Robottikäsi – projektisuunnitelma 25.02.2021

1. **Henkilötiedot**

   Samuli Karvinen, 602945, Biosensing and Bioelectronics, 1. vuosi.

2. **Yleiskuvaus ja vaikeustaso**

   Tavoitteenani on toteuttaa ohjelma, joka simuloi kaksinivelistä robottikättä kaksiuloitteisessa ympäristössä, jossa sillä voidaan siirtää läheisyydessään olevaa esinettä. Olen ajatellut toteuttaa projektin vaativalla vaikeustasolla vaatimuksena:

   - Ohjelmalla on graafinen käyttöliittymä.
   - Molempia nivelkulmia voidaan säätää manuaalisesti liukusäätimillä (suora kinematiikka).
   - Robotti ja esine voidaan palauttaa alkuperäisiin sijanteihin.
   - Robottikäden liikkuminen on animoitu.
   - Robottikädellä voi tarttua lähellä olevaan kappaleeseen ja siirtää sitä sekä päästää siitä irti (ilman kappaleen fysiikkaa).
   - Yksikkötestit edes jollekin osalle ohjelmaa, kuten käänteiselle kinematiikalle ↴
   - Ohjelma osaa siirtää robottikäden käyttäjän osoittamaan pisteeseen tai mahdollisimman lähelle sitä (käänteinen kinematiikka).

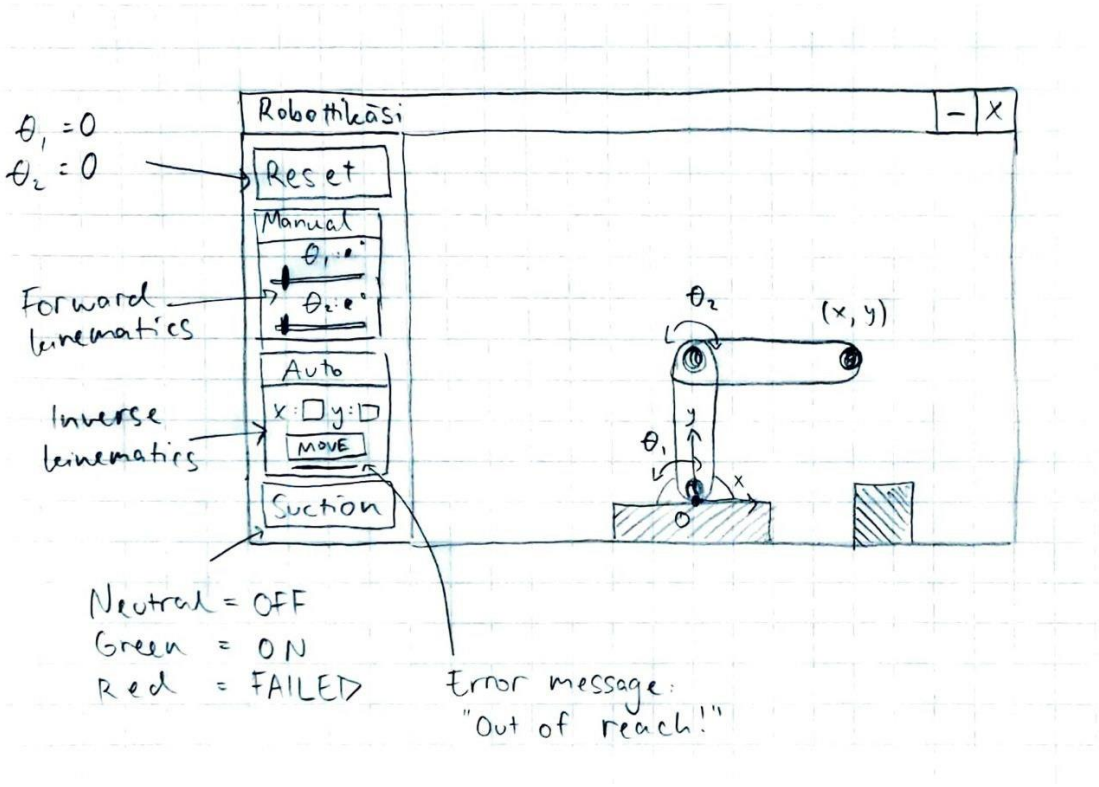
3. **Käyttötapauskuvaus ja käyttöliittymän luonnos**

