KULUNVALVONTAJÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Automaatiotekniikan koulutusohjelma

Valkeakoski, syksy 2016

Oma Allekirjoituksesi

Eetu-Tuomas Viertola

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Valkeakoski
Automaatiotekniikan koulutusohjelma

Tekijä Eetu-Tuomas Viertola **Vuosi** 2016

Toimeksiantaja Kuljetinsähkö Tampere Oy

Työn nimi Kulunvalvontajärjestelmän suunnittelu ja toteutus

TIIVISTELMÄ

Tavoitteena oli valmistaa kulunvalvontalaite, joka tunnistaa pääovesta sisään kulkevat henkilöt, sekä aulassa olevasta kääntöportista kulkevat henkilöt, suunta yksilöitynä. Tunnistus tapahtuisi RFID-tageilla, jotka alueella kulkevilla työntekijöillä olivat jo valmiina, sekä lukijalaitteilla, jotka olivat valmiiksi asennettuna kääntöportille ja pääovelle. Kääntöportin kulkutiedoista myös koottaisiin tuntilistat, jotka toimivat palkanmaksun perusteena.

Tuloksena syntyi mikrokontrolleripohjainen laite, joka ymmärtää signaaleja Wiegand 26-protokollan mukaisilta RFID-lukijoilta, sekä tietokoneohjelma, joka tallentaa kulkutiedot ja koostaa halutunlaiset tuntilistat, jotka voidaan tulostaa paperille. Tuloksen saavuttamisen tärkeimpiä vaiheita oli Wiegandin protokollaan tutustuminen.

Johtopäätöksenä todettiin kulunvalvonnan olevan laaja ja mielenkiintoinen ala, johon liittyy suuret määrät herkkäluontoista tietoa. Toteutustapoja kulunvalvontaan on useita. Projekti todettiin onnistuneeksi, asiakkaan saadessa käyttöönsä halutun laitteiston.

Sivut 81 s. + liitteet 4 s.

HAMK
HÄMEEN AMMATTIKORKEAKOULU

ABSTRACT

Valkeakoski
Degree Programme in Automation Engineering

Author	Eetu-Tuomas Viertola	**Year** 2016
Commissioned by	Kuljetinsähkö Tampere Oy	
Subject of Bachelor's thesis	Designing and manufacturing an acces control system	

ABSTRACT

The goal of the thesis was to produce an access control device, which would recognize persons entering the building using the maindoor and persons accessing the turnstile on the premise. Recognizion was to be implemented through with RFID-identifiers, which the employees on the premises already possessed, and through RFID-readers, which were already installed at the main door and the turnstile. Data acquired from the turnstile was to be assembled into listings of workhours, which were to be used as a basis for calculating salaries.

As a result of this project a microcontroller-based device was produced, which can read RFID-readers using the Wiegand 26-protocol and a computer program, which saves data from the readers and assembles them into required listings, which can then be printed on paper. One of the most important steps in the project was to get familiarized with the Wiegand-protocols.

As a summary it can be stated that access control is a very wide field, which carries a lot of sensitive information. There are many solutions as to the identification of access control. The project was a succes as the commissioner got the equipment they needed for the premises.

Pages 81 p. + appendices 4 p.

SISÄLLYS

1 JOHDANTO

Vuonna 2012 Virolan Puutarhan kulunvalvontalaite rikkoontui verkkopäivityksen yhteydessä. Laitteen korvaamista harkittiin, mutta sen käyttöliittymään ei oltu tyytyväisiä ja arvioitiin, että vain hieman yli alkuperäisen laitteiston uusimishinnalla, voitaisiin suunnitella oma laitteisto ja ohjelmisto korvaamaan vanhat laitteet. Arvio perustui oletukseen, että lukijat käyttävät RS232-pohjaista sarjaliikennöintiä, mikä osoittautui vääräksi oletukseksi.

Tutustuminen lukijoihin osoitti kyseessä olevan Wiegandin protokollaa noudattavat lukijat, minkä pohjalta tutustuttiin Wiegandin protokollaan ja protokollan taustoihin.

Tavoitteena oli tuottaa kulunvalvontaan tarkoitettu laitteisto ja ohjelmisto, jonka tehtävänä on sekä kirjata kiinteistössä tapahtuvaa kulkua, että koostaa kiinteistössä työskentelevän henkilöstön työaikalistat.

Tehtävä aloitettiin tutustumalla Wiegandin protokollaan, suunnittelemalla sitä lukemaan kykenevä elektroniikka, joka toimii välittäjänä tietokoneohjelman, avattavien kohteiden ja RFID-lukijoiden välillä. Tämän jälkeen kirjoitettiin tietokoneohjelma, joka tarkastaa järjestelmässä luetut tunnisteet, sekä tallentaa ja näyttää haluttuihin raportteihin tarvittavat tiedot.

Työn tilaaja on Kuljetinsähkö Tampere Oy ja Virolan Puutarha on tuotteen käyttöönottanut loppuasiakas. Kun tässä työssä puhutaan asiakkaasta, sillä tarkoitetaan Virolan Puutarhaa.

2 WIEGAND

2.1 Historiaa

Useiden kulunvalvonnassa käytettyjen laitteiden liikennöinti käyttää Wiegandin protokollaa, joka perustuu John Wiegandin 70-luvulla patentoimaan Wiegand-ilmiöön. John Wiegand alkoi tutkia fysikaalista ilmiötä, joka on nykyään nimetty hänen mukaansa vuonna 1965. Ilmiössä erityisesti käsitelty ferromagneettinen johdin, muuttaa magneettisuutensa suuntaa äkillisesti, luoden magneettikenttään voimallisen pulssin, joka on helppo havaita elektroniikan avulla. (Walker 1979, 102-104.)

Koska magneettikentän muutokset johtimessa voidaan järjestää kestomagneeteilla, ei Wiegand-ilmiötä hyödyntävissä antureissa tarvita välttämättä erillistä sähkönsyöttöä. Ilmiö ei myöskään ole altis ulkoisille häiriöille, sekä toimii laajalla lämpötila-alueella. Pulssien voimakkuuteen ja pituuteen ei myöskään vaikuta luettavan kohteen liikkumisnopeus. Ilmiölle löytyi nopeasti sovelluksia esimerkiksi teollisuudessa ja tunnistautumistekniikassa. (Walker 1979, 102-104.)

2.2 Wiegand-johdin

Wiegand-johdin koostuu ferromagneettisesta johtimesta jota on pehmitetty venyttämällä ja kiertämällä, minkä jälkeen sen pinta on kovetettu hehkuttamalla. Tämä luo johtimelle erilaiset magneettiset ominaisuudet johtimen ytimeen ja kuoreen. Johdin on yleensä valmistettu metalliseoksesta, joka koostuu pääsiassa koboltista, vanadiumista ja raudasta. (Wigen, 1975, 100.)

Perustilassa johtimen magneettisten alueiden magneettikentät ovat satunnaisesti suuntautuneita ja kumoavat toisensa. Kun johdin tuodaan ulkoiseen magneettikenttään, sen magneettiset alueet järjestäytyvät ulkoisen magneettikentän mukaisesti, minkä jälkeen johdin on magnetisoitu ja omaa oman magneettikenttänsä. Pehmeämpi ydin järjestyy pienemmällä magneettikentällä, kuin kova kuori. Kun magneettikentän vahvuus ylittää kynnyksen, järjestyy myös kovemman kuoren magneettikenttä ja toisin kuin normaalin johtimen magneettikenttä, joka järjestyy hitaasti, tapahtuu kuoren muutos todella nopeasti, luoden huomattavan pulssin magneettikenttään. Nämä muutokset magneettikentässä huomattavia pulsseja ja mikäli niiden lähelle on sijoitettu kela, indusoituu kelaan muutoshetkellä huomattavan vahva jännitepulssi. (Wigen, 1975, 100.)

Kun johtimen magneettikenttä on suunnattu kestomagneetin avulla, ei sen kuljettaminen uudelleen magneetin ohitse muuta valmiiksi suuntautunutta magneettikenttää. Tämä on otettava huomioon käytännön sovelluksissa, joissa usein käytetään yhtä kestomagneettia nollaamaan johtimen magneettikenttä ja toista kestomagneettia varsinaista pulssin lukemista varten. Samalla kelalla voidaan myös lukea molemmat magneettikentän muutokset, milloin toisen pulssin tuottama jännitepulssi on positiivinen ja toinen negatiivinen. Pulssi on havaittavissa sekä ytimen, että kuoren magneettikentän vaihtaessa suuntaa. Ytimen magneettikentän suunnan vaihtumisen aiheuttama pulssi on voimakkuudeltaan huomattavasti vahvempi, kuin kuoren magneettikentän suunnan vaihtumisen aiheuttama pulssi. (Wigen 1975, 102.)

2.3 Wiegand-kulkukortti ja lukulaite

Wiegand-kulkukortti on luottokortin kokoinen kortti, johon on upotettu Wiegand-johtimia kahteen riviin, joista toinen rivi merkitsee ”0”-bittejä ja toinen rivi ”1”-bittejä. Kun kortti kuljetetaan lukijan läpi, jossa ne kulkevat ensin kestomagneetin ohi, milloin niiden magneettikentät asettuvat lukua edeltävään tilaan. Rivit kulkevat magneetin eri napojen ohi, milloin niiden magneettikenttien suunnat ovat vastakkaisia toisiinsa nähden. Lukupäässä on toisinpäin asetettu kestomagneetti, sekä lukukela. Kun magnetisoidut Wiegand-johtimet kulkevat magneetin ja lukukelan ohi, aiheuttavat ne kelan napoihin vastakkaisen suuntaiset jännitepulssit. Nämä pulssit muutetaan luvussa 2.4 kuvatun protokollan mukaisiksi pulsseiksi datalinjoihin. Pulssien ajoitus protokollassa määräytyi kortin fyysisten mittojen mukaan. Alkuperäisessä lukijassa, kortti pudotettiin ohjattuna lukupään ohi. (Davis, 2009.)

2.4 Protokolla

Wiegandin protokolla on yksisuuntainen tiedonsiirtomenetelmä, jota käytetään pääasiassa henkilöiden tunnistamiseen. Protokolla on kehitetty 1970-luvulla ja siitä on kehkeytynyt de facto -standardi kulunvalvontalaitteissa ja suurin osa kulunvalvontalaitteista nykyäänkin tukee sitä. (Davis 2009.)

Wiegandin protokollia on olemassa eripituisia. Jotkut laitevalmistajat käyttävät myös omia variaatioitaan, eroten alkuperäisestä protokollasta. Myös tässä työssä päätettiin käyttää omaa variaatiota, koska alkuperäisen Wiegandin määrittelyjä ei tarvittu. Oma käytäntö on kuvailtu luvussa 3.5.1. 26-bittinen Wiegand valikoitui olemassa olevien RFID-lukijoiden mukaan. Taulukossa 1 on esitetty alkuperäinen 26 bittisien Wiegand viestin rakenne. (HID, 2005.)

Taulukko 1. 26-bit Wiegand viestin rakenne

Bitti numero	Tarkoitus
1	Parillinen pariteetti biteille 2-13
2-9	Laitoskoodi 0-255 (bitti 2 on eniten merkitsevä bitti)
10-25	Tunnistenumero 0-65635 (bitti 10 on eniten merkitsevä bitti)
26	Pariton pariteetti biteille 14-25

Data syötetään kahta johdinta pitkin pulsseina. Toinen johdin syöttää 0-bittejä ja toinen 1-bittejä. Nämä johtimet tunnetaan datalinjoina DATA0 ja DATA1. Kun dataa ei lähetetä johtimen potentiaali on ylösvetovastuksen kautta korkeassa logiikkatasossa, yleensä +5V DC. Kun linja lähettää bitin, johtimessa näkyy nollapotentiaalinen pulssi. Pulssin kesto on 20-100 mikrosekuntia ja pulssien väli on 200 mikrosekuntia – 20 millisekuntia. Tyypilliset kestot ovat 100 mikrosekuntia pulssille ja yksi millisekunti pulssien välille. Pulsseja lähetetään jonossa kunnes haluttu bittimäärä on täynnä. (RobotShop, n.d.)

2.5 Wiegandin protokollaa käyttäviä laitteita

Kulkukortteja enää harvoin valmistetaan käyttäen Wiegandin johtimia. Wiegandin protokolla sen sijaan on edelleen erittäin yleinen kulunvalvontalaitteissa ja sitä käytetään useampaankin käyttötarkoitukseen. Tutustuttiin muutamaan kulunvalvonnassa yleiseen laitteeseen, jotka käyttävät protokollaa.

Yleisin Wiegandin protokollaa käyttävä laite on RFID-lukija, jolla luetaan RFID-tunnisteita, jotka on voitu upottaa esimerkiksi avaimenperään, tai luottokortin kokoiseen korttiin. RFID-tunnisteeseen on koodattu halutun formaatin muotoinen numero, jonka lukija muuttaa Wiegandin protokollan mukaisiksi signaaleiksi. Yleensä lukijaan on sisällytetty muitakin toimintoja, kuten merkkivalo, jolla voidaan näyttää lukutapahtuma tai vika järjestelmässä. Merkkivalo toimii yleensä itsenäisesti, mikäli sen ohjausjohdinta ei ole kytketty ja ohjattuna, mikäli johdin on kytketty. Toinen yleinen varuste on äänisummeri, jonka toiminta on yleensä vastaava kuin

merkkivalon. Kolmas yleinen varuste on valovastus, tai painike, lukijan piiloon jäävältä puolelta, jolla voidaan havaita mikäli lukija irrotetaan kiinteästä asennuspaikastaan. Usein lukijassa on myös johdotus, jolla voidaan valita käyttääkö lukija 26- vai 34-bittistä protokollaa.

Toinen yleinen protokollaa käyttävä laite on numeronäppäimistö, joista tutustuttiin RCI:n valmistamaan 5355R malliin. Näppäimistöjen käyttämät viestien pituudet vaihtelevat valmistajan mukaan, kuten myös signaalin koodaus. 5355R voidaan asettaa lähettämään jokaista painallusta kohden viiden bitin purskeen, joka noudattaa Wiegandin protokollaa, tai 26 bittisen koodin, jossa pariteetit ovat, kuten alkuperäisessä 26 bitin koodissa, mutta data jaetaan neljän bitin purskeisiin, joista jokainen merkitsee yhden numeron painallusta näppäimistöltä. Kyseinen laite sisältää myös RFID-lukijan, joten sillä voitaisiin toteuttaa kahdessa osassa luettava koodi, joista toinen osa tulisi numeronäppäimistöltä ja toinen RFID-tunnisteesta. Tällöin hukatun avaimen väärinkäyttö vaatisi myös tunnusnumeron tietämistä tunnisteen väärinkäyttämiseksi. (RCI 2015.)

Näiden lisäksi usein käytetään biometrisiä tunnisteita, kuten sormenjälkeä tai kasvojen tunnistamista. Tällaisissa tunnisteissa tarvitaan laite, joka lukee biometrisen tunnisteen ja sisältää muistin, johon tunnisteet on tallennettu. Biometrinen tunniste yhdistetään numeroon, joka muunnetaan Wiegandin protokollan mukaisiksi signaaleiksi. (Crypton, n.d.)

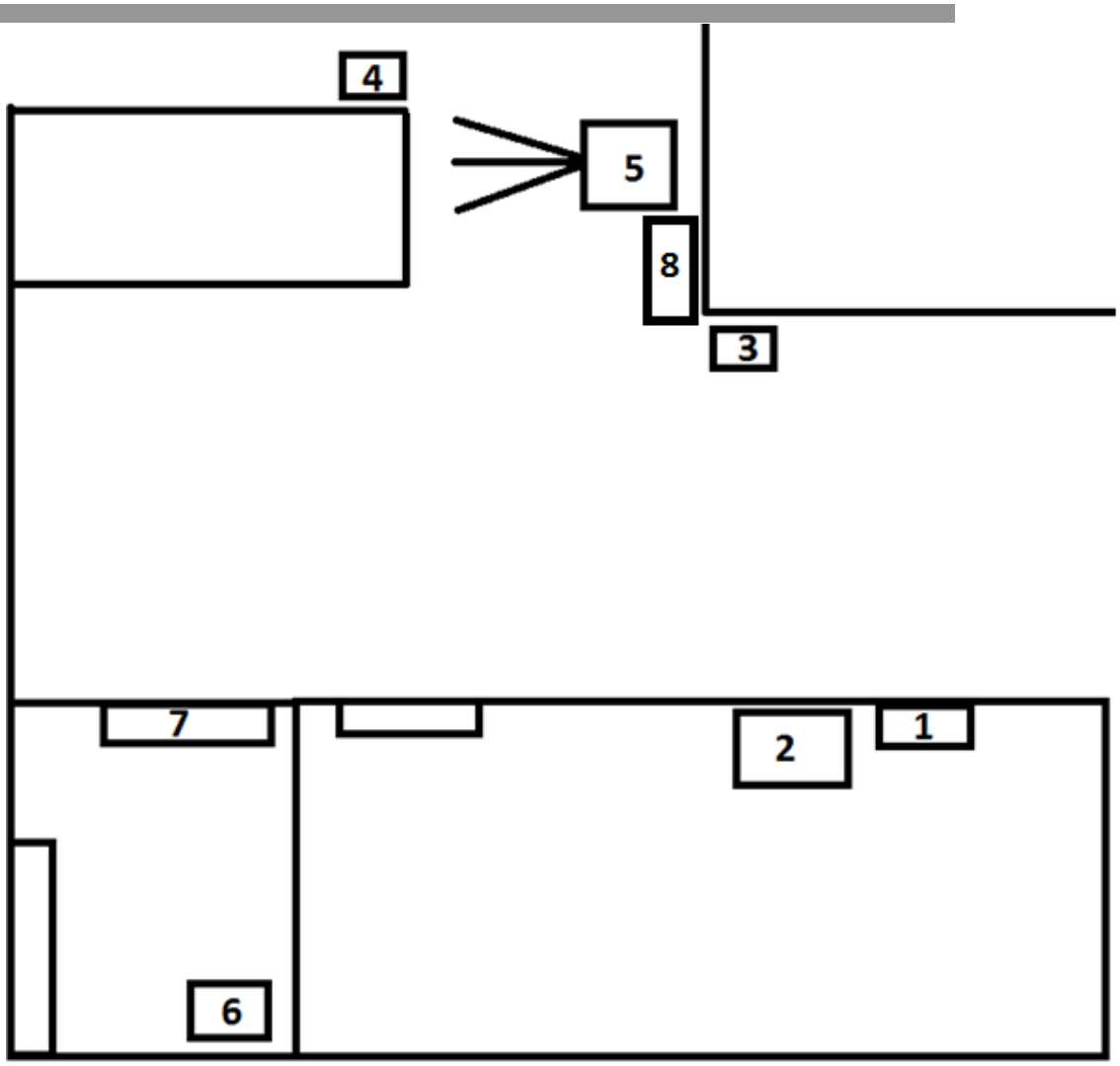
3 KULUNVALVONTALAITE

3.1 Suunnittelu

3.1.1 Laitesijoittelu

Suunnittelu aloitettiin kartoittamalla järjestelmään kuuluvat laitteet. Kuvassa 1 on vapaamuotoinen piirustus tiloista, joissa laitteet sijaitsevat. Kuvassa esitetään laitteiden sijainti tiloissa. Järjestelmään liittyvät laitteet on numeroitu seuraavasti:

- 1. Kulunvalvontalaite
- 2. Tietokone
- 3. RFID-lukija, kääntöportti sisäänpäin
- 4. RFID-lukija, kääntöportti ulospäin
- 5. Kääntöportti
- 6. RFID-lukija, pääovi
- 7. Pääovi
- 8. Paikallaolonäyttö



Kuva 1. Pohjapiirustus ja laitesijoittelu

3.1.2 Komponenttien valinta

Kun järjestelmän laitteet oli kartoitettu määritettiin tarvittavat komponentit. Lukijoita oli kolme, joten digitaalisia tuloja tarvittiin kuusi. Ohjattavia laitteita oli kolme, jotka tarvitsivat potentiaalivapaat sulkeutuvat koskettimet. Lisäksi laite oli oltava helposti yhdistettävissä tietokoneeseen kommunikointia varten.

Lukijat vaativat 12V DC käyttöjännitteen ja kuluttavat tyypillisesti 100mA virtaa. Tämän lisäksi virtaa kuluttavat releet, joilla ohjataan järjestelmän ovea ja porttia. Releitä on kolme ja tyypillinen pienoisreleen tarvitsema virta on 100mA-200mA. Virtalähteeksi valittiin kahden ampeerin hakkurivirtalähde, jonka lähtöpuolelle liitettiin 5x20mm lasiputkisulake. Syöttöpuolta ei suojattu, koska kyseessä on pistotulpalla liitettävä laite, jonka suojauksesta huolehtii syötön pistorasia sulake. Koska laitteessa ei vaihtosähköä muualla kuin syötössä, ei katsottu tarpeelliseksi suojata laitetta erotusmuuntajalla.

Ohjauslaitteeksi valikoitiin Arduino Nano, jonka nopeuden arvioitiin riittävän kolmen Wiegand-protokollaa noudattavan lukijoiden signaalien käsittelyyn. Laitteessa oli riittävästi tuloja ja lähtöjä, joista suurin osa jäi vapaaksi ja niitä voidaan käyttää tulevaisuudessa järjestelmän laajennuksissa. Laite sisältää myös USB-portin, jonka kautta tietokoneen kanssa kommunikointi kävi helposti. Kyseiselle mikrokontrollerille oli saatavilla ruuviliitinlaajennus, minkä ansiosta laitteelle ei tarvinnut valmistaa piirilevyjä ja liitännöistä saatiin luotettavia. Mikrokontrolleri sai käyttösähkönsä USB-portin kautta.

Ohjattavat laitteet käsittivät omat käyttöjännitteensä, joten niitä tuli ohjata potentiaalivapailla koskettimilla. Jokaista laitetta varten tarvittiin rele, jossa olisi vähintään yksi vaihtokosketin. Arduino Nano perustuu ATMega328P-mikrokontrolleriin, jonka lähtöjen suurimmaksi virtakestoisuudeksi on määritelty 40mA (Atmel, 2016). Tämä ei riitä releiden ohjaukseen ilman vahvistusta. Releitä varten olisi voitu rakentaa vahvistus optoeristimillä tai transistoreilla, mutta elektroniikan rakentamista haluttiin välttää ja käyttää helposti saatavilla olevia osia. Valittiin valmiit releyksiköt, jotka vahvistavat mikrokontrollerin antamat ohjaussignaalit optoeristimien avulla releille riittäväksi. Releyksiköiden ohjaus tapahtui mikrokontrollerin lähdöillä, mutta ne ottavat releen tarvitseman sähkön 12V virtalähteeltä, jotta USB-portin virtasyöttö ei rasittuisi.

3.1.3 Käytetyt osat

Käytetyistä osista tehtiin kojeluettelo. Kojeluetteloon ei listattu koteloa, RFID-lukijoita, pohjalevyä, kaapeleita, väliliittimiä, tai tiivisteitä. Nämä tuotteet ovat hyvin yleisluonteisia ja olivat osittain valmiina asennettuna. Taulukossa 2 esitetään kyseinen kojeluettelo. Tuotteiden linkit olivat saatavilla 15.4.2016.

Taulukko 2. Kulunvalvontalaitteen kojeluettelo

Tunnus	Laite	Linkki
CPU	Nano 3.0 mikrokontrolleri	http://www.aliexpress.com/item/1PCS-Nano-3-0-controller-compatible-with-nano-CH340-USB-driver-NO-CABLE-NANO-V3-0/2035011839.html
CPU	Nano ruuviliitinlaajennus	http://www.aliexpress.com/item/Free-shipping-NANO-3-0-controller-Terminal-Adapter-for-NANO-terminal-expansion-board-for-arduino-Nano/1920750076.html
T1	12V DC Virtalähde	http://www.aliexpress.com/item/Driver-4-LED-Strip-DC-12V-2A-24W-Switching-Power-Supply-Light-Display-AC-Top-Sale/32578634909.html
V1-V3/K1-K3	Relekortti 5V / 12V	http://www.aliexpress.com/item/2pcs-lot-12V-1-Channel-with-Optocoupler-H-L-Level-Triger-Relay-Module-red-board-for/1453767519.html
F1	Sulakekotelo 5x20mm	http://www.aliexpress.com/item/20pcs-lot-Panel-Mount-5x20mm-5-20-MM-Fuse-Holders-10A-250V-free-shipping/1209417365.html

Laite koteloitiin 12 moduulin laitekoteloon. Kotelon pohjalle asennettiin alumiinilevy, johon kojeet kiinnitettiin. Väliliittimiksi asennettiin 9-napaiset D-liittimet. Kaapeleiksi asennettiin LiyCy 4x0,5 kaapelia. Vanha asennus oli toteutettu CAT5 UTP-kaapeleilla, jotka korvattiin LiyCy:llä, jossa on kuparista punottu kaapelivaippa, jota voidaan käyttää häiriönsuodattamiseen.

3.1.4 Kytkentä

Liitteessä 1 on esitetty laitteen piirikaavio. Sähkönsyöttö jaettiin kahteen osaan. Tietokoneen USB-portti syöttää Arduinon tarvitseman 5V DC jännitteen. Tätä jännitettä käytetään myös releyksiköiden ohjausjännitteenä. Tämän lisäksi lisättiin erillinen 12V DC virtalähde, jolla syötetään sähkö releille ja RFID-lukijoille. 12V DC syöttö on suojattiin 1A sulakkeella, joka suojaa piiriä oikosulkujen varalta. 5V DC syöttöä ei suojattu, koska se liitetään valmiilla USB-kaapelilla suoraan tietokoneen ja Arduinon välille. Lisäksi kyseistä syöttöä käytetään laitteessa niin vähäisesti, ettei sen suojausta nähty tarpeelliseksi.

Tasajännitteiden nollapotentiaalit yhdistettiin, jotta jännitteet olisivat samassa potentiaalissa ja toimisivat yhdessä oikein. Lukijoille lähtevien kaapelien suojavaipat liitettiin maadoitusliittimiin, häiriöiden minimoimiseksi. 12V DC virtalähteen syöttö tuotiin laitteeseen pistotulppakaapelilla. Tälle kaapelille ei asennettu omaa sulaketta, koska katsottiin, että pistotulppalaitteessa pistorasian sulake tai johdonsuojakatkaisija on riittävä suoja. Oven ja porttien ohjauksessa käytettyjen kaapelien suojausta ei myöskään tehty, koska katsottiin, että suojaus tehdään ohjattavan laitteen puolella. Tämä johtui siitä, että kyseessä on kulunvalvontalaitteen kannalta ulkoinen jännite. Samasta syystä näiden kaapeleiden mahdollisia suojavaippoja ei maadoitettu kulunvalvontalaitteen puolella, jotta vältyttäisiin mahdollisilta maasilmukoilta, joiden mukana mahdolliset häiriöt leviäisivät kaikkiin laitteeseen maadoitettuihin kaapeleihin.

Valitun Arduino Nano-yhteensopivan piirin mikrokontrollerina toimii Atmega328. Tulot ja lähdöt valittiin niin, että tulot osuvat yhteen Atmegan porteista ja lähdöt toiseen porttiin. Tulojen nastat A0-A5 vastaavat Atmegassa portin C nastoja 0-5. Lähtöjen nastat D10-D12 vastaavat atmegassa portin B nastoja 2-4. (Arduino, 2008.)

3.2 Yleistä Arduinosta

Arduino on avoimeen lähdekoodiin perustuva laitteisto ja ohjelmisto. Alun perin Arduino tarkoitettiin opiskelijoiden käyttöön, prototyyppien kehityksen nopeuttamiseksi ja helpottamiseksi. Arduinon saavuttaessa suuremman yleisön, alettiin sitä soveltaa nopeasti monissa eri käyttötarkoituksessa esimerkiksi puettavissa älylaitteissa, 3d-tulostuksessa, sekä IoT-laitteissa. (Arduino, n.d.)

Arduinon piirilevyt ja ohjelmistot ovat avointa lähdekoodia, joten yhteensopivia laitteita voi valmistaa kuka tahansa. Arduino on alkuperäisen valmistajan Yhdysvalloissa käyttämä tuotenimi. Muualla maailmassa alkuperäisten valmistajien piirit myydään nimellä Genuino. Tämän lisäksi on saatavilla useita yhteensopivia piirejä, joita yleensä myydään Arduino yhteensopivina. (Arduino, n.d.)

