OULUN AMMATTIKORKEAKOULU

Tomi Yrjänheikki

JAVAFX-KÄYTTÖLIITTYMÄ SARJAPORTTIKOMMUNIKOINNILLA

JavaFX-käyttöliittymä sarjaporttikommunikoinnilla

Tomi Yrjänheikki
Opinnäytetyö
Kevät 2024
Sähkö ja automaatiotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Sähkö- ja automaatiotekniikan tutkinto-ohjelma, Sähkötekniikka

---

Tekijä: Tomi Yrjänheikki
Opinnäytetyön nimi: JavaFX-käyttöliittymä sarjaporttikommunikoinnilla
Työn ohjaaja: Lari Kylmänen
Työn valmistumislukukausi ja -vuosi: Kevät 2024 Sivumäärä: 26 + 3 liitettä

Tässä opinnäytetyössä tutkittiin JavaFX-käyttöliittymäkomponenttikirjaston soveltamista Eclipse IDE-kehitysympäristössä. Työssä kerrotaan, kuinka jSerialComm- ja TilesFX-kirjastoja voidaan hyödyntää sarjaporttikommunikointiin kykenevän käyttöliittymäsovelluksen toteuttamiseksi. Lisäksi tutkimuksessa hyödynnettiin Arduino-mikrokontrolleria ja Arduino IDE:tä I/O-ohjauksen toteuttamiseen.

Opinnäytetyössä kehitettiin käyttöliittymäsovellus, joka sisältää kytkimiä ulkoisten releiden ohjaamiseen. Käyttöliittymässä kytkimen tilan muuttaminen lähettää sarjaportille dataa, jonka perusteella Arduino kytkee I/O-lähtöjään päälle tai pois päältä. Lukijalle esitellään käyttöliittymän koodaaminen ja läpikäydään kaikki sen osa-alueet.

Opinnäytetyö toimii aloitusoppaana JavaFX-käyttöliittymän toteutukseen. Lukija pystyy seuraamaan työtä ja oppimaan käytännön menetelmiä, jotka mahdollistavat oman käyttöliittymän toteuttamisen.

Asiasanat: Arduino, JavaFX, Java, TilesFX, jSerialComm, Eclipse

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Electrical and Automation Engineering, Option of Electrical Engineering

---

Author: Tomi Yrjänheikki
Title of thesis: JavaFX interface with serial port communication
Supervisor: Lari Kylmänen
Term and year when the thesis was submitted: Spring 2024
Number of pages: 26 + 3 appendices

In this thesis, we explore the application of the JavaFX user interface component library in the Eclipse IDE development environment. The study covers how the jSerialComm and TilesFX libraries can be utilized to implement a user interface application capable of serial port communication. Additionally, the research leverages Arduino microcontrollers and the Arduino IDE for implementing I/O control.

A user interface application was developed in the thesis, containing switches for controlling external relays. Changing the state of the switches in the user interface sends data to the serial port, based on which the Arduino toggles its I/O outputs on or off. The thesis presents the coding of the user interface, thoroughly covering all its components.

The thesis serves as a guide to implementing JavaFX user interfaces. Readers can follow along with the work and learn practical methods that enable the implementation of their own user interfaces.

Keywords: Arduino, JavaFX, Java, TilesFX, jSerialComm, Eclipse

SISÄLLYS

SANASTO

JAR	Tiedostomuoto, joka pakkaa Java-ohjelmointikielellä kirjoitetun sovelluksen suoritettavaksi paketiksi
Java	Ohjelmointikieli ja alusta, joka soveltuu erilaisiin sovelluksiin
JavaFX	Javan käyttöliittymäkomponenttikirjasto
JDK	Java työkalupaketti, joka sisältää kaiken tarvittavan Java-ohjelmien kehittämiseen, kääntämiseen ja suorittamiseen
JRE	Ohjelmisto, joka mahdollistaa Java-sovellusten suorittamisen tietokoneella
jSerialComm	Java-kirjasto sarjaporttien hallintaan Java-sovelluksissa
MOSFET	Sähköinen komponentti, joka pystyy kontrolloimaan sähkövirran virtausta ja toimimaan kytkimenä sähköpiireissä
Olio	Java-ohjelmoinnin perusyksikkö, joka sisältää tilaa ja toiminnallisuutta
Sarjaportti	Tietokoneen liitäntä, joka mahdollistaa tiedonsiirron ulkoisten laitteiden kanssa
TilesFX	JavaFX-kirjasto käyttöliittymien luomiseen

1 JOHDANTO

Tässä opinnäytetyössä käsitellään ABB:n taajuusmuuttajien demoyksiköiden etäkäyttöä. Taajuusmuuttajien demoyksiköitä käytetään ohjelmistollisten parametrimuutosten toimivuuden tarkistamiseen ennen niiden siirtämistä lopulliseen käyttökohteeseen. ABB:n tavoitteena oli mahdollistaa demoyksiköiden etäkäyttö internetin välityksellä, jotta taajuusmuuttajat voitaisiin sijoittaa keskitetysti ABB:n Oulun toimipisteeseen, mikä vähentäisi tarvetta demoyksiköiden kuljettamiseen eri puolille Suomea ja mahdollistaisi parametritestaukset myös toimipisteen ulkopuolella.

Etäkäyttö toteutettiin sijoittamalla demoyksiköiden läheisyyteen tietokone, jossa on tarvittava ohjelmisto taajuusmuuttajien parametrimuutoksia varten. Tietokoneeseen lisättiin tarvittavat tiedonsiirtojohdotukset, joiden kautta saatiin yhteys demoyksiköihin. Etäkäyttö tietokoneeseen toteutettiin Windowsin etätyöpöytäyhteyttä käyttäen. Lisäksi ABB:lle kehitettiin JavaFX-käyttöliittymä tietokoneeseen, jonka kautta annetaan komento halutun taajuusmuuttajamallin demoyksikön käynnistämiseen tai sammuttamiseen. Käyttöliittymä lähettää sarjaportin kautta komennon Arduino-mikrokontrollerille, joka ohjaa MOSFET-transistoreja. MOSFET-transistorien kautta syötetään 24 VDC ohjausjännitettä releille, joiden kautta verkkovirta kytkeytyy haluttuun demoyksikköön.

2 OHJELMAT

Demoyksiköiden virransyötön ohjaaminen toteutettiin käyttöliittymällä ja Arduino-mikrokontrollerilla. Käyttöliittymän ohjelmointiin käytettiin Java-ohjelmointikieltä, sen JavaFX-käyttöliittymäkirjastoa sekä TilesFX- ja jSerialComm-ohjelmointikirjastoja. Arduinon ohjelmointi suoritettiin Arduino IDE:llä. Seuraavissa alaluvuissa esitellään tarkemmin opinnäytetyössä hyödynnettyjen työkalujen taustoja.

2.1 Java

Java on Sun Microsystemsin kehittämä olio-ohjelmointikieli, joka on rakennettu C++:n pohjalta. Javan alkuperäisenä päämääränä oli toimia kevyehkönä kielenä sulautetuissa järjestelmissä televisioissa, kahvinkeittimissä tai muissa vastaavissa käytöissä. Tilanne kuitenkin muuttui graafisten käyttöliittymien yleistyessä web-selaimissa. Java kerkesi ensimmäisenä uudelle markkina-alueelle, josta seurasi sen valtaisa suosion nousu. (1, s. 19.)

2.2 JavaFX

JavaFX on avoimen lähdekoodin ohjelmistokehys ja alusta, jota käytetään graafisten käyttöliittymien luomiseen. Se on toteutettu Java-ohjelmointikielellä, mikä mahdollistaa kaikkien Javan ominaisuuksien ja toimintojen käytön graafisten käyttöliittymien toteuttamisessa. Käyttöliittymä voidaan toteuttaa joko suoraan Javalla tai XML-pohjaisella FXML-kielellä.

JavaFX:n kehitys alkoi Chris Oliverin työskennellessä SeeBeyond-yhtiössä, jolloin sitä kutsuttiin nimellä F3 (Form Follows Function). Sun Microsystems osti SeeBeyondin vuonna 2007, minkä jälkeen F3 sai uuden nimen: JavaFX. Oracle puolestaan hankki Sun Microsystemsin vuonna 2010 ja avasi JavaFX:n lähdekoodin yhteisön kehitykselle vuonna 2013. (2.)

