Tampereen yliopisto

Jami Ylitalo

KESÄMÖKIN LÄMMITYKSEN ETÄOHJAUKSEN JA KÄYTTÖLIITTYMÄSOVELLUKSEN TOTEUTUS

Kandidaatintutkielma
Tekniikan- ja luonnontieteiden tiedekunta
Tarkastaja: Mikko Salmenperä
Huhtikuu 2023

TIIVISTELMÄ

Jami Ylitalo: Kesämökin lämmityksen etäohjauksen ja käyttöliittymäsovelluksen toteutus
Kandidaatintutkielma
Tampereen yliopisto
Teknisten tieteiden kandidaatin tutkinto-ohjelma
Huhtikuu 2023

Etäohjauksen ja –valvonnan tarve ja täten myös kysyntä ovat nousussa. Jatkuvasti kehittyvät WLAN- ja GSM-teknologiat mahdollistavat kehittyneempien etäohjaus-projektien toteutuksen niin teollisuudessa, kuin kuluttajankin tasollakin. Etäohjattavan laitteiston tarjonnan kasvu ja teknologioiden kehittyminen on luonut halua toteuttaa omia etäohjaustoteutuksia. Kevyet ja edulliset projektit tuovat omat projektivaatimuksensa. Työn tavoitteena on asettaa etäohjaus-projektin laitteistolle ja käyttöliittymälle vaatimukset ja toteuttaa toimiva laitteistokokonaisuus kesämökille.

Työ koostuu kahdesta osasta. Teoriaosuudessa tutustutaan tarkemmin käyttöliittymäsuunnitteluun ja XML-tiedostojen käyttöön. Käydään läpi käyttöliittymäsuunnittelua automaatiossa ja XML-tiedostojen käyttöä automaation käyttöliittymäsuunnittelussa ja yleisesti. Samalla esitellään tarkemmin hyvät käyttöliittymäsuunnittelun periaatteet, joita sovelletaan projektin käyttöliittymän toteutuksessa. Teoriaosuudessa käydään myös läpi ohjelmisto, jolla projektin käyttöliittymä toteutetaan. Esitellään Android Studion käyttöliittymäsuunniteluun ja testaukseen tarkoitetut työkalut sekä XML-tiedostojen soveltaminen käytännössä.

Työn toinen osuus on itse projektin toteutus, jossa esitellään kohde, laitteisto ja toteutettu käyttöliittymä. Kohde-esittelyn yhteydessä mietitään sen tuomia lisäyksiä projektivaatimuksiin. Valittu laitteisto on SimPal T40-V2 ja kolme T20-V2 GSM-pistorasiaa. Laitteiston tekstiviesti-rajapintaa hyödynnetään käyttöliittymän toteutuksessa. Käyttöliittymän suunnittelussa otetaan huomioon sille asetetut vaatimukset: responsiivisuus, johdettavuus ja kohdennus. Android-sovelluksena toteutettu käyttöliittymä on minimalistinen ja helppokäyttöinen.

Projektissa suunniteltu etäohjaustoteutus täyttää kaikki sille asetetut vaatimukset. Toteutus osoittaa, kuinka edullisilla laitehankinnoilla ja pienellä tietotaidolla saadaan luotua paljon käyttöarvoa vanhemmallekin vapaa-ajan asunnolle. Omilla laitemodifikaatioilla, ohjelmisto- ja käyttöliittymätoteutuksilla saadaan toteutettua etäohjaustarve edullisesti ja mukautuvasti. Mukautetut toteutukset mahdollistavat laitteiston muokattavuuden myös tulevaisuuden vaatimuksiin ja ideoiden toteutukseen.

Avainsanat: käyttöliittymä, suunnittelu, käyttöliittymäsuunnittelu, Android Studio, GSM, etäohjaus, Java, instrumentointi, XML

ALKUSANAT

Kandidaatintutkielman ja projektin toteutus oli mielenkiintoinen ja sovelluskehitys eteni sulavasti. Työn henkilökohtaisuuden ansiosta projektin päämäärä oli helposti konkretisoitavissa. Käyttöliittymiin tutustuminen erityisesti automaatiossa avasi silmäni mielenkiintoisiin tulevaisuuden työtehtäviin. Automaation tietotekninen puoli ja varsinkin UI/UX suunnittelu herättivät uutta motivaatiota myös opintoihini.

Haluaisin kiittää ohjaajaani Mikko Salmenperää erittäin hyvistä kehitysehdotuksista ja sulavasta ohjauksesta. Iso kiitos myös perheelleni, joka oli apuna projektin toteutuksessa ja pääsee nyt nauttimaan sen kantamista hedelmistä.

Tampereella, 20.4.2023

Jami Ylitalo

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

WLAN	Wireless Local Area Network
UI	User Interface
UX	User eXperience
GUI	Graphical User Interface
CLI	Command Line Interface
NLI	Natural Language Interface
HMI	Human Machine Interface
SCADA	Supervisory Control and Data Acquisition
XML	eXtensible Markup Language
HTML	HyperText Markup Language
CSV	Comma-Separated Values
IDE	Integrated Development Environment
API	Application Program Interfrace
GSM	The Global System for Mobile Communications

1. JOHDANTO

Suomessa on noin puoli miljoonaa kesämökkiä ja noin puolet suomalaisista käyttää mökkiä tai vapaa-ajan asuntoa vuosittain [1], [2]. Talvisin kesämökin käyttöaste on kesää pienempi, mutta yhä useampi vapaa-ajan asunto on ympärivuotisessa käytössä sähköistymisen vuoksi [3]. Mökkien sähköistyminen mahdollistaa myös erilaiset mukavuudet valaistuksesta lämmitykseen.

Etäohjauksen ja -valvonnan tarve sekä kysyntä on jatkuvassa nousussa erityisesti teollisuudessa, mutta myös kuluttaja tasolla. Jatkuvasti kehittyvän WLAN-teknologian (Wireless Local Area Network) lisäksi uusien mobiiliverkkoteknologioiden myötä projektien tarkat etäohjausvaatimukset voidaan toteuttaa vaivattomasti ja luotettavasti [4]. Teknologioiden ja etälaitteiden yleistyminen kuluttajamarkkinoilla on luonut halua toteuttaa omia etäohjaus-projekteja.

Edulliset ja helpot asennusratkaisut ovat tärkeitä vaatimuksia kevyen asteen etäohjaus-projekteissa. Laitteistotyyppi ja hinta riippuvat paljon asennuskohteesta ja projektin budjetista. Toisin kuin uudistaloissa, joissa etäohjauslaitteiston valinta on suhteellisen vapaata, vanhemmissa kohteissa voi vastaan tulla useita rajoitteita. Esimerkiksi taloudelliset tai rakenteelliset esteet supistavat laitteiston valintamahdollisuuksia.

Suppean laitevalikoiman takia, ei laitteiston käytettävyys ole välttämättä halutulla tasolla. Käyttökokemuksesta saadaan huomattavasti mieluisampi pienelläkin tietotaidolla. Omia laitemodifikaatioita, ohjauskoodilla tai vaikka käyttöliittymällä saadaan edullisestakin laitteesta paljon arvoa projektin lopputulokseen.

