

Ryhmä 01-K.O.U.R.A.

Jäsenet/Members:

nimi	pääaine
Aapo Lemettinen	Informaatioteknologia
Viktor Solianoi	Automaatio ja robotiikka
Kasper Koivula	Automaatio ja robotiikka
Daniel Lewis	Automaatio ja robotiikka

K.O.U.R.A. - Kinetically Operated Unmanned Robotic Assistant

- K.O.U.R.A. - Kinetically Operated Unmanned Robotic Assistant
 - 1. Projektin idea ja tavoite
 - 1.1 Projektin idea ja tavoite
 - 1.2 Lopputulos – miten idea toteutettiin
 - 1.3 Esittelyvideo
 - 2. Tiimi ja vastuualueet
 - 3. Toteutus
 - 3.1 Mekaniikka ja ulkorunko, ajon laitteistopuoli
 - 3.2 Sähkö, kytkeytäkaaviot, lohkokaaviot
 - 3.3 Yhteys robottin ja ohjauslaitteen välillä
 - 3.4 Robotikäsi SO-101, ohjelmistopuoli
 - 3.5 Ajon ohjelmistopuoli
 - 3.6. Web-pohjainen ohjaussovellus etäohjattavalle robottiille
 - 3.7 TUNNISTUSOHJELMAT: Tölkki- ja ihmistunnistus YOLO11-mallilla ja YOLO5-mallilla
 - F1–Confidence Curve
 - 2) Precision–Confidence Curve
 - 3) Precision–Recall Curve
 - 4) Recall–Confidence Curve
 - – Confusion matrix, datasetin jakauma, oppimiskäyrät ja esimerkkikuvat
 - 1) Confusion Matrix
 - 2) Normalized Confusion Matrix
 - 3) Datasetin jakauma ja bbox-tiheys
 - 4) Koulutukseen oppimiskäyrät
 - 4. Projektin tulokset – mitä kolmen kuukauden projektin aikana saatui aikaan
 - 5. Mahdollinen projektin jatkokehitys ja kestävä kehitys
 - Teknologiset jatkokehitykset:
 - Autonominen toiminta ja esineiden tunnistus
 - Autonomisen lajittelun ja kuljetus kierrätyspisteelle
 - Ympäristödatan keruu ja analysointi
 - Energiaratkaisut ja itse latautuminen
 - Kestävä kehitys ja kaupunkiympäristön siisteys
 - Laajennus kohti kaupunki- ja palvelurobottia
 - 6. Mitä projektin aikana opittiin + kuvat projektin aikana

1. Projektin idea ja toteutus

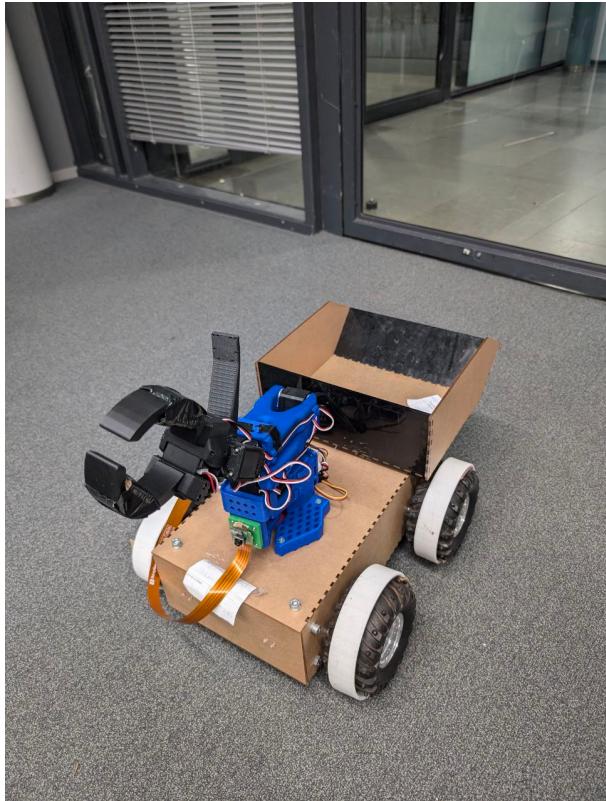
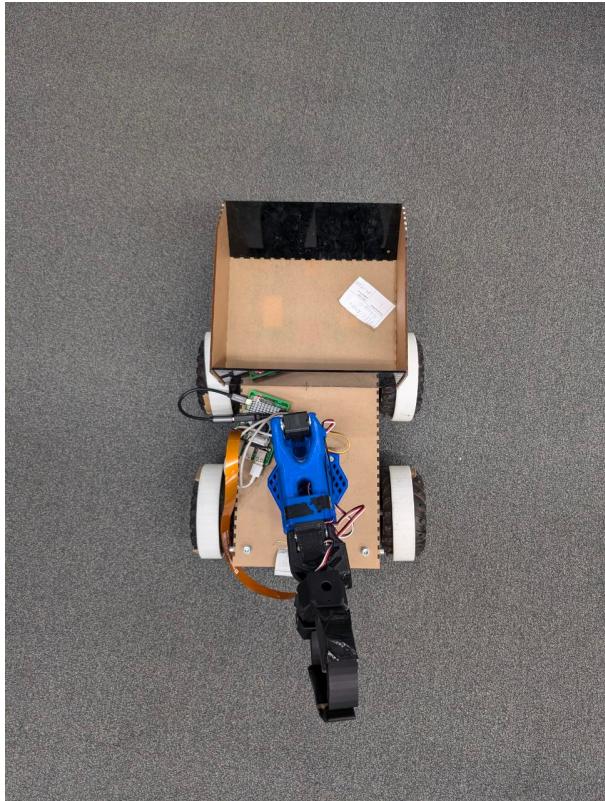
1.1 Projektin idea ja tavoite

Ideana oli rakentaa etäohjattava, langaton robotti, joka liikkuu pyörillä ja jossa on kamera sekä robottikäsi erilaisten esineiden käsittelyistä varten toiminta-alueellaan. Robotin tarkoitus on olla monikäytöinen laite erilaisiin etäohjattaviin tehtäviin ihmisen ohjauksesta ja tulevaisuudessa mahdollisuksien mukaan myös autonomisessa käytössä. Tällaisen robottin kehittämisen ideana oli tarjota ihmisille mahdollisuus työskennellä kaikenlaissä olosuhteissa pelkän kannettavan tietokoneen ja rakennetun robottiin avulla. Autonomiset toiminnot voisivat lisäksi mahduttaa tärkeiden tehtävien suorittamisen korkealla laadulla ilman suoraa ihmisen osallistumista. Tällä hetkellä robotti on ennen kaikkea erikoistunut erilaisten tölkien ja pullojen keräämiseen.

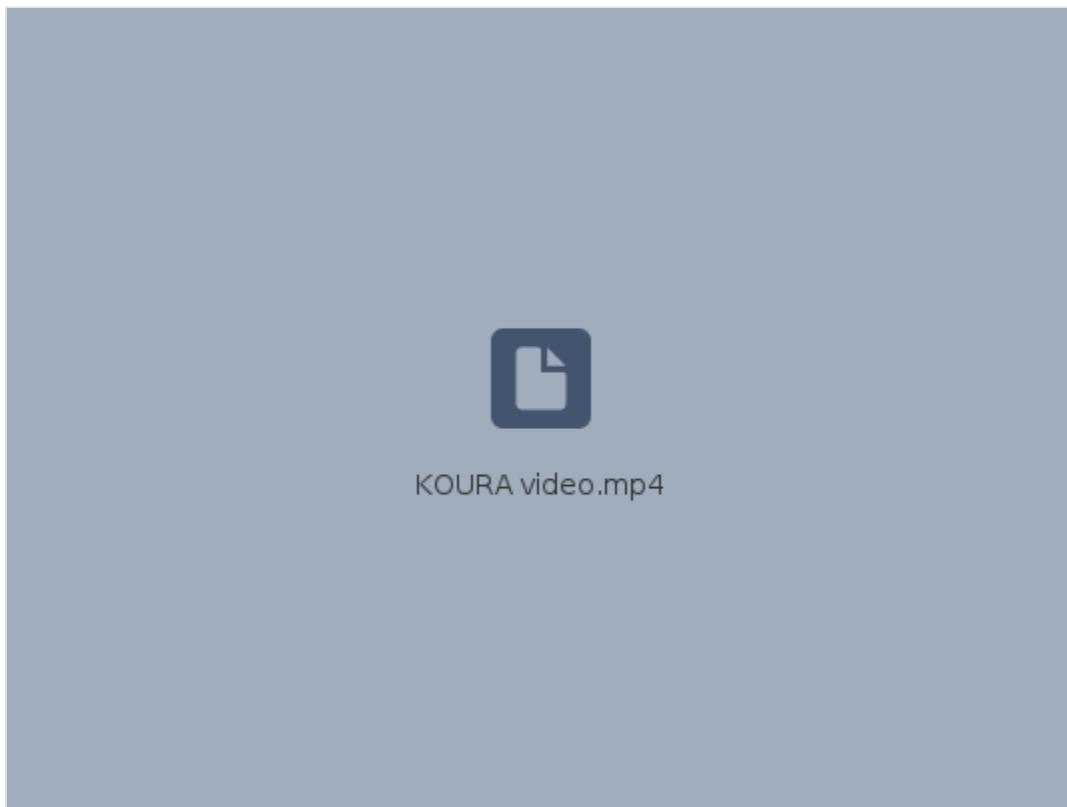
1.2 Lopputulos – miten idea toteutettiin

Projektiin toteutettiin seuraavasti: rakennettiin robotti, jonka runko on tehty akryylilevyistä ja jota ohja Raspberry Pi 5. Robotti liikkuu neljän tasavirtamoottorin avulla, kamerana toimii Raspberry Pi Camera 3, ja robottikätenä on avoimen lähdekoodin SO-101-malli, jossa on muokatut ja osittain itse suunnitellut tarttujarakenteet. Robotti toimii Wi-Fi-yhteydellä ja sitä ohjataan langattomasti web-sovelluksesta kannettavalta tietokoneelta tai pöytätietokoneelta.





