HAAGA-HELIA
ammattikorkeakoulu

Arduino-pohjainen laite liikkeen ja lämpötilan monitoroin-tiin

Aleksi Karppila

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2014

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

10.11.2014

Tekijä(t)
Aleksi Karppila

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Arduino-pohjainen laite liikkeen ja lämpötilan monitorointiin

Sivu- ja liitesivumäärä
34 + 12

Opinnäytetyön otsikko englanniksi
Arduino based device for monitoring motion and temperature

Tässä opinnäytetyössä valmistetaan open source projektina monitorointilaite, jonka toteuttamiseen käytetään Arduino kehitysalustaa ja siihen liitettäviä antureita. Laitteen valmistuksessa pyritään yksinkertaisuuteen ja helposti toteutettavaan ratkaisuun. Laitteen halutaan tarkkailevan lämpötilaa ja liikettä sekä lähettävän varoitusviestin käyttäjälle.

Opinnäytetyö koostuu johdannosta, tietoperustasta, empiirisestä osasta ja pohdinnasta. Johdannossa esitellään opinnäytetyön rakenne ja käsiteltävät aiheet lyhyesti. Tietoperustassa käydään läpi Arduino laitteiden toimintaan liittyviä asioita, kuten Arduino kehitysalustoissa käytettävien pinnien toiminta ja Arduino ohjelmointikielen perusteet. Empiirisessä osassa kuvataan monitorointilaitteen valmistuksen vaiheet, joita ovat monitorointilaitteen suunnittelu ja monitorointilaitteen valmistaminen. Monitorointilaitteen suunnittelussa pyritään löytämään parhaat menetelmät ja laitteessa käytettävät osat, jotta laitteelle asetetut tavoitteet saavutetaan. Monitorointilaite valmistetaan useassa osassa, jonka jälkeen luodut osat yhdistetään lopulliseksi monitorointilaitteeksi.

Opinnäytetyön tuloksena valmistuu monitorointilaite, joka tarkkailee lämpötilaa ja liikettä sekä lähettää varoitusviestin käyttäjälle sähköpostitse, kun monitorointilaite havaitsee liikettä tai ympäröivän maailman lämpötila nousee annettua raja-arvoa korkeammaksi. Monitorointilaitteen käyttäjät voivat määritellä ohjelmakoodiin muun muassa varoitusviestin sisällön ja tarkkailtavan lämpötilan raja-arvon. Opinnäytetyön päätteeksi monitorointilaitteen lähdekoodit ja selvitys laitteen toiminnasta julkaistaan GitHub palvelussa. Monitorointilaitteen julkaiseminen mahdollistaa muille aiheesta kiinnostuneille laitteen jatkokehityksen.

Valmistunut monitorointilaite saavutti sille asetetut tavoitteet. Suunnitellussa on onnistuttu valitsemaan oikeat menetelmät ja osat, jotta laitteen valmistaminen on yksinkertaista. Laitteen valmistuksessa on myös onnistuttu pitämään laitteen kokoaminen ja ohjelmointi helppona.

Asiasanat
Arduino, mikro-ohjaimet, monitorointi, avoin lähdekoodi

HAAGA-HELIA
University of Applied Sciences

Abstract

10.11.2014

Author(s)
Aleksi Karppila

Degree programme
Degree Programme in Business Information Technology

Report/thesis title
Arduino based device for monitoring motion and temperature

Number of pages and appendix pages
34 + 12

The main goal of this thesis is to create a monitoring device as open source project, using Arduino microcontroller board and sensors attached to it. The design of the device is meant to be kept simple and easy to produce. The device is meant to monitor ambient temperature and motion. Also a warning message is wanted to be sent to the user.

The thesis consists of an introduction, theory section, empirical section and discussion. The introduction describes the design of the thesis and briefly explains the topics of the work. The theory section discusses basics and features of Arduino, such as Arduino microcontroller boards, pins used in Arduino microcontroller boards and Arduino programming language. The empirical section describes the steps of creation process of the monitoring device. The steps of creation process include planning and producing of the monitoring device. Planning of the device aims to find the best solutions for producing the actual device. The monitoring device is produced in several parts, which are combined in the end to the final monitoring device.