2.3 TilesFX

TilesFX on JavaFX-kirjasto, jonka tarkoitus on helpottaa graafisen käyttöliittymän toteutusta. Gerrit Grunwald julkaisi kirjaston ensimmäisen version joulukuussa 2016 (3). Kirjasto sisältää useita erilaisia komponentteja, jotka mahdollistavat monipuolisten käyttöliittymien rakentamisen. TilesFX:n avulla käyttäjä voi luoda tarvitsemansa komponentit ja sijoittaa ne käyttöliittymään omiin laatikkoihinsa, joita kirjastossa kutsutaan tileiksi. Kuvassa 1 nähdään TilesFX-kirjaston demonäkymän, joka sisältää esimerkin jokaisesta komponentista.

Kuva 1. TilesFX-komponentit (4)

2.4 jSerialComm

jSerialComm on Java-ohjelmointikirjasto, joka mahdollistaa alustasta riippumattoman yhteyden muodostamisen käytettävän laitteen sarjaportteihin. Kirjasto on resurssienkäytön suhteen kevyt ja sen avulla kaikki käytettävissä olevat sarjaportit saadaan luettaviksi Javan InputStream- ja OutputStream-komponenteille. Lisäksi sarjaporttien tiedonsiirtonopeus (baud rate), data bitit, stop bitit ja pariteetti ovat määriteltävissä kirjaston avulla. (5.)

2.5 Eclipse IDE

Eclipse IDE on avoimen lähdekoodin kehitysympäristö, joka on erityisen tunnettu Javan kehitysympäristönä, mutta tarjoaa myös tukea useille muille ohjelmointikielille. Se on loistava alusta ohjelmien testaamiseen ja mahdolliseen virheenjäljitykseen. Eclipse IDE mahdollistaa myös valmiin Java-koodin kokoamisen ajettavaksi ohjelmaksi. (6.)

2.6 Arduino

Arduinon kehitys alkoi vuonna 2005 kun kaksi italialaista insinööriä, Massimo Banzi ja David Cuartielles, aloittivat yhteistyön muiden avoimen lähdekoodin yhteisön jäsenten kanssa. Tavoitteena oli kehittää edullinen ja helppokäyttöinen elektroniikkakehitysalusta, joka rohkaisisi ihmisiä kokeilemaan ohjelmointia ja siihen liittyvää elektroniikkaa. (7.)

Arduino on avoimeen lähdekoodiin perustuva kehitysalusta, johon kuuluvat fyysinen laitteisto sekä ohjelmisto. Suunnittelun lähtökohtana on ollut prototyyppien rakentamisen ja koodin toiminnallisuuden testaamisen helppous. Arduinon fyysinen laitteisto on mikrokontrolleri, joka kykenee ohjaamaan elektronisia komponentteja ja suorittamaan ohjelmia. Ohjelmointi suoritetaan Arduino Integrated Development Environment (IDE) -ohjelmistolla, joka on kehitetty erityisesti Arduinon ohjelmointiin. (7.)

3 ARDUINO RELEOHJAUS

Arduinon tehtävä etäkäytössä on ohjata MOSFET-piirilevyä Java-sovelluksesta tulevien sarjaporttikomentojen mukaan. Mikrokontrolleriin oli ohjelmoitava toiminnat, joiden kautta se pystyi lukemaan sarjaporttiin tulevat viestit ja niiden sisällön mukaan kytkemään päälle pinnit.

Piirilevyllä on yhdeksän MOSFET-transistoria, joita ohjataan syöttämällä jännitettä niiden gate-pinnille. Jokaisen MOSFET-transistorin gate-pinni kytkettiin Arduinon ohjaukseen, jossa sen jännitettä säädetään nollatason ja 5 VDC:n välillä. Tämä säätö määrittää 24 voltin releiden tilan, mikä puolestaan ohjaa 230 voltin virran syöttöä taajuusmuuttajille.

3.1 MOSFET-piirilevy

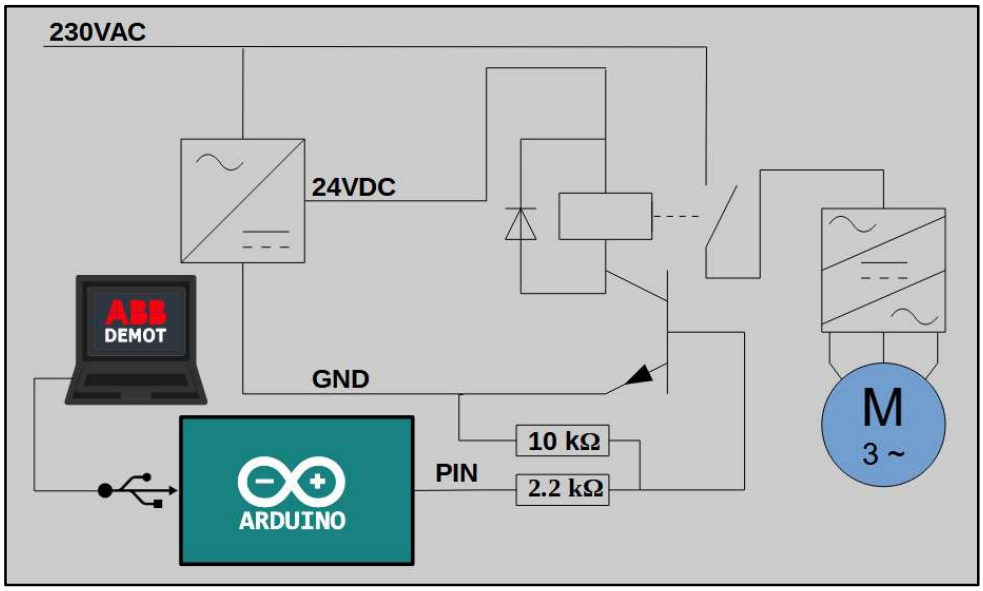
Arduinon ohjaama MOSFET-piirilevy rakennettiin IRLZ34N MOSFET-transistorin ympärille. IRLZ34N on N-kanavan MOSFET, joka lisättiin releiden negatiiviseen lähtöön. Sillä on korkea virran ja jännitteen kestävyys, mikä tekee siitä toimintavarman komponentin releiden ohjaamiseen. Sitä voidaan ohjata logiikkatasolla, eli kytkentä digitaalisissa piireissä on toteutettavissa ilman ajuri- tai tasonmuunninpiiriä.

MOSFET kytkettiin virtapiiriin gate-, source- ja drain-liitäntöjen kautta. Source-liitäntä yhdistettiin matalaan potentiaaliin, drain-liitäntä releen (-) kytkentään ja gate-liitäntä Arduinon ohjausjännitteeseen. Arduinon digitaalisen lähdön ja gaten väliin liitettiin 2.2 kΩ vastus, jolla varmistettiin, ettei Arduinon digitaalisen lähdön virtarajaa ylitetä. Jos Arduino-mikrokontrolleri vioittuisi, gate-liitäntä jäisi 'kelluvaksi', mistä voisi seurata MOSFET-transistorin tuhoutuminen. Tämän vuoksi gate-liitäntään täytyi rakentaa PULL-DOWN-kytkentä. Tämä toteutettiin kytkemällä gate 10 kΩ vastuksen kautta sourceen.

3.2 230 V:n releet

Verkkovirran ohjaus toteutettiin Finder 46.52.9.024.0040-releillä. Releiden kelan jännite on 24 VDC, jänniteluokitus 250 VAC ja virrankatkaisukyky 8 A. Verkkovirta ketjutettiin yhdeksälle releelle yhdestä tulosta. Jokaisesta releestä vedettiin oma lähtö halutulle taajuusmuuttajalle.

Releitä ohjataan MOSFET-piirilevyllä, joka yhdistää releen 24 VDC kelan (-) navan sourceen ja tämän seurauksena rele sulkeutuu luoden verkkovirtayhteyden haluttuun taajuusmuuttajaan. Releiden +/- napojen väliin liitettiin vastadiodi, sillä induktiivisen kuorman sammutustilanteessa saattaa syntyä jännitepiikki, joka voi vahingoittaa muita komponentteja. Vastadiodin tarkoituksena on mahdollistaa virran liike kelassa ja siten estää takaiskuvirran (back EMF) aiheuttamat vauriot. Kuvassa 2 esitetään yksittäisen releen kytkentäkaavio.



