LAPIN AMK
Lapland University of Applied Sciences

Langaton etäisyyden mittaus Android Arduino -järjestelmällä

Topi Ylitalo

Opinnäytetyö
Tekniikka ja liikenne
Tietotekniikka
Insinööri (AMK)

2015

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tekniikka ja liikenne
Tietotekniikka

Tekijä Topi Ylitalo
Ohjaaja Kenneth Karlsson
Vuosi 2015
Työn nimi Langaton etäisyyden mittaus Android Arduino -järjestelmällä
Sivu- ja liitemäärä 44 + 0

Tämän opinnäytetyön tavoitteena oli toteuttaa järjestelmä langatonta etäisyydenmittausta varten. Järjestelmä toteutettiin Android-sovelluksen, Arduino-kehitysalustan sekä siihen yhdistettyjen Bluetooth-lähettimen ja etäisyysanturin avulla. Järjestelmä mittaa etäisyyden muutoksia etäisyysanturilla ja lähettää tiedot langattomasti Arduinon avulla. Mittaustiedot otetaan vastaan ja esitetään käyttäjälle Android-sovelluksessa.

Työn toteutuksen aikana perehdyttiin Android- ja Arduino-ohjelmointiin sekä järjestelmän kehityksessä käytettyihin ohjelmistoihin ja Arduinon komponentteihin. Käyttöliittymä suunniteltiin Pencil-ohjelmistolla ja Android-sovelluksen ohjelmoinnissa käytettiin Android Studiota. Sovelluksen testauksessa kehityskoneella käytettiin Oracle VirtualBox -virtualisointialustaa ja Arduino-ohjelmointiin Arduino IDE:a.

Työn lopputuloksena syntyi järjestelmä, jonka avulla voidaan seurata kondenssivesisäiliön täyttymistä mobiililaitteen avulla. Järjestelmä soveltuu myös useisiin muihin tarkoituksiin, joissa seurataan vaihtelevaa etäisyyttä.

Avainsanat Android, Arduino, Bluetooth, etäisyysanturi

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Technology, Communication and Transport
Degree Programme in Information Technology

Author Topi Ylitalo
Supervisor(s) Kenneth Karlsson
Subject of thesis Wireless distance measurement with Android Arduino system
Number of pages 44 + 0
Year 2015

The aim of this study was to develop a system for wireless distance measurement. The aim was to develop the system using Android application, Arduino board and Bluetooth modem together with distance sensor connected to Arduino board.

During the process, programming of Android and Arduino was studied together with applications and Arduino components used in system development. A user interface was designed using Pencil tool, Android application was developed using Android Studio and Arduino software using Arduino IDE. Android application was tested on a development computer in Oracle VirtualBox.

The study resulted in a system that can be used to monitor condensate level in the container with a mobile device. The system measures changing distance by a distance sensor and transmits data wirelessly using Arduino. The data is received and presented for user on Android application. The system is usable in several other cases when monitoring changing distance.

Key words Android, Arduino, Bluetooth, distance sensor

SISÄLLYS

1 JOHDANTO 6
2 SUUNNITTELU 8
3 KEHITYSTYÖKALUT JA TEKNIIKKA 10
3.1 Android 10
3.2 Android Studio 10
3.3 Arduino 11
3.4 Arduino IDE 12
3.5 Virtual Box 13
3.6 Sensorit ja anturit 14
3.7 Bluetooth 15
4 KEHITYSYMPÄRISTÖN ASENNUS 17
4.1 VirtualBox ja Android-käyttöjärjestelmä 17
4.2 Android Studio 18
4.3 Arduino IDE 19
5 SOVELLUKSEN TOTEUTUS 21
5.1 Arduino kytkennät 21
5.2 Järjestelmän toteutus 23
5.2.1 Projektin luonti Android Studiolla 23
5.2.2 Aktiviteetin toteuttaminen 24
5.2.3 Näytön toteutus 26
5.2.4 Näytön tapahtumien käsittely 27
5.2.5 Bluetooth käynnistys 28
5.2.6 Bluetooth laitteiden etsintä 29
5.2.7 Bluetooth-laitteiden paritus 31
5.2.8 Viestien lähettäminen näytölle 33
5.2.9 Viestien käsittely 34
5.2.10 Tiedoston tallentaminen ja lukeminen 35
5.2.11 Arduino-ohjelma 36
6 YHTEENVETO JA KEHITYSIDEOITA 39
7 LÄHTEET 42

KÄYTETYT MERKIT JA LYHENTEET

ADB	Android Debug Bridge, komentorivityökalu virheiden etsintään
AVD	Android Virtual Device, virtuaalinen Android-laite
BIOS	Basic Input-Output System, tietokoneen käyttöjärjestelmän käynnistävä ohjelma
COM	Communication port, sarjaportti
DDMS	Dalvin Debug Monitor Server, työkalu virheiden etsintään Android-sovelluksista
IDE	Integrated Development Environment, ohjelmointiympäristö
JDK	Java Development Kit, valikoima ohjelmia Java-sovelluskehitykseen
RAM	Random Access Memory, ohjelmien työmuisti
RFCOMM	Radio Frequency Communication, sarjaporttipalvelu Bluetooth-yhteyksille
RS-232	Recommended Standard 232, 1960-luvulla esitelty standardi tietokoneiden väliselle dataliikenteelle (Electronic design 2015).
SDK	Software Development Kit, valikoima sovelluskehitystyökaluja
SPP	Serial Port Profile, sarjaporttiprofiili
TCP	Transmission Control Protocol, tietoliikenneprotokolla
USB	Universal Serial Bus, sarjaväylä oheislaitteiden liittämiseen
XML	Extensible Markup Language, tekstimuotoinen rakenteellinen kuvauskieli

1 JOHDANTO

Nykyään on saatavilla lukuisia ohjelmistoja, piirilevyjä ja komponentteja, joiden avulla pystyy toteuttamaan monenlaisia järjestelmiä helpottamaan päivittäisiä tehtäviä. Useat järjestelmien toteuttamisessa tarvittavat ohjelmistot ovat verkossa ilmaiseksi ladattavissa ja komponentteja voi tilata kohtuullisin kustannuksin verkkokaupoista. Opinnäytetyön idea syntyi varastossa sijaitsevan pakastimen rikkoutumisesta ja sen seurauksena menetetystä pakastimen sisällöstä sekä mahdollisuuksista havaita muita vastaavia riskitekijöitä ajoissa. Varsinaiseksi työn kohteeksi valittiin ilmastointilaitteen kondenssivesiastian täyttymisen seuranta langattomasti. Laite sijaitsee lukitussa tilassa ja vesiastian täyttymisen tilanteen voi työn tuloksena tarkistaa mobiililaitteen näytöltä.

Tämän opinnäytetyön tarkoituksena oli perehtyä Android-ohjelmointiin, Arduino-komponentteihin, Bluetooth-tiedonsiirtoon sekä kehitystyökaluihin, joita tarvitaan tämän tyyppisessä ohjelmistokehityksessä. Työn lopputuloksena syntyi järjestelmä, jossa Android-sovelluksessa esitetään Arduinon langattomasti lähettämää mittausdataa käyttäjälle. Mittaus suoritettiin optisella etäisyysanturilla, jonka mittausarvot lähetettiin Arduino-piirilevyyn kytketyn Bluetooth-lähettimen kautta Android-sovellukselle. Sovellus tulkitsee saadun datan käyttäjälle esitettävään muotoon.

Ohjelmistokehityksen eri vaiheissa tarvitaan useita työtä helpottavia ohjelmistoja. Tämän työn toteutuksessa käytettiin muutamia, jotka ovat ilmaiseksi saatavilla ja joista on mahdollisimman paljon hyötyä kunkin vaiheen suorittamisessa. Android-sovelluksen käyttöliittymän suunnittelussa ja kuvaamisessa käytettiin Pencil-ohjelmistoa. Käyttöliittymän ja toimintojen toteutus tehtiin Android Studio-ohjelmistolla, joka on täysin integroitu ohjelmointiympäristö (IDE). Android Studio oli yhdistetty VirtualBox-virtualisointialustaan, jonka avulla ohjelmoinnin tuloksia voitiin kokeilla Android-käyttöjärjestelmässä. Arduino-kehitysalustan ohjelmoinnissa käytettiin ohjelmointityökalua Arduino IDE, jonka avulla toteutettiin datan lähetyksen tarvitsemat koodit ja siirrettiin ne suoritettavaksi kehityskoneelta Arduino-mikrokontrollerille.

Opinnäytetyössä kerrotaan aluksi järjestelmän suunnittelusta sekä taustoja käytetyistä ohjelmistoista. Seuraavaksi käydään läpi pääkohdat kehitysympäristön rakentamisesta, kuten ohjelmistojen asennuksista sekä tarvittavista kytkennöistä Arduino-piirilevyn ja tarvittavien komponenttien välillä. Android-sovelluksen koodeista avataan keskeisimmät kohdat sekä kerrotaan Arduinon ohjelmoinnista. Lopuksi tehdään yhteenveto työn vaiheista sekä pohditaan järjestelmän kehitysmahdollisuuksia.

2 SUUNNITTELU

Suunnittelun kohteena oli järjestelmä, jonka avulla voidaan seurata langattomasti ilmastointilaitteen kondenssivesiastian täyttymistä. Ilmastointilaite sijaitsi lukitussa tilassa ja sen kondenssivesiastian täyttyminen jäi helposti huomaamatta, ellei sitä käynyt säännöllisesti tarkastamassa. Samanlaista järjestelmää voi käyttää useissa muissakin käyttötarkoituksissa, joissa on tarve mitata vaihtelevaa etäisyyttä hankalissa tai jopa terveydelle vaarallisissa olosuhteissa.

Suunnittelun aikana perehdyttiin järjestelmässä tarvittaviin komponentteihin, toimintoihin sekä Android-sovelluksen käyttöliittymään. Sovellukseen suunniteltiin kolme näyttöä, jotka on esitetty kuviossa 1. Näytöt piirrettiin käyttöliittymien suunnitteluun tarkoitetulla Pencil-piirtotyökalulla. Ensimmäisellä näytöllä esitetään anturilta saatavat mittaustulokset ja se on myös sovelluksen päänäyttö. Toisen näytön kautta tehdään raja-arvojen tallentaminen ja kolmannella näytöllä hallitaan Bluetooth-yhteyksiä laitteiden välillä.