Arduino Nano V3 perustuu ATmega328-mikrokontrolleriin, johon on ohjelmoitu arduinon oma käynnistysohjelma, joka mahdollistaa Arduinon komentojen käytön ohjelmoinnissa, joilla helpotetaan esimerkiksi tulojen

ja lähtöjen käyttöä. Tuloja ja lähtöjä voi ohjata myös kuten, niitä ohjattaisiin käytettäessä ATmega328:n ohjelmointiympäristöä, mikä on ohjelman suorituksen kannalta huomattavasti nopeampaa.

3.3 Ohjelma

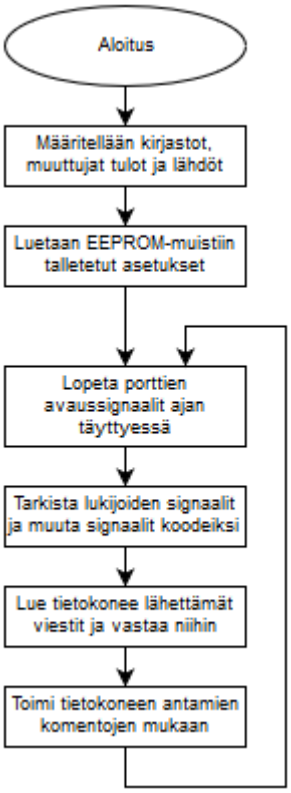
Ohjelma jaettiin karkeasti muutamaan osaan. Ensimmäisenä määriteltiin vakioarvot, joita ohjelmassa käytetään. Nämä vakiot määriteltiin globaaleiksi muuttujiksi ohjelmaan. Tämän jälkeen määriteltiin ohjelman aloitus, joka määrittää Arduinon tulot ja lähdöt oikeaan tilaan. Sitten määriteltiin jatkuvasti suoritettavan ohjelman tarvitsemat muuttujat. Tämän jälkeen ohjelma lukee ohjaussignaalien pituudet EEPROM-muistista. Tämän jälkeen ohjelma jatkuvan silmukan, jossa itse toiminta tapahtuu. Silmukan toiminnot jaettiin useaan osaan, ensimmäisenä tarkistus onko portin tai oven ohjaussignaaleilla ohjausaika täynnä ja niiden sammuttaminen, mikäli aika on täyttynyt. Tämän jälkeen tarkistellaan jokainen RFID-lukija erikseen, onko niiltä tullut uutta tietoa ja onko luenta valmis. Mikäli luenta on valmis, käännetään lukijoiden lähettämä data biteistä tavuiksi. Viimeisenä tarkistetaan onko tietokone lähettänyt oikeanlaisen viestin Arduinolle sarjaliikenteenä. Mikäli on, tietokoneelle lähetetään viesti, joka kertoo mikä on tilanne Arduinolla. Liikennöinti tarkastetaan ylivuotojen ja viiveiden varalta.

Arduinon kannalta ohjelma on jaettu kolmeen osaan. Ensimmäinen osa on ohjelman funktioiden ulkopuolella ja siinä määritellään ohjelman käyttämät kirjastot, sekä globaalit muuttujat.

Toinen osa on alustusosa, jota kutsutaan nimellä setup. Tämä ohjelman osa on funktio, joka suoritetaan yhden kerran, kun arduino käynnistetään. Tässä osassa yleensä asetetaan piirin tulot ja lähdöt ohjelman vaatimiin tiloihin.
Kolmas osa on pääohjelma, jota kutsutaan nimellä main. Tämä ohjelman osa on funktio, jota suoritetaan silmukkana, kunnes piiriltä katkaistaan sähkö.

Funktioita voidaan kirjoittaa vapaasti enemmänkin ja usein ohjelman jakaminen funktioiden avulla pienempiin toimintoihin on järkevää. Tässä tapauksessa ohjelma päätettiin jättää jakamatta funktioihin, vaikka se olisikin helposti jaettavissa oleva ohjelma ja jakaminen tekisi ohjelmasta huomattavasti selkeämmän lukea. Toteutustapa valittiin, koska ohjelmasta haluttiin mahdollisimman muistitehokas ja nopea. Jokainen funktiokutsu tarkoittaa hyppykäskyjä ja muistinkäsittelyä prosessorille, kun ohjelma kirjoitettiin yhteen funktioon, voitiin säästää hieman prosessorin resursseja ohjelman suoritukselta todella lyhyiden Wiegandin pulssien lukemiseen. Ohjelmassa keskityttiin muutenkin mahdollisimman nopeasti toimivaan koodiin, määrittelemällä muuttujien datatyypit tarkasti tarpeen mukaan ja käyttämällä atmegan ohjaustapaa tulojen ja lähtöjen ohjaukseen, arduinon lähtö- ja tulofunktioiden sijasta.

Ohjelma kommentoitiin englannin kielellä, jotta sen julkaisu avoimena lähdekoodina tulevaisuudessa olisi helpompaa. Kuviossa 1 on kuvattu ohjelman toimintaperiaate vuokaaviona.



Kuvio 1. Kulunvalvontalaitteen ohjelman rakenne

3.3.1 Ohjelman vakiomuuttujat ja kirjastot

```
#include <EEPROM.h>

// wiegand constants
#define codebitstotal 26
#define wiegandtimeout 200

// serial constants
#define bytestosend 7
#define bytestorecieve 7
#define serialtimeout 800
```

Yllä on esitetty vakiomuuttujien ja käytettyjen kirjastotiedostojen määrittely. Nämä määrittelyt tehdään ohjelmakoodin alussa, kaikkien funktioiden ulkopuolella.

Aluksi ohjelmaan liitettiin EEPROM.h kirjasto, josta löytyvät funktiot EEPROM-muistin käsittelyyn. EEPROM-muisti on muistia joka säilyttää arvonsa myös kun piiri on pois päältä ja tässä sovelluksessa sinne talletettiin portin ja oven aukaisupulssien pituudet.

RFID-lukijoita varten määriteltiin wiegand-koodin pituus, joka oli valituilla lukijoilla 26 bittiä. Määriteltiin myös maksimiaika millisekunteina wie-

gand signaalien välillä. Tällä ajalla päätellään myös, koska koodinluku on valmis.

Kolmantena määriteltiin arvot tietokoneen ja arduinon väliseen sarjaliikenteeseen. Näihin sisältyivät lähetettävien tavujen määrä, vastaanotettavien tavujen määrä sekä maksimiaika tietokoneen lähettämien viestien välillä, ennen kuin yhteyttä pidetään katkenneena.

3.3.2 Tulojen ja lähtöjen alustus

```
void setup()
{
  // Pin setup. To make the program run as fast as
  // possible, rest of the program uses PIN and PORT.
  pinMode(A0, INPUT); // reader 1 D0
  pinMode(A1, INPUT); // reader 1 D1
  pinMode(A2, INPUT); // reader 2 D0
  pinMode(A3, INPUT); // reader 2 D1
  pinMode(A4, INPUT); // reader 3 D0
  pinMode(A5, INPUT); // reader 3 D1
  digitalWrite(A0, HIGH);
  digitalWrite(A1, HIGH);
  digitalWrite(A2, HIGH);
  digitalWrite(A3, HIGH);
  digitalWrite(A4, HIGH);
  digitalWrite(A5, HIGH);

  pinMode(10, OUTPUT);  // relay for door 1
  pinMode(11, OUTPUT);  // relay for door 2
  pinMode(12, OUTPUT);  // relay for door 3
  pinMode(13, OUTPUT);
}
```

Yllä on esitetty tulojen ja lähtöjen alustus oikeaan tilaan. Tämä tehdään setup() funktiossa, joka suoritetaan kerran aina piirin käynnistyttyä. Ensin määritellään piirin nasta A0-A5 tuloiksi ja asetetaan piirin sisäiset ylösvetovastukset näille tuloille käyttöön. Näihin tuloihin luetaan Wiegand-bitit RFID-lukijoilta.

Tämän jälkeen asetetaan piirin nasta 10-13 lähdöiksi. Nastaa 13 ei varsinaisesti käytetä ohjelmassa. Nastaan 13 on kytketty arduinoon rakennettu led-merkkivalo, jota käytettiin diagnostiikkaan ohjelman testauksen aikana.

3.3.3 Pääohjelman muuttujat ja alustukset

```
void loop()
{
  // variable definitions

  // bit is read from a reader
  byte bitread[3];
  // Serial bytes
  byte rx[bytestorecieve], tx[bytestosend];
  // reader has finished reading the code
  byte readerdone[3]={0,0,0};
  // bytes for storaging codes
```

```
  byte codebytes[3][4];
  // bits got from readers
  byte bit[3][codebitstotal];
  // timeout for reading wiegand code
  unsigned long last_pulse_millis[3];
  // counter, when to flush serial
  unsigned long serialflush;
  // timer for code ok, pulses for doors/etc
  unsigned long gateopen[3];
  // pulse length as ms for doors/etc
  unsigned int gateopentime[3];
  // Datalines from readers
  byte D1;
  // Counter for bits from the reader
  byte counter[3]={0,0,0};
  // which reader code is sent to pc
  byte readerinprogress=0;

  gateopentime[0] = EEPROM.read(253);
  gateopentime[1] = EEPROM.read(254);
  gateopentime[2] = EEPROM.read(255);

  Serial.begin(9600);
```

Ennen varsinaisen ohjelman suorittamista tehdään vielä viimeiset muistien määrittelyt ja tilojen alustukset. Muuttujaan bitread käytetään tunnistamaan onko lukijalta tuleva bitti luettu jo aikaisemmin, vain onko se uusi. Muuttujat rx ja tx ovat sarjaliikennöintiin tietokoneen kanssa varatut tavut. Muuttuja readerdone on tieto millä lukijoilla on koodinluku saatu valmiiksi, mutta koodia ei ole vielä käsitelty. Muuttuja codebytes sisältää RFID-lukijoilta luetut koodit tavuiksi muunnettuna. Muuttuja bit sisältää RFID-lukijoilta luetut koodit bitteinä. Muuttuja last_pulse_millis sisältää millisekunteina ajan, jolloin viimeinen bitti on luettu RFID-lukijalta, sitä käytetään määrittelemään, milloin koodinluku on valmis. Muuttuja serialflush sisältää tiedon, milloin tietokoneelta on viimeksi saatu luettua viesti, sitä käytetään määrittelemään, koska yhteyttä tietokoneeseen pidetään katkenneena. Muuttujaan gateopen talletetaan tieto, milloin ovi tai portti on avattu, sitä käytetään aukaisusignaalin lopettamiseen. Muuttujaan readerinprogress talletetaan tieto, mitä lukijaa arduino ja pc käsittelevät, tarvitaan mikäli useampaa lukijaa luetaan samaan aikaan ja luvut pitää käsitellä jonossa.

Määrittelyjen jälkeen portin ja oven ohjaussignaalien pituus luetaan EEPROM-muistista. Viimeisenä käynnistetään sarjaliikenne, minkä jälkeen ohjelma on valmis aloittamaan pääsilmukkansa.

3.3.4 Ovien ja porttien ohjaussignaalin lopetus

```
//Start the actual loop
  while (true)
  {
    // check if a some of the gate pulses has been on for
enough time

    if (millis() - gateopen[0] > gateopentime[0]) PORTB =
PORTB & B11111011;
```

```
    if (millis() - gateopen[1] > gateopentime[1]) PORTB =
PORTB & B11110111;
    if (millis() - gateopen[2] > gateopentime[2]) PORTB =
PORTB & B11101111;
```

Yllä on esitetty lähtöjen signaalien lopetus. Millis() on Arduinon sisäinen laskuri, joka laskee millisekunteja jatkuvasti piirin ollessa päällä. Muuttujassa gateopen on talletettuna ajat jolloin lähdöt on asetettu päälle. Kun tämä aika vähennetään millisekuntilaskurista, saadaan tietää kuinka kauan lähtö on ollut päällä. On hyvin tärkeää, että tämä aika lasketaan vähentäen, eikä lasketa lopettamisaikaa, lisäämällä lähdön päälleasettamisaikaan päälläoloaikaa. Millis() laskuri on 32-bittinen luku, joka tulee täyteen ja nollautuu noin 50 päivän välein. Mikäli vertailu tehtäisiin lisäämällä, tarvittaisiin ylivuotoon huomattavasti enemmän valvontaa. Vähennyslaskulla ylivuodosta huolimatta kulunut aika on aina oikea arvo. Saatua arvoa verrataan lähdöille määritettyyn päälläoloaikaan ja mikäli arvo on suurempi, asetetaan lähtö pois päältä.

Lähdön ohjaukseen käytetään Atmegan määrittelyn mukaista porttia ja ja-operaattoria, yksilöimään lähtö kahdeksan lähdön portista. Tämä on ohjelman kannalta huomattavasti nopeampi tapa, kuin käyttää arduinon lähdönohjauksiin tarkoitettuja funktioita.

3.3.5 Wiegand-koodin lukeminen

```
// data from reader 1
    D1 = PINC & B00000011;
    if (D1 == B00000011)
    {
      bitread[0] = LOW;
      // pause between bit's on the code or code done
      if (counter[0])
      {
        if (millis() - last_pulse_millis[0] > wiegandti
        meout)
        // timeout on reader 1, check if there's enough
        bits
        {
          if (counter[0] == codebitstotal)
// 26 bits counted reader 1. Calculate the code into bytes.
          {
```

HUOM! Tästä välistä poistettu koodin muuttaminen tavuiksi

```
           readerdone[0] = HIGH;
          }
          counter[0] = 0;
         // Reset the counter, since there's been a timeout
        }
      }
    }
    else if (D1 == B00000001 && !bitread[0])
    {
      last_pulse_millis[0] = millis();
      if (counter[0] < codebitstotal) bit[0][counter[0]] =
      0;
      counter[0]++;
```

```
    bitread[0] = HIGH;
  }
  else if (D1 == B00000010 && !bitread[0])
  {
    last_pulse_millis[0] = millis();
    if (counter[0] < codebitstotal) bit[0][counter[0]] =
    1;
    counter[0]++;
    bitread[0] = HIGH;
  }
```

Yllä on esitetty Wiegand pulssien käsittely ohjelmassa. Lähdekoodista on poistettu datan muuttaminen biteistä tavuiksi, mikä käsitellään seuraavassa luvussa. Ensin luetaan portista C lukijan yksi bitit muuttujaan D1. On otettava huomioon, että bittien tila on käänteinen, eli bitti on pois päältä kun signaali on päällä. Mikäli molemmat bitit ovat päällä, ei sillä hetkellä ole signaalia päällä ja muuttuja bitread asetetaan pois päältä. Tämän jälkeen tarkastetaan onko koodiin laskettu bittejä. Mikäli bittejä on, tarkastetaan kauanko edellisestä bitistä on aikaa ja ylittääkö se Wiegandille määritellyn maksimiajan. Ajan vertailu toimii samalla periaatteella, kuin ajan laskenta luvussa 3.4.4. Mikäli aika on ylittynyt tarkistetaan onko bittejä saatu luettua määritelty määrä. Mikäli bittejä on tarvittava määrä, muutetaan tieto biteistä tavuiksi ja asetetaan readerdone-muttujaan tieto, että koodi on luettu. Tämän jälkeen luettujen bittien laskuri nollataan. Nollaus tehdään, oli bittejä luettu riittävä määrä tai ei. Mikäli bittejä on liian vähän, luenta ohitetaan virheellisenä. Virhettä ei kirjata ylös tai lähetetä tietokoneelle, vaan oletetaan, että virheellisen luennan sattuessa, käyttäjä yrittää lukea tunnisteensa uudelleen.

Mikäli toinen lukijan biteistä on päällä ja muuttujassa bitread ei ole arvoa, päätellään, että lukija on lähettänyt uuden bitin. Tällöin asetetaan last_pulse_millis-muuttujaan millis() laskurin sen hetkinen arvo. Tämän jälkeen, mikäli bittejä on luettu vähemmän kuin koodin maksimimitta, otetaan talteen kumpi bitti oli kyseessä. Lisätään bittilaskurin arvoa ja asetetaan bitread-muuttujaan arvo, jottei samaa bittiä lueta kahteen kertaan. Mikäli sattuisi virhetilanne, missä molempien bittien tieto olisi samaan aikaan päällä, ensin vaikuttunut otettaisiin huomioon. Tämä johtaisi hyvin suurella todennäköisyydellä virheelliseen koodin lukuun.

Tämä koodi on ohjelmassa kolmeen kertaan, kerran joka lukijaa kohden. Osassa muuttujia käytetään osoittajia kertomaan, mistä bitistä, tavusta ja lukijasta on kyse.

3.3.6 Wiegand-koodin muuttaminen biteistä tavuiksi

```
if (bit[0][0]) codebytes[0][0] = 1; else codebytes[0][0] =
0;
if (bit[0][1]) codebytes[0][0] = codebytes[0][0] + 2;
if (bit[0][2]) codebytes[0][0] = codebytes[0][0] + 4;
if (bit[0][3]) codebytes[0][0] = codebytes[0][0] + 8;
if (bit[0][4]) codebytes[0][0] = codebytes[0][0] + 16;
if (bit[0][5]) codebytes[0][0] = codebytes[0][0] + 32;
if (bit[0][6]) codebytes[0][0] = codebytes[0][0] + 64;
if (bit[0][7]) codebytes[0][0] = codebytes[0][0] + 128;
```

```
if (bit[0][8]) codebytes[0][1] = 1; else codebytes[0][1] =
0;
if (bit[0][9]) codebytes[0][1] = codebytes[0][1] + 2;
if (bit[0][10]) codebytes[0][1] = codebytes[0][1] + 4;
if (bit[0][11]) codebytes[0][1] = codebytes[0][1] + 8;
if (bit[0][12]) codebytes[0][1] = codebytes[0][1] + 16;
if (bit[0][13]) codebytes[0][1] = codebytes[0][1] + 32;
if (bit[0][14]) codebytes[0][1] = codebytes[0][1] + 64;
if (bit[0][15]) codebytes[0][1] = codebytes[0][1] + 128;

if (bit[0][16]) codebytes[0][2] = 1; else codebytes[0][2] =
0;
if (bit[0][17]) codebytes[0][2] = codebytes[0][2] + 2;
if (bit[0][18]) codebytes[0][2] = codebytes[0][2] + 4;
if (bit[0][19]) codebytes[0][2] = codebytes[0][2] + 8;
if (bit[0][20]) codebytes[0][2] = codebytes[0][2] + 16;
if (bit[0][21]) codebytes[0][2] = codebytes[0][2] + 32;
if (bit[0][22]) codebytes[0][2] = codebytes[0][2] + 64;
if (bit[0][23]) codebytes[0][2] = codebytes[0][2] + 128;

if (bit[0][24]) codebytes[0][3] = 1; else codebytes[0][3] =
0;
if (bit[0][25]) codebytes[0][3] = codebytes[0][3] + 2;
```

Yllä on esitetty ohjelmakoodi muuntaa Wiegand-koodista kerätyt bitit tavuiksi. Bitit luetaan kahdeksan ryhmissä ja jaetaan neljään tavuun. Ensimmäinen bitti kahdeksan bitin ryhmästä nollaa tai asettaa tavun arvoksi yksi. Tämän jälkeen käydään läpi seitsemän seuraavaa bittiä, lisäten tavuun numero kaksi korotettuna bitin järjestysluvun mukaiseen potenssiin, mikäli bitti on tosi. Järjestysluvut lähtevät nollasta, joten ensimmäinen lisäyslasku voi alkaa luvusta kaksi. Tämä toteutetaan kolmen ensimmäisen tavun kohdalla, mutta koska bittien lukumäärä on 26, jää viimeiselle tavulle vain kaksi bittiä. Ensimmäinen ja toinen bitti tarkistellaan, kuten muissakin tavuissa, joten kun ensimmäiset kolme tavua voivat olla arvoltaan 0-255, voi viimeisen tavun arvo olla vain väliltä 0-3.

3.4 Kommunikointi

3.4.1 Tiedonsiirtotapa ja viestien rakenne

Kulunvalvontalaite kommunikoi tietokoneen kanssa sarjaliikennemuotoisena. Laite lähettää tietokoneelle yhdeksän tavua ja tietokone vastaa seitsemällä tavulla. Tavut on numeroitu nollasta kuuteen, koska sekä tietokoneen, että kulunvalvontalaitteen ohjelmistot käsittelevät luettua jonoa nollasta alkavalla luvulla. Laitteen tietokoneelle lähettämiä kahta ensimmäistä tavua ei huomioida numeroinnissa, koska ne ovat aina samanarvoiset. Taulukoissa 3 ja 4 esitetään tiedonsiirrot kulunvalvontalaitteelta tietokoneelle ja toisinpäin.

Tavut numerosta nolla numeroon neljä on varattu perustiedon lähettämiseen. Kulunvalvontalaite lähettää näissä tavuissa luetun koodin tietokoneelle. Tietokone ei käytä kolmea ensimmäistä tavua mihinkään (tavut 0-2). Nämä tavut varattiin tiedonsiirtoon, koska haluttiin jättää mahdollisuus kirjoittaa koodeja myös kulunvalvontalaitteelle, mahdollista itsenäistä toimintaa varten. Kulunvalvontalaitteen itsenäistä toimintaa ei kuitenkaan

muistin vähyyden ja reaaliaikaisen kellon puutteen johdosta toteutettu. Tietokone käyttää myös tavua numero kolme lukijalaitteen numeron lähetykseen avauspulssien pituutta kirjoitettaessa, sekä muistin sisältönä kirjoitettaessa EEPROM-muistiin.

Tavu numero neljä on varattu komentoihin ja kuittauksiin. Kulunvalvontalaite kertoo tietokoneelle uudesta tapahtumasta tai miten se on reagoinut tietokoneen lähettämiin komentoihin. Tietokone lähettää laitteelle komennon tai kuittaa laitteen lähettämän tiedon luetuksi.

Tavu numero viisi on varattu komentoon liittyvälle lisätiedolle. Laite kertoo tietokoneelle miltä lukijalta koodi on luettu, tai luetun EEPROM-muistin arvon. Tietokone kertoo kulunvalvontalaitteelle kirjoitettavan avauspulssin ajan tai kirjoitettavan tai luettavan EEPROM-muistin osoitteen.

Tavu numero kuusi on varattu synkronointiin ja kaiutukseen. Tietokone lähettää kulunvalvontalaitteelle aina luvun 210. Mikäli seitsemäs luettu tavu ei ole 210, kulunvalvontalaite tietää, että tiedonsiirron synkronointi ei ole kohdallaan ja yrittää tahdistaa liikenteen uudelleen. Laite kaiuttaa lukemansa tavun takaisin tietokoneelle tavussa numero kuusi ja tietokone ei hyväksy viestiä, mikäli kaiutettu tavu eroaa lähetetystä. Tämän lisäksi kulunvalvontalaite aloittaa jokaisen viestinsä tietokoneelle kahdella tavulla jotka eivät näy taulukossa. Tavujen arvot ovat 80 ja 67, joista muodostuu ASCII-koodauksella kirjaimet PC, joka on lyhenne sanoista Personal Computer eli henkilökohtainen tietokone. ASCII-taulukko on esitetty liitteessä 2. Tietokone lukee vastaanottamaansa tavujonoa kunnes löytää kyseiset tavut ja aloittaa viestin luvun siitä kohtaa. Kun yhdistetään kirjaimet PC ja takaisin kaiutettu tavu, saadaan viestien synkronointi pysymään kohdallaan.

Taulukko 3. Tiedonsiirto kulunvalvontalaitteelta tietokoneelle

Tavu	0	1	2	3	4	5	6
Tieto	Koodi1	Koodi2	Koodi3	Koodi4	Komento	Lisätieto	kaiutus
					0=ei tapahtumia		kaiutus
					1=uusi koodi luettu	lukijalaitteen numero	kaiutus
					2=koodi kirjoitettu		kaiutus
					3=koodi poistettu		kaiutus
					4=ovi avattu		kaiutus
					5=portti sisään avattu		kaiutus
					6=portti ulos avattu		kaiutus
					7=porttipulssi kirjoitettu		kaiutus
					8=muisti luettu	arvo	kaiutus
					9=muisti kirjoitettu		kaiutus
					12=pääovi avattu		kaiutus
					13=portti sisäänpäin avattu		kaiutus
					14=portti ulospäin avattu		kaiutus
					15=pääovi suljettu		kaiutus
					16=portti sisäänpäin suljettu		kaiutus
					17=portti ulospäin suljettu		kaiutus

Taulukko 4. Tiedonsiirto tietokoneelta kulunvalvontalaitteelle

Tavu	0	1	2	3	4	5	6
Tieto	Varalla	Varalla	Varalla	Arvo	Komento	Lisätieto	Synkronointi
				lukijalaite	3=määritä aukipulssin aika	pulssin pituus	210
					4=lue EEPROM muisti	muistin numero	
					5=avaa ovi		
					6=avaa portti sisään		
					7=avaa portti ulos		
				arvo	8=kirjoita EEPROM muisti	muistin numero	
					10=kuittaa tehtävät		
					12=avaa pääovi		
					13=avaa portti sisäänpäin		
					14=avaa portti ulospäin		

3.4.2 Tiedon vastaanotto ja koodin lähetys tietokoneelle

```
// serial communication.
if (Serial.available() == bytestorecieve)
    {
      // 7 bytes recieved, that's the right amount from pc.
      Serial.readBytes(rx, bytestorecieve);
      if (!rx[4] && !readerinprogress)
      {
        if (readerdone[0])
        {
          tx[4] = 1; // Tell PC that the code read is new
          tx[5] = 1; // Tell PC which reader read the code.
          // let's send last read code to the pc.
          tx[0] = codebytes[0][0];
          tx[1] = codebytes[0][1];
          tx[2] = codebytes[0][2];
          tx[3] = codebytes[0][3];
          readerinprogress = 1;
        }
        else if (readerdone[1])
        {
          tx[4] = 1; // Tell PC that the code read is new
          tx[5] = 2// Tell PC which reader read the code.
          // let's send last read code to the pc.
          tx[0] = codebytes[1][0];
          tx[1] = codebytes[1][1];
          tx[2] = codebytes[1][2];
          tx[3] = codebytes[1][3];
          readerinprogress = 2;
        }
        else if (readerdone[2])
        {
          tx[4] = 1; // Tell PC that the code read is new
          tx[5] = 3; // Tell PC which reader read the code.
          // let's send last read code to the pc.
          tx[0] = codebytes[2][0];
          tx[1] = codebytes[2][1];
          tx[2] = codebytes[2][2];
          tx[3] = codebytes[2][3];
          readerinprogress = 3;
        }
      }
```

Jokaisella ohjelmakierroksella laite tarkastaa, onko se saanut vastaanotettua seitsemän tavua tietokoneelta. Mikäli seitsemän tavua on vastaanotettu, luetaan tavut muuttujaan rx ja tarkastetaan onko neljännessä vastaanotetussa tavussa muu arvo kuin nolla, jolloin tietokone lähettää komennon kulunvalvontalaitteelle, tai onko jonkin lukijan lukema koodi jo käsittelyssä. Mikäli komentoa sekä käsiteltävää lukijan koodi ei ole ja jossakin lukijassa on saatu luettua koodi, asetetaan lähetettävään viestiin koodi ja asetetaan käsiteltävän lukijan numero muuttujaan readerinprogress, mikä asettaa kyseisen lukijan koodin käsittelyyn. Mikäli käsittelyn aikana luetaan koodeja useammasta lukijasta, käsitellään luetut koodit jonona. Jonoa ei käsitellä lukujärjestyksessä vaan lukijan numeron järjestyksessä pienimmästä suurimpaan. Käsittely on huomattavan nopeaa ja käyttäjä ei huomaa eroa vaikka hänen lukemansa koodi jäisi jonon viimeiseksi.