As a result of the thesis, a monitoring device is made, that sends an email warning message to the user, once it detects motion or ambient temperature rises above the set threshold value. Users of the monitoring device can, for example define the message body of the warning message or set out the threshold value for the monitored temperature. At the end of the thesis, source codes and explanation of the monitoring device will be published in GitHub. Publication of the monitoring device, will make further development of the device possible.

The completed monitoring device accomplishes the goals that were set for it. Planning of the device was successful, as right solutions for creating a simple and easy monitoring device were found. Also combining and coding of the device are managed to keep simple.

Keywords
Arduino, microcontrollers, monitoring, open source

Sisällys

1 Johdanto

Tässä opinnäytetyössä valmistetaan laite, jonka avulla voidaan tarkkailla lämpötilaa ja liikettä. Tarkkailun lisäksi laite lähettää varoitusviestin, kun se havaitsee liikettä tai lämpötila muuttuu annettujen raja-arvojen ulkopuolelle. Opinnäytetyössä valmistuva monitorointilaite toteutetaan Arduino kehitysalustaa ja siihen liitettäviä antureita käyttäen. Arduino perustuu avoimeen laitteistoon ja se on suunniteltu yksinkertaistamaan mikrokontrollereja hyödyntävien laitteiden valmistusta. Lisäksi varoitusviestin lähettämiseen käytetään kolmannen osapuolen ohjelmointikoodin virtualisointipalvelua nimeltä Temboo.

Opinnäytetyössä valmistettavan laitteen kehityksessä on pyritty yksinkertaisuuteen ja helppoon toteuttamiseen, jotta kohderyhmä pysyisi mahdollisimman laajana. Opinnäytetyön tavoitteena on mahdollistaa monitorointilaitteen valmistaminen myös niille, joilla ei ole paljoa aiempaa tietoteknistä kokemusta. Monitorointilaitteen kehitys toteutetaan open source projektina, joka mahdollistaa monitorointilaitteen jatkokehityksen ja muokkaamisen myös kokeneemmille käyttäjille.

Opinnäytetyö koostuu tietoperustan esittelystä, laitteen valmistuksen vaiheista ja varsinaisesta työssä valmistuvasta laitteesta. Opinnäytetyön päättää pohdinta, jossa käsitellään muun muassa opinnäytetyön ja monitorointilaitteen onnistuneisuutta, sekä esitetään laitteelle mahdollisia jatkokehitysehdotuksia.

Opinnäytetyön teoriaosuus aloitetaan esittelemällä Arduino ja sen toiminta. Lisäksi tietoperustassa käsitellään Arduino kehitysalustat ja niiden toiminnan kannalta oleellisten asioiden, kuten pinnien ja liitäntöjen toiminta. Myös laitteessa käytettävien menetelmien ja osien toimintaperiaatteet käydään läpi. Laitteiden ohjelmointi on myös oleellinen osa Arduino laitteiden valmistusta, joten teoriaosuudessa käsitellään Arduino ohjelmoinnin perusteet sekä laitteen valmistuksessa tarvittavat ohjelmointi komennot.

Teoriaosuuden jälkeen käydään läpi varsinaisen monitorointilaitteen valmistusprosessi. Laitteen valmistusprosessin esittely aloitetaan laitteen suunnitteluvaiheen läpi käymisellä. Suunnitteluvaiheen esittely sisältää laitteelle asetetut tavoitteet ja mahdolliset käyttötarkoitukset. Edellä mainittujen lisäksi suunnitteluvaiheessa kerrotaan valituista menetelmistä ja osista, sekä perusteellaan niiden valinta. Varsinaisen monitorointilaitteen valmistus kuvataan vaiheittain ja monitorointilaitteen valmistu on jaettu useaan osaan. Laitteen valmistus aloitetaan kuvaamalla lämpötilaa tarkkailevan laitteen valmistus, jonka jälkeen esitellään liikettä tarkkailevan laitteen valmistamiseen vaaditut toimenpiteet. Molemmat laitteet yhdistetään ja laitetta muokataan, jotta se täyttää sille asetetut vaatimukset. Laitteen varoi-

tusviestin tekemiseen tarvittavat toimenpiteet esitellään, jonka jälkeen varoitusviesti laitetaan toimimaan aiemmassa vaiheessa valmistuneessa monitorointilaitteessa.

Laitteen testaukseen liittyvät toimenpiteet ja tulokset esitellään laitteen valmistuksen jälkeen. Valmistusprosessin kuvauksen päättää yhteenveto, jossa kootaan yhteen laitteen valmistusprosessi.