Bluetooth-yhteyksien hallinta tehdään sovelluksen sisällä omalla näytöllään, jolloin kaikki toiminnot voidaan suorittaa sovelluksen kautta. Bluetooth-yhteyksiä voi hallita myös Androidin omien valikoiden kautta, mutta työn tarkoituksena oli perehtyä Android-ohjelmointiin, joten oma toteutus tuki paremmin tavoitteita. Sovelluksessa käytetyt raja-arvot tallennetaan hallintanäytön kautta määrätyssä muodossa laitteen sisäisessä muistissa olevaan tiedostoon. Raja-arvot määrittävät rajat, joiden mukaan mittaustuloksen alla näytetään raja-arvoihin liitetty viesti tekstimuodossa. Esimerkiksi mittausväli 2–4 cm tarkoittaa säiliön olevan lähes täysi ja vaatii huomiota, jolloin viesti voi olla vaikkapa ”Vaatii tyhjennystä”.

Android-sovelluksen lisäksi tarvittiin anturi mittaamaan etäisyyttä sekä Bluetooth-lähetin langatonta tiedonsiirtoa varten. Tietojen lähettämiseen valittiin avoimen lähdekoodin periaatteilla toimiva Arduino UNO -kehitysalusta ja sen kanssa yhteensopiva optinen etäisyysanturi keräämään mittaustiedot. Arduino UNO -piirilevyllä ei ole valmiina Bluetooth-lähetintä, joten se jouduttiin hankkimaan erikseen. Muut tarvittavat liitännät ja ominaisuudet piirilevyltä löytyivät.

Mittaus- ja lähetinpään tehtävänä on pelkästään lähettää tarvittava tieto Android-sovellukselle, jossa suoritetaan tiedon purkaminen ja muokkaaminen esitettävään muotoon.

Kuvio 1. Android-sovelluksen näytöt

3 KEHITYSTYÖKALUT JA TEKNIIKKA

3.1 Android

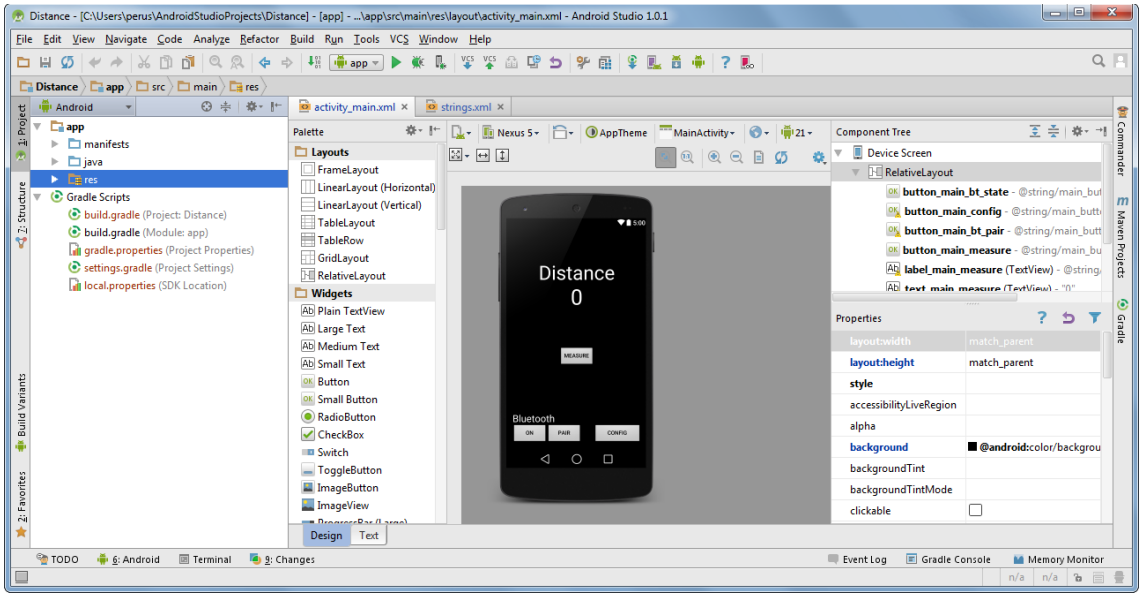
Android on Googlen julkaisema Linux-pohjainen käyttöjärjestelmä, jota käytetään useissa erilaisissa laitteissa. Vuonna 2003 Andy Rubin, Rich Miner, Nick Sears and Chris White perustivat Android Inc. nimisen yrityksen, jossa he alkoivat kehittää käyttöjärjestelmää mobiililaitteisiin. Aluksi he aikoivat toteuttaa käyttöjärjestelmän digitaalikameroihin, mutta heikkojen markkinoiden vuoksi he päättivät myöhemmin keskittyä matkapuhelimiin. Elokuussa 2005 Google osti yrityksen Android Inc.,ja vuonna 2007 Google sekä yli 30 muuta yritystä paljastivat Open Handset Alliance:n, joka koordinoi Android-käyttöjärjestelmän kehitystä. (Open Handset Alliance 2015.) Samalla julkaistiin Androidin ensimmäinen versio mobiililaitteille. Paljastus osoitti, ettei Android tulisi olemaan yhden alustan ja valmistajan käyttöjärjestelmä. Ensimmäisen Android puhelimen toi markkinoille HTC vuonna 2008. (Brachman 2014.)

Android on avoimen lähdekoodin alusta ja se onkin erittäin avoin rakenteensa osalta, jolloin sovelluskehittäjät pääsevät käyttämään sovelluksissaan Android-laitteiden ydintoimintoja. Sovellukset voivat suorittaa toimintoja, kuten esimerkiksi älypuhelimissa soittaa puheluita, lähettää tekstiviestejä, käyttää kameraa tai Bluetooth-yhteyksiä. Avoimuutta korostaa lisäksi Androidin perusta, joka on mobiilikäyttöön muokattu avoimen lähdekoodin Linux. Android tarjoaa sovelluskehittäjille runsaasti valmiita luokkakirjastoja ja työkaluja, joiden avulla mobiililaitteiden tarjoamia mahdollisuuksia voidaan yhdistellä monin tavoin ja luoda käyttäjille uusia käyttökokemuksia. (Open Handset Alliance 2015.)

3.2 Android Studio

Android Studio on Googlen virallinen kehitystyökalu Android-sovelluskehittäjille (Kuvio 2). Se pyrkii tarjoamaan kaikki tarpeelliset kehitystyökalut yhtenä kokonaisuutena ja antamaan sovelluskehittäjille mahdollisimman helposti käytettäviksi Googlen tarjoamia palveluita sekä rajapintoja. Ohjelmointia, testausta,

näyttöjen toteutusta sekä useita muita kehitystyön vaiheita helpottavien toimintojen lisäksi Android Studio tarjoaa myös mahdollisuuden emuloida erilaisia Android-laitteita. Tällöin kehitettävää sovellusta voi kokeilla nopeasti laitetta vastaavalla alustalla, jolloin puutteet toiminnassa havaitaan jo kehityksen varhaisessa vaiheessa. Android Studio tukee sovelluskehitystä Android älypuhelinten ja tablettien lisäksi Android TV, Android Wear, Android Auto ja Google Glass alustoilla. (Android Developer D 2015.)



Kuvio 2. Android Studio IDE:n näkymä näytön kehityksestä

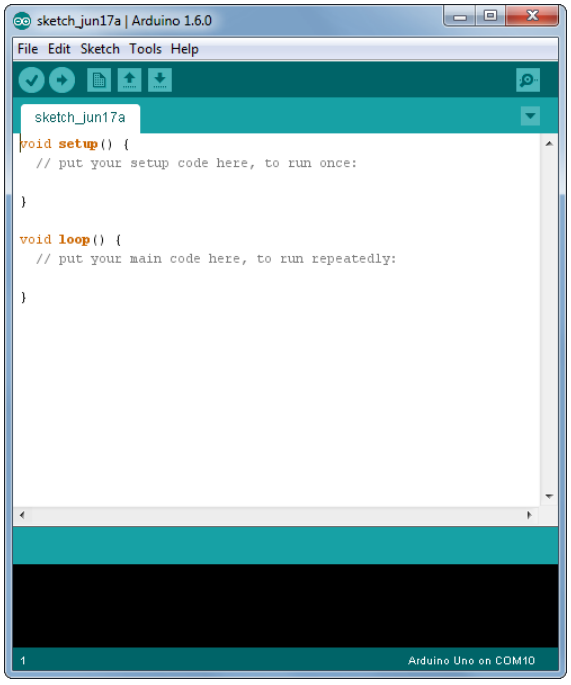
3.3 Arduino

Arduino-projekti sai alkunsa Italialaisessa Ivrean kaupungissa vuonna 2005, kun vuorovaikutteisuutta opiskelevien opiskelijoiden projekteihin tarvittiin edullinen ja helposti opittava ohjainlaite testattavien prototyyppien rakentamiseksi. Tällöin Arduinoa alettiin kehittää Hernando Barraganin Wiring-projektin pohjalta. (Creative 2015.) Prototyyppien rakentamisessa on kyse laitteista, jotka ovat vuorovaikutuksessa ihmisten, tietoverkkojen ja toisten laitteiden kanssa. Arduino mahdollistaa tällaisten laitteiden nopean kehityksen, koska sen avulla voidaan hyödyntää markkinoilla valmiiksi olevia tuotteita eikä kaikkea tarvitse rakentaa uudelleen alusta saakka. (Banzi 2011, 6.)

Arduinon fyysinen alusta on piirilevy, jonka päällä laitteita voidaan yhdistellä halutulla tavalla. Mikrokontrolleri on piirilevyn sydän ja kaikki sen tarvitsema on tehtaalla valmiiksi asennettu piirilevylle. Piirilevyllä on myös liitin, jonka kautta se voidaan kytkeä tietokoneeseen ohjelmointia varten. Arduinon piirilevy on siis pieni mikrokontrollerikortti, joka sisältää kokonaisen pieneen mikropiiriin rakennetun tietokoneen. (Banzi 2011, 17.)