Kevyen etäohjaustoteutuksen vaatimuksia ovat laitteiston ja asennustöiden edullinen hinta, itse asennuksen vaivattomuus ja laitteen helppo käyttö. Varsinkin vanhoissa kohteissa on myös otettava huomioon kohteen esteettinen näkökulma. Laitteiston ei tulisi pilata kohteen rakenteita tai luontonäkymää ja muutenkin olla suhteellisen huomaamatonta. Näiden lisäksi laitteiston ja etäohjauksen tulisi tietysti olla luotettavaa ja mielekästä. Toimivan laitteisto- ja sovellusratkaisun tulisi loppujen lopuksi lisätä kohteelle käyttöarvoa.

Tässä työssä suunnitellaan ja toteutetaan kesämökin etäohjaus laitteiston hankinta sekä laitteistolle käyttöliittymäsovellus projektivaatimusten pohjalta. Käyttöliittymän vaatimusten määrittely perustuu hyviin suunnitteluperiaatteisiin, joihin tutustutaan tarkemmin luvussa 2. Samalla tutustutaan käyttöliittymiin ja niiden käyttöön automaatiossa sekä XML-tiedostoihin. Luvussa 3 esitellään käyttöliittymän suunnitteluympäristöön Android Studioon. Luvussa 4 esitellään kohde, valittu laitteisto ja toteutettu käyttöliittymä. Työn yhteenveto ja jatkokehitysmahdollisuudet ovat luvussa 5.

2. KÄYTTÖLIITTYMÄSUUNNITELU JA XML-TIEDOSTOT

Käyttöliittymä (UI, User Interface) tarkoittaa käyttäjän ja tietokoneen, laitteen tai sovelluksen välisen vuorovaikutuksen pistettä. Se sisältää kaikki elementit, jotka mahdollistavat käyttäjien kommunikoinnin ja järjestelmän hallinnan, kuten valikot, painikkeet, kuvakkeet, tekstikentät ja graafiset näytöt. [4]

On monenlaisia käyttöliittymiä, mukaan lukien graafisia (GUI), komentorivi- (CLI) ja luonnollisen kielen käyttöliittymiä (NLI). Ohjelmiston tai sovelluksen käyttöliittymän tyyppi riippuu järjestelmän tarkoituksesta ja käyttäjien tarpeista ja mieltymyksistä. [4] Esimerkiksi sosiaalisen median käyttöliittymät ovat värikkäitä, helposti käytettäviä ja brändin esille tuovia graafisia käyttöliittymiä. Kun taas esimerkiksi versionhallintatyökalu Gitin Git Bash -käyttöliittymä on yksinkertainen komentorivi-ikkuna, jossa käyttäjän on itse tiedettävä komennot.

Graafinen käyttöliittymä on edellä mainituista suosituin ja nimensä mukaisesti näkyvin. Sen avulla käyttäjän ei tarvitse muistaa monimutkaisia komentoja tai toimintoja, vaan käyttö on helppoa ja intuitiivista.

2.1 Suunnitteluperiaatteet

Käyttöliittymää toteutettaessa on hyvä ottaa huomioon yleisesti hyväksi todettuja periaatteita. Suunnitteluperiaatteiden avulla käyttäjäkokemuksesta (UX) saadaan mahdollisimman mieluisa. [5]

Käyttöliittymän tulisi olla selkeä ja helposti ymmärrettävissä. Yksinkertainen käyttöliittymä vähentää käyttäjän virheitä ja auttaa käyttäjää nopeasti löytämään etsimänsä toiminnot. Selkeä käyttöliittymä on myös esteettisesti miellyttävämpi, mikä parantaa käyttäjäkokemusta huomattavasti. [6]

Kohdennus on hyvä tapa toteuttaa selkeämpi käyttöliittymä. Kohdennuksella tarkoitetaan turhien toimintojen karsimista tai piilottamista, jotta käyttäjän huomio kohdistuu oleellisiin toimintoihin ja elementteihin. [7], [8] Hyvin kohdennettu käyttöliittymä on myös johdettava. Johdettava käyttöliittymä on helppo oppia, eikä käyttäjä tarvitse erillisiä ohjeita operoidakseen sitä. Tarvittaessa kuitenkin ohjeet tai opastukset ovat helposti löydettävissä. [6]

Käyttöliittymän tulisi olla johdonmukaista itsensä kanssa eli konsistenssia. UI-elementtien tulisi olla samalla tavalla aseteltuja ja johdonmukaisia. Myös sovelluksen elementtien ja tekstin kokojen, fonttien, värien ja kuvien tulisi olla yhteneväisiä. [7]

Käytettävyyttä lisää huomattavasti responsiivisuus eli käyttöliittymän reagoiminen käyttäjän toimintoihin. Responsiivinen käyttöliittymä on myös nopea ja sulava, jotta käyttäjän on helppo liikkua sovelluksen osien, esimerkiksi valikoiden ja tekstiruutujen, välillä sujuvasti. [6]

Yhä useampi sovelluksen tai internetsivun käyttöliittymä on toteutettu todella minimalistisesti, minkä vuoksi suurin osa edelläkin mainituista suunnitteluperiaatteista on helppo toteuttaa. Turhien elementtien karsiminen, rajoitettu väriskaala, selkeät fontit ja pelkistetyt muodot saavat käyttöliittymän näyttämään ja tuntumaan kevyeltä ja helpolta käyttää. [9], [10]

2.2 Käyttöliittymät automaatiossa

Automaatiojärjestelmää suunniteltaessa on käyttöliittymä erityisen tärkeässä roolissa prosessin valvonnan, käytettävyyden sekä automaatiotuotteen tai tuoteryhmän laadun kannalta. Käyttöliittymällä on myös tärkeä rooli automaation turvallisuudessa hälytysten ja muiden indikaattoreiden muodossa. Hyvin suunniteltu käyttöliittymä viestii asiakkaalle tai käyttäjälle laadusta sekä prosessin ymmärtämisestä, joka on erityisen tärkeää varsinkin yritysasiakkaalle tehdyissä projekteissa. Moderni käyttöliittymäsuunnittelu on nouseva kilpailuetu automaatiossa.

Automaatiossa - ja teollisuudessa muutenkin - käyttöliittymää kutsutaan yleensä kuvaavasti HMI:ksi (Human Machine Interface) [13]. Ne ovat tärkeä osa SCADA (Supervisory Control and Data Acquisition) systeemiä, joka mahdollistaa prosessin valvonnan [15]. Usein järjestelmän HMI on graafinen käyttöliittymä, mutta ei ole siihen rajoittunut.

