Samu Kiviniitty

Päivityssovellus

Exertus Oy:n Linux moduuleille

Opinnäytetyö
Kevät 2019
SeAMK Tekniikka
Tietotekniikan koulutusohjelma

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan Yksikkö

Tutkinto-ohjelma: Tietotekniikan koulutusohjelma

Suuntautumisvaihtoehto: Sulautetut järjestelmät

Tekijä: Samu Kiviniitty

Työn nimi: Päivityssovellus: Exertus Oy:n Linux moduuleille

Ohjaaja: Hietamäki Marko

Vuosi: 2019 Sivumäärä: 30

Työn aiheena oli tehdä Exertus Oy:lle yksinkertainen päivityssovellus Linux-käyttöjärjestelmällä oleville moduuleille. Sovellusta on tarkoitus pystyä muokaamaan eri asiakkaiden vaatimuksiin.

Työssä käydään läpi Linux-käyttöjärjestelmää ja eri ohjelmointikieliä, joita työssä on käytetty. Näitä ovat muun muassa Bash- ja C-kieli. Lisäksi käydään läpi Exertus Oy:n omaa ohjelmointi- ja suunnittelutyökalua nimeltä Guitu.

Työn lopussa käydään läpi itse päivityssovellusta, mitä vaatimuksia sille asetettiin yrityksen toimesta ja miten päivityssovellus toteutettiin alusta loppuun eri vaiheiden kautta. Vaiheet olivat suunnittelu, vaihtoehtojen vertailu, graafisen käyttöliittymän toteutus, Guitu-ohjelmalla ohjelmointi ja generic-skriptin muokkaaminen halutunlaiseksi, jotta päivityssovellusta päästiin testaamaan.

Avainsanat: Linux, Bash, C, Guitu

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Embedded System

Author: Samu Kiviniitty

Title of thesis: Software Updater for the Linux Modules of Exertus Oy

Supervisor: Hietamäki Marko

Year: 2019 Number of pages: 30

The subject of the thesis was to make Exertus Oy a simple software updating tool for Linux operating system modules. The software should be customisable to the requirements of different clients.

The thesis studied the Linux operating system and the different programming languages used in the thesis, including Bash and C language. Also, Exertus Oy's own programming and design tool called Guitu was studied.

At the end of the thesis, the Software Updater itself was reviewed. Attention was paid to the requirements set by the company and how the Software Updater was implemented from the beginning. The steps were following: design, comparison of options, implementation of the graphical user interface, coding with Guitu and editing the Generic script so that it can be used to test the Software Updater.

Keywords: Linux, Bash, C, Guitu

SISÄLTÖ

Kuvaluettelo

Käytetyt termit ja lyhenteet

Moduuli	Exertus Oy:n valmistama tuote
cp -rf	Komentorivi komento. ”cp” kopioi tiedoston haluttuun paikkaan ja ”-rf” on lisä määrityksiä kopioinnille. r = recurse, f = overwrite.
mv -f	Komentorivi komento. ”mv” siirtää tiedoston haluttuun paikkaan ja ”-f” on lisä määritys siirtämiselle. f = do not prompt before overwriting.
time	Komentorivi komento. Kun ”time” lisätään, jonkin komennon eteen saadaan komentoon mennyt aika näkyviin komentoriville suorituksen jälkeen.
Bash	Bourne again shell.
Guitu	Exertus Oy:n kehittämä ohjelma.
Pinni	Moduulin liittimessä oleva yksittäinen kontakti.
Eventti	Tapahtuma esim. näppäin painallus.
Muuttujapuu	Muuttujista koostuvaa puu. Muuttujapuusta selviää muuttujien äiti-lapsi suhteet.
CANopen	Tietoliikenneprotokolla ja laiteprofiilin määrittely automaatiossa käytettäville sulautetuille järjestelmille.
J1939	Moottoriväylän standardi.

1 JOHDANTO

1.1 Työn tausta

Päivityssovelluksen avulla on mahdollista päivittää Exertus Oy:n Linux-käyttöjärjestelmällä toimia moduuleita. Päivityssovellus mahdollistaa Linux-moduulien päivittämisen helposti graafisen käyttöliittymän kautta, jossa käyttäjä pystyy itse valitsemaan erilaisista päivityksistä juuri haluamansa. Hyvällä päivityssovelluksella käyttäjä tietää koko ajan, mitä on päivittämässä graafisen käyttöliittymän ansiosta.

Exertus Oy kehittää jatkuvasti asiakkaittensa sovelluksia ja omia moduuleitaan vastaamaan uusia ominaisuuksia. Tämä tarkoittaa, että moduulien sovelluksia tarvitsee päivittää kentällä. Kentällä päivitys tapahtuu asiakkaan toimesta, jolloin päivityssovelluksen tarvitsee olla selkeä, yksinkertainen ja sopivasti informatiivinen päivityksen etenemisestä.

1.2 Työn tavoite

Työn tavoitteena on tehdä yritykselle yleispätevä päivityssovellus, jolla helpotetaan asiakkaiden sovellusten päivittämistä. Työn tuloksena syntyvän päivityssovelluksen tarkoitus on olla pohjaratkaisu. Päivityssovellusta voidaan jatkossa hyödyntää eri asiakkaiden tarpeiden mukaan muokkaamalla graafinen ulkoasu vastaamaan asiakkaan mieltymyksiä. Päivitysominaisuuksia voidaan lisätä, poistaa tai muokata vastaamaan asiakkaan vaatimuksia.

1.3 Työn rakenne

Työn rakenne koostuu seuraavista asioista, jotka ovat vaatimusten määrittely, Linux-käyttöjärjestelmään perehtyminen, C- ja Bash-ohjelmoinnin perusteisiin perehtyminen, Guitu-ohjelmaan tutustuminen, yleinen suunnittelu, sovelluksen ulkoasun suunnittelu, sovelluksen toiminnan suunnittelu, laitteiden hankinta ja kokonaisuuden testaus. Työn lopussa on yhteenveto työstä ja tuloksista.

1.4 Exertus Oy

Exertus Oy on vuonna 2003 perustettu riippumaton palveluyritys, jonka päätavoite on auttaa asiakkaitaan tekemään omista tuotteistaan älykkäämpiä (Exertus Oy 2018).

Exertuksen ydinosaaminen liittyy CAN-väylällä hajautettujen koneenohjausjärjestelmien kokonaisvaltaiseen hallintaan. Palvelukonsepti kattaa Exertuksen elektroniikkatuotteiden päälle rakennettujen järjestelmien esisuunnittelun, konsultoinnin, määrittelyn, suunnittelun, toteutuksen ja koulutuksen sekä ylläpidon. (Exertus Oy 2018.)

Verkostoitumalla alan yritysten ja yhteisöjen kanssa Exertus pyrkii saamaan asiakkaiden käyttöön parhaan mahdollisen asiantuntemuksen sekä teknisesti ja taloudellisesti optimaaliset ratkaisut. Exertus tekee yhteistyötä asiakkaan kanssa läpi koko kehitysprosessin. Exertus toimii usein asiakkaan ulkoistettuna tuotekehitysosastona tai sen osana. (Exertus Oy 2018.)

2 LINUX-KÄYTTÖJÄRJESTELMÄ

Linux on käyttöjärjestelmä, jonka kehitti vuonna 1991 Linus Torvalds. Linux-nimi viittaa Linux-ydintä käyttävien Unixin kaltaisten käyttöjärjestelmien perheeseen. Linux-ydintä käytetään yhdessä vapaiden ja avoimien lähdekoodien kanssa. Linux on vapaasti muokattava ja hyvin käytännöllinen käyttöjärjestelmä. (Kothari ym. 2011, 4.)