1.3 Esittelyvideo



YouTube linkki: <https://youtube.com/shorts/VRBAXeBX33M>

2. Tiimi ja vastuualueet

Vastuualueet, Viktor Solianoi:

- etäohjauksen ja yhteyden muodostamisen koodin luominen ja kehittäminen, jossa signaalit välitetään WiFi-verkon sisällä WebSocket-yhteyden kautta, services - osio tekeminen web-soveluksessa.
- ajologiikan ja -koodin kehittäminen robottille sekä kameran toiminnan toteuttaminen Raspberry Pi -ympäristössä
- Raspberry Pi:llä koodin ja tarvittavan datan yhdistäminen siten, että ne voidaan siirtää langattomasti web-sovellukseen toiselle laitteelle
- SO-101-robottikäden leader–follower -ohjauksen koodin kehittäminen, jotta käyttävä voidaan käyttää etänä
- Esittelyvideon editointi

Vastuualueet, Daniel Lewis:

- mekaanikan osien suunnittelu ja toteutus sekä ajolaitteiston kehitys
- 3D-mallinnus, laserleikattavien osien mallinnus
- sähköjärjestelmän suunnittelu, rakentaminen, testaus sekä käyttöönotto
- osien kokoonpano, kaapeloointi, johdotus, sähköliitokset sekä juottaminen

Vastuualueet, Aapo Lemettinen

- SO-101-robottikäsien ohjelmointi, lerobot ja lekiwi ohjauksen kannalta tärkeiden tiedostojen selvitys ja ensimmäisten etäohjaus versioiden koodaus
- raspberryin valmistelu, ohjelmointi, ssh yhteys helpompaan komentojen syöttämiseen ja eri työn vaiheissa koodien testaus.
- Projektin alkuvaiheen: servoja käytävän drive raspberry koodin teko ja drive koodin, sen hetkisen kameraohjelmakoodin ja lerobot teleoperaaten käynnistävän start koodin koodaus.
- Linux ongelmat
- Lipojen lataus

Vastuualueet, Kasper Koivula:

- soveluksen suunnittelu ja koodaaminen
- Ihmisen- ja tulkintunnistusohjelman toteutus
- Värientunnistusohjelman koodaus

3. Toteutus

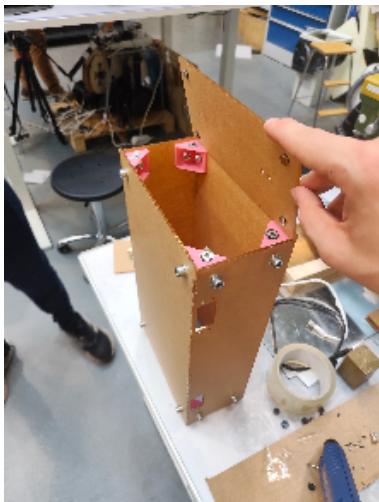
3.1 Mekaniikka ja ulkorunko, ajon laitteistopuoli

KOURA-robotti koostuu kahdesta pääkokonaisuudesta: laserleikatusta akryylirungosta ja sen päälle kiinnitetystä LeRobot SO-101 -robottikädestä. Runko sisältää ajomekaniikan, elektroniikan, akuston sekä tölkien ja pullojen kuljettamista varten rakennetun tavaratilan. Robottikäsi vastaa esineiden poiminnasta ja pudottamisesta tavaratilaan.

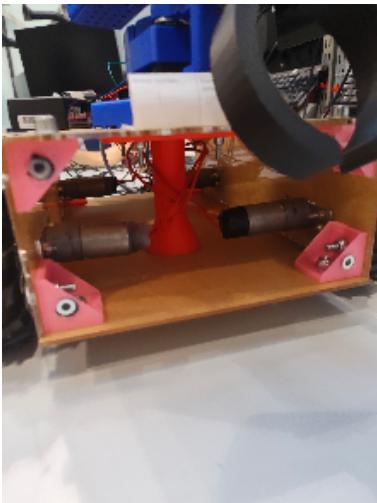
Runko

Rungon perusmuoto on laatikkomainen kotelo, jonka sivut on tehty laserleikatusta akryylistä. Sivulevyt kiinnityvät toisiinsa 3D-tulostettujen kulmapalojen avulla. Liitoksissa käytetään lukkomuttereita, jotta rakenne kestää tärinää ja toistuvaa käyttöä. Laatikkojen mallinnuksessa käytettiin Boxes.py sivua. Laatikon mitat 20cm x 30cm x 10cm, akryylilevyn paksuus 3mm.

Etu- ja takakansi ovat irrotettavia huollettavuuden parantamiseksi. Kannet kiinnityvät runkoon magneettien avulla, ja ne lepäävät rungon sisäpuolisilla laipoilla siten, että magneetit eivät kanna koko kuormaa. Näin kannet on helppo irrotaa, mutta ne pysyvät ajon aikana tukevasti paikoillaan.



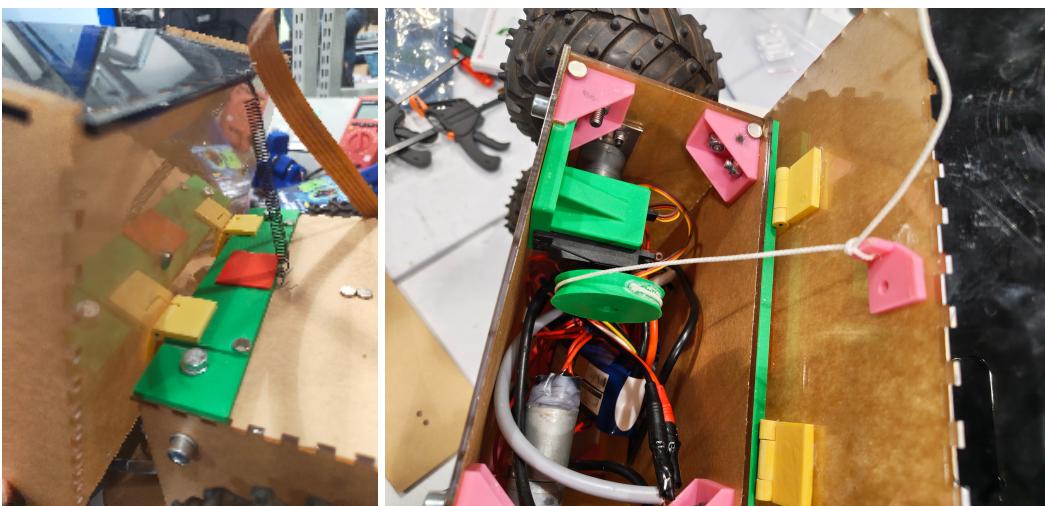
LeRobot SO-101 kiinnitettiin rungon yläosaan. Jotta robottikäden aiheuttamat väentövoimat eivät taivuttaisi akryylikattoa, rungon sisään on lisätty 3D-tulostettu pylväsmäinen tukirakenne. Näin robottikäden kuorma jakautuu rungon ylälevyltä suoraan pohjaan, ja runko säilyy jäykänä myös silloin, kun käsi liikkuu ja nostaa esineitä.



Kouran ajotehtävänä on kuljettaa tölkkejä ja pulloja, ja niiden keräystä varten rungon sisään on rakennettu kippavaa tavaratila. Tavaratila muodostuu laserleikatusta laatikosta, jonka sivut ja pohja on tehty akryylistä, sekä 3D-tulostetuista saranarakenteista, jousi- ja magneettipalautusmekanismista ja servolla sekä narulla ohjattavasta vetomekanismista.

Saranat on mallinnettu ja 3D-tulostettu siten, että ne toimivat yhdessä metallipinnin kanssa, metallipinnit vain puristetaan sisään. Tavaratilan laatikko kiertyy saranan ympäri ja pääsee kallistumaan riittävästi, jotta sisältö tyhjenee selkeästi ulos, mutta ei kuitenkaan niin paljon, että laatikko jäisi väärään asentoon. Palautusmekanismina käytetään jousien ja magneettien yhdistelmää: jousi vetää laatikon automaattisesti takaisin yläasentoon, ja magneetit pitävät sen yläasennossa ajon aikana.

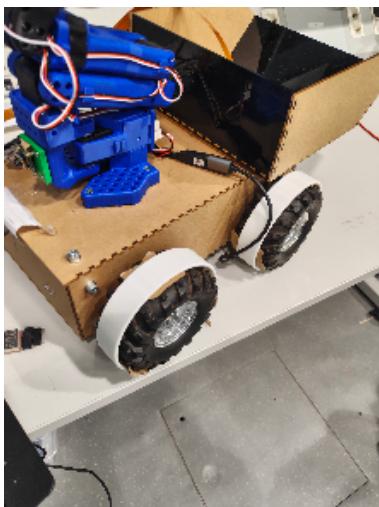
Vetomekanismi on toteutettu 180°-servolla ja narulla. Servo pyörittää 3D-tulostettua vetopyörää, johon naru on kiinnitetty. Naru kulkee takakannessa olevan reiän läpi ja kiinnitetty tavaratilan ankkuripisteeseen. Kun servo pyörii, naru kiristy ja vetää laatikon alas, kipaten sisällön. Kun servo vapautetaan, jousi ja magneetit palauttavat laatikon takaisin yläasentoon.



Robotti liikkuu neljällä pyörällä, ja ajomekaniikka on luisto-ohjattu. Ajossa robotin suuntaa muutetaan pyörittämällä oikean ja vasemman puolen pyöriä eri nopeuksilla tai eri suuntaan: kun toinen puoli pyörii eteenpäin ja toinen taaksepäin, robotti voi käännyä lähes paikallaan. Ratkaisu on mekaanisesti yksinkertainen, koska erillistä ohjausniveltä tai käännyviä etupyöriä ei tarvita.

Pyörinä käytetään maastorenkaita, joissa on alun perin melko korkea kuviointi ja suuri kitka. Luisto-ohjatussa ajossa korkea sivuttaiskitka tekee käänymisestä raskasta, sillä renkaiden on pakko hieman luitaa sivusuunnassa. Kitkan pienentämiseksi maastorenkaiden ympäälle lisättiin 3D-tulostetut ulkorenkaat. Nämä ohuet, sileämät renkaankuoret liu'utetaan maastorenkaiden päälle ja kilataan paikoilleen pahvipaloilla, jolloin ne pysyvät ajossa tukevasti kiinni mutta ovat silti helposti irrotettavissa. Ulkorenkaat vähentävät renkaiden sivuttaiskitkaa ja helpottavat käänymistä erityisesti kuivalla, pitävällä alustalla.

Liike tuotetaan DC-moottoreilla, jotka on kiinnitetty ohut metallilevystä tehtyihin moottorikiinnikkeisiin. Moottoreiden ja renkaiden välissä on asennettu metalliset adapterit, jotka siirtävät moottorin akselin momentin renkaalle. Adapterien kiinnityksissä on käytetty ruuvilukitetta, jotta ajon tärinä ei löysyttäisi liitoksia.



Osien kiinnitykset ja kokoonpano

Runko kootaan 3D-tulostetuilla kulmapaloilla ja kasataan ruuveilla, prikoilla ja lukkomuttereilla kiinni. Etu- ja takakannet kiinnitetyt magneeteilla runkoon, ja magneetit on liimattu paikoilleen pikaliimalla. DC-moottorit on kiinnitetty koneruuveilla käsin tehtyihin ohutmetallilevyihin, ja levyt on puolestaan ruuvattu koneruuveilla, prikoilla ja lukkomuttereilla akryylilevyseiniin. Renkaat kiinnitetään DC-moottoreihin metalliadaptereilla ja koneruuveilla, adapterien kiinnityksissä on käytetty ruuvilukitetta, sillä muuten tärinä löysyttäisi liitokset. LeRobot SO-101 on kiinnitetty uppokantaruuveilla ja lukkomuttereilla pylväsmäiseen tukirakenteeseen, ja rakenne on ruuvattu pohjaan kiinni. LeRobot on kokoonpantu servojen mukana tulleilla ruuveilla ja osilla. Tavaratilaan on pikaliimattu saranat, ja saranat on pikaliimattu kiinni alustaan, joka kiinnittyy kulmapalojen ruuveihin. Magneetit tavaratilan pohjalla sekä vstamagneetit tukirakenteessa on kiinnitetty pikaliimalla. Vetomekanismin servo on kuumaliimattu 3D-tulostettuun pidikkeeseen, joka on puolestaan pikaliimattu rungon kulmapalaan. Vetonarun pyörä on pikaliimattu lautaseen, joka kiinnittyi ruuvilla servoon, ja naru viedään takakanseen läpi olevasta reiästä ja kiinnitetään ankkuriin. Kamera on kiinnitetty adapteriin pienillä ruuveilla ja adapteri kiinni LeRobottiin. Kokoonpanossa käytettiin tavallisia käsitökaluja, joten laitteen huollettavuus on hyvä. Raspberryn kotelo kiinnitettiin kaksipuolisella teipillä laatikon päälle.

