi brojevi se odnose na sliku (Slika 3.4)

# Sustav za praćenje

Namijene sustava za praćenje je određivanje pozicije i orijentacije nekog robota, za ovaj rad specifično površinskih vozila na LABUST-ovom sustavu kamera, no sustav je također napravljen da se može koristiti i na terenu s proizvoljnim sustavom kamera. Te je napravljen da bude lagan i brz za postavit.

## Izbor načina za praćenje

Slika 4.1 ArUco marker

Sustav za praćenje koji je bio izabran je ArUco, to je sistem koji koristi kvadratne markere za određivanje pozicije i orijentaciji, jako je sličan drugim takvim sustavima kao AprilTag, ARTag i drugi, no ArUco je izabran radi najmanje minimalne veličine što daje najveće ‘pixele’ te time i najveću udaljenost na kojoj radi da najmanju površinu samog markera, specifično je izabrana veličina **4x4\_50** što znači da je veličina mreže markera 4x4 te da ih je 50 zbog toga je to 16 pixela za odabir markera koji su crno bijeli, te se iz toga dobije 65536 različitih markera, ali uzimajući u obzir rotaciju markera to ispadne i dalje puno više od 50, jedno bitno svojstvo ovih markera je da imaju minimalnu Hamming-ovu distancu (broj različitih pixela između dva markera) 3 a u praksi se vrti oko 10, zato je izabran baš 4x4\_50 a ne neki drugi, jer daju naj robusniju detekciju, također za potrebe LABUST-a nije potrebno više od 50 markera. Još jedna prednost kod ArUco markera je to što su podržani u OpenCv biblioteci koja se već koristi u kodu.

## Princip rada

Slika na kojoj se prikazuje tekst, snimka zaslona, dijagram, Font

Sadržaj generiran uz AI možda nije točan.

Slika 4.2 Dijagram rada Node-aruco-tracking

Node-aruco-tracking prima slike od svih kamera i dobivene relacije između kamera i drugih frame-ova svijeta, dobivenih u ekstrinzičnoj kalibraciji. Zatim je potrebno dobiti konačne transformacije za svaku kameru u željeni sustav, te ako smo uspješno dobili sve transformacije onda nakon detekcije ArUco markera u svakoj kameri i transformacije u željeni sustav se slaže podatak za slanje ROS2 Node-u. ROS2 Node tad formatira podatke i postavlja dobra vremena poruka (*eng. timestamp*) te objavljuje temu (*eng.topic*) za svaki marker.

## Implementacija

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, softver

Sadržaj generiran uz AI možda nije točan.

Slika 4.3 Početno stanje za Node-aruco-tracking

Prvo je potrebno izabrati okvir (*eng.frame*) u kojem želimo detektirati markere, to se radi izborom u izborniku prikazanom na slici (Slika 4.3), potom program traži sve relacije od svake kamere koja je spojena na ulaz do traženog okvira (*eng.frame*) ako za neku od kamera nije nađen onda se za nju ne računaju pozicije markera. Put između dobivenih relacija se dobije BFS algoritmom što je algoritam pretraživanja stabla te tako pokriva sve puteve, te se vrača put koji sadrži sve transformacije koje se potom množe i dobiva se konačna transformacija. Relacije na koje se misli se dobivaju putem Node-manual-extrinsics te je taj proces detaljnije opisano u navedenom radu[1]. Računanje konačne matrice odvija se po formuli (1).

|  |  |
| --- | --- |
|  | (1) |
| Td\_s Predstavlja transformaciju iz okvira izvora (*eng. sorce*) **s** u okvir destinacije (*eng. destination*) **d**. |  |

Slika na kojoj se prikazuje tekst, elektronika, snimka zaslona, multimedija

Sadržaj generiran uz AI možda nije točan.

Slika 4.4 Parametri Node-aruco-tracking

Kako bi dobili ispravno detektirane markere bitno je ispravno postaviti ulazne parametre (2.) je postavljanje okvira svijeta u kojem želimo dobiti koordinate markera, (3.) se koristi samo za vizualizaciju željene kamere, bitan parametar za dobivanje točne pozicije markera je (1.) veličina stranice markera. Nakon podešavanja tih postavki moguće je verificirati točnost praćenja.

