LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

RFID-teknologia tuotantolinjastossa

Tuotteiden yksilöinti ja seuranta yrityksessä

Kari-Finn Oy

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2018
Henri Korkeila

Lahden ammattikorkeakoulu
Tietotekniikka

KORKEILA, HENRI:

RFID-teknologia tuotantolinjastossa
Tuotteiden yksilöinti ja seuranta
yrityksessä Kari-Finn Oy

Ohjelmistotekniikan opinnäytetyö, 49 sivua

Kevät 2018

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli tutkia, kuinka opinnäytetyön toimeksiantajana toimivan Kari-Finn Oy:n olisi mahdollista ottaa käyttöön moderni tapa yksilöidä ja seurata tuotteitaan hyödyntämällä RFID-teknologiaa. Tätä yksilöinti- ja seurantatapaa olisi tarkoitus hyödyntää tuotantolinjaston tapahtumien seurantaan sekä reklamaatioiden yhteydessä. RFID-teknologialla toteutetun järjestelmän hyötyjä olisi, että valmistettavista tuotteista voitaisiin kerätä tuotteen valmistukseen liittyviä tietoja. Tutkimalla näitä tietoja yrityksen olisi mahdollista kehittää valmistusprosessiaan.

Varsinaiseksi tavoitteeksi muodostui sovelluksen kehittäminen käsikäyttöiselle RFID-lukijalle. Yrityksen vaatimus sovellukselle oli, että käyttöliittymä olisi mahdollisimman yksinkertainen. Sovellus toteutettiin RFID-lukijalle, jonka käyttöjärjestelmänä toimii Windows CE 6.0. Kehitysympäristönä toimi Microsoft Visual Studio 2008, ohjelmointikielenä C#, ja sovelluksen alustana oli .NET Compact Framework. Tietokannaksi valittiin MySQL ja tämän tietokannan käsittelyn helpottamiseksi käytettiin MySQL Connector/Net -adapteria. RFID-lukijan toimintojen käsittelyssä käytettiin laitevalmistajan kehittämää ohjelmointirajapintaa MHL.

Suurimmaksi haasteeksi muodostui työssä käytetyn RFID-lukijan pieniresoluutioinen näyttö, joka puolestaan hankaloitti yksinkertaisen käyttöliittymän visuaalista suunnittelemista. Kehitystyön tuloksena oli käyttövalmis sovellus, joka vastasi yrityksen asettamia vaatimuksia.

Asiasanat: RFID, Windows CE, .NET Compact Framework

Lahti University of Applied Sciences
Degree Programme in Information Technology

KORKEILA, HENRI:

RFID technology on an assembly line
Identification and tracking of products in Kari-Finn Oy

Bachelor's Thesis in software engineering, 49 pages

Spring 2018

ABSTRACT

The aim of the thesis was to study how it would be possible for the target company of this thesis, Kari-Finn Oy, to introduce a modern way of identifying and tracking their products by utilizing RFID technology. This approach of identification and tracking would be used to monitor events of the assembly line, and in connection with customer complaints. The benefit of a system implemented with RFID technology is the possibility to collect information related to the manufacture of the products being manufactured. By analyzing the information, it would be possible for the company to develop its manufacturing process.

The actual goal of the thesis was to develop an application for a handheld RFID reader. The company's requirement for the software was to make the user interface as simple as possible. The application was implemented for an RFID reader that is running the Windows CE 6.0 operating system. The development environment used was Microsoft Visual Studio 2008. The programming language used was C#, and the application is powered by the .NET Compact Framework platform. MySQL was selected as the database, and MySQL Connector/Net adapter was used to facilitate the handling of the database. To be able to use the functions of the RFID reader, a programming interface called MHL, developed by the manufacturer of the reader, was used.

The biggest challenge was working with the low-resolution display of the RFID reader, which in turn made it harder to make the visual design of a simple user interface. The result of the development work was a ready-to-use application that met the demands set by the company.

Key words: RFID, Windows CE, .NET Compact Framework

SISÄLLYS

1 JOHDANTO

Opinnäytetyö tehtiin yritykselle nimeltä Kari-Finn Oy, jonka emoyhtiönä toimii KFR Hollola Oy. Kari-Finn Oy valmistaa pinnansäätölaitteita, joista päätuotteena on pintakytkin. Pintakytkin on pääsääntöisesti kaapelinsa varassa roikkuva ja nestepinnan vaihteluiden mukaan kallistuva muovikelluke. Kellukkeen sisällä on kytkinelementtejä, jotka kallistuman mukaan vaihtavat tilaansa. Pintakytkintä käytetään nestepinnan tarkkailuun ja säätelyyn. Yritys valmistaa myös muita sähköelektronisia tuotteita, joita ovat muun muassa elektrodikytkimet, siilokytkimet, valopintakytkimet, vesitiiviit pumppukaapeliliittimet sekä pumppaamoiden ohjauskeskukset.

Pintakytkimiä yritys valmistaa vuodessa noin 50 000 kappaletta. Tuotteiden vienti on maailmanlaajuista. Yrityksen pääkonttori ja tehdas sijaitsevat Lahdessa. Kari-Finn Oy ja KFR Hollola Oy työllistävät yhteensä noin 15 henkilöä. Vuonna 2017 yrityksen liikevaihto oli noin 1,2 miljoonaa euroa.

Valmistettavien pintakytkimien yksilöinti tuotannossa tapahtuu tulostamalla lasermerkintälaitteella pintakytkimen kanteen mallikohtainen juokseva sarjanumero sekä viikkonumero. Lasermerkattua sarjanumeroa ei kirjata mihinkään järjestelmään ylös, eikä siihen tällöin liitetä muutakaan valmistukseen liittyvää tietoa.

Lahden ammattikorkeakoulun mekatroniikan opiskelijat selvittivät vuonna 2009, että yritys voisi hyödyntää tuotannossaan RFID-teknologiaa tuotteiden yksilöinnissä sekä seurannassa. Mekatroniikan opiskelijat olivat myös tutkineet ja valinneet tarkoitukseen sopivan RFID-lukijan sekä RFID-tunnisteet. RFID-tunnistetta käyttämällä voidaan kerätä tuotannon vaiheista tietoja tuotteesta, ja nämä tiedot voidaan tallentaa keskitettyyn paikkaan, tietokantaan. Yrityksen olisi tällöin mahdollista tarkistaa muun muassa reklamaatioiden yhteydessä tuotteessa käytetyt komponentit, tuotteen valmistusaika sekä tuotannon työntekijät. RFID-tunnistetta sekä kerättyjä tietoja voitaisiin myös hyödyntää tuotannossa koestuksen

yhteydessä.

Projektin tavoitteena oli suunnitella ja toteuttaa ohjelmistopohjainen ratkaisu tuotteiden yksilöintiin käyttäen mekatroniikka opiskelijoiden tekemän selvityksen mukaan valittuja tunnisteita sekä käsikäyttöistä lukijaa. Tutkimusongelma rajattiin sovelluksen toteuttamiseen annetulle laitteistolle, käyttäen lukijan ohjelmointirajapintaa sekä haluttua tietokantaa, joka oli tässä työssä MySQL.

Tässä opinnäytetyössä kerrotaan lyhyesti RFID-teknologiasta, Microsoftin .NET Compact Frameworkista, käytetystä MySQL-tietokannasta, tietokannan käsittelyyn liittyvästä MySQL Connectorista, työssä käytetyistä käsikäyttöisistä RFID-lukijoista, RFID-lukijoiden ohjelmointirajapinnasta sekä RFID-tunnisteesta. Toteutuksessa kerrotaan käsikäyttöiselle RFID-lukijalle kehitetyn ohjelmiston suunnittelusta ja toteutuksesta.

2 TOIMINTAYMPÄRISTÖ

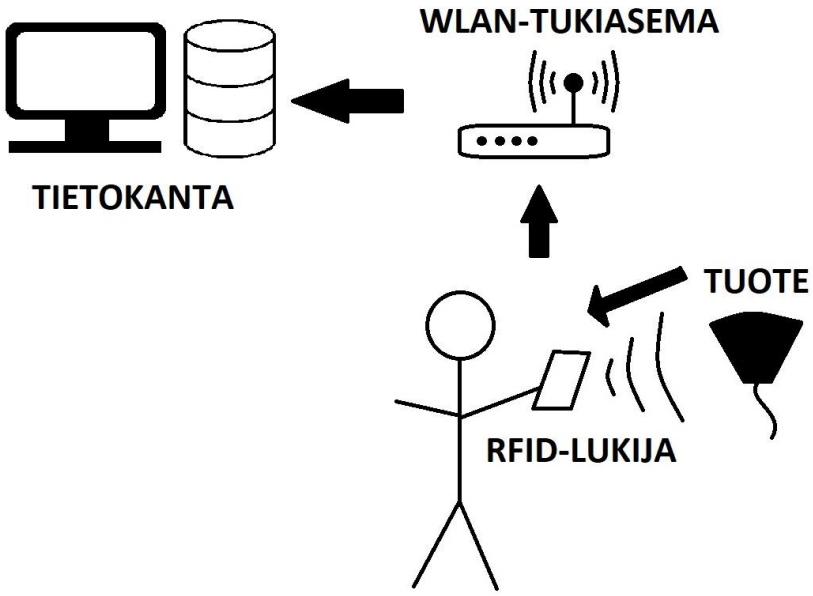
2.1 Yrityksen aikaisemmin käytössä ollut tuotteiden yksilöinti

Yrityksen tuotannon vaiheisiin kokoonpanolinjastolla kuuluvat kaapelin katkaisu haluttuun mittaan, kaapelin kiinnitys muovikartioon, johdon päiden kiinnitys, kytkinrasian tai- rasioiden ja painojen asentaminen kartioon, kannen merkitseminen laserilla, kannen kiinnittäminen, kiinnityksen siistiminen, tuotteen koestus sekä viimeisenä tuotteen pakkaus. Yrityksen tuotteiden yksilöinti on tapahtunut laserilla merkitsemällä kanteen mallinumero, mallikohtainen juokseva sarjanumero sekä valmistusviikko.

2.2 RFID-teknologialla toteutettu tuotteiden yksilöinti

RFID-teknologialla toteutetun tuotteiden yksilöinnin ja seurannan toimintaperiaate on seuraavanlainen. Tuotannossa kiinnitetään RFID-tunniste tuotteen runkoon sisäpuolelle, tuotteen ollessa noin puolivälissä tuotantolinjastoa, kytkinrasioiden asentamisen yhteydessä. Tuotteeseen kiinnitetyn RFID-tunnisteen sarjanumero luetaan käsikäyttöisellä lukijalla. Tämän jälkeen kerätään viivakoodeja lukemalla tiedot valmistettavasta tuotteesta samalla laitteella. Nämä yhteen liitetyt tiedot tallennetaan yrityksen tietokantaan.

Tallentamiseen tarvittava verkkoyhteys on toteutettu langattomasti käyttämällä käsikäyttöisen lukijan langatonta verkkoyhteyttä sekä sijoittamalla yrityksen tehdastilaan wlan-tukiasemia. Tietokantaan kerättyjä tietoja hyödynnetään tuotannon seurannassa yleisesti sekä etenkin asiakaspalautuksien yhteydessä. Tietokantaan kerättyjä tietoja tuotteista voidaan hyödyntää myös tuotantolinjaston lopussa koestuksessa. Kuviossa 1 on tuotteiden yksilöinnin ja seurannan toimintaympäristö yksinkertaistetussa muodossa.



Kuvio 1. Tuotteiden yksilöinnin ja seurannan toimintaympäristö

2.3 Asiakasvaatimukset

Yrityksen vaatimuksena projektille oli käsikäyttöiselle RFID-lukijalle kehitettävän sovelluksen viimeistely mahdollisimman helppokäyttöiseen muotoon kehityksen aikana kerättyjen käyttökokemusten perusteella. Kehitetyn järjestelmän tarkoituksena olisi minimoida virheiden määrä tuotannossa, samalla kehittäen tuotteiden laatua ja seurattavuutta. Kehitetyn järjestelmän tarkoituksena olisi myös antaa yritykselle valmiuksia tuotannon kehittämiseen vaativampia standardiluokituksia kohti ja näin ollen parantaa kansainvälistä kilpailukykyä. Samalla voitaisiin mahdollistaa tuotantomäärien kasvattaminen. Käyttöönottamalla RFID-teknologiaa hyödyntävän järjestelmän voitaisiin luopua tuotteen kanteen lasermerkatusta sarjanumerosta ja valmistusviikosta.

3 RADIO FREQUENCY IDENTIFICATION

3.1 Yleisesti

RFID on lyhenne termistä radio frequency identification, eli suomeksi radiotaajuinen etätunnistus. RFID on yleisnimitys tekniikoille, joissa radiotaajuuksia käytetään ihmisten, eläinten sekä esineiden automaattiseen tunnistamiseen ja seurantaan. RFID-järjestelmiin kuuluu kolme komponenttia: RFID-tunniste, RFID-lukija sekä taustajärjestelmä, johon sisältyvät viestinnän infrastruktuuri ja tietoa hallinnoiva ohjelmisto. (RFID Lab Finland ry 2015.)

3.2 Historia

RFID-teknologian kehitys alkoi toisen maailmansodan aikana. Tuolloin kaikkien käytössä oli uudehko tutkajärjestelmä, jonka Sir Robert Alexander Watson-Wattsin keksi vuonna 1935. Tutka pystyi varoittamaan lähestyvistä lentokoneista, jotka olivat vielä kilometrien päässä. Ei ollut kuitenkaan olemassa mitään tapaa tunnistaa, mitkä lentokoneista olivat vihollisen ja mitkä omia kotiin palaavia lentokoneita. Saksalaiset keksivät, että jos pilotit pyöräyttivät lentokoneensa ympäri palatessaan takaisin tukikohtaan, lentokoneen heijastama radiosignaali muuttuisi. Tätä yksinkertaista menetelmää käyttämällä tutkamiehistöt pystyivät erottamaan, oliko kyseessä saksalainen lentokone vai liittoutuneiden lentokone. Tämä oli käytännössä ensimmäinen passiivinen RFID-järjestelmä. (Roberti 2005; Recall 2015.)

Britit kehittivät Watson-Wattsin johdolla ensimmäisen aktiivisen lentokoneiden tunnistautumisjärjestelmän IFF (identify friend or foe). Jokaiseen brittiläiseen lentokoneeseen laitettiin lähetin. Vastaanottaessaan signaalin tutka-asemilta maanpinnalta lähetin alkoi lähettää signaalia takaisin, joka kertoi, että kyseessä on omien joukkojen kone. (Roberti 2005; Recall 2015.)

50-60-luvulla teknologia kehittyi edelleen, kun tutkittiin uusia käyttökohteita joissa teknologiaa voisi hyödyntää muiden esineiden tunnistamiseen etäältä. Yritykset alkoivat kaupallistamaan varkaudenestojärjestelmiä, jotka käyttivät radioaaltoja määrittääkseen, oliko tuote maksettu vai ei. EAS-tunnisteet (electronic article surveillance), jotka ovat tänäkin päivänä käytössä, sisältävät 1-bittisen tunnisteen. Bitti on joko päällä tai pois. Asiakkaan maksaessa tuotteen kassalla, bitti käännetään pois päältä ja asiakas voi poistua liikkeestä. Mikäli asiakas yrittää poistua liikkeestä maksamatta, lukijaportti ovella havaitsee tunnisteen ja alkaa hälyttää. (Roberti 2005; Recall 2015.)