3D osia joita käytettiin:

[Ulkorengas.stl](#)

[Tavaratilan_alusta.stl](#)

[Sarana2.stl](#)

[Sarana1.stl](#)

[Servon_pidiike.stl](#)

[Vetopyora.stl](#)

[Bottom-V2-v3.stl](#)

[Top-V2-v2.stl](#)

LeRobot SO-101

Robottikäden 3D-tulostettavat osat ovat saatavilla verkosta. Tulostimme osat, joista kolmea osaa muokattiin. Gripperit muokattiin kaareviksi juuri tölkkien keräämästä varten. Ranteeseen lisättiin pylväs kameraa varten. Eteen lisättiin adapteri kameralle. Muita muokkauksia ei tehty malliin.

Muokatut osat:

[Wrist_Roll_Pitch_SO101 - Part 1.stl](#)

[Moving_Jaw_SO101 - Part 1.stl](#)

[Wrist_Roll_Follower_SO101 - Part 1.stl](#)

Kameran adapteri:

[Pi_Camera-LeRobot.stl](#)

3.2 Sähkö, kytkentäkaaviot, lohkokaaviot

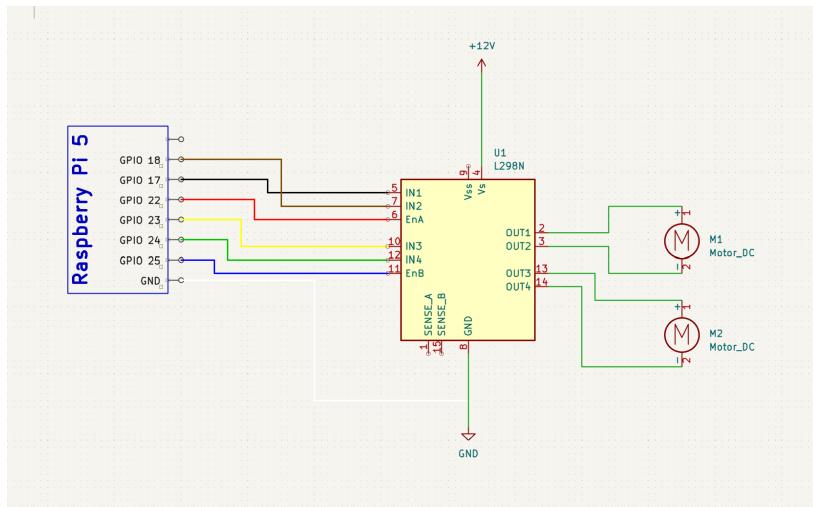
KOURA-robotin sähköjärjestelmä on rakennettu niin, että eri virtaa tarvitsevat osat saavat oman akkunsa. Näin turvataan vakaa sähkö ajan läpi. Eri osat tarvitsevat eri määärän jännitettä, se on toinen hyvä syy jakaa osat omille virtalähteille. Raspberry Pillä on oma akkunsa, LeRobot SO-101 on oma akkunsa ja DC-moottoreilla on oma akkunsa. Raspberry Pi on ohjaat GPIO pinnien avulla L298N moottorinohjainta ja usb kaapelin avulla LeRobottia.

Raspberry Pi:n virransyöttö on yksi kriittisimmistä osista koko järjestelmää. Pi vaatii sekä tarkasti säädetyn 5 voltin jännitteen että hetkellisesti jopa 5 ampeerin virran, erityisesti käynnistyksen ja kuormituspiikkien aikana. Tästä syystä Pi:tä ei kytketä suoraan akkuun, vaan sen eteen on sijoitettu tehokas DC/DC-step down -muunnin, joka muuntaa 12 V akkujännitteestä vakaan 5 V / 5 A -syötön. Ennen lopullista ratkaisua kokeiltiin yksinkertaisempia jännitteenalennusratkaisuja, mutta ne eivät pystyneet syöttämään Pi:tä luotettavasti: laite saattoi käynnistyä, mutta kaatui helposti, kun moottorit kuormittivat järjestelmää. Riittävän suuritehoinen DC/DC-muunnin ratkaisi ongelman, Pi pysyy toiminnessa myös silloin, kun robotti liikkuu ja virtaa kuluu useassa haarassa samanaikaisesti.

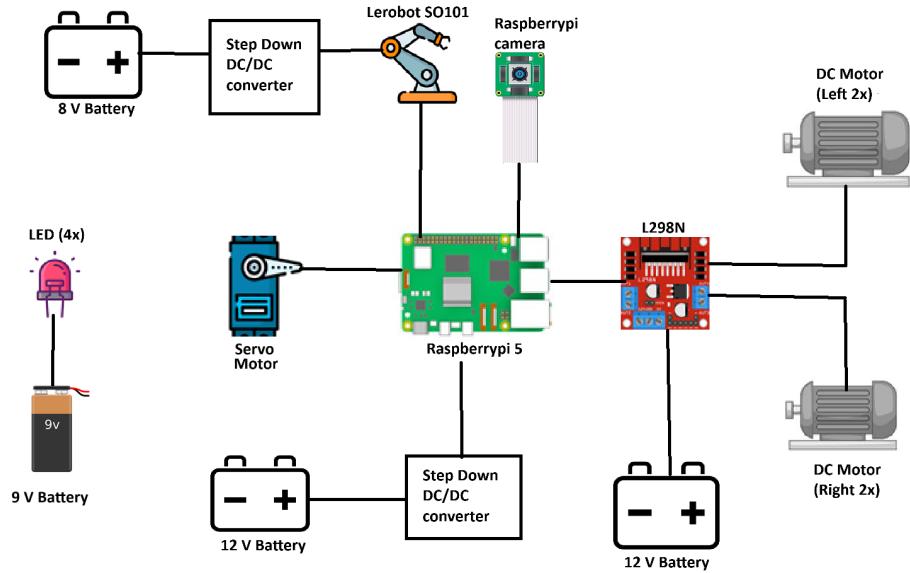
Ajomekanikaasta vastaavat DC-moottorit ohjataan erillisen moottorinohjaimen kautta. Moottorinohjaimen tehtäväն on vastaanottaa ohjaussignaalit Raspberry Pi:ltä, syöttää moottoreille tarvittava jännite ja virta sekä mahdollistaa sekä suunnanvaihto että portaattomasti säädettyä nopeus. Moottoripirin ja ohjauspirstin maadoitus on yhteen, jotta ohjaussignaalit toimivat oikealla viitetasolla, mutta tehopuolella käytetään moottoreille omaa akkua, joka kestää paremmin kuormituspiikkijä.

Koko sähköjärjestelmän johdotuksessa on pyritty yhdistämään helppo huollettavuus ja riittävä mekaaninen kestävyys. Haaroituksissa ja jakopisteissä käytetään sokeripalaliittimiä, joihin voidaan kiinnittää ja joista voidaan irrottaa johtimia ilman juottamista. Pysyvät liitokset, kuten moottoreiden omat johdot, on juotettu suoraan kiinni liittimiin tai komponentteihin, ja niiden ympärille on lisätty kuumaliimaavetosuojaksi. Jokainen liitos tarkistetaan käytännössä yksinkertaisella nykäisytestillä. Tämä on tärkeää erityisesti liikkuvissa osissa, kuten moottoreissa ja tavaratilan mekanismissa, joissa johdot altistuvat jatkuvalle liikkeelle.

Lisätoiminnot täydentävät sähköjärjestelmää. Robotin etuosaan on asennettu LED-ajovalot, jotka kytketään päälle erillisellä mekaanisella kytkimellä. LEDit saavat jännitteen joko omalta pieneltä akultaan eikä vuorovaikuta muun sähköjärjestelmän kanssa. Mekaaninen kytkin on sijoitettu sellaiseen paikkaan että robottikäsi saa sen leittettävä päälle tai pois.



K.O.U.R.A. block diagram



Kouran komponentit ja osat:

Osa	Määrä	
Raspberry Pi 5	1	
Raspberry Pi camera	1	

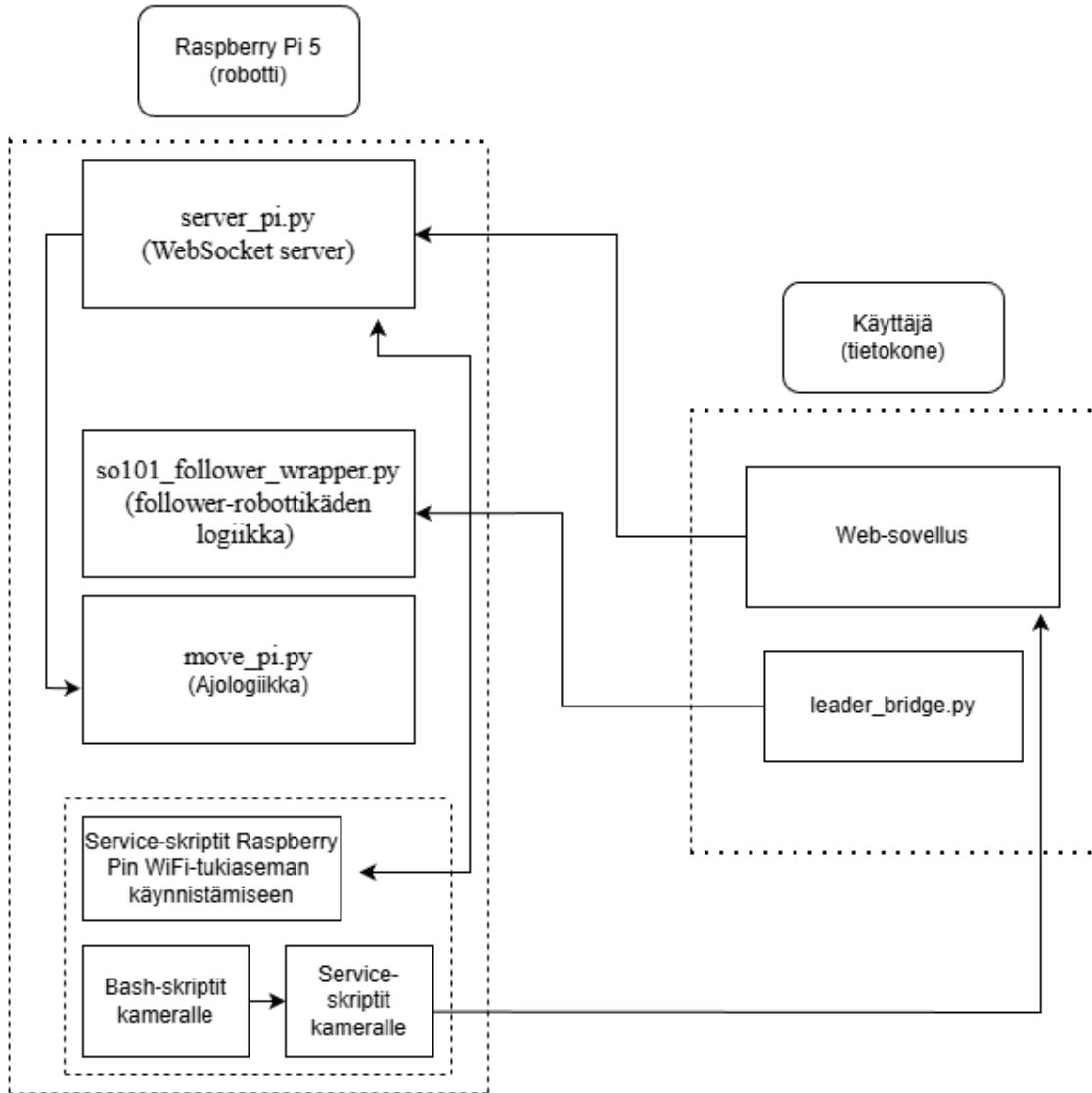
LeRobot SO-101 Leader	1	
LeRobot SO-101 Follower	1	
Joy-IT STEP-DOWN DC/DC-CONVERTER 9-35/5V 5A (Raspberry Pi:lle)	1	
Lm2596 Buck Converter Dc-dc Adjustable Step Down Power Supply Module (LeRobotille)	1	
L298N Moottorinohjain	1	