Linux-käyttöjärjestelmiä on monia erilaisia eri käyttötarkoitukseen tarkoitettuja. Alla on listattuna erilaisia käyttöjärjestelmiä:

- Yhden käyttäjän käyttöjärjestelmät
- Monen käyttäjän käyttöjärjestelmät
- Yhden tehtävän käyttöjärjestelmät
- Monen tehtävän käyttöjärjestelmät
- Reaaliaika käyttöjärjestelmä
- Hajautettu käyttöjärjestelmä
- Sulautetut järjestelmät. (Kothari ym. 2011, 2-3.)

Linux on avointa lähdekoodia eli se on ilmainen. Avoimen lähdekoodin ohjelmistojen tarkoituksena on tuoda lähdekoodit käyttäjien käyttöön ilmaiseksi. Tämä antaa ohjelmoijille mahdollisuuden lisätä, parantaa tai muuttaa koodia. Linux-käyttöjärjestelmä ei ole yksinään avointa lähdekoodia, vaan suurin osa sen sovellusohjelmistosta on myös avointa lähdekoodia. Linux-käyttöjärjestelmä kasvaa nopeasti avoimen lähdekoodinsa ansiosta. Linux-käyttöjärjestelmästä on versioita, jotka eivät ole avointa lähdekoodia. Tämä tarkoittaa, että niitä ei ole saatavilla. (Fox 2014, chapter 1.1.)

3 OHJELMOINTIKIELET

Ohjelmointikielet ovat kieliä, joilla ohjelmoidaan esimerkiksi sovelluksia. Ohjelmointikieliä on paljon erilaisia. Osa ohjelmointikielistä on tehty vain tiettyihin tarkoituksiin, kun taas toiset ovat yleiskäyttöisempiä ja niitä voi käyttää laajemmin erilaisissa projekteissa/sovelluksissa.

Tässä työssä käytettiin vain kahta ohjelmointikieltä ja ne olivat Bash ja perinteinen C. C-ohjelmointikieli toimii Exertus Oy:n Guitu-ohjelmointisovelluksen pohjalla.

3.1 Bash (Bourne again shell)

Shell on standardi käyttöliittymä kaikille Unix- ja Linux-järjestelmille. Shell on oikeastaan täysi ohjelmointikieli, joka sisältää muuttujat ja funktiot ja paljon muitakin kehittyneitä struktuureja, kuten taulukot. (Parker 2011, 29.)

Bourne Again Shell (Bash) on komentojen tulkki ja korkean tason ohjelmointikieli. Komennon tulkkina Bash käsittelee komentorivillä annettuja komentoja vastauksena kehotteeseen. Kun käytetään shell-ohjelmointikieltä, se käsittelee komentoja, jotka on tallennettu shell-skripteihin. Muiden kielten tavoin myös shellissä on muuttujia ja ohjausvirtakäskyjä (esim. for-silmukoita ja if-lauseita). (Helmke & Sobell 2018, chapter 8.)

Bourne Again Shell (bash) on kiistatta yksi tärkeimmistä olemassa olevista komentotulkeista. Ilman bash-shellin monia apuohjelmia ja ongelmanratkaisupotentiaalia. Apuohjelmat, kuten grep, wget, vi ja awk, mahdollistavat käyttäjilleen erittäin tehokkaan merkkikäsittelyn, tiedonlouhinnan ja tiedonhallinnan. Järjestelmänvalvojat, kehittäjät, tietoturvainsinöörit ja haavoittuvuustestaajat ympäri maailmaa ovat jo vuosia vannoneet sen ongelmanratkaisupotentiaalin ja tehokkuuden avulla, että he voivat käsitellä päivittäisiä teknisiä haasteitaan. (Makan 2014, chapter 1.)

Kun aloitetaan kirjoittamaan shell-scripti tiedostoa. Sen tiedoston alkuun pitää määritellä, mitä tulkkia käytetään. Bash-formaatti on seuraavanlainen ”#!/bin/bash”. (Blum & Bresnahan 2014, 270.)

3.2 C-ohjelmointikieli

C on yleiskäyttöinen ohjelmointikieli. Se on liitetty tiiviisti UNIX-järjestelmään, jossa se on kehitetty, koska sekä järjestelmä että useimmat ohjelmassa toimivat ohjelmat on kirjoitettu C-kielellä. Sitä on kutsuttu ”järjestelmäohjelmointikieleksi”, koska se on hyödyllinen kääntäjien ja käyttöjärjestelmien kirjoittamiseen, sitä on käytetty suurten ohjelmien kirjoittamiseen. (Rithcie & Kernighan 1988, introduction.)

1980-luvun alussa havaittiin tarvetta yhtenäistää C-kielen määritelmä. American National Standards Institute (ANSI) on organisaatio, joka käsittelee tällaisia asioita, joten vuonna 1983 C-kielen standardoimiseksi muodostettiin ANSI C -komitea (nimeltään X3J11). Vuonna 1989 valiokunnan työ ratifioitiin, ja vuonna 1990 ensimmäinen virallinen ANSI C -standardin määritelmä julkaistiin. (Kochan 2004, chapter 1.)

Koska C-kieltä käytetään ympäri maailmaa, International Standard Organization (ISO) aktivoitui. Yhteistyössä luotiin standardi ISO / IEC 9899: 1990. Siitä lähtien C-kielelle on tehty lisää muutoksia. Viimeisin standardi hyväksyttiin vuonna 1999. Se tunnetaan nimellä ANSI C99 tai ISO / IEC 9899: 1999. (Kochan 2004, chapter 1.)

Monet C-kielen tärkeistä ajatuksista ovat Martin Richardsin kehittämän BCPL-kielen. BCPL-kielen vaikutus C-kieleen eteni epäsuorasti B-kielen kautta, jonka Ken Thompson kirjoitti vuonna 1970 ensimmäiselle DEC PDP-7 UNIX -järjestelmälle. (Rithcie & Kernighan 1988, introduction.)

BCPL ja B ovat "kirjoittamattomia" kieliä. Sitä vastoin C tarjoaa erilaisia tietotyyppejä. Tärkeimmät tyypit ovat merkkejä ja kokonaislukuja. Lisäksi on olemassa johdettujen datatyyppien hierarkia, joka on luotu osoittimien, taulukoiden, rakenteiden ja liittojen avulla. Lausekkeet muodostetaan operaattoreista ja operandeista. Mikä tahansa lauseke, mukaan lukien tehtävä tai funktiopuhelu, voi olla lauseke. Osoitin tarjoaa koneen itsenäisen osoitearitmeettisen osoitteen. (Rithcie & Kernighan 1988, introduction.)

C-kieli tarjoaa perusrakenteiseen ohjelmointiin tarvittavat perusvirtausrakenteet: lausuntojen ryhmittely, päätöksenteko (if-else), yksi mahdollisten tapausten joukosta (switch), silmukointi päätetestin kanssa yläosassa (while, for) tai alareunassa

(do) ja varhaisen silmukasta poistumisen (break) (Rithcie & Kernighan 1988, introduction.)

4 GUITU

Guitu on Exertus Oy:n kehittämä ohjelmointityökalu, jolla voidaan suunnitella ja toteuttaa käyttöliittymiä ja myös ohjelmoida koodia käyttöliittymän taustalle. Guitulla on siis mahdollista toteuttaa kokonaisia sovelluksia helposti

4.1 Ohjelmointi

Guitu perustuu C-kieleen. C-kielestä löytyy uudempiin C++- ja C#-kieliin verrattuna vain perusfunktiot kirjastoina.