Kuva 2. Releen kytkentäkaavio

3.3 Arduinon koodi

MOSFET-piirilevyä ohjaava koodi toteutettiin Arduino IDE:llä. Koodi pysyi yksinkertaisena, koska releiden ja MOSFET-transistorien automaattiseen sammuttamiseen liittyvät toiminnot sisällytettiin Java-sovellukseen.

Arduinon ’void setup ()’ -funktioissa määriteltiin sarjaporttiyhteys ja sen nopeus. Tämän nopeuden täytyy vastata Java-sovellukseen asetettuja sarjaporttiasetuksia. MOSFETIEN ohjaamiseen tarvittavat pinnit määriteltiin ’pinMode’-funktiolla digitaalisiksi lähdöiksi, mikä mahdollistaa niiden käyttämisen digitaalisien signaalien ulostuloina. Kuvassa 3 on määritetty sarjaporttinopeus sekä pinni 2 digitaaliseksi ulostuloksi.

```
1  void setup()
2  {
3    Serial.begin(9600);
4    pinMode(2, OUTPUT);
```

Kuva 3. Setup-funktio

Mikrokontrollerin 'void loop ()' -funktion koodissa määriteltiin toiminto, joka aktivoituu aina, kun sarjaporttiin saapuu tavu (byte). Tämä tavu tallennetaan muuttujaan ’JavaFX’. Koodissa vertaillaan, vastaako muuttujan arvo annettuja ehtoja, ja jos vastaa, suoritetaan digitaalisen lähdön kytkentä päälle tai pois päältä. Java-sovelluksessa määriteltiin lähetettävät tavut vastaamaan Arduino-koodin tavuja ja näistä seuraavia toimintoja. Kuvassa 4 esitetään koodi, joka ohjaa pinniä ja vaihtelee sen tilaa sarjaporttiin saapuvan datan mukaan. Koko Arduinon koodi on liitteessä 1.

```
16 void loop()
17 {
18   byte JavaFX;
19   if(Serial.available())
20   {
21     JavaFX = Serial.read();
22     if(JavaFX == 10){
23       digitalWrite(2,LOW);
24       }
25     else if(JavaFX == 11){
26       digitalWrite(2,HIGH);
27       }
```

Kuva 4. Loop-funktio

4 JAVAFX-OHJELMISTON TOTEUTUS

Sovelluksen toteuttamiseksi tietokoneelle täytyi ensin asentaa JavaFX-ohjelmistokehityspaketti. Lisäksi asennettiin kehitysympäristö, joka tarjoaa tarvittavat työkalut ohjelmistokehitykseen. Kun tietokoneeseen oli tehty tarvittavat muutokset, aloitettiin Java-sovelluksen koodaus, mikä sisälsi tarvittavien Java-kirjastojen yhdistämisen sovelluksen koodiin sekä käyttöliittymän ja sarjaporttikommunikoinnin toteuttamisen. Lopuksi toimiva sovellus pakattiin ajettavaksi tiedostoksi. Seuraavaksi käydään läpi toteutuksen vaiheet yksityiskohtaisemmin.

4.1 Valmistelut

Ennen ohjelman koodaamista varmistettiin, että tietokoneeseen on asennettu oikea JDK (Java Development Kit). Tämä on tärkeää, koska JavaFX-sovellukset eivät toimi, ellei laitteessa ole niitä tukevaa Java-alustaa. Ohjelmointiin käytetyssä tietokoneessa ei ollut Java-sovellusten kehittämiseen sopivaa alustaa, joten tämä täytyi ladata ja asentaa. Alustaksi valittiin Bellsoftin Liberica JDK, jonka perustana toimii OpenJDK. Asennus toteutettiin lataamalla asennuspaketti Windows-tietokoneeseen valmistajan sivuilta. Asennuksen onnistuminen varmistettiin käyttämällä Windowsin komentokehote-sovellusta, johon syötettiin komento ’java -version’. Komento tuo näkyville koneeseen asennetun Java-version kuvan 5 mukaisesti ja näin voidaan varmistaa OpenJDK:n käytössä oleminen. Tämän jälkeen koneeseen asennettiin Eclipse IDE seuraamalla valmistajan normaalin asennuksen ohjeistusta.

```
C:\Users\OMISTAJA>java -version
openjdk version "21.0.2" 2024-01-16 LTS
OpenJDK Runtime Environment (build 21.0.2+14-LTS)
OpenJDK 64-Bit Server VM (build 21.0.2+14-LTS, mixed mode, sharing)
```

Kuva 5. Java-version tarkistus

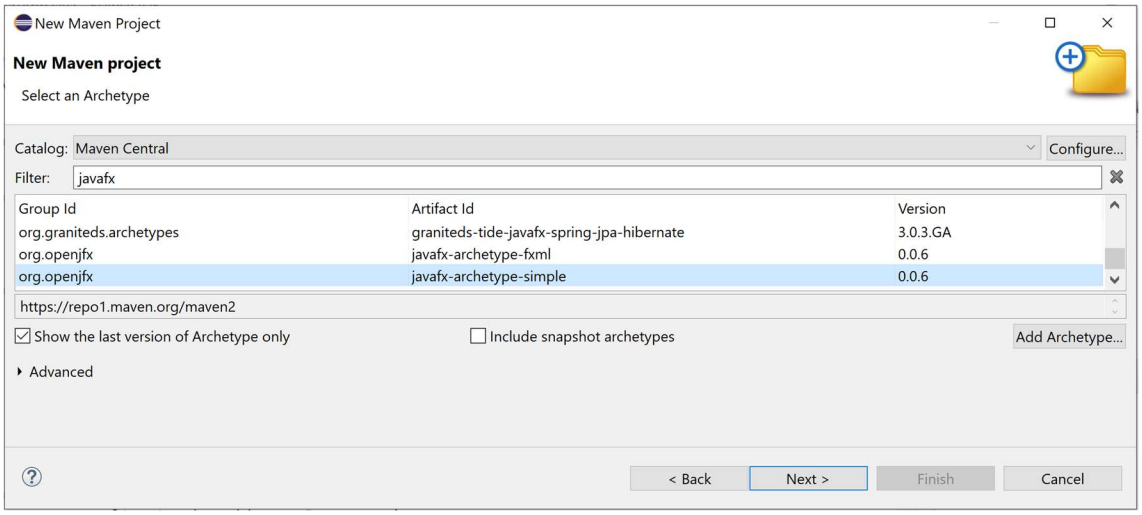
Koska ohjelmointi ja Java-sovelluksen käyttö tapahtuivat eri tietokoneilla, täytyi myös loppukäyttöön tarkoitetulle koneelle asentaa oikea Java-ohjelmisto. Johtuen Java-sovelluksen toteutuksesta erillisellä koneella ei ollut tarvetta asentaa Java Development Kitiä (JDK). Loppukäyttö-tietokoneelle asennettiin Java Runtime Environment (JRE). Bellsoftin Liberica JRE oli ainoa ympäristö, joka mahdollisti JavaFX-sovelluksen käytön loppukäyttökohteessa johtuen

tietokoneen 32-bitin arkkitehtuurista, jota Bellsoft muista ohjelmiston tarjoajista poiketen vielä tukee.

