1. **Henkilötiedot**

   Robottikäsi; Samuli Karvinen, 602945, Biosensing and Bioelectronics, 28.04.2021.

2. **Yleiskuvaus**

   Ohjelma simuloi kaksinivelistä robottikättä kaksiuloitteisessa ympäristössä, jossa sillä voidaan siirtää läheisyydessään olevaa laatikkoa. Projekti on toteutettu vaativalla vaikeustasolla:

   - Ohjelmalla on graafinen käyttöliittymä.
   - Molempia nivelkulmia voidaan säätää manuaalisesti liukusäätimillä
   - Robottikäden liikkuminen on animoitu.
   - Robottikädellä voi tarttua lähellä olevaan laatikkoon, siirtää sitä sekä päästää siitä irti (ottamatta huomioon laatikon fysiikkaa).
   - Yksikkötestit toteutettu suoralle ja käänteiselle kinematiikalle.
   - Ohjelma osaa siirtää robottikäden käyttäjän osoittamaan pisteeseen tai mahdollisimman lähelle sitä.

3. **Käyttöohje**

   Ohjelma käynnistyy suorittamalla window.py tiedosto, jolloin käyttäjälle aukeaa graafinen käyttöliittymä, mikä sisältää robottikäden (Robot) laatikoineen (Square) sekä robottikäden ohjauspaneelin ikkunan vasemmassa laidassa. Ohjauspaneelista löytyy manuaalinen ohjaus (Manual Control), automaattinen ohjaus (Auto Control) sekä robottikäden työkalupisteen (end-effector) imu (Suction).

   Manuaalisesta ohjauksesta löytyy liukusäätimet ensimmäiselle nivelelle (1. joint) sekä toiselle nivelelle (2. joint), joita voidaan liikuttaa -360° ja +360° välillä. Ensimmäinen nivel viittaa robottikäden ”olkapäähän” ja toinen nivel ”kyynärpäähän”.

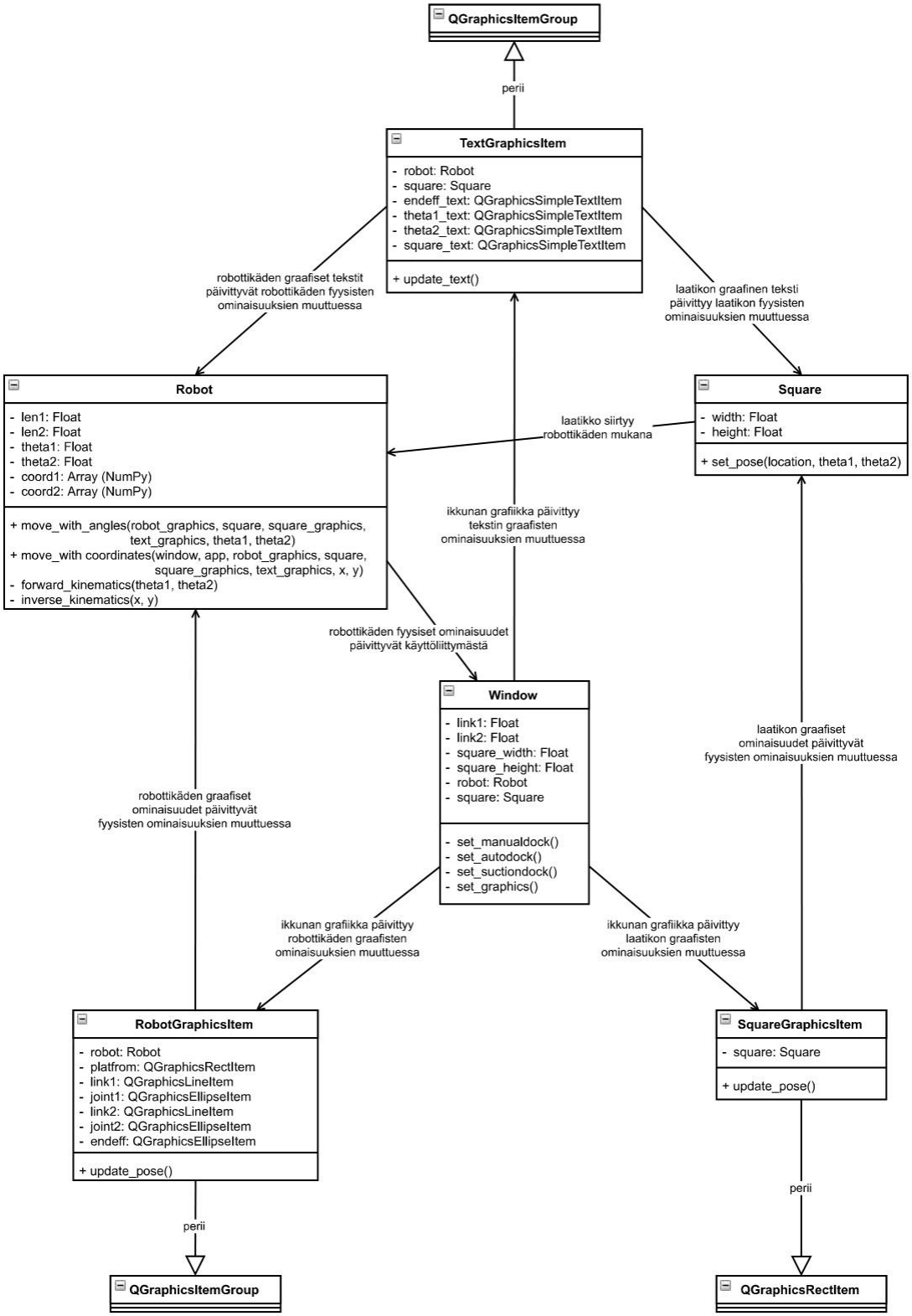
   Automaattisesta ohjauksesta löytyy syötteet x- ja y-koordinaateille, joille käyttäjä voi kirjoittaa sijainnin mihin robottikäden työkalupisteen halutaan siirtyvän. Siirtyminen tapahtuu klikkaamalla hiirellä nappia ’Move’, jolloin robotti liikkuu suoraviivaisesti kohteeseen lineaarisen interpolaation avulla.