Exertus on kehittänyt itse joitain kirjastoja Guitulle mitä C-kieli ei oletuksena omista.

Guitulla ohjelmointi perustuu funtion block diagram -ohjelmointiin, joka on käyttäjälle helpompaa. Käyttäjä vain valitsee haluamansa tyylisiä blokkeja ja yhdistelee niitä viivoilla. Ainoa isompi ongelma tässä on se, että Guitu ei sisällä aina välttämättä sellaista blokkeja. joita ohjelmoija tarvitsee. Tällöin sovelletaan olemassa olevia blokkeja, jotta saadaan tehtyä haluttu koodi.

4.2 Ulkoasu

Ulkoasultaan Guitu on ensikertalaiselle ehkä jonkin verran sekava, mutta kaikki suunnittelu- ja ohjelmointiohjelmat voivat olla alkuun sekavia, kun niistä löytyy paljon erilaisia ominaisuuksia ja toimintoja.

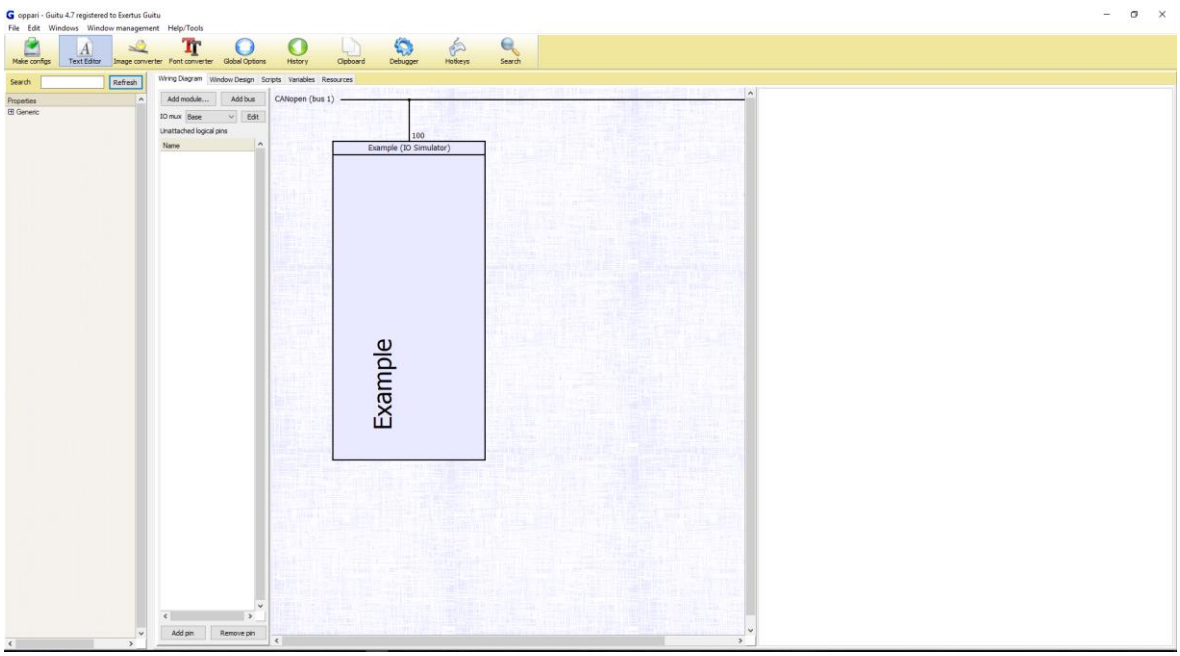
Guitu-ohjelmasta löytyy välilehtiä, joilla on jaettu ohjelmassa eri asiat selkeästi omiin paikkoihinsa, vaikka kokeneempi Guitu-käyttäjä käyttääkin kaikkia välilehtiä yhdessä tehdessään sovellusta.

4.3 Välilehdet

Guitusta löytyy välilehtiä. Välilehtiä ovat Wiring diagram, Window design, Scripts, Variables ja Resources. Jokaisen välilehden alle on sijoitettu välilehden nimeen liittyvät asiat. Välilehdet helpottavat Guitun käyttöä, kun kerralla näytetään vain tiettyjä asioita käyttäjälle.

4.3.1 Wiring diagram

Wiring diagram -välilehden alta löytyy mahdollisuuksia rakentaa Exertus Oy:n omia moduuleita hyödyntäen kytkentää. Se määrittää minkälaisia moduuleita sovellukseen kuuluu ja miten moduulit on kytketty toisiinsa, minkälaisia väyliä moduuleilta löytyy ja väylille pystyy määrittelemään mihin tarkoitukseen niitä käytetään. Erilaisia käyttötarkoituksia väylille on esimerkiksi CANopen tai J1939. Valituille moduuleille voidaan määritellä pinnejä tässä välilehdessä. Pinnien määrittelylle on tietenkin rajoituksia, jotka tulevat fyysisen moduulin mukaan. Tietyllä moduulilla on tiettyjä pinnejä, niitä ei voida määritellä väärin, jotta ne toimivat. Kuvassa 1 näkyy Guitu-ohjelman Wiring diagram -välilehti.



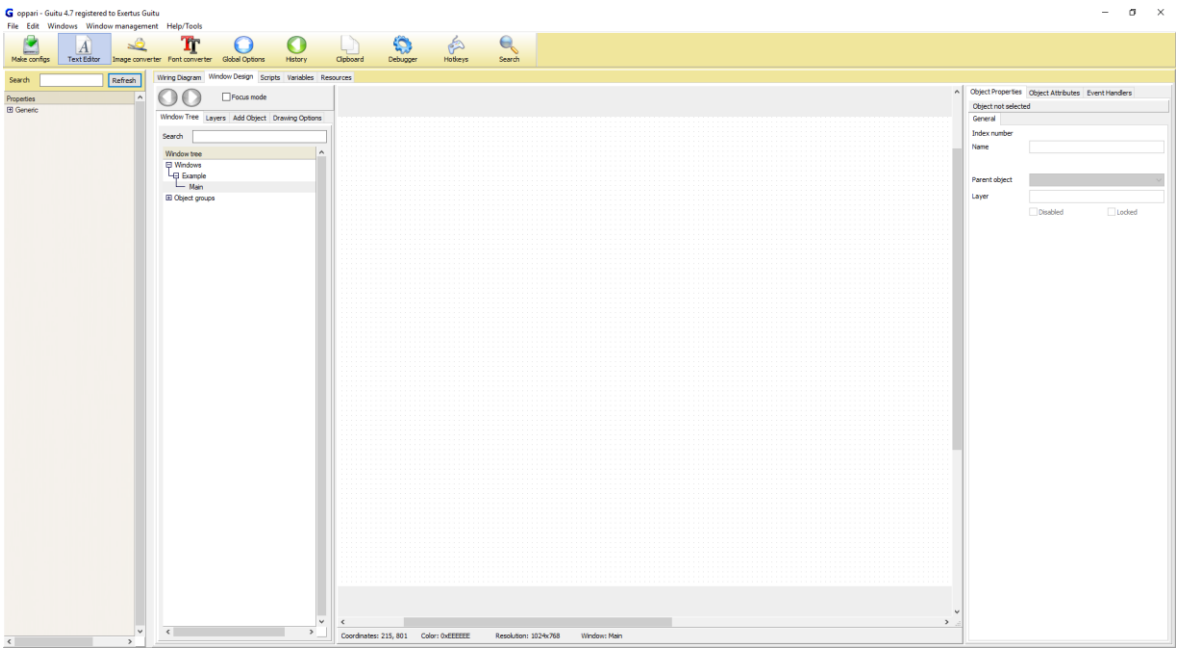
Kuva 1.Guitu Wiring diagram

4.3.2 Window design

Window design -välilehdellä käyttäjä pääsee suunnittelemaan ja toteuttamaan käyttöliittymää sovellukselleen. Käyttöliittymän rakentaminen perustuu ohjelmasta löytyviin objekteihin, joita lisätään ikkunaan, ja joille määritetään haluttuja määrityksiä. Objekteille voi määritellä ajettavia skriptejä. Myös itse ikkunoille voidaan määritellä skriptejä ajoon eri tilanteisiin.