4.2 Projektipohja

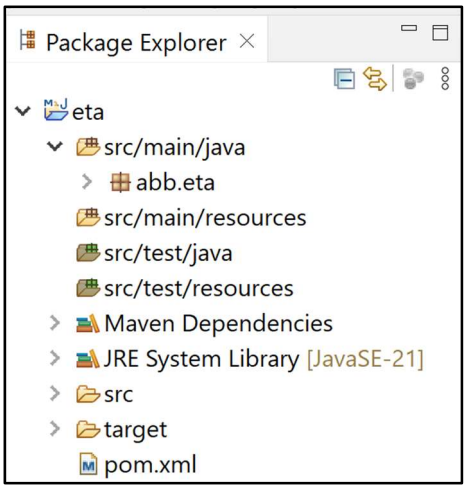
JavaFX-pohjaisen käyttöliittymän toteutuksen aloittamiseen käyttäjällä on useita vaihtoehtoja. Käyttäjä voi manuaalisesti ladata projektiin tarvittavat kirjastot ja määritellä nämä Eclipse IDE:n käyttöliittymästä. Tämä vaatii ohjelmoijalta hyvää ymmärrystä Eclipsen ja Java ohjelmoinnin perusteista. Toisena vaihtoehtona on hyödyntää projekti- ja rakennustyökaluja, jotka on suunniteltu helpottamaan Java-projektien toteutusta. Näitä työkaluja ovat Apache Ant, Apache Maven ja Gradle. Opinnäytetyön toteutuksessa hyödynnettiin avoimen lähdekoodin Apache Mavenia.

Käyttöliittymän luominen opinnäytetyöhön aloitettiin luomalla Maven project. Tämä tapahtuu Eclipsessä valitsemalla File > New > Project... > Maven > Maven Project. Aukeavassa ikkunassa voidaan määrittää projektin tallennuskansio ja sijainti, jonka jälkeen painetaan ”Next”-painiketta. Käyttöliittymään aukeaa uusi ikkuna, jossa pyydetään valitsemaan ’archetype’. Opinnäytetyössä valittiin projektikatalogiksi (Catalog) ’Maven Central’ ja filtteriksi (Filter) ’javafx’. Ponnahdusikkunaan aukeaa käyttäjän valittavaksi valmiita projektipohjia, joita voidaan hyödyntää oman JavaFX-ohjelman luomisen pohjana (kuva 6). Opinnäytetyöhön valittiin Group-id: org.openjfx sekä Artifact Id: javafx-archetype-simple.



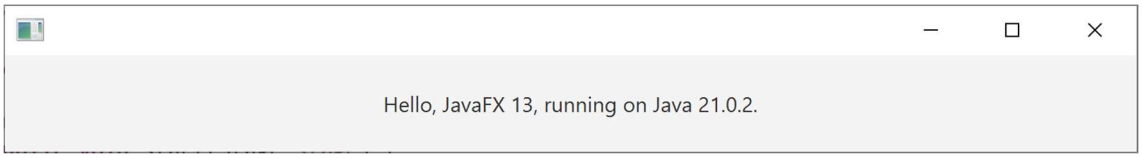
Kuva 6. Maven-sovelluksen pohja

Valinnan jälkeen avautuu uusi ikkuna, johon käyttäjä voi määrittää haluamansa ’group-id’ ja ’artifact-id’ -nimitykset. Opinnäytetyöhön valittiin ’abb’ ja ’eta’. Määrityksen jälkeen Eclipse aloittaa uuden projektin kokoamisen. Kokoamisen jälkeen Eclipsen Package Explorer -palkkiin tulee näkyviin juuri kootun projektin rakenne (kuva 7).



Kuva 7. Maven-sovelluksen rakenne

Pohja on valmis sovellus, joka voidaan koota ja käynnistää Eclipsessä. Käynnistämällä pohja varmistetaan, että tietokoneen ja Eclipsen asetukset on määritelty oikein. Kokoaminen suoritetaan valitsemalla hiiren oikealla näppäimellä sovellus, minkä jälkeen valitaan Run As > Maven build…. Aukeavan ponnahdusikkunan Goals: -kohtaan kirjoitetaan ’clean javafx:run’ -komento ja painetaan ”Run”-painiketta. Java-sovellus käynnistyy ja uuteen ikkunaan aukeaa kuvan 8 mukainen näkymä.



Kuva 8. JavaFX-sovellus

Tässä vaiheessa voitiin aloittaa varsinaisen opinnäytetyöhön liittyvän sovelluksen valmistelu. Suunnitteluvaiheesta tiedettiin, että käyttöliittymäsovelluksessa on oltava tuki serialport-kommunikaatiolle, johon käytettiin jSerialComm-kirjastoa. Sovelluksen käyttöliittymä toteutettiin TilesFX-kirjastoa hyödyntäen. Nämä kirjastot täytyi lisätä projektiin. Lisääminen tapahtuu muokkaamalla projektipuussa sijaitsevaa ’pom.xml’-tekstitiedostoa. Pom-tiedoston kautta voidaan hakea tarvittavat komponentit suoraan maven repositorystä (https://mvnrepository.com/), joka on keskitetty tietovarasto Java-kirjastoille, riippuvuuksille ja komponenteille. Tämän lähestymistavan etuna on vähentynyt tarve ladata ja ylläpitää tarvittavia kirjastoja käyttäjän omalla tietokoneella.

Tarvittavat kirjastot lisättiin sovellukseen etsimällä ne maven repository -sivuston search-toimintaa hyödyntäen. Kun tarvittava kirjasto löytyi, valittiin haluttu versionumero, jolloin pom.xml tiedostoon lisättävä teksti aukeaa käyttäjän kopioitavaksi. Muokattu pom.xml tiedosto on liitteessä 2.

4.3 TilesFX-käyttöliittymä

Käyttöliittymän toteutus aloitettiin avaamalla App.java-tiedosto Eclipsen IDE:n työpöydälle. Tiedosto sisälsi koodia, jolla varmistettiin JavaFX:n toimivuus. Tämä koodi voitiin tässä vaiheessa poistaa ja aloittaa uuden käyttöliittymän muodostaminen.

Ensimmäiseksi toteutettiin yksittäinen tiili TilesFX-kirjastoa hyödyntäen. Tiilityypiksi (Skintype) valittiin kytkin (kuva 9). Tiilityypissä määriteltiin otsikkoteksti (title), teksti (text) ja aktiivinen väri (activeColor). Muokattavia parametreja on huomattavasti enemmän ja niihin palataan myöhemmin opinnäytetyössä. Komennolla: '.build();' lopetetaan tiilen määrittely. Vastaavia tiiliä koodataan sovellukseen yhdeksän kappaletta, joka vastaa Arduinoon liitettyjen releiden määrää.

Kuva 9. TilesFX-kytkin

Tiilien kokoamisen jälkeen määriteltiin, kuinka ne näkyvät avautuvassa sovellusikkunassa. Tähän hyödynnettiin säiliö-komponentteja, jotka mahdollistavat elementtien järjestämisen selkeästi käyttöliittymään. JavaFX mahdollistaa useita asettelumalleja elementtien paikan määrittelemiseen. Nämä asettelumallit ovat yhdisteltävissä keskenään, jos halutaan rakentaa monimutkaisempia käyttöliittymiä, mutta etäkäyttösovelluksessa ei ollut tarvetta erityisen monimutkaiseen asetteluun ja siksi valitsin 'GridPane'- asettelumallin, joka mahdollistaa ruudukkomaisen käyttöliittymän toteuttamisen.

Koodilla ’`GridPane Paneli = new GridPane();`’ luotiin uusi Java-olio nimeltä ’Paneli’, joka mahdollisti elementtien sijoittelun riveille sekä sarakkeille. Elementtejä lisätään paneliin koodilla ’`Paneli.add((rele1), 0, 1)`’, missä elementti sijoittuu sarakkeelle 0 ja riville 1. Elementtien ja avautuvan sovellusikkunan väliin haluttiin luoda reunat, jotka määriteltiin ’`Paneli.setPadding(new Insets(10, 10, 10, 10));`’ koodilla 10 pikselin levyisiksi. Seuraavaksi ’`Paneli.setHgap(10);`’ ja ’`Paneli.setVgap(10);`’ koodeilla aseteltiin elementit ruudukoihin siten, etteivät ne ole keskenään kosketuksissa.