   Robottikäden työkalupisteen imu voidaan aktivoida klikkaamalla hiirellä nappia ’Suck’, joka ilmoittaa imun onnistumisesta väreillä: Imu onnistuu, jos työkalupiste on laatikon päällä värjäten napin vihreäksi; Imu epäonnistuu, jos työkalupiste ei ole laatikon päällä värjäten napin punaiseksi; Imun saa pois päältä painamalla nappia uudestaan, jolloin sen väri muuttuu neutraalin harmaaksi.

4. **Ulkoiset kirjastot**

   - PyQt5; Graafinen käyttöliittymä sekä graafiset esineet.
   - NumPy; array-funktio koordinaattien tallentamiseen sekä manipulointiin; arange-funktio lineaariseen interpolaatioon ja round-funktio float numeroiden pyöristämiseen estäen epätarkkuuksien tuomia virheitä.
   - Math; trigonometrian laskemiseen käytettäviä funktioita, kuten sqrt(), acos() ja atan2().

## 5. Ohjelman rakenne




Kuva 1. Robottikäden UML-kaavio.

Luokat ja niiden tärkeimmät funktiot:

- Robot:
  - move_with_angles; päivittää Robot-luokan ominaisuudet annettujen nivelkulmien perusteella hyödyntäen omaa forward_kinematics funktiota ja käskee RobotGraphicsItemin sekä SquareGraphicsItemin päivittämään itsensä Robot- ja Square-luokkien mukaisesti. Päivittää myös Square luokan ominaisuudet, jos Suction on päällä.
  - move_with_coordinates; päivittää Robot-luokan ominaisuuksia annettujen koordinaattien perusteella for-loopissa vaiheittain lineaarisella interpolaatiolla, kunnes annettu koordinaatti on saavutettu. Funktio hyödyntää inverse_kinematics funktiota. Päivittää myös Square luokan ominaisuudet, jos Suction on päällä. Päivityksen jälkeen funktio käskee RobotGraphicsItemin sekä SquareGraphicsItemin päivittämään itsensä Robot- ja Square-luokkien mukaisesti.
- Square:
  - set_pose; päivittää laatikon sijainnin ja asennon Robot-luokan työkalupisteen mukaisesti, jos Suction on päällä.
- Window:
  - set_manualdock; asettaa manuaalisen ohjauksen telakan nappeine ja ominaisuuksineen ikkunaan.
  - set_autodock; asettaa automaattisen ohjauksen telakan nappeine ja ominaisuuksineen ikkunaan.
  - set_suctiondock; asettaa imunohjauksen telakan nappeine ja ominaisuuksineen ikkunaan.
  - set_graphics; asettaa pääikkunan keskiöön QGraphicsView, johon lisätään QGraphicsScene ja siihen lisätään SquareGraphicsItem, RobotGraphicsItem sekä TextGraphicsItem
- SquareGraphicsItem:
  - update_pose; päivittää laatikon graafiset ominaisuudet Square-luokan mukaisesti.
- RobotGraphicsItem:
  - update_pose; päivittää robottikäden graafiset ominaisuudet Robot-luokan mukaisesti.
- TextGraphicsItem:
  - update_text; päivittää laatikon ja robottikäden graafiset tekstit Square- ja Robot-luokan mukaisesti.