3.4 Arduino IDE

Arduinon toinen pääosa on tietokoneelle asennettava avoimen lähdekoodin Arduino IDE. Se on Java-ohjelmointikielellä tehty kehitysympäristö, jolla kirjoitetaan ohjelmat ja ladataan ne Arduinon mikrokontrollerille suoritettavaksi tietokoneen USB-portin kautta. Arduino IDE:lla kirjoitettava yksinkertainen ohjelmointikieli perustuu Processing -ohjelmointikieleen, joka tulkitaan ensin C-kielelle ja sen jälkeen edelleen piirilevyn ymmärtämälle kielelle. Ohjelman tulkkaaminen lopulliseen muotoon tapahtuu, kun käyttäjä lataa tekemänsä ohjelman tietokoneelta mikrokontrollerille Arduino IDE:n avulla. (Banzi 2012, 20.) Kuviossa 3 esitetään Arduino IDE:n ohjelmointinäkymä sekä luonnoksen metodeja.

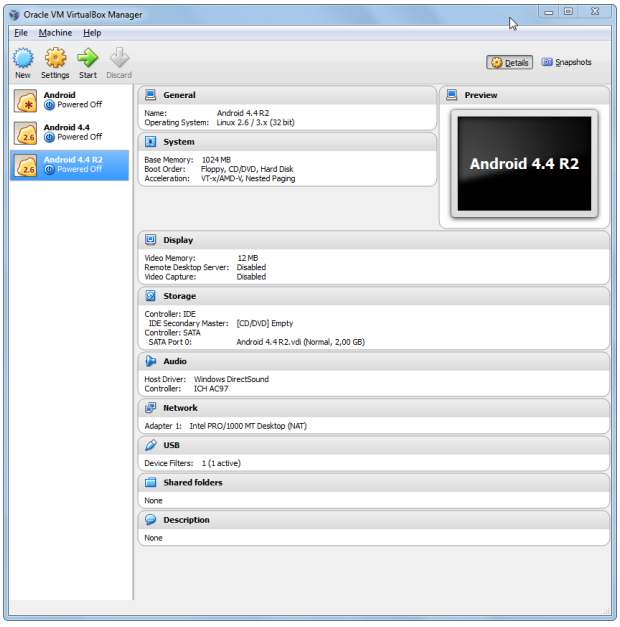


Kuvio 3. Arduino IDE

3.5 Virtual Box

Oracle:n VirtualBox on ilmainen virtualisointialusta, joka voidaan asentaa Windows, Linux, Mac tai Solaris käyttöjärjestelmää käyttäviin tietokoneisiin. Virtualisointi mahdollistaa useiden eri käyttöjärjestelmien ajamisen yhdellä tietokoneella samaan aikaan käytettäessä useita virtuaalikoneita. Samanaikaisten virtuaalikoneiden määrää rajoittaa ainoastaan isäntäkoneen muistin määrä sekä levytila. Virtualisoinnin avulla voidaan palvelinten kapasiteettia hyödyntää tehokkaammin, koska usein niiden kapasiteetista on käytössä vain pieni osa. (VirtualBox 2015.)

Ohjelmistokehityksessä käyttöjärjestelmien virtualisointi mahdollistaa kehitettävän ohjelman kokeilemisen eri käyttöjärjestelmissä samalla tietokoneella helposti ilman uudelleenkäynnistyksiä. Ohjelmistojen testaaminen ja kehittäminen voivat toisinaan sotkea käyttöjärjestelmän toiminnan, mutta virtualisoinnin avulla ympäristön palauttaminen alkutilaan on vaivatonta. Kopioimalla virtualisoitu kehitysympäristö kaikille projektin ohjelmistokehittäjille nopeutetaan heidän pääsemistä varsinaiseen kehitystyöhön ilman aikaa vievää ja mahdollisesti virheille altista asennustyötä. Oracle VirtualBox -virtualisointialustan hallinnointinäkymä on esitetty kuviossa 4.

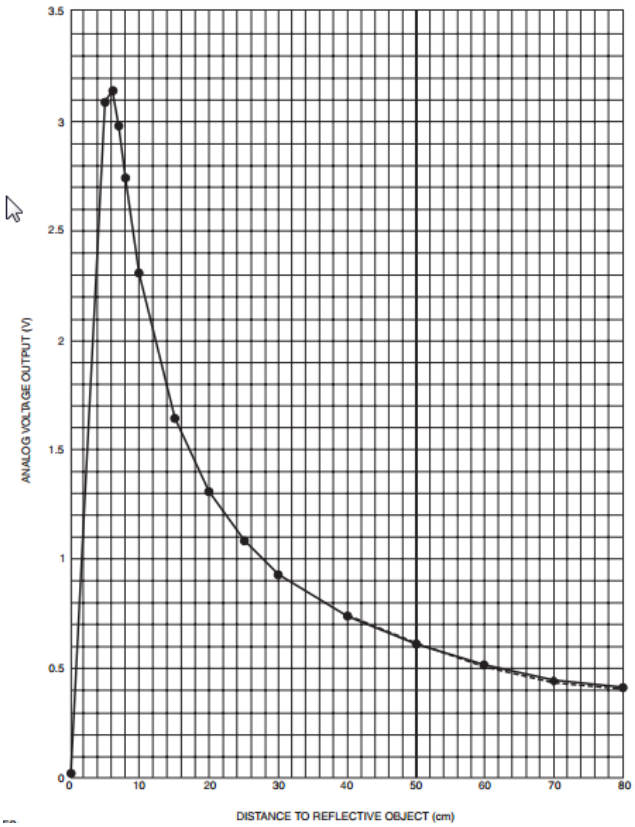


Kuvio 4. Oracle VirtualBox

3.6 Sensorit ja anturit

Sensorit ja anturit ovat laitteita, jotka muuttavat fyysisiä suureita sähkövirran vaihteluksi. Esimerkiksi valo, lämpö ja etäisyys ovat suureita, joita mikrokontrolleri ei voi ymmärtää ilman sensoreita. Mikrokontrolleri saa sensoreilta suureiden mukaan vaihtelevaa sähkövirtaa, jonka mukaan se voi päätellä kuinka toimia. Mikrokontrolleri tekee päätöksen ohjelmoijan laatiman ohjelmiston mukaisesti ja varsinainen reaktio tapahtuu käyttölaitteessa, joka voi olla esimerkiksi valo tai sähkömoottori. (Banzi 2011, 28.)

Kuviossa 5 esitetään tulokset, kuinka etäisyysanturilta tuleva jännite muuttuu etäisyyden mukaan eli etäisyys muutetaan mikrokontrollerin ymmärtämään muotoon. Kuvion muodostavalla anturilla suurin mittausetäisyys on 80 cm, jolloin ulostulojännite on pienimmillään. Suurimmillaan jännite on pienimmän mittausetäisyyden alueella, joka on noin 8 cm:n kohdalla. Valmistajan ilmoittama mittausväli kuviossa 5 esitetylle etäisyysanturille on 10–80 cm (Sharp 2005).



Kuvio 5. Etäisyysanturin ulostulojännitteen muutos etäisyyden mukaan.(Sharp 2005).

3.7 Bluetooth

Bluetooth-tekniikka mahdollistaa tiedonsiirron laitteiden välillä langattomasti, jos ne sijaitsevat riittävän lähellä toisiaan. Etäisyys laitteiden välillä voi olla enintään noin 100 metriä, mutta monissa tapauksissa huomattavasti vähemmän. Tekniikkaa käytetään maailmanlaajuisesti miljoonissa laitteissa, kuten autoissa, puhelimissa, kuulokkeissa, leluissa jne. Sen avulla voidaan siirtää energiatehokkaasti musiikkia, kuvia, dataa, videoita tai mitä yleensä voidaan siirtää esimerkiksi langallisen yhteyden välityksellä. Tietoa siirretään radioaalloilla digitaalisina signaaleina laitteesta toiseen taajuudella 2,400–2,485 GHz. Taajuusalue sijaitsee lisenssivapaalla Industry, Medical, Science (ISM) taajuusalueella. (Bluetooth 2015.)

Bluetooth on ruotsalaisen verkkolaitevalmistaja Ericssonin keksintö vuodelta 1994, jolloin yrityksessä pyrittiin kehittämään langaton yhteys entisen RS-232

langallisen yhteyden tilalle. Vuonna 1998 Ericsson, Nokia, Intel, Toshiba ja IBM muodostivat yhdessä Bluetooth SIG:n (Special Interest Group), jonka jälkeen yksikään yrityksistä ei omistanut enää tekniikkaa yksin. Tavoitteena oli luoda yksi standardi tiedonsiirtoon sekä pyrkiä yhdessä edistämään Bluetooth-tekniikan kehitystä. Samalla nimi Bluetooth otettiin käyttöön. Nimellä on pohjoismainen yhteys, koska se annettiin 900-luvulla eläneen tanskalaisen kuningas Harald Blåtandin mukaan eli englanniksi kuninkaan nimi oli Harald Bluetooth. (Bluetooth 2015.)

Tällä hetkellä Bluetooth v4.0 on Bluetooth-tekniikan tuorein versio. Se sisältää uusia ominaisuuksia, mutta ei kuitenkaan ole edeltäjäänsä nopeampi. Suurin nopeus on tällä hetkellä 24 Mbit/s. Bluetooth v4.0:n uusista ominaisuuksista Bluetooth low energy -tekniikka mahdollistaa pienien lähettimien toiminnan pitkän aikaa ilman paristojen vaihtoa. Tämä ominaisuus luo lukuisia uusia käyttökohteita Bluetooth-laitteille. (Bluetooth 2015.)

4 KEHITYSYMPÄRISTÖN ASENNUS

Kehityskoneena oli Intel-prosessorilla varustettu kannettava tietokone, jossa oli asennettuna Windows 7 käyttöjärjestelmä, Java SE Development Kit 8 (JDK 8) sekä erillinen USB Bluetooth -sovitin. Aluksi varmistettiin, että kehityskone täytti kaikkien asennettavien ohjelmistojen vaatimukset. Ennen asennuksia täytyi kehityskoneen BIOS päivittää sekä ladata kaikki tarvittavat ohjelmistot verkosta, jossa ne ovat vapaasti saatavilla. Ohjelmistojen asennukset tehtiin pääasiassa oletusasetuksin, jolloin asennuksissa tuli mukana työn kannalta tarpeettomiakin ominaisuuksia. Ylimääräiset ominaisuudet voi jättää asentamatta, kun kokemus lisääntyy Android-ohjelmistokehityksestä ja tarvittavasta kehitysympäristöstä.