Koodilla ’`var scene = new Scene(Paneli, 1920, 1080);`’ määriteltiin uusi ikkuna-olio (scene) ja minkä kokoisena ikkunana Java-sovellus aukeaa, kun se käynnistetään. Stage-komennoilla määriteltiin tarkemmin aukevan ikkunan ominaisuuksia, kuten skaalautuvuutta, otsikkoa ja aukaistava ikkuna-komponentti (setScene). Lopputuloksena syntyi kuvan 10 mukainen käyttöliittymänäkymän määrittely. Sovelluksen toimintaa testattiin tästä eteenpäin valinnoilla Run > Run As > Java Application.

```
GridPane Paneli = new GridPane();
Paneli.add((rele1), 0, 1);
Paneli.add((rele2), 1, 1);
Paneli.add((rele3), 2, 1);
Paneli.add((rele4), 0, 2);
Paneli.setPadding(new Insets(10, 10, 10, 10));
Paneli.setHgap(10);
Paneli.setVgap(10);
var scene = new Scene(Paneli, 1920, 1080);
stage.setResizable(true);
stage.setScene(scene);
stage.setFullScreen(false);
stage.setTitle("Opinnäytetyö");
stage.show();
```

Kuva 10. Käyttöliittymänäkymän määritys

4.4 Sarjaporttiyhteys

Java-sovelluksen tarkoituksena on ohjata Arduinon I/O-pinnejä sarjaportin kautta lähetettävillä komennoilla. Tämän mahdollistamiseksi täytyi sovellukseen lisätä toiminnallisuus sarjaporttien hallintaan. Tässä hyödynnettiin jSerialComm-kirjastoa, joka oli lisätty pom.xml-tiedoston kautta projektiin.

Ensimmäiseksi luotiin ’SerialPort’-luokka, joka antaa Java-rajapinnan sarjaportin hallintaan. Sarjaportille annettiin nimeksi ’arduino’. Koska Arduino-mikrokontrolleri on kytketty tiettyyn porttiin, käyttöliittymästä jätettiin pois portinvalitsin, sen sijaan määritettiin portin osoite manuaalisesti ’getCommPort(”COM11”)’-komennolla (kuva 11). Tämän etuna on, ettei käyttäjän tarvitse määrittää

sarjaporttia joka kerta sovelluksen käynnistyessä ennen varsinaisen käyttötarkoituksen mahdollistamista.

```
public class App extends Application {
    public static SerialPort arduino = SerialPort.getCommPort("COM11");
```

Kuva 11. Sarjaportin määrittäminen

Seuraavaksi määriteltiin sarjaportin perusparametrit ’setComPortParameters’-osiossa niin, että ne vastaavat Arduino-mikrokontrollerin parametrejä. Aikakatkaisut luku- ja kirjoitusajan (’setComPortTimeouts’) osalta asetettiin nollaan (kuva 12).

```
public static void main(String[] args) {
    arduino.setComPortParameters(9600, 8, 1, 0);
    arduino.setComPortTimeouts(SerialPort.TIMEOUT_WRITE_BLOCKING, 0, 0);
```

Kuva 12. Sarjaportin parametrit

4.5 Sovelluksen toiminnallisuus

Java-sovellukseen liitetyt kirjastot ja komponentit mahdollistivat tässä vaiheessa toiminnallisen ohjelman toteutuksen. Sovelluksen on jatkuvasti seurattava käyttöliittymän relekytkimien tilaa. Tämä toteutettiin luomalla koodiin ’AnimationTimer’-olio, joka mahdollistaa toistuvat animaatiot ja päivitykset Java-käyttöliittymään. Olio kutsuu ’handle’-metodia jokaisella animaatiokierroksella, joten sijoitimme toiminnallisen koodin ’handle’-metodin sisään (kuva 14). Toiminnallisessa koodissa käytetään muuttujia ’rele1arvo’, joka kertoo kytkimen nykyisen asettelun, ja ’rele1old’, joka taas kertoo edellisen asettelun. Nämä määriteltiin Java-koodissa staattisiksi kokonaislukumuuttujiksi, joiden arvoa muutetaan tilanteen mukaan (kuva 13).

```
static      Integer     rele1arvo = 10;
static      Integer     rele1old = 11;
```

Kuva 13. Kokonaislukumuuttujat

Toiminnallisen koodin ensimmäisessä vaiheessa tarkistetaan, onko tiilen 'rele1'- kytkin aktivoitu. Jos 'rele1' on aktiivinen, määrätään 'rele1arvoksi' kokonaisluku 11, muussa tapauksessa arvoksi määrätään 10. Kokonaisluku on vapaavalintainen mutta koodin selkeyden kannalta päätin ensimmäisen numeron viittavan releen numeroon ja toisen siihen, onko se päällä vai pois päältä. Esimerkiksi luku ’71’ kertoo releen numero seitsemän olevan päällä.

Seuraavassa vaiheessa tarkistetaan, täyttyvätkö ehdot 'rele1'-kytkimen aktivoitumiselle ja onko 'rele1arvo' eri kuin 'rele1old'. Jos ehdot ovat tosia, 'rele1old' arvoksi asetetaan 'rele1arvo' ja lähetetään sarjaporttiin 'rele1arvo', joka olisi rele1 tapauksessa ’11’. Mikäli edellinen ehto ei täyty, tarkistetaan seuraavaksi, onko 'rele1'-kytkin passiivisena ja onko 'rele1arvo' eri kuin 'rele1old'. Jos nämä ehdot täyttyvät, muutetaan 'rele1old' vastaamaan 'rele1arvoa' ja lähetetään tämä arvo sarjaporttiin. Jos rele1 on passiivisena, arvo '10' lähetetään sarjaporttiin. Kuvassa 14 on esitetty toimiva koodi kokonaisuudessaan.

```
AnimationTimer timer = new AnimationTimer() {
    @Override
    public void handle(long now) {
        //RELE1-------------------------------------------------------
        if(rele1.isActive()) {
            rele1arvo=11;
          }
        else if(!rele1.isActive()) {
            rele1arvo=10;
          }
        if (rele1.isActive() && (!rele1arvo.equals(rele1old))) {
            rele1old = rele1arvo;
            try {
                arduino.getOutputStream().write(rele1arvo.byteValue());
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        else if(!rele1.isActive() && (!rele1arvo.equals(rele1old))) {
            rele1old = rele1arvo;
            try {
                arduino.getOutputStream().write(rele1arvo.byteValue());
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        }
};
```

Kuva 14. AnimationTimer-olio

Sama toiminnallinen koodi toistettiin kaikille yhdeksälle releelle tarvittavin muutoksin. Tuloksena on Java-sovellus, jossa on yhdeksän Tile-komponenttia Kytkimen aktivointi lähettää sarjaporttiin numeerisen arvon, jotka Arduino mikrokontrolleri pystyy lukemaan.

Koodi oli tässä vaiheessa toimiva, mutta siitä puuttui vielä tärkeitä ominaisuuksia. Jos sovellus sammutettiin, jäivät päälle kytketyt releet edelleen virrallisiksi. Java-koodiin lisättiin kuvan 15 esittämä osio, joka aktivoituu, kun sovellus sammutetaan.

Koodiin määriteltiin staattinen kokonaislukumuuttuja, jonka alkuarvoksi annettiin ’10’. Kun ohjelma sammutetaan, sarjaportille lähetetään jokaisen releen poiskytkentään johtava arvo 40 millisekunnin

välein. Tällä varmistetaan, etteivät releet jää virrallisiksi sekä se, että Java-sovelluksen käynnistyessä kytkimien arvot vastaavat todellista tilannetta, missä kaikki releet ovat virrattomia.

```
stage.setOnCloseRequest(e -> {
    while(sammutus <= 90) {
    try {
        arduino.getOutputStream().write(sammutus.byteValue());
        sammutus +=10;
        Thread.sleep(40);
    } catch (IOException | InterruptedException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
        Platform.exit();
        System.exit(0);
    }
    }
    Platform.exit();
    System.exit(0);
    }
    );
```

Kuva 15. Sovelluksen sammutuskomentosarja

Sovelluksen avulla pystyttiin kytkemään releet päälle ongelmitta, kun Arduino kytkettiin määriteltyyn sarjaporttiin, mutta yhteyden mahdollinen katkeaminen virheen seurauksena ei näkynyt sovelluksesta. Tilanne korjattiin luomalla Tile-komponentti, joka kertoo käyttäjälle yhteyden tilan simuloidulla led-valolla ja tilaa kuvaavalla tekstillä (kuva 16). Ilman koodia ’`Arduino_Yhteys.setActive(!Arduino_Yhteys.isActive());`’ sarjaportin tila ei vaikuta ledin väriin käyttöliittymässä.

```
Tile Arduino_Yhteys = TileBuilder.create()
    .skinType(eu.hansolo.tilesfx.Tile.SkinType.LED)
    .prefSize(200, 80)
    .backgroundColor(Color.DARKGRAY)
    .textSize(Tile.TextSize.BIGGER)
    .descriptionAlignment(Pos.CENTER)
    .textVisible(true)
    .build();

Arduino_Yhteys.setActive(!Arduino_Yhteys.isActive());
```