Käyttäjällä on myös mahdollisuus tehdä valmiista objekteista uusia omia objektiryhmiä, joita voi käyttää ikkunoissaan. Käyttäjä voi tehdä tavallisten ikkunoiden lisäksi pop-up-ikkunoita ja alaikkunoita.

Kaikille objekteille ja ikkunoille voidaan tehdä myös erilaisia eventtejä, esimerkiksi voidaan tehdä ikkunaan eventti, jossa tarkkaillaan, painaako käyttäjä siinä ikkunassa jotain nappia ja sen jälkeen tehdään käyttäjän määrittelemä toiminto, joka voi olla ikkunan vaihto tai skriptin ajo. Kuvassa 2 näkyy Guitu-ohjelman Window design -välilehti.

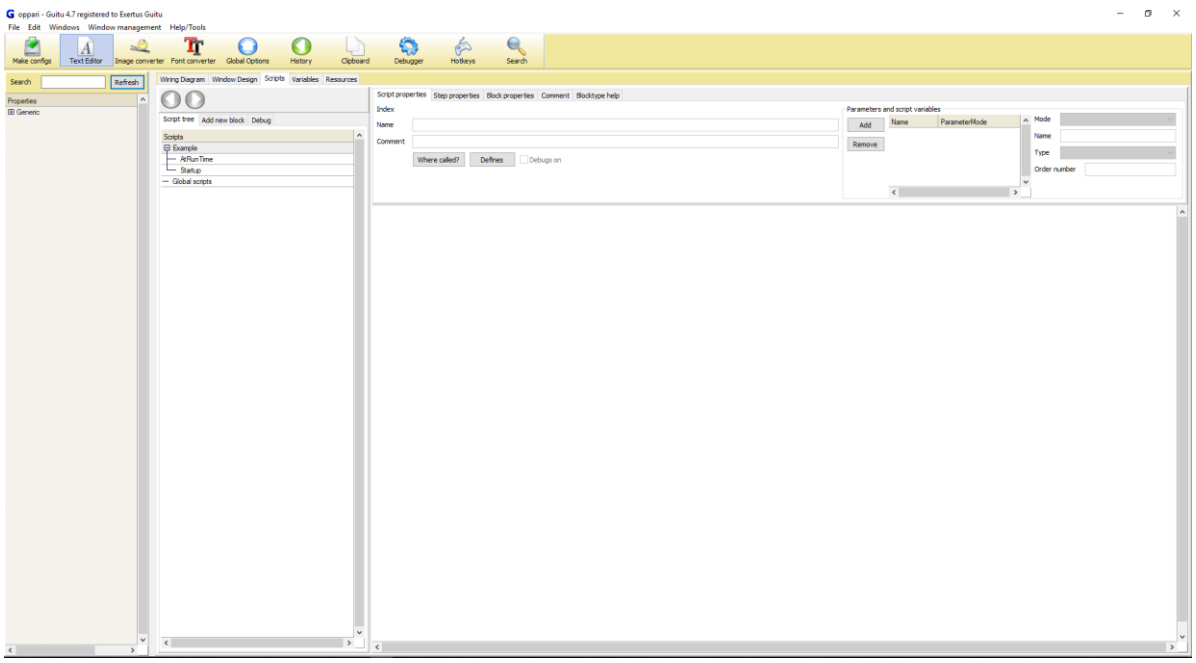


Kuva 2.Guitu Window design

4.3.3 Scripts

Scripts-välilehden alle käyttäjä tekee kaikki skriptit. Skriptit koostuvat askelista ja jokaisen askeleen sisältö ajetaan vasemmalta oikealla ja ylhäältä alas järjestyksessä. Skriptien tekemiseen ohjelmaan on tehty blokkeja, joita hyödyntäen käyttäjä pääsee toteuttamaan sovelluksensa koodia.

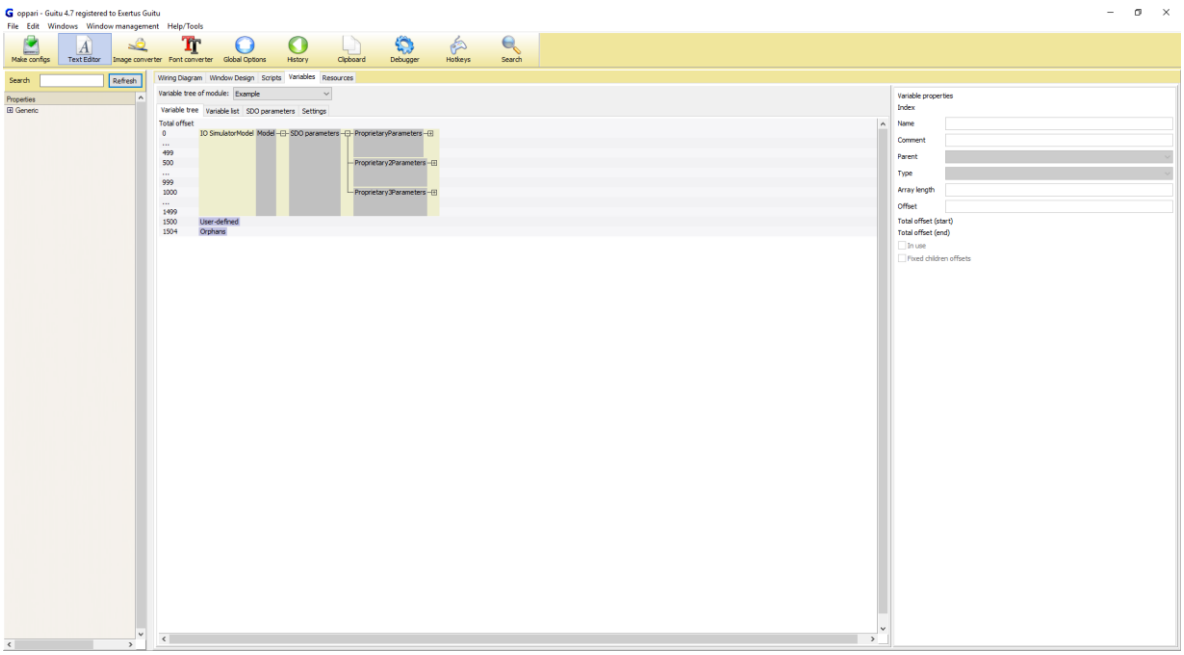
Skripteihin on Guituun tehty valmiita blokkeja. On tilanteita, jolloin käyttäjän haluamaa blokkia ei ole ja pääsee soveltamaan muita blokkeja, että saa aikaan haluamansa asian. Kuvassa 3 näkyy Guitu-ohjelman Scripts-välilehti.