## 6. Algoritmit

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

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

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

missä ($x_1$,$y_1$) on toisen nivelen koordinaatti, $L_1$ on ensimmäisen varren pituus ja $\theta_1$ on ensimmäinen nivelkulma.

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

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

missä ($x_2$,$y_2$) on työkalupisteen koordinaatti, $L_2$ on toisen varren pituus ja $\theta_2$ 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 (käänteisen kinematiikan jälkeen tarvitaan graafiselle käyttöliittymälle vielä suoraa kinematiikkaa, sillä käden sijainti piirretään käyttäen koordinaatteja kulmien sijaan). Käytän tässä geometrista lähestymistapaa algebran sijaan, sillä se on itselle intuitiivisempi:

$$\theta_2 = \pm\text{acos}(\frac{x_2^2+y_2^2-L_1^2-L_2^2}{2*L_1*L_2}) \quad (5)$$

$$\theta_1 = atan2(y_2, x_2) - atan2(L_2 * \sin(\theta_2),\ L_1 + L_2 * \cos(\theta_2)) \quad (6)$$

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

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

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

7. **Tietorakenteet**

NumPy:n array-tietotyyppi soveltuu hyvin koordinaattien tallennukseen. Float-tietotyyppiä tarvitaan nivelkulmien ja koordinaattien tallentamisessa.

8. **Tiedostot**

Ohjelma ei käsittele mitään tiedostoja.

9. **Testaus**

Ohjelman graafista käyttöliittymää testattiin ajamalla ohjelmaa toistuvasti ja katsomalla tekeekö se halutut toimenpiteet. Suunnitelmasta manuaalinen ohjaus testattiin katsomalla visuaalisesti, piirtyykö liukukytkimien muutos ikkunaan ja antaako se halutut arvot robotin kulmille sekä koordinaateille.

Automaattisessa ohjauksessa syöttöä testattiin kokeilemalla, mitä arvoja ohjelma antaa siihen laittaa ja nykyisin se salli ainoastaan negatiivisia sekä positiivisia kokonaislukuja. Liian kaukana olevat koordinaatit testattiin kokeilemalla, milloin ohjelma antaa virheilmoituksen. Lineaarista interpolaatiota testattiin tarkastelemalla liikkeen suoraviivaisuutta visuaalisesti.

Imua testattiin painamalla 'Suck'-nappia erilaisissa tilanteissa, joissa sen väri muuttui riippuen, onnistuiko imu vai ei.

Ohjelman kinematiikkaa testattiin kinematics_test.py ohjelmassa yksikkötestauksella, jossa testattiin Robot-luokan funktioita *forward_kinematics* sekä *inverse_kinematics*. testForwardKinematics-testissä katsotaan osaako *forward_kinematics* muuttaa robotin nivelkulmat koordinaateiksi sekä testInverseKinematics-testissä katsotaan osaako *inverse_kinematics* muuttaa robotille annetut koordinaatit nivelkulmiksi. Molemmissa testeissä on vain kaksi tilannetta johtuen ajan rajallisuudesta. Molemmat testit menivät läpi.