4.1 VirtualBox ja Android-käyttöjärjestelmä

VirtualBox versio 4.3.20 ladattiin osoitteesta https://www.virtualbox.org/wiki/Downloads, ja asennuspaketti on suoritettava tiedosto (.exe). VirtualBox asennus tehdään asennusvelhon opastamana, jonka tärkeimpinä kohtina käyttäjä valitsee asennuspolut sekä asennetaanko VirtualBoxin ehdottamia järjestelmälaitteita. Valinnoista riippuu, voiko VirtualBox käyttää esimerkiksi isäntäkoneen USB-porttiin liitettyä laitetta. Työn kannalta oli tärkeää sallia USB-porttiin liittyvän laitteen asennus, koska Bluetooth-sovitin täytyi saada käyttöön Android-käyttöjärjestelmässä. VirtualBox vaati lisäksi laajennuksen (Extension Pack) asennuksen, joka mahdollisti tuen USB 2.0 laitteille. Extension Pack on ladattavissa verkosta samalta sivulta kuin VirtualBox.

VirtualBox asennuksen jälkeen luotiin virtuaalikone, johon asennettiin Android käyttöjärjestelmä. Virtuaalikoneen luominen tehdään myös vaiheittain asennusohjelman ohjaamana. Ensimmäisessä vaiheessa virtuaalikoneelle täytyi antaa tyyppi ja versio, joihin valittiin tyypiksi Linux ja versioksi Linux 2.6/3.x (32 bit). Seuraavaksi asetettiin RAM-muistin määräksi 1024MB ja luotiin VDI (VirtualBox Disk Image) tyyppinen virtuaalinen kiintolevy. Kiintolevyn koko asetettiin kiinteästi 2 GB:iin.

Virtuaalikoneen luonnin jälkeen aloitettiin Android 4.4 käyttöjärjestelmän asennus. Virtuaalikoneen käynnistyksen jälkeen kysyttiin käynnistyslevyä, johon valittiin osoitteesta http://www.android-x86.org/download ladattu Android 4.4 ISO-tiedosto. Seuraavaksi valittiin Android asennus kiintolevylle sekä luotiin kiintolevylle uusi 2 GB:n ”buuttaava” pääosio. Pääosio alustettiin käyttämään ext3 -tiedostojärjestelmää ja asennettiin GRUB-käynnistyslatain. Tämän vaiheen jälkeen Android käynnistettiin ja asetettiin sen alkuasetukset, joista langattoman verkon ja Google-tilien asetukset jätettiin asettamatta.

Luodun virtuaalikoneen verkkoasetuksiin lisättiin portin uudelleenohjaus portista 5555 porttiin 5555 ja protokollaksi asetettiin TCP. Asetukset liittyvät Android Debug Bridge (ADB) -työkaluun, jonka avulla voidaan ottaa yhteys virtuaalikoneessa olevaan Androidiin. ADB koostuu kolmesta osasta, joista kaksi toimii kehityskoneella ja yksi emulaattorissa tai Android-laitteessa. Kehityskoneella ovat ADB:n asiakas ja palvelin, joista sovelluskehittäjä toimii asiakkaan kanssa. Palvelin toimii puolestaan taustapalveluna hoitamassa yhteyksiä asiakkaan ja esimerkiksi Android-puhelimen välillä. Android-laitteessa tai emulaattorissa toimii taustapalveluna prosessi, johon kehityskoneen asiakas voi ottaa yhteyden palvelimen hallinnoimana. (Android Developer C 2015.)

4.2 Android Studio

AndroidStudio version 1.0.1 asennuspaketti on suoritettava tiedosto ja asennus tehdään asennusvelhon opastamana vaiheittain. Paketti ladattiin osoitteesta https://developer.android.com/sdk/index.html. Suoritettavien vaiheiden aikana valitaan asennettavat komponentit sekä asennuspolut. Asennusvelho tarkastaa asennuksen aikana, että täyttääkö tietokone vaatimukset JDK:n ja muistin määrän osalta. Käyttäjä voi valita asennetaanko komponentit Android Software Development Kit (SDK), Android Virtual Device (AVD) ja Intel Hardware Accelerated Execution Manager (Intel HAXM). HAXM on Intelin tekniikkaa, jolla nopeutetaan emulaattorin toimintaa, mutta se asettaa myös lisää vaatimuksia tietokoneen prosessorille.

Android SDK sisältää koodikirjastojen lisäksi monenlaisia Android-sovelluskehityksen kannalta pakollisia työkaluja. Sitä voi käyttää myös muiden kehitysympäristöjen kanssa kuin Android Studion. Android SDK:n sisältämät työkalut on jaettu kahteen ryhmään, joista toiset ovat riippumattomia Android-alustasta. Alustariippuvaiset työkalut ovat yleensä muokattu tukemaan tuoreinta versiota Androidista, mutta ovat myös taaksepäin yhteensopivia vanhojen versioiden kanssa. Tärkeimmät SDK:n työkalut ovat SDK Manager, AVD Manager, emulaattori ja Dalvin Debug Monitor Server (DDMS). Edellä mainitut ovat riippumattomia Android-versiosta. Android SDK ei ole täydellinen paketti vaan se on lähinnä aloituspaketti, jota voi täydentää SDK managerin avulla lataamalla omaan tarpeeseen sopivat komponentit. (Android Developer E 2015.)

Android Virtual Device (AVD) on virtuaalikone, jonka avulla kehitettävää Android-sovellusta voi kokeilla oikeaa laitetta mallintavassa ympäristössä. AVD voi mallintaa laitteen eri ominaisuuksia, kuten näytön kokoa, muistin määrää tai sen avulla voi kokeilla esimerkiksi kameraa ja Bluetooth -yhteyttä. Erilaisten laitteiden malleja voi lisäksi ladata verkosta tai luoda itse (Techotopia 2015.) Asennusvelho voi luoda asennuksen yhteydessä valmiiksi AVD:n, mutta niitä voi luoda myöhemmin AVD Managerin avulla.

4.3 Arduino IDE

Arduino IDE -version 1.6.0 asennuspaketti on suoritettava tiedosto ja asennus tehdään asennusvelhon opastamana. Paketti ladattiin osoitteesta https://www.arduino.cc/en/Main/Software. Ensimmäisenä kohtana valitaan kaikki asennettavat komponentit, joista valittiin kaikki oletusvalintojen mukaisesti. Vaihtoehtoina ovat Arduinon USB-ajuri, käynnistyskuvakkeiden luonti ja INO-tiedostojen liittäminen Arduino IDE:en. Seuraavana kohtana valitaan asennuskansio, jonka jälkeen kysytään vielä vahvistus asennuksesta. Näiden vaiheiden jälkeen suoritetaan Arduino IDE:n varsinainen asennus.

Asennuksen valmistumisen jälkeen kytkettiin Arduino-piirilevy tietokoneen USB-porttiin ja tarkastettiin laitehallinnan kautta yhdistämisen onnistuminen. Laitehal-

linnassa näkyi Portit-valinnan alla Arduino UNO sekä sen perässä COM-portti, johon piirilevy oli yhdistetty. Seuraavaksi käynnistettiin Arduino IDE, jonka Tools-menusta valittiin Boards-valikko sekä sen alta avautuvista vaihtoehdoista valittiin Arduino UNO. Toisena kohtana Tools-menusta valittiin Port-valikko ja avautuvista vaihtoehdoista valittiin sama COM-portti, joka näkyi laitehallinnan kautta. Näiden vaiheiden jälkeen Arduino IDE oli valmis lataamaan ohjelmat Arduinon piirilevylle.

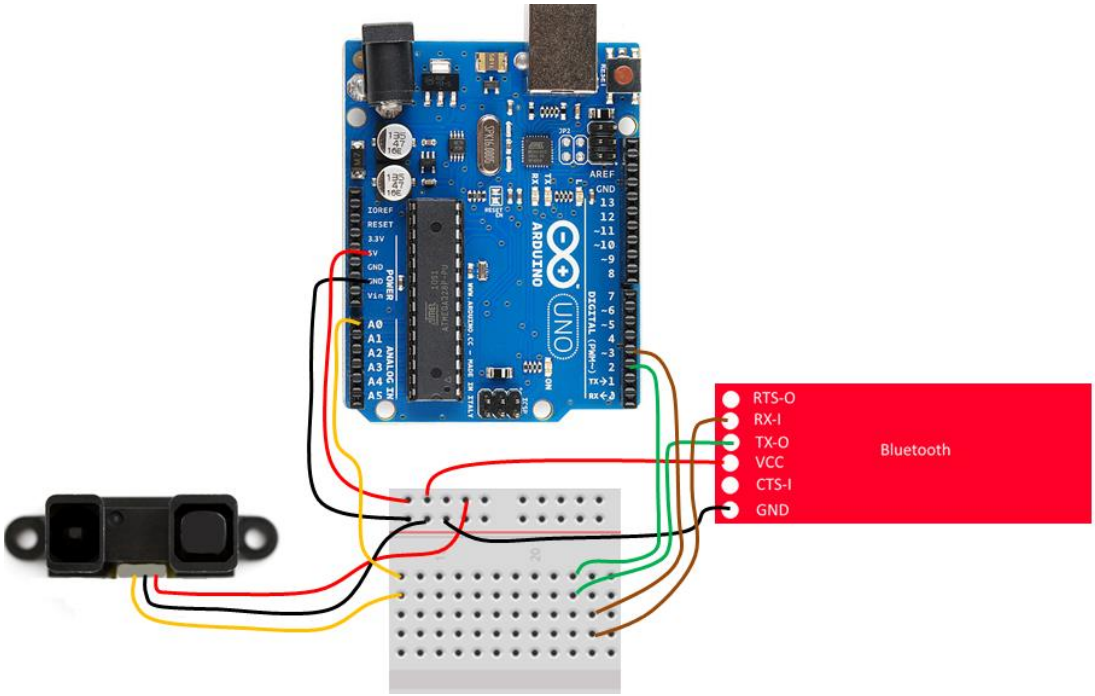
5 SOVELLUKSEN TOTEUTUS

5.1 Arduino kytkennät

Arduino UNO -piirilevy kytkettiin etäisyysanturiin Sharp GP2Y0A02YK0F (Kuvio 6) sekä Bluetooth-lähettimeen Bluetooth Mate Silver (Kuvio 7) kytkentälevyn kautta. Piirilevyn liitännät voidaan jakaa useammalle laitteelle kytkentälevyn avulla, joten se on suositeltava hankinta piirilevyn yhteyteen. Etäisyysanturin ja Bluetooth-lähettimen tarvitsema virta otettiin Arduino-piirilevyn 5 V:n liittimestä ja ne maadoitettiin piirilevyn GND-maadoitusliittimeen. Bluetooth-lähettimen virtaliitin on VCC ja maadoitusliitin GND. Kytkennät on esitetty kuviossa 8.