Kuva 16. Arduinon tilaa kuvaava Tile

Sovelluksen ’AnimationTimer’-osioon lisättiin koodia kuvan 17 mukaisesti. Koodilla ohjataan ’Arduino_Yhteys’ -komponentin ominaisuuksia riippuen sarjaportin yhteyden tilasta. Koodissa määritellään, että sarjaporttiyhteyden puuttuessa ledin väriksi asetetaan punainen ja teksti 'ARDUINOON EI YHTEYTTÄ'. Yhteyden ollessa kunnossa ledin väriksi määrätään vihreä ja teksti ’YHTEYS MUODOSTETTU’. Koodissa käytetään myös ’`Arduino_Yhteys.setBackgroundImageOpacity`’ -metodia, joka muuttaa taustakuvan läpinäkyvyyttä sarjaporttiyhteyden tilan mukaan. Tämän toiminnon merkitys käydään läpi tuonnempana kuvan lisäämistä käsittelevässä osiossa.

```
//ARDUINO YHTEYDEN TILA--------------------------------------
if(!arduino.openPort()) {
    Arduino_Yhteys.setActiveColor(Bright.RED);
    Arduino_Yhteys.setDescription("ARDUINOON EI YHTEYTTÄ    VARMISTA USB-JOHTO");
    Arduino_Yhteys.setBackgroundImageOpacity(0.5);
}
else if(arduino.openPort()) {
    Arduino_Yhteys.setActiveColor(Bright.GREEN);
    Arduino_Yhteys.setDescription("YHTEYS MUODOSTETTU");
    Arduino_Yhteys.setBackgroundImageOpacity(1);
}
```

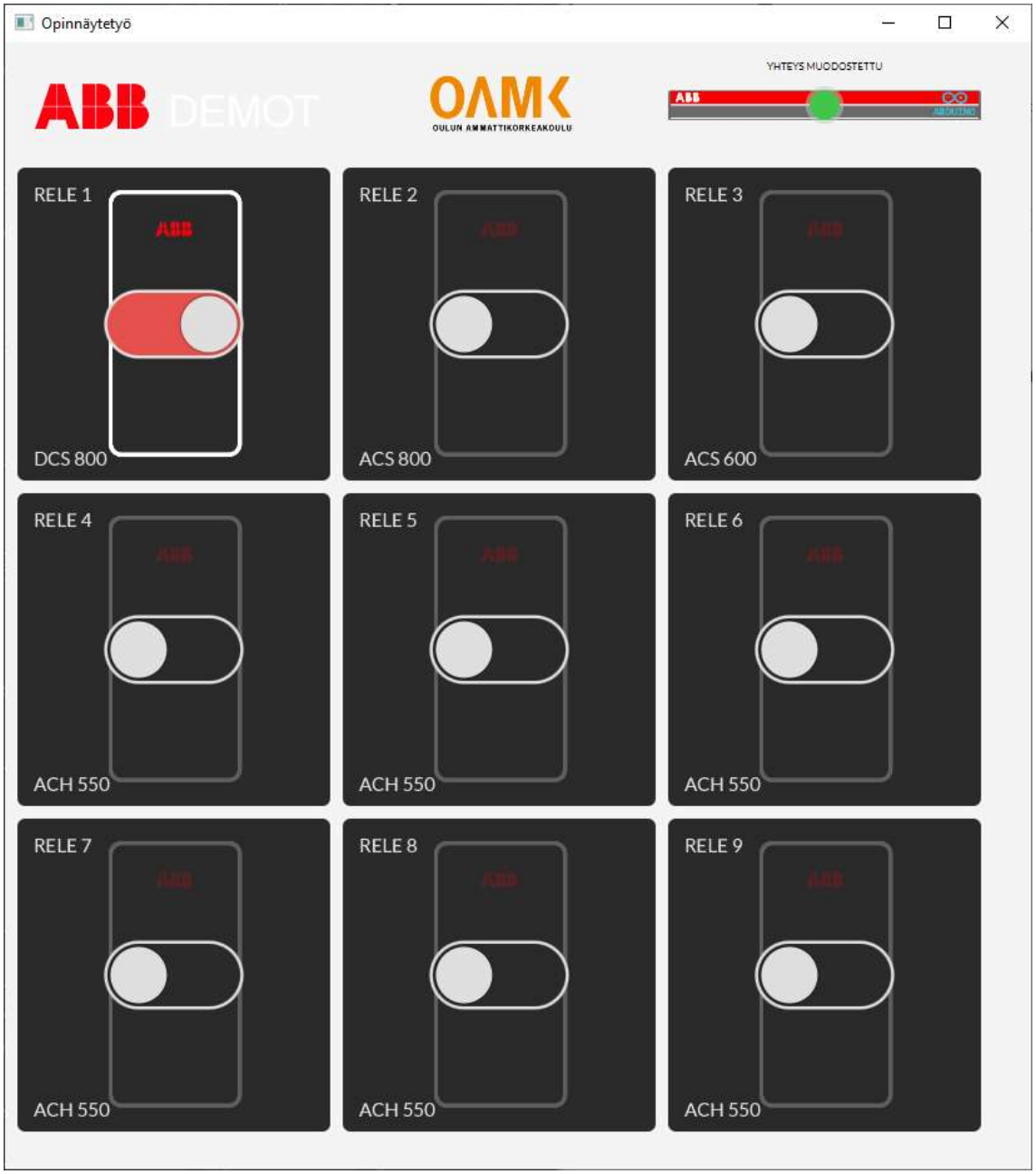
Kuva 17. Sarjaporttiyhteyden tilaa seuraava koodi

Java-sovellus oli tässä vaiheessa toimiva, mutta visuaalisesti hieman karu. Ohjelmaan lisättiin kuvaelementtejä ja niihin toiminnallisuuksia, jotka selventävät edelleen käyttöliittymään releiden ja Arduinon yhteyden tiloja.

Kuvaelementtien lisääminen aloitettiin luomalla Windowsin Paint3D:llä kuva, joka sijoitetaan 'Arduino_Yhteys' -komponenttiin. Kuvan luomisen jälkeen se tallennettiin PNG-muodossa läpinäkyvälle pohjalle varmistaen, ettei Java-sovelluksessa näy kuvan pohjaväriä sovellukseen määritellyn värin päällä. Kuvien lisääminen sovellukseen toteutettiin luomalla kansio sovelluksen sisälle. Eclipsessä tämä tapahtuu valinnoilla File > New > Source Folder, jonka jälkeen valitaan aukeavassa valintaikkunassa Java-sovellus (Project name:) ja nimetään kansio halutulla nimellä (Folder name:). Kansion nimeksi annettiin ’kuvat’ ja tähän kansioon sijoitettiin kaikki kuvatiedostot, joita sovelluksessa käytettiin. Lisääminen tapahtuu valitsemalla Eclipsessä File > New > File, mikä aukaisee saman valintaikkunan kuin uutta kansiota luodessa. Koska kuvat lisättiin erikseen määriteltyyn kansioon, valittiin aikaisemmin luotu ’kuvat’-kansio parent folderiksi ja siirryttiin ”Advanced”-napilla linkittämään haluttu kuva tietokoneelta Java-projektiin. Linkitys toteutettiin raksittamalla ’Link to file in the file system’, jonka jälkeen ”Browse”-napilla päästiin valitsemaan haluttu kuvatiedosto tietokoneen tallennustilasta. Näiden toimintojen jälkeen kuvatiedosto on käytettävissä Java-projektissa. Lisäämällä ’`.backgroundImage(new Image("ABB23.png"))`’ kuvassa 16 esitettyyn Tilen määrittelyyn saatiin kuva lisättyä sovellukseen.