3.4.3 Tietokoneen antaman komennon toteutus ja viestiin vastaaminen

```
// mirror last read byte back to pc. It should always be
// 210. If it's not, then there's a sync or com problem
// going on and the pc will flush all it's communication
// and try again.
tx[6] = rx[6];
if (rx[6] == 210)
 {
   // execute a command from pc
   if (rx[4] == 8) // command 8, write EEPROM
   {
     tx[4] = 9;
     EEPROM.write(rx[5], rx[3]);
   }
   else if (rx[4] == 4) // command 4, read EEPROM
   {
     tx[4] = 8;
     tx[5] = EEPROM.read(rx[5]);
   }
   else if (rx[4] == 5) // command 5, open reader 1 relay
   {
     PORTB = PORTB | B00000100; // reader 1 relay
     gateopen[0] = millis();
     tx[4] = 4;
     tx[5] = 0;
     readerdone[0] = 0;
   }
   else if (rx[4] == 6) // command 6, open reader 2 relay
   {
     PORTB = PORTB | B00001000; // reader 2 relay
     gateopen[1] = millis();
     tx[4] = 5;
     tx[5] = 0;
     readerdone[1] = 0;
   }
   else if (rx[4] == 7) // command 7, open reader 3 relay
   {
     PORTB = PORTB | B00010000; // reader 3 relay
     gateopen[2] = millis();
     tx[4] = 6;
     tx[5] = 0;
     readerdone[2] = 0;
   }
   else if (rx[4] == 10) // command 10, everything done
```

```
    {
      if (readerinprogress)
      {
        readerdone[readerinprogress - 1] = 0;
        readerinprogress = 0;
      }
      tx[4] = 0; // PC tells us work is done
      tx[5] = 0; // reset bytes and that's it.
    }
    // the communication must begin with ”PC” or the
    // pc won't regognize it.
    Serial.write("PC");
    Serial.write(tx, bytestosend);
    serialflush = millis();
  }
```

Ohjelma asettaa lähetettävän tavun numero kuusi arvoksi vastaanotetun tavun numero kuusi arvon. Mikäli arvo on 210, tarkistetaan onko tietokone lähettänyt tunnetun komennon ja suoritetaan komento. Komennot on esitelty taulukossa 4. Komennon suorittamiseen vastataan taulukon 3 mukaisesti. Mikäli saadaan komento avata ovi tai kääntöportti, nollataan luku valmis-tieto kyseisen paikan lukijalta. Kun saadaan tietokoneelta kuittaus, että kaikki tarvittava on tehty, nollataan käsittelyssä oleva lukija, sekä tietokoneelle lähetettävät tilatiedot. Mikäli jonossa odottaa käsittelyä uusi lukijalta luettu koodi, se käsitellään seuraavalla ohjelmakierroksella.

3.4.4 Tiedonsiirtovirheen käsittely

```
else {
        while (Serial.available()) Serial.read();
      }
    }
    else if (Serial.available() > bytestorecieve)
    // there are too many bytes in the buffer, let's flush
    // them and get on with it.
    {
      while (Serial.available()) Serial.read();
    }
    else if (millis() - serialflush > serialtimeout)
    // flush what's been read, it has been too long
    // to trust that.
    {
      while (Serial.available() > 0) Serial.read();
      // Coms have failed, the mirrored byte isn't 210
      // so the pc will know about this.
      tx[6] = 0;
      Serial.write("PC");
      Serial.write(tx, bytestosend);
      serialflush = millis();
    }
  }
}
```

Yllä olevan koodin alussa oleva else viittaa tarkistukseen oliko viimeisen tavun arvo 210. Mikäli näin on, luetaan vastaanotettujen tavujen puskuri tyhjäksi ja jäädään odottamaan seuraavaa viestiä. Samoin toimitaan, mikäli vastaanotettujen tavujen määrä on enemmän kuin odotetun viestin pituus. Mikäli viestejä ei ole vastaanotettu yli 800 millisekuntiin, mikä on

ohjelman alussa määritelty maksimiaika viestin vastaanottoon, nollataan lähetyksen tavu numero kuusi ja lähetetään viesti tietokoneelle. Tämä uudelleenkäynnistää keskeytyneen tai epäsynkroniin joutuneen tietoliikenteen. Tavun numero kuusi ansiosta tietokone tietää, että tieto voi olla vanhentunutta, eikä siihen tule luottaa.

4 WINDOWS-OHJELMA

4.1 Kehitysympäristö

Kehitysympäristönä käytettiin VB.net 2013 kehityssovellusta, joka on osa Microsoft Visual Studio ohjelmistoa. Visual Studion express ja community versiot ovat saatavilla ilmaiseksi Microsoftilta. Raportin kirjoittamisen aikaan myös 2015 versiot Visual Studiosta oli julkaistu. Vaikka ohjelmiston toteuttamiseen käytettiin professional versiota, ovat projektitiedot käytettävissä myös ilmaisissa versioissa.

Projekti aloitettiin suomenkielisenä, mutta kommentoinnissa ja muuttujien nimissä siirryttiin myöhemmin englantiin, ajatellen mahdollista julkaisua avoimena lähdekoodina. Tästä johtuen lähdekoodeissa esiintyy sekä suomen että englannin kieltä.

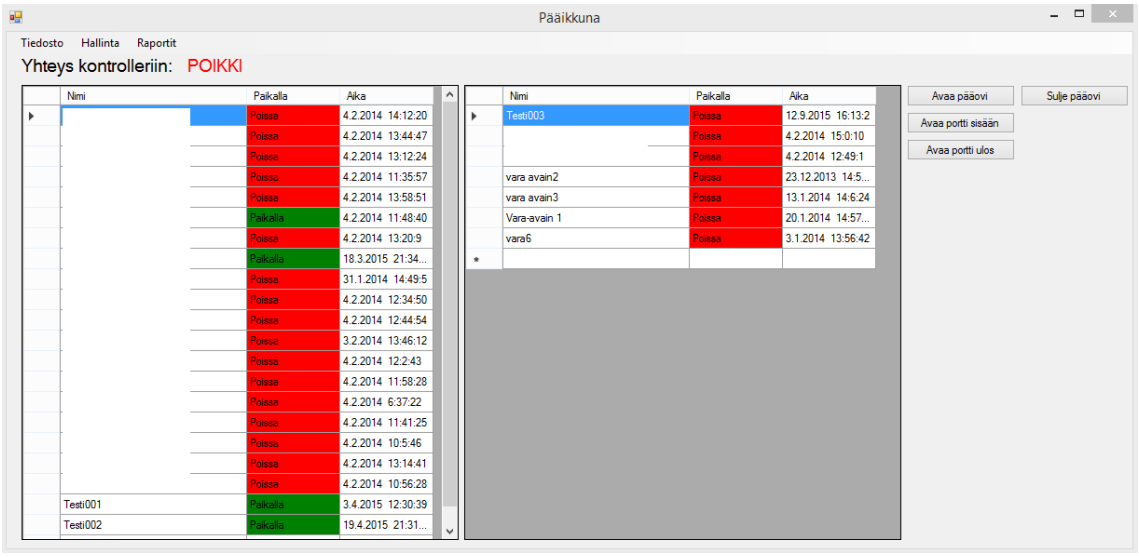
4.2 Globaalit muuttujat

Seuraavaksi on esitetty globaalien muuttujien määrittelyn lähdekoodi. Tämä on tehty 00_Globaalit muuttujat.vb-nimisessä moduulissa ja nämä muuttujat ovat käytettävissä missä tahansa ohjelman ikkunassa ja aliohjelmassa.

```
Module _00_Globaalit_muuttujat
    Public Sarjaporttinimi As String ' Sarjaportin nimi
    Public porttimuuttuu As Boolean
    Public luetuttavut(6) As Byte
    Public kirjoitatavut(6) As Byte

    Public EEPROM(255) As Byte
    Public kirjoitaEEPROM(1) As Boolean
    Public lueEEPROM(1) As Boolean

    Public päivitäpaikallaolijat(1) As Boolean
End Module
```

Yllä luodaan liikennöinnin käyttämät globaalit muuttujat. Ensimmäinen muuttuja, ”Sarjaporttinimi”, on nimi, jolla Windows tunnistaa sarjaportin. Toinen muuttuja ”porttimuuttuu”, on tieto tietoliikenteestä vastaavalle aliohjelmalle, että liikennöinti tulee pysäyttää ja aloittaa uudelleen, koska sarjaporttia, jota käytetään liikennöintiin, on vaihdettu. Neljäs muuttuja, ”luetuttavut(6)”, sisältää seitsemän tavun taulukon ja sisältää viimeiset seitsemän kulunvalvontalaitteelta luettua tavua. Viides muuttuja ”kirjoitatavut(6)”, sisältää seitsemän tavun taulukon, joka sisältää seitsemän seuraavaa kulunvalvontalaitteelle lähetettävää tavua.

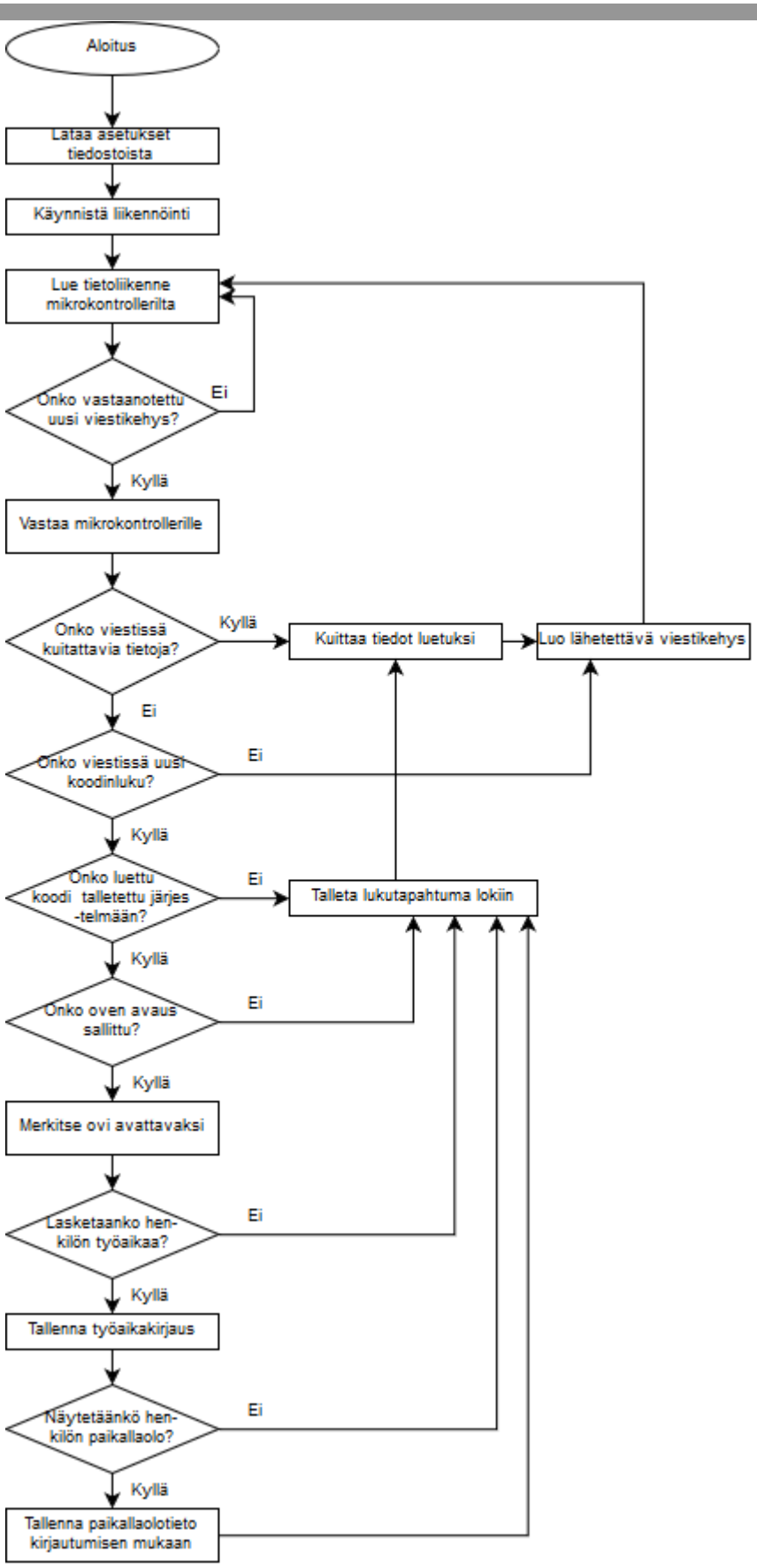
4.3 Pääikkuna



Kuva 2. Ohjelman pääikkuna

Ohjelman pääikkunassa on nähtävissä sisään kirjautuneet käyttäjät, joiden läsnäolo on määritelty näytettäväksi. Kyseisistä käyttäjistä on näkyvissä myös viimeisimpien kirjautumisien ajat. Paikallaolotietojen vierellä oikealla puolella, on painikkeet kääntöportin ja oven ohjaamiseen ilman käyttäjän tunnistamista. Tämä toiminta on tarkoitettu vieraiden kulun sallimiseksi. Ikkunan ylälaidassa on valikko, josta voidaan avata ohjelman muut ikkunat. Pääikkunan yleisnäkymä on esitetty kuvassa 2. Mikäli pääikkuna suljetaan, koko ohjelma suljetaan. Jos pääikkunaa ei manuaalisesti piiloteta, se on aina näytöllä, vaikka se voikin jäädä muiden ikkunoiden alle. Ikkunan vasemassa ylälaidassa näytetään, onko ohjelmalla yhteyttä mikrokontrolleriin.

Pääikkunan avauksen yhteydessä ladataan myös ohjelman asetukset ja se hoitaa taustatoiminnoissaan myös tietoliikennettä, luettujen koodien tarkastusta ja lokitiedostojen kirjoittamista. Kuviossa 2 on esitetty pääikkunan käyttäjästä riippumatoman toiminnan periaate vuokaaviona.



Kuvio 2. Pääikkunan käyttäjästä riippumattomat toiminnot

4.3.1 Pääikkunan paikalliset muuttujat

```
Public Class Pääikkuna
    Private virheet As UInt64 = 0
    Private viestit As UInt64 = 0
    Private viesti As String = ""
    Private liikennöintivirhe As Boolean
```

Ennen varsinaista ohjelmakoodia määritellään ikkunan sisäiset muuttujat, jotka ovat kaikkien ikkunan aliohjelmien käytössä. Ensimmäisenä ovat virhe ja viestilaskurit, joilla voi seurata sarjaliikenteen kulkua. Kolmantena on viesti, joka on tarkoitettu ohjelman sisäisten häiriöviestien välittämiseksi käyttäjälle. Viimeisenä on bittitieto onko liikennöinti käynnissä ja oikean muotoista.

4.3.2 Pääikkunan lataus

Kun pääikkuna ladataan, suoritetaan yhden kerran aliohjelma, joka tarkistaa löytyykö tietokoneesta tarvittavat kansiot lokitiedostoille, sekä asetustiedostot järjestelmää varten. Mikäli kansioita ja tiedostoja ei löydy, sellaiset luodaan. Tämän jälkeen ladataan asetukset tiedostoista, asetetaan liikennöintitavut sopivaan tilaan ja käynnistetään liikennöinti. Tämän jälkeen päivitetään paikallaolijalistat tiedostosta, sekä kutsutaan näkyviin paikallaolija-ikkuna. Alla käydään läpi lähdekoodi joka suorittaa nämä toiminnat.

```
Private Sub Pääikkuna_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load
        ' Tarkastetaan avatessa onko asetustiedos-
        ja/kansiota olemassa. Jos ei ole, luodaan.
       CheckForFolder("c:\Kulunvalvonta")
       CheckForFolder("C:\Kulunvalvonta\Asetukset")
       CheckForFolder("C:\Kulunvalvonta\Avaimet")
       CheckForFolder("C:\Kulunvalvonta\Oviloki")
       CheckForFolder("C:\Kulunvalvonta\Oviloki2")
       CheckForFolder("C:\Kulunvalvonta\Oviloki3")
       CheckForFolder("C:\Kulunvalvonta\Henkilöt")
       CheckForFolder("C:\Kulunvalvonta\Henkilöt\Paikalla")
```

Yllä esitetty koodi tarkistaa tarvittavien kansioiden olemassaolon ja luo tarvittavat kansiot, mikäli niitä ei ole olemassa. CheckForFolder on aliohjelma, joka esitellään luvussa 4.3.9.

```
If Not System.IO.File.Exists
("C:\Kulunvalvonta\Asetukset\Sarjaportti.data") Then
' luodaan sarjaportti-tiedosto, jos sellaista ei ole
FileOpen(1, "C:\Kulunvalvonta\Asetukset\Sarjaportti.data",
OpenMode.Output)
Write(1, "COM1")
FileClose(1)
End If

If Not System.IO.File.Exists
("C:\Kulunvalvonta\Asetukset\EEPROM.data") Then
' luodaan sarjaportti-tiedosto, jos sellaista ei ole
```

```
FileOpen(1, "C:\Kulunvalvonta\Asetukset\EEPROM.data", Open-
Mode.Output)
For n = 0 To 255
          PrintLine(1, 0)
Next
FileClose(1)
End If

If Not System.IO.File.Exists
("C:\Kulunvalvonta\Asetukset\Aukiaika1.data") Then
' luodaan sarjaportti-tiedosto, jos sellaista ei ole
FileOpen(1, "C:\Kulunvalvonta\Asetukset\Aukiaika1.data",
OpenMode.Output)
Write(1, 50)
FileClose(1)
End If

If Not System.IO.File.Exists
("C:\Kulunvalvonta\Asetukset\Aukiaika2.data") Then
' luodaan sarjaportti-tiedosto, jos sellaista ei ole
FileOpen(1, "C:\Kulunvalvonta\Asetukset\Aukiaika2.data",
OpenMode.Output)
Write(1, 30)
FileClose(1)
End If

If Not System.IO.File.Exists
("C:\Kulunvalvonta\Asetukset\Aukiaika3.data") Then
' luodaan sarjaportti-tiedosto, jos sellaista ei ole
FileOpen(1, "C:\Kulunvalvonta\Asetukset\Aukiaika3.data",
OpenMode.Output)
Write(1, 30)
FileClose(1)
End If

If Not System.IO.File.Exists
("C:\Kulunvalvonta\Asetukset\Kerroin.data") Then
' luodaan sarjaportti-tiedosto, jos sellaista ei ole
FileOpen(1, "C:\Kulunvalvonta\Asetukset\Kerroin.data",
OpenMode.Output)
PrintLine(1, 20, ",", 19)
FileClose(1)
End If
```

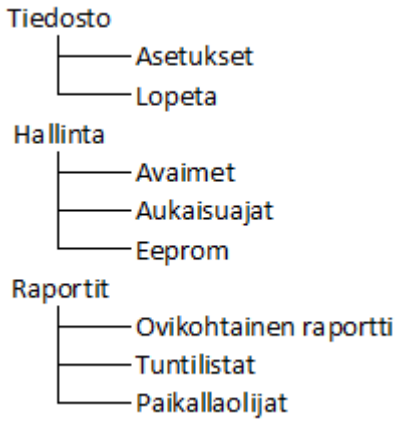
Yllä esitetty koodi tarkistaa asetustiedostojen olemassaolon ja luo tiedostot, mikäli niitä ei ole olemassa. Tiedostoihin asetetaan oletusasetukset.

```
FileOpen(1, "C:\Kulunvalvonta\Asetukset\Sarjaportti.data",
OpenMode.Input)
Input(1, Sarjaporttinimi)
FileClose(1)

        FileOpen(1,
        "C:\Kulunvalvonta\Asetukset\EEPROM.data", OpenMo-
        de.Input)
        For n = 0 To 255
            Input(1, EEPROM(n))
        Next
        FileClose(1)

        'kirjoitettaville tavuille alkuasetukset
        kirjoitatavut(0) = 1
        kirjoitatavut(1) = 1
```

```
        kirjoitatavut(2) = 0
        kirjoitatavut(3) = 0
        kirjoitatavut(4) = 0
        kirjoitatavut(5) = 0
        kirjoitatavut(6) = 210
        ' laitetaan sarjaporttiliikenne käyntiin
        Liikennöinti.RunWorkerAsync()

        Call päivitäpaikaololista()
        Paikallaolijat.Visible = True
    End Sub
```

Yllä oleva koodi lataa asetukset asetustiedostoista. Mikäli asetustiedostoja ei ollut olemassa, olisi niiden luomatta jättäminen aiheuttanut virheen tässä kohdassa ohjelman suoritusta. Tämän lisäksi koodi päivittää paikallaolijalistan näytölle. Paikallaololistan päivitys tehdään omalla aliohjelmallaan joka on esitetty luvussa 4.3.8.

4.3.3 Pääikkunan valikot



Kuva 3. Pääikkunan valikot puu-esityksenä

Kuvassa 3 on esitetty ohjelman pääikkunan valikot. Lopettamista lukuunottamatta valikoista tuodaan esille ohjelman hallintaan tai raportointiin käytettäviä ikkunoita. Lopeta valinta sulkee koko ohjelman.

```
Private Sub AsetuksetToolStripMenuItem_Click(sender As Sys-
tem.Object, e As System.EventArgs) Handles AsetuksetToolSt-
ripMenuItem.Click
        Asetukset.Visible = True
End Sub

Private Sub LopetaToolStripMenuItem_Click(sender As Sys-
tem.Object, e As System.EventArgs) Handles LopetaToolStrip-
MenuItem.Click
        End
End Sub

Private Sub HallintaToolStripMenuItem_Click(sender As Sys-
tem.Object, e As System.EventArgs) Handles HallintaToolSt-
ripMenuItem.Click
        Avainten_hallinta.Visible = True
End Sub
```

```
Private Sub AukaisuajatToolStripMenuItem_Click(sender As
System.Object, e As System.EventArgs) Handles Aukaisuajat-
ToolStripMenuItem.Click
        Aukaisuajatasetus.Visible = True
End Sub

Private Sub EEPROMToolStripMenuItem_Click(sender As Sys-
tem.Object, e As System.EventArgs) Handles EEPROMToolStrip-
MenuItem.Click
        Piirin_tietojen_varmuuskopiointi.Visible = True
End Sub

Private Sub PääoviToolStripMenuItem_Click(sender As Sys-
tem.Object, e As System.EventArgs) Handles PääoviToolStrip-
MenuItem.Click
        Pääovi_raportti.Visible = True
End Sub

Private Sub PaikallaolijatToolStripMenuItem_Click(sender As
System.Object, e As System.EventArgs) Handles Paikallaoli-
jatToolStripMenuItem.Click
        Paikallaolijat.Visible = True
End Sub

Private Sub TuntilistatToolStripMenuItem_Click(sender As
System.Object, e As System.EventArgs) Handles Tuntilistat-
ToolStripMenuItem.Click
        Tuntilistat.Visible = True
End Sub
```

Yllä esitetty koodi käsittelee valikoiden toiminnat. Jokainen valikon valinta käsitellään omassa aliohjelmassaan. Suurin osa valikoista tuo näkyviin toisen ikkunan ohjelmasta, poikkeuksena ”Lopeta”, joka lopettaa ohjelman suorituksen.

4.3.4 Pääikkunan painikkeet

Pääikkunassa on neljä painiketta, joilla ohjataan oven ja kääntöportin toimintaa. Nappien painaminen muuttaa mikrokontrollerille lähetettävän komennon taulukon 4 mukaisesti, ilman että RFID-tunnistetta on luettu. Neljännen painikkeen (sulje pääovi) toiminta on poistettu mikrokontrollerin koodista, sekä liikennöintitaulukosta. Poisto tehtiin, kun todettiin, että pääoven salpa toimii pulssilla, eikä sitä voi lukita auki jatkuvalla signaalilla. Täten signaalinen päälle jättämisestä ei ole mitään hyötyä, eikä signaalin lopettamiseen tarvita omaa komentoa. Nappi jätettiin kuitenkin ohjelmaan mahdollisia tulevaisuuden muutoksia ajatellen.

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
        kirjoitatavut(4) = 5
End Sub

Private Sub Button2_Click(sender As System.Object, e As
System.EventArgs) Handles Button2.Click
        kirjoitatavut(4) = 6
End Sub

Private Sub Button3_Click(sender As System.Object, e As
System.EventArgs) Handles Button3.Click
```

```
        kirjoitatavut(4) = 7
End Sub

Private Sub Button4_Click(sender As System.Object, e As
System.EventArgs) Handles Button4.Click
        kirjoitatavut(4) = 15
End Sub
```

Edellä on esitetty koodit, jotka on sidottu nappien toimintaan. Painike1 (Button1) avaa pääoven, painike2 (Button2) avaa kääntöportin sisäänpäin ja painike3 (Button3) avaa kääntöportin ulospäin. Painike4 (Button4) lopetti aiemmin pääoven avaussignaalin, mutta poistettiin myöhemmin käytöstä. Toiminta aikaansaadaan muokkaamalla mikrokontrollerille lähetettävää viestikehystä.

4.3.5 Pääikkunan näytön päivitys

Pääikkunassa olevat muuttuvat elementit päivitetään näkyviin NäytönPäivitykset-nimisellä ajastimella, joka suoritetaan 100 millisekunnin välein. Ajastimella päivitetään näytölle mikrokontrollerin yhteystieto, sekä ikkunassa piilossa olevat elementit, joissa näkyy sen hetkinen lähetettävä ja viimeksi vastaanotettu tietoliikennekehys. Nämä tiedot piilotettiin, koska käyttäjän ei tarvitse nähdä niitä, mutta niistä on hyötyä etsiessä vikoja ja tarkkaillessa laitteen ja mikrokontrollerin toimintaa.

```
Private Sub PaivitaNaytto(sender As System.Object, e As
System.EventArgs) Handles NäytönPäivitykset.Tick
        If liikennöintivirhe = True Then
            virheteksti.Text = "POIKKI"
            virheteksti.ForeColor = Color.Red
        Else
            virheteksti.Text = "PÄÄLLÄ"
            virheteksti.ForeColor = Color.Green
        End If
```

Edellä esitetty koodi päivittää mikrokontrollerin yhteystiedon näytille, sekä vaihtaa sen väriä tilanteen mukaan.

```
        Luettu.Text = luetuttavut(0)
        Label2.Text = luetuttavut(1)
        Label3.Text = luetuttavut(2)
        Label4.Text = luetuttavut(3)
        Label5.Text = luetuttavut(4)
        Label6.Text = luetuttavut(5)
        Label7.Text = luetuttavut(6)

        Label16.Text = kirjoitatavut(0)
        Label15.Text = kirjoitatavut(1)
        Label14.Text = kirjoitatavut(2)
        Label13.Text = kirjoitatavut(3)
        Label12.Text = kirjoitatavut(4)
        Label11.Text = kirjoitatavut(5)
        Label10.Text = kirjoitatavut(6)


        Label8.Text = "Viestit: " + Str(viestit)
        Label9.Text = "Virheet: " + Str(virheet)
```

Edellä esitetty koodi päivittää näytölle vastaanotetun ja lähetetyn viestikehyksen, sekä tiedon montako viestiä mikrokontrolleri ja tietokone ovat vaihtaneet, sekä niissä mahdollisesti sattuneiden virheiden määrän.

```
        If päivitäpaikallaolijat(0) = True Then
            Call päivitäpaikaololista()
            päivitäpaikallaolijat(0) = False
        End If
End Sub
```

Edellä esitetty koodi kutsuu tarvittaessa aliohjelmaa, joka päivittää päänäytön paikallaolijalistauksen. Paikallaololistan päivitys tehdään omassa aliohjelmassaan, joka on esitetty luvussa 4.3.8. Päivitys tehdään vain, mikäli paikallaolossa on tapahtunut muutoksia. Muuttuja jossa tämä määritetään asetetaan liikennöinnin aliohjelmassa, jota ajetaan pääikkunan taustaprosessina. Liikennöinnin aliohjelma esitetään luvussa 4.3.7.

4.3.6 Pääikkunan taustaprosessit

Pääikkunan taustaprosesseja ajetaan ajastimella, jota suoritetaan 100ms välein. Ajastin tarkastaa onko liikennöinti käynnissä ja onko järjestelmässä käyttäjälle esitettäviä viestejä. Mikäli liikennöinti ei ole käynnissä, se käynnistetään ja mahdolliset viestit esitetään viestilaatikoina.

```
Private Sub Taustaprosessit_Tick(sender As System.Object, e
As System.EventArgs) Handles Taustaprosessit.Tick
        Dim roska As String
        If Liikennöinti.IsBusy = False Then
            Liikennöinti.RunWorkerAsync()
        End If

        If viesti > "" Then
            roska = viesti
            viesti = ""
            MsgBox(roska)
        End If
End Sub
```

Edellä esitetty koodi tarkistaa liikennöinnin ja käyttäjälle näytettävät viestit. Liikennöinti on omassa säikeessään suoritettava ohjelma, joka ei häiritse pääohjelman suoritusta. Liikennöinti toteutettiin omaan säikeeseensä, koska muuten ikkunoiden päivitys häiriintyisi, mikäli tietoliikenteessä esiintyy viiveitä.

4.3.7 Tietoliikenne

Tietokoneen ja mikrokontrollerin välinen tietoliikenne suoritetaan omassa säikeessään pääikkunan alaisuudessa. Mikrokontrolleri näyttäytyy tietokoneelle sarjaporttina, jolta odotetaan viestejä, joihin vastataan asetusten sekä tietokoneen käyttäjän toimintojen perusteella.