DC-vaihdeoottori	4	
Servo 180°	1	
Akku, LiPo 5000 mAh (DC-moottoreille)	1	
Akku, LiPo 2200 mAh (Raspberry Pi:lle)	1	

Akku, LiPo 6000 mAh (LeRobotille)	1	
9 V paristo	1	
Valkoinen LED-valo	4	

3.3 Yhteys robotin ja ohjauslaitteen välillä

Koko software-puoli on kokonaiskaaviossa esitetty suunnilleen alla olevan kuvan mukaisesti:



Ohjaus

Raspberry Pi ja ohjauslaitteen välisen yhteyden muodostamiseksi käytetään seuraavaa toimintaketjua:

- Käynnistyessään Raspberry Pi luo heti oman WiFi-tukiaseman.
- Käynnistyessään Raspberry Pi käynnistää välittömästi **server_pi.py**-sovelluksen, joka toimii WebSocket-yhteyden palvelinpuolen. Raspberry Pi käynnistää myös **move_pi.py**- ja **so101_follower_wrapper.py**-skriptit, jotka sisältävät tarvittavan logiikan komentojen vastaanottamiselle ja suorittamiselle Raspberry Pi:llä. Lisäksi Raspberry Pi:llä käynnistetään tarvittavat skriptit kameran käynnistämiseksi, kuvien muuntamiseksi ja niiden siirtämiseksi web-sovellukseen puolelle.
- Käyttäjä yhdistää tietokoneensa Raspberry Pi luomaan WiFi-verkkoon.
- Käyttäjä käynnistää web-sovelluksen, jossa backend-puolella on valmiiksi toteutettu WebSocket-client puoli.

```
cd ~/kouranpm run dev
```

- Käyttäjä syöttää sivulle Raspberry Pi:hin sidotun "**IP:port**"-osoittein ja muodostaa yhteyden robottiin.
- Robottikäyttä varten käyttäjä kytkee leader-robottikäden tietokoneeseen ja käynnistää tietokoneella **leader_bridge.py**-ohjelman, joka edellyttää, että yhteys Raspberry Pi:hin on ensin muodostettu sivun kautta.

```
conda activate lerobot
python leader_bridge.py --teleop-port /dev/ttyACM3 --teleop-id sol01_leader_laptop --pi-ws-url ws://10.42.0.1:8080/ws --fps 30
```

- Yhteys on nyt muodostettu. Robottia voidaan ohjata ja kameran videokuva välittyy.

Käytännössä kaikki on toteutettu siten, että kun tarvittavat ohjelmistot on asennettu, käyttäjän tarvitsee vain käynnistää web-sovellus ja robottikäyttö varten oma ohjelma. Kaikki robotin käyttöön tarvittava koodi käynnistyy Raspberry Pi:llä automaattisesti, ja käyttäjän tarvitsee vain käynnistää Raspberry Pi virtapainikkeesta. Kun yhteys Raspberry Pi:hin on muodostettu web-sovelluksen kautta, robotti on heti käytövalmis.

Liityvät kooditiedostot:

•Raspberry:

- [server_pi.py](#) - [server_pi.py](#)

Käynnistää käyttäjälle WebSocket-yhteyden IP-osoitteeseen 10.42.0.1 ja välittää tarvittavat komennot web-sovellukselta ohjelmille, joissa on määritelty komentojen käsitteilylogiikka haluttuja toimintoja varten.

- [service scriptit \(kamera\)](#) - [video_script.service](#) , [camera_converter.service](#)

Käynnistää kameran, muuntavat kuvat tarvittavaan formaattiin ja välittää ne web-sovellukselle annettuun osoitteeseen.

-[start ja wifi scriptit \(yhteys\)](#) - [start_script.service](#) , [wifi-switch.service](#)

Käynnistää WiFi-tukiaseman ja kaikki robotin toimintaa varten tarvittavat ohjelmat, jotta kun Raspberry käynnistetään virtapainikkeesta, kaikki on heti valmiina robotin käyttöön.

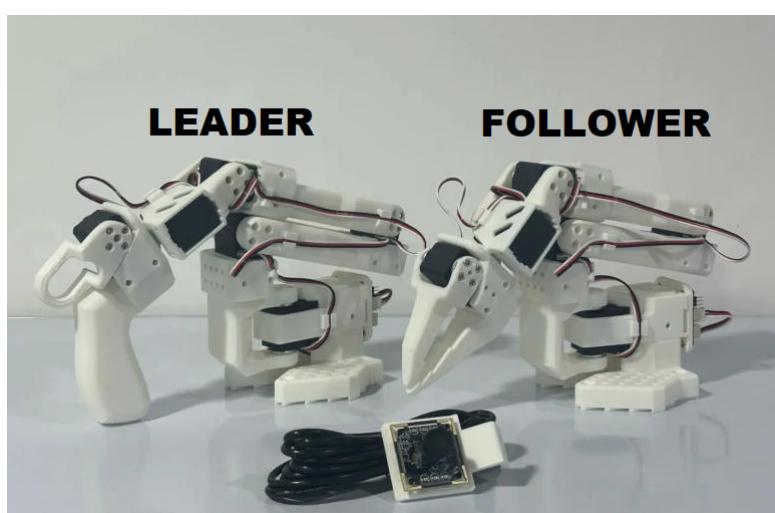
•Käyttäjä:

-[Web-sovellus \(3.6 luku\)](#)

3.4 Robottikäsi SO-101, ohjelmistopuoli

LeRobot SO-101 on avoimen lähdekoodin ohjelmisto ja robottikäsimalli. Kuka tahansa voi tulostaa tarvittavat osat 3D-tulostimella ja tilata sopivan ohjauspöörin sekä servomoottorit tämän robottikäden kokoamista varten. Järjestelmä perustuu leader–follower-malliin, jossa leader-robottikäsi ohjaa follower-robottikäyttöä, ja follower toistaa leaderin liikkeit. Jokaisessa robottikädessä on 6 servomoottoria eli 6 akselia, mikä on erittäin sopivaa meidän valitsemaamme käyttötarkoitukseen.

LeRobotin ja SO-101-robottikäden materiaaleihin voi tutustua täällä: <https://huggingface.co/docs/lerobot/sol01> sekä GitHub-repositorioissa <https://github.com/search?q=SO101&type=repositories>



Käytännössä kirjaston valmis koodi tarjoaa mahdollisuuden robottikäsiin käyttöön, niiden kalibointiin, oman sovelluskoodin kirjoittamiseen tästä robottikäytöistä varten sekä koneoppimiskokeiluihin.

Aalto-yliopistossa SO-101-robottikäsiä voi käyttää kysymällä niitä Thinkin' Rocks -organisaatiolta, joka toimii StartUp Saunassa, Viima-rakennuksessa.

Jokaisessa robottikädessä on USB-C-portti ja 5 V virtalähde. Perinteinen tapa ohjata robottikäsiä on liittää molemmat kädet samaan tietokoneeseen ja käynnistää teleoperate-skripti, joka kopioi leader-käden liikkeet follower-käelle ja välittää liikeinformaation lähes viiveettä näiden USB-C-kaapelien ja -porttien kautta. Projektissamme meillä ei kuitenkaan ollut mahdollisuutta käyttää molempia robottikäsiä samassa laitteessa, vaan leader-käätä käytetään robottia ohjaavalla kannettavalla tietokoneella, ja follower-käsi on kiinnitetty itse robottialustaan ja kytketty Raspberry Pi:hin.

Robottikäden ohjaus toteutettiin projektissamme seuraavasti:

- Kannettavalla tietokoneella robottikäsi asennetaan ja kalibroidaan porttien ja työtilan mukaisesti. Sama prosessi tapahtuu myös Raspberrylla.
- Kehitettiin leader_bridge.py -koodi, jonka tehtävänä on oikean kalibroinnin jälkeen lukea servomoottorien asennot ja lähetä ne WebSocket-yhteyden kautta Raspberry Pi:lle. Raspberryllä so101_follower_wrapper.py vastaanottaa servojen arvot ja siirtää robottikäden vastaaviin asentoihin. Leader-robottikäsi on kytketty kannettavaan USB-C-kaapelilla ja verkkovirtaan, kun taas follower-robottikäsi on kytketty virtalähteensä toimivaan akkuun ja USB-C:n kautta Raspberry Pi:hin.

Liityvät kooditiedostot:

•Raspberry:

- [so101_follower_wrapper.py](#) - ottaa vastaan tietoa leader-robottikäitä käytäväiltä kannettavalta ja siirtää follower-robottikäden samaan asentoon kuin leader. Yhteyden katketessa robottikäsi jää viimeiseen asentoon odottamaan uusia tietoja ja siirtyy heti samaan asentoon kuin leader-käsi, kun yhteys palautuu. Jos yhteys ylikuormittuu tiedolla, se käynnistetään automaattisesti uudelleen.

[so101_follower_wrapper.py](#)

- [lerobot kirjasto](#)

•Käyttäjä:

- [leader_bridge.py](#) - muodostaa yhteyden Raspberry Pi:hin luodun WebSocket-yhteyden kautta, lukee leader-robottikäden liikkeet ja välittää ne Raspberry Pi:lle jatkokäsittelyä varten.