Kuva 3.Guitu Scripts

4.3.4 Variables

Variables-välilehden takaa löytyy itse sovelluksen muuttujapuu. Muuttujapuun koostumukseen vaikuttaa se mitä käyttäjä on valinnut Wiring diagram -välilehdelle. Kuvassa 4 näkyy Guitu-ohjelman Variables-välilehti.



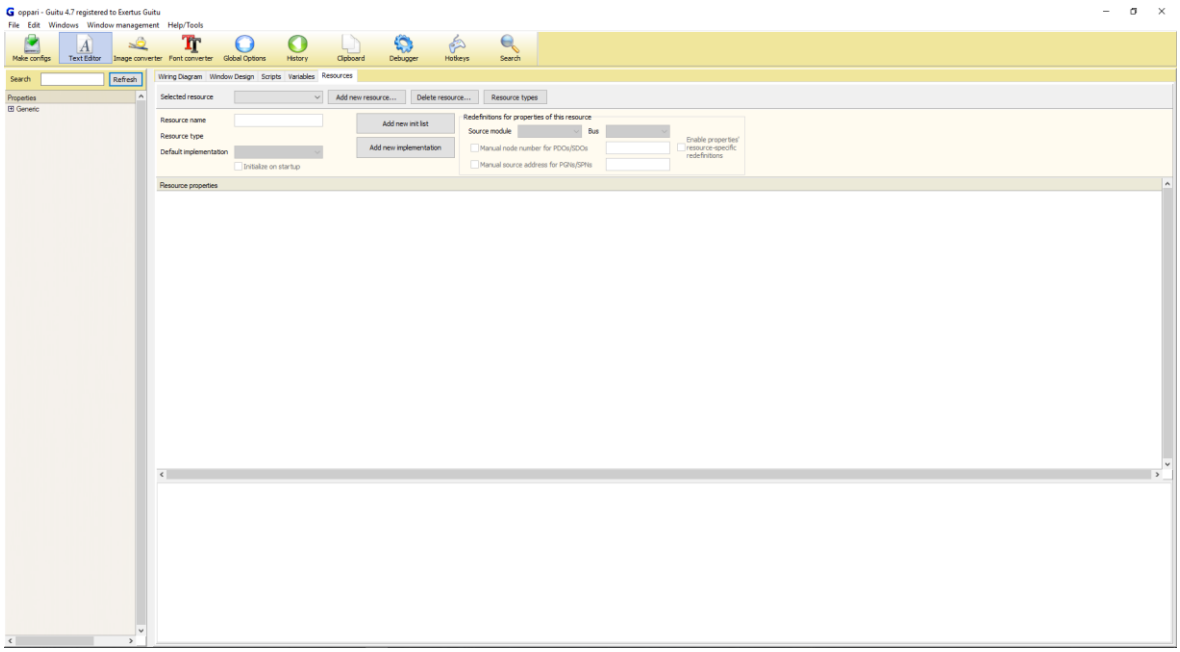
Kuva 4.Guitu Variables

Käyttäjä voi itse lisäillä muuttajapuuhun omia muuttujiaan tai käyttää siellä valmiiksi jokaiselle moduulille löytyviä muuttujia.

Guitu tukee kaikkia perinteisiä muuttujatyyppejä, joita C-kielestä löytyy.

4.3.5 Resources

Guitu-ohjelmasta löytyy myös Recources-välilehti, jossa voidaan tehdä erilaisia resursseja projekteihin. Tässä työssä ei ole käytetty resursseja. Resursseihin voi tehdä esimerkiksi erilaisia näppäinresursseja. Kuvassa 5 näkyy Guitu-ohjelman Resources-välilehti.



Kuva 5.Guitu Resources

5 PÄIVITYSSOVELLUS

Työn tarkoituksena on luoda Exertus Oy:lle päivityssovellus Linux-käyttöjärjestelmää käyttäville moduuleille. Päivityssovelluksen pitää pysyä yksinkertaisena ja sitä pitää pystyä helposti muokkaamaan eri asiakkaiden toiveiden mukaiseksi.

Päivityssovelluksen pitää pystyä päivittämään itse sovellus ja myös firmware. Lisäominaisuuksina päivityssovelluksella olisi hyvä saada päivitettyä tarvittaessa parametrit ja kielitiedosto moduulille.

5.1 Vaatimukset

Päivityssovellukselle asetettiin tietyt vaatimukset Exertus Oy:n toimesta ja ne olivat seuraavat: Käyttöliittymän pitää olla yksinkertainen ja helposti muokattavissa eri asiakkaiden mieltymyksiä vastaamaan. Päivityssovelluksella pitää pystyä päivittämään sovellus, firmware, parametrit ja kielitiedosto. Tarvittaessa asiakaskohtaisesti sovellukseen pitää pystyä lisäämään ja poistamaan erilaisia päivitysvaihtoehtoja.

Päivityssovelluksen olisi hyvä olla myös hyvin selkeä ja käyttäjän pitää koko ajan tietää mitä tapahtuu. Päivityssovellus ei saa vaikuttaa toisten sovellusten toimintaan mitenkään.

5.2 Suunnittelu

Päivityssovelluksen suunnittelu lähti liikkeelle siitä, mitä käyttäjän pitää pystyä valitsemaan ja mitä tapahtuu käyttäjältä piilossa. Tästä aloitettiin käyttöliittymän suunnittelu.

Yksinkertaista toteutusta miettien ajatuksena käyttöliittymään tulisi vain 2 nappia ja näiden nappien lisäksi tulisi vain valinta, mitä käyttäjä haluaa päivittää moduulille. Käyttöliittymään olisi myös hyvä tuoda esille moduulissa asennettujen versioiden

tiedot ja päivitysversioiden tiedot. Näin käyttäjä näkee heti, onko moduulissa jo oikea versio ja onko hän päivittämässä oikeata versiota moduulille vai onko päivityssovelluksen kansioihin jäänyt väärä versio.

Koodissa käytetään Linux-käyttöjärjestelmän tukemia komentorivikomentoja, joilla voidaan kopioida tiedostot oikeaan paikkaa, josta moduuli seuraavalla käynnistyksellä osaa käynnistää päivitetyn sovelluksensa.

Päivityssovelluksen pitäisi pystyä toimimaan täysin omana prosessina moduulilla. Päivityssovellusta ei tarvitse asentaa moduulille vaan se käynnistettäisiin USB-tikulta automaattisesti, jos sovellus/moduuli sen sallii.

Päivityssovelluksen toimintaperiaate on sellainen, että kun USB-tikku kytketään moduuliin, niin moduulissa oleva asennusskripti kopioi USB-tikulta ennalta määritetyt tiedostot omaan tmp-hakemistoonsa, josta päivityssovellus käynnistettäisiin.

Yrityksen tietoturvaasiat on rajattu opinnäytetyön ulkopuolelle. Näin halutaan välttää, etteivät Exertus Oy:n sertifikaatit ym. pääsisi yleiseen levitykseen.

5.3 Vaihtoehtojen vertailu

Työtä tehdessä tuli eteen erilaisten vaihtoehtojen vertailua, joista isoimpana nousi esille kysymys, jos päivityksessä tapahtuukin jotain ennalta arvaamatonta ja päivitetty sovellus ei toimikkaan. Työssä pitää ottaa huomioon myös päivitykseen kuluva aika. Päivityksen pitää tapahtua nopeasti, jotta se antaa hyvän vaikutelman asiakkaalle.