2 Arduinon toiminta ja laitteet

Arduino on alusta, joka mahdollistaa ympäristöä tarkkailevien ja hallitsevien tietoteknisten laitteiden valmistamisen. Arduino sai alkunsa vuonna 2005 ja sen perustajia olivat David Cuartielles, Massimo Banzi ja Dave Mellis. Arduinon idea syntyi kun Massimo Banzi halusi kehittää oppilailleen helpomman keinon opiskella teknologiaa. Arduino perustuu avoimeen laitteistoon ja se sisältää mikrokontrolleri-elektroniikka-alustan sekä ohjelmointiympäristön. Arduinon laitteisto on suunniteltu 8-bittisen Atmel AVR mikrokontrollerin tai 32-bittisen Atmel ARM -mikrokontrollerin ympärille. (Arduino a; Lahart 2009; Medea 2013.)

Arduinoa voidaan käyttää interaktiivisten laitteiden luomiseen, joilla voidaan hallinnoida esimerkiksi erilaisia moottoreita, valoja sekä useita muita näkyviä toimintoja. Kehitetyt Arduino laitteet voivat toimia omatoimisesti, mutta ne voidaan myös laittaa kommunikoimaan tietokoneella toimivien ohjelmien kanssa. (Arduino a.) Kommunikointi ulkoisen tietokoneen kanssa mahdollistaa todella monien käytännöllisten ja monimutkaisten, mutta helposti toteutettavien laitteiden luomisen.

Arduinon mukaan sillä on useita ominaisuuksia, jotka erottavat sen muista samankaltaisista mikrokontrollereihin perustuvista alustoista. Alle on listattu Arduinon (Arduino a) antamia esimerkkejä eduista:

- Arduino on suhteellisen halpa verrattuna muihin mikrokontrollereihin perustuviin alustoihin.
- Arduinon ohjelmointiympäristö soveltuu hyvin aloittelijoille, mutta se on silti tarpeeksi joustava kokeneemmille ohjelmoijille.
- Toisin kuin useimmat muut mikrokontrollerijärjestelmät, Arduino toimii muillakin käyttöjärjestelmillä kuin Windowsilla. Windowsin lisäksi Arduino toimii Macintosh OSX:llä ja Linux käyttöjärjestelmillä.
- Arduinon ohjelmisto kuuluu vapaaseen lähdekoodiin, joten sen jatkokehittäminen on mahdollista asiasta kiinnostuneille. Arduinon ohjelmointikieltä voidaan myös laajentaa käyttämällä C++ ohjelmakirjastoja.
- Arduino perustuu avoimeen laitteistoon, joten laitteen suunnitelmat on julkaistu Creative Commons lisenssillä. Tämä mahdollistaa kehittäjille omien laiteversioiden luomisen. Lisäksi käyttäjät voivat säästää rahaa ja oppia ymmärtämään laitteen toiminnan paremmin kokoamalla laitteen itse.

Arduino laitteiden toiminta perustuu virtapiireihin. Virtapiireistä puhuttaessa jännitteellä tarkoitetaan eroa kahden kappaleen varauksessa, jonka yksikkö on voltti. Kun kaksi eri tavalla varautunutta pistettä yhdistetään, aiheuttaa jännite virran. Virtapiireissä virtaus tapahtuu suuremman energian omaavasta pisteestä pienemmän energian omaavaan pisteeseen. Virtapiirissä on oltava reitti virtalähteestä pienimmän energian omaavaan pisteeseen, jotta virtapiiri voi toimia. Pienimmän energian omaava piste on miinusnapa. (Fitzgerald & Shiloh 2013, 21-22; Karvinen & Karvinen 2009, 31.)

2.1 Arduino laitteet ja liitännät

Arduino projekteissa käytettävät osat voidaan jakaa karkeasti kolmeen alueeseen, jotka ovat anturit, toimilaitteet ja kehitysalustat. Anturit tarkkailevat ympäröivää maailmaa ja muuttavat havaitsemansa muutoksen energiaksi. Antureita ovat esimerkiksi painikkeet sekä lämpötila-anturit ja liikeanturit. Moottorit ja valot ovat taas esimerkkejä toimilaitteista, jotka muuntavat sähköisen energian fyysiseksi ja näkyväksi energiaksi. Kehitysalustat toimivat rajapintana anturien ja toimilaitteiden välillä niin kuin käyttäjä on ne ohjelmoinut. (Fitzgerald & Shiloh 2013, 5.) Lisäksi Arduino kehitysalustoihin voidaan liittää erinäisiä lisälaitteita toimintojen laajentamiseen. Myös muut sähkökomponentit kuten johdot ovat tärkeä osa Arduino laitteita.