[leader_bridge.py](#)

- [lerobot kirjasto](#)

3.5 Ajon ohjelmistopuoli

Moottorien ohjaus tapahtuu move_pi.py-ohjelmalla, joka käynnistyy automaattisesti aina, kun Raspberry käynnistyy. L298N-moduuli ohjaaa kahta robottipuolta – vasenta ja oikeaa – eli etu- ja takapöörät on synkronoitu oman puolensa mukaan.

Liikettä määrittelee pääasiassa set_direction-funktio, jolla on kaksi parametriä: linear ja angular.

•linear voi saada seuraavat arvot:

-1 -> ajaa taaksepäin

0 -> ei suoraa etenemistä

+1 -> ajaa eteenpäin

•angular voi saada seuraavat arvot:

-1 -> käänös vasemmalle

0 -> täysin suora ajo ilman käänymistä

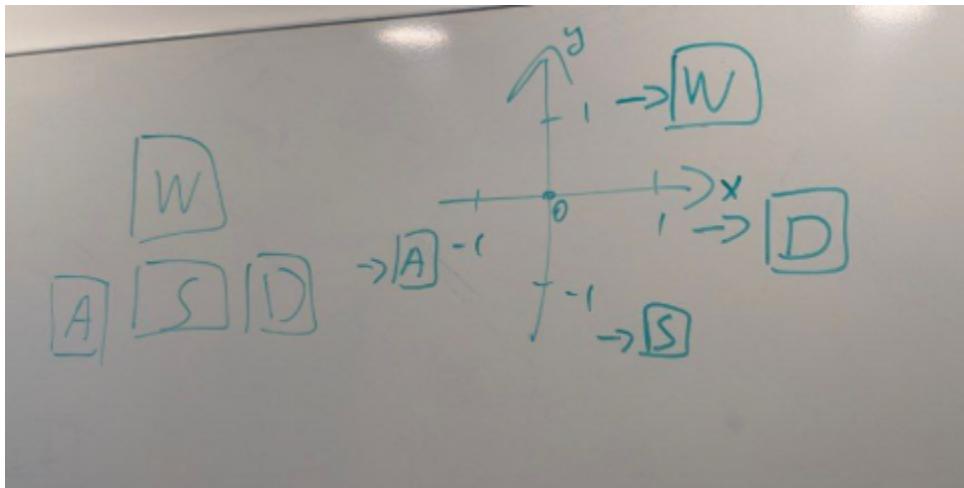
+1 -> käänös oikealle

Kun robotti käännyy esimerkiksi arvoilla linear: 1 ja angular: 1 tai -1 (eli robotti ajaa eteenpäin ja käännyy samalla), ulomman puolen pyörät pyörivät normaalilla nopeudella ja sisemmän puolen pyörät nopeudella, joka on 0,3 kertaa perusnopeus.

Kun robotti käännyy paikallaan arvoilla linear: 0 ja angular: 1 tai -1, toisen puolen pyörät pyörivät yhteen suuntaan ja toisen puolen pyörät vastakkaiseen suuntaan, jolloin robotti voi käännyä paikallaan.

Jos robotti on pois päältä, funktilo yksinkertaisesti pysäyttää moottorit.

Web-sovelluksen puolella toteutettiin myös W-, A-, S- ja D-näppäinten lukeminen, jotka välittävät tarvittavat arvot set_direction-funktioille. Esimerkiksi W+A lähettää arvot linear: 1 ja angular: 1, kun taas pelkkä A lähettää arvot linear: 0 ja angular: 1.



Liityvät kooditiedostot:

•Raspberry:

[move_pi.py](#) - asettaa tarvittavan ajologiikan, kun se saa oikeat komennot. Lisäksi siinä on logiikka tavaratilan tyhjentämiseen ja nostamiseen takaisin ylös.

[move_pi.py](#)

3.6. Web-pohjainen ohjaussovellus etäohjattavalle robotille

The screenshots show the K.O.U.R.A Teleoperation Platform interface. The top screenshot displays a connection profile named 'Testiajo' with host '10.42.0.1' and port '8080'. The bottom screenshot shows an empty list of profiles with a button to 'Luo profili' (Create profile).

Tein projektin web-pohjaisen ohjausovelluksen, jonka kautta käyttäjä voi ohjata robottia etänä sekä nähdä reaalialaista dataa robotin tilasta. Sovellus on rakennettu Reactilla ja se toimii selaimessa, eikä vaadi erillistä asennusta.

Idea ja tarkoitus

Sovelluksen tavoite oli tehdä robotin ohjaamisesta selkeää ja nopeaa, sekä mahdollistaa sekä manuaali- että automaattiohjaus. Lisäksi sovelluksen kautta pystyy seuraamaan telemetriaa ja hallitsemaan yhteysprofileja, jotta robotti voidaan yhdistää eri verkkoihin helposti.

Sovellus yhdistyy Raspberry Pi 5:een **WebSocket-yhteyden** kautta. Pi vastaanottaa ohjauskomennot ja lähetää takaisin telemetriaa.

Käyttäjä voi sovelluksella:

- ohjata robotin liikkeitä (eteenpäin, taaksepäin, sivulle, kääntyminen)
- Nähdä raspberry pi:n lähettämän kamerakuvan, kahdesta eri näkökulmasta sovelluksessa.
- vaihtaa manuaali-/automaattitilan (robotti ei vielä toimi automatisesti, mutta alkuperäisen suunnitelman mukaisesti sovelluksessa on valmiit ainekset sen käyttöönottoon.)
- valita tunnistettavat värit ja objektiityypit (robotti käyttää tätä automaattiajossa, joten ei käytössä)

- seurata reaalialkaista dataa: (eivät ole totuudenmukaiset, koska antureita ei ole käytössä, mutta toimivat havainneilmiönä, mihin kaikkeen robottia voi hyödyntää.)
- akun varaus
- lämpötila
- kosteus
- yhteyden vahvuus / viive
- luoda ja hallita yhteysprofiileja (esim. eri WiFi-verkot tai eri robotit)

Tekninen toteutus

Sovellus koostuu React-pohjaisesta frontendistä ja WebSocket-yhteydestä robottiin.

Ohjaus toimii niin, että käyttöliittymä välittää käyttäjän ohjauskomennot WebSoketiin kautta Raspberry Pi 5:lle, joka ohjaa robotin pyörien servoja differentiaaliohjauksella.

Sovelluksessa käytin seuraavia teknologioita:

Käyttöliittymä

- *React 18* – UI:n perusrakenne
- *JavaScript (ES6+)* & *JSX* – logiikka ja komponentit
- *Tailwind CSS* – nopea ja modulaarinen tyylkirjasto
- *shadcn/ui* – käyttöliittymäkomponentit (mm. painikkeet, valikot, kortit, selector-komponentit, sivupalkki)

Ikonit

- *Lucide React* – ikonit ohjaukseen ja telemetriaan (esim. Wifi, Gamepad, Thermometer)

Sivujen hallinta

- *react-router-dom* – reititys ja navigointi sivujen välillä

Datan haku ja välimuisti

- *@tanstack/react-query* – palvelinperäisen datan hallinta (telemetria, profiilit)

Suorituskyky ja havainnot

Sovelluksen toiminta todellisessa käyttöympäristössä oli vakaata ja yllättävän suorituskykyistä. WebSocket-yhteys Raspberry Pi 5:n ja selaimen välillä pysyi pääsääntöisesti hyvin tasaisena, ja viive oli käytännössä niin pieni, ettei sitä ohjaustilanteessa huomaa. Ohjauskomennot siirtyivät Pi:lle lähes reaalialjassa ja robotin differentiaaliohjattu liike vastasi hienosti käyttöliittymän painalluksiin.

Käytettävyys

Koko sovellus on suunniteltu mahdollisimman suoraviivaiseksi:

Käyttäjä näkee ohjauspaneelin, robotin tilan ja valikon yhdellä silmäyksellä.

Objekteja poimiva koura itsessään ei ole sovelluksen ohjaama – sitä ohjataan erillisellä ohjaimella. Sovelluksen tehtävä on hallita ajamista ja saada informaatioita.

Sovellus: [KOURA sovellus 10.12.zip](#)

3.7 TUNNISTUSOHJELMAT: Tölkki- ja ihmistunnistus YOLO11-mallilla ja YOLO5-mallilla

Projektissa toteutin kaksi erillistä tunnistusohjelmaa. Päädyin kahden mallin ratkaisuun, koska yhden mallin kouluttaminen hoitamaan sekä tarkka tölkki-/pullotunnistus että ihmistunnistus heikensi molempien luotettavuutta.

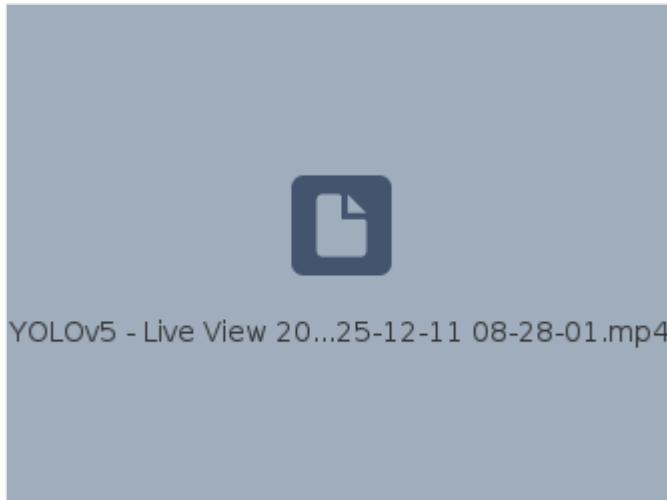
1. Ihmistunnistus + yleiset objektit (valmis malli, jota hyödynnetään Yolov5)

Ensimmäinen ohjelma tunnistaa:

- ihmiset

- älypuhelimet
- pullot
- muut yleiset perusobjektit YOLO5V-mallin valmiista luokista

Tämä ohjelma on tarkoitettu tilanteisiin, joissa robotin pitää havaita ihmisen tai muut suuremmat esteet. Se piirtää automaattisesti suorakulmion tunnistetun objektiin ympärille. Malli perustuu githubista löytyvään kirjastoon, mutta on muovattu Kouran robotin tarpeisiin sopivaksi.



video mallin toiminnasta Tokion keskustassa:

Linkki: <https://github.com/ultralytics/yolov5>

1. Tölkki-/pullo-/lasipullotunnistus

Toinen ohjelma on täysin erikseen koulutettu tunnistamaan:

- alumiinitölkit
- muovipullot
- lasipullot

Tämä vaati oman mallin, koska halusin tunnistuksen toimivan juuri robotin käyttöympäristössä ja valituilla objektiluokilla mahdollisimman tarkasti.

Mallin koulutusprosessi

Rakensi datasetin itse kuvamalla tölkkejä ja pulloja robotin todellisessa käyttöympäristössä. Tämä parantaa mallin tarkkuutta oikeassa tilanteessa, koska valaistus, etäisyys ja kuvakulma vastaavat käyttötilannetta.

Käytin YOLO11v-mallia, joka on kevyt mutta tehokas neuroverkko. Tein datasetin näin:

1. Keräsin ja kuvasin suuren määän kuvia.

Mukana oli häiriöobjekteja, erilaisia taustoja ja valaistuksia, jotka malli oppii robustiksi.