Kuvio 6. Infrapuna etäisyysanturi (Digi-Key 2015)

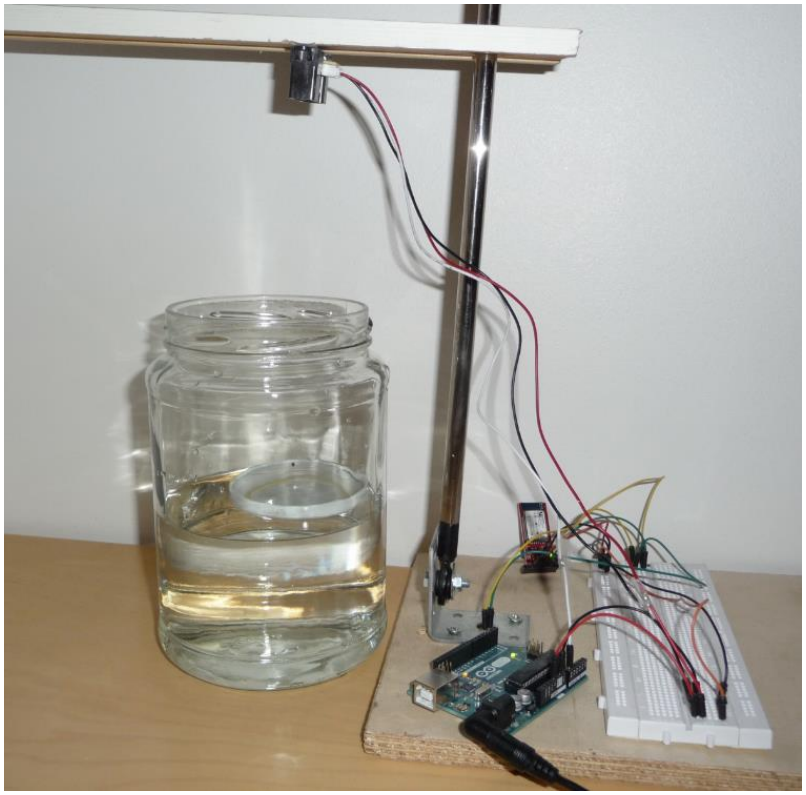
Kuvio 7. Bluetooth-lähetin (SparkFun 2015)



Kuvio 8. Arduino-piirilevyn, etäisyysanturin ja Bluetooth-lähettimen kytkennät

Arduino UNO -piirilevyllä on 14 digitaalista ja 6 analogista liitintä. Digitaalisia liittimiä voi käyttää datan lähetykseen ja vastaanottamiseen, mutta analogiset liittimet ovat pelkästään vastaanottamista varten. Työssä käytettiin analogista liitintä A0 ottamaan vastaan etäisyysanturin lähettämää dataa sekä kahta digitaalista liitintä. D3 digitaalisen liittimen kautta otetaan vastaan Bluetooth-lähettimen RX-I-liittimeltä tulevaa dataa ja D2-liittimen kautta voidaan siirtää dataa lähetettäväksi Bluetooth-lähettimen TX-O-liittimeen. Analogisista ja digitaalisista liittimistä voidaan käyttää mitä tahansa liitintä, koska niiden toimintoja ohjataan mikrokontrollerille ladattavan ohjelmiston kautta.

Arduino UNO -piirilevyn tarvitsema virta voidaan syöttää USB-liittimen kautta tietokoneelta tai erikseen kytkettävällä virtalähteellä, joiden väliltä virtalähde valitaan automaattisesti. Erillinen virtalähde voi olla esimerkiksi paristo tai muuntaja, mutta virtalähteen jännitteen suositellaan olevan välillä 7–12 V. (Arduino 2015.) Työssä käytettiin kehitysvaiheessa USB-liittimen kautta syötettävää virtaa ja testausvaiheessa 12 V:n muuntajaa. Kuviossa 9 on esitetty mittauksia varten rakennettu testijärjestelmä.



Kuvio 9. Mittauksen testijärjestelmä

5.2 Järjestelmän toteutus

5.2.1 Projektin luonti Android Studiolla

Sovelluksen toteuttaminen aloitettiin luomalla Android Studiossa uusi Java-projekti. Android Studion käynnistämisen jälkeen avautui ikkuna, josta valittiin uuden projektin luonti. Seuraavassa ikkunassa annettiin projektille nimi, polku projektin kotikansioon sekä domain-nimi. Domain-nimen perustella muodostui luokkien pakettirakenne. Tämän jälkeen avautuvassa ikkunassa valittiin laitteeksi puhelin ja tablet, joille sovellus oli tarkoitus tehdä. Seuraavaksi Android Studio antoi valita, aloitetaanko projekti aivan alusta vai luodaanko sovellukselle jokin tarjottavista pohjista. Ikkunasta valittiin tyhjän Activity:n luonti, jolloin Android Studio loi valmiiksi tyhjän näytön, Activity-luokan ja muita tarvittavia resursseja sekä nimesi ne viimeisessä ikkunassa annettujen arvojen mukaisesti.

Android Studion luomaa sovellusta kokeiltiin heti virtuaalikoneessa. VirtualBox käynnistettiin ja sen jälkeen myös siinä ajettava Android-virtuaalikone käynnis-

tettiin. Seuraavaksi annettiin Windowsin komentokehotteessa Android Debug Bridge komento ”adb connect 127.0.0.1”, joka avasi yhteyden virtuaalikoneessa ajettavaan Androidiin. Komennossa ei tarvitse antaa porttia 5555, koska sitä käytetään oletuksena. Yhdistämisen jälkeen käynnistettiin juuri luotu sovellus Android Studiossa *Run*-komennolla ja hetken kuluttua avautui dialogi Android-laitteen valintaa varten. Dialogissa oli valittavissa virtuaalikoneessa käynnissä oleva Android-laite, jonka valitsemisen tuloksena virtuaalikoneessa avautui Android Studion luoman sovelluksen pääikkuna.

5.2.2 Aktiviteetin toteuttaminen

Android-sovelluksessa aktiviteetti (Activity) on komponentti, joka yleensä tarjoaa yhden näytön sovelluksen kanssa kommunikointiin. Sovelluskehittäjä periyttää aktiviteetin toteuttavan luokan Activity-luokasta ja toteuttaa siihen tarvittavat toiminnot. Lisäksi siinä määritetään mihin näyttöön luokka liittyy. OnCreate on ensimmäinen metodi, jota Android-käyttöjärjestelmä kutsuu aktiviteetin käynnistyksessä. OnCreate-metodi täytyy toteuttaa, ja siinä määritetään setContentView-metodilla aktiviteetin näytön toteuttava XML-tiedosto. Koodiesimerkissä 2 asetetaan näytön toteuttavaksi tiedostoksi activity_main.xml, jonka tunniste on R.layout.activity_main.

Tässä työssä aktiviteetti luotiin Bluetooth-yhteyksien etsintänäytölle sekä päänäytölle ja konfigurointinäytölle. Ennen kuin Android voi käynnistää aktiviteetin, niin aktiviteetin täytyy olla lisättynä AndroidManifest.xml-tiedostoon (Koodiesimerkki 1). Uuden näytön avaaminen tehtiin Intent-luokan avulla, kuten tehdään koodiesimerkissä 3 Bluetooth-yhteyksien etsintänäytön avaamiseksi. Intent-luokkaa voi käyttää muissakin tehtävissä, mutta usein sitä käytetään näyttöjen avaamisessa. Se toimii yhdistävänä tekijänä aktiviteettien välillä ja sen avulla voi lähettää tietoa aktiviteetista toiseen (Android Developer F 2015).

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name="fi.distance.activity.MainActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <activity
         android:name="fi.distance.activity.BluetoothActivity"
         android:label="Bluetooth"
         android:parentActivityName="fi.distance.activity.MainActivity" >
         <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="fi.distance.activity.MainActivity" />
      </activity>
      …
   </application>
</manifest>
```

Koodiesimerkki 1. AndroidManifest.xml

```
public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      …
   }
}
```

Koodiesimerkki 2. MainActivity-luokka ja onCreate-metodi

```
Intent intent = new Intent(this, BluetoothActivity.class);
startActivity(intent);
```

Koodiesimerkki 3. Aktiviteetin käynnistys

5.2.3 Näytön toteutus

Aktiviteetteihin liitettävien näyttöjen ulkoasu toteutettiin XML-tiedostojen avulla. Tiedostoon lisätään näytöllä tarvittavat komponentit ja niiden asetteluun voidaan käyttää erilaisia asettelumalleja. Tässä työssä käytettiin asettelumalleja LinearLayout ja RelativeLayout, joilla on toisistaan poikkeava tapa asetella komponentit näytölle. LinearLayout asettelee lapsikomponentit riveille vaaka- tai pystysuunnassa, kun RelativeLayout asettelee ne suhteessa toisiinsa. Koodiesimerkissä 4 olevassa XML-tiedostossa on käytetty LinearLayout-asettelumallia pystysuunnassa tekstikentän (TextView) ja painikkeen (Button) asettelussa. Kuviossa 10 näkyy miten komponentit asettuvat sen mukaisesti näytölle. LinearLayout vaikutti helpommin hallittavalta toteuttajan kannalta.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >
    <TextView android:id="@+id/text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Hello, I am a Button" />
</LinearLayout>
```

Koodiesimerkki 4. Näytön toteuttava xml-tiedosto (Android Developer K 2015)

Kuvio 10. XML-tiedoston muodostama näyttö

5.2.4 Näytön tapahtumien käsittely

Käyttäjän toimet näytöllä käsiteltiin näyttöön liitetyssä aktiviteetissa, kuten tehdään koodiesimerkissä 5. Näytön objekteille täytyi antaa yksilöivä tunniste (ID) XML-tiedostossa, jonka perusteella ne voitiin tunnistaa Activity-luokassa. Koodiesimerkissä 5 yksilöivä tunniste painikkeelle on R.id.button_main_config. Activity-luokissa toteutettiin rajapintaluokka View.OnClickListener, jonka onClick-metodissa käsiteltiin tarvittavat tapahtumat, kuten nappien painallukset. View-luokan getId-metodin avulla saatiin selville tapahtuman laukaissut objekti.

```
public class MainActivity extends Activity implements View.OnClickListener {
  …
  public void onClick(View v){
      switch(v.getId())
      {
          case R.id.button_main_config:
              Intent intent = new Intent(this, ConfigActivity.class);
              startActivity(intent);
              break;
              …
      }
}
```

Koodiesimerkki 5. Tapahtumien käsittely Activity-luokassa

5.2.5 Bluetooth käynnistys

Bluetooth-yhteyden muodostaminen aloitettiin tutkimalla onko Android-laitteessa Bluetooth tuettuna. Tämä tehtiin kutsumalla BluetoothAdapter-luokan getDfautAdapter-metodia, joka palauttaa null-arvon tuen puuttuessa tai vastaavasti BluetoothAdapter-objektin tuen löytyessä. BluetoothAdapter on hyvin keskeinen luokka Bluetooth-toimintojen kannalta. Sen avulla voidaan suorittaa tärkeimpiä Bluetooth-toimintoja, kuten toisten Bluetooth-laitteiden etsintä sekä paritus, näyttää paritetut laitteet ja aloittaa yhteys laitteiden välillä. (Android Developer G 2015.)