2.1.1 Arduino kehitysalustat

Arduino kehitysalustat on piirilevyihin valmistettuja pienikokoisia tietokoneita. Arduino kehitysalustat koostuvat Atmelin valmistamista mikrokontrollereista, joiden lisäksi kehitysalustasta riippuen niissä on erinäisiä liitäntöjä, pinnejä ja ominaisuuksia. Yleisimmät liitännät kehitysalustoissa ovat USB ja sähköliitäntä, mutta joissain kehitysalustoissa on esimerkiksi myös Ethernet ja WiFi ominaisuudet. Arduino kehitysalustat ovat Arduino laitteiden ydin, sillä ne ohjaavat laitteiden toimintaa. Kehitysalustojen analogisiin tai digitaalisiin pinneihin voidaan liittää antureita tai toimilaitteita, kuten liikeantureita ja LED-valoja. Kehitysalustat ohjaavat laitteen toimintaa anturien, toimilaitteiden ja mahdollisesti ulkoisen tietokoneen välillä. (Arduino p; Karvinen & Karvinen 2009, 33.)

Arduino kehitysalustoja on useita erilaisia ja ne voidaan jakaa virallisiin Arduino kehitysalustoihin, Aduino kehitysalustojen kopioihin ja Arduino yhteensopiviin kehitysalustoihin (Banzi 2013). Arduinon virallisilla ja muiden valmistamilla kehitysalustoilla on eroja, jotka on hyvä tietää.

Viralliset Arduino kehitysalustat löytyvät Arduinon virallisilta sivuilta. Viralliset kehitysalustat on valmistettu Arduinon kanssa yhteistyötä tekevien valmistajien toimesta. Viralliset

Arduino kehitysalustat toimivat suoraan Arduino ohjelmointiympäristössä, ne noudattavat Arduinon standardisoitua asettelua ja niiden dokumentointi löytyy Arduinon verkkosivulta. Lisäksi ne ovat oikein lisensoitu ja niissä on lupa käyttää Arduinon virallista logoa. Vastineeksi viralliset valmistajat maksavat Arduinolle pientä tekijänoikeusmaksua. (Banzi 2013.)

Arduino kehitysalustojen kopiot taas ovat identtisiä tai lähes identtisiä virallisten Arduino kehitysalustojen kanssa. Ne toimivat kuitenkin eri tuotemerkin alla, eikä niistä makseta tekijänoikeusmaksua Arduinolle. Arduino yhteensopivat kehitysalustat ovat jossain määrin yhteensopivia Arduinon kanssa, mutta niiden toiminta Arduinon kanssa saattaa olla hyvin vähäistä. (Banzi 2013.)

Arduinon virallisia kehitysalustojakin on useita erilaisia. Virallisten kehitysalustojen erot tulevat esiin kolmessa eri seikassa. Kehitysalustoissa on eroja prosessointi kapasiteetissa, eli mikrokontrollerin muistissa, kellotaajuudessa ja kaistanleveydessä. Prosessointilaitteisto riippuu usein siitä, mitä mikrokontrollerisirua siinä on käytetty. Toinen ero tulee kehitysalustojen ulkomuodossa, sillä Arduino kehitysalustoja on useita erikokoisia ja erimuotoisia. Kolmas eroja tekevä asia on kehitysalustoissa valmiina olevat toiminnot. Toiminnot käsittävät kaiken muun paitsi laskentatehoon vaikuttavat asiat, kuten mikrokontrollerissa olevat pinnit, liitännät, sisäänrakennetut LED-valot ja painikkeet. Esimerkiksi joissain kehitysalustoissa voi olla liitäntöjä, kuten Ethernet tai erillinen virtaliitin, kun taas toisissa saattaa olla pelkkä USB-liitin. (Castle 2013.)