Ihmisen ja tietokoneen tai prosessin välinen piste on kehittynyt varsinkin tehdasautomaatiossa yksinkertaisista painikkeista moderneihin käyttöliittymäelementteihin. Kehityksen myötä myös automaatiosuunnittelutyökalut ovat kehittyneet kattamaan koko automaatiosuunnittelun viitekehyksen. Sovellukset ja ympäristöt kuten Valmet DNA, Siemens TIA Portal sekä Omronin CX-Supervisor tarjoavat käyttöliittymäsuunnitteluun kattavat ja suhteellisen vapaat työkalut. [12], [13]

2.3 XML-tiedostot käyttöliittymäsuunnittelussa

Käyttöliittymän näkyvät (frontend) elementit, näiden elementtien asettelut, sovelluksen värit ja fontit sekä teemat on mielekästä tallentaa helppoon ja luettavaan muotoon. Tällöin elementtien mukautettu luominen, muokkaaminen ja päivittäminen ovat suunnittelijalle helppoa ja sovelluksen hierarkia on helposti hahmoteltavissa.

XML:ää (eXstensible Markup Language) käytetään usein UI-suunnittelussa elementtien tallennuksessa sen helppolukuisuuden, mutta erityisesti sen muokattavuuden ansiosta. Toisin kuin HTML-tiedostoissa (Hypertext Markup Language), XML ei käytä ennalta määriteltyjä tunnisteita, vaan käyttäjä voi itse määritellä tunnisteensa. [18] XML mahdollistaa siis mukautettujen UI-elementtien tallentamisen mukautetulla tavalla paitsi ihmiselle, myös tietokoneelle luettavassa muodossa.

Myös automaation käyttöliittymäsuunnittelussa käytetään usein XML-tiedostoja. Esimerkiksi Valmet DNA Engineering-suunnitteluympäristö tukee XML-tiedostoja automaatiomoduulien tallennusmuotona. Tämän ansiosta moduulit ovat helposti tuotavissa muihin suunnittelutyökaluihin ja ohjelmointiympäristöihin.

Omronin CX-supervisor puolestaan käyttää CSV-tiedostoja (Comma Separated Values) esimerkiksi käyttöliittymäelementtien generoimiseen [14]. Tämä mahdollistaa elementtien attribuuttien, kuten laiteosoitteiden ja mittayksiköiden määrittelyn Excelissä. Niin sanottu Excel-generointi mahdollistaisi myös XML-tiedostojen käytön.

3. ANDROID STUDIO KEHITYSYMPÄRISTÖNÄ

Android Studio on Googlen omistama ja vuonna 2013 julkaisema ohjelmointiympäristö. Se on integroitu kehitysympäristö (IDE, Integrated Development Environment), joka perustuu tšekkiläisen JetBrains-ohjelmistoyrityksen IntelliJ IDEA Java- ja Kotlin-kehitysympäristöön. Ennen Android-studiota kehittäjien oli toteutettava koodi, kääntäminen ja testaus eri alustoilla. Androidin kasvaessa Google päätti kehittää modernin ja valmiin kehitysalustan Android-sovelluskehitykseen. [17] Koska Android Studio on Android-sovellusten virallinen ohjelmointiympäristö, sen työkalut ja ominaisuudet mahdollistavat ohjelmistokehityksen melkein mille tahansa, eli noin 24 000:lle eri Android-laitteelle [20].

3.1 Layout Editor

Yksi Android Studion räätälöidyistä suunnittelutyökaluista on Layout Editor, joka mahdollistaa XML-tiedoston muokkaamisen graafisella käyttöliittymällä. Se tarjoaa erilaisia valmiita käyttöliittymämalleja sekä mahdollistaa täysin mukautettujen käyttöliittymien ja käyttöliittymäkomponenttien suunnittelun. Sovelluksen näkymä on jaettu sovelluksen todelliseen kuvaan ja sen piirroskuvaan tai blueprint-kuvaan.

Android Studio kutsuu erilaisia käyttöliittymäinstansseja Activityiksi. Esimerkiksi sisäänkirjautumiselle voi olla oma Activitynsä, joka käyttäjän tunnistettuaan kutsuu uutta Activityä, tällä kertaa itse applikaatiota. Activityjen sisällön määrittelee Layout, joka koostuu taas puolestaan View-nimisistä elementeistä, kuten painikkeesta tai tekstiikkunasta. Activityt mahdollistavat erilaiset käynnistystilat käynnistyksen taustatietojen mukaan [19].

Layoutien yhteensopivuuden useille eri laitteille mahdollistaa ConstraintLayout-niminen käyttöliittymän määritelmä, jossa käyttöliittymän elementtien asettelu on riippuvainen muista elementeistä ja näytön reunoista. Rajoitteiden asettelu nopeuttaa suunnitteluprosessia ja suunnittelijan ei tarvitse itse keskittää käyttöliittymäelementtejä.

3.1.1 Valmiit ja mukautetut UI-elementit

Layout Editor tarjoaa useita valmiita elementtejä sen Palette -nimisessä sivupalkissaan. Palette tarjoaa muutamia hyödyllisiä elementtejä, kuten painikkeita, tekstikenttiä ja erilaisia Googlen omia toimintoja, kuten Maps-ikkunan ja mainostilan. Sovelluskehityksessä välttämättä törmää tilanteeseen, jossa valmiit elementit eivät toteuta haluttua toimintoa. Layout Editor mahdollistaa myös omien UI-elementtien suunnittelun XML-tiedostoja hyödyntäen.

Suunnittelija voi helposti luoda uuden XML-tiedoston projektin Resourcers-kansioon ja käyttää näin luotua elementtiä kaikissa projektin näkymissä. Mukautettujen XML-tiedostojen käyttökohteet eivät rajoitu kuitenkaan käyttöliittymiin. Suunnittelija voi luoda esimerkiksi väreille tai mittayksiköille omia määrittelytiedostoja.

3.1.2 Device Manager

Layout Editorin Device Manager mahdollistaa sovelluksen testaamisen erilaisilla oikeilla ja simuloiduilla laitteilla. Miltei reaaliaikainen sovellustestaus mahdollistaa sovelluksen nopean kehityksen ja virhekorjauksen (debugging). Sovelluksen testaaminen eri kokoisilla näytöillä on erittäin tärkeää käytettävyyden ja ylipäätään sovelluksen toiminnan kannalta.

Device Manager mahdollistaa joko fyysisten laitteiden käytön tai laitteen emuloinnin virtuaalisesti. Fyysisiä laitteita voidaan käyttää helposti USB-johdon tai WiFi-yhteyden avulla. Emuloimalla voidaan sovellusta testata melkein millä vain Android-pohjaisella laitteella. Device Managerista löytyy valmiina useita puhelimia, tabletteja, televisioita ja älykelloja, mutta kehittäjä voi pienellä vaivalla tuoda ympäristöön oman laiteprofiilin.

Device Manager antaa kehittäjän valita virtuaalisen laitteen Android version yleensä Lollipop versiosta (Android versio 5.1) alkaen. Tämä mahdollista sovelluskehityksen ja testauksen myös vanhemmille Android-laitteille. API (Application Program Interface) level kertoo ohjelmoijalle kyseiselle versiolle tarjolla olevista luokista ja XML-elementeistä. 2014 julkaistun Lollipop version API level on 22, kun taas uudemman Android 11 on 30. [17], [20]

3.2 Ohjelmointikielet

Android Studio mahdollistaa sovelluskehityksen melkein millä tahansa ohjelmointikielellä. On myös mahdollista käyttää montaa eri ohjelmointikieltä yhdessä

projektissa. Java oli Android Studion virallinen ohjelmointikieli ennen kuin Kotlin otettiin käyttöön sen rinnalle 2017 [22].