Ensimmäinen RFID-patentti on vuodelta 1973, jonka sai Mario W. Cardullo. Patentti on aktiiviselle tunnisteelle, jossa oli uudelleenkirjoitettava muisti. Samana vuonna yhdysvaltalainen yrittäjä Charles Walton sai patentin passiiviselle transponderille, jolla pystyi avaamaan oven käyttämättä avainta. (Roberti 2005; Recall 2015.)

Teknologiaa alettiin käyttää useammalla eri tavalla. Sitä käytettiin muun muassa ydinmateriaalia kuljettavien rekkojen tunnistamiseen, näiden kulkiessa turvallisuusalueiden automaattisista porteista. Tätä ideaa käytettiin myöhemmin kaupallisessa mielessä, kun automaattisia tietullimaksupalveluita alettiin kehittää 1980-luvun puolivälissä. RFID-teknologiaa alettiin käyttää myös lehmien merkitsemiseen, mikä puolestaan helpotti lääkeannosten tasojen ja määrien seurantaa. (Roberti 2005; Recall 2015.)

3.3 RFID-järjestelmän toimintaperiaate

RFID-lukija lukee tunnisteen langattomasti radioaaltojen avulla. RFID-teknologia on hyvin verrattavissa viivakoodiin. RFID-tunniste kiinnitetään kohteeseen, jota halutaan seurata, siinä missä viivakoodikin. Toisin kuin viivakoodi, RFID-tunniste ei vaadi suoraa näköyhteyttä lukijaan, ja tämän takia RFID-tunniste on mahdollista sisällyttää tuotteen sisään. (RFID Lab Finland ry 2015.)

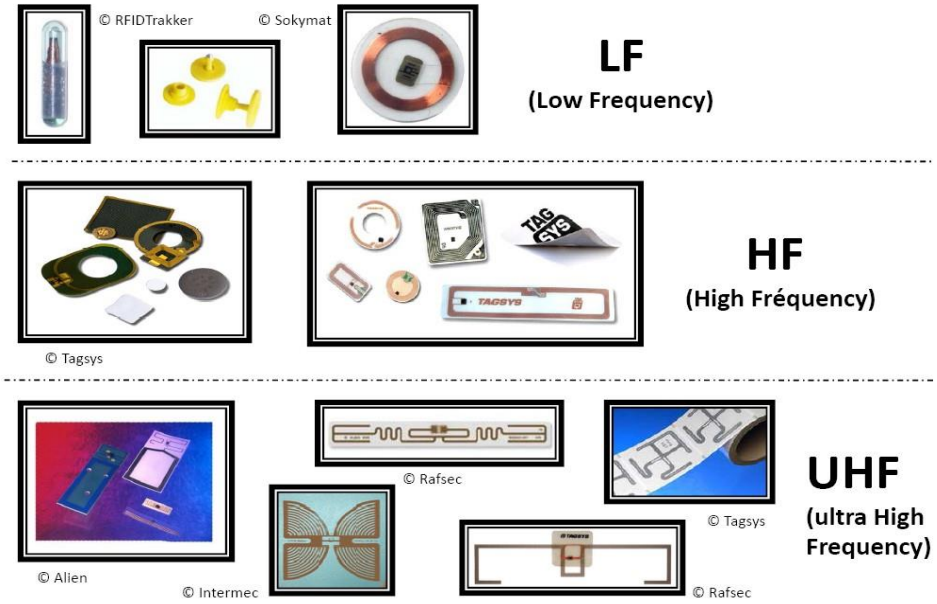
Voidakseen kommunikoida keskenään tunnisteen ja lukijan pitää kytkeytyä toisiinsa jollakin tapaa. Näistä kytkentätavoista käytetyimmät ovat matalilla taajuusalueilla käytetty induktiivinen kytkentä (inductive coupling) sekä korkeammilla taajuusalueilla käytetty sähkömagneettinen kytkentä (radiative coupling / backscatter). Induktiivinen kytkentä tapahtuu niin sanotussa lähikentässä (near-field). Induktiivinen kytkentä perustuu lukijan synnyttämään magneettikenttään, jonka lukija ja tunniste jakavat. Tästä jaetusta magneettikentästä tunniste saa tarvitsemansa energian aktivoituakseen. Tunniste kommunikoi lukijan kanssa vaihtelemalla kuinka paljon tunniste kuormittaa antenniaan. Käytettäessä induktiivista kytkentää lukijan ja tunnisteen välinen lukuetäisyys on tyypillisesti kosketuksesta yhteen metriin. (Smiley 2016; RFID4U 2018b.)

Sähkömagneettinen kytkentä puolestaan toimii lähikentän ulkopuolella, niin sanotussa kaukokentässä (far-field). Sähkömagneettisessa kytkennässä tunnisteen antenni vastaanottaa lukijan lähettämän radiosignaalin sähkömagneettiset aallot, joista passiivinen tunniste saa tarvitsemansa energian aktivoituakseen. Aktivoiduttuaan tunniste vastaa heijastamalla osan lukijan lähettämästä radiosignaalista takaisin lukijalle. Sähkömagneettista kytkentää käytettäessä etäisyydet lukijan ja tunnisteen välillä ovat tyypillisesti yhdestä metristä neljään metriin. (Smiley 2016; RFID4U 2018b.)

3.4 Tunnisteet

Tunniste sisältää mikrosirun, jonka muistiin on tallennettu uniikki sarjanumero. Tunniste sisältää myös antennin, joka on yhdistetty mikrosiruun. Tunnisteita on olemassa erikokoisia ja -muotoisia, riippuen käyttökohteesta ja ympäristöstä, missä tunnisteita käytetään. On kortteja, tarroja, erilaisia avaintageja, nappeja, implantteja sekä muita esineitä, joihin on sisällytetty RFID-tunniste. Riippuen tunnisteen luokituksesta, tunnisteen muisti voi olla tyypiltään vain-luku (read-only), uudelleenkirjoitettava (read/write) tai kerran kirjoitettava (write-once). Vain-luku-tyyppisen tunnisteen muistia ei ole mahdollista muokata, koska se

sisältää ainoastaan jo siihen valmistuksessa sisällytettyä tietoa. Uudelleenkirjoitettavan tunnistetyypin muistiin voidaan kirjoittaa toistuvasti. Kerran kirjoitettavan tunnistetyypin muistiin on mahdollista kirjoittaa vain yhden kerran, mutta tunnistetta voidaan kuitenkin lukea monia kertoja. RFID-tunnisteita on olemassa kolmen tyyppisiä: passiivisia, puolipassiivisia sekä aktiivisia. Puolipassiivisessa tunnisteessa on yhdistelmä aktiivisen ja passiivisen tunnisteen ominaisuuksia. Kuvassa 1 näkyy eri taajuusalueita käyttäviä RFID-tunnisteita. (Ahsan, Shah & Kingston 2010, 2 - 3; IEEE GlobalSpec 2018.)



Kuva 1. RFID-tunnisteita (CNRFID 2018)

Passiivinen tunniste vaatii toimiakseen RFID-lukijan radiosignaalin, jonka sähkömagneettiset aallot toimivat tunnisteen virtalähteenä. Verrattaessa aktiivisiin tunnisteisiin passiiviset tunnisteet voivat olla hyvin pieniä, niitä on edullisempi valmistaa, eikä niitä tarvitse ajoittain vaihtaa tai huoltaa. Passiiviset tunnisteet vaativat kuitenkin lukijan signaalin voimakkuudelta aktiivisia tunnisteita enemmän, mikä tarkoittaa myös, että passiivisten

tunnisteiden lukuetäisyydet ovat huomattavasti pienempiä. (IEEE GlobalSpec 2018.)

Aktiivinen tunniste eroaa passiivisesta tunnisteesta siten, että aktiivinen tunniste sisältää oman virtalähteensä, joka on yleensä paristo tai akku. Tämän virtalähteen avulla aktiivinen tunniste voi olla koko ajan yhteydessä RFID-lukijaan toimiessaan itse lähettimenä, jolloin lukuetäisyydet tunnisteen ja lukijan välillä voivat olla jopa yli 100 metriä. Aktiiviset tunnisteet ovat kuitenkin huomattavasti kalliimpia, ja ne vaativat aika ajoin huoltoa virtalähteen vaihtamisen muodossa. (IEEE GlobalSpec 2018.)

3.5 Lukijat

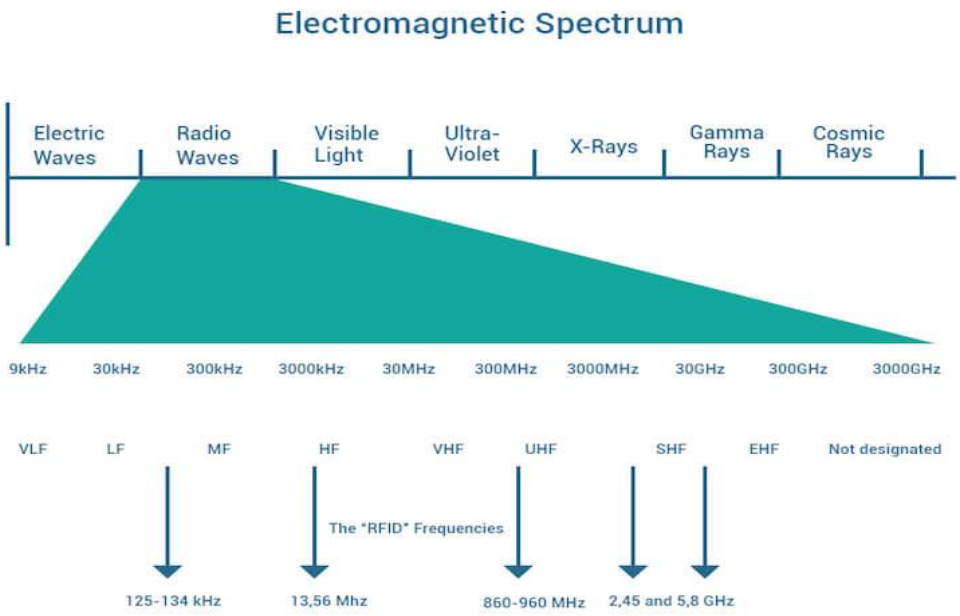
Lukija on RFID-järjestelmissä laite, jonka avulla kommunikoidaan tunnisteiden kanssa. Lukija sisältää yhden tai useamman antennin, joiden avulla lukija voi lähettää ja vastaanottaa radioaaltoja. Passiivisia tunnisteita luettaessa lukija kommunikoi lähettämällä signaalin tunnisteelle, johon tunniste vastaa palauttamalla vastauksen lukijan kyselyyn. (Violino 2005.)

Koska aktiiviset tunnisteet sisältävät oman lähettimen, voivat aktiiviset tunnisteet tarvittaessa olla jatkuvasti yhteydessä lukijaan. Kuvasta 2 löytyy erilaisia RFID-lukijoita. Kuvassa on käsikäyttöisiä lukijoita sekä kiinteästi asennettavia lukijoita. (Violino 2005.)

Kuva 2. Erilaisia RFID-lukijoita (Achievetec 2014)

3.6 Taajuusalueet

RFID-järjestelmät toimivat monilla eri taajuusalueilla. Käytetty taajuusalue vaikuttaa tunnisteen käyttötarkoituksen soveltuvuuteen. Taajuusaluetta valittaessa rajaavia tekijöitä ovat muun muassa maakohtaiset radiotaajuuksia koskevat säädökset, tunnisteen ja lukijan väliset lukuetäisyydet, tunnisteiden luku- ja tiedonsiirtonopeudet, tunnisteiden fyysinen koko, tunnisteiden toimintaympäristö sekä tunnisteiden hinta. RFID-teknologian käyttämä taajuusalue on esitelty kuviossa 2. (RFID4U 2018a.)



Kuvio 2. RFID taajuusalueet (RFID4U 2018c)

LF (low frequency) -taajuusalue kattaa taajuudet 30 - 300 kilohertsiä (kHz), mutta LF-taajuusaluetta käyttävissä RFID-järjestelmissä yleisesti käytetyt taajuudet ovat 125 kHz ja 134 kHz. Koska taajuusaluetta käyttävät tunnisteet ovat passiivisia, lukuetäisyydet ovat lyhyitä eli noin 10 senttimetrin luokkaa. Taajuusaluetta käytettäessä tiedonsiirtonopeudet ovat hitaimmat verrattaessa korkeampiin taajuusalueisiin, ja tunnisteet sisältävät tyypillisesti hyvin vähän muistia. Käyttämänsä antennin rakenteen takia LF-tunnisteet ovat kalliita valmistaa, ja ne ovat paksumpia kuin korkeampia taajuusalueita käyttävät tunnisteet. LF-taajuutta käytettäessä nesteet ja metalli eivät vaikuta tunnisteiden lukutehokkuuteen, vaan tunnisteet voidaan lukea ongelmitta, matalampaa taajuutta käyttävän signaalin läpäisten vaikeasti läpäistävät materiaalit helpommin. LF-tunnisteita käytetään muun muassa kulunvalvontaan, eläinten tunnistukseen sekä autoissa ajonestolaitteina. (RFID4U 2018a.)

RFID-järjestelmissä HF (high frequency) -taajuusalueella käytetään yhtä taajuutta, joka on 13,56 megahertsiä (MHz). Taajuusaluetta käyttävät tunnisteet ovat myös passiivisia ja lukuetäisyydet voivat yltää yhteen metriin. Tiedonsiirtonopeudet ovat tätä taajuusaluetta käytettäessä

hitaammat kuin korkeampia taajuusalueita käytettäessä, mutta kuitenkin nopeammat kuin matalammalla LF-taajuusalueella. Jotkin HF-tunnisteet voivat sisältää neljä kilotavua muistia. HF-tunnisteet ovat halvempia valmistaa kuin LF-tunnisteet antennin rakenteen takia. HF-tunnisteita voidaan lukea helposti esineistä, jotka sisältävät vettä, kudosta, metallia, puuta tai nesteitä. Lähistöllä olevat muut metalliset esineet voivat kuitenkin vaikuttaa lukutapahtumaan. HF-tunnisteita käytetään muun muassa luottokorteissa, kirjastoissa sekä älykorteissa. (RFID4U 2018a.)

UHF (ultra high frequency) -taajuusalueet joita käytetään RFID-järjestelmissä ovat tyypillisesti 433 MHz sekä 860 - 960 MHz. 433 MHz:n taajuutta käytetään aktiivisten tunnisteiden kanssa. 860 - 960 MHz:n taajuusaluetta käytetään pääasiallisesti passiivisille tunnisteille sekä joillekin puolipassiivisille tunnisteille. Käytetyt taajuusalueet vaihtelevat eri maiden säädöksien mukaan. Esimerkiksi Suomessa käytetään taajuusaluetta 865,6 - 867,8 MHz ja Yhdysvalloissa on käytössä taajuusalue 902 - 928 MHz (GS1 2016). Myös maksimi sallittu lukuteho sekä kanavien määrä ovat maakohtaisesti säädettyjä. Lukuetäisyydet ovat maksimissaan 12 metriä passiivisia tunnisteita käytettäessä. UHF-taajuusaluetta käyttävät lukijat ovat yleensä kalliimpia kuin HF-taajuusalueen lukijat, mutta UHF-tunnisteet tulevat koko ajan edullisemmiksi. Tunnisteen antennit ovat ohuita ja helppoja valmistaa, mikä puolestaan mahdollistaa tunnisteen erittäin ohuen rakenteen. Mikäli UHF-tunniste on kiinnitetty metalliseen esineeseen, se voi vaikeuttaa tunnisteen lukemista. Tunnistetta ei voida lukea, mikäli lukijan ja tunnisteen välissä on vettä tai muutakaan sähköä johtavaa materiaalia. Nämä elementit heijastavat radiosignaalia liikaa ja näin ollen estävät tunnisteen toiminnan. UHF-tunnisteita käytetään muun muassa kuormalavojen seurantaan. (RFID4U 2018a.)