```
Private Sub BackgroundWorker1_DoWork(sender As System.Object, e As System.ComponentModel.DoWorkEventArgs)
Handles Liikennöinti.DoWork
On Error GoTo virhe
```

```
Dim EEPROMluku As Byte = 0
Dim lukusarjaportista(6) As Byte
Dim n As UInteger
Dim ekaluku As Boolean = True
Dim avaa As Boolean = True
Dim portti As Byte
Dim viimeisinkoodi As String
Dim tarkastukset1 As String = ""
Dim tarkastukset2(3) As Integer
Dim tarkastukset3(9), löytyi As Boolean

Dim di As New IO.DirectoryInfo("c:\Kulunvalvonta\Avaimet\")

Dim diar1 As IO.FileInfo() = di.GetFiles()
Dim dra As IO.FileInfo
Dim alustus(1) As Char
```

Edellä esitetty koodi luo liikennöinnin käyttämät sisäiset muuttujat. Muuttujaan diar1 luetaan muuttujassa di esitetyn kansion tiedostolistaus, tässä kansiossa oleva tiedostolistaus on samalla myös lista järjestelmän tuntemista avaimista.

```
Dim sarjaportti As New IO.Ports.SerialPort
' luodaan ja avataan yhteys sarjaporttiin
sarjaportti.PortName = Sarjaporttinimi
sarjaportti.BaudRate = 9600
sarjaportti.Parity = IO.Ports.Parity.None
sarjaportti.DataBits = 8
sarjaportti.ReadTimeout = 500
sarjaportti.WriteTimeout = 500
sarjaportti.StopBits = IO.Ports.StopBits.One
sarjaportti.Handshake = IO.Ports.Handshake.None
sarjaportti.DtrEnable = False

sarjaportti.Open()

kirjoitatavut(6) = 210
kirjoitatavut(5) = 0
kirjoitatavut(4) = 0
kirjoitatavut(3) = 0
kirjoitatavut(2) = 0
kirjoitatavut(1) = 0
kirjoitatavut(0) = 0
```

Edellä esitetty koodi luo sarjaportille ja liikennöintiin tarvittavat muuttujat, asettaa niihin liikennöinnissä käytettävät asetukset ja avaa yhteyden sarjaporttiin. Tämän jälkeen alustetaan mikrokontrollerille lähetettävien tavujen data.

```
Do
liikennealku:
  Do
    alustus(0) = alustus(1)
    sarjaportti.Read(alustus, 1, 1)
  Loop Until alustus(0) = "P" And alustus(1) = "C"

  Do
Loop Until sarjaportti.BytesToRead >= 7
```

Edellä esitetyssä koodissa aloitetaan liikennöinnissä käytetty silmukka, jota suoritetaan, kunnes liikenteessä tapahtuu virhe, tai ohjelma suljetaan. Rivi ”liikennealku:” on piste johon ohjelmassa hypätään, mikäli oikea määrä tavuja saadaan luettua mikrokontrollerilta, mutta viimeisen tavun kaiutus ei ole onnistunut.

Toisessa silmukassa luetaan yksittäisiä tavuja sarjaportin lukupuskurista, kunnes kaksi peräkkäistä tavua muodostavat ascii-merkkeinä sanan ”PC”. Kolmas silmukka odottaa että puskurissa on vähintään seitsemän merkkiä luettavaksi, minkä jälkeen ohjelman suoritusta jatketaan eteenpäin.

```
sarjaportti.Read(lukusarjaportista,    0,    lukusarjaportis-
ta.Length)

' System.Threading.Thread.Sleep(200)

sarjaportti.Write(kirjoitatavut, 0, 1)
sarjaportti.Write(kirjoitatavut, 1, 1)
sarjaportti.Write(kirjoitatavut, 2, 1)
sarjaportti.Write(kirjoitatavut, 3, 1)
sarjaportti.Write(kirjoitatavut, 4, 1)
sarjaportti.Write(kirjoitatavut, 5, 1)
sarjaportti.Write(kirjoitatavut, 6, 1)


If kirjoitatavut(4) = 5 Then kirjoitatavut(4) = 0

luetuttavut(0) = lukusarjaportista(0)
luetuttavut(1) = lukusarjaportista(1)
luetuttavut(2) = lukusarjaportista(2)
luetuttavut(3) = lukusarjaportista(3)
luetuttavut(4) = lukusarjaportista(4)
luetuttavut(5) = lukusarjaportista(5)
luetuttavut(6) = lukusarjaportista(6)
```

Edellä esitetyssä koodissa luetaan ensin tyhjäksi koko sarjaportin puskuri, minkä jälkeen tarkistetaan mikäli viimeisin viesti mikrokontrollerilta kertoo, että ovi on avattu, milloin nollataan mahdollinen avauskäsky. Tämä tehdään, koska käyttöönotossa huomattiin, että nykyisellä ohjelmakoodilla ovi ja portit voivat saada tuplapulssin. Tuplapulssi ei vaikuta kääntöportin toimintaan, mutta oven salvan solenoidista kuului kaksi kertaa vetävän solenoidin ääni. Viestinnässä käytetyt tiedot on esitetty taulukoissa 3 ja 4. Tämän jälkeen kirjoitetaan valmiina muuttujissa oleva viesti sarjaporttiin.

```
If lukusarjaportista(6) = 210 Then
  viestit = viestit + 1
Else
  If ekaluku Then
    ekaluku = False
  Else
    virheet = virheet + 1        ' ensimmäisellä kierroksel-
    la tulee aina yksi luku "väärin"
  End If  ' mikäli vääriä lisää, niin lasketaan virheitä.
  GoTo liikennealku  ' yhteys ei kuitenkaan olevarsinaises-
  ti poikki
End If     ' toimintoja ei kuitenkaan saa tehdä
liikennöintivirhe = False
```

Edellä esitetty koodi tarkistaa ensin kaiuttaako mikrokontrolleri viimeisen liikennöintitavun oikein takaisin. Jos kyseessä on ensimmäinen luku mikrokontrollerilta, eikä kontrollerille ole vielä lähetetty viestikehystä, ei myöskään kaiutusta voi vielä tapahtua, joten ensimmäisellä lukukerralla virhettä ei lasketa. Tässä lasketut virheet tarkoittavat, että tietoliikennettä saapuu kontrollerilta, mutta siinä esiintyy virheitä.

```
' porttien avauksen tarkastus
If luetuttavut(4) = 1 And kirjoitatavut(4) <> 10 Then
     viimeisinkoodi = Trim(Str(luetuttavut(0))) + "."
     + Trim(Str(luetuttavut(1))) + "."
     + Trim(Str(luetuttavut(2)))
     + "." + Trim(Str(luetuttavut(3)))
     portti = luetuttavut(5)
```

Edellä esitetyssä koodissa on ensimmäisenä tarkastus, joka tarkistaa onko uusi koodi luettu ja mikäli mikrokontrolleri ilmoittaa että uusi koodi on saatavilla, tarkistetaan ollaanko kyseistä koodia jo kuittaamassa. Mikäli molemmat ehdot täyttyvät, luetaan saatu koodi ohjelman käyttämään muotoon, sekä otetaan muuttujaan ”portti” talteen tieto miltä lukijalaitteelta koodi luettiin.

```
'Tarkistetaan onko avain olemassa
di.Refresh()
diar1 = di.GetFiles()
     For Each dra In diar1
          If Trim(dra.ToString) = Trim(viimeisinkoodi) Then
               löytyi = True
```

Kun uusi koodi on luettu, päivitetään avaimien tiedostokansion tiedostolistaus, siltä varalta että uusia avaimia on luotu, sitten viime koodinluvun. Tämän jälkeen luettua koodia verrataan löytyneisiin avaintiedostonnimiin yksi kerrallaan, kunnes oikea tiedosto löytyy tai tiedostot on käyty läpi.

```
If portti = 1 Then
FileOpen(1, "c:\Kulunvalvonta\Avaimet\"
+ Trim(viimeisinkoodi), OpenMode.Input)
' luetaan tiedostosta onko oikein avata ovea

Input(1, Nimet.Text)
Input(1, tarkastukset1)
Input(1, tarkastukset2(0))
Input(1, tarkastukset2(1))
Input(1, tarkastukset2(2))
Input(1, tarkastukset2(3))
Input(1, tarkastukset3(0))
Input(1, tarkastukset3(1))
Input(1, tarkastukset3(2))
Input(1, tarkastukset3(3))
Input(1, tarkastukset3(4))
Input(1, tarkastukset3(5))
Input(1, tarkastukset3(6))
Input(1, tarkastukset3(7))
Input(1, tarkastukset3(8))
Input(1, tarkastukset3(9))
FileClose(1)
```

Tämän jälkeen tarkistetaan mitä ovea tai porttia käsitellään ja luetaan avaimen asetustiedostosta tiedot, milloin avaimella kulkeminen on sallittu. Avaintiedostoon talletetut asetukset on kuvailtu kappaleessa 4.9.2. Jokaiselle avattavalle kohteelle on omat vastaavat tarkastuksensa, jotka alkavat porttinumeron tarkastuksesta ja päättyvät End if-komentoon joka sulkee porttitarkastuksen if-lauseen. Koska kaikki kolme tarkastelua ovat porttinumeroa ja viestikehykseen kirjoitettavaa tavua lukuun ottamatta identtiset, vain oven tarkastuksen lähdekoodi on esitetty tässä kappaleessa.

```
If tarkastukset3(8) = True Then
           avaa = True
```

Ensimmäisenä tarkistetaan mikäli kulku on sallittu aina, päivästä ja kellosta riippumatta. Mikäli näin on, asetetaan muuttuja avaa todeksi, eikä muita tarkastuksia tarvitse suorittaa.

```
ElseIf tarkastukset3(Now.DayOfWeek - 1) = True Then
```

Edellä oleva lauseke tarkistaa onko avaimella sallittu avata ovea kuluvana päivänä. Mikäli ei ole tarkastuksia jotka ovat ennen hyppypistettä ”eiavataovea:” ei toteuteta. Elementti ”Now.DayOfWeek” palauttaa kuluvan viikonpäivän numerona yhdestä seitsemään. Siitä vähennetään yksi, jotta laskenta alkaisi nollasta, kuten tiedostosta luetut luvat päiville alkavat.

```
If tarkastukset2(0) > Now.Hour Then GoTo eiavataovea

If tarkastukset2(0) = Now.Hour And tarkastukset2(1) >
Now.Minute Then GoTo eiavataovea
```

Edellä olevat ehtolauseet tarkistavat onko kello ylittänyt aloituksen sallitusta kulkuajasta. Mikäli kello on vähemmän kuin sallitun ajan alku hypätään suorituksessa hyppypisteeseen ”eiavataovea:”

```
If tarkastukset2(2) < Now.Hour Then GoTo eiavataovea

If tarkastukset2(2) = Now.Hour And tarkastukset2(3) <
Now.Minute Then GoTo eiavataovea
```

Edellä olevat ehtolauseet tarkistavat mikäli kello on ylittänyt sallitun kulkuajan avaimella. Mikäli kello on ylittänyt sallitun kulkuajan hypätään suorituksessa hyppypisteeseen ”eiavataovea:”.

```
avaa = True
End If
```

Mikäli mikään kieltoehdoista ei täyttynyt, asetetaan muuttuja ”avaa” todeksi ja jatketaan ohjelman suoritusta.

```
eiavataovea:
FileOpen(1, "c:\Kulunvalvonta\Oviloki\"
+ Trim(Str(Now.Year)) + "." + Trim(Str(Now.Month)),
OpenMode.Append)

PrintLine(1,  viimeisinkoodi,  ",",  tarkastukset1,  ",",
Now.Day, ",", Now.Month, ",", Now.Year, ",", Now.Hour, ",",
Now.Minute, ",", Now.Second, ",", avaa)
```

```
FileClose(1)
```

Hyppypaikan ”eiavataovea:” nimi ei viittaa tuleviin toimintoihin, vaan siihen että jätettiin asettamasta muuttuja ”avaa” todeksi. Hyppypaikassa kirjoitetaan lokitiedostoon tieto koodinlukutapahtumasta, sekä tieto sallittiinko oven tai portin avaaminen. Tiedoston nimi muodostuu vuodesta ja kuukaudesta, pisteellä erotettuna. Näin ollen joka kuukaudelle luodaan oma tiedostonsa ja koska harvemmin muuttuva luku, eli vuosiluku, on tiedoston nimessä ensin, voidaan tiedostot järjestää nimen mukaiseen järjestykseen, niin että ne pysyvät oikeassa järjestyksessä. Mikäli luvut olisi asetettu tiedoston nimeen toisessa järjestyksessä, nimien mukainen järjestys listaisi tiedostot ensisijaisesti kuukauden mukaan ja vasta sen jälkeen vuosiluvun mukaan.

Tiedostoon kirjoitetaan rivi, joka sisältää päiväyksen, kellonajan, nimen johon avain yhdistettiin sekä tiedon sallittiinko avaus boolean arvona, jossa tosi tarkoittaa että avaus sallittiin, epätosi että avaus kiellettiin. Tiedot on eroteltu pilkuilla.

Edellä olevat koodit esittävät koodinkäsittelyn lukijasta 1, joka on tarkoitettu ulko-oven avaamiseen. Ulko-ovi käsitellään vain sisäänpäin kulkiessa, eikä se vaikuta läsnäolotietoon, eikä tuntilistoihin. Lukija 2 käsittelee kulkemisen kääntöportista sisään, mikä kirjaa aloituksen työajan alkaneeksi, sekä henkilön paikalla olevaksi. Lukija 3 käsittelee kulkemisen kääntöportista ulos, mikä kirjaa työajan päättyneeksi, sekä henkilön poissaolevaksi. Tulevia koodeja on edeltänyt vastaavat tarkastukset avausehdoista kuin lukijassa 1, ne ovat kuitenkin lähes identtisiä edeltäneiden koodien kanssa, joten niitä ei esitetä tässä. Kolmannen lukijan koodissa ei ole vertailuja vuorokaudenajasta, vaan työajan lopetus on sallittu mihin tahansa kellonaikaan.

```
'tiedot paikallaololistaan
If tarkastukset3(7) = True And avaa = True Then

  FileOpen(1, "C:\Kulunvalvonta\Henkilöt\Paikalla\"
  + Trim(tarkastukset1), OpenMode.Output)

  PrintLine(1, True, ",", Now.Date, ","
  , Trim(Str(Now.Hour)) + ":" + Trim(Str(Now.Minute))
  + ":" + Trim(Str(Now.Second)))

  FileClose(1)
End If
```

Edellä esitetty koodi tarkistaa talletetaanko henkilöstä paikallaolotietoa. Mikäli tieto talletetaan ja portille annettiin avauslupa kirjoitetaan paikallaolotieto tiedostoon. Tiedostoon kirjoitetaan ensin läsnäolotieto boolean arvona, jossa tosi tarkoittaa paikalla ja epätosi ei paikalla. Kumpi tieto kirjoitetaan, riippuu siitä onko koodi luettu lukijasta kaksi vai kolme.

```
päivitäpaikallaolijat(0) = True
päivitäpaikallaolijat(1) = True

Next
```

Kun tiedosto on kirjoitettu, asetetaan muuttujat, joista pääikkuna ja paikallaolija-ikkuna tietävät paikallaololistan muuttuneen. Kumpikin ikkuna tarkastaa jatkuvasti muuttujien boolean arvoja ja kun ne muuttuvat arvoon tosi, ikkunat päivittävät listat tiedostoista ja muuttavat arvot takaisin epätosiksi. Komento next vaihtaa tiedostoa etsittävästä avaimesta ja aloittaa tarkastelun alusta.

```
If löytyi = False And portti = 1 Then
  FileOpen(1, "c:\Kulunvalvonta\Oviloki\"
  + Trim(Str(Now.Year)) + "." + Trim(Str(Now.Month))
  , OpenMode.Append)

  PrintLine(1, viimeisinkoodi, ",", "Tuntematon avain",
  ",", Now.Day, ",", Now.Month, ",", Now.Year, ",",
  Now.Hour, ",", Now.Minute, ",", Now.Second, ",", avaa)

  FileClose(1)

ElseIf löytyi = False And portti = 2 Then
  FileOpen(1, "c:\Kulunvalvonta\Oviloki2\" +
  Trim(Str(Now.Year)) + "." + Trim(Str(Now.Month))
  , OpenMode.Append)

  PrintLine(1, viimeisinkoodi, ",", "Tuntematon avain",
  ",", Now.Day, ",", Now.Month, ",", Now.Year, ",",
  Now.Hour, ",", Now.Minute, ",", Now.Second, ",", avaa)

  FileClose(1)

ElseIf löytyi = False And portti = 3 Then
  FileOpen(1, "c:\Kulunvalvonta\Oviloki3\" +
  Trim(Str(Now.Year)) + "." + Trim(Str(Now.Month))
  , OpenMode.Append)

  PrintLine(1, viimeisinkoodi, ",", "Tuntematon avain",
  ",", Now.Day, ",", Now.Month, ",", Now.Year, ",",
  Now.Hour, ",", Now.Minute, ",", Now.Second, ",", avaa)
                    FileClose(1)
End If
löytyi = False

kirjoitatavut(4) = 10 ' kuitataan koodi luetuksi
End If
```

Edellä esitetty koodi suoritetaan kun kaikki avaintiedostot on tarkastettu. Mikäli yksikään avaintiedosto ei sopinut luettuun koodiin, kirjoitetaan lokiin tieto, että ovea tai kääntöporttia on yritetty avata järjestelmään kuulumattomalla avaimella. Kirjoitus on kuten hyppypaikassa ”eiavataovea:”, mutta koska avaimelle ei ole nimeä kirjoitetaan nimen tilalle ”Tuntematon avain”.

Kun tarkastus on ohi muokataan lähetettävää viestikehystä niin, että mikrokontrolleri tietää koodin olevan luettu, eikä lähetä sitä enää tietokoneelle. Ehtolause joka aloitti uuden koodin tarkastelun päättyy tähän.

```
'jos on avattavaa, avataan
If avaa = True And luetuttavut(4) = 0
And kirjoitatavut(4) <> 10 Then
```

```
  avaa = False
  kirjoitatavut(4) = 4 + portti
  'viesti = kirjoitatavut(4)
End If
```

Kun muuttuja ”avaa” sisältää arvon tosi, kirjoitetaan viestikehykseen tieto mikä portti halutaan avattavaksi. Tämä ei tapahdu samalla ohjelmakierroksella, kuin avaimen tarkastelu, koska kuittaus koodin lukemisesta käsitellään ensin.

```
' portti/ovi on avattu, kuitataan tieto käsitellyksi
If luetuttavut(4) = 4 Or luetuttavut(4) = 5
Or luetuttavut(4) = 6 And kirjoitatavut(4) <> 10 Then
  kirjoitatavut(4) = 10
  avaa = False
End If
```

Mikrokontrolleri ilmoittaa tietokoneelle, että haluttu lukko on avattu, milloin tietokone kuittaa takaisin kontrollerille saaneensa tiedon.

```
' lue EEPROM kokonaisuudessaan piiriltä
If luetuttavut(4) = 0 And kirjoitatavut(4) <> 10
And lueEEPROM(0) = True Then
  kirjoitatavut(4) = 4
  kirjoitatavut(5) = EEPROMluku
End If
```

Mikäli muuttujan ”lueEEPROM(0)” arvo on tosi, halutaan lukea EEPROM-muistin sisältö mikrokontrollerilta tietokoneelle. Muuttuja ”EEPROMluku” toimii EEPROMin osoitteena ja mikäli muita komentoja tai kuittauksia ei ole viestikehyksissä näkyvissä, kirjoitetaan lähtevään kehykseen komento lukea EEPROMia muuttujan osoittamasta osoitteesta.

```
If luetuttavut(4) = 8 And kirjoitatavut(4) <> 10 Then
  EEPROM(EEPROMluku) = luetuttavut(5)
  If EEPROMluku = 255 Then
    EEPROMluku = 0
    lueEEPROM(0) = False
    lueEEPROM(1) = True
  End If
  If lueEEPROM(0) = True Then EEPROMluku = EEPROMluku + 1
  kirjoitatavut(4) = 10
End If
```

Kun mikrokontrolleri ilmoittaa että muisti EEPROMista on luettu, tallennetaan muisti ohjelman muuttujiin ja tarkastetaan, oliko luettu muisti numeroltaan 255, joka on viimeinen luettava muisti. Mikäli viimeinen muisti on luettu, asetetaan ohjelman sisäinen muuttuja, joka komentaa lukemaan koko EEPROMin sisällön arvoon epätosi ja luku valmis muuttuja arvoon tosi. Nämä muuttujat ovat nimeltään ”lueEEPROM(0) ja ”lueEEPROM(1)”. Mikäli kyseessä ei ollut viimeinen luettava muisti, lisätään osoitemuuttujaan ”EEPROMluku” yksi. Lopuksi kirjoitetaan lähtevään muistikehykseen kuittaus, että tieto on käsitelty.

```
' kirjoita EEPROM kokonaisuudessaan piirille
If luetuttavut(4) = 0 And kirjoitatavut(4) <> 10
And kirjoitaEEPROM(0) = True Then
  kirjoitatavut(4) = 8
```

```
  kirjoitatavut(5) = EEPROMluku
  kirjoitatavut(3) = EEPROM(EEPROMluku)
End If
```

Edellä esitetyssä koodissa käsitellään tiedon kirjoittamista mikrokontrollerin EEPROM-muistiin. Kirjoitus toimii lähes samoin kuin lukeminen, mutta viestikehykseen kirjoitetaan myös muistiin talletettava arvo.

```
If luetuttavut(4) = 9 And kirjoitatavut(4) <> 10 Then
  If EEPROMluku = 255 Then
    EEPROMluku = 0
    kirjoitaEEPROM(0) = False
    kirjoitaEEPROM(1) = True
  End If
  If kirjoitaEEPROM(0) = True Then EEPROMluku = EEPROMluku
  + 1
  kirjoitatavut(4) = 10
End If
```

Kirjoitettavan EEPROM-muistitaulukon eteneminen tapahtuu samaan tapaan kuin luettavankin. Vain muuttujien nimet eroavat, kun lukumuuttujissa nimet alkoivat ”lueEEPROM” alkavat ne kirjoittaessa ”kirjoitaEEPROM”. Osoitemuuttuja ”EEPROMluku” on yhteinen sekä luvulle, että kirjoitukselle.

```
' portin aukioloaika kirjoitettu
If luetuttavut(4) = 7 And kirjoitatavut(4) = 3 Then
  viesti = "Aika kirjoitettu kontrolleriin ja tiedostoon."
  kirjoitatavut(4) = 10
End If
```

Edellä oleva koodi saa viestin, että avauspulssin pituus joko ovelle, tai kääntöportille on tallennettu mikrokontrolleriin. Käyttäjälle annetaan viesti, että talletus on tapahtunut sekä tiedostoon, että mikrokontrolleriin. Tiedoston talletus ja viestikehyksen muokkaus tapahtuu ikkunassa”Aukaisuajat” joka on esitelty kappaleessa 4.6. Lähtevään viestikehykseen kirjoitetaan kuittaus, että tieto on käsitelty.

```
If luetuttavut(4) > 11 And luetuttavut(4) < 18 Then
  kirjoitatavut(4) = 10
End If
```

Edellä esitetty koodi kuittaa oven ja portin avaus- ja sulkemisilmoitukset käsitellyksi. Näitä tietoja ei varsinaisesti käytetä mihinkään, mutta niiden pohjalta voidaan seurata mikrokontrollerin toimintaa.

```
If luetuttavut(4) = 0 Then
  If kirjoitatavut(4) = 10 Then kirjoitatavut(4) = 0
End If
```

Edellinen koodi tarkistaa mikäli mikrokontrollerilta ei tule tietoa mistään tapahtumasta. Mikäli tietoa ei ole tarkastetaan onko kuittaustieto tiekoneelta mikrokontrollerille voimassa ja mikäli on, nollataan se. Tämän jälkeen tietoliikenne jatkaa liikennöintiä, kunnes toisessa viestikehyksessä, joko tietokoneelta mikrokontrollerille tai toisinpäin on ilmoituksia uusista toiminnoista.

```
' luetaan porttia kunnes portin asetuksia muutetaan
Loop Until porttimuuttuu = True
porttimuuttuu = False
Exit Sub
```

Mikäli sarjaportin asetuksia on muutettu, lopetetaan silmukka ja suljetaan säie joka suorittaa liikennöintiä. Hetken kuluttua ajastin käynnistää liikenteen uudelleen, muutetulla sarjaportilla.

```
virhe:
sarjaportti.Close()
sarjaportti.Dispose()
liikennöintivirhe = True
End Sub
```

Mikäli ohjelman suorituksessa tapahtuu virhe, joka voi liittyä esimerkiksi sarjaportin käsittelyyn tai tiedostonkäsittelyyn, hypätään hyppypaikkaan ”virhe:”. Sarjaportti suljetaan ja asetetaan muisti ”liikennöintivirhe” todeksi, minkä jälkeen suljetaan säie joka suorittaa liikennöintiä. Hetken kuluttua ajastin käynnistää liikennöinnin uudelleen.

4.3.8 Paikallaololistan päivitys

Paikallaololistan päivitys suoritetaan omassa aliohjelmassaan, joka on sidottu pääikkunaan. Tämä koskee vain pääikkunan paikallaololistaa, omassa ikkunassa olevan listan aliohjelma on sidottu sen omaan ikkunaan.

```
Private Sub päivitäpaikaololista()
Dim di As New IO.DirectoryInfo
("c:\Kulunvalvonta\Henkilöt\Paikalla\")
```

Aliohjelman aluksi määritellään kansio, joka sisältää paikallaolotiedostot.

```
Dim diar1 As IO.FileInfo() = di.GetFiles()
Dim dra As IO.FileInfo
Dim paikalla As Boolean
Dim päivä, aika As String
Dim montako As Integer = 0
Dim läsnä As String = "Poissa"
päivä = ""
aika = ""
```

Tämän jälkeen määritellään aliohjelman muuttujat. Osan arvo asetetaan tyhjäksi, koska ohjelmointiympäristö antaa varoituksen, jos niihin ei aseteta arvoa ennen niiden käsittelyä. Tämä ei varsinaisesti ole ongelma, mutta ympäristö ei näe arvon lukemista tiedostosta varmaksi arvon asettamiseksi.

```
DataGridView1.Rows.Clear()
DataGridView2.Rows.Clear()
```

Tämän jälkeen tyhjennetään näytössä olevat taulukot. Tämä tehdään poistamalla niissä näkyvät rivit.

```
'list the names of all files in the specified directory
For Each dra In diar1
```

```
FileOpen(1,      "c:\Kulunvalvonta\Henkilöt\Paikalla\"      +
Trim(dra.ToString()), OpenMode.Input)
Input(1, paikalla)
Input(1, päivä)
Input(1, aika)
FileClose(1)
```

Tämän jälkeen aloitetaan silmukka, jossa käydään läpi kaikki kansiosta löytyneet tiedostot. Tiedoston nimi on sama kuin henkilön nimi, jonka paikallaoloa tarkastellaan. Tiedostosta luetaan, onko henkilö paikalla, sekä kirjautumispäivä ja aika-.

```
If paikalla = True Then
  läsnä = "Paikalla"
Else
  läsnä = "Poissa"
End If
```

Koska läsnäolotieto on boolean muotoinen, muutetaan sen tosi ja epätosiarvot tekstimuotoon ilmaisemaan onko henkilö paikalla vai poissa.

```
If montako > 20 Then
  DataGridView2.Rows.Add(dra.ToString,
  läsnä, päivä + "  " + aika)
Else
  DataGridView1.Rows.Add(dra.ToString,
  läsnä, päivä + "  " + aika)
End If
```

Tämän jälkeen lisätään nimi, paikallaolotieto, sekä kirjautumispäivä ja aika uutena rivinä taulukkoon. Ensimmäiset 20 nimeä lisätään vasemman puoleiseen taulukkoon, mitä seuraavat nimet lisätään oikeanpuoleiseen taulukkoon. Mikäli oikeanpuoleiseen taulukkoon tulee enemmän nimiä, ilmestyy taulukon oikeaan reunaan rullauspalkki, jota vierittämällä saa esiin nimet, jotka eivät mahdu kerralla näytölle.

```
If paikalla = True And montako < 21 Then
  DataGridView1.Item("Column3", DataGridView1.Rows.Count
  - 2).Style.BackColor = Color.Green
ElseIf montako < 21 Then
  DataGridView1.Item("Column3", DataGridView1.Rows.Count
  - 2).Style.BackColor = Color.Red
End If

If paikalla = True And montako > 20 Then
  DataGridView2.Item("DataGridViewTextBoxColumn2",     Data-
GridView2.Rows.Count - 2).Style.BackColor = Color.Green
ElseIf montako > 20 Then
  DataGridView2.Item("DataGridViewTextBoxColumn2",     Data-
GridView2.Rows.Count - 2).Style.BackColor = Color.Red
End If
```

Tämän jälkeen tarkastetaan kumpaan taulukkoon nimi lisättiin ja muutetaan paikallaolotiedon solu punaiseksi, mikäli henkilö ei ole paikalla ja vihreäksi, mikäli henkilö on paikalla.