Bluetooth-yhteyden muodostaminen vaatii oikeuksien lisäämisen AndroidManifest.xml tiedostoon (Koodiesimerkki 6). Android.permission.BLUETOOTH antaa sovellukselle oikeudet muodostaa yhteyden paritetun Bluetooth-laitteen kanssa. Android.permission.BLUETOOTH_ADMIN antaa puolestaan oikeudet laitteelle etsiä toisia Bluetooth-laitteita ja muodostaa parin niiden kanssa. Merkittävä piirre Android-käyttöjärjestelmässä on, ettei sovelluksilla ole oletuksena oikeuksia tehdä mitään sellaista, jolla olisi vaikutuksia toisiin sovelluksiin, käyttöjärjestelmään tai käyttäjään. Sovelluksen asennuksessa Android-laitteelle käyttäjältä kysytään lupa, saako asennettava sovellus suorittaa toimintoja, jotka liittyvät AndroidManifest.xml tiedostossa annettuihin oikeuksiin (Android Developer H 2015).

Android-laitteen Bluetooth-sovitin aktivoitiin Intent- ja Activity-luokkien avulla, kuten koodiesimerkissä 7 esitetään. Activity-luokasta käytettiin startActivityForResult-metodia, joka käynnistää aktiviteetin Bluetooth:n aktivoimiseksi Android-käyttöjärjestelmässä. Ennen aktivointia Android-käyttöjärjestelmä kysyy käyttäjältä vahvistuksen toiminnon suorittamiseksi. Metodi startActivityForResult palauttaa myös paluuarvon ja se voidaan ottaa kiinni Activity-luokan onActivityResult-metodissa parametrina annetun requestCode:n avulla.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   <uses-permission android:name="android.permission.BLUETOOTH"/>
   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
  …
</manifest>
```

Koodiesimerkki 6. Bluetooth-oikeudet AndroidManifest.xml -tiedostossa

```
Intent turnOnIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
activity.startActivityForResult(turnOnIntent,   1);
```

Koodiesimerkki 7. Bluetooth aktivointi

5.2.6 Bluetooth laitteiden etsintä

Bluetooth-sovittimen aktivoinnin jälkeen voitiin etsiä lähistöllä olevia muita Bluetooth-laitteita ja näyttää löydetyt laitteet käyttäjälle. Etsintä käynnistettiin BluetoothAdapter-luokan startDiscovery-metodilla. StartDiscovery-metodi on asynkroninen, jolloin metodista palataan heti ja haku jatkuu Android käyttöjärjestelmässä. Bluetooth-laitteiden etsinnän tuottamat viestit otettiin vastaan BroadcastReceiver-luokan onReceive-metodissa (Koodiesimerkki 9). BroadcastReceiver on abstrakti luokka, joten siitä periytettiin luokka bcReceiver ja toteutettiin sen onReceive-metodi. Uudesta luokasta luotu bcReceiver-objekti annettiin intentFilter-objektin kanssa parametriksi Context-luokan registerReceiver-metodille, joka rekisteröi BroadcastReceiver:n vastaanottamaan IntentFilter-objektissa määrättyjä viestejä (Koodiesimerkki 8).

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(BluetoothDevice.ACTION_FOUND);
intentFilter.addAction(BluetoothDevice.ACTION_UUID);
intentFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
intentFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
```

Koodiesimerkki 8. IntentFilter viestien asetus

```
public class bcReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
      String action = intent.getAction();
      switch (action) {
          case BluetoothDevice.ACTION_FOUND:
                BluetoothDevice device = intent.getParcelableExtra(
                      BluetoothDevice.EXTRA_DEVICE);
              …
          case BluetoothDevice.ACTION_UUID:
              …
          case BluetoothAdapter.ACTION_DISCOVERY_STARTED:
              …
          case BluetoothAdapter.ACTION_DISCOVERY_FINISHED:
              …
      }
  }
}
```

Koodiesimerkki 9. BroadcastReceiver onReceive-metodin toteutus

Viesti BluetoothAdapter.ACTION_DISCOVERY_STARTED saapuu, kun Bluetooth-laitteiden etsintä alkaa. Haun päättyessä tulee BluetoothAdapter.ACTION_DISCOVERY_FINISHED viesti. BluetoothDevice.ACTION_FOUND saapuu toisen Bluetooth-laitteen löydyttyä ja tästä viestistä poimittiin talteen siihen liitetty laite eli BluetoothDevice-objekti. BluetoothDevice-objekti sisältää tietoja toisesta laitteesta, kuten esimerkiksi nimen ja osoitteen. Laite saadaan otettua Intent-objektista metodilla getParcelableExtra.

Haun päätyttyä listalle lisättyihin laitteisiin tehtiin kysely BluetoothDevice-luokan fetchUuidsWithSdp-metodilla, joka palauttaa laitteen palveluiden UUID:t (Universally Unique Identifier). Jos kysely toiselle laitteelle onnistui, niin laite lisättiin näytettäväksi käyttäjälle. UUID on tunniste, jonka avulla voidaan yksilöidä esimerkiksi oma sovellus. Tässä työssä käytettiin SPP:n (Serial Port Profile) UUID:a, jonka tunniste on 00001101-0000-1000-8000-00805F9B34FB, avaamaan yhteys Arduinon lähettimen kanssa (Android Developer I 2015).

5.2.7 Bluetooth-laitteiden paritus

Bluetooth-laitteiden etsinnän jälkeen tehtiin paritus oman laitteen ja löydettyjen laitteiden välillä, mutta se ei kuitenkaan vielä merkitse yhteyden avaamista (Koodiesimerkki 10). Laitteiden parituksella tarkoitetaan, että laitteet tietävät toistensa olevan olemassa ja ne voivat avata keskenään salatun yhteyden. Laitteiden välillä on yhteys, kun laitteet jakavat RFCOMM (Radio Frequency Communication) kanavan ja voivat siirtää dataa välillään. (Android Developer I 2015.) RFCOMM-protokolla emuloi sarjakaapeliyhteyttä kahden RS-232 sarjaportin välillä, mutta tiedonsiirto tapahtuu langattomasti (Bluetooth Developer portal 2015).

```
Method method = device.getClass().getMethod("createBond", (Class[]) null);
method.invoke(device, (Object[]) null);
```

Koodiesimerkki 10. Laitteiden paritus

Laitteiden parituksen jälkeen voidaan avata yhteys tiedonsiirtoa varten. Yhteyden avaus suoritettiin uudessa säikeessä, jotta sovellus voi tarvittaessa jäädä odottamaan yhteyden avautumista jumiuttamatta koko sovellusta. Thread-luokasta periytettiin oma luokka, jossa aluksi luotiin BluetoothSocket-olio käyttäen BluetoothDevice-luokan metodia createRfcommSocketToServiceRecord. Seuraavaksi yritettiin yhteyttä run-metodissa toiseen laitteeseen BluetoothSocket-luokan connect-metodilla. Yhteyden avauduttua luotiin uusi säie vastaanottamaan viestit Bluetooth-yhteyden kautta, joka on esitetty koodiesimerkissä 11.

```
public class BluetoothConnectThread extends Thread {
    private BluetoothSocket mmServerSocket;
    public BluetoothConnectThread(BluetoothDevice device) {
        ...
        mmServerSocket = device.createRfcommSocketToServiceRecord(
                UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"));
        ...
    }

    public void run() {
        ...
        mmServerSocket.connect();
        bluetoothThread = new BluetoothThread(mmServerSocket);
        bluetoothThread.start();
        ...
    }
}
```

Koodiesimerkki 11. Bluetooth-yhteyden avaaminen

Koodiesimerkissä 12 esitetään, kuinka toisen Bluetooth-laitteen lähettämä data otettiin vastaan käyttämällä InputStream-oliota. InputStream-olio saatiin BluetoothSocket-luokan getInputStream-metodilla. Viestit luettiin silmukassa käyttäen apuna Reader-luokan kääreluokkaa BufferedReader ja sen metodia readLine.

```
public class BluetoothThread extends Thread {
    private InputStream inputStream;
    public BluetoothThread(BluetoothSocket socket) {
        …
        inputStream = socket.getInputStream();
    }

    public void run() {
         InputStreamReader reader = new InputStreamReader(inputStream);
         BufferedReader bufferedReader = new BufferedReader(reader);
         String sensorMessage;
         while ((sensorMessage = bufferedReader.readLine()) != null) {
              …
         }
         ….
    }
    …
}
```

Koodiesimerkki 12. Viestien vastaanottaminen Bluetooth-yhteyden kautta

5.2.8 Viestien lähettäminen näytölle

Android-käyttöjärjestelmässä säikeet eivät voi vaihtaa tietoja suoraan, joten Bluetooth-viestejä vastaanottavasta säikeestä lähetettiin tarvittavat tiedot pääsäikeeseen näytölle esitettäväksi Handler-luokan avulla. Handler-luokasta periytettiin oma luokka MessageHandler, jossa toteutettiin handleMessage-metodi viestien poimintaa varten viestijonosta. Handler toimii vain sen säikeen kanssa, jossa Handler-olio on luotu. Handler-olio luotiin pääsäikeessä, jolloin se on kiinnitetty pääsäikeeseen sekä pääsäikeen viestijonoon.