   Kuvaus: Kappaleen siirtäminen robottikäden avulla kurssiassistentin toimesta manuaalisesti
   Käyttäjä: Kurssiassistentti
   Tavoite: Kurssiassistentti tarttuu robottikäden avulla lähellä olevaan kappaleeseen, siirtää sitä haluamaansa paikkaan ja päästää irti
   Laukaisija: Kurssiassistentin halu kokeilla robottikäden toimintaa
   Esiehto: Kurssiassistentilla on pääsy kyseiseen projektiin
   Jälkiehto: Robottikäsi tekee kuten kurssiassistentti haluaa
   Käyttötapauksenkulku:

   1. Kurssiassistentti käynnistää robottikäsi-ohjelman, jolloin ikkuna käyttöliittymineen avautuu (katso kuva 1).
   2. Kyseisessä ikkunassa kurssiassistentti voi manuaalisesti ohjata robottikäden molempia nivelkulmia liukusäätimillä, jolloin säätimen yläpuolella olevan kulman numero muuttuu. Nivelkulmista saadaan suoran kinematiikan avulla toisen nivelen sekä robottikäden päädyssä olevan ’end-effectorin’ xy-koordinaatit, joiden avulla robotin asento päivitetään ikkunassa. Tässä siis liukusäädintä muuttamalla lähetetään uusi haluttu nivelkulma ohjelman osioon, joka muuttaa nivelkulmat koordinaateiksi, jotka puolestaan lähetetään koordinaatistoa ylläpitävään osioon ja käsketään tämän jälkeen käyttöliittymää päivittymään.
   3. Saavutettuaan ’end-effectorilla’ vieressä olevan kappaleen, kurssiassistentti painaa ’Suction’ näppäintä, jolloin robottikäden imu menee päälle ja käsi tarrautuu kappaleeseen kiinni. Tällöin kappaleen koordinaatit korvataan ’end-effectorin’ koordinaateilla. Tällöin myös Suction-nappi muuttuu neutraalista väristä vihreäksi. Jos kurssiassistentti painaa Suction-näppäintä, kun robottikäsi ei ole kappaleen kohdalla, nappi muuttuu punaiseksi (joka voidaan neutralisoida painamalla nappia uudelleen)
   4. Kurssiassistentti voi liikuttaa kappaletta kappaleen haluamaansa paikkaan ohjaamalla robottikäden nivelkulmia.
   5. Kun haluttu paikka on saavutettu, kurssiassistentti painaa uudelleen Suction-nappia, jolloin imu lähtee pois päästäen kappaleesta irti ja nappi muuttuu takaisin neutraalin

väriseksi. Tällöin kappaleen koordinaatiksi jää voimaan ’end-effectorin’ koordinaatit juuri ennen painallusta.

6. (Kurssiassistentti voi palauttaa robottikäden ja kappaleen alkuasentoon/-sijaintiin painamalla Reset-näppäintä).