Mikroaaltotaajuudet, joita käytetään RFID-järjestelmissä, ovat 2,45 ja 5,8 gigahertsiä (GHz). Mikroaaltotaajuuksia käyttäviä tunnisteita on saatavilla passiivisina, puolipassiivisina sekä aktiivisina. Passiiviset mikroaaltotunnisteet ovat usein pienempiä kuin passiiviset UHF-tunnisteet. Lukuetäisyyksien osalta passiiviset mikroaaltotunnisteet ovat kuitenkin

samaa tasoa passiivisten UHF-tunnisteiden kanssa. Passiiviset mikroaaltotunnisteet ovat myös kalliimpia kuin passiiviset UHF-tunnisteet, mikä johtuu vähäisestä kysynnästä. Käytetyn lyhyemmän aallonpituuden takia mikroaaltotunnisteet ovat helpompia suunnitella toimimaan metallisten esineiden läheisyydessä. Aktiivisia mikroaaltotunnisteita käytetään muun muassa tietullimaksujen keräämiseen. (RFID4U 2018a.)

3.7 Esimerkkejä käyttökohteista

RFID-teknologiaa hyödynnetään kaupoissa nopeiden maksutapahtumien yhteydessä. Maksukorttien lähimaksuominaisuus on toteutettu NFC (Near Field Communication) -teknologian avulla, joka puolestaan pohjautuu RFID-teknologiaan. Maksaminen on mahdollista myös matkapuhelimella tai muulla mobiililaitteella, josta löytyy NFC-ominaisuus. (RFID Lab Finland Ry 2018.)

Logistiikassa käytetään myös RFID-teknologiaa, jonka avulla voidaan tehostaa jakeluketjun hallinnointia, kuljetusten seurantaa ja omaisuuden valvontaa. RFID mahdollistaa myös valmistusprosessin automaattisen ja kustannustehokkaan seurannan. Tavaran vastaanottoa voidaan nopeuttaa RFID-teknologian avulla, keräilytarkkuuden kasvaessa ja virheiden vähentyessä samalla. (RFID Lab Finland Ry 2018.)

Henkilötunnistuksessa ja kulunvalvonnassa käytetään RFID-teknologiaa hyvin laajasti. Esimerkiksi työpaikoilla voidaan kirjata työntekijöiden työajat käyttämällä henkilökohtaisia RFID-tunnisteita. Samaa tunnistetta voidaan käyttää hälytysjärjestelmissä, henkilöiden kulkuluvan tunnistukseen ja ovien avaamiseen. Joissakin parkkihalleissa on käytössä RFID-järjestelmiä, joiden avulla voidaan tunnistaa kausipaikkalaiset automaattisesti. Seuraavassa kuvassa 3 on esimerkki RFID-teknologian käyttökohteesta kirjaston lainausautomaattina. (RFID Lab Finland Ry 2018.)

Kuva 3. RFID-teknologia kirjastossa (Ikonen 2007)

3.8 Hyötyjä ja haittoja

RFID-teknologian hyötyjä ovat, että tunnisteet kestävät kulutusta ja voivat toimia moitteetta erittäin vaativissa sekä vaihtelevissa ympäristöissä. RFID-tunnisteet eivät tarvitse näköyhteyttä tai fyysistä kontaktia RFID-lukijaan. Tunnisteita on myös mahdollista lukea monta yhdellä kertaa, jolloin tunnisteiden lukemiseen on käytettävä vähemmän työvoimaa. RFID-tunnisteita käytettäessä voidaan yksilöidä ja seurata yksittäisiä tuotteita. Uudelleenkirjoitettavan muistin sisältäviä tunnisteita voidaan käyttää toistuvasti. (Techspirited 2018.)

Haittapuolia ovat tunnisteiden verrattain kallis hinta. Tunnisteiden koko ja paino voivat aiheuttaa ongelmia joissakin tapauksissa. Vaikka RFID-tunnisteet toimivat vaativissa olosuhteissa, metallit ja nesteet voivat vaikeuttaa tunnisteiden lukemista tietyn tyyppisiä tunnisteita käytettäessä. (Techspirited 2018.)

4 .NET COMPACT FRAMEWORK

4.1 Yleisesti

.NET Compact Framework (.NET CF) on Microsoftin kehittämä alusta. Alusta mahdollistaa laitteistosta riippumattomien ohjelmistojen kehittämisen, Windows Embedded CE ja Windows Mobile mobiilikäyttöjärjestelmien laitteille. .NET CF pohjautuu aikaisemmin kehitettyyn .NET Framework -alustaan, joka puolestaan on suunnattu Windows-käyttöjärjestelmiä käyttäville pöytätietokoneille sekä kannettaville. .NET CF pitää sisällään mobiilikäyttöjärjestelmille optimoidun virtuaalikoneen, Common Language Runtime (CLR), jossa .NET -alustalle toteutettujen ohjelmistojen hallitut ohjelmakoodit suoritetaan. .NET CF sisältää osittaisen ja pienemmän version .NET Framework -alustan luokkakirjastoista. .NET CF -alustaan sisältyy myös luokkia, jotka on suunniteltu juuri kyseistä alustaa varten. .NET CF tukee ohjelmistokehityksessä ainoastaan C# -ja Visual Basic -ohjelmointikieliä. (MSDN 2018a.)

.NET Framework -alustan luokkakirjasto sisältää kattavasti luokkia, rajapintoja sekä arvotyyppejä, joiden avulla voidaan toteuttaa ohjelmistoja. Näistä keskeisimpinä mainittakoon System -nimiavaruus, jonka alla olevat luokat ja rajapinnat toteuttavat muun muassa muuttujatyypit, poikkeusten käsittelyn, kokoelmat, resurssienhallinnan sekä tiedostonkäsittelyn. .NET Framework -alustan luokkakirjastosta löytyvän Windows Forms -nimiavaruuden sisältämät luokat, mahdollistavat graafisten käyttöliittymien rakentamisen helposti, käyttämällä kirjaston tarjoamia valmiita käyttöliittymäkomponentteja. (MSDN 2009.)

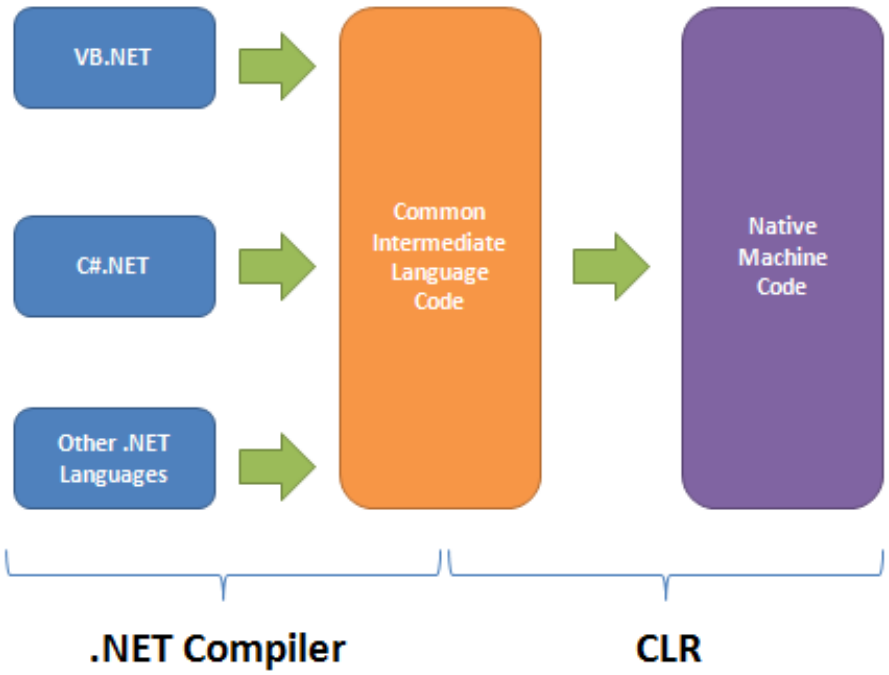
4.2 .NET Framework -ja .NET CF -alustojen vertailua

.NET Framework -ja .NET CF -alustat ovat samankaltaisia, mutta niistä löytyy myös eroja. Merkittävin ero lienee alustojen koko, sillä .NET Compact Framework on nimensä mukaisesti tiivistetty versio täysiversioisesta .NET Framework -alustasta. .NET CF toteuttaa täyden

version luokkakirjastoista noin 30 %, ollen kooltaan kuitenkin kokonaisuudessaan vain 8 % täydestä .NET Framework -alustasta. CLR on myös saatu kutistettua noin 12 prosenttiin täydestä versiosta. Koska .NET CF on tiivistetty versio, on siitä pitänyt jättää pois joitakin ominaisuuksia. Näitä pois jätettyjä ominaisuuksia ovat muun muassa ASP.NET tuki sekä tuki C++ -ohjelmointikielelle. (MSDN 2018c.)

4.3 Common Language Runtime (CLR)

Yksi olennaisimmista osista .NET Framework -alustaa on Common Language Runtime (CLR), joka tarjoaa suoritusympäristön .NET alustoille toteutetuille ohjelmistoille, virtuaalikoneen muodossa. CLR toimi siten, että .NET Framework -alustalle, korkean tason ohjelmointikielellä kirjoitettu ohjelmakoodi käännetään ensin niin sanotuksi tavukoodiksi, matalan tason ohjelmointikoodiksi. Tästä ohjelmointikoodista käytetään .NET Framework -alustan yhteydessä termiä Common Intermediate Language (CIL). Tämä käännetty tavukoodi on kohdelaitteistosta riippumatonta. CLR-virtuaalikoneen tehtäväksi jää suorittaa CIL tavukoodi, kääntämällä se ohjelman suorituksen aikana (Just-In-Time) kohdelaitteistolle sopivaksi, natiiviksi konekieliseksi koodiksi. CLR-suoritusympäristön toimintaperiaate näkyy kuviossa 3. (Microsoft 2016.)



Kuvio 3. Common Language Runtime -suoritusympäristö (Bercero 2009)

Etuja tämän CLR-virtuaalikoneen käyttämiselle ovat esimerkiksi automaattinen muistinhallinta, automaattisen roskienkeräyksen (garbage collection) muodossa. Muita etuja ovat poikkeusten hallinta, usean ohjelmointikielen tukeminen samassa projektissa sekä tyyppiturvallisuus (type safety). (Microsoft 2017.)

5 MYSQL

5.1 MySQL-tietokanta

MySQL on Oracle Corporationin suosittu, avoimen lähdekoodin SQL-tietokannan hallintajärjestelmä. Tietokanta on strukturoitu kokoelma dataa. Tietokanta voi sisältää ihan mitä vain yksinkertaisesta ostoslistasta, kuvagalleriaan. Jotta tietokantaan voidaan lisätä dataa, päästä tietokannassa olevaan dataan käsiksi ja käsitellä tuota dataa, tarvitaan siihen tietokannan hallintajärjestelmä, kuten MySQL-palvelin. (Oracle 2018a.)

MySQL-tietokannat ovat niin sanottuja relaatiotietokantoja. Tämä tarkoittaa, että tietokantaan tallennettu data on jaettu erillisiin tauluihin, eikä yhteen isoon tauluun. Nämä taulut sisältävät tietueita, jotka ovat kokoelmia dataa. Tietue puolestaan koostuu tietokannan pienimmistä yksiköistä, kentistä, jotka sisältävät itse datan. Esimerkkinä relaatiotietokannasta, yksi kentän arvo voi toimia viittauksena toisessa taulussa olevalle tietueelle. Tätä viittausarvoa sanotaan vierasavaimeksi. Relaatiotietokannan rakenteet ovat organisoitu fyysisiksi tiedostoiksi, jotka mahdollistavat nopeat kirjoitus- ja lukuoperaatiot. (Oracle 2018a.)

Termi SQL tulee sanoista Structured Query Language. SQL on yleisin standardoitu kyselykieli, jota käytetään tietokantojen käsittelyyn. SQL-kyselykielen avulla tietokantaan on mahdollista tehdä hakuja, muutoksia sekä lisäyksiä. (Oracle 2018a.)

5.2 MySQL Connector/Net

MySQL Connector/Net on ADO.NET-sovitin MySQL-tietokannalle. MySQL Connector/Net mahdollistaa MySQL-tietokannan käsittelyn .NET -alustalle kehitettäville ohjelmistoille, käyttämällä .NET -alustalle yhteensopivaa ohjelmointikieltä kuten C# :a tai Visual Basicia. MySQL Connector/Net on kirjoitettu puhtaasti C# -ohjelmointikielellä. (Oracle 2018b.)

ADO.NET (ActiveX Data Object.NET) on osa .NET Framework -alustaa. ADO.NET on kokoelma luokkia, jotka tarjoavat .NET -alustan ohjelmoijalle palvelut tiedonsiirtoon. ADO.NET tarjoaa yhteyden tietolähteisiin kuten esimerkiksi Microsoft SQL Serveriin sekä myös tietolähteisiin, jotka ovat saatavilla OLE DB :n ja XML :n kautta. ADO.NET kirjaston avulla ohjelmistot voidaan yhdistää tietolähteisiin sekä käyttää sitä tietolähteissä olevan datan noutamiseen, muokkaamiseen ja päivittämiseen. (MSDN 2018b.)

6 KÄYTETYT RFID-LUKIJAT JA -TUNNISTEET

6.1 RFID-lukijat

Tässä projektissa käytetyt kaksi käsikäyttöistä RFID-lukijaa ovat molemmat Nordic ID nimisen yrityksen valmistamia. Lukijat ovat malleiltaan PL3000 UHF RFID 200 mW (kuvassa 4) sekä Morphic (kuvassa 5). Näistä lukijoista PL3000 tuli valituksi opinnäytetyön toimeksiantajana toimivan yrityksen käyttöön, mekatroniikkaopiskelijoiden aikaisemman tehdyn selvityksen mukaan. Morphic-mallin lukijan hankki opinnäytetyön toimeksiantajana toimivan yrityksen edustaja. Tarkoituksena oli saada toinen samanlainen lukija kuin käytössä ollut PL3000 -mallin lukija. PL3000 -mallia ei enää valmistettu ja myyty kun toista lukijaa oltiin hankimassa. Tämän takia päädyttiin hankkimaan laitevalmistajan ehdottama vastaavan mallinen lukija, joka oli Morphic.