```
montako = montako + 1
Next
End Sub
```

Kun tarkastukset henkilön kohdalla on tehty, lisätään muuttujaan ”montako” yksi. Tällä muuttujalla tarkastellaan monesko nimi on käsittelyssä. Tämän jälkeen käsitellään seuraava löytynyt tiedosto. Kun kaikki löytyneet tiedostot on käsitelty, poistutaan aliohjelmasta.

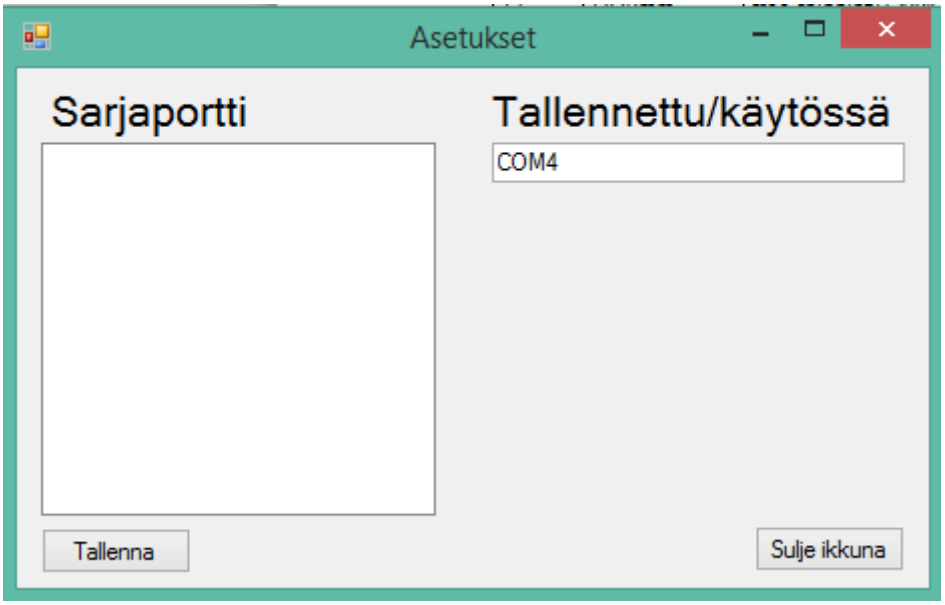
4.3.9 Kansion olemassa olon tarkastus ja luominen

Kansion olemassaolon tarkistus on tehty omaksi aliohjelmakseen.

```
Sub CheckForFolder(ByRef foldername As String)
' Check if folder exists, create if doesn't
Try
  If Not System.IO.Directory.Exists(foldername)
  Then System.IO.Directory.CreateDirectory(foldername)
Catch ex As Exception
  MsgBox("Virhe kansioiden käsittelyssä:" + ex.ToString)
End Try
End Sub
```

Aliohjelma tarkistaa onko kansiota olemassa ja yrittää luoda sen, mikäli sitä ei vielä ole. Mikäli kansion luominen aiheuttaa virheen, ilmoitetaan käyttäjälle Windowsin palauttama virhetieto, sekä ilmoitus kansion käsittelyssä sattuneesta virheestä.

4.4 Asetukset



Kuva 4. Asetusikkuna

Asetusikkunassa on lista tietokoneessa saatavilla olevista sarjaporteista, painike portin käyttöönottoa varten, tekstikenttä joka näyttää käytössä olevan sarjaportin, sekä painike ikkunan sulkemista varten. Ikkuna aukeaa pääikkunan päälle. Ikkuna on esitetty kuvassa 4. Kuvakaappauksen ottohetkellä, tietokoneessa ei ollut liitettynä yhtäkään sarjaporttia, joten lista näkyy tyhjänä.

4.4.1 Asetusikkunan lataus

```
Private Sub Asetukset_Load(sender As System.Object, e As
System.EventArgs) Handles MyBase.Load

For Each sp As String In My.Computer.Ports.SerialPortNames
  ListBox1.Items.Add(sp)
Next

TextBox1.Text = Sarjaporttinimi

End Sub
```

Edellä esitetty koodi suoritetaan, kun asetusikkuna avataan. Se tarkistaa Windowsin tuntemat sarjaportit ja listaa ne ”Listbox1”-elementtiin näytölle, josta käyttäjä voi klikkaamalla valita käytettävän sarjaportin.

4.4.2 Tallenna-painike

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click

' tarkastetaan että listasta on valittu sarjaportti
If ListBox1.SelectedIndex >= 0 Then
  ' päivitetään sarjaportti tiedostoon

  FileOpen(1,
  "C:\Kulunvalvonta\Asetukset\Sarjaportti.data",
  OpenMode.Output)

  Write(1, ListBox1.SelectedItem)
  FileClose(1)

  TextBox1.Text = ListBox1.SelectedItem
  Sarjaporttinimi = ListBox1.SelectedItem
  porttimuuttuu = True
  MsgBox("Tallennettu!")
Else
  MsgBox("VALITSE SARJAPORTTI ENSIN!")
End If
End Sub
```

Edellä oleva koodi suoritetaan, kun ”Tallenna”-painiketta painetaan. Se tarkistaa ensimmäisenä että listasta on valittu sarjaportti. Mikäli sarjaporttia ei ole valittu käyttäjälle esitetään viestilaatikko, jossa kehotetaan valitsemaan sarjaportti. Mikäli listalta on sarjaportti valittuna, talletetaan valitun sarjaportin nimi asetustiedostoon.

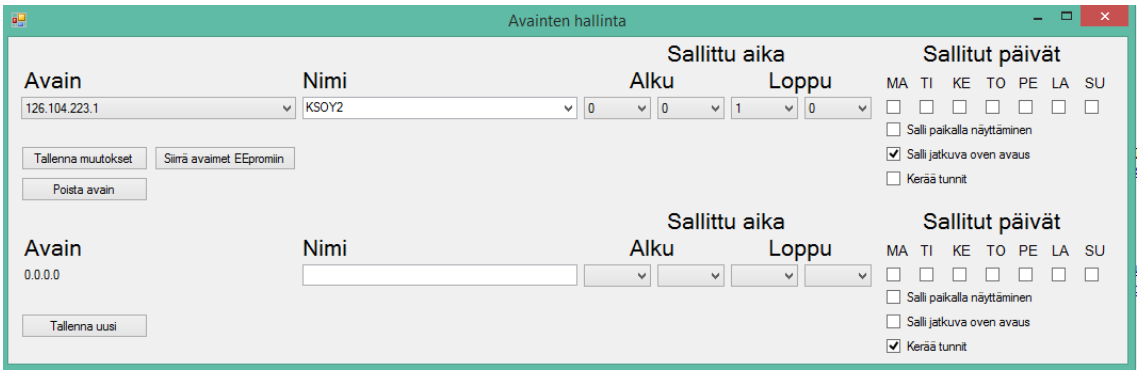
4.4.3 Sulje ikkuna-painike

```
Private Sub Button2_Click(sender As System.Object, e As
System.EventArgs) Handles Button2.Click
  Me.Visible = False
End Sub
```

Edellä esitetty koodi suoritetaan, kun ”Sulje ikkuna”-painiketta painetaan. Se piilottaa asetusikkunan näkyvistä. Ikkunaa ei varsinaisesti suljeta, se

vain piilotetaan näkyvistä. Tämän seurauksena, jos sarjaportti, jota halutaan käyttää, ei ollut käytettävissä, kun ikkuna avattiin ensimmäisen kerran, täytyy koko ohjelma sulkea ja avata uudelleen, jotta sarjaportti tulee näkyviin.

4.5 Avainten hallinta



Kuva 5. Avaintenhallinnan ikkuna

Kuvassa 5 on esitetty avainten hallintaan tarkoitettu ikkuna. Ikkuna on jaettu kahteen riviin. Ensimmäisellä rivillä käsitellään järjestelmän jo tuntemia avaimia. Toisella rivillä lisätään järjestelmään uusia avaimia. Ensimmäisen rivin avain valitaan joko avain- tai nimilistasta. Toisella rivillä käytetään missä tahansa lukijassa viimeksi luettua avainta ja nimi kirjoitetaan tekstikenttään. Muuten asetuksien käsittely on samanlaista molemmilla riveillä. Nimi-kenttään kirjoitetaan nimi joka avaimeen yhdistetään. Samaan nimeen voi olla yhdistettynä useampia avaimia.

”Alku”-kenttiin valitaan listasta tunnit ja minuutit, jotka merkitsevät sallitun sisäänpääsyn alkuaikaa. ”Loppu”-kenttiin valitaan vastaavasti kellonaika, jonka jälkeen avaimella ei enää saa kulkea sisään. Rastitetaan päivien alta ruudut, jotka kertovat minä viikonpäivinä kulkeminen on sallittu. Tämän lisäksi valitaan saako avaimen näyttää paikallaolo-listassa, sallitaanko avaimen käyttö jatkuvasti, kellosta ja päivästä välittämättä, sekä kerätäänkö avaimesta työaikalistoja.

Kun valinnat on tehty, painetaan täytetystä rivistä riippuen joko ”Tallenna muutokset”- tai ”Tallenna uusi”-painiketta, milloin järjestelmä luo tai päivittää avaintiedoston, sekä luo tarvittaessa kansion henkilön työaikaseurantaa varten. Ylemmällä rivillä voi myös poistaa avaimia järjestelmästä, valitsemalla avaimen tai henkilön listasta ja painamalla ”Poista avain”-painiketta. Tämä poistaa avaintiedoston, mutta jättää henkilön työaikaseurannan tiedot järjestelmään. Näin voidaan vaihtaa hukkuneet avaimet uusiin, tai poistaa avaimet järjestelmästä.

”Siirrä avaimet EEPROMiin”-painike siirtää tavut, joista avaimet koostuvat kappaleessa 4.7 esiteltyyn EEPROM-listaukseen. Tämän tarkoituksena oli alun perin sallia avaimien käyttö myös silloin kun tietokone ei jostain syystä ole käytettävissä. Ajatuksesta kuitenkin luovuttiin, koska tietojen talletus ei olisi ollut mahdollista nykyisellä laitteistolla, koska käytetty Arduino ei itsessään sisällä reaaliaikakelloa, eikä riittävästi muistia kirjautu-

mistietojen tallettamiseen. Toiminta kuitenkin jätettiin ohjelmaan, mahdollisia tulevia muutoksia varten.

4.5.1 Avainten hallinta ikkunan lataus

```
Private Sub Avainten_hallinta_Load(sender As System.Object,
e As System.EventArgs) Handles MyBase.Load
        Call lataalista()
End Sub
```

Kun ikkuna ladataan, kutsutaan aliohjelmaa ”lataalista”, joka päivittää olemassa olevat avaimet ja nimet näytön listoihin.

4.5.2 Taustatoiminnot

```
Private Sub Timer1_Tick(sender As System.Object, e As Sys-
tem.EventArgs) Handles Timer1.Tick
        Label1.Text = Trim(Str(luetuttavut(0))) + "." +
Trim(Str(luetuttavut(1))) + "." + Trim(Str(luetuttavut(2)))
+ "." + Trim(Str(luetuttavut(3)))
End Sub
```

4.5.3 Tallenna uusi-painike

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click

If AlkuTunnit2.SelectedIndex >= 0 Then
Else
  MsgBox("Syötä aloitustunnit ensin!")
  Exit Sub
End If

If Alkuminuutit2.SelectedIndex >= 0 Then
Else
  MsgBox("Syötä aloitusminuutit ensin!")
  Exit Sub
End If

If LoppuTunnit2.SelectedIndex >= 0 Then
Else
  MsgBox("Syötä lopetustunnit ensin!")
  Exit Sub
End If

If LoppuMinuutit2.SelectedIndex >= 0 Then
Else
  MsgBox("Syötä lopetusminuutit ensin!")
  Exit Sub
End If

If UusiNimi.Text = "" Then
  MsgBox("Nimi ei saa olla tyhjä!")
  Exit Sub
End If
```

Painettaessa ”Tallenna uusi”-painiketta, tarkistetaan ensin, onko pakollisissa kentissä arvoja. Pakollisia kenttiä ovat aloitus- ja lopetusajat, sekä nimi.

```
FileOpen(1, "C:\Kulunvalvonta\Avaimet\"
+ Trim(Label1.Text), OpenMode.Output)

PrintLine(1, UusiNimi.Text)
PrintLine(1, AlkuTunnit2.SelectedIndex)
PrintLine(1, Alkuminuutit2.SelectedIndex)
PrintLine(1, LoppuTunnit2.SelectedIndex)
PrintLine(1, LoppuMinuutit2.SelectedIndex)
PrintLine(1, MA2.Checked)
PrintLine(1, TI2.Checked)
PrintLine(1, KE2.Checked)
PrintLine(1, TO2.Checked)
PrintLine(1, PE2.Checked)
PrintLine(1, LA2.Checked)
PrintLine(1, SU2.Checked)
PrintLine(1, SalliNäyttö2.Checked)
PrintLine(1, SalliAvaus2.Checked)
PrintLine(1, KerääTunnit2.Checked)
FileClose(1)
```

Kun pakolliset kentät on täytetty, talletetaan avaintiedosto. Avaintiedoston rakenne on esitetty luvussa 4.12.2.

```
If KerääTunnit2.Checked = True Then
  If Not System.IO.Directory.Exists
  ("C:\Kulunvalvonta\Henkilöt\" + Trim(UusiNimi.Text)) Then
    System.IO.Directory.CreateDirectory
    ("C:\Kulunvalvonta\Henkilöt\" + Trim(UusiNimi.Text))
  End If
End If
```

Mikäli avaimesta kerätään työaikatiedot, tarkistetaan onko avaimeen liitetyllä henkilöllä vielä tiedonkeruukansiota ja luodaan kansio, mikäli sitä ei ole olemassa.

```
If SalliNäyttö2.Checked = True Then
  FileOpen(1, "C:\Kulunvalvonta\Henkilöt\Paikalla\"
  + Trim(UusiNimi.Text), OpenMode.Output)

  PrintLine(1, False, ",", Now.Date, ",",
  Trim(Str(Now.Hour)) + ":" + Trim(Str(Now.Minute)) +
  ":" + Trim(Str(Now.Second)))

   FileClose(1)
End If
```

Seuraavaksi tarkistetaan näytetäänkö avaimen paikallaolotieto. Mikäli tieto näytetään luodaan paikallaolotiedosto. Mikäli henkilöllä oli jo paikallaolotiedosto, se ylikirjoitetaan. Henkilö kirjataan ei paikallaolevaksi. Paikallaolotiedoston rakenne on esitetty luvussa 4.12.5.

```
Call lataalista()
End Sub
```

Viimeisenä avaimen luomisen jälkeen, päivitetään nimi- ja avainlistat ikkunan ylempien rivien valikoihin. Tämä tehdään kutsumalla ”lataalista”-aliohjelmaa.

4.5.4 Nimi- ja avainlistojen päivitys

Ikkunan ylemmän rivin nimi- ja avainlistat päivitetään kutsumalla aliohjelmaa ”lataalista”. Seuraavaksi on esitetty aliohjelman lähdekoodi, sekä toiminta.

```
Sub lataalista()
' make a reference to a directory
Dim di As New IO.DirectoryInfo("c:\Kulunvalvonta\Avaimet\")
Dim diar1 As IO.FileInfo() = di.GetFiles()
Dim dra As IO.FileInfo

'list the names of all files in the specified directory

Avaimet.Items.Clear()
For Each dra In diar1
  Avaimet.Items.Add(dra)
Next
```

Ensimmäisenä listataan tiedostot kansiosta ”c:\Kulunvalvonta\Avaimet\”. Jokaisen tiedoston nimi on järjestelmään tallennettu avain. Tiedostolistaus lisätään riveinä alasvetovalikkoon nimeltä ”Avaimet”.

```
Dim nimi As String = ""
Nimet.Items.Clear()
For Each item As Object In Avaimet.Items
  FileOpen(1, "c:\Kulunvalvonta\Avaimet\"
   + Trim(item.ToString()), OpenMode.Input)

  Input(1, nimi)
  Nimet.Items.Add(nimi)
  FileClose(1)
Next
```

Kun avaimet on lisätty omaan valikkoonsa, käydään valikossa olevat avaimet läpi. Käyttäen avainta tiedostonnimenä, avataan jokainen avaintiedosto ja luetaan tiedoston ensimmäinen rivi, johon on talletettu avaimeen liitetty nimi. Nimet lisätään ”Nimet”-nimiseen alasvetovalikkoon, jokainen omana rivinään.

```
Nimet.SelectedIndex = 0
Avaimet.SelectedIndex = 0

End Sub
```

Kun listat on päivitetty, muutetaan kummassakin listassa ensimmäinen olemassa oleva objekti valituksi.

4.5.5 Valittu nimi muuttuu

Kun käyttäjä vaihtaa valittua nimeä ”Nimet”-alasvetovalikossa, päivitetään myös valittu avain vastaamaan valitun henkilön avainta.

```
Private Sub Nimet_SelectedIndexChanged(sender As Sys-
tem.Object, e As System.EventArgs) Handles Ni-
met.SelectedIndexChanged

  Avaimet.SelectedIndex = Nimet.SelectedIndex

End Sub
```

Jokaisella alasvetovalikon objektilla on indeksi-numero. Nimet on ladattu avaintiedostoista siinä järjestyksessä, kun ne ovat avain-valikossa, joten indeksi-numerot vastaavat toisiaan. Kun nimet-valikon indeksi muuttuu, muutetaan myös avain-valikon valittu indeksi samaan arvoon kuin nimet-valikossa.

4.5.6 Valittu avain muuttuu

```
Private Sub Avaimet_SelectedIndexChanged(sender As Sys-
tem.Object, e As System.EventArgs) Handles Avai-
met.SelectedIndexChanged

  Nimet.SelectedIndex = Avaimet.SelectedIndex
```

Kun käyttäjä valitsee listasta käsiteltäväksi eri avaimen, päivitetään myös nimi-valikon valinta vastaamaan valittua avainta. Tämä aliohjelma suoritetaan myös, kun käyttäjä valitsee eri nimen käsiteltäväksi, koska se muuttaa molempien valikoiden valittua indeksiä. Tämä ei aiheuta päättymätöntä silmukkaa, koska vaikka valittuun indeksiin ohjataan arvo, aina kun kumpi tahansa näistä aliohjelmista suoritetaan, ei indeksi muutu kuin kerran, koska siihen asetetaan sama arvo, mikä siinä oli jo.

```
  FileOpen(1, "c:\Kulunvalvonta\Avaimet\"
  + Trim(Avaimet.SelectedItem.ToString), OpenMode.Input)

  Input(1, Nimet.Text)
  Input(1, AlkuTunnit1.SelectedIndex)
   Input(1, AlkuMinuutit1.SelectedIndex)
   Input(1, LoppuTunnit1.SelectedIndex)
   Input(1, LoppuMinuutit1.SelectedIndex)
   Input(1, MA1.Checked)
   Input(1, TI1.Checked)
   Input(1, KE1.Checked)
   Input(1, TO1.Checked)
   Input(1, PE1.Checked)
   Input(1, LA1.Checked)
   Input(1, SU1.Checked)
   Input(1, SalliNäyttö1.Checked)
   Input(1, SalliAvaus1.Checked)
   Input(1, KerääTunnit1.Checked)

   FileClose(1)
End Sub
```

Seuraavaksi avataan avain-tiedosto ja luetaan sinne talletetut asetukset. Asetukset luetaan suoraan ikkunan ylemmällä rivillä oleviin asetuskenttiin.

4.5.7 Tallenna muutokset-painike

```
Private Sub Button2_Click(sender As System.Object, e As
System.EventArgs) Handles Button2.Click
  Dim roska As Integer
  roska = Avaimet.SelectedIndex

  FileOpen(1, "C:\Kulunvalvonta\Avaimet\"
  + Trim(Avaimet.SelectedItem.ToString), OpenMode.Output)

  PrintLine(1, Nimet.Text)
  PrintLine(1, AlkuTunnit1.SelectedIndex)
  PrintLine(1, AlkuMinuutit1.SelectedIndex)
  PrintLine(1, LoppuTunnit1.SelectedIndex)
  PrintLine(1, LoppuMinuutit1.SelectedIndex)
  PrintLine(1, MA1.Checked)
  PrintLine(1, TI1.Checked)
  PrintLine(1, KE1.Checked)
  PrintLine(1, TO1.Checked)
  PrintLine(1, PE1.Checked)
  PrintLine(1, LA1.Checked)
  PrintLine(1, SU1.Checked)
  PrintLine(1, SalliNäyttö1.Checked)
  PrintLine(1, SalliAvaus1.Checked)
  PrintLine(1, KerääTunnit1.Checked)
  FileClose(1)

  If KerääTunnit1.Checked = True Then
    If Not System.IO.Directory.Exists
    ("C:\Kulunvalvonta\Henkilöt\" + Trim(Nimet.Text)) Then

      System.IO.Directory.CreateDirectory
      ("C:\Kulunvalvonta\Henkilöt\" + Trim(Nimet.Text))
    End If
  End If

  If SalliNäyttö1.Checked = True Then
    FileOpen(1, "C:\Kulunvalvonta\Henkilöt\Paikalla\"
    + Trim(Nimet.Text), OpenMode.Output)

    PrintLine(1, False, ",", Now.Date,
    ",", Trim(Str(Now.Hour)) + ":" + Trim(Str(Now.Minute))
    + ":" + Trim(Str(Now.Second)))

    FileClose(1)
  Else
    If System.IO.File.Exists
    ("C:\Kulunvalvonta\Henkilöt\Paikalla\"
    + Trim(Nimet.Text)) Then

    System.IO.File.Delete
    ("C:\Kulunvalvonta\Henkilöt\Paikalla\"
    + Trim(Nimet.Text))
  End If
End If
Call lataalista()
Avaimet.SelectedIndex = roska
End Sub
```

Avaimen tietojen päivittäminen eroaa uuden avaimen luomisesta kahdella tapaa. Ensimmäisenä, avaintiedoston nimi ei ole viimeksi järjestelmässä luettu avain, vaan Avaimet-listasta valittu avain. Toinen ero liittyy paikal-

laolotiedon näyttämiseen. Mikäli paikallaolotietoa ei haluta näyttää, mutta paikallaolotiedosto on paikalla, se poistetaan. Muilta osin tietojen päivitys on identtinen toiminta uuden avaimen tallentamisen kanssa.

4.5.8 Siirrä avaimet EEPROMiin-painike

Siirrä avaimet EEPROMiin-painikkeen tarkoitus on muuttaa järjestelmään talletettujen avaimien nimet merkkijonoista numeerisiksi tavuiksi ja muodostaa niistä taulukko mikrokontrollerin EEPROM-piirille talletettavaksi. Vaikka kyseistä tietoa ei otettukaan käyttöön mikrokontrollerin käytettäväksi yhteydettömässä tilassa, jätettiin toiminto Windows-ohjelmaan, mahdollisia tulevaisuuden muutoksia ajatellen.

```
Private Sub Button3_Click(sender As System.Object, e As
System.EventArgs) Handles Button3.Click
  Dim roska As String = ""
  Dim m, o As Byte
  Dim tavut(4) As Byte
  Dim bytes() As Byte     'Byte array
  Dim enc As New System.Text.ASCIIEncoding
```

Aluksi määritellään aliohjelman tarvitsemat paikalliset muuttujat. Tämän jälkeen aloitetaan silmukka, jossa käsitellään jokainen listasta löytyvä avain.

```
  For Each item As Object In Avaimet.Items
    roska = ""
    ReDim tavut(4)
    m = 0
    bytes = enc.GetBytes(Trim(item.ToString()))
    For n = 0 To bytes.Length - 1
      If Not IsNumeric(Chr(bytes(n))) Then
        tavut(m) = Convert.ToByte(roska)
        roska = ""
        m = m + 1
      Else
        roska = roska + Trim(Chr(bytes(n)))
      End If
    Next
```

Edellä esitetty koodi purkaa neljä pisteellä erotettua lukua merkkijonosta neljään numeeriseen tavuun.

```
    tavut(m) = Convert.ToByte(roska)
    EEPROM(o) = tavut(0)
    EEPROM(o + 1) = tavut(1)
    EEPROM(o + 2) = tavut(2)
    EEPROM(o + 3) = tavut(3)
    o = o + 4
  Next
```

Seuraavaksi puretut tavut siirretään taulukkoon ”EEPROM” juoksevassa järjestyksessä ja lasketaan muuttujan ”o” avulla tavujen paikkaa taulukossa.

```
 MsgBox("Vain 63 ensimmäistä avainta siirtyy
 EEPROM-tietoihin." + vbNewLine
```

```
 + "Siirrä ja tallenna EEPROMtiedot,
 jotta ne tulevat voimaan")

 Piirin_tietojen_varmuuskopiointi.Visible = True
End Sub
```

Lopuksi ilmoitetaan käyttäjälle, että vain 63 ensimmäistä avainta voidaan tallettaa kontrollerin EEPROMiin, sekä että siirto täytyy tehdä EEPROMin varmuuskopiointi-ikkunassa. Tiedosto EEPROMin tiedoista täytyy myös erikseen tallentaa samassa ikkunassa. Kun käyttäjä kuittaa ilmoituksen luetuksi, kyseinen sivu tuodaan näkyviin.

4.5.9 Poista avain-painike

```
Private Sub Button4_Click(sender As System.Object, e As
System.EventArgs) Handles Button4.Click
  If MsgBox("Avain poistetaan, oletko varma?",
  MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then
```

Kun käyttäjä painaa ”Poista avain”-painiketta, tarkistetaan käyttäjältä onko hän varma, haluavansa poistaa avaimen. Mikäli käyttäjä valitsee ei, loppuu aliohjelman suoritus. Mikäli kyllä, suoritus jatkuu seuraavaksi esitetyn mukaisesti.

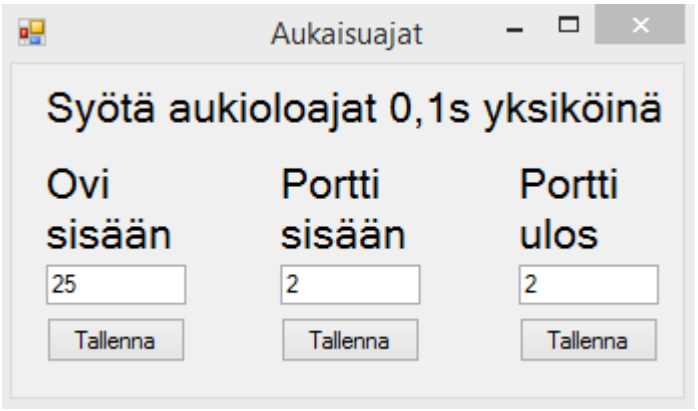
```
    If System.IO.File.Exists
    ("C:\Kulunvalvonta\Henkilöt\Paikalla\"
    + Trim(Nimet.Text)) Then

      System.IO.File.Delete
      ("C:\Kulunvalvonta\Henkilöt\Paikalla\"
      + Trim(Nimet.Text))
    End If
```

Mikäli avaimen nimeen on sidottu paikallaolotiedosto, se poistetaan ensimmäisenä.