1. Labeloin kuvat itse.

Käytin Label Studion kaltaista työkalua, jolla merkitsin jokaisen tölkkin tai pullon.

1. Järjestin tiedostot YOLO-muotoon:

1. images/

1. labels/

1. classes.txt

Kaikki omissa kansioissaan.

1. Latasin datasetin Colabiin ja tein automaattisen train/val-jonon

90 % meni koulutukseen, 10 % validointiin.

1. Koulutin mallin YOLO11:llä.

Valitsin mallikooksi [yolo11s.pt:n](#), joka toimii hyvin myös Raspberry Pi 5:llä. Epoch-määrä oli riittävä sihen, että mAP vakiintui ja mallin tarkkuus nousi.

1. Sain ulos best.pt-mallin, jota käytän nyt sovelluksen ja robotin yhteydessä.

Suorituskyky ja havainnot

Tölkki-/pullo-/lasipullomalli toimi käytännön testeissä selvästi paremmin kuin alkuperäinen yhden mallin ratkaisu. Malli tunnisti tölkit ja muovipullot myös silloin, kun ne olivat osittain vinossa tai hieman varjossa. Tunnistusohjelma ei ole käytössä kourassa, mutta webkameran testausten perusteella malli toimii hyvin.

Haasteita ilmeni silloin, kun objekti oli lähes kokonaan pilossa esimerkiksi tuolinjalan tai laatikon takana. Tämä on täysin odotettavaa, koska YOLO vaatii näkyvää muotoa tehdäkseen rajaksen. Lisäksi äärimmäisen hämärässä valoissa havainto ei aina ollut varma — tämä liittyy datasetin alkuperäisiin kuvasolosuhdeisiin.

Ihmismalli toimi erittäin luotettavasti. Se tunnisti ihmiset hyvin eri etäisyysiltä, myös selkä edellä tai sivuprofilista. Tämä on tärkeää turvallisuuden kannalta, koska robotti toimii lattialla ja ihmisiä saattaa kulkea lähellä. Malli tunnisti myös muita yleisiä objekteja (känykkä, pullo), mikä helpottaa tilanteita, joissa robotti kulkee monimutkaisessa ympäristössä.

Haasteet ja jatkokehitys

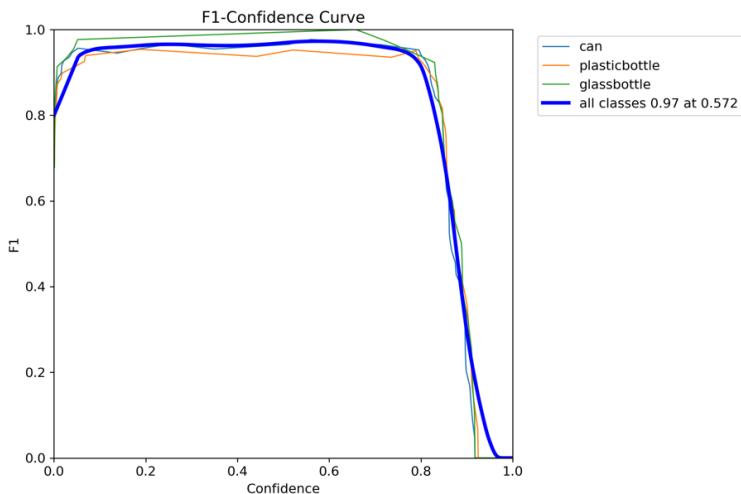
Koko projektin suurin mallikohtainen haaste oli se, ettei yksi YOLO-malli pystynyt hoitamaan sekä ihmistunnistusta että tölkien/pullojen tarkkaa luokittelua ilman, että jompikumpi kärsi. Yhdessä mallissa oli yksinkertaisesti liikaa eri luokkia, ja niiden visuaalinen kirjo oli liian suuri. Tämä johti heikompiin rajauskiin ja virhedetektioita tuli paljon. Ratkaisin tämän jakamalla kokonaisuuden kahteen malliin, jotka on koulutettu eri tarkoituksiin.

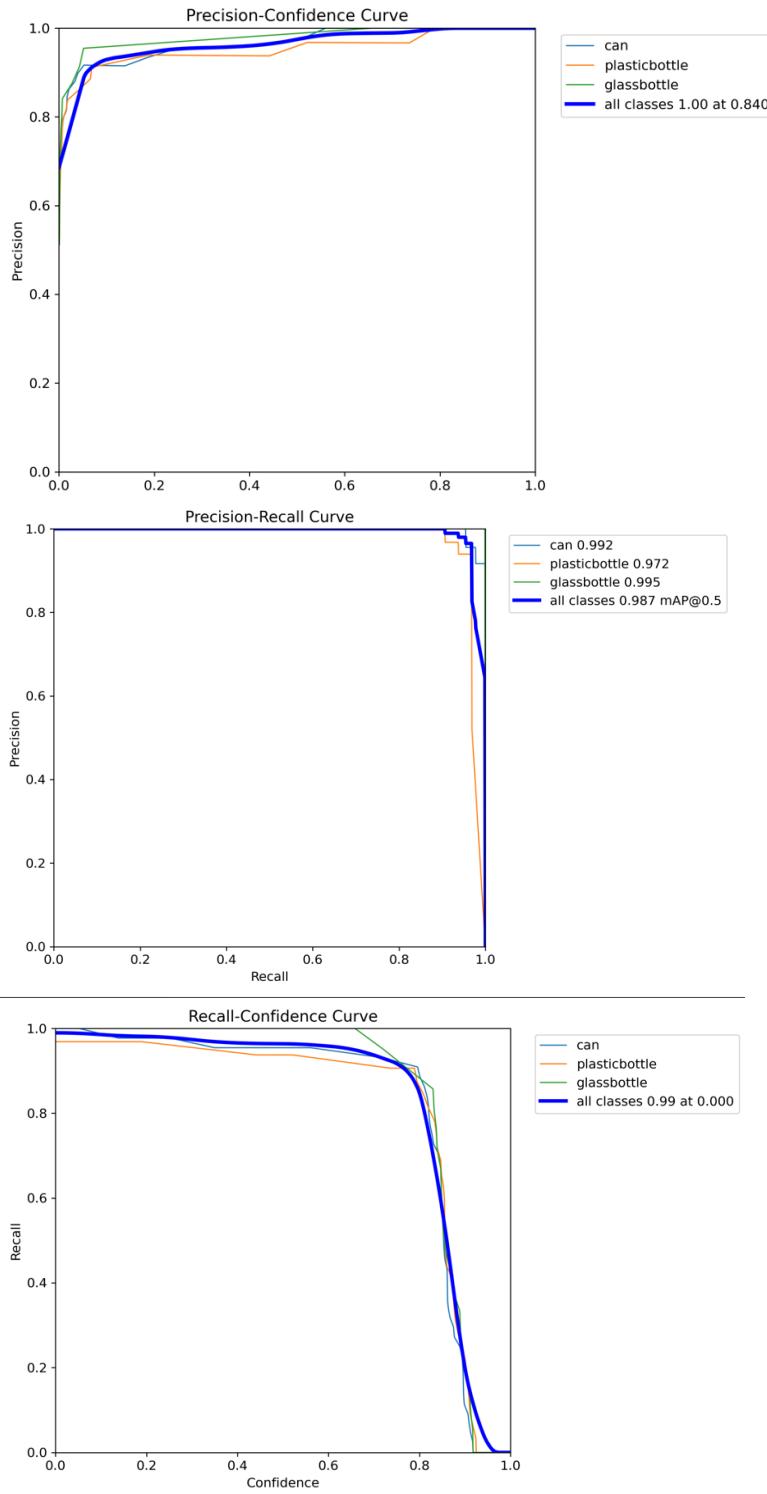
Toinen haaste on lasipullen ja muovipullen erotaminen: Lasi- ja muovipullo näyttävät todella samalta, joten tunnistus on erittäin vaikeaa niiden välillä. Sain korjattua ongelmaa siten että lisäsin kuvia molemmista kategorioista.

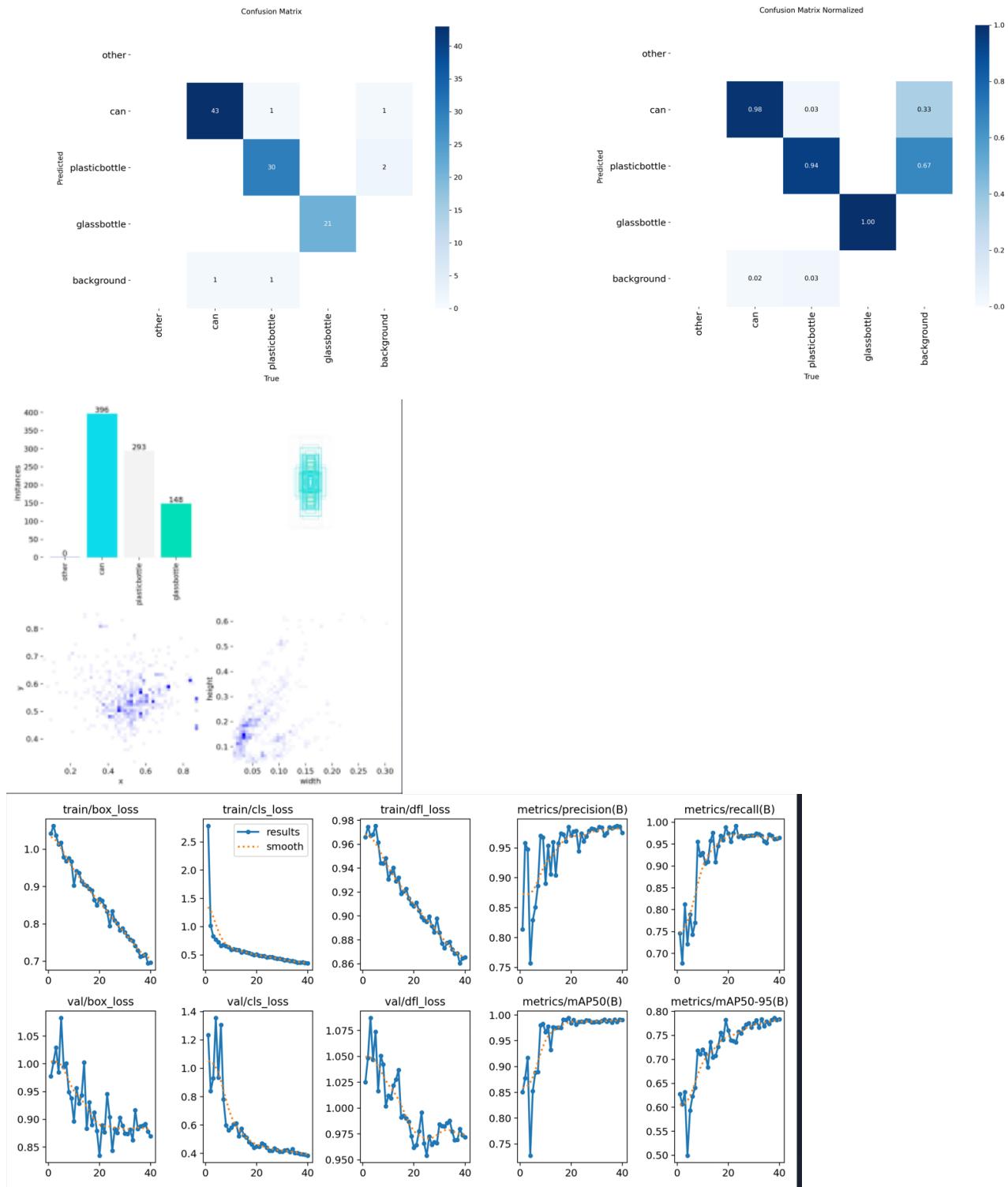
Jatkokehitysmahdollisuksia:

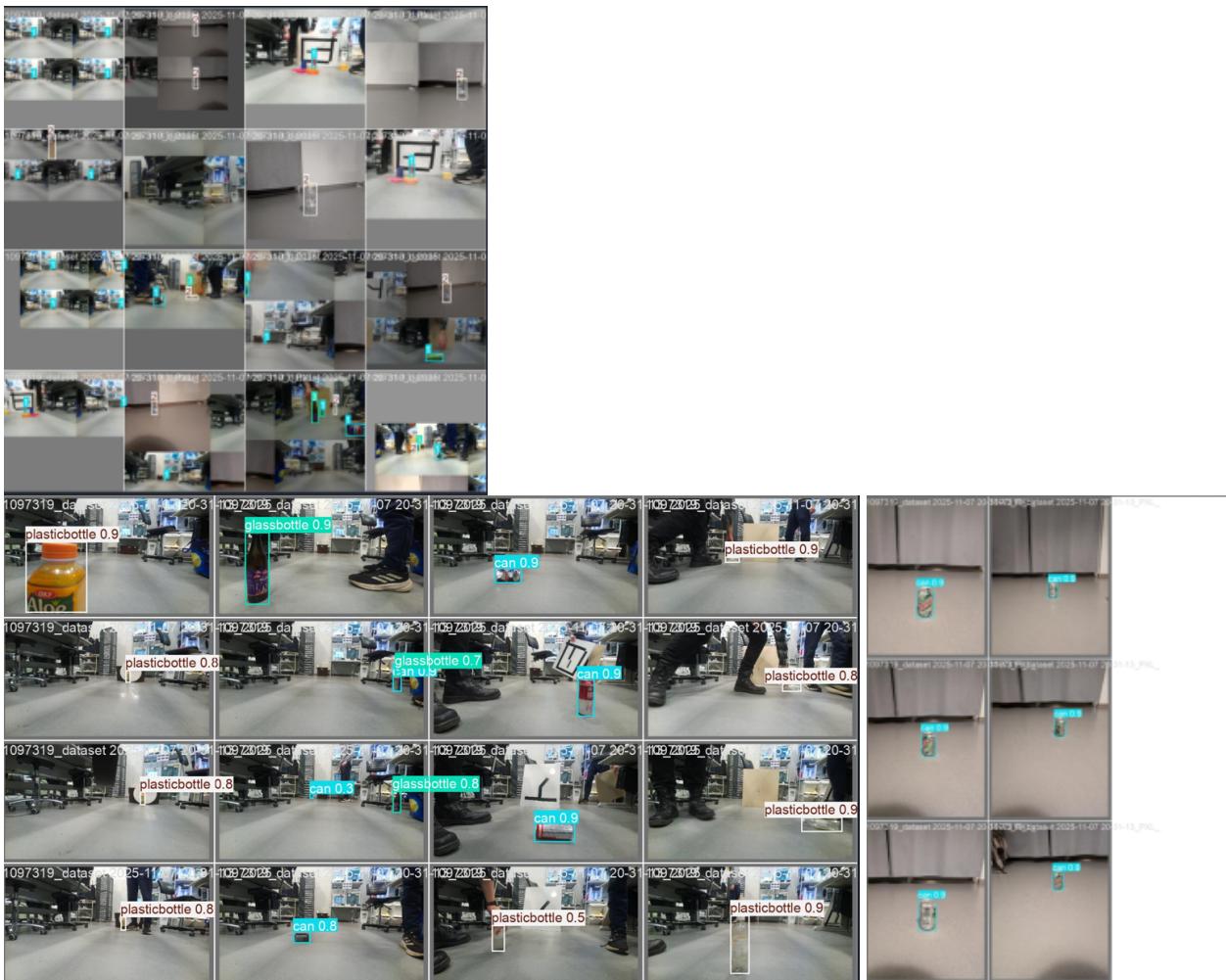
- lisääen datasettiin erilaisia tölkkejä, eri värijä ja eri valaistuksia lisää robustisuutta
- optimoin mallit Raspberry Pi 5:lle (esim. quantization, TensorRT), jolloin FPS kasvaa
- yhdistän mallit tulevaisuudessa, jos datasetti ja mallikoko sen sallivat
- lisääen segmentointimallin, joka erottaa objektiin taustasta entistä tarkemmin parempi poiminta kouralla

Data koulutusprosessista ja tuloksista:









Selitteet datasta:

F1–Confidence Curve

- X-akseli: **confidence-raja** (0–1), eli kuinka varma mallin pitää olla ennen kuin se tekee detektion.
- Y-akseli: **F1-score**, joka yhdistää sekä precisionin että recallin (harmoninen keskiarvo).

Käyrät:

- Erilliset käyrät **can**, **plasticbottle**, **glassbottle** ja paksu sininen "all classes".
- Legendassa: *all classes* 0.97 at 0.572 paras F1 **0.97**, kun confidence-raja on noin **0.57**.

Tulkinta:

- F1 on lähes koko alueella n. 0.9–1.0, eli malli löytää kohteet hyvin, ja väärä havaintoja on vähän.
- Lian korkea confidence (>0.8–0.9) pudottaa F1:n, koska recall romahdattaa (malli on liian "varovainen").

Käytännön johtopäätös:

- Kannattaa käyttää conf-rajaa jossain **0.5–0.6** paikkeilla, silloin tasapaino osumien ja turhien detektoiden välillä on paras.

2) Precision–Confidence Curve

- X: **confidence-raja**
- Y: **precision** = "kuinka moni positiivinen havainto oli oikein".

Legendassa: *all classes* 1.00 at 0.840.

Tulkinta:

- Kun conf-raja kasvaa, precision nousee lähelle **1.0**.
- Korkeilla rajoilla malli ei juurikaan tee väärää positiivisia, kun se piirtää laatikon, se on lähes aina oikeassa.
- Tästä näkee, että malli ei spämmää laatikoita mihin sattuu, vaan on tarkka.

3) Precision–Recall Curve

- X: **recall** (kuinka moni oikea objekti löytyi)
- Y: **precision**

Legendassa:

- can 0.992
- plasticbottle 0.972
- glassbottle 0.995
- all classes 0.987 mAP@0.5

Tulkinta:

- Käyrät menevät lähes kulmikkaana "nelionä" ylhäällä sekä precision että recall ovat lähes koko ajan 0.9–1.0.
- mAP@0.5 **0.987** keskimäärin malli on **erittäin tarkka** kaikille luokille.
- Glassbottle on hitusen paras (0.995), plasticbottle hieman heikompi (0.972), mutta erot ovat pieniä.

4) Recall–Confidence Curve

- X: **confidence-raja**
- Y: **recall**

Legendassa: *all classes 0.99 at 0.000*.

Tulkinta:

- Kun conf = 0 (eli hyväksytään kaikki mahdolliset ehdotukset), recall on lähes **0.99** malli löytää melkein kaikki oikeat objektit.
- Kun conf-raaja nostetaan, recall alkaa hitaasti laskea ja putoaa jyrkästi vasta jossain 0.8–0.9 tienoilla.
- Tämä tukee samaa päätelmää kuin F1-käyrä: noin 0.5–0.6 on hyvä kompromissi, muuten alkaa menettää havaintoja.

– Confusion matrix, datasetin jakauma, oppimiskäyrät ja esimerkkikuvat

1) Confusion Matrix

Vasen ruutu: **raaka confusion matrix**

- Akselit:
 - **True** (todellinen luokka) vaakasuunnassa
 - **Predicted** (mallin ennuste) pystysuunnassa
- Luokat: *other, can, plasticbottle, glassbottle, background*.

Diagonaali:

- can **43** oikein
- plasticbottle **30** oikein
- glassbottle **21** oikein

Off-diagonal (virheet):

- muutama tapaus, joissa:
 - can other
 - plasticbottle background
 - background can / plasticbottle

Tulkinta:

- Lähes kaikki tapaukset ovat diagonaalilla malli sekoittaa luokka **hyvin harvoin**.
- Tyypillisin virhe: pullomainen asia tulkitaan vääräksi pulloluokaksi tai taustaksi, mutta määräät ovat pieniä.

2) Normalized Confusion Matrix

Sama kuin edellinen, mutta **normalisoitu**, eli muutettu prosenteiksi per oikea luokka.

- Esimerkiksi:
 - can-rivillä diagonaali n. **0.98** noin 98 % oikeista can-objekteista tunnistetaan oikein.
 - plasticbottle-diagonaali n. **0.94**.
 - glassbottle-diagonaali **1.00**, eli validointidatassa kaikki lasipullot menivät oikein.

Tulkinta:

- Kaikkien luokkien recall on korkea, lasipulloilla jopa täydellinen (tällä datasetillä).
 - "other" ja "background" eivät juuri sotke mallia, mikä on hyvä merkki käytännön ajoa varten.
-

3) Datasetin jakauma ja bbox-tiheys

Pylväsdiagrammi (Instances)

- Näyttää, montako instanssia kutakin luokkaa datasetissä on.
 - can **396**
 - plasticbottle **293**
 - glassbottle **148**

Tulkinta:

- Luokat eivät ole täysin tasapainossa, mutta mitään ei ole aivan naurettavan vähän.
 - Glassbottle on vähiten edustettu, mikä voi selittää miksi sen luotettavuus voi myöhemmin kärsiä, jos data ei laajene – mutta nyt se silti toimii hyvin.
-

4) Koulutuksen oppimiskäyrät

Ylärivi: train-viivat

- **train/box_loss** – laatikoiden paikkavirhe. Laskee tasaisesti malli oppii paikantamaan tarkemmin.
- **train/cls_loss** – luokitusvirhe. Tosi korkea alussa, laskee jyrkästi ja tasoittuu malli oppii nopeasti, ei enää parane paljon loppupäässä.
- **train/dfl_loss** – “distribution focal loss”, liittyy tarkempaan boxin regressioon; myös laskee tasaisesti.

Ylärivi oikealla: metrics/precision(B) ja metrics/recall(B)

- Precision nousee nopeasti yli 0.95 ja stabiloituu.
- Recall nousee myös lähelle 0.97–0.99 ja pysyy siellä.
- Tämä vastaa ensimmäisen kuvan käyriä malli oppii todella hyvin.