Kuva 4. PL3000 -RFID-lukija (iXtenso 2007)

Kuva 5. Morphic -RFID-lukija (Nordic ID 2018b)

Molemmat lukijoista sisältävät UHF RFID -lukumoduulin. Nämä moduulit täyttävät ISO 18000-6C -standardin, joka käsittää EPC Class 1 Gen 2 -protokollan. Tätä protokollaa käyttävät myös projektissa käyttöön valitut tunnisteet. UHF RFID -lukumoduulien lukutehoissa on eroa: PL3000 -lukijassa maksimi lukuteho on 200 milliwattia (mW) ja Morphic -lukijassa se on 100 mW. Nimelliset maksimi lukuetäisyydet ovat vastaavasti: 1,6 metriä PL3000 -lukijalla ja 0,5 metriä Morphic -lukijalla. Molemmista lukijoista löytyy myös viivakoodinlukija-moduuli. Se on laserskanneri, jonka avulla pystyy lukemaan tavallisia yksiulotteisia 1D-viivakoodeja. Viivakoodinlukijat ovat GS1 DataBar -standardin kanssa yhteensopivia. (Nordic ID 2018a; Nordic ID 2018c.)

Tiedonsiirtoyhteyksien käyttämiseen molemmista lukijoista löytyy langattoman WLAN -verkon käytön mahdollistava WiFi-moduuli, joka täyttää IEEE 802.11 b, g, i ja e -spesifikaatiot. Käyttöjärjestelmänä molemmissa laitteissa on Microsoftin Windows Embedded CE 6.0. Laitteisiin on myös asennettu .NET Compact Framework -alustan versiot 2.0 ja 3.5. (Nordic ID 2018a; Nordic ID 2018c.)

Molemmissa laitteissa on kosketusnäyttö, joka on resoluutioltaan 320 x 240 pikseliä (QVGA). Näyttöjen fyysiset koot ovat kuitenkin erilaiset. PL3000 -lukijan näyttö on 3,5 tuumaa, ja Morphic -lukijan näyttö on 2,2

tuumaa. Laitteissa on myös näppäimistöt tiedon syöttämiseen sekä erilaisille funktioille. Molemmat laitteet omaavat IP54-luokituksen, ja molempien laitteiden pitäisi kestää pudotus 1,5 metrin korkeudelta rikkoutumatta. (Nordic ID 2018a; Nordic ID 2018c.)

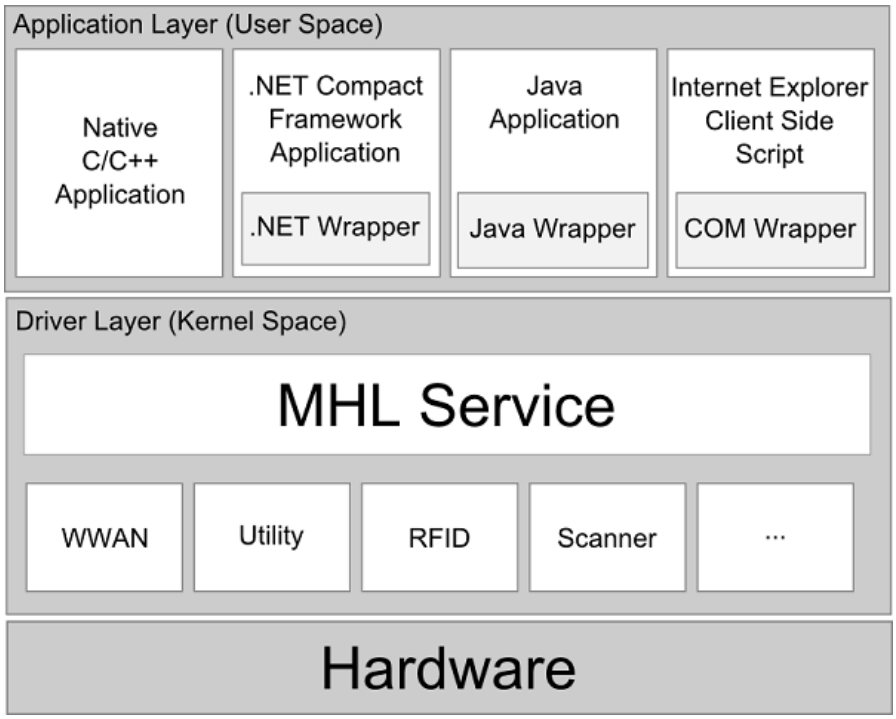
Ilmeisimmät erot löytyvät laitteiden fyysisistä ko'oista. PL3000 -lukija (250 x 40 x 90 mm) on huomattavasti isompi kuin Morphic -lukija (147 x 54 x 35 mm). Myös laitteiden välinen painoero on huomattava, sillä PL3000 painaa 550 grammaa. Morphic painaa 170 - 235 grammaa, riippuen käytetyn akun kapasiteetista. Käytetyssä PL3000 -lukijassa on lisävarusteena pistoolikahva liipaisimella, jossa on lisäakku. (Nordic ID 2018a; Nordic ID 2018c.)

Laitteet eroavat komponenteiltaan muun muassa käytettyjen prosessorin, ram- ja flash -muistien osilta. PL3000 -lukijassa on käytössä ARM9 -prosessori (Sharp LH7A400). Morphic -lukija sisältää 532 MHz kellotaajuudella toimivan ARM11 -prosessorin. Keskusmuistina PL3000 -lukijassa on 128 megatavua (MB) SDRAM -muistia, ja Morphic -lukijassa on 256 MB DDR -muistia. Tallennustilana on molemmissa laitteissa Flash -muistia: PL3000 -lukijassa 32 megatavua ja Morphic -lukijassa 288 megatavua. (Nordic ID 2018a; Nordic ID 2018c.)

6.2 Laitteiden ohjelmointirajapinta

MHL (Multiple Hardware Layer) on käsikäyttöisten RFID-lukijoiden valmistajan Nordic ID:n kehittämä ohjelmointirajapinta (API – Application Programming Interface), joka tarjoaa laitteiston hallintapalvelut laitteistolle kehitettävien ohjelmistojen käyttöön. MHL on rooliltaan hyvin samankaltainen kuin tyypillinen ajonaikaisena kirjastona (DLL – Dynamic Link Library) toteutettu ohjelmointirajapinta, joka linkitetään sovellukseen ohjelmakoodin käännösvaiheessa. MHL eroaa tyypillisestä ohjelmointirajapinnasta, sillä se on toteutettu palveluna. Käytettäessä MHL rajapintaa, laitteen funktiokutsut tehdään välikerroksena toimivaan palveluun, joka puolestaan kommunikoi laiteajureiden kanssa. Kuviossa 4 näkyy MHL rajapinnan rakenne. DLL-kirjastoa käytettäessä, haittapuolena

on mahdollinen tilanne, jossa DLL-kirjaston päivittäminen tai muuttaminen rikkoo sitä käyttävän ohjelmiston. (Nordic ID 2010.)



Kuvio 4. MHL-rajapinnan rakenne (Nordic ID 2010)

MHL-ohjelmointirajapintaa käytettäessä, kutsutun funktion puuttuminen ei aiheuta sovelluksen kaatavaa poikkeusta, vaan siitä annetaan virheilmoitus. MHL-luokka piilottaa kaikki natiivit kutsut palvelulle ja laiteajureille, jolloin MHL on enemmän perinteisen ohjelmointirajapinnan kaltainen. Käytettäviä funktioita ei siis kutsuta suoraan, vaan se tapahtuu kutsumalla yleisiä MHL-funktioita näiden todellisten funktioiden nimillä. Tällaisia yleisiä MHL-funktioita ovat muun muassa GetInt, GetBool ja GetString, joiden parametrina toimii todellisen funktion nimi. Tämän tyylinen toteutus takaa, että sovellus ei kaadu koskaan, vaikka MHL-laiteajureita tai laitteistoa päivitettäisiin tai muutettaisiin. Kertaalleen kehitetty MHL rajapintaa käyttävä ohjelmisto toimii myös Nordic ID:n uudemmissa laitteissa olettaen tietenkin, että käytettyjen yleisten MHL-funktioiden paluuarvot tarkistetaan. MHL on yhtä altis virheille kuin mikä

tahansa muu ohjelmointirajapinta, jos vain oletetaan jonkin laitteiston osan olevan olemassa. Tämän takia tulee suorittaa tarkistus, joka tapahtuu tutkimalla laitteistoa kutsuvan funktion paluuarvoa. (Nordic ID 2010.)

Tuettuja ohjelmointikieliä ovat .NET CF -alustalle C# sekä Visual Basic. Muita tuettuja ohjelmointikieliä ovat Java sekä natiiveille ohjelmistoille C- ja C++ -kielet. Myös Internet Explorerin asiakasohjelman puoleinen skriptaus on tuettu. Tällöin ei kuitenkaan voida käyttää GetBin- ja SetBin-funktioita, sillä JScript ei tue osoittimien käyttöä. .NET CF -alustaa käytettäessä, tuettuina kehitysympäristöinä ovat Visual Studio 2005 sekä Visual Studio 2008. (Nordic ID 2010.)

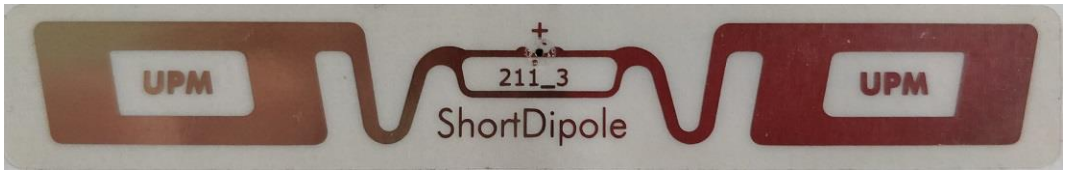
MHL-ohjelmointirajapinnan toiminnallisuus koostuu neljästä perusosasta: Get- ja Set -funktioista, profiilienhallinnasta sekä luettelointifunktioista. Get- ja Set -funktioiden avulla suoritetaan kyselyt ja komennot MHL :n todellisiin funktioihin. Get- ja Set -funktiopareja on olemassa yksi jokaista tuettua tietotyyppiä kohden. Tuettuja tietotyyppejä ovat boolean (totuusarvot), integer (kokonaisluku), unsigned integer (DWORD, etumerkitön kokonaisluku), string (merkkijono), double (64-bittinen liukuluku), ja tavulohkot (GetBin- ja SetBin -funktioita käytettäessä). (Nordic ID 2010.)

Profiilienhallintaa käytetään MHL :n ominaisuus ajureiden funktioissa, joiden asetukset voidaan tallentaa profiiliin. Kaikkia ominaisuuksia ei ole mahdollista tallettaa profiiliin. Profiilienhallinta kehitettiin siitä syystä, että profiilien avulla sovellukset voisivat ottaa täyden kontrollin laitteen näppäimistöstä ja viivakoodiskannerista sovelluksen ollessa käynnissä. Laitteen oletusasetukset olisi helppo palauttaa sovelluksen sulkeutuessa. Sovelluksen käynnistyessä, olemassa olevat asetukset tallennetaan väliaikaiseen profiiliin, jonka jälkeen asetuksiin voidaan tehdä tarvittavat muutokset. Sovelluksen sulkeutuessa, ladataan tämä aikaisemmin luotu profiili, joka palauttaa laitteen asetukset sellaisiksi kuten ne olivat ennen sovelluksen käynnistämistä. Luettelointifunktioita käytetään harvemmin sovelluksia kehitettäessä. Luettelointifunktiot luetteloivat dynaamisesti

MHL :n käytettävissä olevat ajurit ja funktiot sovelluksen suorituksen aikana. (Nordic ID 2010.)

6.3 RFID-tunniste

Projektissa käytetty RFID-tunniste on malliltaan UPM Raflatac yrityksen valmistama ShortDipole^{x}, joka on nähtävissä kuvassa 6. Se on liimattavassa tarramuodossa oleva tunniste, joka on kooltaan 97 x 15 millimetriä. (UPM Raflatac 2009.)



Kuva 6. UPM Raflatac ShortDipole^{x} RFID-tunniste

Tunniste käyttää kommunikointiin EPC Class 1 Generation 2 -protokollaa. Tunnisteen käyttötaajuus kattaa RFID-teknologian käyttämän UHF-taajuusalueen globaalisti, eli 860 - 960 MHz. Tunnisteessa on 240-bittinen EPC-muisti sekä 64-bittinen TID-muisti (transponder ID), joka sisältää tunnisteen uniikin 32-bittisen sarjanumeron. (UPM Raflatac 2009.)

7 SOVELLUKSEN TOTEUTUS RFID-LUKIJALLE

7.1 Kehitystyön prosessi

Projekti alkoi toteuttamalla sovelluksen prototyyppi sovellusohjelmoinnin työkurssille, keväällä 2010. Prototyypin tarkoituksena oli osoittaa, että ohjelmointitekniikan opiskelijan olisi mahdollista suunnitella ja kehittää kohdeyrityksen tarpeita vastaava sovellus käsikäyttöiselle RFID-lukijalle. Projektin aluksi saatiin ensimmäiset spesifikaatiot sekä kohdeyrityksen asiakasvaatimukset, mekatroniikkaopiskelijoiden aikaisemmin tehdyn selvityksen mukaan. Tämän selvityksen perusteella mekatroniikkaopiskelijat olivat jo valinneet valmiiksi sopivan käsikäyttöisen RFID-lukijan sekä RFID-tunnisteet, joita projektissa käytettiin.

Vaatimuksena projektissa oli sovelluksen toteuttaminen käsikäyttöiselle RFID-lukijalle. Sovelluksen avulla RFID-tunniste voitaisiin lukea, kerätä tietoa valmistettavasta tuotteesta sekä lähettää kerätyt tiedot jonkinlaiseen tietokantaan. Tuotetietojen keräämisen toivottiin tapahtuvan viivakoodeja lukemalla. Kerättyjä tietoja olisivat ainakin tuotteen tyyppi, tuotteessa käytetyt komponentit, tuotteen kokoonpanijan henkilöllisyys sekä valmistusaika. Sovelluksen tulisi myös pystyä hakemaan tietokannasta tuotteesta tallennetut tiedot, kun tuotteeseen kiinnitetty RFID-tunniste luetaan. Tätä ominaisuutta tulisi hyödynnettyä tuotteiden koestamisen yhteydessä, jossa tietokantaan voitaisiin liittää tietoa koestuksen tuloksista.

Sovelluksen asetuksiin haluttiin myös säädin RFID-lukutehon muuttamista varten. Lukutehoa säätämällä voidaan muuttaa lukijan ja tunnisteen välistä lukuetäisyyttä. Tätä käytettäisiin tilanteissa, mikäli lukija lukisi vahingossa yhden lukutapahtuman aikana monta tunnistetta kerralla. Sovelluksen käyttöliittymän tulisi myös olla mahdollisimman yksinkertainen ja loppukäyttäjän käytettävissä olevien kontrollien määrä tulisi olla vähäinen.

Toimintaperiaate sovelluksessa tulisi olla seuraavanlainen: ensin luettaisiin RFID-tunniste, jonka sarjanumero ilmestyisi näkyviin sovelluksessa

olevaan tekstikenttään. Tämän jälkeen luettaisiin yksi kerrallaan kerättäviä tietoja omiin tekstikenttiinsä, valmiiksi tulostettuja viivakoodeja lukemalla. Kun tiedot olisi kerätty, sovelluksen tulisi lähettää kaikki kerätyt tiedot johonkin valittuun tietokantaan. Näiden tietojen pohjalta aloitettiin prototyypin suunnittelu ja kehitys.