Slika na kojoj se prikazuje tekst, snimka zaslona, računalo, softver

Sadržaj generiran uz AI možda nije točan.

Slika 4.5 UDP povezivanje

Za slanje podataka UDP-om je potrebno postaviti valjanu IP adresu i port (4. i 5.) te pritisnuti gumb set (6.), ako je UDP kanal uspješno otvoren gumb će pozeleniti kao na slici (Slika 4.5), a ako je došlo do problem gumb će pocrveniti. Još je potrebno omogućiti slanje postavljanjem opcije sending (6.). Svi brojevi odnose se na sliku (Slika 4.4).

Slika na kojoj se prikazuje tekst, snimka zaslona, računalo, elektronika

Sadržaj generiran uz AI možda nije točan.

Slika 4.6 Prošireno sučelje

Pritiskom na prikaz kamere otvara se uvećani prozor te je on namijenjen za validaciju praćenja, postavljanjem opcije (2.) se omogućuje prikaz koordinata i rotacije markera (6.) nam označava x koordinatu i rotaciju oko x osi (*eng. roll*) isto tako i za druge dvije osi (6. i 7.) dobijemo y, z, rotacija oko y (*eng. pitch*), rotacija oko z (*eng. yaw*). Izborom opcije (1.) dobivamo prikaz koordinatnih osi okvira svijeta (*eng. world frame*) (4.) u kojem se vrši detekcija markera (5.). Svi brojevi odnose se na sliku (Slika 4.6).

Pozicija markera u okviru svijeta se dobiva po formuli (2) a transformacija korištena u formuli (2) je opisana u formuli (1).

|  |  |
| --- | --- |
|  | (2) |
|  |  |

Gdje je Tw\_cn transformacija iz okvira kamere u okvir svijeta. A Tm\_cn je transformacija iz okvira kamere u okvir markera.

Ako je opcija sending i ip uspješno postavljen (Slika 4.5) onda se nakon dobivanja pozicija markera formira json string (način spremanja podataka pogodan za slanje preko mreže) sa:

* Podatci za svaki marker
  + ID markera
  + X pozicija markera
  + Y pozicija markera
  + Z pozicija markera
  + Yaw (rotacija oko z osi) u stupnjevima
  + Roll (rotacija oko x osi) u stupnjevima
  + Pitch (rotacija oko y osi) u stupnjevima
  + Kvaternione (qw, qx, qy, qz)
* Kamera s koje je detektirano
* Vrijeme detekcije
* Procesno kašnjenje
* Kašnjenje kamere

Taj string se potom šalje na postavljenu ip adresu.

## ROS Node

Node prima json string sa ip adrese postavljenje u .launch datoteci, inicijalno je konfiguriran da prima podatke sa adrese 0.0.0.0:12345 što znači da će primati podatke sa bilo koje adrese na portu 12345. Nakon prijama podataka Node obrađuje podatke te se kreira tema (*eng. topic*) za svaki marker. Svaka ta tema sadrži poruku s pozicijom i rotacijom markera (Slika 4.7 i Slika 4.8), te se u vremensku oznaku (eng. timestamp) uračunava kašnjenje, ali se ne koristi vrijeme koje je dobiveno u poruci jer svako računalo malo drugačije prati vrijeme te radi lakše sinkronizacije s ostatkom sustava koristi vrijeme dobiveno u ROS node-u od kojeg se oduzima kašnjenje dobiveno UDP porukom.

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Sadržaj generiran uz AI možda nije točan.

Slika 4.7 Kreirane teme od UDP poruke

Slika na kojoj se prikazuje tekst, Font, snimka zaslona, dizajn

Sadržaj generiran uz AI možda nije točan.