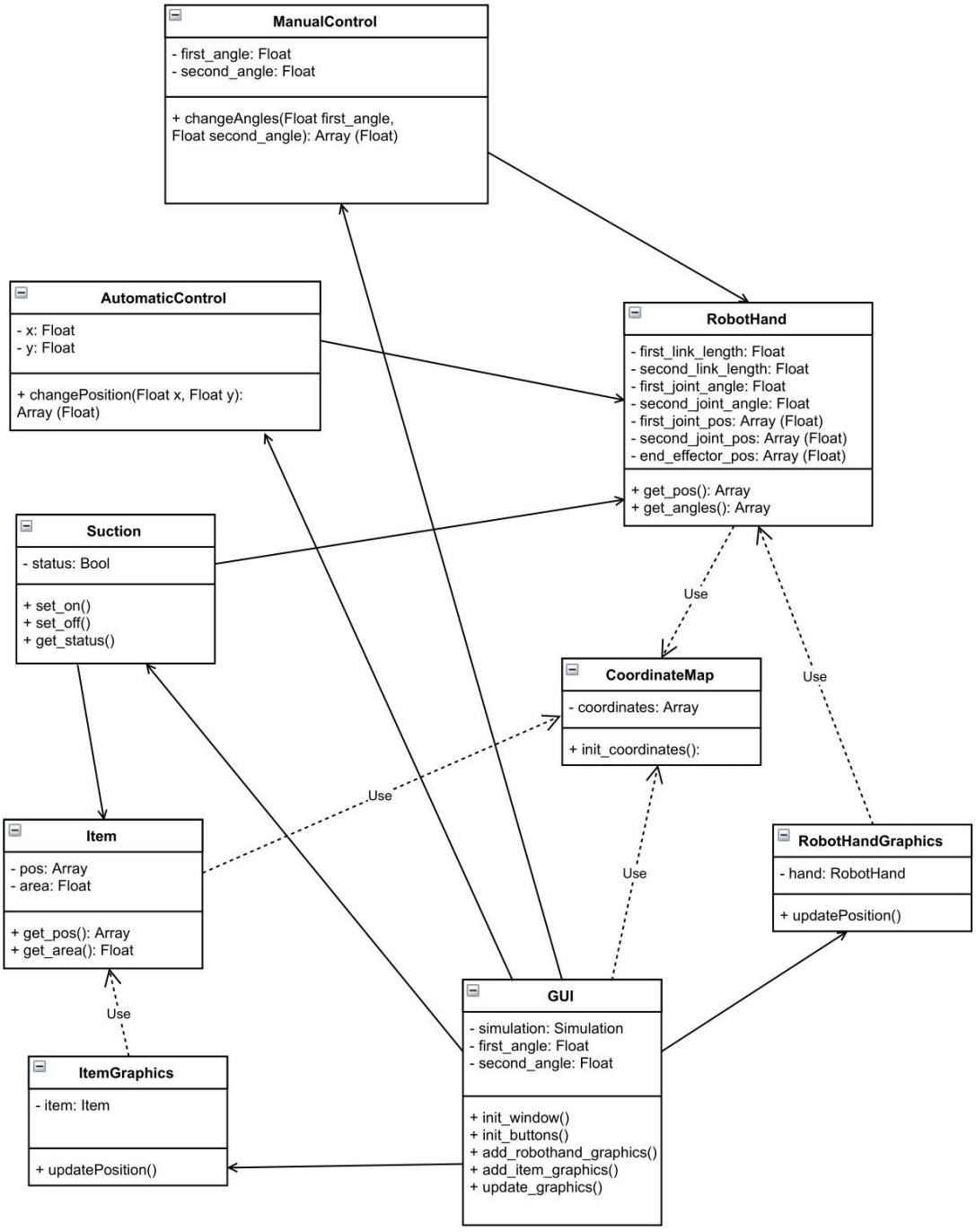


Kuva 1. Käyttöliittymän luonnos, jossa on esitetty kaikki ohjaukseen tarvittavat moduulit sekä graafinen esitys robottikädestä ja sen poimittavasta esineestä (Lähde: Samuli Karvisen kotiarkisto).