Alapuolella kaksi eri vaihtoehtoa miten päivitysohjelman kopiointi voidaan suorittaa.

- Kopioidaan suoraan USB-tikulta sovelluksen kansioon.
- Kopioidaan tmp-kansioon, josta kopioidaan sovelluksen kansioon

Jälkimmäinen vaihtoehto valittiin, koska sillä voidaan välttää päivityssovelluksen päivityksen aikana tapahtuvia häiriöitä USB-tikun liitoksessa tai jos käyttäjä vahingossa irrottaa USB-tikun liian aikaisin. Tiedostojen kopiointi tmp-kansioon ensin ja

sieltä vasta sovelluksen kansioon mahdollistaa sen, että USB-tikkua ei tarvitse pitää kytkettynä moduuliin, muuta kuin sen ajan, että päivityssovellus on käynnistynyt.

Seuraavana asiana tuli eteen se, jos kopiointi kestää liian kauan tai siinä tulee jokin ongelma vastaan. Kopiointiin ja siirtoon menevä aika testattiin Linux-moduulilla, jotta saatiin tietoon, kumpi on nopeampi tapa saada tiedostot toiseen kansioon. Linux-käyttöjärjestelmästä löytyy komento ”time”, joka kertoo toisen komennon suorittamiseen menevän ajan, kun komennon eteen lisätään ”time”. Esimerkki tiedostolla huomattiin, että tiedostojen siirtäminen oli huomattavasti nopeampaa kuin niiden kopiointi. Kuvassa 6 näkyy testi tiedoston kopioinnin ajat molemmilla komennoilla.

```
# time cp -rf test1/ test2
real    0m 23.15s
user    0m 0.00s
sys     0m 6.94s
# time mv -f test1/ test3
real    0m 0.02s
user    0m 0.00s
sys     0m 0.00s
```

Kuva 6.Kopioinnin ja siirtämisen aikavertailu

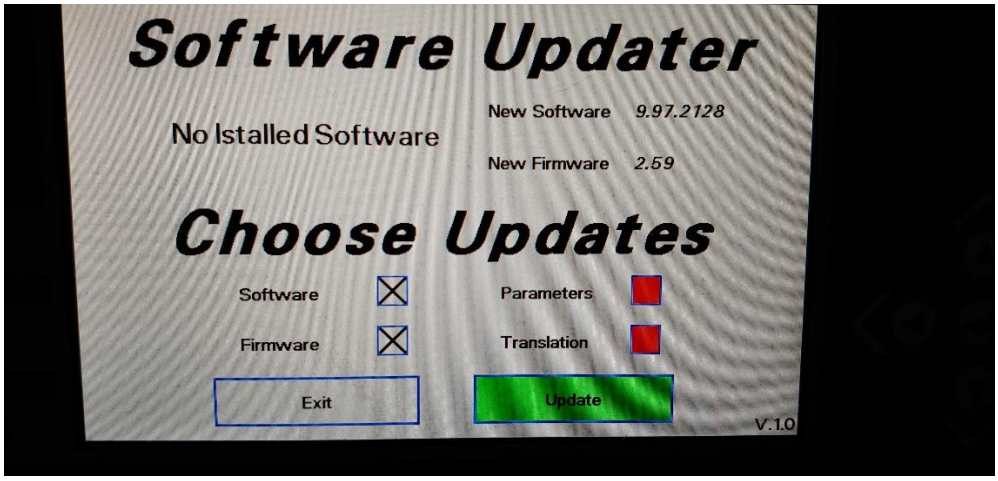
Päivityssovelluksessa käytetään kopioinnin sijaan siirtämistä. Tiedostojen siirtäminen oli huomattavasti nopeampaa, kuin kopiointi.

Suunnitteluvaiheessa tuli myös pohdittua, jos tiedostojen siirtäminen epäonnistuu ja päivitetty sovellus ei toimi. Pitkän pohdinnan ja erilaisten asioiden punnitsemisen jälkeen päädyttiin unohtamaan tämä mahdollinen ongelma, koska useamman testin jälkeen huomattiin, että tiedostojen siirtäminen onnistui aina testeissä. Pieni huomio oli, että USB-tikulta siirrettäessä tiedostoja Linux-moduulille, saattoi tapahtua synkronointiongelmia, mutta tämäkin asia kierretään kopioimalla tiedostot Linux-moduulin tmp-kansioon heti päivityssovelluksen käynnistyksessä.

5.4 Graafinen käyttöliittymä

Graafisesta käyttöliittymästä oli tarkoitus tehdä mahdollisimman yksinkertainen, mistä käyttäjä näkee helposti ja selkeästi ydinasiat päivityksestä. Käyttöliittymästä käyttäjän on pystyttävä helposti valitsemaan päivitettävät asiat.

Käyttöliittymään tulevia elementtejä tulisi olla mahdollisimman vähän, mutta kuitenkin sen verran, että kaikki tarpeellinen tulee näytettyä käyttäjälle. Valittuja elementtejä käyttöliittymään ovat vanha ja uusi versio sovelluksesta ja firmwaresta, eri päivitysten valinnat, peruuta- ja päivitä-painike ja viimeisenä elementtinä käyttöliittymään lisättiin päivityssovelluksen versio, jotta käyttäjä pystyy tulevaisuudessa näkemään mitä päivityssovellusta hän käyttää. Tämän päivityssovellusversion lisäyksen ideana oli se, että jos päivityssovellukseen tulee muutoksia, versiosta näkee suoraan mikä päivityssovellusversio on käytössä. Kuvassa 7 näkyy päivityssovelluksen päänäkymä.

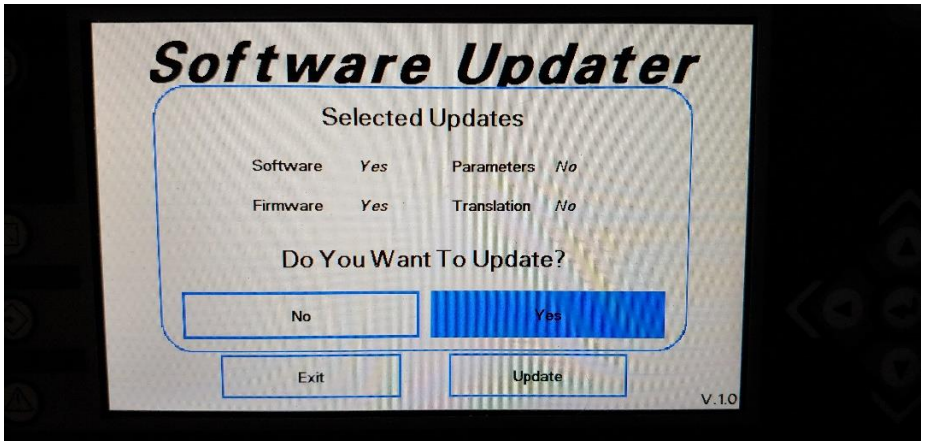


Kuva 7.Päivityssovelluksen päänäkymä

Ohjelman käynnistykseen lisättiin muutama pop-up-ikkuna kertomaan käyttäjälle asioita, kuten jos asennettu sovellus tai firmware on uudempi tai sama. Lisäksi kerrotaan, jos moduulissa ei ole sovellusta asennettuna. Tällöin päivitysohjelma kertoo sen ja valitsee automaattisesti kaikki mahdolliset päivitykset aktiivisiksi. Kuvassa 8 näkyy pop-up-ikkuna, joka avautuu näytölle, jos moduulissa ei ole valmiiksi asennettua sovellusta.