Sovelluksen prototyypin kehitys aloitettiin tutustumalla ensin toimitettuun käsikäyttöiseen RFID-lukijaan ja sen telakkaan. Telakkaa käytetään lukijan akun lataamiseen sekä lukijan ja tietokoneen väliseen tiedonsiirtoon. Tämän laitteeseen tutustumisen jälkeen testiympäristössä asennettiin sovelluskehitystyökalut, sovelluksen kehittämistä varten valittuun ohjelmointiympäristöön (IDE, integrated development environment) Microsoft Visual Studio 2008. Samalla asennettiin myös tarvittavat laiteajurit.

Kun ohjelmointirajapintaan oltiin tutustuttu riittävästi, valittiin kehitystyön ohjelmointikieleksi C#. Vaikuttava tekijä tähän oli ohjelmointirajapinnan dokumentoinnista löytyneet esimerkit, jotka oli toteutettu pääasiallisesti kyseisellä ohjelmointikielellä. Valintaan vaikutti myös mahdollisuus suunnitella ja toteuttaa sovelluksen graafinen käyttöliittymä ohjelmointiympäristössä, valmiita komponentteja käyttämällä.

Kun ohjelmointiympäristö oli saatu asennettua valmiiksi, aloitettiin kehitys testaamalla laitteen ohjelmointirajapinnan funktioita. Tämä tapahtui luomalla alustalle sopiva tyhjä malliprojekti, jonka graafinen käyttöliittymä luotiin asettelemalla erilaisia valmiita graafisen käyttöliittymän komponentteja ohjelmointiympäristössä. Laitteen ohjelmointirajapinnan dokumentaation ymmärtäminen tuotti aluksi vaikeuksia, mutta testaamalla erilaisia funktioita päästiin kehitystyössä eteenpäin.

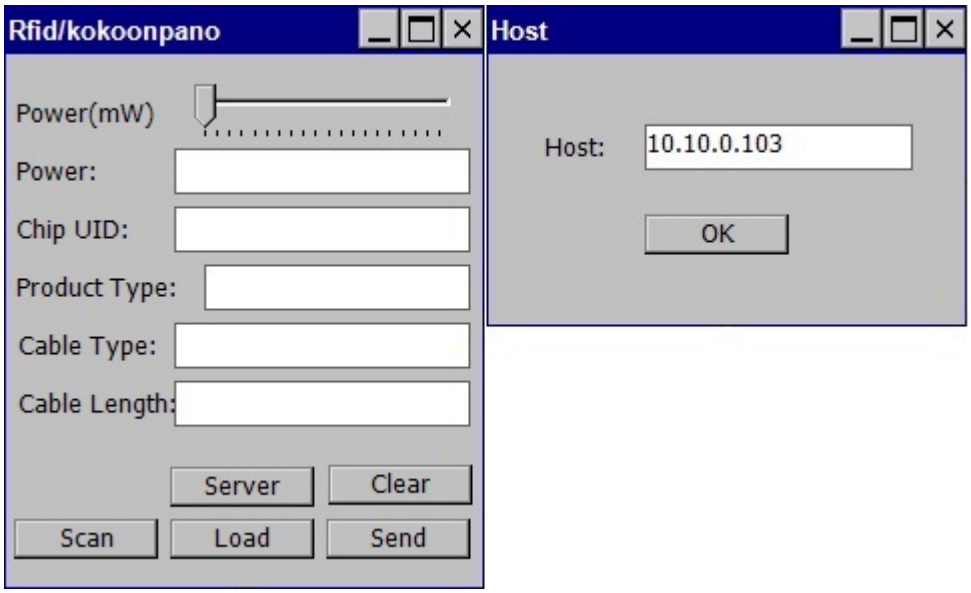
Kun laitteen ohjelmointirajapinnan toiminnallisuutta oli selvitetty, tuli seuraavaksi haasteeksi sopivan tietokannan valinta. Tietokannaksi valittiin MySQL. Valinnan syinä olivat, että kyseisen tietokannan käytöstä oli aikaisempaa kokemusta, sekä löydetty MySQL Connector/NET-sovitin. Tämä sovitin mahdollistaa MySQL-tietokannan käsittelyn, käyttämällä

.NET -alustan kanssa yhteensopivaa ohjelmointikieltä. Kyseisestä sovittimesta löytyi yhteensopiva versio laitteen .NET Compact Framework -alustalle. Tietokannan valinnan jälkeen testiympäristöön asennettiin WAMP, johon sisältyvät muun muassa MySQL-palvelin sekä tietokannan hallintaa helpottava PHPMyAdmin. Testausta varten luotiin tietokanta ja tarvittava taulu, käyttäen PHPMyAdmin -hallintatyökalua. Tämän jälkeen tutustuttiin MySQL Connector/NET-sovittimen dokumentaatioon, jonka aikana tehtiin pieni testisovellus käsikäyttöiselle RFID-lukijalle. Testisovelluksen avulla voitiin todentaa, että kyseisen sovittimen ja tietokannan käyttäminen olisi mahdollista projektissa.

Sovelluksen prototyypin rakentaminen tapahtui aikaisemman luodun projektin pohjalle, jossa testattiin laitteen ohjelmointirajapintaa. Tähän projektiin lisättiin tarvittavat tekstikentät sekä muut kontrollit, kuten painikkeet. Prototyypin ohjelmoinnissa hankaluuksia tuotti käsikäyttöisestä RFID-lukijasta löytyvän liipaisimen liittäminen tarkisteltavaksi kontrolliksi koodissa. Tämä kuitenkin onnistui tutkimalla ensin, mikä liipaisimen koodi on painettaessa. Sitten tämä koodi tallennettiin profiiliin, jonka jälkeen koodi voitiin vaihtaa jonkin muun tarkisteltavan näppäimen koodiksi. Tässä tapauksessa koodiksi vaihdettiin ’z’ -näppäimen koodi, jota tarkistelemalla voidaan selvittää, painettiinko liipaisinta. Prototyypin kehittämisessä meni arviolta yhteensä noin 25 tuntia, jonka jälkeen prototyyppi esiteltiin toimeksiantajayritykselle.

Prototyypin esittelytilaisuudessa ilmeni ongelma, jossa RFID-tunnisteesta luettu tieto ei ollutkaan uniikki arvo, vaan EPC-muistista (Electronic Product Code) noudettu alustamaton tuotteen sarjanumero. Tämä puolestaan aiheutti tilanteen, jossa tietokantaan tietoja kirjoittaessa ei tallentunut kuin yksi tietue. Toimeksiantajayrityksen edustajat olivat kuitenkin tyytyväisiä lyhyehkössä ajassa toteutettuun sovelluksen prototyyppiin. Yrityksen edustajat halusivat, että sovellus toteutettaisiin kokonaisuudessaan yrityksen tarpeita vastaavalle tasolle. Toimeksiantajayritykselle esitellyn prototyypin graafinen käyttöliittymä näkyy kuvassa 7. Esittelytilaisuudessa ilmenneen ongelman korjaus aloitettiin, ottamalla yhteyttä käsikäyttöisen RFID-lukijan valmistajaan.

Yrityksen tuotetuesta saatiin tarvittavat ohjeistukset sekä esimerkkikoodit RFID-tunnisteen TID-muistialueelta (TID – transponder ID) löytyvän, uniikin sarjanumeron lukemiseen.



Kuva 7. Prototyypin graafinen käyttöliittymä

Sovelluksen varsinainen kehitystyö aloitettiin kesällä 2010. Ennen ohjelmoinnin aloittamista, käytiin kohdeyrityksessä asentamassa MySQL-tietokanta yrityksen Windows Server 2003 -käyttöjärjestelmää käyttävälle palvelimelle. Asennetun MySQL-tietokannan versio oli tuolloin uusin tarjolla oleva versio: 5.1.46. Sovelluksen kehitystyö tapahtui kohdeyrityksen tiloissa, ja heidän tarjoamallaan kannettavalla tietokoneella. Ohjelmointityö aloitettiin asentamalla kannettavalle tietokoneelle ohjelmointiympäristö Visual Studio 2008 sekä tietenkin käsikäyttöisen RFID-lukijan sovelluskehitystyökalut ja ajurit. Tietokannan tutkimisen helpottamiseksi käytettiin avoimen lähdekoodin sovellusta nimeltä HeidiSQL, jonka avulla tietokantaa pystyy helposti tarkistelemaan sekä tarvittaessa muokkaamaan.

Kehitystyön aluksi prototyypissä käytetty .NET Compact Framework -alustan versio 2.0 vaihdettiin uudempaan versioon 3.5, joka mahdollisti joustavammat työskentelymahdollisuudet graafisten käyttöliittymäkomponenttien kanssa. Tämä helpotti esimerkiksi tekstien keskittämistä tekstikentissä, mikä oli vaikeasti toteutettavissa vanhemmassa .NET CF -alustan versiossa 2.0. Sovelluksen graafista käyttöliittymää muokattiin jatkuvasti loppukäyttäjille sopivammaksi kehitystyön aikana.

Kesällä sovellukseen lisättiin tekstimuodossa oleva asetustiedosto, johon tallennetaan arvot käyttäjästä, MySQL-tietokantapalvelimen IP-osoitteesta sekä RFID-lukijan lukutehosta. Asetustiedostoon tallennetaan edellä mainittujen arvojen muutokset, jotta arvoja ei tarvitse joka kerta syöttää uudestaan aina ohjelman käynnistyessä.

Erinäisten standardityyppisten tuotteiden tietojen automaattinen täyttö tekstikenttiin tapahtuu lukemalla ainoastaan tuotteen tyyppi viivakoodilla. Näissä standardityyppisissä tuotteissa on aina samat tiedot. Kenttien välillä siirtyminen, eli kentän fokuksen siirtäminen, toteutettiin myös automaattiseksi. Tämä tehtiin käyttöliittymän käyttämisen helpottamiseksi, jotta käyttäjän ei tarvitse erikseen valita seuraavaa kenttää. Sovellukseen toteutettiin myös Odota -ikkuna tietojen siirtämisen ajaksi. Ikkunan tehtävä on indikoida, että tietoa siirtyy laitteen ja tietokannan välillä. Samalla ikkuna estää käyttäjän mahdolliset ylimääräiset virhesyötteet.

Syksyllä 2010 sovellus jaoteltiin luokkiin. Aikaisemmin lähes kaikki ohjelmointikoodi sisältyi Form1 -luokkaan. Uudet luodut luokat olivat nimeltään: Asetukset, MHLKasittelyt, Mysli, TiedostonKasittely. Näistä ja muista sovelluksen luokista kerrotaan tarkemmin opinnäytetyön kohdassa 7.2. Luokkarakenteen luomisen lisäksi koodin määrää vähennettiin yhdistelemällä samankaltaisia funktioita yleiskäyttöisemmiksi funktioiksi.

Sovelluksen toimintaa muutettiin asettamalla viivakoodin maksimi lukuaika viiteen sekuntiin. Samalla laitteen toimintaan tehtiin muutos, jossa viivakoodin lukeminen lopetetaan myös, kun mikä tahansa viivakoodi

saadaan luettua. Syy muutoksiin oli, että viivakoodin lukemiseen käytetty oletusarvoinen kymmenen sekunnin aika oli liian pitkä. Näillä viivakoodin lukemiseen liittyvillä muutoksilla pyrittiin optimoimaan sovelluksen käyttöä nopeuttamalla toimintoja.

Viivakoodien lukemiseen lisättiin validointi, jolloin tiettyyn viivakoodikenttään voi lukea vain siihen kuuluvan tiedon. Validointi toteutettiin lisäämällä viivakoodin sisältämän tiedon alkuosaan merkintä, esimerkiksi ’@01=näkyvä tieto’. ’@’ ja ’=’ -merkit ovat vakioita. Ensimmäinen numero merkitsee näkymän numeroa, joka alkaa nollasta. Toinen numeroista merkitsee viivakoodikentän numeroa. Nämä validoinnin osat eivät näy käyttäjälle mitenkään, ja tekstikenttiin tulostuu näkyviin varsinainen tietokantaankin tallennettava tieto. Väärän tyyppistä viivakoodia luettaessa, luettavaan tekstikenttään ilmestyy teksti ’VÄÄRÄ’.

Käyttöliittymää paranneltiin siten, että fokuksen saanut tekstikenttä väritetään, jolloin käyttäjä tietää paremmin, mikä lukukentistä on aktiivinen. Tekstikenttiin implementoitiin myös tarkistusfunktio, joka tapahtuu ennen kuin tietoja ollaan tallentamassa tietokantaan. Mikäli jostakin kentästä puuttuu tarvittava tieto tai se on validoinnin kannalta väärä, kenttä saa automaattisesti fokuksen. Käyttöliittymään lisättiin myös väri-indikaattori laitteen langattoman verkkoyhteyden tilasta, joka päivittyy sovellukseen lisätyn ajastinfunktion mukaan viiden sekunnin välein.

Kokoonpano -näkymään lisättiin Säilytä tiedot kentissä -valintaruutu-kontrolli. Kontrollin tehtävänä on varmistaa, että jo kertaalleen viivakoodeja lukemalla kerätyt tiedot säilyvät omissa tekstikentissään senkin jälkeen, kun tiedot tallennetaan tietokantaan. Tällöin jokaisen tuotteen kohdalla tarvitsee lukea ainoastaan RFID-tunniste uudestaan. Tämä puolestaan vähentää tarvetta lukea muuttumattomat tiedot jokaisen saman mallisen tuotteen kohdalla uudestaan.

Näkymään lisättiin käyttäjälle näkyvä RFID-laskuri, jotta käyttäjä voi nähdä kuinka monta tunnistetta on luettu. Samalla voidaan varmistua, että RFID-tunnisteeseen liitetty tieto on siirtynyt tietokantaan. Käyttöliittymästä

poistettiin erilliset Tuo -painikkeet, ja tietojen lukeminen tietokannasta tapahtuu automaattisesti RFID-tunnisteen lukemisen jälkeen.

Käyttöliittymään lisättiin käyttäjänimen kohdalle tarkistus siten, että Asetus -näkymä on näkyvissä vain käyttäjälle nimeltä Admin. Muilta käyttäjiltä näkymä on piilotettu. Sovellukseen lisättiin myös Tila -näkymä, jossa on mahdollista vaihtaa tuotteen tila. Samalla tietokantaan lisättiin vaadittava kenttä, johon tieto voidaan kirjoittaa. Tilat kuvastavat tuotteen tilaa tuotantolinjastolla. Valittavia tiloja ovat: TUOTANTO, KOESTETTU, POISTO sekä VARASTO. Tila päivittyy automaattisesti tuotannon eri vaiheiden välillä. Esimerkiksi kun tuotteen tiedot lisätään ensimmäisen kerran tietokantaan, tulee tilaksi TUOTANTO. Tämä tila muuttuu koestuksen jälkeen automaattisesti KOESTETTU -tilaksi.

Joulukuussa 2010 sovellukseen Kokoonpano -näkymään lisättiin työnumero-tekstikenttä, johon viivakoodilla luetaan kaikki tarvittava tieto yksilöitävästä tuotteesta. Työnumero lisättiin myös tietokantaan omaksi kentäksi. Työnumero saadaan yrityksen käyttämästä ERP (Enterprise Resource Planning) – toiminnanohjausjärjestelmästä nimeltä LemonSoft. Työnumero-viivakoodiin oli sidottu kaikki muu tarvittava tieto tuotteesta. Tietojen erittely viivakoodissa oli toteutettu käyttämällä ’|’ -merkkiä. Eritelty tieto jaetaan omiin tekstikenttiinsä, viivakoodin lukemisen jälkeen. Tällä tavoin pystyttiin vähentämään käyttäjiltä vaadittujen syötteiden määrää mahdollisimman pieneksi, jolloin sovellusta on yksinkertaisempi käyttää.