```
    If System.IO.File.Exists("C:\Kulunvalvonta\Avaimet\"
    + Trim(Avaimet.Text)) Then

      System.IO.File.Delete("C:\Kulunvalvonta\Avaimet\"
      + Trim(Avaimet.Text))

    End If
    Call lataalista()
  End If
End Sub
End Class
```

Tämän jälkeen tarkistetaan, että avaintiedosto on olemassa ja poistetaan se. Päivitetään listat näytölle ja lopetetaan aliohjelma.

4.6 Aukaisuajat



Kuva 6. Aukaisupulssien pituuden asetus

”Aukaisuajat”-ikkunassa asetetaan porteille annettavan aukaisupulssin pituus. Ikkuna on esitetty kuvassa 6. Pituus tulee määrittää järjestelmään liitettyjen laitteiden mukaan. Tämän työn tapauksessa sekä oven että kääntöportin lukot aukeavat pulssilla ja pysyvät auki riittävän kauan, että käyttäjä ehtii kulkea haluamansa laitteen ohi, joten hyvin lyhyet pulssit riittävät. Jos käytetäisiin laitetta joka pysyy auki, vain pulssin ollessa päällä, tulisi määrittää aika, joka käyttäjältä menee koodin lukemisesta oven tai portin avaamiseen ja määrittää aukioloajat sellaisiksi, että käyttäjä ehtii avata oven tai portin pulssin aikana. Portin molemmille suunnille ja ovelle määritellään omat aikansa. Pituutta määritellessä yksikkö on sekunnin kymmenesosa. Kun ”Tallenna”-painiketta painetaan, tallennetaan kyseisen laitteen asetus tiedostoon ja siirretään mikrokontrollerille. Koska mikrokontrollerille kirjoitettava tieto on tavu, on asetuksen maksimiarvo 255.

4.6.1 Aikaisuajat-ikkunan lataus

```
Private Sub Aukaisuajatasetus_Load(sender As System.Object,
e As System.EventArgs) Handles MyBase.Load

FileOpen(1,     "C:\Kulunvalvonta\Asetukset\Aukiaika1.data",
OpenMode.Input)

Input(1, TextBox1.Text)

FileClose(1)

FileOpen(1,     "C:\Kulunvalvonta\Asetukset\Aukiaika2.data",
OpenMode.Input)

Input(1, TextBox2.Text)

FileClose(1)

FileOpen(1,     "C:\Kulunvalvonta\Asetukset\Aukiaika3.data",
OpenMode.Input)

Input(1, TextBox3.Text)

FileClose(1)
```

```
End Sub
```

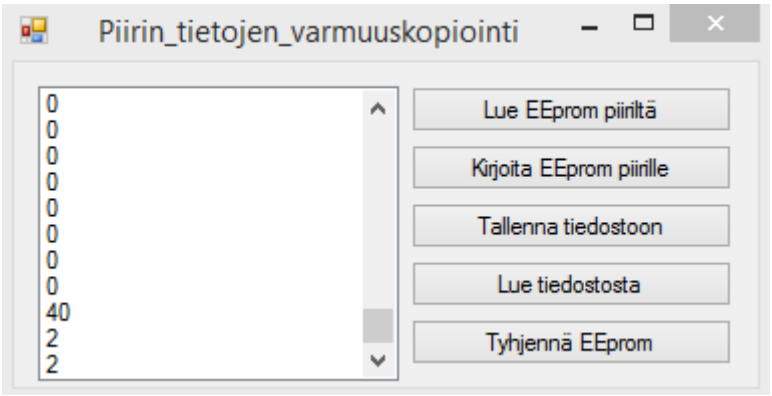
Kun ”Aikaisuajat”-ikkuna ladataan, luetaan jokaisen aikaisuajan asetustiedostosta järjestelmään talletetut arvot ja asetetaan ne niille tarkoitettuun tekstikenttiin, joissa käyttäjä voi tarvittaessa muuttaa niitä.

4.6.2 Tallenna-painikkeet

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click

FileOpen(1,     "C:\Kulunvalvonta\Asetukset\Aukiaika1.data",
OpenMode.Output)

  Print(1, TextBox1.Text)

  FileClose(1)

  kirjoitatavut(3) = 1
  kirjoitatavut(5) = TextBox1.Text
  kirjoitatavut(4) = 3
End Sub
```

”Tallenna”-painiketta painettaessa talletetaan ensin asetettu arvo asetustiedostoon. Tämän jälkeen muokataan lähetettävä viestikehys antamaan mikrokontrollerille komento, tallettaa avauspulssin pituus taulukon 4 mukaisesti. Painikkeita on kolme ja ne suorittavat muuten saman koodin, paitsi että numero tiedostonimessä ja kolmannessa tavussa muuttuu, valitun kohteen mukaisesti. Tiedoston sisältö on kuvattu luvussa 4.12.1. Yksi vastaa oven aikaa, kaksi portin sisäänpäin aikaa ja kolme portin ulospäin aikaa. Luvussa 4.3.7 esitetty liikennöinti viestittää käyttäjälle viestilaatikolla, kun tiedot on talletettu.

4.7 EEPROM



Kuva 7. EEPROM tietojen talletus ja lataus

Kuvassa 7 on esitetty ikkuna, joka on tarkoitettu mikrokontrolleriin talletettujen tietojen varmuuskopiointiin. Mikrokontrollerin EEPROM muistin sisältö voidaan lukea piiriltä tai kirjoittaa piirille, sekä tallettaa tiedostoon, tai lukea tiedostosta. EEPROM piirin tiedot voidaan myös nollata. Viimeiset kolme tavua tiedoista ovat oven ja portin avauspulssien pituudet. Niitä

edeltäviä tavuja ei tällä hetkellä käytetä mihinkään ja ne ovat ohjelmassa mukana mahdollisia tulevaisuuden muutoksia varten.

4.7.1 Piirin tietojen varmuuskopiointi-ikkunan lataaminen

```
Private Sub Piirin_tietojen_varmuuskopiointi_Load(sender As
System.Object, e As System.EventArgs) Handles MyBase.Load

ListBox1.Items.Clear()
For n = 0 To 255
  ListBox1.Items.Add(EEPROM(n))
Next
End Sub
```

Kun ikkuna ladataan, täytetään ikkunan lista EEPROM-muuttujaan talletetuilla arvoilla.

4.7.2 Lue EEPROM piiriltä ja Kirjoita EEPROM piirille

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click

If lueEEPROM(0) = True Then
  MsgBox("Luenta on jo käynnissä")
  Exit Sub
End If
If kirjoitaEEPROM(0) = True Then
  MsgBox("Kirjoitus on jo käynnissä")
  Exit Sub
End If
lueEEPROM(0) = True
End Sub
```

Kun ”Lue EEPROM piiriltä”-painiketta painetaan, tarkistaa ohjelma ensin, onko piirin luku tai kirjoitus jo käynnissä. Mikäli luku tai kirjoitus on käynnissä, ilmoitetaan siitä käyttäjälle viestilaatikolla. Mikäli luku tai kirjoitus ei ole käynnissä, asetetaan muuttuja ”lueEEPROM(0)”-muuttujan arvoksi tosi, mikä käynnistää piirin muistien lukemisen mikrokontrollerilta. Luku hoidetaan liikennöinnin yhteydessä ja on esitetty luvussa 4.3.7. Kun luku tai kirjoitus on valmis, tiedotetaan siitä käyttäjälle luvussa 4.7.6 kuvatulla tavalla.

”Kirjoita EEPROM piirille”-painike sisältää muuten identtisen koodin, mutta todeksi asetettava muuttuja on ”kirjoitaEEPROM(0)”, eikä ”lueEEPROM(0)”.

4.7.3 Tallenna tiedostoon

```
Private Sub Button3_Click(sender As System.Object, e As
System.EventArgs) Handles Button3.Click

FileOpen(1, "c:\Kulunvalvonta\Asetukset\EEPROM.data",
OpenMode.Output)

For Each item As Object In ListBox1.Items
```

```
  PrintLine(1, Trim(item.ToString()))
Next
FileClose(1)

MsgBox("EEPROM-tiedosto tallennettu!")
End Sub
```

Painettaessa ”Tallenna tiedostoon”-painiketta avataan EEPROMille tarkoitettu datatiedosto käsiteltäväksi uutena tiedostona ja kirjoitetaan ikkunan listan sisältö sinne. Tiedoston sisältö on kuvailtu luvussa 4.12.1. Kun tallennus on valmis, ilmoitetaan siitä käyttäjälle viestilaatikolla.

4.7.4 Lue tiedostosta

```
Private Sub Button4_Click(sender As System.Object, e As
System.EventArgs) Handles Button4.Click

FileOpen(1, "C:\Kulunvalvonta\Asetukset\EEPROM.data", Open-
Mode.Input)

For n = 0 To 255
  Input(1, EEPROM(n))
Next

FileClose(1)
```

Kun ”Lue tiedostosta”-painiketta painetaan, ladataan ensin EEPROMille tarkoitetun datatiedoston sisältö ”EEPROM”-muuttujaan.

```
ListBox1.Items.Clear()
For n = 0 To 255
  ListBox1.Items.Add(EEPROM(n))
Next
MsgBox("EEPROM-tiedosto ladattu!")
End Sub
```

Kun lataus on tehty, tyhjennetään näytön lista ja siirretään muuttujaan ”EEPROM” asetetut arvot listaan, jokainen omalle rivilleen. Kun lista on täytetty, ilmoitetaan siitä käyttäjälle viestilaatikolla.

4.7.5 Tyhjennä EEPROM

```
Private Sub Button5_Click(sender As System.Object, e As
System.EventArgs) Handles Button5.Click

If lueEEPROM(0) = True Then
  MsgBox("Luenta on jo käynnissä")
  Exit Sub
End If

If kirjoitaEEPROM(0) = True Then
  MsgBox("Kirjoitus on jo käynnissä")
  Exit Sub
End If

For n = 0 To 255
  EEPROM(n) = 0
Next
```

```
kirjoitaEEPROM(0) = True
End Sub
```

”Tyhjennä EEPROM” painike toimii kuten ”Kirjoita EEPROM piirille”-painike, sillä erotuksella, että ennen kuin kirjoituksen käynnistävää bittiä asetetaan todeksi, asetetaan ”EEPROM”-muuttujaan talletetut arvot nolliksi.

4.7.6 Ikkunan taustatoiminnot

Ikkunan taustatoiminnat hoidetaan ajastimella, joka tarkastaa onko mikrokontrollerille tehty kirjoitus, tai siltä tehty luku valmistunut.

```
Private Sub Timer1_Tick(sender As System.Object, e As System.EventArgs) Handles Timer1.Tick

If lueEEPROM(1) = True Then
  lueEEPROM(1) = False
  ListBox1.Items.Clear()
  For n = 0 To 255
    ListBox1.Items.Add(EEPROM(n))
  Next
  MsgBox("Luku valmis!")
End If
```

Mikäli liikennöinti on luvussa 4.3.7 kuvatulla tavalla asettanut muuttujan ”lueEEPROM(1)” arvon todeksi, asetetaan arvo epätodeksi, tyhjennetään näytön lista ja täytetään se uudelleen mikrokontrollerilta luetuilla arvoilla. Tämän jälkeen käyttäjälle ilmoitetaan toiminnan valmistuminen viestilaatikolla.

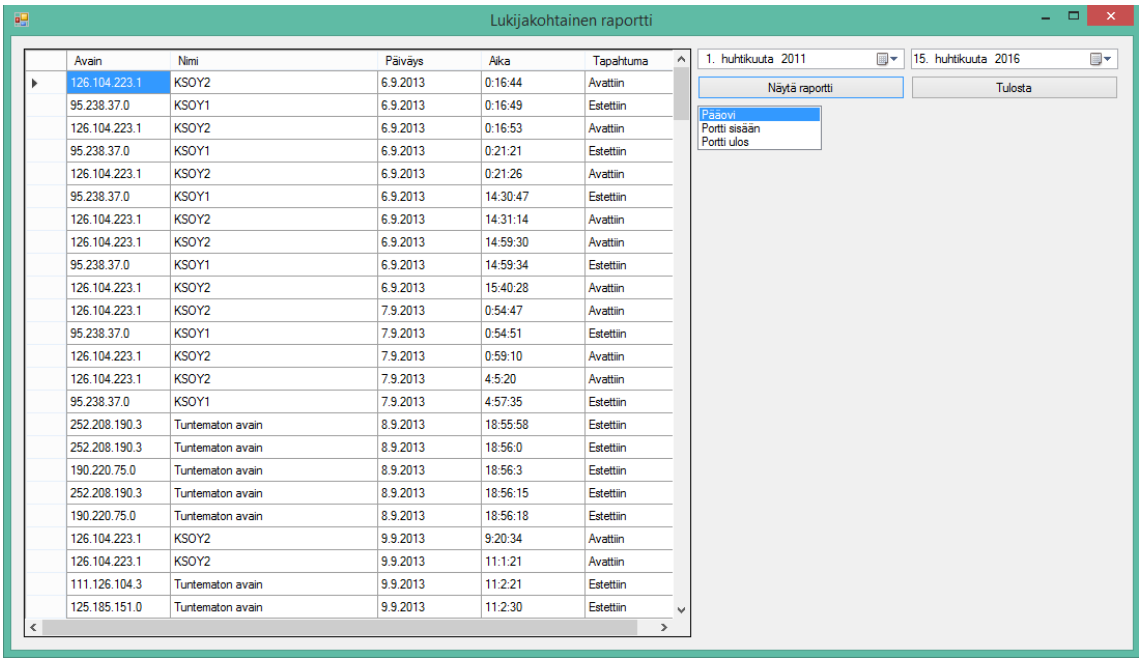
```
If kirjoitaEEPROM(1) = True Then
  kirjoitaEEPROM(1) = False
  MsgBox("Kirjoitus valmis!")
End If
End Sub
```

”kirjoitaEEPROM(1)”-muuttujaa käsitellään kuten ”lueEEPROM(1)”-muuttujaa, sillä erotuksella että näytön listaa ei päivitetä, kun kirjoitus on valmis.

4.8 Raportit

Ohjelma tuottaa kolmen kaltaisia raportteja. Ovikohtaiset raportit valvomaan mahdollisia luvattomia avausyrityksiä, tuntilistat kertomaan käyttäjien työajat palkanmaksun perusteeksi, sekä reaaliaikaisen paikallaolijalistan, josta myös käyttäjät voivat tarkastaa onnistuiko kirjautuminen sisään tai ulos. Jokaisen raporttityypin lukeminen tapahtuu omassa ikkunassaan.

4.9 Ovikohtainen raportti



Kuva 8. Ovikohtainen raportti

Ovikohtaisen raportin ikkunassa on taulukko, jossa itse raportti näytetään, kentät raportin aloitus ja lopetuspäiville, napit raportin näyttämiseen, sekä tulostamiseen paperille sekä lista, josta valitaan lukija, jolta raportti koostetaan. Ikkuna on nähtävissä kuvassa 8.

Aloitus ja lopetusaika valitaan klikkaamalla haluttua kenttää, milloin ohjelma tuo näytille kalenterin, josta päivä valitaan. Listasta valitaan haluttu lukija ja painetaan ”Näytä raportti”-painiketta, milloin ohjelma tuo raportin taulukkoon. Itse raportissa näytetään luettu koodi, koodiin sidottu nimi, lukemisen päiväys ja kellonaika, sekä tieto ovattiinko ovea tai porttia. Taulukossa näkyvä raportti voidaan tulostaa paperille ”Tulosta”-painikkeella, joka tuo esiin tulostuksen esikatselun, josta voidaan jatkaa tulostukseen. Näytöllä olevan raportin voi kopioida esimerkiksi Microsoft Exceliin.

4.9.1 Ikkunan käyttämät muuttujat

```
Dim mRow As Integer = 0
```

Ikkunalla on vain yksi muuttuja jonka tarvitsee säilyä aliohjelmien ulkopuolella. Tätä muuttujaa käytetään muistamaan käsitellyt rivit tulostaessa raporttia.

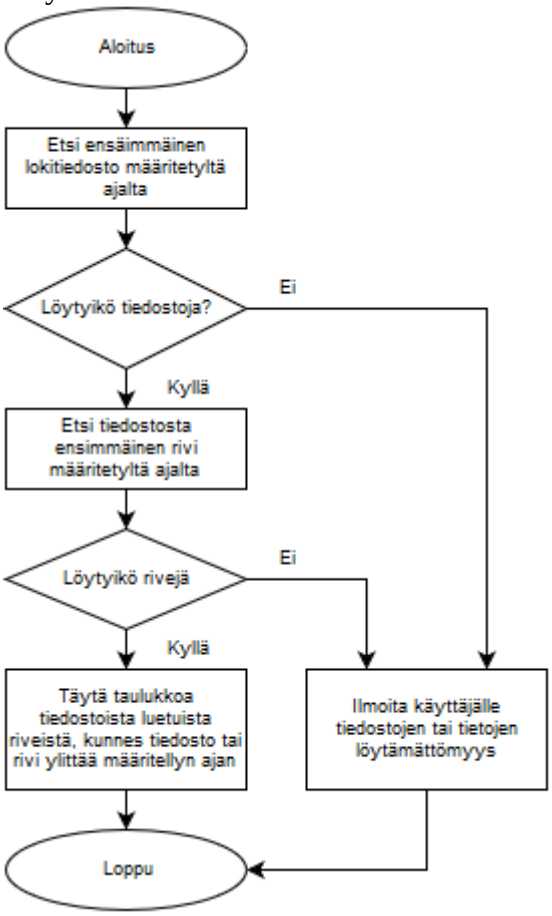
4.9.2 Ikkunan lataaminen

```
Private Sub Pääovi_raportti_Load(sender As System.Object, e
As System.EventArgs) Handles MyBase.Load
   ListBox1.SelectedIndex = 0
 End Sub
```

Kun ikkuna ladataan, muutetaan lukijalistan ensimmäinen objekti valituksi. Ensimmäinen objekti listassa on pääovi. Tämä ei olisi välttämätöntä, mutta varmistaa, ettei käyttäjä yritä luoda raporttia valitsematta lukijaa.

4.9.3 Raportin koostaminen

Raportti koostetaan, kun ”Näytä raportti”-painiketta painetaan. Raportti koostetaan omassa aliohjelmassaan. Raportin koostamisen periaate on esitetty vuokaaviona kuviossa 3.



Kuvio 3. Raportin koostaminen

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
  Dim päivä2 As Date
  Dim vuosi, kuukausi, päivä As Integer
  Dim avain="", nimi="", päiväys, kello As String
  Dim pv, kk, v, roska As Integer
  Dim avaus As Boolean
  Dim kansio As String
```

Aliohjelman aluksi määritellään aliohjelman käyttämät muuttujat.

```
If ListBox1.SelectedIndex = 1 Then
  kansio = "C:\Kulunvalvonta\Oviloki2\"
ElseIf ListBox1.SelectedIndex = 2 Then
  kansio = "C:\Kulunvalvonta\Oviloki3\"
Else
  kansio = "C:\Kulunvalvonta\Oviloki\"
End If
```

Tarkistetaan mikä lukija on valittuna ja asetetaan kansio, mistä raportti luetaan valinnan mukaisesti.

```
DataGridView1.Rows.Clear()

päivä2 = DateTimePicker1.Value

vuosi = päivä2.Year
kuukausi = päivä2.Month
päivä = päivä2.Day
```

Tyhjennetään taulukko ja luetaan muuttujiin valittu aloituspäivä.

```
If Not System.IO.File.Exists(Trim(kansio)
+ Trim(Str(vuosi)) + "." + Trim(Str(kuukausi))) Then

  Do
    vuosi = päivä2.Year
    kuukausi = päivä2.Month
    päivä = päivä2.Day

    If päivä2 > Now Or päivä2 > DateTimePicker2.Value Then
      MsgBox("Tiedostoja ei löydy!")
      Exit Sub
    End If

    päivä2 = päivä2.AddDays(1)
  Loop Until System.IO.File.Exists(Trim(kansio)
  +   Trim(Str(vuosi)) + "." + Trim(Str(kuukausi)))

End If
```

Tarkistetaan löytyykö valittua ajanjaksoa lokitiedostoista. Lokitiedostojen kansio määritettiin aliohjelman alussa. Tiedostojen nimi koostuu vuodesta ja kuukaudesta pisteellä erotettuna. Tiedostoa etsitään lisäämällä aloituspäivään yksi päivä, kunnes aloittava lokitiedosto löytyy, tai laskemalla saatu aika ylittää nykyhetken (muuttuja ”Now”) tai raporttiin valitun lopetushetken. Mikäli tiedosto löytyi, jatketaan koodin suorittamista eteenpäin, mikäli ei, ilmoitetaan siitä käyttäjälle viestilaatikolle ja lopetetaan aliohjelman suoritus.

```
Do
  vuosi = päivä2.Year
  kuukausi = päivä2.Month
  If System.IO.File.Exists(Trim(kansio)
  + Trim(Str(vuosi)) + "." + Trim(Str(kuukausi))) Then

    FileOpen(1, Trim(kansio) + Trim(Str(vuosi)) + "."
    + Trim(Str(kuukausi)), OpenMode.Input)

    Input(1, avain)
    Input(1, nimi)
    Input(1, pv)
    Input(1, kk)
    Input(1, v)
    päiväys = Trim(Str(pv)) + "." + Trim(Str(kk)) +
    "." + Trim(Str(v))

    Input(1, roska)
    kello = Trim(Str(roska)) + ":"
```

```
    Input(1, roska)
    kello = kello + Trim(Str(roska)) + ":"
    Input(1, roska)
    kello = kello + Trim(Str(roska))
    Input(1, avaus)
```

Kun halutulta ajalta on löytynyt tiedosto, aloitetaan silmukka, jossa käydään läpi tiedoston sisältö. Tiedoston rakenne on kuvailtu luvussa 4.12.3. Ensin avataan tiedosto ja luetaan sieltä ensimmäinen rivi.

```
If pv < DateTimePicker1.Value.Day And
kk = DateTimePicker1.Value.Month Then
  Do
  Input(1, avain)
  Input(1, nimi)
  Input(1, pv)
  Input(1, kk)
  Input(1, v)

  päiväys = Trim(Str(pv)) + "." + Trim(Str(kk)) + "."
  + Trim(Str(v))

  Input(1, roska)
  kello = Trim(Str(roska)) + ":"
  Input(1, roska)
  kello = kello + Trim(Str(roska)) + ":"
  Input(1, roska)
  kello = kello + Trim(Str(roska))
  Input(1, avaus)
Loop Until pv => DateTimePicker1.Value.Day Or EOF(1)
```

Tarkistetaan onko ensimmäinen rivi aikaisemmalta ajalta kuin raportin asetettu aloitusaika. Mikäli rivi on aikaisemmalta ajalta, luetaan rivejä, kunnes tiedosto loppuu tai löytyy rivi, joka on halutulta ajalta.

```
If pv >= DateTimePicker1.Value.Day Then
  If avaus = True Then
    DataGridView1.Rows.Add
    (avain, nimi, päiväys, kello, "Avattiin")
  Else
    DataGridView1.Rows.Add
    (avain, nimi, päiväys, kello, "Estettiin")
  End If
End If
```

Seuraavaksi tarkastellaan viimeksi luettua riviä, mikäli se on halutulta ajalta lisätään rivi taulukkoon. Tiedostosta luettu muuttuja ”avaus” on muodoltaan boolean ja sen perusteella kirjoitetaan tieto avattiinko haluttu kohde, vai estettiinkö kulku.

```
ElseIf pv > DateTimePicker2.Value.Day
And kk = DateTimePicker2.Value.Month Then
  FileClose(1)
  Exit Sub
```

Mikäli ensimmäinen luettu rivi oli myöhemmältä ajalta kuin asetettu lopetusaika, suljetaan tiedosto ja lopetetaan aliohjelma. Tällöin taulukko jää tyhjäksi.

```
Else
  If avaus = True Then
    DataGridView1.Rows.Add
    (avain, nimi, päiväys, kello, "Avattiin")
  Else
    DataGridView1.Rows.Add
    (avain, nimi, päiväys, kello, "Estettiin")
  End If
End If
```

Mikäli ensimmäinen luettu rivi ei ollut liian aikaiselta tai liian myöhäiseltä ajalta, on sen oltava valitulta ajalta, milloin rivi lisätään taulukkoon.

```
If Not EOF(1) Then

Do
  Input(1, avain)
  Input(1, nimi)
  Input(1, pv)
  Input(1, kk)
  Input(1, v)

  päiväys = Trim(Str(pv)) + "." + Trim(Str(kk))
  + "." + Trim(Str(v))

  Input(1, roska)
  kello = Trim(Str(roska)) + ":"
  Input(1, roska)
  kello = kello + Trim(Str(roska)) + ":"
  Input(1, roska)
  kello = kello + Trim(Str(roska))
  Input(1, avaus)
```

Mikäli tiedostoa ei oltu luettu loppuun, aloitetaan silmukka jossa käsitellään tiedosto loppuun. Ensimmäisenä luetaan seuraava rivi.

```
  If pv > DateTimePicker2.Value.Day
  And kk = DateTimePicker2.Value.Month
  And v = DateTimePicker2.Value.Year Then Exit Do
```

Tarkastetaan onko rivi myöhemmältä ajalta kuin raportin asetettu lopetusaika. Mikäli on poistutaan silmukasta.

```
If avaus = True Then
  DataGridView1.Rows.Add
  (avain, nimi, päiväys, kello, "Avattiin")
Else
  DataGridView1.Rows.Add
  (avain, nimi, päiväys, kello, "Estettiin")
End If
```

Lisätään luettu rivi taulukkoon.

```
Loop Until EOF(1) Or kk >= DateTimePicker2.Value.Month
And pv > DateTimePicker2.Value.Day
And v >= DateTimePicker2.Value.Year
End If
```

Rivejä käsitellään samalla tavalla kunnes tiedoston viimeinen rivi on luettu tai ylitetään raportin asetettu lopetusaika.

```
FileClose(1)
End If
päivä2 = päivä2.AddMonths(1)

Loop Until päivä2.Month > DateTimePicker2.Value.Month
Or pv >= DateTimePicker2.Value.Day
And kk >= DateTimePicker2.Value.Month
And v >= DateTimePicker2.Value.Year

End Sub
```

Tiedostoja käsitellään kunnes käsiteltävä kuukausi tai viimeisin luettu rivi ylittää raportille asetetun lopetusajan.

4.9.4 Raportin tulostus

```
Private Sub Button2_Click(sender As System.Object, e As
System.EventArgs) Handles Button2.Click
        PrintPreviewDialog1.Document = PrintDocument1
        PrintPreviewDialog1.ShowDialog()
End Sub
```

Kun ”Tulosta raportti”-painiketta painetaan suoritetaan aliohjelma, joka ensin kutsuu toista aliohjelmaa, joka koostaa tulostettavan asiakirjan, joka voi koostua useista sivuista, riippuen raportin pituudesta. Sen jälkeen käyttäjälle esitetään esikatseluikkuna, jossa käyttäjä voi tarkastella tulostetta ja päättää tulostetaanko vai hylätäänkö se.

```
Private Sub PrintDocument1_PrintPage(sender As
System.Object, e As
System.Drawing.Printing.PrintPageEventArgs)
Handles PrintDocument1.PrintPage

Dim newpage As Boolean = True
```

Toisen aliohjelman aluksi määritellään aliohjelman käyttämät muuttujat.

```
With DataGridView1
  Dim fmt As StringFormat = New
  StringFormat(StringFormatFlags.LineLimit)

  fmt.LineAlignment = StringAlignment.Center
  fmt.Trimming = StringTrimming.EllipsisCharacter
```

Seuraavaksi määritellään, tulostettava tieto luetaan sivulla näkyvästä taulukosta. Tämän jälkeen määritellään että tulostettava teksti sijoitetaan aina solun keskelle ja mikäli teksti ei mahdu soluun, se katkaistaan viimeiseen näkyvään merkkiin.

```
  Dim y As Single = e.MarginBounds.Top
  Do While mRow < .RowCount
    Dim row As DataGridViewRow = .Rows(mRow)
    Dim x As Single = e.MarginBounds.Left
    Dim h As Single = 0
```

Aloitetaan silmukka joka käsittelee kaikki taulukon rivit erikseen. Siirretään käsiteltävä rivi omaan muuttujaansa, sekä määritellään solujen tulostuksen sijaintiin käytettyjä muuttujia.

```
    For Each cell As DataGridViewCell In row.Cells
      Dim rc As RectangleF =
      New RectangleF
      (x, y, cell.Size.Width, cell.Size.Height)

      e.Graphics.DrawRectangle
      (Pens.Black, rc.Left, rc.Top, rc.Width, rc.Height)
```

Aloitetaan rivin solujen käsittely. Määritellään solujen koko ja piirretään ne paperille. Paperi ei tarkoita tässä vielä fyysistä tulostamista vaan jää tietokoneen muistiin.

```
If (newpage) Then

  e.Graphics.DrawString(DataGridView1.
  Columns(cell.ColumnIndex).HeaderText,
  .Font, Brushes.Black, rc, fmt)

  e.Graphics.DrawString("Ovikohtainen raportti - " +
  Trim(ListBox1.SelectedItem.ToString) + ": " +
  DateTimePicker1.Value.Date + " - " +
  DateTimePicker2.Value.Date, .Font,
  Brushes.Black, 100, 85)

Else
```

Mikäli kyseessä on uusi sivu, piirretään ensin taulukon otsikot. Tämän jälkeen piirretään taulukon yläpuolelle otsikko, joka käsittää Tiedon että tulostetaan ovikohtaista raporttia, mistä ovesta on kyse, sekä raportin alku ja loppupäivät viivalla erotettuna.

```
  e.Graphics.DrawString(DataGridView1.Rows
  (cell.RowIndex- 1).Cells(cell.ColumnIndex).
  FormattedValue.ToString(), .Font, Brushes.Black, rc, fmt)
End If
```

Mikäli kyseessä ei ollut uusi sivu, tulostetaan käsiteltävä rivi taulukosta.

```
  x += rc.Width
  h = Math.Max(h, rc.Height)
Next
```

Lisätään seuraavan solun vaakapiirustuskohtaan piirretyn solun leveys. Asetetaan muuttujaan ”h” korkein solusta löytynyt rivi. Tämän jälkeen siirrytään käsittelemään rivin seuraavaa solua, tai jatketaan aliohjelman suoritusta eteenpäin, mikäli kyseessä oli rivin viimeinen solu.

```
newpage = False
y += h
mRow += 1
```

Kun rivi on piirretty asetetaan muuttujan ”newpage” arvoksi epätosi, joka kertoo seuraavan rivin käsittelylle, että kyseessä ei ole uusi sivu ja otsikoita ei tarvitse enää piirtää. Lisätään seuraavan rivin piirustuskorkeuteen tä-

män rivin käyttämä korkeus. Korkeus lasketaan ylhäältä alaspäin. Tämän jälkeen lisätään muuttujaan ”mRow” yksi. Muuttuja ”mRow” sisältää käsittelyssä olevan rivin numeron.