Slika 4.8 Primjer poruke formirane u ROS node-u

# Testiranje sustava

Za svaki test su iznova rađene intrinzična i ekstrinzična kalibracija kako bi se dobila najveća preciznost. Te je kod testiranja bitno imati dobro postavljen sustav, kod testiranja kašnjenja bitno je postaviti sustav kao što je navedeno u poglavlju (3.2), a za testiranje praćenja potrebno je imati dobru ekstrinzičnu kalibraciju. Kako bi dobili što bolju kalibraciju bitno je postaviti kamere na undestorted što je omogućeno nakon intrinzične kalibracije, te onda pokrenuti ekstrinzičnu kalibraciju, te je onda bitno izabrati oko 20 točaka idealno po 3 u istoj ravnini kako bi se dobila što bolja kalibracija, te je to olakšano s funkcionalnosti korištenja kalibracijske ploče za odabir točaka, jedino je bitno provjeriti jesu li parovi točaka dobro generirani pri uzimaju točaka s ploče jer se orijentacija ploče pogađa pa postoji mogućnost da su parovi točaka zrcaljeni. Nakon dobivene kalibracije sustav je spreman za praćenje.

## Testiranje sustava za kašnjenje

Sustav za kašnjenje je testiran s lokalnim kamerama i na jednoj od kamera u LABUST-u te se javio problem gomilanja ulaznih poruka te se zbog toga povećavalo kašnjenje kroz vrijeme radi sporog rada programa na višim rezolucijama, u međuvremenu je sustav popravljen, no LABUST mjerenje nije bilo ponovljeno, inicijalno mjerenje je dalo rezultat od 1982ms te to nije bilo očekivano, očekivano je bilo kašnjenje između 400 i 800ms.

Za testiranje kašnjenja je korišten sustav za kašnjenje i Arduino, te za dobivanje očekivanog kašnjenja je korištena kamera koja se testira, internet stranica sa štopericom i kamera na pametnom telefonu. Očekivano kašnjenje dobiveno je slikanjem ekrana sa štopericom i prikaza kamere koja prikazuje štopericu (Slika 5.1 i Slika 5.2).

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, Robna marka

Sadržaj generiran uz AI možda nije točan.

Slika 5.1 Mjerenja očekivanog kamera 1

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, Robna marka

Sadržaj generiran uz AI možda nije točan.

Slika 5.2 Mjerenje očekivanog kamera 2

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |

Iz mjerenja dobivamo da kamera 1 u prosjeku kasni 61.67ms, a kamera 2 kasni 63ms. Ovo nije idealan test jer se slika na ovom monitoru iscrtava svake 4.2ms ali je za okvirne vrijednost radi testiranja dovoljno dobro. Prosječna greška ovog testa se dobije formulom (3) i njena vrijednost je 1.18ms. Po prirodi sustav za detekciju kašnjenja ima grešku od ±1/brzina\_iscrtavanja\_kamere, kod testiranih kamera je to 1/30, ako uzimamo srednju vrijednost greške kamere i ekrana (±16.67 i ±2.1 ms) po formuli (4), dobivamo da je prosječna greška ±16.8ms po testu, za tri testa greška ispadne 9.7ms po formuli (3).

Mjerenjem kašnjenja u Node-delay-measurment dobiveni su sljedeći rezultati. Mjerenja su rađena prema opisu korištenja navedenog node-a u poglavlju (3.2), za svaku kameru napravljeno je 10 mjerenja prikazanih u tablici (Tablica 5.1).

Tablica 5.1 Kašnjenje dobiveno i odstupanja

|  |  |  |  |
| --- | --- | --- | --- |
| Dobiveno: | | Odstupanje od očekivanog: | |
| Kamera 1 | Kamera 2 | Očekivano K 1  (61.67 ± 9.7) | Očekivano K 2  (63 ± 9.7) |
| 79 | 47 | 17.33 | -16 |
| 68 | 51 | 6.33 | -12 |
| 68 | 55 | 6.33 | -8 |
| 52 | 66 | -9.67 | 3 |
| 51 | 45 | -10.67 | -18 |
| 63 | 70 | 1.33 | 7 |
| 42 | 51 | -19.67 | -12 |
| 48 | 69 | -13.67 | 6 |
| 43 | 52 | -18.67 | -11 |
| 61 | 64 | -0.67 | 1 |
| Prosjek: | | | |
| 57.5 ±5.27 | 57 ±5.27 | -4.193 | -6 |

Ovi rezultati nisu loši jer grešku kamere moramo uzeti u obzir kod testiranja s led diodom, greška za 10 testova ispadne ±5.27ms po formuli (3). Te iz ovih 10 testova se vidi da je u prosijeku sustav ako uzmemo u obzir samo grešku testiranja jako blizu očekivanoj vrijednosti. Sve dobivene vrijednosti padaju unutar limitacija sustava, jer je maksimalna greska kamere ±33.33ms.