Java on korkean tason olioperusteinen ohjelmointikieli, joka on suunniteltu alustariippumattomaksi. Alustariippumattomuuden mahdollistaa JVM (Java Virtual Machine), joka on vastuussa Java-koodin käännetyn tavukoodin ajamisesta. Android-laitteilla koodin kääntämisen tavukoodiksi hoitaa DVM (Dalvik Virtual Machine), jolloin käyttäjän ei tarvitse huolehtia Java-versioista käyttääkseen Android Studiolla tehtyjä sovelluksia. [23], [24]

Kotlin on moderni Javan kanssa yhteensopiva ohjelmointikieli, joka käyttää Javan tavoin JVM-ympäristöä käännetyn koodin suorittamiseksi. Kotlinin erikoisuuksia ovat sen turvallisuus ja yksinkertainen syntaksi. Se on kerännyt suosiota sen tullessa toiseksi viralliseksi ohjelmointikieleksi Android-sovelluskehitykseen. [22], [25] Osa kehittäjistä on siirtynyt Javasta Kotliniin sen turvallisuuden takia. Javan ja Kotlinin yhteensopivuuden ja Android Studion monikielisen sovelluskehityksen ansioista kehittäjien ei tarvitse vaihtaa koko sovelluksensa ohjelmointikieltä, vaan hyödyntää Kotlinia siellä missä sitä tarvitaan.

4. PROJEKTIN TOTEUTUS

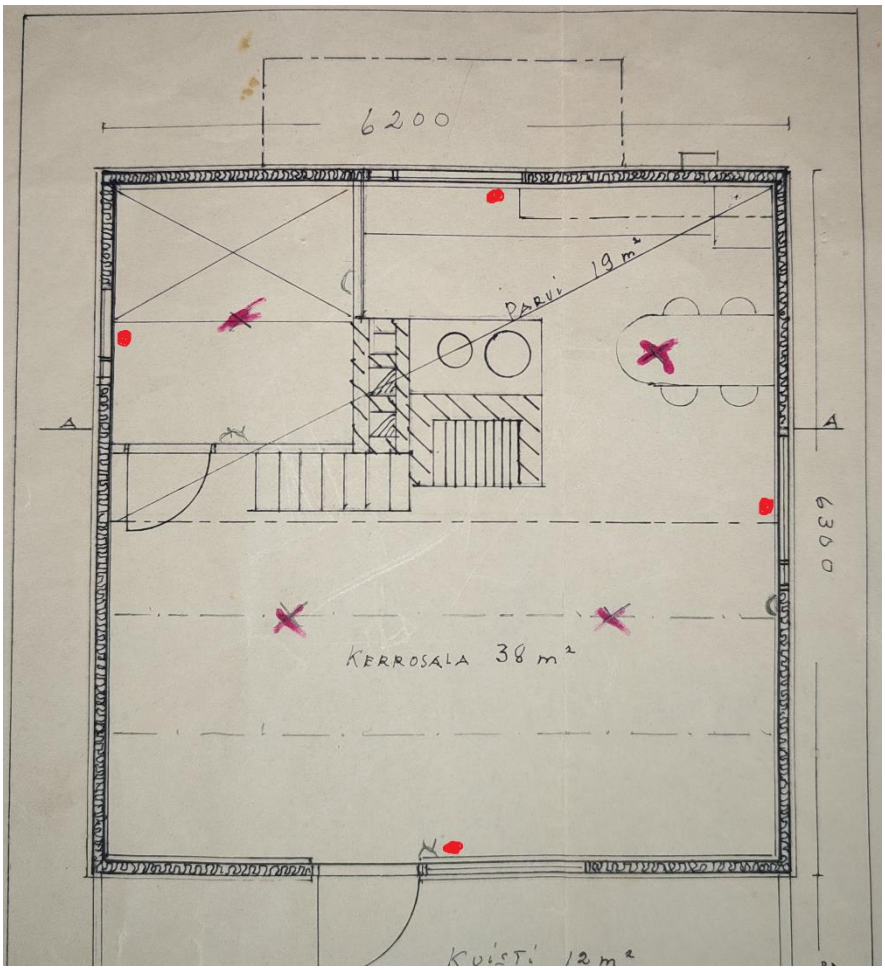
Projektin lähtökohtaisena ideana on täyttää kesämökin lämmityksen etäohjaustarve. Etäohjaus tuo paljon käyttöarvoa mökille varsinkin talvisin, jolloin sisälämpötilat ovat pakkasella ja polttopuun kulutus on suurta. Etäohjaustoteutuksen tulisi olla kevyt ja edullinen, mutta ennen kaikkea helppokäyttöinen. Laitteiston valinnassa otetaan huomioon projektin asetetut vaatimukset hinnan ja laitteiston helppokäyttöisyyden suhteen. Laitteiston tulisi siis olla etäohjattavaa ilman WLAN-yhteyttä, edullista, helposti asennettavissa sekä huomaamatonta.

Laitteistolle suunnitellaan käyttöliittymä parantamaan käyttökokemusta. Käyttöliittymä suunnitellaan ja toteutetaan Android Studiolla, jonka kattavat Android-sovelluskehitys työkalut mahdollistavat nopean ja laiteriippumattoman suunnittelun. Käyttöliittymä toteutetaan lähinnä opettelumielessä, mutta myös tuomaan käyttöarvoa etäohjaus kokonaisuudelle. Toteutuksen tulee täyttää sille asetetut vaatimukset hyvän käyttöliittymäsuunnittelun periaatteista ja sen tulisi olla myös esteettisesti miellyttävä.

4.1 Kohteen esittely

Kesämökki sijaitsee Keski-Suomessa Joutsan kunnassa Jääsjärven rannalla. Kohteessa ei ole internetyhteyttä, mutta GSM-yhteydet ovat suhteellisen luotettava, mutta hitaat. Mökin etäisyys ja hitaat yhteydet vaikuttavat käyttöliittymäsuunnitteluun ja sovelluksen responsiivisuuden saavuttamiseen.

Mökkiin on vedetty sähköt rakennuksen yhteydessä 1980-luvulla ja lämmitys hoidetaan sähköpattereilla, joita mökissä on yhteensä neljä. Sähkölämmityksen lisäksi mökissä on erittäin iso varaava takka. Varsinkin talvisin, jolloin mökin sisälämpötila saattaa pahimmillaan olla jopa alle –10°C, on puun kulutus merkittävä. Mökki on rajallisesti omavarainen puun osalta, joten puuta säästävät toimenpiteet ovat tervetulleita.



Kuva 1: Mökin pohjapiirrustus

Kuvassa 1 esitettynä suunnitellut pistorasian paikat. Lämpöpatterit ovat sijoiteltu jokaiselle mökin seinälle ikkunan alle. Etuoven viereen sijoitetaan T40 päälaite, jolloin se saa mahdollisimman hyvän signaalin. Mökissä on myös parvi, jonne laitteita ei sijoiteta.