4. Ohjelman rakennesuunnitelma

Minulla ei ole ollenkaan aikaisempaa kokemusta rakennesuunnittelussa, mutta yritin parhaani. Päätin ottaa mallia viikon 3 harjoittelutehtävästä, jossa eroteltiin graafiset luokat erikseen (Katso kuva 2). Ajatuksena siis olisi, että ohjelma rakentuu käyttöliittymäikkunaan, johon lisätään napit ja osio koordinaatistolle, mihin puolestaan lisätään robottikäden ja esineen grafiikat. Nappeja käyttämällä aktivoituu Suction, ManualControl ja AutomaticControl, jotka puolestaan muokkaavat RobotHandin ja Itemin ominaisuuksia. Muutokset tallentuvat niihin, joiden graafiset luokat puolestaan lukevat muutokset ja päivittävät niiden perusteella omat graafiset ominaisuutensa. Lopuksi käyttöliittymä päivittää koordinaatiston grafiikat, jolloin muutokset tulevat käyttäjälle näkyviin. En ole varma onko nuolet oikealla tavalla merkitty, mutta toivottavasti idea edes tulee jotenkin julki.

Suurin ongelmani on koordinaatiston muodostaminen. Minulla on idea, miten koordinaatisto rakennetaan kokonaisluvuilla, mutta miten se tehdään Floattina? Animoidut liikkeet näyttäisivät silloin sulavammilta eikä tökkisi jokaisen kokonaisluvun välillä. Jos siis tuo Float ei onnistu niin sitten kaikki laskuista tulevat koordinaatit täytyy pyöristää kokonaisluvuiksi.



Kuva 2. Alustava suunnitelma UML-kaaviossa.

5. **Tietorakenteet**

   Listat soveltuvat parhaiten kaksiulotteisen koordinaatiston kartoittamiseen ja NumPyn array-tietotyyppi soveltuu hyvin koordinaattien tallennukseen. Muita tietorakenteita ei ymmärtääkseni tarvita.

6. **Tiedostot ja tiedostoformaatit**

   Ohjelma ei käsittele tiedostoja, sillä kaikki tarvittava tieto robottikäden ohjaukselle tulee käyttäjältä.

7. **Algoritmit**

   • Suoralla kinematiikalla (*Forward Kinematics*) voidaan robottikäden nivelkulmien avulla ratkaista robottikäden xy-koordinaatit (napakoordinaatisto → karteesiset koordinaatit). Tämän avulla ohjelma voi laskea robottikäden päivittyvän sijainnin käyttäjän antamista nivelkulmista. Käytän tässä geometrista lähestymistapaa algebran sijaan, sillä se on itselle intuitiivisempi:

   $$x_1 = L_1 \cos\theta_1 \qquad (1)$$

   $$y_1 = L_1 \sin\theta_1 \qquad (2)$$

   missä (x_1,y_1) on toisen nivelen koordinaatti, L_1 on ensimmäisen varren pituus ja θ_1 on ensimmäinen nivelkulma.

   $$x_2 = x_1 + L_2 \cos(\theta_1 + \theta_2) \qquad (3)$$

   $$y_2 = y_1 + L_2 \sin(\theta_1 + \theta_2) \qquad (4)$$

   missä (x_2,y_2) on 'end-effectorin' koordinaatti, L_2 on toisen varren pituus ja θ_2 on toinen nivelkulma.

   • Käänteisellä kinematiikalla (*Inverse Kinematics*) voidaan robottikäden xy-koordinaattien avulla ratkaista robottikäden nivelkulmat (karteesiset koordinaatit → napakoordinaatisto). Tämän avulla robottikäsi kykenee liikkumaan käyttäjän antamaan koordinaattiin. Käytän tässä geometrista lähestymistapaa algebran sijaan, sillä se on itselle intuitiivisempi:

   $$\theta_1 = \tan^{-1}\frac{y_2}{x_2} + \cos^{-1}\left(\frac{L_1^2 - L_2^2 + x_2^2 + y_2^2}{2L_1\left(\sqrt{x_2^2 + y_2^2}\right)}\right) \qquad (5)$$

   $$\theta_2 = \cos^{-1}\left(\frac{L_1^2 + L_2^2 - x_2^2 - y_2^2}{2L_1L_2}\right) - 180° \qquad (6)$$

   • Lineaarisella interpolaatiolla mahdollistetaan robottikäden suora vaiheittainen liike paikasta $v_{initial}$ paikkaan v_{final}:

   $$v(s) = (1 - s)v_{initial} + sv_{final} \qquad s \in [0,1] \qquad (7)$$

   missä v on sijaintivektori (x, y) ja s on skalaari, joka muuttuu nollasta yhdeksi halutulla tarkkuudella.