Kuva 8.Käynnistyksen pop-up-ikkuna, jos ei ole valmiiksi asennettua sovellusta

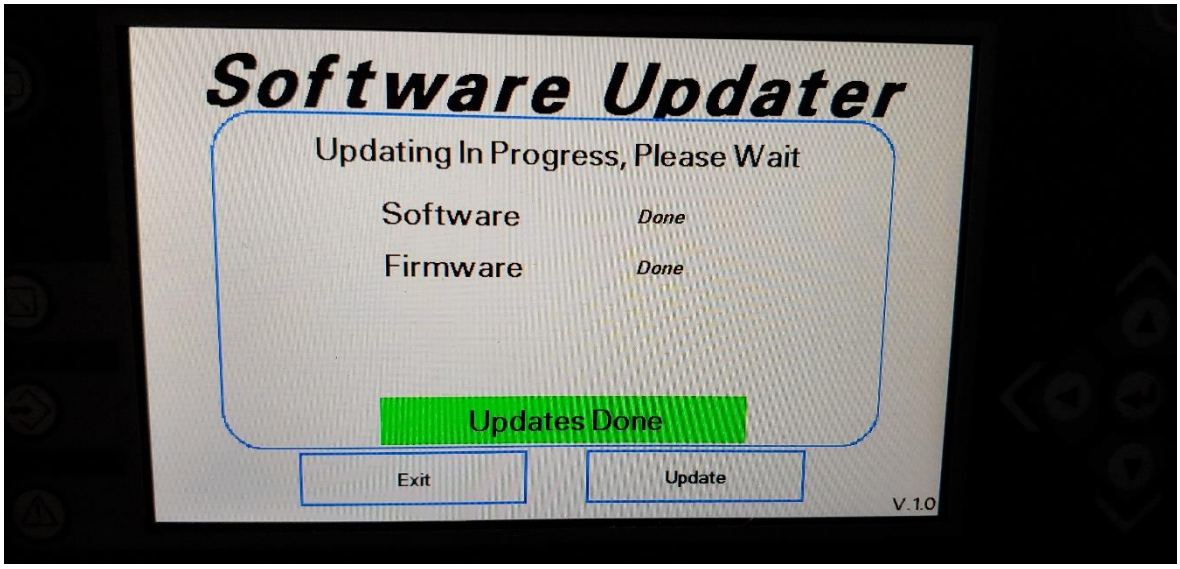
Pop-up-ikkunoita lisättiin myös muualle, jotta käyttäjälle voidaan näyttää enemmän tietoa päivityksestä ja käyttäjän valinnoista. Yksi pop-up-ikkuna lisättiin estämään vahinkopäivitykset. Tämä toteutettiin lisäämällä informaatio-pop-up päivityspainikkeen painallukseen eli kun käyttäjä painaa päivitä, niin tulee näkyviin pop-up-ikkuna, josta käyttäjä voi tarkistaa, mitä päivityksiä on valinnut. Tässä on vielä mahdollista perua päivitys, jos käyttäjä on valinnut vääriä päivityksiä tai unohtanut valita joitain päivityksiä aktiivisiksi. Kuvassa 9 näkyy pop-up-ikkuna, jossa käyttäjän on mahdollista vielä kerran tarkastaa, onko hän valinnut oikeat päivitykset.



Kuva 9.Päivityksen lisävarmistus-pop-up-ikkuna

Kun käyttäjä valitsee Yes-valinnan, päivityssovellus alkaa päivittämään valittuja päivityksiä moduulille ja päivittää moduulin näytölle avautuneeseen pop-up-ikkunaan tilannetta jatkuvasti, jotta käyttäjä tietää päivitysvaiheen ja sen että päivitys etenee

ilman ongelmia eteenpäin. Päivityssovellus myös ilmoittaa samassa pop-up-ikkunassa, jos jokin päivitys epäonnistuu. Kuvassa 10 näkyy pop-up-ikkuna, jossa päivityssovellus ilmoittaa käyttäjälle päivitysprosessin etenemisen.



Kuva 10.Päivitysprosessin pop-up-ikkuna

Päivitysprosessin pop-up-ikkuna sulkeutuu muutaman sekunnin kuluttua. Tämän jälkeen tulee uusi pop-up-ikkuna, jossa ilmoitetaan, onnistuivatko kaikki päivitykset tai mikä päivitys epäonnistui. Päivityssovellus myös ilmoittaa käyttäjälle, tulevatko päivitykset voimaan heti vai vasta moduulin uudelleen käynnistämisen jälkeen. Päivityssovellus jää odottamaan käyttäjältä OK-painikkeen painamista ennen kuin jatkaa eteenpäin. Kun käyttäjä painaa OK-painiketta päivityssovellus ilmoittaa sulkeutuvansa 5 sekunnin kuluessa.

5.5 Guitu-ohjelmakoodi

Guitu-koodi koostui pääasiassa käyttöliittymään liittyvistä eventeistä ja itse koodin puolella tiedostojen siirtelystä käyttäjän valinnan mukaan.

Koodi tehtiin ajettavaksi vain UI-säikeessä, mikä aiheutti sen, että koodiin piti lisätä käyttöliittymän päivityksiä tarpeeksi usein. Jos käyttöliittymä ei päivittyisi koko ajan, saattaisi se luoda käyttäjälle sellaisen kuvan sovelluksesta, että se ei toimi kunnolla.

Koodin skriptit voidaan jakaa eri ryhmiin kuten:

- Käynnistyksessä ajettavat skriptit
- Koko ajan ajettava skripti
- Käyttöliittymän eventit
- Päivitysskriptit

Skriptit, jotka ajetaan käynnistyksessä, ovat erilaisia tarkasteluita, esim. onko moduulille asennettu sovellusta vai ei. Jos on asennettu, niin mitkä ovat asennetun sovelluksen ja firmwaren versionumerot. Käynnistyksessä tarkastellaan myös mitä mahdollisia päivityksiä päivityssovellus voi toteuttaa ja mitä ei. Toisin sanoen, mitä tiedostoja päivityssovellus USB-tikulta löytyy, ja mitkä niistä ovat oikeanlaisia tiedostoja.

Koko ajan ajossa olevia skriptejä ei ole muita kuin käyttöliittymän päivittäminen.

Käyttöliittymässä olevia eventtejä ovat käyttäjän painikkeiden painallukset, joista ajetaan tietylle painikkeelle asetettu skripti eventin kautta. Eventeihin määriteltyjä skriptejä ovat päänäkymässä päivitysten valinnat, päivityksen aloittaminen tai Exit-painikkeesta päivityssovelluksen sammuttaminen.

Päivitysskriptit ovat skriptejä, jotka suorittavat valittujen päivitysten päivittämisen pääsääntöisesti siirtämällä uudet tiedostot vanhojen päälle. Joissakin tapauksissa päivitysskriptit luovat kansioita, linkkejä ja tiedostoja.

5.6 Generic-skriptin muokkaus

Guitu-ohjelma luo oletuksena muutaman skriptin, jotka liittyvät sovelluksen asentamiseen ja käynnistämiseen. Koska päivityssovellusta ei haluta asentaa moduulille niin näistä skripteistä tarvitsee muokata yhtä skriptiä, joka on niin sanottu generic-skripti, jonka tehtävänä on tarkistaa USB-tikulla olevat tiedostot, luoda sovelluskansio moduulille ja ajaa install-skripti, joka kopioi oikeat tiedostot moduulin sovelluskansioon. Viimeisenä asiana generic-skripti vielä käynnistää sovelluksen.