4.2 Laitteisto

Laitteisto valittiin mökin rajoitteiden sekä hinnan perusteella. Vanhan sähkökaapin uudistus esimerkiksi älyreleillä ei olisi ollut taloudellisesti kannattavaa, joten päädyttiin halvempaan ratkaisuun: GSM-yhteydellä toimiva älypistorasia.

Etäohjattavia älypistorasioita, jotka sopivat projektin rajoitteisiin löytyi periaatteessa vain yksi: SimPal-T40-V2 GSM-pistorasia (Kuva 2). T40 sisältää lämpötila-anturin, jonka mittausalue on -10...+50°C. Mittausalueensa ansiosta se soveltuu täydellisesti sisälämpötilojen seurantaan. Kovilla pakkasilla sisälämpötila saattaa olla alle anturin mittausalueen, jonka varalta voidaan käyttöliittymäsovelluksessa toteuttaa eräänlainen

lämpötilavahti. T40 vaatii toimiakseen SIM-kortin ja liittymän, joka tuo kuukausikuluja, jotka eivät ole huomattavat.

Kuva 2: T40-V2 GSM-pistorasia

T40-V2 järjestelmää laajentamaan hankittiin kolme T20-V2 pistorasiaa, jotka kytkeytyvät langattomasti T40 emolaitteeseen. T20 sisältävät myös lämpötila-anturit (mittausalue -10...+50°C). Sijoittamalla T40 ja T20-pistorasiat tasaisesti eri puolille mökkiä saadaan kattava keskiarvo mökin oikeasta lämpötilasta.

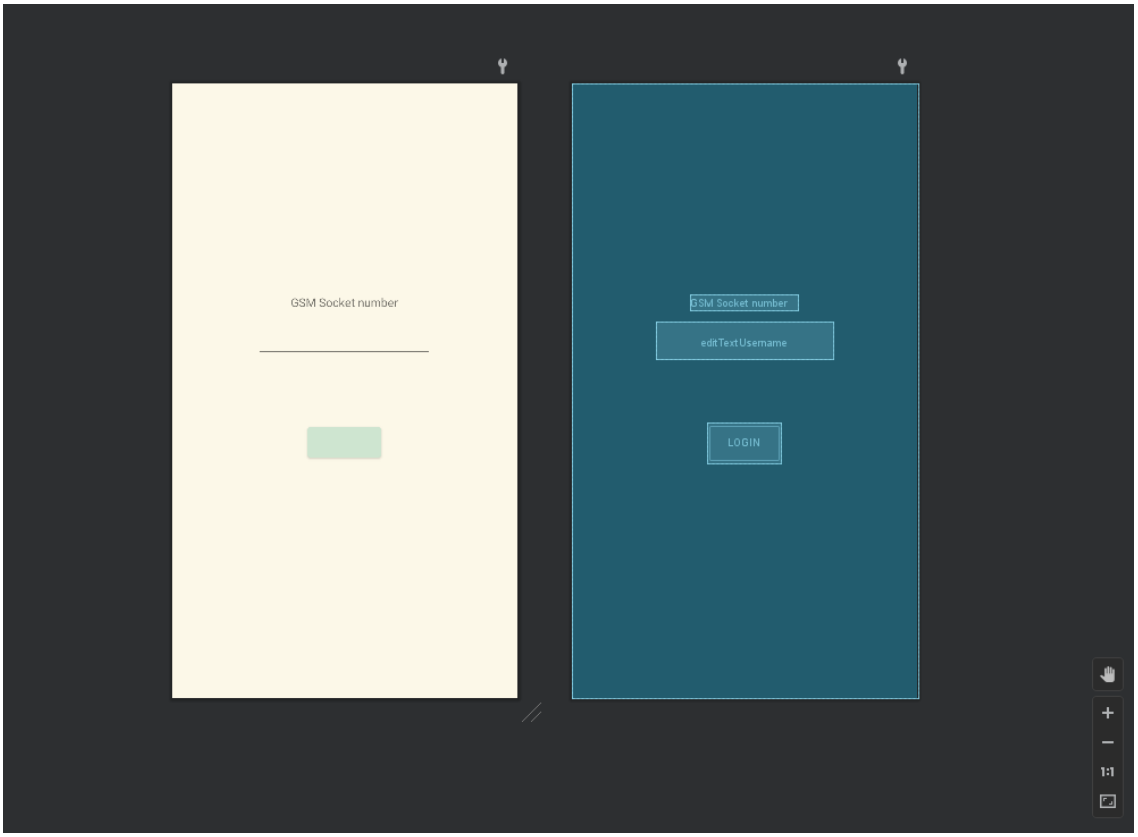
T40-V2 tekstiviestirajapinta toimii lähettämällä tekstiviestillä koodeja laitteelle, joka vastaa tässä tapauksessa noin 10 sekuntin kuluttua varmistus viestillä tai kysytyllä informaatiolla. Lähetettävät koodit ovat pääasiassa muotoa: #Koodi#parametri1#parametri2#…. Kirjautuessa sovellus lähettää päivitysviestin laitteelle, jonka avulla se alustaa sovelluksen tiedot automaattisesti kirjautumisen yhteydessä.

4.3 Käyttöliittymän toteutus

Käyttöliittymän suunnittelussa ja sen toteutuksessa otetaan huomioon hyvät suunnitteluperiaatteet ja esteettiset mieltymykset. Toteutus on erittäin yksinkertainen viiden painikkeen kokonaisuus, jossa itse lämpötilalukema herättää käyttäjän huomion heti. Käyttöliittymässä on myös muutamia responsiivisuutta parantavia toimintoja, jotka helpottavat jo alun perinkin yksinkertaisen sovelluksen käyttöä.