## Testiranje sustava za praćenje

Slika na kojoj se prikazuje tekst, Trokut, snimka zaslona, kvadrat

Sadržaj generiran uz AI možda nije točan.

Slika 5.3 Primjer testa

Test je rađen na dva markera s poznatim pozicijama za dani okvir svijeta. Prvo je napravljena kalibracija i dodan je okvir svijeta za testiranje, ishodište tog okvira je u rubu prvog markera veličine 3.25 cm te x i z osi idu uz stranice tog markera, drugi marker je na stalnoj poznatoj udaljenosti od prvog. Test je prikazan na (Slika 5.1), test je postavljen na 67 i 71 cm od pojedine kamere.

|  |  |
| --- | --- |
| Slika na kojoj se prikazuje snimka zaslona, grafika, umjetničko djelo  Sadržaj generiran uz AI možda nije točan.  Slika 5.4 Detekcija na prvoj kameri marker1 | Slika na kojoj se prikazuje tekst, snimka zaslona, dizajn  Sadržaj generiran uz AI možda nije točan.  Slika 5.5 Detekcija na prvoj kameri marker2 |

Na slikam (Slika 5.2 i Slika 5.3) su prikazane detekcije oba markera na prvoj kameri. Dobivene i očekivane vrijednosti su:

Tablica 5.2 Pozicije markera na prvoj kameri

|  |  |  |  |
| --- | --- | --- | --- |
| M1-C1 | Dobiveno[cm] | Očekivano[cm] | Odstupanje |
| x | 1.8 | 1.633 | 0.167 |
| y | 1.6 | 1.633 | 0.33 |
| z | 0.4 | 0 | 0.4 |
| M2-C1 | Dobiveno[cm] | Očekivano[cm] | Odstupanje |
| x | 10.4 | 10.03 | 0.37 |
| y | 8.4 | 8.16 | 0.34 |
| z | 0.8 | 0 | 0.8 |

Tablica 5.3 Rotacija markera na prvoj kameri

|  |  |  |  |
| --- | --- | --- | --- |
| M1-C1 | Dobiveno[°] | Očekivano[°] | Odstupanje |
| roll | 2.9 | 0 | 2.9 |
| pitch | -3.5 | 0 | 3.5 |
| yaw | -88.2 | -90 | 1.8 |
| M2-C1 | Dobiveno[°] | Očekivano[°] | Odstupanje |
| roll | 1.4 | 0 | 1.4 |
| pitch | -0.8 | 0 | 0.8 |
| yaw | -89.7 | -90 | 0.3 |

|  |  |
| --- | --- |
| Slika na kojoj se prikazuje tekst, snimka zaslona, grafika, Font  Sadržaj generiran uz AI možda nije točan.  Slika 5.6 Detekcija na drugoj kameri marker1 | Slika na kojoj se prikazuje snimka zaslona, grafika, grafički dizajn, Font  Sadržaj generiran uz AI možda nije točan.  Slika 5.7 Detekcija na drugoj kameri marker2 |

Na slikam (Slika 5.4 i Slika 5.5) su prikazane detekcije oba markera na drugoj kameri. Dobivene i očekivane vrijednosti su:

Tablica 5.4 Pozicije markera na drugoj kameri

|  |  |  |  |
| --- | --- | --- | --- |
| M1-C1 | Dobiveno[cm] | Očekivano[cm] | Odstupanje |
| x | 1.0 | 1.633 | 0.633 |
| y | 2.3 | 1.633 | 0.667 |
| z | -1.5 | 0 | 1.5 |
| M2-C1 | Dobiveno[cm] | Očekivano[cm] | Odstupanje |
| x | 9.7 | 10.03 | 0.33 |
| y | 8.2 | 8.16 | 0.04 |
| z | -0.4 | 0 | 0.4 |

Tablica 5.5 Rotacija markera na drugoj kameri

|  |  |  |  |
| --- | --- | --- | --- |
| M1-C1 | Dobiveno[°] | Očekivano[°] | Odstupanje |
| roll | 0.4 | 0 | 0.4 |
| pitch | -2.5 | 0 | 2.5 |
| yaw | -87.5 | -90 | 2.5 |
| M2-C1 | Dobiveno[°] | Očekivano[°] |  |
| roll | 2.1 | 0 | 2.1 |
| pitch | -2.2 | 0 | 2.2 |
| yaw | -88.8 | -90 | 1.2 |