Bluetooth-yhteyttä kuuntelevassa säikeessä luotiin Message-objekti, johon lisättiin lähetettävä data Bundle-oliota käyttäen (Koodiesimerkki 13). Lähetettävä data laitettiin Bundle-luokan olioon, koska sitä voidaan käyttää Android-sovelluksissa tiedon välityksessä esimerkiksi aktiviteettien välillä. Bundle-luokan putString-metodilla asetettiin lähetettävä arvo sekä avain, jonka perusteella arvo saadaan poimittua Handler-luokan handleMessage-metodissa (Koodiesimerkki

14). Handler-objekti täytyi luoda pääsäikeessä ja välittää viestejä kuuntelevaan säikeeseen, jotta viestit saatiin käsiteltyä ja esitettyä oikeassa muodossa näytöllä.

```
Message msgObj = new Message();
Bundle bundle = new Bundle();
bundle.putString(“MsgKey”, sensorMessage);
msgObj.setData(bundle);
mHandler.sendMessage(msgObj);
```

Koodiesimerkki 13. Viestien lähettäminen säikeestä

```
public class MessageHandler extends Handler{
    ...
    public void handleMessage(Message message) {
        String sensorMessage = message.getData().getString("MsgKey");
        ...
    }
}
```

Koodiesimerkki 14. Viestien käsittely Handler-luokassa

5.2.9 Viestien käsittely

Arduinon lähettämät viestit luettiin säikeessä BufferedReader-luokan readLine-metodilla, jolloin viesteissä täytyi olla merkintä ”/n” jokaisen mittaustuloksen perässä. Lisäksi etäisyysanturin arvoon lisättiin alkuun kirjain yksilöimään anturia. Tässä työssä käytettiin ainoastaan yhtä anturia, mutta niitä on mahdollista liittää useampia ja sen vuoksi ne täytyy pystyä tunnistamaan. Lisätyt merkit poistettiin Arduinolta saadusta viestistä ja arvo muunnettiin kahden desimaalin tarkkuudella esitettäväksi luvuksi.

Etäisyysanturin lähettämä arvo on integer-tyyppinen luku, joka täytyy muuntaa ymmärrettävään mittayksikköön eli tässä tapauksessa senttimetreiksi. Erilaiset etäisyysanturit lähettävät erilaisia arvoja ja valmistajat ilmoittavat eri anturityypeillensä sopivat muuntokaavat. Työssä käytetyn anturin lähettämät arvot muunnetaan senttimetreiksi seuraavalla kaavalla: *Distance (cm)* =

9462/(SensorValue - 16.92) (RobotItaly 2015). Muunnokseen käytetty Java-koodi on esitetty koodiesimerkissä 15.

```
public static BigDecimal convertSensorValue(String sensorValue){
    int measureValue = Integer.valueOf(sensorValue);
    double centimeters = 9462/(measureValue - 16.92);
    BigDecimal distance = new BigDecimal(centimeters);
    return distance.setScale(2, RoundingMode.CEILING);
}
```

Koodiesimerkki 15. Anturin arvon muuntaminen senttimetreiksi

5.2.10 Tiedoston tallentaminen ja lukeminen

Android-sovellukseen tehtiin ominaisuus mittaustuloksiin liittyvien viestien näyttämiseksi käyttäjälle, minkä avulla voitiin tutustua myös tiedoston käsittelyyn Android-ohjelmoinnissa. Tietojen tallentamisen voi hoitaa myös tietokannan avulla, mutta nyt tallennettavien tietojen määrä oli niin pieni, että tiedostoon tallentaminen soveltui tarkoitukseen hyvin. Tarvittavat tiedot tallennettiin tiedostoon määrämuotoisena, jotta ne pystyttiin purkamaan olioihin käyttöä varten.

Androidissa voi käyttää sisäistä tai ulkoista tietovarastoa tiedostojen tallentamiseen, joista nyt käytettiin sisäistä. Käytettäessä sisäistä tietovarastoa tiedostot ovat oletuksena vain sovelluksen käytettävissä, ne poistetaan sovelluksen poiston yhteydessä ja ne ovat aina sovelluksen käytettävissä. Ulkoista tietovarastoa on suositeltavaa käyttää silloin, kun tiedostojen halutaan olevan muidenkin sovellusten saatavilla ja niillä ei ole tiukkoja vaatimuksia tietoturvan suhteen. Ulkoisilla tietovarastoilla tarkoitetaan esimerkiksi muistikortteja, USB-laitteita ja muita liitettäviä tallennusvälineitä (Android Developer J 2015).

Tiedoston luonnissa ja kirjoituksessa käytettiin tallennuskansion absoluuttisen polun palauttavaa Context-luokan getFilesDir-metodia. Polku ja tallennettavan tiedoston nimi annettiin parametreiksi File-luokan konstruktoriin, jonka jälkeen luotua File-oliota käytettiin FileWriter-kääreluokan konstruktorin parametrina. FileWriter-oliota käytettiin vielä parametrina BufferedWriter-kääreluokassa, jon-

ka jälkeen BufferedWriter-kääreluokan write-metodilla kirjoitettiin rivit tiedostoon. Tiedoston kirjoittaminen esitetään koodiesimerkissä 16.

```
protected static void writeData(String dataString, Context context) throws IOException{
        File dataFile = new File(context.getFilesDir(), “DataStorageFile”);
        if ( !dataFile.exists() ) dataFile.createNewFile();
        BufferedWriter writer = new BufferedWriter(new FileWriter(dataFile, false));
        writer.write(dataString);
        writer.close();
    }
```

Koodiesimerkki 16. Tiedostoon kirjoittaminen

Tiedoston lukemisessa käytettiin kirjoittamisen tapaan Context- ja File-luokkia, mutta lukemisessa käytettiin FileReader- ja BufferedReader-luokkaa. Lisäksi paluuarvon käsittelyssä käytettiin StringBuilder-luokkaa. Tiedoston lukeminen esitetään koodiesimerkissä 17.

```
protected static String readData(Context context) throws IOException{
    String dataString = "";
    File dataFile = new File(context.getFilesDir(), “DataStorageFile”);
    BufferedReader bufferedReader = new BufferedReader(new FileReader(dataFile));
    StringBuilder stringBuilder = new StringBuilder();
    String line;
    while ((line = bufferedReader.readLine()) != null){
        stringBuilder.append(line);
    }
    bufferedReader.close();
    return stringBuilder.toString();
}
```

Koodiesimerkki 17. Tiedoston lukeminen

5.2.11 Arduino-ohjelma

Arduino-ohjelmasta pyrittiin tekemään mahdollisimman yksinkertainen jättämällä datan käsittely Android-sovelluksen tehtäväksi. Arduino ottaa anturilta tulevan tiedon vastaan ja asettaa sen eteen anturin tunnisteen sekä loppuun merkin

”/n”. Niiden avulla yksittäinen mittaustieto saadaan poimittua Android-sovelluksessa. Anturin tunnistetiedon avulla voitaisiin käsitellä useampien antureiden tietoja. Koodiesimerkissä 18 on Arduino-koodi esitettynä kokonaisuudessaan, joten koodin määrä datan vastaanottamiseksi etäisyysanturilta ja lähettämiseksi Android-sovellukselle on melko vähäinen.

Aluksi koodissa otetaan käyttöön SoftwareSerial-kirjasto, jonka avulla voidaan suorittaa sarjaliikennettä Arduinon digitaalisten liittimien välillä. Kirjasto ladataan ohjelman mukana Arduino-piirilevylle, joten kaikki mukaan liitettävät kirjastot vievät muistia piirilevyltä. Seuraavaksi luotiin muuttujat tarvittaville liittimille, joiden arvoksi asetettiin käytettävien liittimien numerot sekä luotiin apumuuttujat mittausdataa varten. Arduino UNO -piirilevyllä led-valo on yhdistetty kiinteästi liittimeen 13.

Arduino-ohjelmissa täytyy olla aina setup- ja loop-metodit. Setup-metodia kutsutaan kerran käynnistyksessä ja se on sopiva paikka tehdä tarvittavat alustustoimet, kuten avata liikenne sarjaportteihin. Koodiesimerkissä 18 yhteys avataan ensin nopeudella 115200 bit/s, mutta seuraavilla riveillä nopeus pudotetaan nopeudelle 9600 bit/s ja samalla kirjoitettiin muutamia rivejä lähettäväksi. Toimintaa kokeiltiin myös erilaisilla asetuksilla, mutta yhteys ei toiminut kunnolla. Toiminta voi olla myös ominaista Bluetooth Mate Silver -lähettimelle.

Loop-metodissa tehtiin datan vastaanottaminen anturilta ja lähettäminen Android-sovellukselle, koska loop-metodia silmukoidaan niin kauan kuin Arduino-piirilevylle syötetään virtaa. Loop-metodin sisällä käytettiin digitalWrite-metodia ledin sytyttämiseen ja sammuttamiseen. HIGH-vakiolla ledille syötetään jännite 5V tai 3,3V riippuen piirilevylle syötettävästä jännitteestä ja LOW-vakiolla jännite on nolla. AnalogRead-metodilla luettiin analogisesta liittimestä etäisyysanturin lähettämää dataa. Delay-metodin avulla pysäytettiin silmukointi 200 ms:n ajaksi, jolloin ledin vilkkumisen ehti havaita Arduinon piirilevyllä. Ledin käyttäminen ei ole lähetyksen vuoksi tarpeellista, mutta sen avulla pystyi havaitsemaan ohjelman latautumisen piirilevylle sekä ohjelman toiminnan. SoftwareSerial-luokan

print-metodilla kirjoitettiin dataa sarjaporttiin, josta se lähetetään edelleen Bluetooth-lähettimen kautta Android-sovellukselle.

```
/*
 * Datan lähettäminen Arduino-piirilevyn avulla
*/
#include <SoftwareSerial.h>

int bluetoothSendPinDigital = 2;
int bluetoothReceivePinDigital = 3;
int val=0;

int sensorPinAnalog = 0;
int sensorVal = 0;

int led = 13;

SoftwareSerial bluetooth(bluetoothSendPinDigital, bluetoothReceivePinDigital);

void setup()
{
  bluetooth.begin(115200);
  bluetooth.print("$$$");
  delay(100);
  bluetooth.println("U,9600,N");
  bluetooth.begin(9600);
}

void loop()
{
  digitalWrite(led, HIGH);
  sensorVal = analogRead(sensorPinAnalog);
  bluetooth.print("L" + String(sensorVal) + "\n");
  delay(200);
  digitalWrite(led, LOW);
}
```

Koodiesimerkki 18. Mittaustietojen lähettäminen Arduinolla

6 YHTEENVETO JA KEHITYSIDEOITA

Opinnäytetyön lopputuloksena valmistui järjestelmä, jolla voi mitata muuttuvaa etäisyyttä monissa eri käyttötarkoituksissa ja -paikoissa. Järjestelmän avulla voi myös parantaa turvallisuutta, jos mittauspaikka on sellaisessa kohteessa, että sen lähelle on riskialtista mennä. Arduinon, Bluetooth-lähettimen ja etäisyysanturin muodostama paketti toimii itsenäisesti akun tai muuntajan avulla, jolloin se lähettää mittausdataa jatkuvasti vastaanotettavaksi Android-sovelluksessa. Android-sovellusta voi käyttää tietokoneella virtuaalikoneessa tai asentamalla se Android-laitteeseen, mutta laitteissa täytyy olla myös Bluetooth-ominaisuus tuettuna.