4.3.1 Käyttöliittymänäkymät

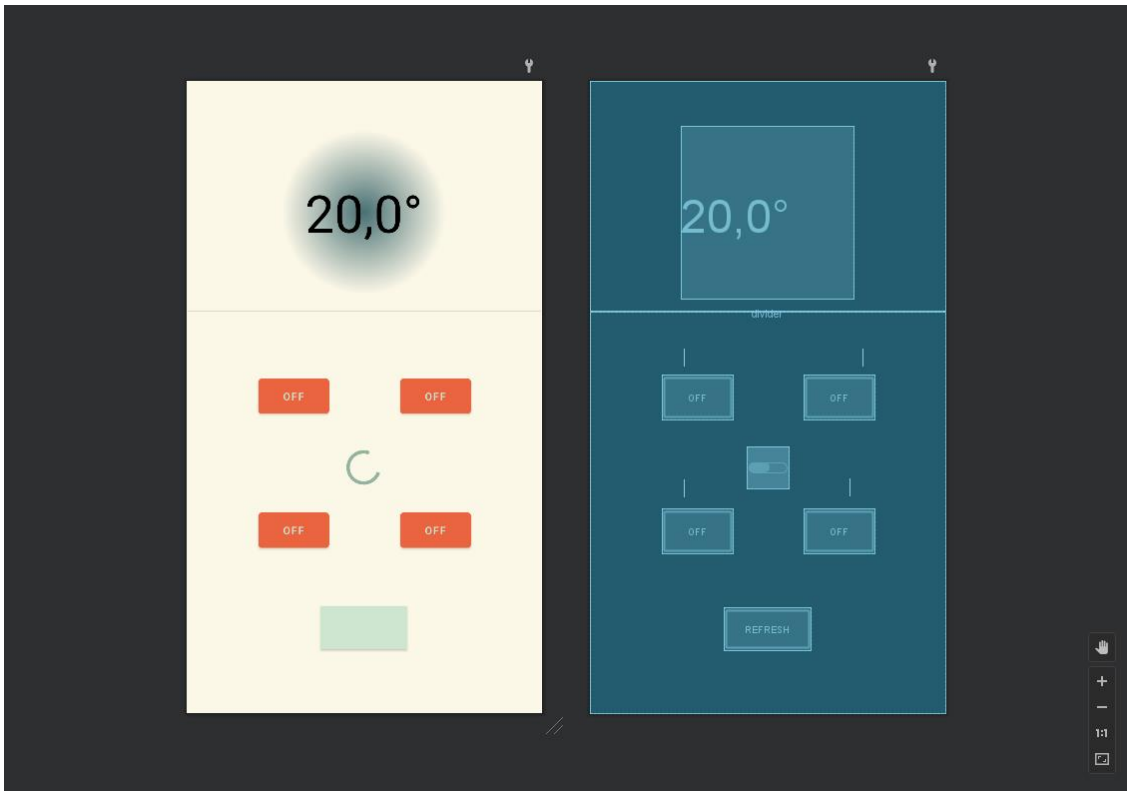
Käyttöliittymän suunnittelu alkoi hahmottelemalla perustoiminnot: sovellukseen kirjautuminen T40-V2 SIM-kortin numerolla ja yksinkertainen lämpötilanäyttö. Android Studion valmiit Layout-sapluunat eivät suoraan tarjonneet sopivaa kirjautumisnäyttöä, joten valittiin Empty Layout eli tyhjä sapluuna. Liitteessä A on kirjautumissivun Java-koodi. Ensimmäisellä käynnistys kerralla sovellus kysyy laitteen SIM-kortin numeroa, joka toimii kirjautumisavaimena. Numero tallennetaan seuraavaa kertaa varten sharedPreference API:n avulla. Kirjautuessa sovellus lähettää päivitysviestin laitteelle, jonka avulla se alustaa sovelluksen tiedot automaattisesti kirjautumisen yhteydessä. Kuvassa X Kirjautumisnäytön graafinen esitys ja Liitteessä B XML-esitys.



Kuva 3: Kirjautumisikkuna ja sen piirroskuva

Päänäkymä koostuu itse lämpötilalukemasta ja jokaista laitetta ohjaavista ON/OFF painikkeista. Ensimmäisen kirjautumiskerran automaattisessa tietojen haussa ja

muissa laitteille lähetettävissä komennoissa on otettu huomioon merkittävä viive pyörivällä latausikonilla. Tämä hillitsee komentojen liian nopean antamisen, vaikkakaan se ei sitä estä. Vaihtoehtoisesti voitaisiin kaikki elementit ottaa pois käytöstä komentojen suorituksen ajaksi.



Kuva 4: päänäkymä ja sen piirroskuva

4.3.2 Muut XML-tiedostot

Itse käyttöliittymän lisäksi sovelluksen värit, teemat ja alustustekstit ovat omissa XML-tiedostoissaan. Projektin avautuessa Android Studio alustaa colors.xml nimisen tiedoston omilla väreillään. Värit ovat tallennettuna avain-arvo-pareiksi, joissa värit ovat heksadesimaaleina. XML-tiedosto helpottaa sovelluksen väriteeman suunnittelussa Kuvassa 5 sovelluksen väripaletti, joka toteutetaan XML-tiedostolla liitteessä C.

Kuva 5: Sovelluksen väripaletti

Värien tapaan Android Studio tarjoaa oletuksena strings.xml tiedoston, joka toimii String-tyyppisten muuttujien tallennuspaikkana. Tämä on erityisen hyödyllistä varsinkin käyttöliittymän alustusvaiheessa niin sanottujen paikanpitäjä muuttujien kanssa. XML mahdollistaisi myös taulukkojen tallennuksen, joka mahdollistaisi esimerkiksi laitteelta saadun lämpötiladatan tallennuksen.

Lämpötilalukeman takana oleva korostus elementti on toteutettu mukautetulla XML-tiedostolla. Elementti käyttää shape nimistä tunnistetta, jolle voidaan antaa erilaisia muotoja ja värejä. Tässä tapauksessa muoto on ovaali ja väri on kahden värin yhdistelmä, joka toteutetaan gradient-tunnisteella.

4.3.3 Koodi ja toiminta

Sovellus kommunikoi laitteistoille tekstiviesti rajapinnan avulla. Viestit ovat määriteltynä strings.xml tiedostossa, jotta niitä ei tarvitse erikseen alustaa koodissa. Painikkeiden painalluksia “kuunnellaan” onClickListener nimisellä Java rajapinnalla. Painettaessa painiketta rajapinnan toteutus tarkistaa laitteen tilan ja lähettää tarvittavan ohjausviestin

laitteelle. T40 ja T20 tarvitsevat erilaiset rajapinnan toteutuksen. T20 ohjauskomennot vaativat nimitunnisteen. Esimerkiksi komento T20 käynnistykseen olisi: #61#nimi#.

Tekstiviestien lähetys toteutetaan SmsManager-luokan avulla. Luokka tarjoaa erilaisia tekstiviestin lähetykseen liittyviä metodeja, joista sendTextMessage yksinkertaisesti lähettää tekstiviestin haluttuun numeroon. Luokka ei kuitenkaan mahdollista viestin vastaanottamista, joka joudutaan toteuttamaan BroadcastReceiver-luokkaa jatkavalla MessageReceiver luokalla.

5. YHTEENVETO

Työn tavoitteena oli tutustua käyttöliittymäsuunnitteluun ja Android sovelluskehitykseen mökin lämmityksen etäohjaus projektin muodossa. Samalla tutustuttiin UI/UX suunnittelun hyviin periaatteisiin ja toimintatapoihin sekä lyhyesti XML-tiedostojen hyödyntämiseen käyttöliittymäsuunnittelussa. Käyttöliittymän hyvän lopputuloksen lisäksi, suunnittelutyö ja selkeät määrittelyt ovat tärkeä osa sovelluskehitystä.

Tutustuttiin myös Android Studioon kehitysympäristönä, erityisesti käyttöliittymäsuunnittelun kannalta hyödyllisiin toimintoihin. Layout Editorin tarjoamat kehitystyökalut, kuten Device Manager, ovat miltei välttämättömiä laiteriippumattoman sovelluskehityksen ja erityisesti testauksen kannalta. Layout Editorin yhteydessä tarkasteltiin XML-tiedostoja käytännön sovelluskehitysympäristössä.