2.1.2 Osat ja komponentit

Arduino kehitysalustoihin liitetään erilaisia osia ja komponentteja, joiden avulla voidaan tehdä laitteita, jotka suorittava erinäisiä toimintoja. Tärkeimpiä osia ovat anturit, toimilaitteet, muut sähkökomponentit ja toimintojen laajentamiseen käytettävät lisälaitteet. Erilaisia osia ja komponentteja on lukematon määrä, joten tässä osiossa käsitellään osia ja komponentteja vain yleisesti. Lisäksi käydään läpi tässä Arduino projektissa käytettävät osat ja komponentit.

Anturit ovat laitteita, jotka tarkkailevat jotain määriteltyä ympäröivän maailman toimintaa ja toimivat havaitsemansa muutoksen perusteella. Tarkkailtavia toimintoja voivat olla muun muassa muutokset valon määrässä, lämpötilassa, kosteudessa tai liikkeessä. Eniten Arduinossa käytetään antureita, jotka tarkkailevat fyysistä painetta, valoa, lämpötilaa, kallistusta, liikettä tai infrapunasignaaleja. (ladyada.net 2012; Wigmore 2012.)

Arduinossa anturit voivat olla digitaalisia tai analogisia. Analogiset anturit voivat lähettää signaaleja, joiden voimakkuus voi olla mitä tahansa määritellyltä väliltä, perustuen siihen mitä anturi havaitsee. Digitaaliset anturit puolestaan voivat lähettää signaaleja vain sille määritellyillä voimakkuuksilla, joita useimmissa tapauksissa on kaksi. Toisin sanoen Arduinon digitaaliset anturit useimmissa tapauksissa voivat lähettää vain signaaleja, joiden voimakkuus on nolla tai viisi volttia. Arduinon tapauksessa analogiset anturit taas voivat lähettää signaaleja, joiden voimakkuus on mitä tahansa nollan ja viiden voltin väliltä. (Sparkfun.)

Toimilaitteet ovat laitteita jotka aiheuttavat jonkin näkyvän tai fyysisen toiminnon, kun ne vastaanottavat signaalin (Fitzgerald & Shiloh 2013, 21). Arduinossa käytettäviä toimilaitteita ovat esimerkiksi LED-valot ja moottorit. Toimilaitteita käytetään digitaalisissa pinneissä, joten yksinkertaistettuna ne ovat joko päällä tai pois päältä.

Arduinoon on liitettävissä paljon muitakin sähkökomponentteja, kuten johtoja, vastuksia, diodeja ja koekytkentälevyjä. Johdot mahdollistavat virran käytännössä esteettömän kulun ja niitä käytetään komponenttien yhdistämisessä toisiinsa. Esimerkiksi anturit ja toimilaitteet kytketään itse kehitysalustaan johtojen avulla. Koekytkentälevyjä käytetään helpottamaan laitteiden kytkentää toisiinsa ja niitä käytetäänkin usein virtapiirien toimivuuden testaamiseen. (Arduino e.)

Arduino lisälaitteet ovat laitteita, joiden avulla Arduino kehitysalustoihin voidaan lisätä erinäisiä toimintoja. Lisälaitteet liitetään usein miten Arduino kehitysalustan päälle. (Arduino f.) Lisälaitteiden avulla voidaan lisätä esimerkiksi Ethernet tai WiFi yhteensopivuus kehitysalustoihin joissa niitä ei ole valmiina.

2.1.3 Projektissa käytettävät osat

PIR on lyhenne sanoista Pyroelectric Infrared ja PIR tekniikkaa käytetään PIR antureissa liikkeen havaitsemiseen. PIR antureissa liikkeentunnistin on jaettu kahteen puoliskoon, jotka on valmistettu infrapunaa havaitsevasta materiaalista. Anturin ollessa toimettomana, molemmat anturin puoliskot havaitsevat saman määrän infrapunaa ympäröivästä maailmasta. Kun anturin havaitsemisalueella tapahtuu liikettä, toinen puoliskoista havaitsee liikkeen ensin, jolloin toinen puolisko havaitsee enemmän infrapunaa kuin toinen. PIR anturi siis lähettää positiivisen signaalin kehitysalustalle, kun PIR anturin puoliskot havaitsevat toisistaan eroavan määrän infrapunaa. Negatiivisen signaalin anturi taas lähettää silloin, kun molemmat sensorin puoliskot havaitsevat saman määrän infrapunaa. (Adafruit 2014.)