Alariivi: val-viivat + mAP

- **val/box_loss, val/cls_loss, val/dfl_loss** – vastaavat validatiodatalle.
 - Ne eivät laske yhtä siististi, vaan heilahtelevat normaalialla, koska validatiodata on pienempi ja satunnaisempi.
 - Trendi on kuitenkin laskeva, ei selvää overfitting-piikkiä.
- **metrics/mAP50(B)** – mAP @ IoU 0.5. Nousee nopeasti ~0.98 paikkeille ja tasaantuu.
- **metrics/mAP50-95(B)** – tiukempi metriikka. Nousee jonkin 0.8+ tasolle ja jatkaa pientä parantumista.

Tulkinta:

- Malli oppii tasaisesti ilman romahduksia.
- Suurimmat parannukset tulevat ensimmäisillä epochilla, loppupäässä tulee hienosäätöä.

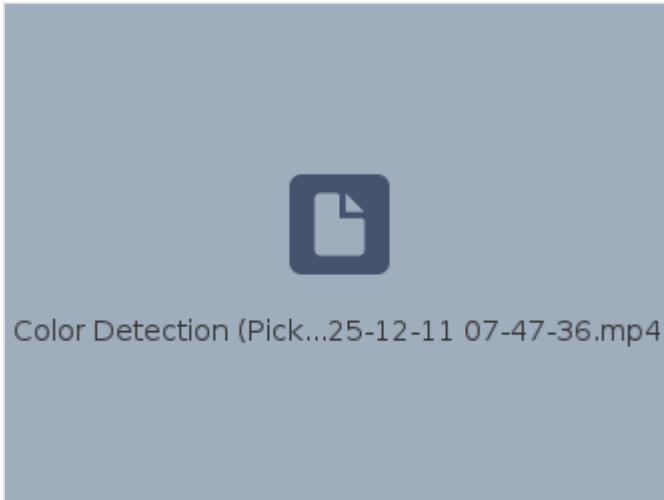
Tunnistusohjelma:

[best.pt](#)

toimii tällä hetkellä vain paikallisesti omalla koneella, mutta best.pt on tosiaan itse malli

Värientunnistusohjelma:

Tölkin/pullontunnistusohjelman rinnalle kehitin myös värientunnistusohjelman. Ohjelman ideana oli alunperin auttaa tölkien tunnistamisessa ja niiden lajittelussa, mutta koska tölkien/pullojen tunnistusohjelmasta tuli sen verran hyvä, niin sille ei ole enää niin suurta tarvetta. Kuitenkin mahdollinen hyötykäyttö ohjelmalle voisi olla erilaisien objektiien lajiteleminen värin perusteella. Hyöty tässä on että vaivaiset 200 riviä koodia voi tehdä merkityksellistä työtä ilman suurta laskentatehoa tai suurta neuroverkkoa.



video ohjelman käytöstä:

Ohjelma: [color_detect.py](#)

Tietolähteet ohjelman kehityksessä:

<https://agneya.medium.com/color-detection-using-python-and-opencv-8305c29d4a42>

<https://www.ultralytics.com/blog/leveraging-color-detection-in-computer-vision-applications>

4. Projektin tulokset – mitä kolmen kuukauden projektin aikana saatiin aikaan

- Toimiva runko, jossa on renkaat, joilla robotti liikkuu sujuvasti eteen- ja taaksepäin sekä kääntyy, sekä tavaratala, joka mahdollistaa kerättyjen tavaroiden nopean tyhjentämisen.
- Toimiva teleoperate yhteys etänä web-sovelluksen kautta.
- Toimiva kamerakuvan välitys raspberrytsta takaisin sovellukseen helpomman etäohjeemisen mahdollistamiseksi.
- Toimiva kappaletunnistusohjelma helpottamaan operointia tilanteissa, kun raspberry kameran kuvanalaatu on huono tai jatkokehitystilanteessa autonomista esineenkeräystä varten.
- Robottia varten on luotu kätevä web-sovellus sen ohjausta ja käyttöä varten.

5. Mahdollinen projektin jatkokehitys ja kestävä kehitys

Projekti nykyinen lopputulos luo vahvan perustan monipuolisen ja skaalautuvan robottijärjestelmän jatkokehitykselle. Tulevaisuudessa robotin kehittämistä on mahdollista laajentaa merkittävästi sekä teknisesti että toiminnallisesti, erityisesti autonomian, ympäristötiedon keruun ja kestävän kehityksen näkökulmasta.

Teknologiset jatkokehitykset:

- Koodin turvallisuuden parantaminen sekä luotettavan toiminnan varmistamiseksi että sen estämiseksi, että ulkopuoliset eivät voi kytkeytyä robotin ohjaukseen
- signaalinsiirtojärjestelmän jatkokehitys – tällä hetkellä robottia ohjataan Wi-Fi:n kautta, mutta laajempaa käyttöä varten se tarvitsee pidemmän ohjausetäisyyden, esimerkiksi radiotaajuuskien tai verkon kautta hyödyntäen lähipiä tukiasemia

Autonominen toiminta ja esineiden tunnistus

Yksi keskeisimmistä jatkokehityssuunnista on robotin autonomian lisääminen. Nykyisessä muodossaan robotti toimii etäohjattuna, mutta seuraava looginen askel on robottiin integroitava autonomisen toimintamallin. Tämä voidaan toteuttaa hyödyntämällä konenäköä ja tekoälypohjaista jo olemassa olevaa objektintunnistusta, jonka avulla robotti kykee itsenäisesti havaitsemaan ympäristössään olevia tölkkejä ja pulloja.

Tunnistuksen perusteella robotti voisi:

- paikantaa ympäristössä olevat kerättävät esineet
- navigoida itsenäisesti esineiden luo
- tarttua niihin robottikäden avulla
- luokitella esineet esimerkiksi materiaalin perusteella (alumiinitölkki, muovipullo, lasipullo)

Autonominen lajittelu ja kuljetus kierrätyspisteelle

Autonomian syventämisen myötä robotti voisi toimia itsenäisenä keräyslaitteena, joka ei ainoastaan kerää tölkkejä ja pulloja, vaan myös kuljettaa ne ennalta määritellyihin kierrätyspisteisiin. Tämä vaatisi robotiltä:

- karttatiemon hyödyntämistä (paikannus ja reitti-suunnittelu)
- esteiden tunnistusta ja väistämistä kaupunkiympäristössä
- kykyä muistaa ja tunnistaa kierrätyspisteiden sijainnit

Tällainen toiminta mahdollistaisi robotin käytön esimerkiksi puistoissa, kampusalueilla tai kaupunkien julkisilla alueilla, joissa roskaantuminen on yleistä.

Ympäristödatan keruu ja analysointi

Robotti voidaan kehittää myös liikkuvaksi mittausalustaksi, joka kerää arvokasta dataa kaupunkiympäristöstä. Robottiin voidaan liittää erilaisia antureita, joiden avulla se mittaa muun muassa:

- tien tai kulkualustan tasaisuutta ja tärinää
- ilman ja maan kosteutta
- lämpötilaa
- kerättyjen tölkien ja pullojen lukumäärää ajan kuluessa

Kerätyn datan avulla voidaan:

- arvioida infrastruktuurin kuntoa
- seurata ympäristöolo-suhteiden muutoksia
- tuottaa tilastotietoa kierrätyksen tehokkuudesta eri alueilla

Tämä tekee robotista paitsi kerääjän, myös tiedonkeruulaiteen, jota kunnat ja kaupungit voisivat hyödyntää päättöksenteossa.

Energiaratkaisut ja itse latautuminen

Jatkokehityksessä robotin käyttömukavuutta ja itsenäisyyttä voidaan parantaa lisäämällä kyky itsenäiseen latautumiseen. Robotti voisi esimerkiksi:

- seurata omaa akkunsa varauastasoa
- keskeyttää tehtävänsä ajoissa
- siirtyä automaattisesti latausasemalle
- jatkaa toimintaansa latauksen jälkeen ilman ihmisen puuttumista

Tämä on keskeinen ominaisuus pidempiaikaista autonomista käyttöä ajatellen.

Kestävä kehitys ja kaupunkiympäristön siisteyks

Projektiin jatkokehitys tukee vahvasti kestävän kehityksen periaatteita. Autonominen keräysrobotti vähentää roskaantumista, tehostaa kierrätystä ja voi pitkällä aikavälillä vähentää manuaalisen siivoustyön tarvetta. Samalla robotti edistää resurssien tehokasta käyttöä ja ympäristötietoisuutta kaupunkiympäristössä.

Laajennus kohti kaupunki- ja palvelurobottia

Robotti voidaan jatkokehittää osaksi monirobottijärjestelmää samaan tapaan kuin nykyiset ruokakuljetusrobotit, joissa useat samanlaiset autonomiset robotit toimivat yhtä aikaa kaupunkiympäristössä. Tällöin tässä projektissa kehitetty robotti voisi toimia yhtenä yksikkönä useamman robotin joukossa, erikoistuen tölkien ja pullojen keräämiseen sekä ympäristödatan mittauaiseen, mikä mahdollistaisi laajempien alueiden tehokkaan ja skaalautuvan ylläpidon.

Tällöin robotti ei olisi enää vain yksittäinen laite, vaan osa laajempaa kaupunkiympäristön digitalisaatiota ja automaatiota edistävää järjestelmää.

Tekoälyllä luotu havainnekuva robottin tulevasta käyttötarkoituksesta:



6. Mitä projektin aikana opittiin + kuvat projektin aikana

Työtä on tehty paljon. Projektin aikana kartutimme taitoja 3D-mallinnuksessa, ohjelmoinnissa, sähkökytkentöjen järkevässä ja turvallisessa toteutuksessa, Linuxin käytössä sekä WiFi-yhteyksien hyödyntämisessä. Tämä oli mielenkiintoinen kokemus ja tärkeä pohja tulevia projekteja varten.

Mitä opimme ja mitä voisimme suositella muille:

- **Komponenttien huolellinen suunnittelu on tärkeää:** kannattaa valita kaikki komponentit ja varmistaa niiden toimivuus ennen kuin projekt siirtyy aktiiviseen toteutusvaiheeseen. Meidän tapauksessamme jouduimme esimerkiksi vaihtamaan servomoottoreista DC-moottoreihin, koska servot eivät soveltuneet ajamiseen: kiinnitykset eivät olleet tarpeeksi luotettavia.



- **Koodin luotettavuus ja yksinkertaisuus ovat erittäin tärkeitä robottien rakentamisessa.** Järjestelmän tai mikrokontrollerin ylikuormittaminen ei ole hyvä asia.

- Kannattaa hyödyntää ja kehittää valmiita ratkaisuja, jos tavoitteena on saada robotti toimimaan eikä rakentaa kaikkea täysin alusta asti itse. Internetistä ja GitHubista löytyy paljon hyviä muiden opiskelijoiden ja harrastajien projekteja, ja niiden ideoita sekä toteutustapoja kannattaa käyttää ja soveltaa omaan työhön omalla näkökulmalla – se voi nopeuttaa prosessia huomattavasti, ja te voitte itse kehittää valmiita ratkaisuja ja tuoda niihin oman panoksenne.

Kuvat projektin aikana