Mökin lämmityksen etäohjaukseen valittu laitteisto oli SimPalin T40-V2 GSM-pistorasia. Pistorasia valittiin sen ominaisuuksien ja mökin asettamien rajoitteiden perusteella. T40-V2 lisäksi hankittiin kolme kappaletta T20-V2 pistorasioita, jotka laajensivat lämpötilan mittausaluetta koko mökin alueelle. Vaikka projektille ei ollut asetettu tarkkaa budjettia, olivat laitehankinnat laitetarjonnan edullisimmasta päästä. Laitteiston kokonaishinnaksi tuli noin 150 €. Laitteisto täytti myös muut vaatimukset. Laitteiston asennus oli erittäin helppoa ja se on lähes huomaamaton.

Sovelluksen Java-koodia on suhteellisen vähän: vain noin 500 riviä. Tämän ansiosta sovelluksen lataaminen, käynnistys ja operointi on todella sulavaa ja nopeaa. Androidin tarjoamat tekstiviesti rajapinta oli kuitenkin suhteellisen vanhanaikainen. Ongelmia syntyi varsinkin lupakyselyiden kanssa. Lupakyselyt on jätetty kokonaan sovelluskehittäjän vastuulle ja oikeaa toimintamallia oli vaikea löytää. Tämä käytännössä esti sovelluksen jakamisen Google Play Storessa, sen tarkkojen tietoturvavaatimusten takia.

Itse käyttöliittymä toteutettiin Android Studiossa hyödyntäen hyviä suunnitteluperiaatteita. Toteutuksessa nähtiin Android Studion XML-tiedostojen hyödyntäminen käytännössä ja Layout Editorissa toteutetut kaksi eri Activityä. Näkymät toimivat käytännössä hyvin ja sovellus ei kysynyt kirjautumistietoja enää toisella kirjautumiskerralla. Sovelluskehitys toteutettiin melko nopealla aikataululla, joten kaikkia haluttuja ominaisuuksia ei ehditty toteuttaa.

Lopputoteutus täytti miltei kaikki projektille asetetut vaatimukset. Toteutuksessa löytyi kuitenkin pieniä ärsytyksen aiheita käytettäessä. Viestin lähetys, vastaanotto ja datan prosessointi -sykli osoittautui todella hitaaksi. Tämä johtui luultavasti tekstiviestien

hitaasta saapumisesta mökin etäisyyden takia. Ohjelmointivaiheessa myös datan muuntaminen luettavaan muotoon osoittautui todella hankalaksi sen esitysmuodon ja tekstiviestien vastaanottamisesta vastuussa olevan rajapinnan takia. Näistä huolimatta sovelluksesta saatiin mukava käyttää ja laitteisto täytti sille asetetut vaatimukset.

Työn jälkikehitysmahdollissuudet ovat laajat. Esimerkiksi lämmityksen ohjauksen sitominen pörssisähkön hintaan API:n avulla. Hinta rajapintoja on tarjolla paljon ja ne julkaistaan kerran päivässä, jolloin sovellus voisi käydä poimimassa datan XML tai JSON muodossa, josta se suunnittelisi sähköjen ohjauksen seuraavalle päivälle. Lisäksi erilaiset lämpötilaohjaukset ja pistorasioiden itsenäinen ohjaus lämpötilan perusteella. Erilaisten ohjausten lisäksi, itse sovelluksen saatavuutta ja tietoturvaan liittyviä kehityskohteita on paljon. Sovelluksen jakaminen esimerkiksi Googlen Play Storessa vaatisi lupakyselyitä ja tekstiviesti luokan uudelleen kirjoittamista Googlen tietoturva vaatimusten täyttämiseksi.

Android Studion tarjoamaa laiteriippumatonta suunnittelua voisi tulevaisuudessa hyödyntää esimerkiksi älykellolle tarkoitettuun sovelluksessa. Älykello-sovellus voisi olla yksinkertaistettu malli puhelin sovelluksesta, josta näkisi esimerkiksi vain lämpötilan ja yhdellä painikkeella pystyisi ohjaamaan koko laitteistoa.

Sovelluksen toteutus myös eri ohjelmointikielellä tai ympäristöllä olisi tulevaisuudessa kannattavaa käytettävyyden kannalta. Esimerkiksi HTML, CSS ja JavaScript teknologioilla, joita yleensä käytetään modernissa Frontend-kehityksessä. Sovelluksen muuttaminen Web-sovellukseksi mahdollistaisi sen helpomman jakelun ja laiteriippumattomuuden varsinkin uuden HTML 6 teknologian avulla.

LÄHTEET

[1] Suomen virallinen tilasto (SVT): Rakennukset ja kesämökit [verkkojulkaisu]. ISSN=1798-677X. 2019, Kesämökit 2019 . Helsinki: Tilastokeskus [viitattu: 2.2.2023].
Saatavissa: http://www.stat.fi/til/rakke/2019/rakke_2019_2020-05-27_kat_001_fi.html

[2] Anu Rämö: Mökkeilijöiden määrä lasketaan miljoonissa – vapaa-ajan asumisen suosio kasvussa [verkkojulkaisu]. Helsinki: Tilastokeskus [viitattu 22.2.20223] Saatavissa: https://www.tilastokeskus.fi/tietotrendit/blogit/2020/mokkeilijoiden-maara-lasketaan-miljoonissa-vapaa-ajan-asumisen-suosio-kasvussa/

[3] Elenia: Mökki kannattaa sähköistää [verkkojulkaisu]. [viitattu 22.2.2023]. Saatavissa: https://www.elenia.fi/artikkelit/mokki-kannattaa-sahkoistaa

[4] Tatu Niemi: mobiiliverkkotekniikat teollisuusautomaatiossa. Saatavissa: https://trepo.tuni.fi/handle/10024/141256

[5] ISO 9241-110:2020. [viitattu 28.2.2023]. Saatavissa: https://www.iso.org/standard/75258.html

[6] Into-Digital: Verkkopalvelun käyttöliittymä (UI) [verkkojulkaisu]. [viitattu 28.2.2023]. Saatavissa: https://into-digital.fi/osaamisemme/design/kayttoliittyma/?gclid=Cj0KCQiA6fafBhC1ARIsAIJjL8mR6lIUVhRFB1xlDwI8_m-uvBI5cAVhT61xwxxSGAAmYPsWXJaSd6YaAoRREALw_wcB

[7] William Lidwell, Kritina Holden, Jill Butler: Universal Principles of Design. [viitattu 28.2.2023]. Saatavissa: https://arc345ergofactors.files.wordpress.com/2016/03/william-lidwell-kritina-holden-jill-butler-universal-principles-of-design-rockport-publishers-2003.pdf

[8] Don Norman: The design of everyday things. [viitattu 28.2.2023]. Saatavissa: http://thuvienso.bvu.edu.vn/handle/TVDHBRVT/19318

[9] LM&Someco: Visuaalinen suunnittelu ja kohdentaminen vahvistavat asiakaskokemusta [verkkojulkaisu]. [viitattu 28.2.2023]. Saatavissa: https://lmsomeco.fi/blogi/visuaalinen-suunnittelu-ja-kohdentaminen-vahvistavat-asiakaskokemusta/