Graafiseen käyttöliittymään lisättiin nollauspainike RFID-laskurille sekä erillinen Nollaa laskuri -ikkuna. Ikkunan tehtävä on varmistaa, ettei laskurin nollaaminen tapahdu vahingossa. RFID-tekstikenttään lisättiin teksti ’Luetaan…’ RFID-tunnisteen lukutapahtuman ajaksi. Tekstin tarkoitus on selventää loppukäyttäjälle mitä sovelluksessa tapahtuu. Jouluun 2010 mennessä laadittiin mahdollisimman yksiselitteiset käyttöohjeet loppukäyttäjille sovelluksen ja laitteen käyttämiseen.

Kesällä 2011 kehitystyötä jatkettiin edelleen. Tuolloin kohdeyritys oli hankkinut uuden käsikäyttöisen RFID-lukijan samalta valmistajalta kuin

aikaisemmin hankittu PL3000. Uuden lukijan malli oli Morphic. Koska uudessa laitteessa oli erilainen RFID-lukijamoduuli, ei aikaisemmassa laitteessa toiminut RFID-lukutoimintoon käytetty funktio enää toiminutkaan uudessa laitteessa. Tästä syystä kyseinen funktio jouduttiin ohjelmoimaan uudestaan molemmille laitteille yhteensopivaksi. Aikaisemmin käytetty laitteen rajapinnan funktio palautti vastauksena lukujonon, joka oli suoraan käyttökelpoisessa muodossa. Uudessa toteutuksessa käytetty rajapinnan funktio palauttaa RFID-tunnisteesta luetun tiedon binäärimuodossa, josta se muunnetaan haluttuun muotoon heksadesimaaleiksi.

Oletusarvoisesti päällä ollut laitteen värinätoiminto kytkettiin sovelluksen käynnistyessä pois päältä. Värinätoiminto aktivoitui viivakoodin lukemisen jälkeen, oli lukutapahtuma onnistunut tai epäonnistunut. Muutos tehtiin, sillä käyttäjät kokivat värinätoiminnon häiritseväksi työskennellessä. Värinätoiminto kytketään takaisin päälle, kun sovellus suljetaan.

Sovelluksen sisäiseen rakenteeseen tehtiin myös parannuksia. Esimerkiksi MySQL-tietokantaa käsitteleviin funktioihin lisättiin tarkisteltava paluuarvo, jonka avulla voitiin muun muassa varmentaa, että tiedon kirjoittaminen tietokantaan oli onnistunut. Sovelluksen Kokoonpano -näkymään lisättiin kenttä rasiatyypille, ja samalla vastaava tietokenttä lisättiin tietokantaan. Rasiatyyppi on arvoltaan joko NORMAALI tai Au (kullattu), ja se tarkoittaa millaiset kontaktipinnat mikrokytkimessä on. Käytetty rasiatyyppi puolestaan vaikuttaa koestuksessa käytettävään käyttöjännitteeseen.

Sovellukseen lisättiin uusi luokka nimeltään Kaapelit. Luokan tehtävänä on käsitellä tekstitiedostoon listatut kaapelin tyypit ja materiaalit, jotka listattiin tilan säästämiseksi Työnumero -viivakoodissa numeromuotoon. Tähän ratkaisuun päädyttiin, sillä tulostettavasta Työnumero -viivakoodista olisi tullut fyysisesti liian suuri viivakoodiin sisällytetyn suuren tietomäärän takia. Viivakoodi ei olisi enää sopinut tulostettavaksi yrityksessä käytettävään työmääräimeen. Viivakoodin pienentäminen skaalaamalla työmääräimeen tulostettavaa viivakoodikuvaa ei onnistunut, sillä viivakoodista tuli liian sumea, että sitä ei ollut enää mahdollista lukea

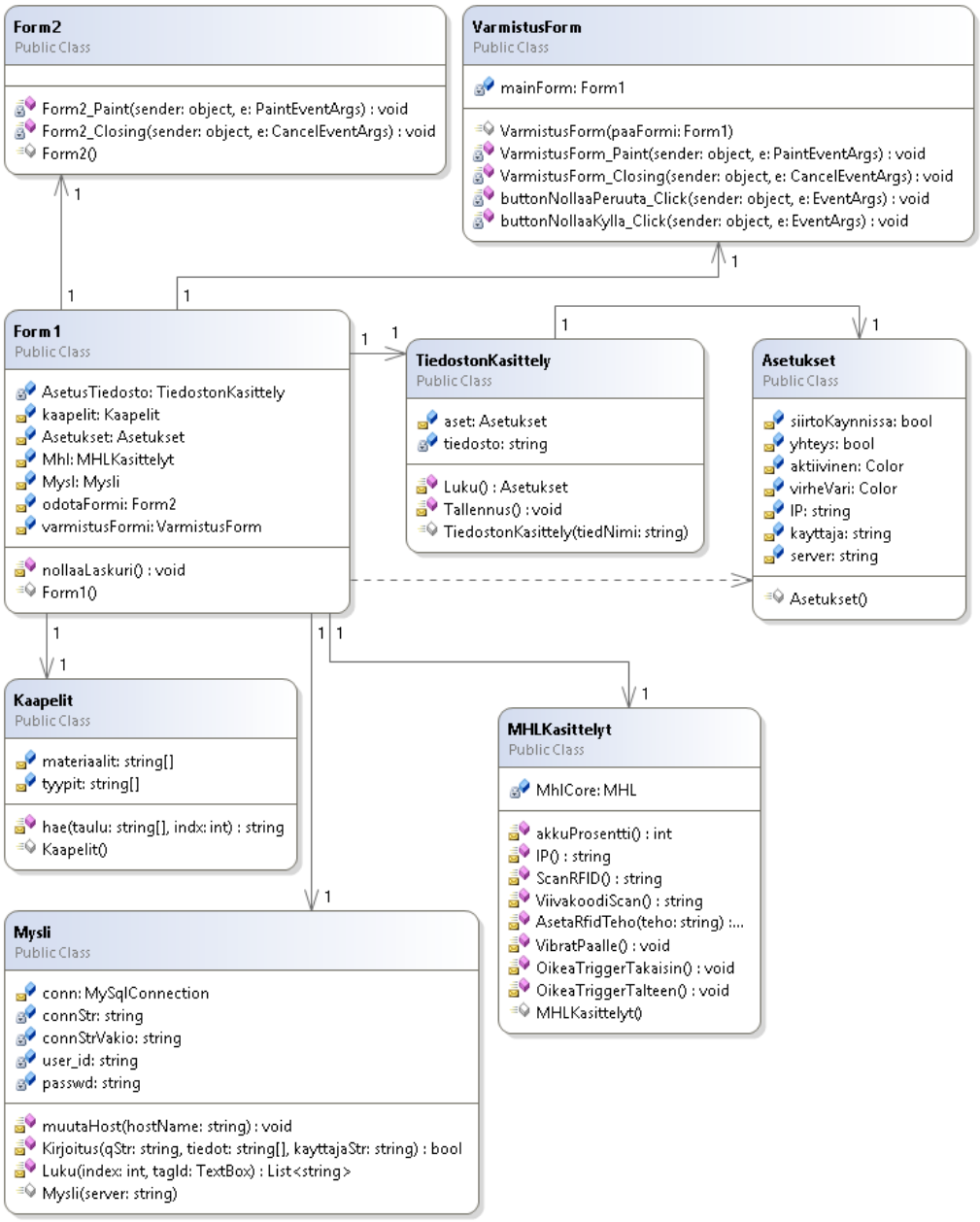
onnistuneesti. Kaapelin tietojen listaamisella saatiin Työnumero -viivakoodi pienennettyä takaisin järkevän kokoiseen muotoon. Tekstitiedostoon on listattu kaapelitiedot sovitussa järjestyksessä, jossa tiedot ovat eriteltyinä puolipisteillä.

Sovelluskoodia siistittiin yleistämällä funktioita entisestään. Samalla koodin kommentointia täydennettiin. Käyttöliittymään tehtiin parannuksia lisäämällä näkymiin laitteen akun tila prosenttiarvoina. Tämä tila päivittyy samaan tapaan kuin langattoman verkkoyhteyden tilaa tarkkaileva funktio, eli aikaisemmin mainitun ajastimen mukaan. Akun ollessa latauksessa prosenttien tilalla lukee: ’Lataa…’. Langattoman verkkoyhteyden sekä akun tilan indikaattoreille lisättiin kuvakkeet selventämään käyttöliittymän ilmettä.

7.2 Luokkarakenne

Sovellus koostuu erilaisista luokista, joilla jokaisella on oma roolinsa ja funktionsa. Näitä luokkia ovat: Program, Form1, Form2, VarmistusForm, Asetukset, MHLKasittelyt, Mysli, TiedostonKasittely ja Kaapelit. Program -luokka on sovelluksen niin sanottu pääluokka, jota käytetään vain sovelluksen käynnistyessä Form1 -ikkunan lataamiseen ja näyttämiseen. Kuviossa 5 on kuvattu sovelluksen toteuttavien luokkien rakennetta luokkakaavion muodossa. Huomioitavaa on, että luokkakaavioon ei kuitenkaan ole sisällytetty Program-luokkaa, sillä sen ainut tehtävä on luoda instanssi Form1 -luokasta.

Form1 -luokka toteuttaa sovelluksen pääikkunan, jossa on useita näkymiä. Luokkaan sisältyy myös graafisen käyttöliittymän asettelutiedosto. Tämä luokka sisältää valtaosan sovelluksen rakenteesta, joka käsittää graafisten käyttöliittymäkomponenttien käsittelyt. Tämä johtuu siitä, että sovelluksen graafisen käyttöliittymän suunnittelu sekä rakentaminen suoritettiin ohjelmointiympäristön graafisen käyttöliittymän rakentamiseen tarkoitettujen työkalujen avulla. Form2 -luokka toteuttaa Odota -ikkunan, joka on esillä käyttäjälle sen aikaa, kun tietoja luetaan tietokannasta. Muuta toiminnallisuutta luokalla ei ole.



Kuvio 5: Luokkakaavio sovelluksesta

VarmistusForm -luokka toteuttaa Nollaa laskuri -ikkunan, jota käytetään varmistamaan RFID-laskurin nollaus. Kun laskuri halutaan nollata, nollataan Form1 -luokasta löytyvä laskurin näyttämä lukuarvo ja samalla tyhjennetään säiliöön tallennetut RFID-tunnisteiden TID-muistialueelta löytyvät sarjanumerot. Näitä säiliön sisältämiä sarjanumeroita tarkistellaan, jotta varmistutaan että luettu tunniste on todellisuudessa uusi RFID-

tunniste, eikä esimerkiksi vahingossa aikaisemmin luettu tunniste. Kun uusi tunniste on luettu ja tarkistettu ettei sitä löydy säiliöstä, lisätään tunnisteen TID-muistista löytyvä sarjanumero säiliöön ja samalla laskurin lukuarvoa nostetaan yhdellä. Tämä tarkistus tehdään, kun tietoja ollaan viemässä tietokantaan.

Asetukset luokka sisältää asetuksiin liittyviä tietoja kuten käyttäjän nimen, RFID-lukutehon, MySQL-tietokantapalvelimen IP-osoitteen sekä erilaiset kentissä käytettyjen värien arvot. Asetukset -luokka sisältää myös kaksi totuusarvoa: yhteys sekä siirtoKaynnissa. Yhteys -arvoa käytetään tarkistuksissa, että onko langaton verkkoyhteys toiminnassa. Arvo päivittyy pääikkuna -luokan toteuttaman ajastimen avulla viiden sekunnin välein. Tämä tarkistelu tapahtuu MHLKasittelyt -luokasta löytyvän IP-funktion avulla. Yhteys -arvolle löytyy sovelluksen pääikkunasta oma indikaattori sekä kuvake. Mikäli yhteys on olemassa, indikaattorin väri on vihreä. Mikäli yhteyttä ei ole, indikaattorin väri on punainen. SiirtoKaynnissa -totuusarvoa käytetään apuna Yhteys -totuusarvon kanssa tarkistuksissa varmistamaan, onko MySQL-tietokantayhteyden kirjoitus- tai lukuoperaatio käynnissä.

MHLKasittelyt -luokan tehtävä on laitteen ohjelmointirajapinnan funktioiden keskittäminen yhteen paikkaan. Viivakoodin lukemiseen käytettävä aika muutetaan viiteen sekuntiin tämän luokan muodostinfunktiossa. Muodostinfunktiossa kytketään pois päältä myös laitteen värinätoiminto. Luokassa on funktio laitteen värinätoiminnon kytkemiseksi päälle, jota käytetään, kun sovellus suljetaan. Luokasta löytyvät myös funktiot RFID-tunnisteen ja viivakoodin lukemiseen. Muita funktioita ovat Scan -näppäimen tai liipaisimen koodin tallentaminen profiiliin ja profiilin palauttaminen, RFID-lukijan lukutehon asettaminen, laitteen akun sekä langattoman verkkoyhteyden tilojen tiedustelu. Kuvassa 8 on esimerkki luokan funktiosta, jota käytetään RFID-tunnisteiden lukemiseen.

```
internal string ScanRFID()
{
    string uid = "";    // asetetaan
    byte[] idDec;
    int handle = MhlCore.OpenDrv("RFID");   // avataan RFID-ajuri
    MhlCore.Execute(handle, "RFID.Inventory");  // suoritetaan RFID-inventaario
    byte[] idList = MhlCore.GetBin(handle, "RFID.IdList");  // haetaan lista
                                                            // löydetyistä tunnisteista
    uint i;
    uint chipId = 200;  // oletusarvoksi väärä/ei-validi
    for (i = 0; i < 128; i++)   // käydään lista läpi
    {
        if (idList[i] == 1)
        {
            chipId = i;
            break;  // lopetetaan listan etsiminen ja jatketaan eteenpäin
        }
    }
    if (chipId != 200)
    {
        MhlCore.SetDword(handle, "RFID.CurrentId", chipId); // valitaan löydetty
                                                            // tunniste
        MhlCore.SetDword(handle, "RFID.BlockPointer", 0);   // valitaan luettavan
                                                            // muistialueen aloituskohta
        MhlCore.SetDword(handle, "RFID.BlockCount", 8);     // valitaan luettavien
                                                            // tavujen määrä
        MhlCore.SetDword(handle, "RFID.EPCC1G2.Bank", 2);   // valitaan EPC Class 1 Gen 2
                                                            // tyyppisen tunnisteen TID-muistialue
        idDec = MhlCore.GetBin(handle, "RFID.BlockData");   // luetaan muistialueen
                                                            // tieto binäärimuodossa
        foreach (int des in idDec)
        {
            string tmp = des.ToString("X2");    // muunnetaan luettu binäärimuodossa
                                                // oleva tieto merkkijonoon heksadesimaali-muotoisena
            uid += tmp;
        }
        uid = uid.Trim();
        if (uid.Length == 0)
            uid = "Tunnistetta ei löytynyt";
    }
    else
        uid = "Tunnistetta ei löytynyt";
    MhlCore.CloseDrv(handle);   // suljetaan RFID-ajuri
    return uid;
}
```

Kuva 8: Ote sovelluskoodista

Mysli -luokka sisältää tarvittavat tiedot ja funktiot MySQL-tietokannan käsittelyyn. Muuttujia luokassa ovat muun muassa tietokantayhteyden mahdollistavat: tietokannan käyttäjänimi, salasana sekä käytettävän tietokannan nimi. Näitä muuttujia käytetään niin sanotun Connection-merkkijonon muodostamiseen, jota tarvitaan aina, kun tietokantayhteys avataan tietojen kirjoittamista tai lukemista varten. Merkkijonoon tarvittava MySQL-palvelimen IP-osoite saadaan Asetukset -luokasta. Mysli -luokka sisältää funktiot muutaHost, Kirjoitus ja Luku. MuutaHost -funktion tehtävä on muuttaa Connection -merkkijonoa, jos MySQL-palvelimen IP-osoitetta muutetaan. Kirjoitus -funktiota käytetään nimensä mukaisesti tietokantaan kirjoittamiseen. Funktion paluuarvo on tyypiltään boolean eli totuusarvo.