Rezultati su približni očekivanim vrijednostima te je najveće odstupanje pozicije 1.5cm po z osi a u prosjeku je odstupanje pozicije 0.498cm, a najveće odstupanje kuta je 2.5° oko Z osi, a u prosjeku je odstupanje kuta 1.8°. Ovo nije naj precizniji test jer se u sučelju ispisuje samo jedna decimala, ali se u ROS2 Node šalju double vrijednosti s 15 decimala. Dobiven vrijednosti nisu loše te bi se daljim procesiranjem u ROS2 Node-u mogla još dodatno povećati preciznost. Bitno je napomenuti da kvaliteta intrinzične matrice jako utječe na rezultate, jer se očekuje kvadratni marker, greške u intrinzičnoj kalibraciji se najviše osjete na z poziciji, i rotacijama oko x i y osi. Još je bitno postaviti udaljenost među kamerama što točnije, što je veća udaljenost kamera od željenog okvira to je veća konačna greška pozicije.

## Testiranje ostatka sustava

Sustav je testiran na drugom sustavu kamera u LABUST-u, te je došlo do problema sa performansama i zamrzavanjem sustava zbog lošeg upravljanja dretvama, u kodu je napravljena neka zaštita za više dretveni rad, ali ima jako puno koda koji nije potrebno štitiiti te zauzima vrijeme u dretvi zbog tog dolazi do blokiranje, npr. u Node-manual-extrinsics se javlja jako puno funkcija za iscrtavanje sučelja i drugih ne kritičnih dijelova koji zauzimaju znatan dio programskog vremena i zaključavaju pristup drugim dretvama (mutex.lock()) to se puno više ističe na višim rezolucijama zbog veće zahtjevnosti procesiranja i iscrtavanja, no to nije jedini problem, u LABUST-u se javlja problem zamrzavanja ulaznog video prijenosa (*eng. stream*), tome nije uspješno ustanovljen uzrok za vrijeme pisanja ovog rada. Sustav je rađen na Windows-u ali je također u sklopu projekta cilj bio prenijeti kod na Linux te je to bilo donekle uspješno, ali neke funkcionalnosti u kodu su zahtijevale dosta ispravljanja grešaka (*eng. debugging*), te je zbog toga nastavljena implementacija na Windows-u.

Ostale funkcionalnosti sustava rade kako je opisano u radu[1] za lokalno spojene kamere, a kod IP kamera u LABUST-u se javljaju prije navedeni problemi te je jako teško ispitati funkcionalnosti zbog loših performansi programa. Radi tih problema testiranje inije napravljeno u LABUST-u.

# Zaključak

Rad zadovoljava funkcionalnosti potrebne za praćenje površinskih vozila, ali zbog dizajna originalnog koda dolazi do loših performansi sustava, radi problema s više drevnosti nije pogodan za stalnu uporabu. Rad je idealan za testiranje i korištenje na terenu za brzo dobivanje svih potrebnih parametara kamere, kod jednostavnijih sustava kamera je sasvim funkcionalan i može biti korišten za praćenje objekata, te je jako brz način za dobiti podatke o poziciji i orijentaciji objekata unutar ROS2 sustava.

Sustav je prilično intuitivan no neki od Node-ova zbog limitacija dizajna inicijalnog sustava su nepotrebno kompliciraniji nego što bi mogli biti, te bi se uz malo popravaka mogao povećati potencijal nadogradnje sustava. Kao što je prikazano u ovom i inicijalnom radu[1] sustav je dosta precizan te su sve funkcionalnosti unutar očekivanog raspona ili prihvatljive točnosti, za očekivani rad je potrebna dodatna optimizacija baze sustava kako bi se program mogao zapravo koristiti u punom potencijalu. Zbog velike količine koda i kompleksnosti sustava optimizacija nije lagana, a dodatna nadogradnja samo komplicira stvari. Sustav je obećavajuća ali trenutno nije u stanju za pouzdano korištenje već samo za neke kratkoročne projekte ili testove.