10. **Ohjelman tunnetut puutteet ja viat**

Jos linkit ovat eripituisia, ohjelma kaatuu automaattisessa ohjauksessa liikeradan mennessä läheltä alustaa (*platform*). Tämä johtuu siitä, että liike ei ole robottikädelle 'fyysisesti' mahdollista. Tässä tilanteessa acos-funktio saa arvoja, jotka ylittävät arvon 1.0 tai alittavat arvon -1.0, mikä aiheuttaa 'math domain error'-virheilmoituksen. Tämä voitaisiin korjata estämällä ne

lineaariset interpolaatiot, jotka läpäisevät robottikädelle ulottumattoman alueen. Kyseistä ongelmaa ei siis tapahdu nykyisessä asetelmassa, sillä linkit ovat siinä yhtä suuret.

11. **3 parasta ja 1 heikoin kohta**

    Parhaimmat kohdat:
    - Ohjauspaneeli; kytkimet ja painikkeet toimivat juuri halutulla tavalla.
    - Graafiset esineet; piirtyvät skeneen oikein ja päivittyvät kuten pitää.
    - Animointi; sulavaa ja siirtyy kuten käyttäjä haluaa.

    Heikoimmat kohdat:
    - Ensimmäisen nivelen nivelkulman arvo hypähtää tietyissä tilanteissa 360°; tässä ei visuaalisesti ei tapahdu muutosta, mutta esimerkiksi sijainnissa (-50, -1): $\theta_1 = -259.6°$ ja sijainnissa (-50, 1): $\theta_1 = 98.45°$. Tämä todennäköisesti johtuu funktiosta acos2(y, x).

12. **Poikkeamat suunnitelmasta**

    Otin suunnitelmassa erikseen huomioon koordinaatiston, mutta sitä ei tarvinnutkaan itse luoda, sillä PyQt5 kirjasto sisällyttää sen. Lisäsin vielä esineiden grafiikkaan tekstit, jotka kertovat robottikäden nivelkulmien suuruudet, työkalupisteen koordinaatit sekä laatikon koordinaatit. Suunnittelin aluksi, että tekisin kaikki luokat aluksi ilman käyttöliittymän koodausta, mutta päätin kuitenkin aloittaa ensin käyttöliittymästä, johon lisäsin vaihe vaiheelta eri ominaisuuksia. Tällöin oli paljon helpompi nähdä visuaalisesti, onnistuiko joku menetelmä ja heti korjata se. Aloitin siis heti alussa lukemaan PyQt5 dokumentaatiota ja etsimään esimerkkejä, kuinka ikkunoita ja graafisia esineitä luodaan.

13. **Toteutunut työjärjestys ja aikataulu**

    Tein projektin alussa paljon hommia suuren mielenkiinnon takia, joten sain ohjelman kehityksen parissa viikossa puoleen väliin. Jouduin pitämään noin kuukauden verran taukoa ohjelmoinnista johtuen muista kouluprojekteista. Tiedossa kuitenkin oli, kuinka paljon tekemistä oli vielä jäljellä, joten sain projektin ajoissa valmiiksi. Projektin järjestys oli seuraavanlainen (pahoittelut englannin kielestä, mutta kirjoitin gittiin englanniksi… muokkasin tietysti ja tiivistin):

    - 17.03.2021
        - Added files auto.py, item.py, manual.py, robot.py, suction.py and window.py.
        - Started working with GUI in Window class where manual, auto and suction docks were added, which include their widgets.
    - 18.03.2021
        - Added functionality for the manual and auto control in Window class. They still need to be connected for the robot class and update the change in graphics.
        - Added grid layout for suction dock and ticks for manual sliders in Window class.
    - 24.03.2021
        - Removed auto.py, manual.py; their functionalities are now included in Robot class.
        - Added graphics view and scene to the Window class which include the graphics of robot and square.
        - Added forward kinematics into Robot class and made it so that controlling from the graphical interface will update the Robot class information and the Robot class will call the graphics functions to update on the scene.
        - Changed the name of Item class and its graphics item as Square and SquareGraphicsItem.
        - Included Square class into Window class and made the location of the square random when initialized.