Kuvan läpinäkyvyyttä muutetaan AnimationTimerissa komennolla: ’`Arduino_Yhteys.setBackgroundImageOpacity(1);`’ riippuen sarjaporttiyhteyden tilasta (kuva 17). Yhteyden ollessa kunnossa kuva on täysin näkyvä, kun taas yhteyden puuttuessa kuva on 50 % läpinäkyvä.

Vastaavalla menetelmällä toteutettiin releitä ohjaaville Tileille taustakuva, joka muuttaa läpinäkyvyyttään kytkimen tilan mukaan. Sovellukseen lisättiin myös ABB:n ja Oulun Ammattikorkeakoulun logot, jotka näkyvät käyttöliittymässä. Kuvassa 18 esitetään valmis käyttöliittymä, missä kytkimen aktivointi muuttaa taustakuvan kirkkautta ja Arduinon sarjaporttiyhteyden tila määrittää simuloidun ledin tekstin sekä värin. Sovelluksen koodi on liitteessä 3.



Kuva 18. Valmis JavaFX-sovellus

4.6 Sovelluksen kokoaminen

Kun Java-sovellus oli testattu toimivaksi, se voitiin koota ajettavaksi ohjelmaksi. Kokoaminen tapahtuu Eclipsessä valinnoilla File > Export > Runnable JAR file. Valinta avaa työpöydälle uuden ikkunan, jossa ’Launch configuration:’ -kohdassa etsitään sovellus, joka halutaan koota. ’Export destination:’ määrää, minne koottu ohjelma tallennetaan sekä sen nimen. ’Library handling:’ -valintana toimii ’Package required libraries into generated JAR’, joka sisällyttää tarvittavat ohjelmakirjastot sovellukseen. Kun painetaan ”Finish”-nappia, Eclipse kokoaa ajettavan JAR-tiedoston, joka on valmis käytettäväksi kaikilla laitteilla, joissa on JavaFX tuen omaava ympäristö.

5 TESTAAMINEN

Testaaminen aloitettiin siirtämällä valmisteltu Java-sovellus loppukäyttöön tarkoitetulle tietokoneelle. Koska ohjelma koottiin valmiiksi ajettavana JAR-tiedostona, se voitiin siirtää vapaavalintaiseen kansioon, tämän lisäksi sille luotiin pikakuvake työpöydälle. Windows-käyttöjärjestelmä määrättiin käynnistämään etäkäyttösovellus automaattisesti uuden käyttäjän kirjautuessa sisään.

Java-sovellus käynnistyi halutulla tavalla, mutta sarjaporttiyhteyttä ei saatu luotua. Tämä johtui loppukäyttökoneen ja ohjelmoinnin suorittaneen koneen eroavista sarjaportti nimityksistä. Ongelma ratkaistiin vaihtamalla sarjaportin osoite vastaamaan isäntäkoneen sarjaporttia Eclipsessä ja kokoamalla sovellus uudelleen ohjelmointiin käytetyllä koneella.

Muutoksen jälkeen ohjelma toimi moitteettomasti. Kaikki yhdeksän relettä toimivat halutulla tavalla. Myös turvatoimet osoittautuivat toimiviksi. Näitä testattiin sammuttamalla ohjelma, mistä seurasi kaikkien releiden poiskytkeytyminen ennen sovelluksen sammumista. Ohjelman vakautta on nyt testattu kahden kuukauden ajan jättämällä sovellus auki tietokoneelle ilman minkäänlaisia merkkejä epävakaudesta tai yhteyskatkoksista. Loppukäyttöön tarkoitetulle koneelle kirjauduttiin myös useita kertoja etätyöpöytäyhteyden kautta ja sovellus toimi edelleen halutulla tavalla.

6 LOPPUPÄÄTELMÄ

Java-sovellus toimii hyvin nykyisillä asetuksillaan ja toiminnoillaan. JavaFX:n kirjastot tarjoavat hyvät työkalut tyylikkään käyttöliittymän toteutukseen. Sovelluksen toteutus vaati varsinkin kokemattomalta ohjelmoijalta melko paljon, mutta lopputulos oli vaivan arvoinen.

Ohjelmoinnin ja testauksen jälkeen on syntynyt kehitysideoita, joita lisätään sovellukseen, mikäli niille nähdään tarvetta. Ensimmäinen lisäys olisi sarjaporttiyhteyden muuttaminen yksisuuntaisesta kaksisuuntaiseksi. Tässä tapauksessa sovellus myös kuuntelisi sarjaporttia, johon releitä ohjaava Arduino-mikrokontrolleri ilmoittaisi relekomennon tilan, josta seuraisi käyttöliittymässä tilanmuutoksia. Tällä hetkellä tilanmuutokset käyttöliittymään toteutuvat, kun sarjaporttiin lähetetään relekomento. Toinen lisäys olisi nappi tai kytkin Tile-komponentteihin, jotka aktivoimalla käyttöliittymään tulisi tieto siitä, onko kyseinen demo-taajuusmuuttaja paikallaan ja valmis etäkäytettäväksi vai onko se viety pois paikanpäältä.

LÄHTEET

1. Vesterholm, Mika & Kyppö, Jorma. 2015. Java ohjelmointi. Helsinki: Talentum Media Oy ja tekijät.

2. Sharan, K. & Späth, P. 2022. Learn JavaFX 17: Building user experience and interfaces with Java. Second edition. United States: Apress Media. Hakupäivä 14.1.2024. Oula-Finna – kirjakokoelma. Vaatii käyttöoikeuden.

3. Grunwald, Gerrit 2016. TilesFX. Hakupäivä 19.1.2024. https://harmoniccode.blogspot.com/2016/12/tilesfx.html.

4. Github. 2024. TilesFX. Hakupäivä 12.2.2024. https://hansolo.github.io/tilesfx/.

5. Fazecast Inc. jSerialComm. Hakupäivä 19.1.2024. https://github.com/Fazecast/jSerialComm/wiki/Library-Features.

6. Eclipse Foundation. What is Eclipse. Hakupäivä 19.1.2024. https://www.eclipse.org/home/whatis/.

7. Maasto-Matti. 2023. Arduino – kaikki mitä sinun tulee tietää. Hakupäivä 5.1.2024. Arduino – kaikki mitä sinun tulee tietää | Maastomatti.

```
void setup()
{
  Serial.begin(9600);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);

}
//--------------------------
void loop()
{
  byte JavaFX;
  if(Serial.available())
  {
    JavaFX = Serial.read();
    if(JavaFX == 10){
      digitalWrite(2,LOW);
      }
    else if(JavaFX == 11){
      digitalWrite(2,HIGH);
      }
    else if(JavaFX == 20){
      digitalWrite(3,LOW);
      }
    else if(JavaFX == 21){
      digitalWrite(3,HIGH);
      }
    else if(JavaFX == 30){
      digitalWrite(4,LOW);
      }
    else if(JavaFX == 31){
      digitalWrite(4,HIGH);
      }
    else if(JavaFX == 40){
      digitalWrite(5,LOW);
      }
    else if(JavaFX == 41){
      digitalWrite(5,HIGH);
      }
    else if(JavaFX == 50){
      digitalWrite(6,LOW);
      }
    else if(JavaFX == 51){
      digitalWrite(6,HIGH);
      }
    else if(JavaFX == 60){
      digitalWrite(7,LOW);
      }
    else if(JavaFX == 61){
      digitalWrite(7,HIGH);
      }
    else if(JavaFX == 70){
      digitalWrite(8,LOW);
      }
    else if(JavaFX == 71){
      digitalWrite(8,HIGH);
      }
    else if(JavaFX == 80){
      digitalWrite(9,LOW);
      }
    else if(JavaFX == 81){
      digitalWrite(9,HIGH);
      }
    else if(JavaFX == 90){
      digitalWrite(10,LOW);
      }
    else if(JavaFX == 91){
      digitalWrite(10,HIGH);
      }
  }
}
```