(1-6): https://www.atlantis-press.com/proceedings/icovet-18/55913312

(7): https://robotacademy.net.au/masterclass/paths-and-trajectories/?lesson=114

8. **Testaussuunnitelma**

- Reset;
    - Palauttaa kaikki arvot sekä käyttöliittymän oletusarvoihin. Tarkastetaan muuttujien arvoista sekä graafisesta liittymästä, tekeekö ohjelma näin.
- Manuaalinen ohjaus;
    - Liukusäätimiä säätämällä robottikäden tulisi liikkua reaaliajassa sekä nivelkulmien muutos on vastattava oikeita koordinaatteja. Tässä voidaan yksikkötestata suoraa kinematiikkaa.
- Automaattinen ohjaus;
    - Syöttö sallii ainoastaan numeroita, jos syötetään jotakin muuta, kuten kirjaimia, ohjelma esittää virheilmoituksen. Metodia testataan syöttämällä virheellisiä syötteitä.
    - Koordinaatit eivät saa mennä robottikäden ulottumattomiin, jolloin ohjelma esittää virheilmoituksen. Tätä voidaan testata yksikkötestauksella
    - Oikean sijainnin syötettäessä robotin 'end-effectorin' kuuluisi liikkua suoraviivaisesti syötettyyn pisteeseen. Tässä siis yksikkötestataan käänteistä kinematiikkaa sekä lineaarista interpolaatiota.
- Imu;
    - Napin väri-indikaattori on toimittava ja imun onnistuttava. Nappi täytyy olla neutraalin värinen aina kun imu ei ole käytössä, vihreä onnistuneessa imussa ja punainen virheellisessä imussa. Vihreä valo syttyy, jos käsi on esineen kohdalla ja punainen jos näin ei ole. Tätä voidaan testata kokeilemalla nappeja eri tilanteissa.
- Koordinaatisto;
    - Koordinaatit eivät saa mennä koordinaatiston ulkopuolelle.

9. **Kirjastot ja muut työkalut**

- Math; algoritmien laskemiseen
- PyQt5; graafinen käyttöliittymä
- NumPy; vektoreiden laskemiseen (sekä todennäköisesti koordinaatiston hallintaan)

10. **Aikataulu**

- Viikko 10: ManualControl-luokan koodaaminen ja testaus ilman käyttöliittymää
- Viikko 11: AutomaticControl-luokan koodaaminen ja testaus ilman käyttöliittymää
- Viikko 12: CoordinateMap-, RobotHand- ja Item-luokan koodaaminen ja testaus ilman käyttöliittymää
- Viikko 13: PyQt5 tarkemmin tutustuminen ja sen kanssa harjoittelu
- Viikko 14: GUI:n ikkunan ja nappien koodaaminen sekä testaus + Suction-luokan koodaaminen ja testaaminen
- Viikko 15: RobotGraphics ja ItemGraphics koodaaminen, testaus sekä yhdistäminen GUI:n kanssa
- Viikko 16: Kaikkien osien yhdistäminen ja testaus
- Viikko 17: Lisää testausta ja dokumentaatiota
- Viikko 18: Dokumentointi

11. **Kirjallisuusviitteet ja linkit**

- https://robotacademy.net.au/
- https://www.atlantis-press.com/proceedings/icovet-18/55913312
- https://stackoverflow.com/questions/9082829/creating-2d-coordinates-map-in-python
- https://www.tutorialspoint.com/pyqt/index.htm
- https://numpy.org/devdocs/user/absolute_beginners.html