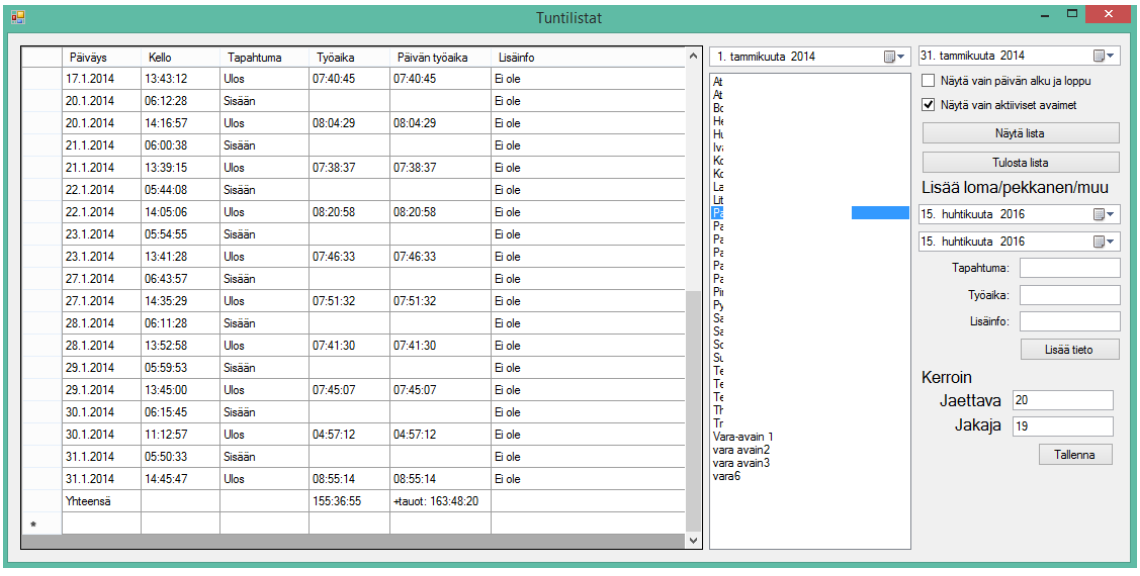
```
If y + h > e.MarginBounds.Bottom Then
  e.HasMorePages = True
  mRow -= 1
  newpage = True
  Exit Sub
End If
Loop
```

Mikäli seuraavan rivin viemä tila ei mahdu sivun marginaaleihin asetetaan ensin päälle tieto, että uusia sivuja on tulossa. Tällöin kun aliohjelmasta poistutaan, aliohjelma suoritetaan välittömästi uudelleen ja se luo seuraavan sivun. Tämän jälkeen vähennetään käsiteltävän rivin numerosta yksi, koska rivinumeroihin oli jo lisätty yksi, vaikka uutta riviä ei oltu käsitelty. Asetetaan päälle tieto että uusi sivu alkaa ja poistutaan aliohjelmasta. Mikäli seuraava rivi mahtuu sivulla palataan ohjelmassa uuden rivin käsittelyyn.

```
mRow = 0
End With
End Sub
```

Kun viimeinen rivi on piirretty paperille, asetetaan käsiteltävän rivin muuttuja nollaksi, lopetetaan taulukon käsittely ja lopetetaan aliohjelman suoritus.

4.10 Työaikalistat



Kuva 9. Tuntilistat koostava ikkuna

Työaikalistoja tuottava ikkuna koostuu taulukosta, jossa itse työaikalista näytetään, listan alku- ja loppuajat määrittävistä kentistä, nimilistasta, kahdesta valintaruudusta, listan näyttämisen ja tulostamisen toiminta-

painikkeista, lisätapahtuman listaan lisäävistä kentistä, sekä kertoimen määrittävästä kentästä. Ikkunan ulkoasu on nähtävissä kuvassa 9.

Taulukko ja aloitus- ja lopetusaika toimivat samoin kuin ovikohtaisessa raportissa ja käyttö on kuvailtu luvussa 4.9.

Ensimmäisellä valintaruudulla, nimeltään ”Näytä vain päivän alku ja loppu”, valitaan näytetäänkö työntekijän yhden päivän tekemistä kirjautumisista vain ensimmäinen ja viimeinen, vai näytetäänkö myös mahdolliset päivän aikana välissä tehdyt ulos- ja sisäänkirjautumiset. Asiakkaan työntekijät kirjautuvat ulos taukojen ajaksi, joten lista saattaisi venyä hyvin pitkäksi, paperin kulutusta ajatellen, mikäli kaikki kirjautumiset näytettäisiin. Myös kertoimen kentät liittyvät taukoihin. Kertoimeen määritetään jaettava ja jakaja, joiden osamäärällä kerrotaan työntekijän työaika. Tällä kompensoidaan työntekijän pitämät tauot päivän aikana. Toinen vaihtoehto olisi ollut laskea työaikaan lisäminuutteja puutarha-alan työehtosopimuksen mukaisesti. Kertoimen käyttö ei ole yhtä tarkka tapa, mutta se on helpompi toteuttaa ja työntekijälle hieman edullisempi kuin työehtosopimuksen määrittämät tauot. Taukolaskenta toteutettiin asiakkaan toiveen mukaisesti. Kerroin tallennetaan tiedostoihin tekstikenttien alapuolella sijaitsevalla ”Tallenna”-painikkeella.

Valintaruudulla ”Näytä vain aktiiviset avaimet” valitaan näytetäänkö nimilistassa vain ne nimet joihin on tällä hetkellä liitetty avain, vai kaikki järjestelmään koskaan lisätyt nimet. Nimilista päivittyy kun ruudun valinta muuttuu. Taulukko päivitetään kun nimilistasta valitaan nimi. Myös ”Näytä nimet”-painike päivittää listan. Painiketta voidaan käyttää mikäli taulukon täyttämisen jälkeen järjestelmässä on tapahtunut uusia kirjautumisia.

Jos työaikalistaan halutaan lisätä rivejä, jotka eivät synny lukijoilta kerätyistä kirjautumisista. Päiväyskenttiin asetetaan tapahtuman alku- ja loppupäiväys. Mikäli tapahtuma käsittää vain yhden päivän, asetetaan kenttiin samat päiväykset. Tapahtuma kenttään kirjoitetaan ylimääräisen rivin syy, esimerkiksi loma tai pekkanen. Työaikaan kirjoitetaan kokonaisluku, josta asiakas päättää itse yksikön suureen, joka voi olla esimerkiksi päiviä tai tunteja. Lisäinfo kenttään voidaan kirjoittaa lisätietoja tapahtumasta, esimerkiksi sairasloman tapauksessa ”Lääkärintodistus toimitettu”. ”Lisää tieto”-painikkeella tiedot kirjoitetaan henkilön henkilökohtaisiin tiedostoihin, joiden rakenne on kuvailtu luvussa 4.12.4. Tieto kirjoitetaan erikseen jokaista päivää kohden.

Taulukko koostuu seuraavista tiedoista:

- Kirjautumisen päiväys
- Kirjautumisen kellonaika
- Kirjautumisen tapahtuma, yleensä kirjautuminen sisään tai ulos
- Uloskirjautumisen kohdalla kertynyt työaika, alkaen viime sisäänkirjautumisesta
- Uloskirjautumisen kohdalla koko päivänä kertynyt työaika
- Mahdollinen tapahtumaan liittyvä lisätieto

Kun taulukko täytetään, viimeiselle riville lasketaan yhteen kertynyt työaika, sekä ilman taukoja, että tauot mukaan laskettuna. Tämän jälkeen taulukkoa jatketaan käsin lisättyjen tietojen listaamisella. Käsin lisätyn tiedon työaika lasketaan erikseen yhteen.

4.10.1 Ikkunan käyttämät muuttujat

```
Dim mRow As Integer = 0
Dim newpage As Boolean = True
```

Ikkunan muuttujia on vain kaksi ja ne liittyvät työaikalistan tulostamiseen. Tulostaminen tapahtuu lähes identtisesti ovikohtaisen raportin kanssa, kuten luvussa 4.9.4 on esitetty, eikä sen lähdekoodia esitetä uudelleen. Ainoa ero on sivun otsikossa, johon tulostetaan ”Tuntilista:” ja työntekijän nimi, listan aikajakson lisäksi.

4.10.2 Ikkunan lataus

```
Private Sub Tuntilistat_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load

 ' read multiplier info from a file

 FileOpen(1, "C:\Kulunvalvonta\Asetukset\Kerroin.data",
 OpenMode.Input)

 Input(1, TextBox4.Text)
 Input(1, TextBox5.Text)
 FileClose(1)
 Call taytaaktiivisetavaimet()
End Sub
```

Kun ikkuna ladataan, luetaan ensimmäisenä asetustiedostosta kertoimen tekstikenttiin arvot. Tiedostojen rakenne on kuvailtu luvussa 4.12.1. Tämän jälkeen kutsutaan aliohjelmaa, joka täyttää nimilistaan nimet, joihin on liitetty avain.

4.10.3 Näytä vain aktiiviset avaimet

```
Private Sub ShowActiveKeysOnly_CheckedChanged
(sender As Object, e As EventArgs)
Handles ShowActiveKeysOnly.CheckedChanged

If ShowActiveKeysOnly.Checked = True Then
  Call taytaaktiivisetavaimet()
Else
  taytakaikkinimet()
End If

End Sub
```

Kun ”Näytä vain aktiiviset avaimet”-valintaa muutetaan, tarkistetaan valinnan tila ja kutsutaan valinnan mukaisesti aliohjelmaa joka päivittää nimilistaan nimet, joihin on linkitetty avaimet, tai kaikki järjestelmään syötetyt nimet.

4.10.4 Nimilistan päivitys kaikilla järjestelmän nimillä

```
Sub taytakaikkinimet()
  ListBox1.Items.Clear()
  Dim dra As IO.DirectoryInfo
  Dim di As New IO.DirectoryInfo
  ("c:\Kulunvalvonta\Henkilöt\")

  Dim diar1 As IO.DirectoryInfo() = di.GetDirectories()

  For Each dra In diar1
    If dra.ToString = "Paikalla" Then
    Else
      ListBox1.Items.Add(dra)
    End If
  Next
ListBox1.Sorted = True
End Sub
```

Aliohjelma joka täyttää nimilistaan kaikki järjestelmään syötetyt nimet, tyhjentää ensimmäisenä nimilista, listaa sitten henkilökansion alikansiot ja tarkastaa jokaisen kansion nimen. Mikäli kansion nimi on ”Paikalla”, ei tehdä mitään, mikäli se on jotain muuta, lisätään kansion nimi listalle. Lopuksi asettaa nimilistan järjestymään aakkosjärjestykseen.

4.10.5 Nimilistan päivitys aktiivisilla avaimilla

```
Sub taytaaktiivisetavaimet()
Dim nimi, roska As String
Dim collecthours As Boolean
ListBox1.Items.Clear()
Dim dra As IO.FileInfo
Dim di As New IO.DirectoryInfo("c:\Kulunvalvonta\Avaimet\")
Dim diar1 As IO.FileInfo() = di.GetFiles
```

Nimilistan päivitys aktiivisilla avaimilla aloitetaan määrittelemällä aliohjelman käyttämät muuttujat, tyhjentämällä nimilista ja listaamalla järjestelmän avaintiedostot.

```
For Each dra In diar1
  FileOpen(1, "c:\Kulunvalvonta\Avaimet\"
  + dra.ToString, OpenMode.Input)

  Input(1, nimi)
  Input(1, roska)
  Input(1, roska)
  Input(1, roska)
  Input(1, roska)
  Input(1, roska)
  Input(1, roska)
  Input(1, roska)
  Input(1, roska)
```

```
  Input(1, roska)
  Input(1, roska)
  Input(1, roska)
  Input(1, roska)
  Input(1, roska)
  Input(1, collecthours)
  FileClose(1)
  If collecthours Then ListBox1.Items.Add(nimi)
Next
```

Seuraavaksi luetaan avaintiedoston rivit. Vain ensimmäisen rivin nimi ja viimeisen rivin tietoa kerätäänkö henkilön työaikalistaa tarvitaan, joten muut tiedot luetaan muuttujaan ”roska”, jota ei käytetä mihinkään. Tarkastellaan kerätäänkö henkilön työaikalistaa ja mikäli kerätään, lisätään nimi nimilistaan.

```
ListBox1.Sorted = True
End Sub
```

Viimeisenä asetetaan nimilista järjestymään aakkosjärjestykseen.

4.10.6 Tapahtuman lisääminen käsin

```
Private Sub Button1_Click(sender As System.Object, e As
System.EventArgs) Handles Button1.Click
  Dim aika1 As Date
  Dim aika2 As Date
  aika1 = DateTimePicker3.Value
  aika2 = DateTimePicker4.Value
```

Kun ”Lisää tieto”-painiketta painetaan, määritellään ensimmäisen aliohjelman käyttämät muuttujat ja asetetaan niihin käyttäjän määrittämät aloitus ja lopetusaika.

```
  If Not IsNumeric(TextBox2.Text) Then
    MsgBox("Työajassa on oltava kokonaisluku."
    + vbNewLine + "Jos et halua aikaa, käytä arvoa 0." +
    vbNewLine + "Jos haluat laskea päivät käytä arvoa 1.")

    Exit Sub
  End If
```

Tarkistetaan että työaikakenttään on syötetty numeerinen arvo. Mikäli arvo ei ole numeerinen, ilmoitetaan käyttäjälle viestilaatikolla, että kentässä täytyy olla numeerinen arvo ja lopetetaan aliohjelman suoritus.

```
  If ListBox1.SelectedIndex >= 0 Then
  Else
    MsgBox("Henkilöä ei ole valittu!")
    Exit Sub
  End If
```

Seuraavaksi tarkistetaan, että nimilistasta on valittu nimi. Mikäli nimeä ei ole valittu, ilmoitetaan siitä käyttäjälle viestilaatikolla ja lopetetaan aliohjelman suoritus.

```
  If aika1 = aika2 Then
    FileOpen(1, "C:\Kulunvalvonta\Henkilöt\"
    + Trim(ListBox1.SelectedItem.ToString)
    + "\Lomat-" + Trim(Str(aika1.Year))
    + "." + Trim(Str(aika1.Month)), OpenMode.Append)

    PrintLine(1, aika1.Day, ",", aika1.Month, ",",
    aika1.Year, ",", TextBox1.Text, ",",
    TextBox2.Text, ",", TextBox3.Text)

    FileClose(1)
```

Seuraavaksi tarkistetaan onko aloituspäivä sama kuin lopetuspäivä. Mikäli päivät ovat samat, kirjoitetaan henkilökohtaiseen tiedostoon lisättävä rivi. Henkilökohtaiset tiedostot on kuvailtu luvussa 4.12.4.

```
Else
  Do
    FileOpen(1, "C:\Kulunvalvonta\Henkilöt\"
    + Trim(ListBox1.SelectedItem.ToString)
    + "\Lomat-" + Trim(Str(aika1.Year)) + "."
    + Trim(Str(aika1.Month)), OpenMode.Append)

    PrintLine(1, aika1.Day, ",", aika1.Month, ",",
    aika1.Year, ",", TextBox1.Text, ",", TextBox2.Text,
    ",", TextBox3.Text)

    FileClose(1)
    aika1 = aika1.AddDays(1)
  Loop Until aika1 > aika2
End If
```

Mikäli aloituspäivä ja lopetuspäivä eivät ole samat, aloitetaan silmukka, jossa kirjoitetaan rivi henkilökohtaiseen tiedostoon, lisätään seurattavaan päiväykseen päivä ja kirjoitetaan uusia rivejä, kunnes seurattava aika asettaa käyttäjän asettaman loppuajan. Rivit ovat päiväystä lukuun ottamatta identtisiä.

```
MsgBox("Tiedot lisätty!")
End Sub
```

Kun tiedosto on kirjoitettu, ilmoitetaan siitä käyttäjälle viestilaatikolla ja aliohjelma päättyy.

4.10.7 Kertoimen tallennus

```
Private Sub Button4_Click(sender As System.Object, e As
System.EventArgs) Handles Button4.Click

FileOpen(1, "C:\Kulunvalvonta\Asetukset\Kerroin.data",
OpenMode.Output)

PrintLine(1, TextBox4.Text, ",", TextBox5.Text)
FileClose(1)
MsgBox("Tallennettu")

End Sub
```

Kun kertoimen tekstikenttien alapuolella sijaitsevaa ”Tallenna”-painiketta painetaan, kirjoitetaan kertoimen asetustiedostoon tekstikenttiin täytetyt arvot. Talletuksesta ilmoitetaan käyttäjälle viestilaatikolla.

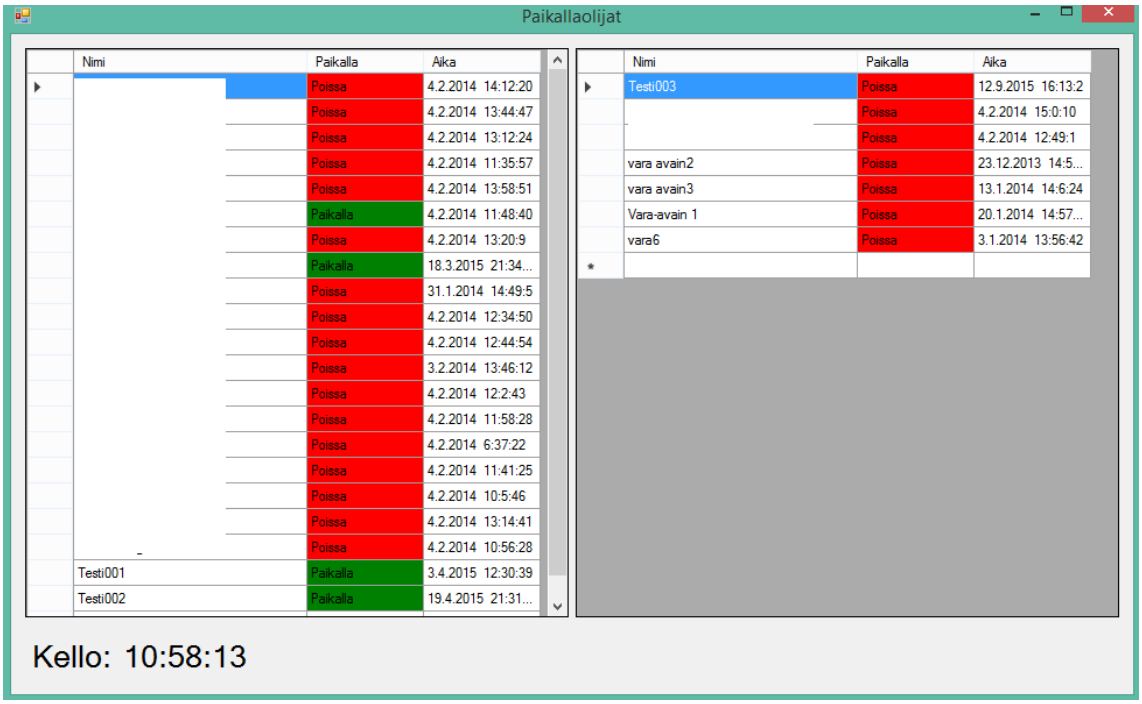
4.10.8 Työaikalistan muodostaminen

Työaikalista muodostetaan taulukkoon lähes identtisesti oviraportin kanssa, mikä on kuvattu luvussa 4.9.3, joten lähdekoodia ei esitetä tässä. Erona on se, että uuden rivin yhteydessä asetetaan muuttujiin talteen edellinen kirjautumisaika ja tieto oliko kirjautuminen ulos vai sisään. Mikäli kyseessä ei ollut ensimmäinen rivi, vertaillaan mikäli kirjautuminen oli sama, kuin viime rivillä, milloin lisätään ”Lisäinfo”-kenttään teksti ”Kirjautuminen puuttuu”. Mikäli edellinen kirjautuminen oli sisään ja uusi ulos, lasketaan näiden välissä kulunut aika, joka lisätään työaika kenttään. Saman vuorokauden aikana toteutuneet työajat summataan ja lisätään ”Päivän työaika”-kenttään. Työaikoja summataan myös yhteen koko listan ajan ja näytetään viimeisellä rivillä yhteen laskettu tieto, sekä taukokertoimella korjattu työaika yhteensä.

Tämän jälkeen suoritetaan toinen lähes identtinen listan muodostus, joka jatkuu tähän asti täytetyn taulukon perään. Tähän osaan listaa ladataan käsin syötetyt tiedot. Erona edelliseen on se, että aikaa ei lasketa, vaan se on syötetty kokonaislukuina käsin, eikä näiden summaa esitetä kertoimella korjattuna ja luettavan tiedoston nimi eroaa hieman kirjautumistiedoston nimestä.

Listojen tiedot luetaan henkilökohtaisista tiedostoista, joiden rakenne on kuvattu luvussa 4.12.4.

4.11 Paikallaolijat



Kuva 10. Paikallaolijat näyttävä ikkuna

Paikallaolijalista näytetään kääntöportin viereisessä näytössä. Näytöltä näkee onko henkilö kirjautunut sisään vai ulos, sekä ajan milloin henkilö on viimeksi kirjautunut. Näytöllä näkyy myös sen hetkinen kellonaika, jonka avulla käyttäjät voivat tarkastaa, että järjestelmä on oikeassa ajassa. Näyttö päivittyy reaaliajassa. Kun henkilö kirjautuu sisään tai ulos tarkistetaan, onko hänelle luotu paikallaolotiedostoa. Mikäli tiedosto löytyy, kirjoitetaan tiedostoon uusi aika ja paikalla- tai poissatieto, sen mukaan kirjautuuko henkilö sisään vai ulos. Tämän jälkeen näyttö päivitetään. Päivitys tapahtuu identtisesti pääikkunan listan päivityksen kanssa, mikä on esitetty luvussa 4.3.8.

Ikkunan taustatoimintoja hoidetaan ajastimella, jonka koodi suoritetaan kymmenen kertaa sekunnissa.

```
Private Sub Timer1_Tick(sender As System.Object, e As Sys-
tem.EventArgs) Handles Timer1.Tick

If päivitäpaikallaolijat(0) = True Then Exit Sub

If päivitäpaikallaolijat(1) = True Then
  Call päivitäpaikaololista()
  päivitäpaikallaolijat(1) = False
End If
```

Ensimmäisenä tarkastetaan onko paikallaololistassa tapahtunut muutoksia. Tämä toiminta on esitetty pääikkunan osalta luvussa 4.3.8.

```
Dim kello As String
If Now.Hour < 10 Then
  kello = "0" + Trim(Str(Now.Hour))
Else
```

```
  kello = Trim(Str(Now.Hour))
End If

If Now.Minute < 10 Then
  kello = kello + ":0" + Trim(Str(Now.Minute))
Else
  kello = kello + ":" + Trim(Str(Now.Minute))
End If

If Now.Second < 10 Then
  kello = kello + ":0" + Trim(Str(Now.Second))
Else
  kello = kello + ":" + Trim(Str(Now.Second))
End If

Label2.Text = kello

End Sub
```

Tämän jälkeen muutetaan kellonaika merkkijonoksi, jossa esitetään kuluva tunti, minuutti ja sekunti, esitettynä kaksi numeroisina lukuina, jotka on erotettu kaksoispisteellä ja asetetaan merkkijono näytöllä näkyvään tekstiin.

4.12 Tiedostot

Ohjelman luoma tiedostorakenne jaettiin osiin, jotta tiedostot säilyisivät selkeinä. Ohjelma luo tiedostoja varten kansion tietokoneen c-aseman juureen, jonka nimi on ”Kulunvalvonta”. On suositeltavaa, ettei tämän nimistä kansiota olisi vielä käytössä, jotta ohjelman käyttämät tiedostot eivät sekoittuisi muiden tiedostojen kanssa. Ohjelma luo kansioon alikansioita, joihin kerätään dataa lukijoiden käytöstä, työtunneista sekä avaimien linkityksestä henkilöihin.

Suunnitellessa tiedostojen rakennetta täytyy ottaa huomioon, että mitä suurempi tiedosto, sen kauemmin sen lukeminen kestää. Mikäli tiedosto on useita megatavuja pitkä ja haettava tieto on tiedoston lopussa, täytyy koko tiedosto käsitellä ennen kuin haettu tieto löytyy. Tästä syystä esimerkiksi vuoden tietojen kerääminen yhteen tiedostoon, ei ole järkevää, mikäli tietoa tallennetaan usein. Mikäli tiedostot taas ovat hyvin pieniä, niiden määrä kasvaa todella nopeasti. Jos kerätylle tiedostolle luotaisiin päiväkohtainen tiedosto, pysyisivät tiedostot yksittäisinä hyvin pieninä ja nopeina käsitellä, mutta niiden määrä kasvaisi nopeasti todella suureksi. Suuri määrä pieniä tiedostoja hankaloittaa tiedostojen varmuuskopiointia ja listaamista, jotka hidastuvat suuresti. Pienet tiedostot lisäävät myös tallennustilan kulutusta, koska tiedoston sisällön lisäksi levytilaa kuluttavat tiedoston tiedot. Tämän lisäksi tallennusmedioille on määritelty varausyksikkö, joka on pienin tila minkä tiedosto varaa itselleen. Mikäli tiedostot ovat tätä yksikköä pienempiä, kuluttavat ne kuitenkin vähintään varausyksikön tilan tallennusmedialta. Uuden tiedoston luontitiheydeksi määritettiin yksi kuukausi, joka todettiin sopivaksi kompromissiksi isojen ja pienten tiedostojen välillä.

4.12.1 Asetustiedostot

Asetustiedostot sijaitsevat ”Asetukset” nimisessä alikansiossa. Asetustiedostoja on kuusi ja ne sisältävät oven ja kääntöportin avauspulssien pituuden, mikrokontrollerin EEPROMiin kirjoitetun datan, tuntilistojen laskennassa käytettävän kertoimen, sekä sarjaportin tunnukset. Jokainen tiedosto on tekstityyppiä ”.data”-päätteellä.

Aukiolopulssien pituudet on talletettu tiedostoihin ”Aukiaika1.data”, ”Aukiaika2.data” sekä ”Aukiaika3.data”. Järjestyksessä ne merkitsevät ovea, kääntöporttia sisäänpäin, sekä kääntöporttia ulospäin. Ne sisältävät yhden tekstirivin, joka koostuu yhdestä numerosta. Kyseinen numero on aukipulssin pituus sekunnin kymmenesosina ilmaistuna. Numeron muoto on tavu, arvot voivat olla kokonaislukuja väliltä 0-255.

EEPROMin data on tallennettu tiedostoon ”EEPROM.data” ja se käsittää 256 riviä, joista kukin koostuu yhdestä numerosta. Jokainen numero vastaa yhtä tavua mikrokontrollerin EEPROM-muistissa ja arvot voivat olla kokonaislukuja väliltä 0-255.

”Kerroin.data” sisältää yhden rivin, joka koostuu kahdesta pilkulla erotetusta numerosta. Numerot ovat muotoa Integer niiden arvot voivat olla kokonaislukuja väliltä -2147483648 ja 2147483647. Näiden lukujen osamäärästä saadaan kerroin, jonka käyttö on kuvattu luvussa 4.10.

”Sarjaportti.data” sisältää yhden rivin, joka koostuu merkkijonosta. Tämä merkkijono on nimi jolla Windows tunnistaa käytettävän sarjaportin.

4.12.2 Avainten tiedostot

Järjestelmään tallennetut avaimet sijaitsevat kansiossa ”C:\Kulunvalvonta\Avaimet”. Tiedoston nimi koostuu neljästä tavusta, joista avaimen tunnus koostuu ohjelmassa. Tiedoston nimessä ei ole lisättyä päätettä ja tiedostot ovat tekstimuodossa. Taulukossa 5 on esitetty tiedoston rakenne.