Toteutuksen aikana sai kokemusta Android-sovelluksen toteutuksesta sekä ohjelmistoista, joita toteutuksen aikana tarvitaan. Java-ohjelmointitaidot helpottivat toteutuksen aloitusta ja melko nopeasti Android-sovelluksen tekemiseen liittyvät asiat pystyi omaksumaan. Android Studio helpotti työtä monessa asiassa ja erityisesti näyttöjen toteutuksessa. Android Studiossa näki heti, kuinka näytön muodostavien XML-tiedoston muokkaaminen vaikuttaa itse näytön ulkoasuun, mutta sillä pystyi myös suoraan lisäämään komponentteja näytölle vain raahaamalla ja pudottamalla. Kokemuksen karttuessa XML-tiedostojen muokkaaminen suoraan tuntui kuitenkin nopeammalta tavalta käsitellä näyttöjä. Android Studio sisälsi valmiiksi paljon muutakin kehityksessä tarvittavaa heti asennuksen jäljiltä, eikä ympäristön räätälöintiä tarvitse paljoa tehdä alkuun pääsemiseksi.

Ohjelmistot toimivat kehityskoneena toimineessa muutaman vuoden vanhassa kannettavassa tietokoneessa riittävän sujuvasti, vaikka se oli nykyisiin koneisiin verrattuna varustettu melko vähäisillä resursseilla. Android Studion toimintoja joutui toisinaan odottamaan, mutta ei kuitenkaan niin kauan, että tämä olisi häirinnyt käyttöä. Kehitettävä Android-sovellus avautui virtuaalikoneeseen nopeasti kokeiltavaksi ja toiminta oli samanlaista myös Android-puhelimessa, joten tämän suhteen ei tullut yllätyksiä. Muutenkin kehitysympäristö toimi kehitystyön ajan ilman ongelmia ja aikaa ei siten kulunut ongelmien selvittelyyn. Erikseen

USB-porttiin liitettävä Bluetooth-sovitin osoittautui hyväksi hankinnaksi, koska VirtualBox ei saanut aina otettua Bluetooth-sovitinta itselleen Windows-käyttöjärjestelmältä. Bluetooth oli käytettävissä VirtualBox -virtualisointialustalla, kun Bluetooth-sovittimen otti irti USB-portista ja kytki takaisin virtualisointialustan ollessa käynnissä. Sisäänrakennetun Bluetooth-sovittimen kanssa tämä olisi voinut olla haasteellista.

Arduino-komponenttien avulla voi rakentaa monenlaisia järjestelmiä valojen vilkuttamisesta robotteihin saakka. Arduinon ohjelmointi on helppo oppia ja ohjelmiston lataaminen piirilevylle on nopeaa. Arduino IDE on hyvin pelkistetty ja sisältää ainoastaan tarpeellisimmat toiminnot. Verkkosivuilta löytyy runsaasti materiaalia oppimisen tueksi ja verkkokaupoista voi hankkia piirilevyyn liitettäviä komponentteja lukuisiin käyttötarkoituksiin. Tässä opinnäytetyössä käytetyn etäisyysanturin mittaustulokset vaihtelivat jatkuvasti, vaikka etäisyys ei muuttunut. Vaihteluväli ei kuitenkaan ollut niin suuri, että se olisi haitannut käyttämistä toteutetussa järjestelmässä. Laadukkaammilla antureilla mittaustulokset olisivat varmasti tarkemmat, mutta niiden hinnat ovat moninkertaiset edullisimpiin verrattuna eli sellaisiin kuin tässä työssä käytettiin. Infrapunaetäisyysanturi ei myöskään soveltunut suoraan vedenpinnan tason mittaukseen. Veden pinnalle täytyi laittaa kelluva kappale, jonka avulla etäisyysanturi sai tarvittavan heijastuksen sekä tätä kautta mitattua veden korkeuden.

Bluetooth-lähettimen toiminta oli odotusten mukaista. Mittaustulokset saatiin luettua rakennuksen sisällä noin 10 metrin etäisyydeltä, vaikka lähettimen ja mobiililaitteen välissä oli seiniä. Seinien rakenne vaikuttaa lähettimen kantamaan, mutta niitä ei tässä työssä tutkittu tarkemmin. Mittausetäisyyteen vaikuttaa lisäksi vastaanottavan laitteen ominaisuudet.

Järjestelmän kehitystä voi jatkaa parantamalla näyttöjen ulkoasua ja lisäämällä grafiikkaa havainnollistamaan mittaustuloksia. Eri suureita mittaavien anturei-den lisääminen järjestelmään kasvattaisi valvottavien kohteiden määrää, jolloin esimerkiksi lämpötilan seuranta onnistuisi mobiililaitteen avulla. Edellä mainitut kehitysideat ovat toteutettavissa pienillä työmäärillä, mutta suuremmatkin muu-

tokset ovat mahdollisia. Arduinoon on saatavilla komponentteja, joiden avulla tiedonsiirto onnistuu muillakin tavoilla kuin Bluetooth:n kautta. Tällöin mittaustietojen seuraaminen onnistuu lähes mistä tahansa. Lisäksi järjestelmään on mahdollista lisätä tietojen lähettäminen mobiililaitteelta Arduinolle, jolloin voidaan ohjata laitteiden toimintaa mobiililaitteen avulla eikä vain vastaanottaa tietoa. Arduinon ja Androidin suosio kehittäjien keskuudessa on helppo ymmärtää jo tämänkin työn toteutuksen aikana saadun kokemuksen perusteella.

7 LÄHTEET

Android Developer A 2015. Android Studio Overview. Viitattu 15.4.2015. https://developer.android.com/tools/studio/index.html.

Android Developer B 2015. Application Fundamentals. Viitattu 15.4.2015. http://developer.android.com/guide/components/fundamentals.html.

Android Developer C 2015. Android Debug Bridge. Viitattu 28.4.2015. http://developer.android.com/tools/help/adb.html.

Android Developer D 2015. Android Studio. Viitattu 15.9.2015. https://developer.android.com/sdk/index.html

Android Developer E 2015. Tools help. Viitattu 15.9.2015. https://developer.android.com/tools/help/index.html.

Android Developer F 2015. Intent. Viitattu 15.9.2015. http://developer.android.com/reference/android/content/Intent.html.

Android Developer G 2015. BluetoothAdapter. Viitattu 15.9.2015. http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html

Android Developer H 2015. Manifest. Viitattu 15.9.2015. http://developer.android.com/reference/android/Manifest.permission.html.

Android Developer I 2015. BluetoothDevice. Viitattu 15.9.2015. http://developer.android.com/reference/android/bluetooth/BluetoothDevice.html.

Android Developer J 2015. Storage Options. Viitattu 15.9.2015. http://developer.android.com/guide/topics/data/data-storage.html

Android Developer K 2015. Layouts. Viitattu 22.9.2015. http://developer.android.com/guide/topics/ui/declaring-layout.html

Arduino 2015. Arduino / Genuino UNO. Viitattu 4.5.2015. http://www.arduino.cc/en/Main/ArduinoBoardUno.

Banzi, M. 2011. Arduino perusteista hallintaan. Saksa: Norderstedt.

Bluetooth 2015. Fast Facts. Viitattu 6.9.2015. http://www.bluetooth.com/Pages/Fast-Facts.aspx.

Bluetooth Developer portal 2015. RFCOMM with TS 07.10. Viitattu 9.6.2015. https://developer.bluetooth.org/TechnologyOverview/Pages/RFCOMM.aspx.

Brachman, S. 2014. A Brief History of Google's Android Operating System. Viitattu 15.4.2015. http://www.ipwatchdog.com/2014/11/26/a-brief-history-of-googles-android-operating-system/id=52285.

Creative 2015. History of Arduino. Viitattu 15.4.2015. http://creativityprojects.blogspot.fi/2013/03/history-of-arduino_4195.html.

Digi-Key 2015. Sharp Microelectronics GP2Y0A02YK0F. Viitattu 22.9.2015. http://www.digikey.com/product-detail/en/GP2Y0A02YK0F/425-2062-ND/720167.

Electronic design 2015. What's The Difference Between The RS-232 And RS-485 Serial Interfaces? Viitattu 29.10.2015. http://electronicdesign.com/what-s-difference-between/what-s-difference-between-rs-232-and-rs-485-serial-interfaces

Open Handset Alliance 2015. Android. Viitattu 16.4.2015. http://www.openhandsetalliance.com/android_overview.html.

Oracle 2015. Introduction to Virtualization. Viitattu 14.9.2015. http://docs.oracle.com/cd/E11081_01/doc/doc.21/e10898/intro.htm.

RobotItaly 2015. Sharp Distance Sensor 2Y0A02 (20-150cm). Viitattu 14.6.2015. http://www.robot-italy.com/en/3522-sharp-distance-sensor-2y0a02-20-150cm.html.

Sharp 2005. GP2Y0A21YK Optoelectronic Device. Viitattu 20.9.2015. http://www.sharpsma.com/webfm_send/1208

SparkFun 2015. SparkFun Bluetooth Mate Silver. Viitattu 4.4.2015. https://www.sparkfun.com/products/12576.

Techotopia 2015. Creating an Android Virtual Device (AVD) in Android Studio. Viitattu 24.4.2015. http://www.techotopia.com/index.php/Creating_an_Android-_Virtual_Device_%28AVD%29_in_Android_Studio

VirtualBox 2015. Manual. Viitattu 13.4.2015 https://www.virtualbox.org/manual/ch01.html.