Prednosti ovog sustava su brzina dobivanja željenog efekta bez neke velike pripreme, rapidno testiranje drugih ROS sustava bez potrebe izrade sustava za praćenje iznova za svaki različit sustav kamera, lagan i brz način za validaciju drugih sustava iste funkcionalnosti i nepoznate točnosti. Mana ovog sustava je limitacija na jednostavnije sustave kamera zbog loših performansi, relativno visoka memorijska i procesorska zahtjevnost za pokretanje i limitiranost na Windows OS.

# Literatura

1. Weber, A. (2025). *Podvodna lokalizacija u eksperimentalnom bazenu korištenjem kamera* (Diplomski rad).Zagreb: Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva. Preuzeto s <https://urn.nsk.hr/urn:nbn:hr:168:857633>
2. Paul Rensing. *Measuring Camera Display Latency*. FRC Team 2877, LigerBots

# Sažetak

**Lokalizacija površinskih vozila korištenjem sustava kamera**

Rad se bavi nadogradnjom postojećeg sustava rađenog u C++ za kalibriranje i testiranje sustava kamera te dodaje mogućnost mjerenja kašnjenja i lokalizacije markera u željenom okviru svijeta. Testovima je pokazano da sustav za određivanje kašnjenja daje očekivane rezultate, testiranje rad sustava za praćenje markera je pokazalo prosječna greška od 0.5mm na udaljenosti 70cm što ovisno o primijeni može biti prihvatljivo. Također su istaknuti problemi i limiti sustava na koje se naišlo u testiranju. Sustav je izmijenjen kako bi bio što fleksibilniji za bilokakvu konfiguracija kamera.

**Ključne riječi**: aruco, računalni vid, lokalizacija, kašnjenje, ROS, GUI

# Summary

**Localization of Surface Vehicles Using a Camera System**

The work involves upgrading an existing C++-based system for calibrating and testing camera systems by adding the capability to measure delay and localize markers within a desired world frame. Tests have shown that the delay measurement system provides expected results. Testing of the marker tracking system showed an average error of 0.5 mm at a distance of 70 cm, which may be acceptable depending on the application. The limitations and issues encountered during testing were also highlighted. The system was modified to be as flexible as possible for any camera configuration.

**Keywords:** ArUco, computer vision, localization, delay, ROS, GUI

# Skraćenice

UDP User Datagram Protocol protokol za prijenos podataka mrežom

IP Internet Protocol adresa računala na mreži

HSV Hue, Saturation, Value način definiranja boje

ROI Region of intrest dio slike koji je u interesu

ROS *Robot operating system* program za robotske sustave

OS *Operating system* sustav koji se nalazi na računalu

# Privitak

Privitak je također opcionalno poglavlje (u dogovoru s mentorom).

Sadržaj koji se stavlja u privitak je, općenito, nešto što je, kao cjelinu, prikladno izdvojiti iz sadržaja samog rada.

Mogući primjer je tehnička dokumentacija vezana uz završni rad - npr. električka i položajna shema sklopa, sastavnica, predložak tiskane veze, plan bušenja, ispis programa s detaljnim opisom.

Drugi primjer uključuju upute za korištenje rezultata rada (softvera ili hardvera), detaljni ispisi mjerenja čiji su rezultati sažeto ili grafički prikazani u radu. Ako se radi o softveru, uobičajeno je navesti podatke o platformi na kojoj se izvodi (npr., karakteristike uređaja i operacijskog sustava te pomoćnog softvera), kao i upute za instalaciju.

U privitku nemojte koristiti stilove razine Heading, već samo (nenumerirani) stil Podnaslov.

Na primjer:

Instalacija programske podrške

Upute za korištenje programske podrške

.

.

.

Ostali savjeti

U izborniku Tools - opcija Language postavite Croatian kao *default* jezik. Na kraju provedite strojnu provjeru teksta (*spell checking*, ako ga imate ugrađenoga), ali svakako pažljivo i pročitajte vlastiti tekst.

Ako nemate ugrađeni *spelling checker* za hrvatski jezik, možete se poslužiti Hascheckom (izgovara se Hašek, a ime dolazi od kratice za Hrvatski akademski *spelling checker*), dostupan putem poveznice <https://ispravi.me/>.