[10] S. M. Sani and Y. K. Shokooh, "Minimalism in designing user interface of commercial websites based on Gestalt visual perception laws (Case study of three top brands in technology scope)," *2016 Second International Conference on Web Research (ICWR)*, Tehran, Iran, 2016, pp. 115-124, doi: 10.1109/ICWR.2016.7498455. Saatavissa: https://ieeexplore.ieee.org/abstract/document/7498455

[11] Letizia Bollini (2017) Beautiful interfaces. From user experience to user interface design, The Design Journal, 20:sup1, S89-S101, DOI: 10.1080/14606925.2017.1352649. Saatavissa:

https://www.tandfonline.com/doi/abs/10.1080/14606925.2017.1352649

[12] Liam Bee: PLC and HMI Development with Siemens TIA Portal. Saatavissa: https://learning.oreilly.com/library/view/plc-and-hmi/9781801817226/

[13] Manufacturingtomorrow.com: Human Machine Interface HMI and its importance in industrial automation. Saatavissa: https://www.manufacturingtomorrow.com/article/2022/01/human-machine-interface-hmi-and-its-importance-in-industrial-automation/18191

[14] Omron: CX-Supervisor user manual. Saatavissa: https://assets.omron.eu/downloads/manual/en/v7/w10e_cx-supervisor_users_manual_en.pdf

[15] Computer Security Resource Center: Supervisor Control and Data Acquisition. Saatavissa: https://csrc.nist.gov/glossary/term/supervisory_control_and_data_acquisition

[16] Antti Venho: Laatusäätöjärjestelmän automaatiomoduulien määrällinen karakterisointi. Saatavissa: https://www.theseus.fi/handle/10024/345263

[17] Android developers dokumentaatio. Saatavissa: https://developer.android.com/studio/intro

[18] IBM development: Build dynamic user interfaces with android and XML (8.10.2022) Saatavissa: https://developer.ibm.com/tutorials/x-andddyntut/

[19] Android developers: Introduction to activities. Saatavissa: https://developer.android.com/guide/components/activities/intro-activities

[20] Quartz: There are now more than 24000 different Android devices. Saatavissa: https://qz.com/472767/there-are-now-more-than-24000-different-android-devices

[21] Jackson, W. (2017) Android Apps for Absolute Beginners: Covering Android 7, Fourth Edition. Fourth Edition. Saatavissa: https://learning.oreilly.com/library/view/android-apps-for/9781484222683/?sso_link=yes&sso_link_from=tampere-university

[22] Kotlin Docs: Kotlin dokumentaatio. Saatavissa: https://kotlinlang.org/docs/home.html

[23] IBM: What is Java? Saatavissa: https://www.ibm.com/topics/java

[24] Towardsdatascience.com: JVM vs DVM. Saatavissa: https://towardsdatascience.com/jvm-vs-dvm-b257229d18a2

[25] M. Martinez and B. Gois Mateus, "Why Did Developers Migrate Android Applications From Java to Kotlin?," in *IEEE Transactions on Software Engineering*, vol. 48, no. 11, pp. 4521-4534, 1 Nov. 2022, doi: 10.1109/TSE.2021.3120367. Saatavissa: https://ieeexplore-ieee-org.libproxy.tuni.fi/document/9576593

LIITTEET

LIITE A: KIRJAUTUSMISSIVUN KOODI

```
public class MainActivity extends AppCompatActivity {

    private ArrayList<String> socketNames;
    private EditText socketNumber;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //alustetaan UI elementit ja ns keksit
        Button loginButton = findViewById(R.id.loginButton);
        EditText socketNumber = findViewById(R.id.editTextUsername);
        SharedPreferences sh = getSharedPreferences("UserData", MODE_PRIVATE);

        //tarkastaa onko kirjauduttu jo ja vaihtaa näkymää
        if ( !sh.getString("socketNumber", "").equals("") ){
            startActivity(new Intent(MainActivity.this, MainQui.class));
        }
        SharedPreferences.Editor editor = sh.edit();

        //Sisäänkirjautuessa tallennetaan numero seuraavaa loginnia varten
        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(!socketNumber.getText().toString().equals("") ){
                    editor.putString("socketNumber", socketNumber.getText().toString());
                    editor.apply();
                    startActivity(new Intent(MainActivity.this, MainQui.class));
                }
            }
        });
    }

    public ArrayList<String> getSocketNames(){
        return this.socketNames;
    }

    public String getSocketNumber(){
        return this.socketNumber.getText().toString();
    }
}
```

LIITE B: VIESTIN VASTAANOTTO

```
1  public class MessageReceiver extends BroadcastReceiver {
2
3      private static MainQui.SmsListener mListener;
4      private String message;
5
6      @Override
7      public void onReceive(Context context, Intent intent) {
8          Bundle bundle = intent.getExtras();
9          Object[] pdus = (Object[]) bundle.get("pdus");
10         SmsMessage[] messages = new SmsMessage[pdus.length];
11
12         for(int i = 0; i < messages.length;  i++){
13             messages[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
14         }
15
16         String address = messages[0].getOriginatingAddress();
17         String fullmessage = "";
18         String trueNumber = MainQui.getSocketNumber();
19
20         Log.d("received from" , address);
21         Log.d("true number is", trueNumber);
22
23         if( address.equals(trueNumber) ){
24             for(SmsMessage message : messages){
25                 fullmessage += message.getMessageBody();
26             }
27             fullmessage = fullmessage.trim();
28             message = fullmessage;
29
30             if( ! message.contains("Schedule")){
31                 MainQui.updateInfo(message);
32             }else{
33                 Log.d("confirmation received", message);
34             }
35             MainQui.toggleLoading();
36         }
37     }
38
39     public String getLatestMessage(){
40         return this.message;
41     }
42 }
```

LIITE C: XML-TIEDOSTO ESIMERKKI

```
Colors.xml:

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="my_primary_color">#CEE5D0</color>
    <color name="my_prim_dark_color">#94B49F</color>
    <color name="my_orange">#EB6440</color>
    <color name="my_green">#94B49F</color>
    <color name="circle_fade_1">#497174</color>
    <color type="color" name="app_background">#FCF8E8</color>
</resources>

Strings.xml:

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">SocketQUI</string>
    <string name="socket_1">Socket 1</string>
    <string name="socket_3">Socket 3</string>
    <string name="temperature_c">Temperature (°C)</string>
    <string name="_20">20</string>
    <string name="refresh">Refresh</string>
    <string name="socket_2">Socket 2</string>
    <string name="socket_4">Socket 4</string>
    <string name="off">OFF</string>
    <string name="OFF">OFF</string>
    <string name="OFF2">OFF</string>
    <string name="OFF3">OFF</string>
    <string name="gsm_socket_number">GSM Socket number</string>
    <string name="login">Login</string>
    <string name="temp_placeholder_text">20,0°</string>
    <string name="refresh_button_text">refresh</string>
    <string name="textview_placeholder" />
</resources>
```