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>oppari</groupId>
    <artifactId>opaeta</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-controls</artifactId>
            <version>13</version>
        </dependency>
        <dependency>
            <groupId>com.fazecast</groupId>
            <artifactId>jSerialComm</artifactId>
            <version>2.7.0</version>
        </dependency>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-web</artifactId>
            <version>16</version>
        </dependency>
        <dependency>
            <groupId>eu.hansolo</groupId>
            <artifactId>tilesfx</artifactId>
            <version>11.48</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
                <configuration>
                    <release>11</release>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.openjfx</groupId>
                <artifactId>javafx-maven-plugin</artifactId>
                <version>0.0.6</version>
                <executions>
                    <execution>
                        <!-- Default configuration for running -->
                        <!-- Usage: mvn clean javafx:run -->
                        <id>default-cli</id>
                        <configuration>
                            <mainClass>oppari.opaeta.App</mainClass>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

POM.XML LIITE 3

```
package oppari.opaeta;

import java.io.IOException;
import com.fazecast.jSerialComm.SerialPort;
import eu.hansolo.tilesfx.Tile;
import eu.hansolo.tilesfx.TileBuilder;
import eu.hansolo.tilesfx.colors.Bright;
import javafx.animation.AnimationTimer;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;



public class App extends Application {
    public static SerialPort arduino = SerialPort.getCommPort("COM11");
    static      Integer     sammutus = 10;
    static      Integer     rele1arvo = 10;
    static      Integer     rele1old = 11;
    //LISÄÄ RELEET 1-9-------------------------------------------------
    @Override
    public void start(Stage stage) {

            Tile rele1 = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.SWITCH)
                .title("RELE 1")
                .text("DCS 800")
                .backgroundImage(new Image("demo_1.png"))
                .backgroundImageOpacity(0.5)
                .activeColor(Tile.RED)
                .build();

            Tile rele2 = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.SWITCH)
                .title("RELE 2")
                .text("ACS 800")
                .backgroundImage(new Image("demo_1.png"))
                .backgroundImageOpacity(0.5)
                .backgroundImageKeepAspect(true)
                .activeColor(Tile.RED)
                .build();

            Tile rele3 = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.SWITCH)
                .title("RELE 3")
                .text("ACS 600")
                .backgroundImage(new Image("demo_1.png"))
                .backgroundImageOpacity(0.5)
                .backgroundImageKeepAspect(true)
                .activeColor(Tile.RED)
                .build();

            Tile rele4 = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.SWITCH)
                .title("RELE 4")
                .text("ACH 550")
                .backgroundImage(new Image("demo_1.png"))
                .backgroundImageOpacity(0.5)
                .backgroundImageKeepAspect(true)
                .activeColor(Tile.RED)
                .build();

            Tile rele5 = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.SWITCH)
                .title("RELE 5")
                .text("ACH 550")
                .backgroundImage(new Image("demo_1.png"))
                .backgroundImageOpacity(0.5)
                .backgroundImageKeepAspect(true)
                .activeColor(Tile.RED)
                .build();

            Tile rele6 = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.SWITCH)
                .title("RELE 6")
                .text("ACH 550")
                .backgroundImage(new Image("demo_1.png"))
                .backgroundImageOpacity(0.5)
                .backgroundImageKeepAspect(true)
                .activeColor(Tile.RED)
                .build();

            Tile rele7 = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.SWITCH)
                .title("RELE 7")
                .text("ACH 550")
                .backgroundImage(new Image("demo_1.png"))
                .backgroundImageOpacity(0.5)
                .backgroundImageKeepAspect(true)
                .activeColor(Tile.RED)
                .build();

            Tile rele8 = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.SWITCH)
                .title("RELE 8")
                .text("ACH 550")
                .backgroundImage(new Image("demo_1.png"))
                .backgroundImageOpacity(0.5)
                .backgroundImageKeepAspect(true)
                .activeColor(Tile.RED)
                .build();

            Tile rele9 = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.SWITCH)
                .title("RELE 9")
                .text("ACH 550")
                .backgroundImage(new Image("demo_1.png"))
                .backgroundImageOpacity(0.5)
                .backgroundImageKeepAspect(true)
                .activeColor(Tile.RED)
                .build();

            Tile abblogo = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.IMAGE)
                .prefSize(200, 80)
                .backgroundColor(Color.TRANSPARENT)
                .image(new Image("ABBdemot.png"))
                .build();

            Tile oamklogo = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.IMAGE)
                .prefSize(200, 80)
                .backgroundColor(Color.TRANSPARENT)
                .image(new Image("oamk.png"))
                .build();

            Tile Arduino_Yhteys = TileBuilder.create()
                .skinType(eu.hansolo.tilesfx.Tile.SkinType.LED)
                .prefSize(200, 80)
                .backgroundColor(Color.TRANSPARENT)
                .textSize(Tile.TextSize.BIGGER)
                .descriptionAlignment(Pos.CENTER)
                .textVisible(true)
                .descriptionColor(Color.BLACK)
                .backgroundImage(new Image("ABB23.png"))
                .build();

            Arduino_Yhteys.setActive(!Arduino_Yhteys.isActive());

            AnimationTimer timer = new AnimationTimer() {
                @Override
                public void handle(long now) {
                    //RELE1----KOPIOI JA MUOKKAA RELEILLE 2-9--------------------
                    if(rele1.isActive()) {
                        rele1arvo=11;
                      }
                    else if(!rele1.isActive()) {
                        rele1arvo=10;
                      }
                    if (rele1.isActive() && (!rele1arvo.equals(rele1old))) {
                        rele1old = rele1arvo;
                        rele1.setBackgroundImage(new Image("demo_1.png"));
                        rele1.setBackgroundImageOpacity(1);
                        try {
                            arduino.getOutputStream().write(rele1arvo.byteValue());
                        } catch (IOException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }
                    else if(!rele1.isActive() && (!rele1arvo.equals(rele1old))) {
                        rele1old = rele1arvo;
                        rele1.setBackgroundImage(new Image("demo_1.png"));
                        rele1.setBackgroundImageOpacity(0.5);
                        try {
                            arduino.getOutputStream().write(rele1arvo.byteValue());
                        } catch (IOException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }
                    //ARDUINO YHTEYDEN TILA--------------------------------------
                    if(!arduino.openPort()) {
                        Arduino_Yhteys.setActiveColor(Bright.RED);
                        Arduino_Yhteys.setDescription("ARDUINOON EI YHTEYTTÄ    VARMISTA USB-JOHTO");
                        Arduino_Yhteys.setBackgroundImageOpacity(0.5);
                    }
                    else if(arduino.openPort()) {
                        Arduino_Yhteys.setActiveColor(Bright.GREEN);
                        Arduino_Yhteys.setDescription("YHTEYS MUODOSTETTU");
                        Arduino_Yhteys.setBackgroundImageOpacity(1);
                    }
                    }
            };

            GridPane Paneli = new GridPane();
            Paneli.add((abblogo),0,0);
            Paneli.add((oamklogo),1,0);
            Paneli.add((Arduino_Yhteys),2,0);
            Paneli.add((rele1), 0, 1);
            Paneli.add((rele2), 1, 1);
            Paneli.add((rele3), 2, 1);
            Paneli.add((rele4), 0, 2);
            Paneli.add((rele5), 1, 2);
            Paneli.add((rele6), 2, 2);
            Paneli.add((rele7), 0, 3);
            Paneli.add((rele8), 1, 3);
            Paneli.add((rele9), 2, 3);
            Paneli.setPadding(new Insets(10, 10, 10, 10));
            Paneli.setHgap(10);
            Paneli.setVgap(10);
            var scene = new Scene(Paneli, 820, 900);
            stage.setResizable(true);
            stage.setScene(scene);
            stage.setFullScreen(false);
            stage.setTitle("Opinnäytetyö");
            stage.show();
            stage.setOnCloseRequest(e -> {
                while(sammutus <= 90) {
                try {
                    arduino.getOutputStream().write(sammutus.byteValue());
                    sammutus +=10;
                    Thread.sleep(40);
                } catch (IOException | InterruptedException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                    Platform.exit();
                    System.exit(0);
                }
                }
                Platform.exit();
                System.exit(0);
                }
                );
            timer.start();
    }



    public static void main(String[] args) {
        arduino.setComPortParameters(9600, 8, 1, 0);
        arduino.setComPortTimeouts(SerialPort.TIMEOUT_WRITE_BLOCKING, 0, 0);
        Launch();
    }

}
```