Luku -funktiota käytetään tietojen lukemiseen tietokannasta. Funktion paluuarvona on merkkijonoja sisältävä säiliö, joka on tyypiltään List.

TiedostonKasittely -luokkaa käytetään tekstimuodossa olevan asetustiedoston käsittelyä varten. Luokka sisältää funktiot: Luku ja Tallennus. Luokan muodostinfunktiossa luodaan Asetukset -luokan ilmentymä, johon Luku -funktiota käyttämällä luetaan tekstitiedostosta asetusarvot. Asetusarvoja ovat: käyttäjä, MySQL-palvelimen IP-osoite sekä RFID-lukijan lukuteho. Luku -funktio palauttaa aikaisemmin muodostinfunktiossa luodun Asetukset -luokan ilmentymän Form1 -luokan käyttöön, josta funktiota kutsutaan sovelluksen avautuessa. Tallennus -funktiota käytetään, kun asetusarvot halutaan tallentaa laitteen flash-muistiin. Tämän funktion kutsuminen tapahtuu aina, kun jotakin asetusarvoista muutetaan.

Kaapelit -luokkaa käytetään yleisesti käytettyjen kaapeleiden tietoja sisältävän tekstitiedoston käsittelyyn. Kaapelitiedot sisältävä tekstitiedosto sijaitsee myös laitteen flash-muistissa asetustiedoston tapaan. Luokan muodostinfunktio avaa tekstitiedoston, lukee sisällön ja käsittelee tiedot kahteen merkkijono-tauluun. Näitä merkkijono-tauluja ovat kaapeli_tyyppi sekä kaapeli_materiaali. Taulujen arvot ovat eritelty puolipisteillä, ja taulut ovat itse rajattu aloitusarvolla ja lopetusarvolla joita ovat ’[kaapeli_tyyppi]’ sekä ’[kaapeli_materiaali]’. Näille arvojen hakemiselle on luokassa Hae -funktio, jossa ensimmäisenä määritellään, kummasta merkkijono-taulusta arvoa haetaan sekä aikaisemmin yrityksen edustajan kanssa sovittu vastaava indeksinumero haettavasta arvosta. Tämä toteutus tehtiin työssä aikaisemmin mainitusta syystä, jossa Työnumero -viivakoodiin sisällytettiin liikaa tietoa, jotta se olisi mahtunut järkevästi tulostettavaan työmääräimeen.

7.3 Tietokantarakenne

Sovelluksen käyttämä tietokanta koostuu yhdestä taulusta, joka on nimeltään ’rfid’. Taulun pääavaimena toimii tietokenttä ’tag_id’, kentän tietotyypin ollessa varchar. Lähes kaikkien muiden tietokannassa

käytettyjen kenttien tietotyyppi on varchar, pois lukien aikaa ilmaisevat kentät. Näitä aikaa ilmaisevia kenttiä ovat kokoonpano1_aika sekä koestus_aika, joiden tietotyyppinä toimii timestamp. Timestamp on aikaa ilmaiseva tietotyyppi, joka on formaatissa VVVV-KK-PP TT:MM:SS. Se tarkoittaa suomeksi vuodet-kuukaudet-päivät tunnit:minuutit:sekunnit. Timestamp -tietotyyppisten kenttien oletusarvoina toimii CURRENT_TIMESTAMP, joka tarkoittaa ajankohtaa, jolloin tietue on luotu. Muiden tietokenttien oletusarvoina on merkki ’-’. Taulukosta 1 löytyy sovelluksessa käytetyn tietokantataulun rakenne, kenttien tietotyypit ja oletusarvot sekä esimerkit arvoista.

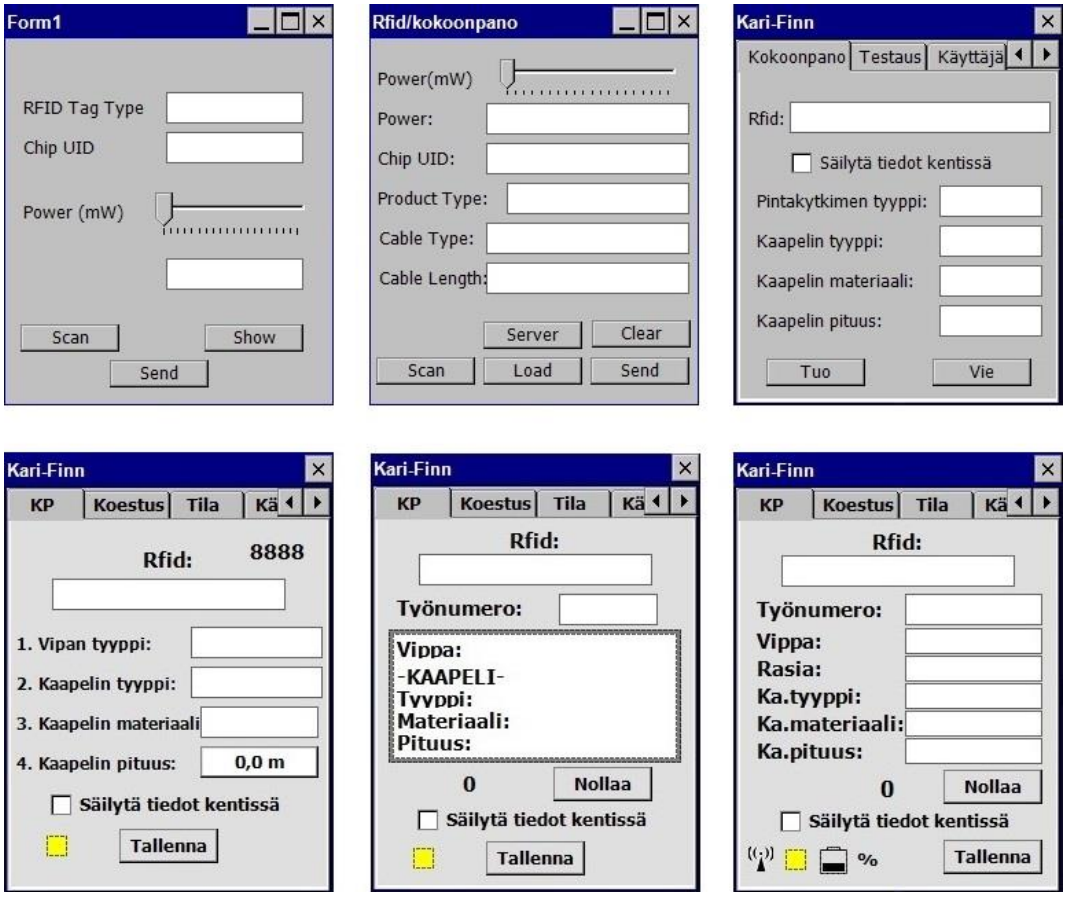
Taulukko 1. Tietokantataulun rakenne

Kentän nimi	Kentän tietotyyppi	Oletusarvo	Esimerkki arvo
tag_id	varchar (30)	’-’	E20060040236658A
tyonumero	varchar (30)	’-’	15486
kytkin_tyyppi	varchar (30)	’-’	M1HAu
rasia_tyyppi	varchar (30)	’-’	Au
kaapeli_tyyppi	varchar (30)	’-’	2x1mm2
kaapeli_materiaali	varchar (30)	’-’	PVC
kaapeli_pituus	varchar (30)	’-’	2,0m
kokoonpanija1_id	varchar (50)	’-’	Pekka
kokoonpano1_aika	timestamp	CURRENT_TIMESTAMP	2010-05-21 08:03:09
koestus_tulos	varchar (30)	’-’	EI TOIMI
hylkays_syy	varchar (30)	’-’	Toleranssi
hylkays_syy_kommentti	varchar (200)	’-’	’-’
koestaja_id	varchar (50)	’-’	Sami
koestus_aika	timestamp	CURRENT_TIMESTAMP	2010-05-21 12:54:22
tila	varchar (30)	’-’	KOESTETTU

7.4 Käyttöliittymä

Form1 -luokka toteuttaa sovelluksen pääikkunan, jossa erilaiset näkymät on jaettu välilehtiin. Näitä näkymiä ovat: KP, Koestus, Tila, Käyttäjä sekä Asetukset. Näkymät, jotka ovat myöhemmissä kuvissa eriteltyinä, ovat juuri siinä mittasuhteessa, kuten ne näkyvät käytetyissä käsikäyttöisissä RFID-lukijoissa. Tämä laitteen näytön pieni resoluutio teki käyttöliittymän suunnittelusta haasteellista.

Kuvassa 9 näkyy Kokoopano -näkymän kehittyminen sovelluksen kehityksen aikana. Ensimmäisenä kuvassa on prototyypin ensimmäinen versio, vierestä löytyy prototyypin valmis versio, joka esiteltiin kohdeyritykselle. Kolmantena kuvassa on versio kesältä 2010 ja neljäntenä versio syksyltä 2010. Viides versio on joulukuulta 2010 ja viimeisenä on näkymän lopullinen versio.



Kuva 9. Graafisen käyttöliittymän evoluutio

Ensimmäisenä on vuorossa KP -näkymä, joka on nähtävillä kuvassa 10. KP on lyhenne sanasta kokoonpano. Kyseistä näkymää käytetään tietojen keräämiseen tuotteesta, kun tuote on vielä kokoonpanossa.

Kuva 10. KP -näkymä

Ikkunan otsikossa lukee yrityksen nimi sekä lisäksi käyttäjän nimi, kunhan se on ensin luettu viivakoodilla. Näkymästä löytyy yksi RFID-tekstikenttä, sekä kuusi viivakoodi-tekstikenttää. Näistä jokainen on merkitty omalla nimellään. Tekstikenttiin lukeminen tapahtuu painamalla Scan -painiketta tai erillistä liipaisinta. Kun tiedon luku RFID-kenttään epäonnistuu, ilmestyy kenttään teksti ’Tunnistetta ei löytynyt’. Fokus pysyy edelleen samassa kentässä, kunnes tunnisteen lukeminen onnistuu. Tällöin fokus siirtyy automaattisesti seuraavaan kenttään. Mikäli tieto on viivakoodia luettaessa oikeanlaista, siirtyy fokus automaattisesti seuraavaan kenttään. Muutoin kenttään ilmestyy teksti ’VÄÄRÄ’ ja kentän väri muuttuu punaiseksi. Aktiivinen kenttä on oranssinvärinen. KP -näkymässä tarvittavia luettavia kenttiä ovat RFID sekä Työnumero, sillä Työnumero -viivakoodi sisältää kaikki valmistettavan tuotteen tiedot jotka sovellus täyttää käsittelemällä kyseisen viivakoodin.

Mikäli on tarvetta vaihtaa jonkin tietyn kentän tietoa, voidaan kenttä valita sormella painamalla tai laitteesta löytyvällä osoitinkynällä painamalla haluttua kenttää. Tällöin vain luetaan viivakoodi Scan -painiketta tai

erillistä liipaisinta painamalla. Kun tämä viivakoodin luku on suoritettu onnistuneesti, siirtyy fokus automaattisesti Tallenna -painikkeelle, jota painamalla kerätyt tiedot tallentuvat tietokantaan.

Tietojen tietokantaan tallentamisen ajaksi näytetään käyttäjälle Odota -ikkunaa. Tallenna -painikkeen painamisen voi suorittaa joko Scan -painiketta tai liipaisinta painamalla, tai sitten painamalla painiketta kosketusnäytöltä. Kun tiedot on saatu kirjoitettua tietokantaan onnistuneesti kaikki kentät nollautuvat ja fokus palaa RFID-kenttään.

Tietojen uudelleen syöttämiseltä voidaan välttyä, mikäli käytetään Säilytä tiedot kentissä -valintaruutu-kontrollia. Tällöin kaikki viivakoodilla luettavat tiedot säilyvät kentissä ja ainoastaan RFID-kenttä nollautuu. Tämä on hyödyllistä, kun on useita samanmallisia tuotteita, joiden tietoja syötetään toistuvasti tietokantaan.

Kun RFID-tunnisteeseen sidotut tiedot kirjoitetaan tietokantaan ensimmäistä kertaa, kasvaa RFID-laskuri yhdellä. RFID-laskurin voi halutessaan nollata painamalla Nollaa -painiketta. Tällöin näkyviin avautuu erillinen Nollaa laskuri -ikkuna. RFID-laskuri nollautuu myös automaattisesti, laskurin arvon ylittäessä luvun 999. Näkymässä on myös kuvakkeet ja indikaattorit langattoman verkkoyhteyden sekä akun tiloille. Verkkoyhteyden ollessa kunnossa väri on vihreä, ja mikäli yhteyttä ei ole väri on punainen. Akun tila on ilmoitettu prosentteina ja ladattaessa akkua kohdassa lukee ’Lataa…’.

Koestus -näkymää käytetään, kun tuotteeseen halutaan liittää tai muokata koestuksessa kerättyjä tietoja. Näkymä löytyy kuvasta 11. Kerättäviä tietoja ovat läpäisy, hylkäyssyy, sekä mahdollinen tarkentava kommentti hylkäyssyystä. Nämä tiedot kerätään viivakoodeja lukemalla. Läpäisy -tekstikenttään sallittavia arvoja ovat: ’TOIMII’ sekä ’EI TOIMI’. Mikäli läpäisy-tekstikentän arvo on viivakoodin lukemisen jälkeen ’TOIMII’, muuttuvat alemmat viivakoodi-tekstikentät pois käytöstä siten, ettei niihin voida syöttää tietoa. Tämän jälkeen fokus siirtyy automaattisesti Tallenna -painikkeelle.

Kuva 11. Koestus -näkymä

Hylkäyssyy -tekstikentän mahdollisia arvoja ovat esimerkiksi: ’Sahkoinen’ ja ’Toleranssi’. Kommentti-tekstikenttään on tarpeen mukaan mahdollista kirjoittaa käsikäyttöisen RFID-lukijan fyysisellä näppäimistöllä tai kosketusnäytöllä olevalla näppäimistöllä. Tähän kenttään ei kuitenkaan ole pakko kirjoittaa mitään.