Tämä generic-skripti tarvitsee aika isoa muutosta, jotta saadaan päivityssovellus toimimaan halutulla tavalla ja ilman että oikealle sovellukselle käy mitään. Koska generic-skriptiä muokataan, aiheuttaa se tietoturvaongelmia, mutta näihin ongelmiin on ratkaisu, jota ei tuoda julki tässä työssä.

Isoimmat muutokset alkuperäiseen oletus-generic-skriptiin ovat tiedostojen kopiointiin liittyviä ja sovellusten käynnistämiseen liittyviä. Oletus-skriptissä käynnistetään lopuksi kaikki mahdolliset sovellukset, mikä ei ole hyvä asia, koska se käynnistää silloin myös moduulilla olevan oikean sovelluksen uudelleen, joka voi aiheuttaa vakaviakin seurauksia, jos sovellus on esimerkiksi ajossa ja ohjaa moottoria. Kaikkien sovellusten käynnistäminen muutettiin ainoastaan päivityssovelluksen käynnistykseksi.

Tiedostojen kopiointiin tehty muutos on aika yksinkertainen. Kaikki halutut tiedostot kopioidaan haluttuihin tmp-kansioihin, joista niitä on helppo hallita ja ne on helppo poistaa. Jos niiden poistamisessa tulee jokin ongelma tai vikatilanne, tmp-kansiot poistuvat automaattisesti, jos Linux-moduuli sammutetaan.

5.7 Päivityssovelluksen käyttö

Päivityssovelluksen toiminta on tehty hyvin yksinkertaiseksi käyttäjälle ja se sisältää seuraavia askelia:

- Moduulin käynnistäminen
- USB-tikun kytkeminen moduuliin
- Haluttujen päivitysten valitseminen
- Päivityksen aloittaminen
- Päivitysprosessi
- Päivitykset ovat valmiita ja tulevat voimaan vasta seuraavalla käynnistyksellä. Erinäisten turvallisuussyiden takia päivitykset eivät tule voimaan heti,

paitsi jos moduulilla ei ollut sovellusta, tällöin päivityssovellus käynnistää oikean sovelluksen automaattisesti.

- USB-tikun voi irrottaa koska tahansa, kunhan päivityssovellus on ehtinyt käynnistymään

Päivityssovelluksen käyttäminen on nopeaa ja helppoa. Käyttäjälle kerrotaan jokaisessa vaiheessa, mitä on tapahtumassa. Esimerkiksi päivityksen aikana käyttäjälle näytetään pop-up-ikkunalla, mikä päivitys on tehty, mikä on työn alla ja mitä on tulossa. Lopuksi vielä ilmoitetaan käyttäjälle, että kaikki päivitykset onnistuivat. Jos jokin päivitys ei onnistunut, siitä ilmoitetaan käyttäjälle.

6 TESTAUS

Testausta tehtiin kaikissa vaiheissa suunnittelusta aina valmiiseen työhön asti.

Suunnitteluvaiheessa testattiin eniten, miten käyttöliittymän grafiikat ja elementit sopivat yhteen ja minkälaisella asettelulla ne näyttävät selkeitä. Tavoitteena oli, että käyttäjä näkee kaiken oleellisen heti, eikä käyttäjän tarvitse etsiä näytöltä tietoja.

Toinen iso testauksen kohde työssä oli erilaisten Linux-komentojen testaus. Linux-komentojen testauksessa selvitettiin, mitä komentoja voitaisiin käyttää päivitysso-velluksessa.

Itse koodin ja päivityssovelluksen testausta tehtiin koko työn ajan huomattavan paljon, koska pienten asioiden testaus on helpompaa. Pieniä kokonaisuuksia on helpompi korjata toimimaan oikein testauksen jälkeen kuin isompien kokonaisuuksien.

Lopputestaus tehtiin hyvin laajalti. Samalla yritettiin saada aikaan paljon erilaisia vika- ja häiriötilanteita, jotta niitä varten olisi mahdollista tehdä suojauksia. Mitään isompia muutoksia lopputestauksen jälkeen päivityssovellukseen ei enää tehty tässä vaiheessa, koska päivityssovellus täytti kaikki vaatimukset. Tietyt vikatilanteet, joissa päivityssovelluksen toimintaan tuli häiriöitä, olivat niin radikaaleja ja tekemällä tehtyjä, että niitä ei otettu enää huomioon lopullisessa päivityssovelluksessa.

7 YHTEENVETO JA TULOKSET

Yhteenvetona työ onnistui suunnitellulla tavalla ja vastasi vaatimuksia, joita alussa työlle asetettiin. Exertus Oy oli tyytyväinen työn tulokseen ja he tulevat todennäköisesti käyttämään työtä pohjana asiakaskohtaisissa päivityssovellusratkaisuissa.

Työn tuloksena syntyi päivityssovellus Exertus Oy:n käyttöön, mikä mahdollistaa Exertus Oy:n Linux-moduuleille graafisen sovelluspäivityksen. Päivityssovellus pystyy päivittämään moduuliin sovelluksen, firmwaren, kielitiedoston ja parametrit muutamalla napin painalluksella ja vielä kohtalaisen nopeasti.

LÄHTEET

Exertus Oy. 2018. Yritysinfo. [Verkkosivu]. [Viitattu 11.12.2018]. Saatavissa: http://www.exertus.fi/?lang=fi&id=127&page=Yritys

Kothari, D.P., Shriram, K.V. & Sundaram, R.M.D. 2011. Linux. [Verkkokirja]. New Age International. [Viitattu 21.4.2019]. Saatavana ProQuest Ebook Central -palvelusta. Vaatii käyttöoikeuden.

Fox, R. 2014. Linux with Operating System Concepts. [Verkkokirja]. Chapman and Hall/CRC. [Viitattu 6.5.2019]. Saatavana O'Reilly Media -palvelusta. Vaatii käyttöoikeuden.

Blum, R. & Bresnahan, C. 2014. Linux Command Line and Shell Scripting Bible. [Verkkokirja]. John Wiley & Sons, Incorporated. [Viitattu 21.4.2019]. Saatavana ProQuest Ebook Central -palvelusta. Vaatii käyttöoikeuden.

Rithcie, D. & Kernighan, B. 1988. The C Programming Language, Second Edition. [Verkkokirja]. Prentice Hall. [Viitattu 21.4.2019]. Saatavana O'Reilly Media -palvelusta. Vaatii käyttöoikeuden.

Kochan, S.G. 2004. Programming in C, Third Edition. [Verkkokirja]. Sams. [Viitattu 21.4.2019]. Saatavana O'Reilly Media -palvelusta. Vaatii käyttöoikeuden.

Parker, S. 2011. Shell Scripting: Expert Recipes for Linux, Bash and More. [Verkkokirja]. John Wiley & Sons, Incorporated. [Viitattu 21.4.2019]. Saatavana ProQuest Ebook Central -palvelusta. Vaatii käyttöoikeuden.

Helmke, M. & Sobell, M.G. 2018. Practical Guide to Linux Commands, Editors, and Shell Programming, A, 4th Edition. [Verkkokirja]. Prentice Hall. [Viitattu 21.4.2019]. Saatavana O'Reilly Media -palvelusta. Vaatii käyttöoikeuden.

Makan, K. 2014. Penetration Testing with the Bash shell. [Verkkokirja]. Packt Publishing. [Viitattu 21.4.2019]. Saatavana O'Reilly Media -palvelusta. Vaatii käyttöoikeuden.