Taulukko 5. Avaintiedoston rakenne

Tieto	Datatyyppi
Nimi	String
Sallitun ajan alku: Tunti	Integer
Sallitun ajan alku: Minuutti	Integer
Sallitun ajan loppu: Tunti	Integer
Sallitun ajan loppu: Minuutti	Integer
Sallittu päivä: Maanantai	Boolean
Sallittu päivä: Tiistai	Boolean
Sallittu päivä: Keskiviikko	Boolean
Sallittu päivä: Torstai	Boolean
Sallittu päivä: Perjantai	Boolean
Sallittu päivä: Lauantai	Boolean
Sallittu päivä: Sunnuntai	Boolean
Salli paikallaolon näyttö	Boolean
Salli ovien avaus aina	Boolean
Kerää tuntilista	Boolean

4.12.3 Ovikohtaiset tiedostot

Ovikohtaiset raportit sijaitsevat kansioissa ”C:\Kulunvalvonta\Oviloki”, ”C:\Kulunvalvonta\Oviloki2” ja ”C:\Kulunvalvonta\Oviloki3”. Yksi tiedosto sisältää yhden kuukauden tapahtumat ja sen nimi koostuu vuodesta ja kuukauden numerosta, pisteellä erotettuna. Jokainen lukutapahtuma RFID-lukijoista on kirjattu omalle rivilleen. Rivin tiedot ovat pilkulla erotettuja. Taulukossa 6 on esitetty rivin rakenne.

Taulukko 6. Oviraporttitiedoston rivin rakenne

Järjestys	Tieto	Datatyyppi
1	Luettu koodi	String
2	Koodiin liitetty nimi	String
3	Päivä	Integer
4	Kuukausi	Integer
5	Vuosi	Integer
6	Tunnit	Integer
7	Minuutit	Integer
8	Sekunnit	Integer
9	Tapahtuma	Boolean (Tosi=avattiin, epätosi=ei avattu)

4.12.4 Henkilökohtaiset tiedostot

Henkilökohtaiset tiedostot sisältävät tiedot henkilöiden töihin ja töistä kirjautumisista. Jokaiselle henkilölle on luotu oma alikansionsa, jonka nimi on järjestelmään syötetty henkilön nimi. Ne sijaitsevat kansiossa ”C:\Kulunvalvonta\Henkilöt”. Samassa kansiossa on myös alikansio paikallaolijatietoja varten. Yksi tiedosto sisältää yhden kuukauden tapahtumat

ja sen nimi koostuu vuodesta ja kuukaudesta, pisteellä erotettuna. Jokainen kirjautuminen on kirjattu omalle rivilleen. Rivin tiedot ovat pilkulla eroteltuja. Taulukossa 7 on esitetty rivin rakenne.

Taulukko 7. Henkilön kirjautumistiedoston rivin rakenne

Järjestys	Tieto	Datatyyppi	
1	Päivä	Integer	
2	Kuukausi	Integer	
3	Vuosi	Integer	
4	Tunti	Integer	
5	Minuutti	Integer	
6	Sekunti	Integer	
7	Kirjautumistieto	String	"Sisään" tai "Ulos"
8	Lisätieto	String	

Lisäksi kansio sisältää käsin lisätyt tiedot omissa tiedostoissaan. Tiedoston nimi, on kuten kirjautumistiedoston nimi, mutta sen eteen on lisätty teksti ”Lomat-”. Tiedostoissa on jokaiselle kirjatulle tapahtumalle oma rivinsä ja rivin tiedot ovat pilkulla eroteltuja. Taulukossa 8 on esitetty rivin rakenne.

Taulukko 8. Käsin lisättyjen tiedostojen rivin rakenne

Järjestys	Tieto	Datatyyppi
1	Päivä	Integer
2	Kuukausi	Integer
3	Vuosi	Integer
4	Tapahtuma	String
5	Työaika	Integer
6	Lisätieto	String

4.12.5 Paikallaolon näyttävät tiedostot

Paikallaolotiedostot sisältävät tiedon, onko henkilö kirjautunut töihin ja viimeisimmän kirjauksen ajankohdan. Paikallaolotiedostot sijaitsevat kansiossa ” C:\Kulunvalvonta\Henkilöt\Paikalla”. Tiedosto koostuu yhdestä rivistä, jonka tiedot on eroteltu pilkuilla. Taulukossa 9 on esitetty tiedoston rakenne.

Taulukko 9. Paikallaolotiedoston rakenne

Järjestys	Tieto	Datatyyppi
1	Paikallaolo	Boolean (True=paikalla, false=poissa)
2	Päiväys	Integer
3	Kellonaika	Integer

4.13 Henkilörekisteri ja tietoturva

Kulunvalvonnasta syntyy henkilörekisteri, jota koskee henkilötietolaki. Henkilörekisteristä on laadittava rekisteriseloste on pidettävä kaikkien saatavilla (Henkilötietolaki 1999/523§ 10).

Vaikka kohteessa oli ollut käytössä kulunvalvonta jo ennen tämän projektin toteuttamista, ei rekisteriselostetta oltu laadittu, joten sellainen laadittiin. Rekisteriseloste on esitetty liitteessä x. Rekisteripohja on saatavilla Tietosuojavaltuutetun toimiston kotisivulta (Tietosuojavaltuutetun toimisto, 2014).

Ohjelmiston kehityksessä ei keskitytty tietoturvaan, siitä huolimatta, että järjestelmä sisältää paljon henkilökohtaista tietoa. Mikäli ohjelmistoa jatkokehitetään, on siihen mahdollista lisätä esimerkiksi salasanavarmennus tietojen käsittelyä ja lukemista varten. Asennuskohteessa tietokone, johon ohjelmisto on asennettu, ei ole yhdistettynä tietoverkkoihin, minkä vuoksi, myöskään internetistä tapahtuvat tietomurrot eivät ole mahdollisia. Tämän lisäksi tiloissa on tallentava kameravalvonta, jolla voidaan varmentaa tiloissa liikkuneita henkilöitä. Kameravalvonnassa ei kuitenkaan näy suoraan tietokoneen käyttäjät, mikä voidaan nähdä turvallisuusriskinä.

5 VANHAN JA UUDEN JÄRJESTELMÄN VERTAILU

5.1 Vanha järjestelmä

Alkuperäinen kulunvalvontajärjestelmä oli ALT-Locking Oy:n markkinoima e-Access –järjestelmä, joka koostui kahdesta oviyksiköstä, joita hallittiin lähiverkon kautta. Laitteet vikaantuivat asiakkaan verkkopäivitysten yhteydessä korjauskelvottomiksi.

Oviyksiköt toimivat itsenäisesti ja sisälsivät tietokannat joihin kirjautumistiedot kerättiin. Tietojen luku tapahtui käynnistämällä ensin tietokoneelta ohjelmisto, joka toimi web-palvelimena ja yhdyskäytävänä yksiköiden tietokantoihin. Ohjelman käynnistämisen jälkeen käyttö tapahtuu internet-selaimella. Internet-selaimella tiedot siirrettiin ensin yksiköiltä tietokoneeseen, minkä jälkeen niistä voitiin koostaa työaikalistat tekstitiedostoiksi. Yhteen yksikköön oli mahdollista liittää kaksi lukijalaitetta.

Käytetyt yksiköt oli hankittu 2000-luvun alussa, tarkka aika ei ole tiedossa. Laitteen nykyisestä esitteestä voidaan päätellä, että uusien laitteiden käytettävyys on kehittynyt huomattavasti, vanhempiin yksiköihin verrattuna. Esitteessä mainitaan mm:

- Yksi yksikkö voi ohjata yhdestä neljään ovea
- Käyttö miltä tahansa web-selaimen sisältävältä laitteelta
- Mahdollisuus pilvipalveluiden käyttöön tiedonkeruussa
- SQL-pohjainen tiedonkeruuohjelmisto
- Hälytyksien ja ilmoitusten lähetys sähköpostina

Kun huomioidaan, maininta SQL-pohjaisesta ohjelmistosta, sekä pilvipalvelusta, voidaan olettaa, että vanha tapa käynnistää web-palvelin omalta

tietokoneelta, on korvattu, tai on mahdollista korvata, pilvipalvelulla, jossa tiedot kerätään ALT Locking Oy:n omistamille palvelintietokoneille. Tätä tukee myös tukipalveluun soitetun puhelun aikana tehdyt tiedustelut laitteiden mahdollisesta uusimisesta. Pilvipalvelu olisi lisännyt laitteiden käyttökustannuksia kuukausimaksun muodossa. (ALT Locking Oy, n.d.)

5.2 Erot vanhan ja uuden järjestelmän välillä

Yksi huomattavimmista eroista e-Accesin ja uuden järjestelmän välillä, on se että e-Access ei vaadi erillistä tietokonetta toimiakseen. Tietojen luku laitteesta vaatii kuitenkin tietoverkon, jota ei vaadita uudessa järjestelmässä. Mikäli e-Access -yksiköitä käytettäisiin pilvipalvelun kanssa, tarvittaisiin tietojen lukemiseen myös internet-yhteys. Taulukossa 10 on esitetty e-Accesin esitteessä listatut ominaisuudet ja verrattu niitä uuden järjestelmän ominaisuuksiin. Vertailtaessa on syytä ottaa huomioon, että uusi järjestelmä kehitettiin täysin asiakkaan tarpeiden mukaisesti, jättäen huomiotta vanhan laitteiston ominaisuudet, jotka eivät olleet käytössä. (ALT Locking Oy, n.d.)

Taulukko 10. e-Accesin ja uuden järjestelmän ominaisuuksien vertailu

Ominaisuus	Uusi järjestelmä
Henkilöryhmä	Henkilöitä ei voi ryhmitellä, jokainen käyttäjä on yksilö
Pääsyoikeudet	Oikeuksia ei määritellä ovikohtaisesti
Viikko-ohjelma	Kulkuoikeudet voidaan määritellä viikonpäivän ja kellonajan perusteella
Kalenteri	Ei ole
Avaus kortilla	On
Avaus kortti+pin yhdistelmällä	Järjestelmä ei tue numeronäppäimistöjä.
Oven aukipitopainike	Ohjelmistossa on oville avauspainike, mutta ei aukipitopainiketta
Ovi- telki ja optiovalvonta	Ei ole
Tapahtumaloki	Kattaa jokaisen kortinluvun ovilta
Älykkäät tiedonhaut	Haetaan henkilöön tai oveen sidottuja tietoja päivämäärien perusteella
Huoltotarve-ennuste	Ei ole
Palomuuri, SSL	Ei sisällä verkkotoimintoja
Tiedonsiirto hallintoon ja tietokantaan	Ei ole
Ohjelmiston päivitykset selaimella	Päivitykset asennetaan manuaalisesti

6 YHTEENVETO

Asiakkaan äkillinen tarve laitteelle mahdollisti mielenkiintoisen projektin kulunvalvonnan parissa. Varsinainen työ jouduttiin aluksi toteuttamaan kovalla kiireellä, koska laite piti saada pikaisesti käyttöön. Projektin aloittamisesta kymmenen päivän kuluessa käytössä oli ensimmäinen prototyyppi laitteistosta. Tämän jälkeen laitteistoa ja ohjelmistoa parannettiin ja niissä esiintyviä virheitä korjattiin useaan otteeseen, kuukausien ajan.

Projekti oli mielenkiintoinen ja laaja. Se opetti suurien datamäärien järjestelyä, sekä ohjelmointia niiden käsittelyyn.

Kehitetyn laitteen kaupallista potentiaalia arvioitiin myös muihin kohteisiin myytäviksi ja todettiin että potentiaalia kyllä olisi, etenkin sen toteuttamishinnan puolesta, mutta sen tuotesuojaus olisi hankalaa, koska se on valmistettu yleisesti saatavilla olevista osista, ilman mitään patentoivaksi riittävää osaa. Tulevaisuuden suunnitelma projektin tuotteiden osalta onkin jatkaa niitä harrastuspohjalta, kehittäen ohjelmistosta modulaarinen

versio, jossa laitteita voidaan liittää useampi ja muuttaa tietoliikenne kulkemaan ethernet-verkon yli. Tämän jälkeen harkitaan projektin julkaisua avoimena lähdekoodina, niin laitteiston kuin ohjelmistonkin osalta.

Koska lähdekoodi kirjoitettiin alun perin kiireessä ja se on laaja kokonaisuus, sisälsi se aluksi paljon virheitä ja virheitä löytyy varmasti jatkossakin. Tunnettuja ongelmia ovat mm:

- Tulostettaessa ovikohtaisia raportteja, tietokoneohjelma saattaa kaatua
- Mikäli työaikalista koostetaan juuri samalla hetkellä, kun henkilö käyttää lukijaa, saatetaan samaa tiedostoa yrittää käyttää samanaikaisesti kahdesta kohdasta ohjelmaa, mikä kaataa ohjelman

Lisäksi lähdekoodissa on paljon kohtia, joita voidaan optimoida toiminnan nopeuttamiseksi ja koodin luettavuuden parantamiseksi.

Laitteisto kehitettiin asiakkaan tarpeen mukaisesti kolmelle lukijalle, tätä määrää olisi mahdollista nostaa, mikä voisi laskea valmistuskuluja isommille järjestelmille.

Valmistetusta laitteesta on otettava huomioon, että laitteen valmistus oli kertaluontoinen ja prototyyppi-luonteinen laite. Tästä syystä laitteen komponentteja valitessa ei ole kiinnitetty huomiota komponenttien CE-hyväksyntöihin ja laitteessa on käytetty ulkomailta tilattuja niin sanottuja kloonilaitteita. Laitteen kotelointi ja kaapelointi on toteutettu kotimaasta hankituilla hyväksytyillä tuotteilla, minkä vuoksi katsottiin, ettei laite aiheuta sähköiskun vaaraa käyttäjille. Lisäksi laitteen sähkönsyöttöä lukuunottamatta laite toimii pienoisjännitteellä, jota käyttäviä sähkölaitteita saa rakentaa, korjata ja asentaa ilman sähköalan virallista koulutusta. Laitteessa voidaan käyttää myös ulkoista virtalähdettä, milloin laite olisi täysin pienoisjännitelaite.

Laitteen kehittäminen prototyypistä kaupalliseksi laitteeksi vaatisi käytännössä laitteiston uudelleensuunnittelua piirilevytasolta, milloin kaikki laitteen elektroniikka voitaisiin sijoittaa yhteen piirilevyyn. Tämä mahdollistaisi laitteen edullisen massatuotannon, pieneen koteloon mahtumisen, sekä varmistaisi kaikkien piirien suunnittelun yhtenäisyyden.

Wiegandin tekniikat osottautuivat hyvin mielenkiintoisiksi ja auttoivat ymmärtämään joidenkin antureiden ja laitteiden toimintaa. Myös kulunvalvontaan tarkoitettujen laitteiden kirjo osottautui laajaksi.

Projekti oli onnistunut, sillä asiakas sai tarvitsemansa laitteen käyttöön nopealla aikataululla. Myöhempien kehityskeskustelujen perusteella asiakas on myös hyvin tyytyväinen laitteen käyttöliittymään, jonka kokee helpommaksi ja selkeämmäksi, kuin alkuperäisen rikkoutuneen laitteen käyttöliittymän.

LÄHTEET

ALT Locking Oy, (n.d). eAcces Internet-aikakauden kulunvalvontaa. Haettu 13.11.2016 osoitteesta
http://www.advancedlocking.com/public/esite7fin_p.pdf

Arduino (2008). Arduino Nano (V2.3) User Manual. Haettu 1.5.2016 osoitteesta
https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf

Arduino (n.d.). Genuino. Haettu 13.11.2016 osoitteesta
https://www.arduino.cc/en/Main/GenuinoBrand

Arduino (n.d.). What is Arduino?. Haettu 13.11.2016 osoitteesta
https://www.arduino.cc/en/Guide/Introduction

Atmel (2016). 8-bit AVR Microcontrollers - ATmega328/P – DATASHEET COMPLETE. Haettu 27.11.2016 osoitteesta
http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_datasheet.pdf

Crypton Electronic Co Ltd (n.d.). How to connect biometric device with our Access control. Haettu 13.11.2016 osoitteesta
http://www.crypton.hk/article.php?id=19

Davis M (2009). Who invented the proximity card. Haettu 6.11.2016 osoitteesta
https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-mike_davis_who_invented_the_proximity_card.pdf

Henkilötietolaki 1999/523. Haettu 7.5.2016 osoitteesta
http://www.finlex.fi/fi/laki/ajantasa/1999/19990523

HID (2005), Technology White Paper. Haettu 14.4.2016 osoitteesta
https://www.hidglobal.com/sites/default/files/hid-understanding_card_data_formats-wp-en.pdf

RCI (2015). 5355R 125kHz WIEGAND PROCIMITY READER & KEYPAD COMBO. Haettu 13.11.2016 osoitteesta
http://www.rutherfordcontrols.com/media/7259/ZL5355_R0115-2.pdf

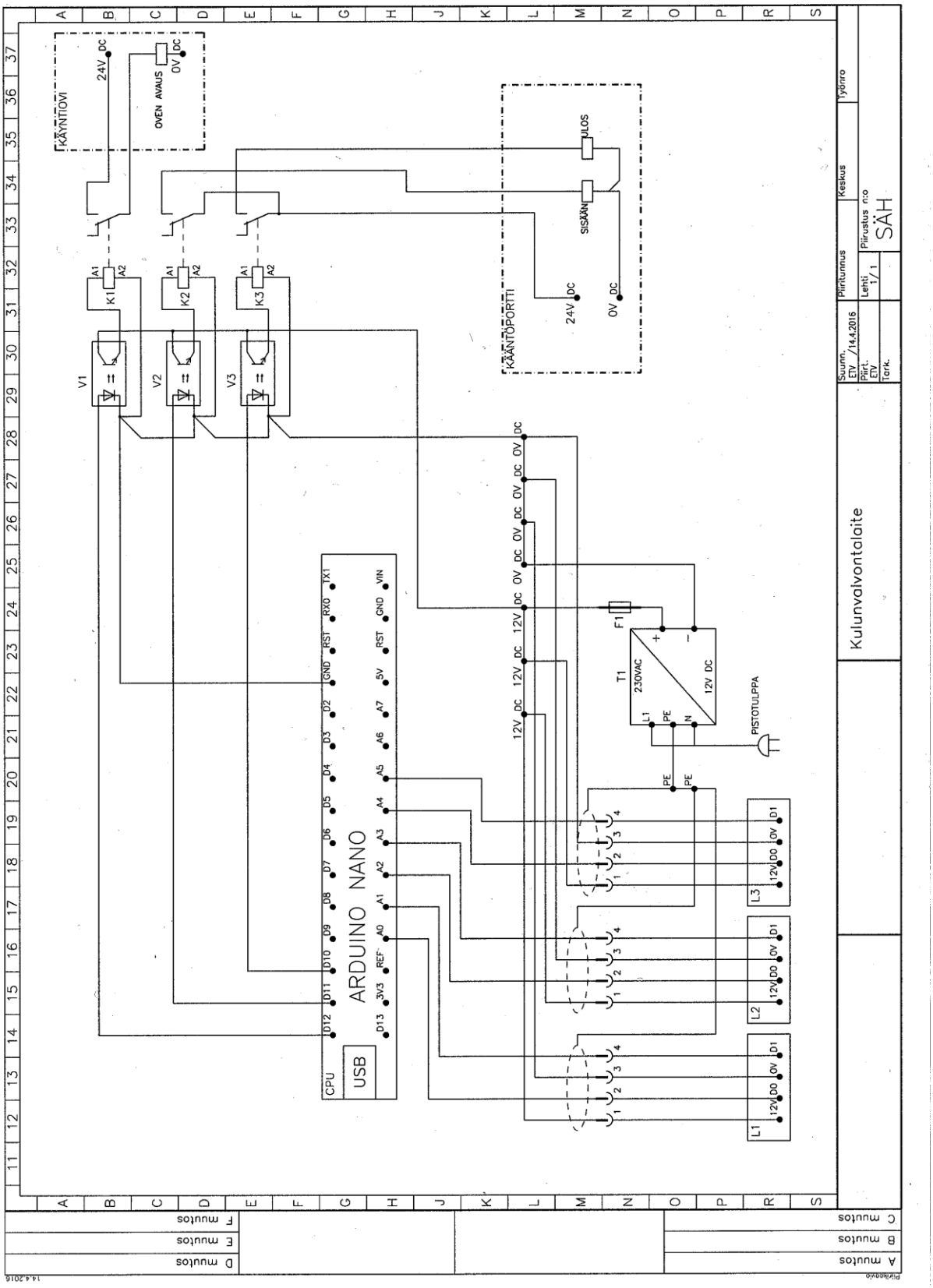
RobotShop (n.d.). Wiegand protocol format. Haettu 14.4.2016 osoitteesta
http://www.robotshop.com/media/files/pdf/wiegand-protocol-format-pr25.pdf

Tietosuojavaltuutetun toimisto (2014). Rekisteri- ja tietosuojaselosteet. Haettu 7.5.2016 osoitteesta
http://www.tietosuoja.fi/fi/index/materiaalia/lomakkeet/rekisteri-jatietosuojaselosteet.html

Walker G (1979). Wiegand's wonderful wires. *Popular Science* 214 (5), 101-104 ja 165. Haettu 6.11.2016 osoitteesta https://books.google.fi/books?id=iwEAAAAAMBAJ&pg=PA102&lpg=PA102&d#v=onepage&q&f=false

Wigen P (1975). Wiegand wire: new material for magnetic-based devices. *Electronics, New York* 48 (14), 100-105. Haettu 6.11.2016 osoitteesta http://www.epanorama.net/sff/Power%20Electronics/Transformers_and_Magnetics/Magnetics%20-%20Wiegand%20Wire.pdf

Liite 1 – Kulunvalvontalaitteen piirikaavio



Liite 2 – ASCII-taulukko

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*
10	A	1010	12	*[LINE FEED]*
11	B	1011	13	*[VERTICAL TAB]*
12	C	1100	14	*[FORM FEED]*
13	D	1101	15	*[CARRIAGE RETURN]*
14	E	1110	16	*[SHIFT OUT]*
15	F	1111	17	*[SHIFT IN]*
16	10	10000	20	*[DATA LINK ESCAPE]*
17	11	10001	21	*[DEVICE CONTROL 1]*
18	12	10010	22	*[DEVICE CONTROL 2]*
19	13	10011	23	*[DEVICE CONTROL 3]*
20	14	10100	24	*[DEVICE CONTROL 4]*
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*
22	16	10110	26	*[SYNCHRONOUS IDLE]*
23	17	10111	27	*[ENG OF TRANS. BLOCK]*
24	18	11000	30	*[CANCEL]*
25	19	11001	31	*[END OF MEDIUM]*
26	1A	11010	32	*[SUBSTITUTE]*
27	1B	11011	33	*[ESCAPE]*
28	1C	11100	34	*[FILE SEPARATOR]*
29	1D	11101	35	*[GROUP SEPARATOR]*
30	1E	11110	36	*[RECORD SEPARATOR]*
31	1F	11111	37	*[UNIT SEPARATOR]*
32	20	100000	40	*[SPACE]*
33	21	100001	41	**!**
34	22	100010	42	**"**
35	23	100011	43	**#**
36	24	100100	44	**$**
37	25	100101	45	**%**
38	26	100110	46	**&**
39	27	100111	47	**'**
40	28	101000	50	**(**
41	29	101001	51	**)**
42	2A	101010	52	*****
43	2B	101011	53	**+**
44	2C	101100	54	**,**
45	2D	101101	55	**-**
46	2E	101110	56	**.**
47	2F	101111	57	**/**
48	30	110000	60	**0**
49	31	110001	61	**1**
50	32	110010	62	**2**
51	33	110011	63	**3**
52	34	110100	64	**4**
53	35	110101	65	**5**
54	36	110110	66	**6**
55	37	110111	67	**7**
56	38	111000	70	**8**
57	39	111001	71	**9**
58	3A	111010	72	**:**
59	3B	111011	73	**;**
60	3C	111100	74	**<**
61	3D	111101	75	**=**
62	3E	111110	76	**>**
63	3F	111111	77	**?**
64	40	1000000	100	**@**
65	41	1000001	101	**A**
66	42	1000010	102	**B**
67	43	1000011	103	**C**
68	44	1000100	104	**D**
69	45	1000101	105	**E**
70	46	1000110	106	**F**
71	47	1000111	107	**G**
72	48	1001000	110	**H**
73	49	1001001	111	**I**
74	4A	1001010	112	**J**
75	4B	1001011	113	**K**
76	4C	1001100	114	**L**
77	4D	1001101	115	**M**
78	4E	1001110	116	**N**
79	4F	1001111	117	**O**
80	50	1010000	120	**P**
81	51	1010001	121	**Q**
82	52	1010010	122	**R**
83	53	1010011	123	**S**
84	54	1010100	124	**T**
85	55	1010101	125	**U**
86	56	1010110	126	**V**
87	57	1010111	127	**W**
88	58	1011000	130	**X**
89	59	1011001	131	**Y**
90	5A	1011010	132	**Z**
91	5B	1011011	133	**[**
92	5C	1011100	134	****
93	5D	1011101	135	**]**
94	5E	1011110	136	**^**
95	5F	1011111	137	**_**
96	60	1100000	140	**`**
97	61	1100001	141	**a**
98	62	1100010	142	**b**
99	63	1100011	143	**c**
100	64	1100100	144	**d**
101	65	1100101	145	**e**
102	66	1100110	146	**f**
103	67	1100111	147	**g**
104	68	1101000	150	**h**
105	69	1101001	151	**i**
106	6A	1101010	152	**j**
107	6B	1101011	153	**k**
108	6C	1101100	154	**l**
109	6D	1101101	155	**m**
110	6E	1101110	156	**n**
111	6F	1101111	157	**o**
112	70	1110000	160	**p**
113	71	1110001	161	**q**
114	72	1110010	162	**r**
115	73	1110011	163	**s**
116	74	1110100	164	**t**
117	75	1110101	165	**u**
118	76	1110110	166	**v**
119	77	1110111	167	**w**
120	78	1111000	170	**x**
121	79	1111001	171	**y**
122	7A	1111010	172	**z**
123	7B	1111011	173	**{**
124	7C	1111100	174	**|**
125	7D	1111101	175	**}**
126	7E	1111110	176	**~**
127	7F	1111111	177	*[DEL]*

http://www.hki.uni-koeln.de/node/14691

REKISTERISELOSTE
Henkilötietolaki (523/1999) 10 §

Lue täyttöohjeet ennen rekisteriselosteen täyttämistä. Käytä tarvittaessa liitettä.

Laatimispäivä: **7.5.2016**

1a Rekisterinpitäjä	Nimi: Oksanen Sami Tapani Osoite: [redacted] Muut yhteystiedot (esim. puhelin virka-aikana, sähköpostiosoite): [redacted]
2 Yhteyshenkilö rekisteriä koskevissa asioissa	Nimi: Sami Oksanen Osoite: [redacted] Muut yhteystiedot (esim. puhelin virka-aikana, sähköpostiosoite): [redacted]
3 Rekisterin nimi	Virolan puutarhan kulunvalvonta
4 Henkilötietojen käsittelyn tarkoitus	Tietojen käsittelyn tarkoitus on työajan valvonta, sekä kulunvalvonta.
5 Rekisterin tietosisältö	Rekisteri sisältää tiedot ulko-oven, sekä kääntöportin avauksista ja avausyrityksistä. Rekisteri sisältää myös etäluettaviin avaimien numeroihin linkitetyt nimet, sekä heidän kulkutietonsa, joiden avulla työaikalistat koostetaan.
6 Säännönmukaiset tietolähteet	Tietolähteinä toimivat RFID lukijat ulko-oven ulkopuolella, sekä kääntöportin molemmilla puolilla.

REKISTERISELOSTE 2

7 Tietojen sään-nönmukaiset luovutukset	Kerätyt tiedot luovutetaan työntekijöille palkkakuittien yhteydessä, sekä tarvittaessa pyynnöstä. Tietoja ei luovuteta ulkopuolisille.
8 Tietojen siirto EU:n tai ETA:n ulkopuolelle	Tietoja ei siirretä EU:n tai ETA:n ulkopuolelle.
9 Rekisterin suojauksen periaatteet	A Manuaalinen aineisto Tietoja säilytetään paperilla vain palkkakuitteja varten, tai erikoistapauksissa, jolloin tulosteita ei säilytetä esillä ja niistä huolehditaan varovaisesti. B ATK:lla käsiteltävät tiedot Tietoja säilytetään tietokoneella, valvotussa toimistossa. Tietokone ei ole yhdistettynä tietoverkkoihin.