Tämä näkymä toimii hyvin samalla tapaa kuin aikaisemmin käsitelty KP -näkymä. Poikkeuksena on kuitenkin se, että RFID-tunnistetta luettaessa, ja mikäli tunnisteeseen on jo liitetty tietoja koestuksesta, luetaan tiedot tietokannasta ja ne kirjoitetaan tekstikenttiin käyttäjän nähtäville. Näitä arvoja on mahdollista tämän jälkeen muokata, ja tallentaa muutokset tietokantaan. Tietokannasta luettaessa tai sinne kirjoittaessa käyttäjälle näytetään Odota -ikkunaa. Kun tiedot on tallennettu tietokantaan Tallenna -painiketta painamalla, sovellus tyhjentää kaikki tekstikentät ja siirtää fokuksen takaisin RFID-tekstikenttään.

Tila -näkymää käytetään, kun halutaan muuttaa tuotteen tilaa tuotannossa. Tila -näkymä on nähtävillä kuvassa 12. RFID-tunnisteen lukemisen onnistuttua, sovellus hakee tuotteeseen liitetyt tiedot, joita ovat: tuotteen malli, koestuksen tulos sekä tuotteen nykyinen tila tuotannossa. Tämä tila on mahdollista muuttaa valintavalikon kautta. Valittavissa olevat tilat on väritetty eri väreillä ja valittavia tiloja ovat: TUOTANTO (keltainen), KOESTETTU (limenvihreä), POISTO (punainen) ja VARASTO (vaaleansininen).

Kuva 12. Tila -näkymä

Tilaa ei voi kuitenkaan vaihtaa, ellei Varmistus -valintaruutua ole aktivoitu. Valintaruudun ollessa tyhjänä Tallenna -painike ei ole aktiivinen eikä näin ollen painettavissa. Painike muuttuu aktiiviseksi, kun valintaruutu aktivoidaan. Tällöin on mahdollista muuttaa tuotteen tila tietokantaan painamalla Tallenna -painiketta. Tietokannasta luettaessa tai sinne kirjoittaessa käyttäjälle näytetään Odota -ikkunaa. Kun tiedot on tallennettu tietokantaan painamalla Tallenna -painiketta, sovellus tyhjentää kaikki tekstikentät ja siirtää fokuksen takaisin RFID-tekstikenttään.

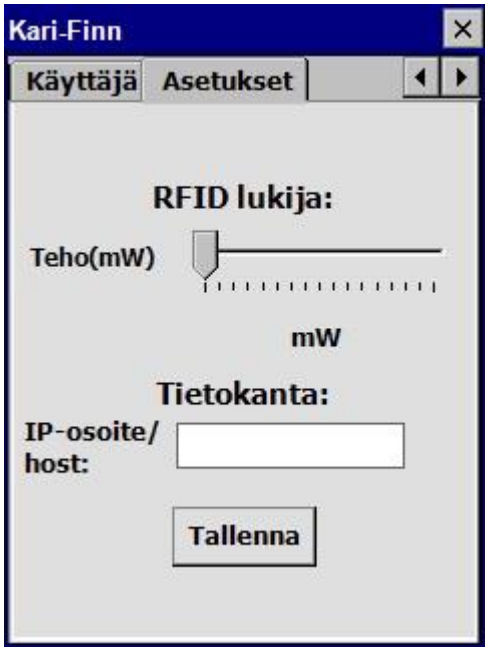
Käyttäjä -näkymää käytetään käyttäjän vaihtamiseen. Näkymä on nähtävillä kuvassa 13. Käyttäjän vaihtaminen tapahtuu lukemalla viivakoodi tekstikenttään. Mikäli luettu käyttäjä on validoinnin osalta väärä tai sama kuin nykyinen käyttäjä, ei Vaihda käyttäjä -painiketta aktivoida. Fokus pysyy tässä tapauksessa tekstikentässä. Vaihda käyttäjä -painiketta painettaessa muuttuu pääikkunan otsikkoon käyttäjän nimi, ja samalla tieto tallennetaan asetustiedostoon. Mikäli luetun käyttäjän nimen arvo on Admin, ilmestyy ikkunaan näkyviin välilehti Asetukset. Muille käyttäjille välilehti ei ole näkyvillä.

Kuva 13. Käyttäjä -näkymä

Asetukset -näkymässä, joka löytyy kuvasta 14, on mahdollista muuttaa sovelluksen kahta asetusta. Nämä asetukset ovat RFID-lukijan lukuteho sekä tietokantapalvelimen IP-osoite tai hostname. Aikaisemmin käytössä olleet asetukset ladataan sovelluksen käynnistyessä asetustiedostosta.

Lukutehon muuttaminen tapahtuu liukusäätimen avulla ja valittu teho on näkyvissä liukusäätimen alla milliwatteina. Tietokantapalvelimen IP-osoite

tai hostname voidaan lukea viivakoodilla tai näppäimistöllä syöttämällä. Mikäli kumpaakaan asetusarvoista ei ole muutettu, pysyy Tallenna -painike toimimattomana. Kun taas jompaakumpaa asetuksista muutetaan, Tallenna -painike muuttuu aktiiviseksi. Painiketta painamalla otetaan käyttöön uudet valitut asetukset ja samalla nämä asetukset tallennetaan asetustiedostoon. Asetukset -näkymä on näkyvissä vain käyttäjälle Admin.



Kuva 14. Asetukset -näkymä

Muita ikkunoita pääikkunan lisäksi ovat: Odota… sekä Nollaa laskuri. Nämä ikkunat ovat esillä kuvassa 15.

Odota… -ikkunan toteuttaa luokka Form2. Ikkunaa näytetään käyttäjälle sen aikaa, kun tietoja luetaan tietokannasta tai tietokantaan kirjoitetaan. Ikkunan tarkoituksena on ilmoittaa käyttäjälle, että mitä sovelluksessa tapahtuu sekä samalla estää mahdolliset virhesyötteet.

Kuva 15. Sovelluksen ikkunat: Nollaa laskuri ja Odota…

Nollaa laskuri -ikkunaa käytetään RFID-laskurin nollauksen varmistamiseen. Ikkunan toteuttaa luokka VarmistusForm. Painamalla Peruuta -painiketta ikkuna piilotetaan, jolloin palataan takaisin sovelluksen pääikkunan Kokoonpano -näkymään. Painettaessa Kyllä -painiketta varmistetaan RFID-laskurin nollaus, joka tyhjentää TID-sarjanumerot sisältävän säiliön sekä muuttaa Kokoonpano -näkymässä olevan laskurin lukeman arvoon nolla. Tämän jälkeen ikkuna piilotetaan ja fokuksen saa pääikkunan Kokoonpano -näkymän RFID-tekstikenttä.

8 YHTEENVETO

Projektin tavoitteena oli sovelluksen kehittäminen käsikäyttöiselle RFID-lukijalle, jonka avulla yrityksen olisi mahdollista ottaa käyttöön RFID-teknologian avulla toteutettu tuotteiden yksilöinti ja seuranta. Aikataulullisesti projektin oli tarkoitus kestää vuoden 2010 kesästä jouluun, mutta sovellusta jatkokehitettiin vielä seuraavan vuoden kesällä. Projekti oli onnistunut, sillä tavoitteisiin päästiin ja kehitetty sovellus jäi tilaan, jossa sitä voidaan hyödyntää yrityksen tuotannossa tarkoituksenmukaisesti.

Kehitystyön keskeisimmäksi ongelmaksi muodostui riittävän yksinkertaisen käyttöliittymän suunnittelu. Tämä ongelma saatiin ratkaistua lyhyiden käyttökokemusten perusteella, ja kehittämällä käyttöliittymää entisestään jokaiseen uuteen versioon. Toistuvaan saman mallisten tuotteiden yksilöintiin riittää vähintään, että ainoastaan yhteen sovelluksen kentistä tarvitsee lukea tieto, jonka jälkeen tarvittavat tiedot tallennetaan tietokantaan. Tämän toiminnallisuuden suorittamiseksi, käsikäyttöisestä RFID-lukijasta tarvitsee painaa vain yhtä painiketta kaksi kertaa.

Yritys ei ollut ottanut sovellusta käyttöön ainakaan vuoden 2011 kesän loppuun mennessä. Sovelluksen käytöstä ei ole raportoituja pidempiaikaisia käyttökokemuksia.

Sovellusta olisi vielä mahdollista kehittää edelleen. Seuraavan version kehittämistä varten tulisi saada palautetta sovelluksesta käyttäjiltä, heidän pidempiaikaisten käyttökokemusten pohjalta. Palautteen ansiosta sovellusta olisi mahdollista kehittää käyttäjien tarpeita vastaaviksi.

Tässä vielä listattuna muutamia kehitysideoita sovelluksen jatkokehitystä ajatellen. Tietokannan hajauttaminen useampaan tauluun olisi kannattavaa, jotta tietokantaa olisi mahdollista hyödyntää aikaisempaa tehokkaammin sovelluksen sisäisessä toiminnassa. Graafista käyttöliittymää voitaisiin kehittää modernimpaan ja entistä käyttäjäystävällisempään muotoon.

Muita kehitysideoita ovat koodista puuttuvien tarkistuksien toteuttaminen, etenkin laitteen rajapintafunktioiden yhteyteen. Esimerkiksi käytettyjen RFID-lukijoiden RFID-lukutehojen maksimiarvot eroavat siten, että Morphic-mallin arvo on 100 mW ja PL3000-mallissa arvo on 200 mW. Myös sovelluksen käyttämien tekstitiedostojen käsittelyistä puuttuvat tarkistukset tulisi toteuttaa. Näiden tarkistusten tehtävänä olisi varmistaa, että tekstitiedosto olemassa tiedostoa luettaessa. Tiedoston puuttuessa tulisi sovelluksen luoda oletusversio kyseisestä tiedostosta.

LÄHTEET

Achievetec 2014. RFID [viitattu 12.4.2018]. Saatavissa: http://www.achievetec.com/rfid.php

Ahsan, K., Shah, H. & Kingston, P. 2010. RFID Applications: An Introductory and Exploratory Study. IJCSI International Journal of Computer Science Issues 1/2010, 1 - 7. [viitattu 8.10.2015]. Saatavissa: http://www.ijcsi.org/papers/7-1-3-1-7.pdf

Bercero, C. 2009. What is CLR (Common Language Runtime) [viitattu 14.4.2018]. Saatavissa: http://carlosbercero.com/post/?post=What_is_CLR_%28Common_Language_Runtime%29

CNRFID 2018. RFID frequency ranges [viitattu 12.4.2018]. Saatavissa: http://www.centrenational-rfid.com/rfid-frequency-ranges-article-16-gb-ruid-202.html

GS1 2016. Regulatory status for using RFID in the EPC Gen2 (860 to 960 MHz) band of the UHF spectrum [viitattu 29.3.2018]. Saatavissa: https://www.gs1.org/sites/default/files/docs/epc/uhf_regulations.pdf

IEEE GlobalSpec 2018. RFID Tags Information [viitattu 27.3.2018]. Saatavissa: https://www.globalspec.com/learnmore/data_acquisition_signal_conditioning/data_input_devices/rfid_tags

Ikonen, R. 2007. RFID Hyvinkään Laurea-kirjastossa [viitattu 14.4.2018]. Saatavissa: https://www.kreodi.fi/arkisto/artview269.html

iXtenso 2007. Nordic ID PL-Reihe: Flexibilität neu definiert [viitattu 17.4.2018]. Saatavissa: http://ixtenso.com/de/story/9126-nordic-id-pl-reihe-flexibilitaet-neu-definiert.html

Microsoft 2016. What is "managed code"? [viitattu 14.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/dotnet/standard/managed-code

Microsoft 2017. Common Language Runtime (CLR) [viitattu 14.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/dotnet/standard/clr

MSDN 2009. .NET Framework Class Library [viitattu 14.4.2018]. Saatavissa: https://msdn.microsoft.com/en-us/library/ms229335(v=vs.90).aspx

MSDN 2018a. .NET Compact Framework [viitattu 13.4.2018]. Saatavissa: https://msdn.microsoft.com/en-us/library/f44bbwa1.aspx

MSDN 2018b. ADO.NET [viitattu 17.4.2018]. Saatavissa: https://msdn.microsoft.com/en-us/library/aa286484.aspx

MSDN 2018c. Differences Between the .NET Compact Framework and the .NET Framework [viitattu 13.4.2018]. Saatavissa: https://msdn.microsoft.com/fi-fi/library/2weec7k5%28en-us%29.aspx

Nordic ID 2010. Nordic ID Multiple Hardware Layers [MHL] [viitattu 17.4.2018]. Saatavissa: https://support.nordicid.com/MHL/

Nordic ID 2018a. Nordic ID Morphic datasheet [viitattu 18.4.2018]. Saatavissa: https://www.prosign.dk/Nordicid/Nordic%20ID%20Morphic_Datasheet_English_29%208.pdf

Nordic ID 2018b. Nordic ID Morphic [viitattu 17.4.2018]. Saatavissa: https://www.nordicid.com/en/home/products-barcode-uhf-rfid-reader-writer/mobile-readers/nordic-id-morphic_uhf-rfid_barcode_reader-writer-scanner/

Nordic ID 2018c. Nordic ID PL3000 UHF RFID 200 mW datasheet [viitattu 18.4.2018]. Saatavissa: https://www.prosign.dk/Nordicid/PL3000_200_ENG.pdf

Oracle 2018a. 1.3.1 What is MySQL? [viitattu 16.4.2018]. Saatavissa: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Oracle 2018b. Chapter 1 Introduction to MySQL Connector/Net [viitattu 16.4.2018]. Saatavissa: https://dev.mysql.com/doc/connector-net/en/connector-net-introduction.html

Recall 2015. A brief history of RFID technology [viitattu 27.8.2015]. Saatavissa: http://www.recall.fi/united-kingdom/articles/rfid-brief-history

RFID4U 2018a. How to Select a Correct Tag – Frequency [viitattu 29.3.2018]. Saatavissa: https://rfid4u.com/rfid-basics-resources/how-to-select-a-correct-tag-frequency/

RFID4U 2018b. Inductive and Backscatter Coupling [viitattu 10.4.2018]. Saatavissa: https://rfid4u.com/rfid-basics-resources/inductive-and-backscatter-coupling/

RFID4U 2018c. The RF in RFID [viitattu 12.4.2018]. Saatavissa: https://rfid4u.com/rfid-basics-resources/the-rf-in-rfid/

RFID Lab Finland ry 2015. RFID-tietoutta [viitattu 8.9.2015]. Saatavissa: http://rfidlab.fi/rfid-tietoutta

RFID Lab Finland ry 2018. RFID-teknologian soveltamisalueita [viitattu 3.4.2018]. Saatavissa: http://www.rfidlab.fi/rfid-teknologia/soveltamisalueet/

Roberti, M. 2005. The history of RFID technology [viitattu 27.8.2015]. Saatavissa: http://www.rfidjournal.com/articles/view?1338

Smiley, S. 2016. Operating Principles: Coupling [viitattu 5.4.2017]. Saatavissa: https://blog.atlasrfidstore.com/operating-principles-coupling

Techspirited 2018. Pros and Cons of RFID Technology [viitattu 27.3.2018]. Saatavissa: https://techspirited.com/pros-cons-of-rfid-technology

UPM Raflatac 2009. UPM Raflatac ShortDipole^{x} [viitattu 26.4.2018]. Saatavissa: http://www.fastrfid.com/raflatac/UHF/tech_speck_3001276_a4.pdf

Violino, B. 2005. The Basics of RFID Technology [viitattu 5.4.2017]. Saatavissa: http://www.rfidjournal.com/articles/view?1337