- 25.03.2021
  - Graphics for robot hand and square implemented.
  - Manual control implemented.
  - Suction implemented.
  - Square movement implemented.
  - Added some gradient color for the graphics items.
- 19.04.2021
  - Added graphical joints for the RobotGraphicsItem and made the Square GraphicsItem to rotate according to the arm movement when suction is on.
  - Made the rotation of the SquareGraphicsItem more natural by adding the difference between the sum of robot joint angles and the rotation of the square.
- 21.04.2021
  - First try on making the automatic movement.
  - Adjusted the manual control sliders to move according to the automatic movement.
- 22.04.2021
  - Robot hand works automatically.
  - Removed unnecessary if condition from the inverse kinematics function.
- 24.04.2021
  - Increased the increments on automatic control to make the trajectory smoother.
  - Added visual coordinates for the square and end-effector which need some fixing, but they work.
- 26.04.2021
  - Added TextGraphicsItem class for the visual texts in the scene.
  - Fixed a bug where robot arms angles were wrong and a bug which made the square rotate wrong. Removed unused libraries.
- 27.04.2021
  - Cleaned some comments.
  - First version of the unit testing for the kinematics kinematics_test.py implemented.
- 28.04.2021
  - Fixed a bug where the software crashed when the arm was at the edges of its reach and started to move automatically.
  - Finished the tests in kinematics_test.py.
  - Cleaned comments in robot.py.

## 14. Arvio lopputuloksesta

Mielestäni projekti oli todella hyvää harjoittelua graafisen käyttöliittymän oppimisessa. Ohjelma on hyvin hiottu, toimii kuten pitää ja se on intuitiivinen käyttää, mutta ensimmäisen nivelen kulma saattaa välillä pyörähtää 360°, mikä ei kuitenkaan vaikuta ohjelman toimintaan. Tämän korjaamiseksi pitäisi tutkia atan2()-funktion toimintaa ohjelmaa ajettaessa ja selvittää, miksi se tekee näin. Robottikäden molempien nivelten ja työkalupisteen kulma sekä sijainti olisi voinut ilmoittaa samassa käyttäen transformaatiomatriiseja, mikä olisi todennäköisesti helpottanut laskemista sekä datan käsittelyä. Robottikäden ja laatikon törmäyksenkin olisi voinut ottaa huomioon, mutta resurssit sen implementointiin eivät valitettavasti tällä kertaa riittäneet.

Ohjelma soveltuu hyvin muutosten ja laajennusten tekemiseen, sillä se on hyvin kommentoitu ja selkeä rakenteinen. Uusia ohjausmenetelmiä voi lisätä ikkunaan luomalla uuden funktion Window luokkaan ja esineitä voi helposti lisätä muuttujaan Window-luokassa ja laittamalla pari koodin pätkää Window luokan set_graphics funktioon. Esineiden riippuvuuden robottikäteen nähden voidaan puolestaan implementoida Robot-luokassa.

15. **Viitteet**


1. https://robotacademy.net.au/masterclass/inverse-kinematics-and-robot-motion/?lesson=289 (käänteinen kinematiikka)
2. https://robotacademy.net.au/masterclass/robotic-arms-and-forward-kinematics/?lesson=260 (suora kinematiikka toisen nivelen sijainnille)
3. https://robotacademy.net.au/masterclass/robotic-arms-and-forward-kinematics/?lesson=262 (suora kinematiikka työkalupisteen sijainnille)
4. https://zetcode.com/gui/pyqt5/ (PyQt5 kirjastoon tutustumista)
5. https://doc.qt.io/qt-5/qtwidgets-index.html (PyQt5 widgetit, kuten QDockWidget, QLineEdit, QPushButton ja QGraphicsItem)
6. https://likegeeks.com/pyqt5-drawing-tutorial/ (PyQt5 piirtämistutoriaali)
7. https://pythonforundergradengineers.com/unicode-characters-in-python.html (Unicode symbolit pythonissa)
8. https://doc.qt.io/qt-5/qintvalidator.html (Koordinaattien syötteen suodatus)
9. https://doc.qt.io/qt-5/qpen.html#pen-style (Widgettien piirtämistyyli)
10. https://doc.qt.io/qt-5/qbrush.html#setStyle (Widgettien värjäys)
11. https://numpy.org/doc/stable/user/basics.creation.html (Arrayn luominen)
12. https://numpy.org/doc/stable/reference/generated/numpy.around.html (Pyöristäminen)
13. https://numpy.org/doc/stable/reference/generated/numpy.arange.html (For-loopin range)
14. https://docs.python.org/3/library/math.html (trigonometriset funktiot)


16. **Liitteet**

Ei tarvittavia liitteitä.