Lämpötila-antureja voidaan käyttää ympäristön lämpötilan seuraamiseen. Lämpötila-anturit voivat toimia monella eri tavalla ja lämpötilan tarkkailuun sekä muuntamiseen käytettävissä tekniikoissa on eroja. Tässä projektissa käytetään analogista lämpötila-anturia, joka lähettää kehitysalustoille jännitteen ympäröivän lämpötilan perusteella. Lämpötila-anturi siis lähettää vaihtelevan määrän voltteja tunnetulla suhteella, sen mukaan mikä ympäröivä lämpötila on. Yksinkertaistettuna anturi lähettää suuremman jännitteen kun lämpötila on korkea ja pienemmän määrän kun lämpötila on matala. Koska anturi lähettää jännitteen tunnetulla lämpötilasta riippuvalla suhteella, voidaan jännite muuttaa helposti ja tarkasti lämpötilalukemaksi. (Adafruit 2012.)

Edellä jo todettiin, että Arduino kehitysalustoja on useanlaisia. Tähän projektiin on valittu käytettäväksi Arduino Yún kehitysalusta. Arduino Yúnissa on Atmega32u4 mikrokontrollerin lisäksi erillinen Atheros AR9331 prosessori. Atheros prosessori tukee Linuxin OpenWrt:hen perustuvaa jakelua OpenWrt-Yun. Kehitysalustassa on tuki Ethernetille ja WiFille. Lisäksi siitä löytyy muun muassa USB-A portti, paikka micro-SD kortille, analogisia ja digitaalisia pinnejä yhteensä kaksikymmentä kappaletta sekä micro USB-liitäntä. Arduino Yúnin suurin ero muihin Arduino kehitysalustoihin on se, että se voi kommunikoida edellä mainitun sisäänrakennetun Linux jakelun kanssa. Atmega32u4 ja Atheros AR9331 on sillattava, jotta ne voivat kommunikoida keskenään. Siltaus voidaan tehdä Bridge ohjelmakirjaston avulla. Arduino Yúnille voidaan antaa virtaa monella eri tavalla ja se voi kommunikoida muiden tietokoneiden kanssa usealla eri tavalla, mutta tässä projektissa käytetään micro USB-liitäntää molempien toteuttamiseen. (Arduino g.)

2.2 Liitännät, pinnit ja painikkeet

Arduino kehitysalustoista löytyy erilaisia liitäntöjä, pinnejä ja painikkeita. Kyseisten ominaisuuksien määrä riippuu käytettävästä Arduino kehitysalustasta. Tässä kappaleessa käsitellään yleisimmät Arduino kehitysalustoista löytyvät pinnit, painikkeet ja liitännät. Oleellisimpia pinnejä useimmissa tapauksissa ovat analogiset ja digitaaliset pinnit sekä virta ja maadoitus pinnit. Muita pinnejä ovat esimerkiksi reset, ioref ja aref pinnit.

2.2.1 Analogiset ja digitaaliset pinnit

Analogisiin ja digitaalisiin pinneihin kiinnitetään antureita ja toimilaitteita. Digitaalisia pinnejä on kahdenlaisia, tavallisia digitaalisia pinnejä ja PWM merkittyjä digitaalisia pinnejä. (Society of Robots.)

Kaikki digitaaliset pinnit voivat toimia tulo- ja lähtöliitäntänä, eli ne voivat lähettää sekä vastaanottaa signaaleja. Digitaaliset pinnit toimivat kehitysalustan asettamalla jännitetasolla, joka Arduinon tapauksessa on usein miten 0 – 5 volttia. Tässä tapauksessa digitaaliset pinnit voivat siis lähettää tai vastaanottaa viiden tai nollan voltin jännitteen. Tästä syystä digitaalisia pinnejä käytetään yleensä lähtöliitäntänä esimerkiksi moottoreille tai LED-valoille, jolloin ne ovat päällä, kun niille lähetetään viisi volttia ja ne ovat pois päältä, kun niille lähetetään nolla volttia. (Society of Robots.)

Digitaaliseen pinniin liitetyn toimilaitteen toimintaa voidaan kuitenkin muuttaa esimerkiksi siten, että digitaaliseen pinniin liitetty moottori pyörii puolella nopeudella verrattuna siihen mitä se pyörisi, jos se vastaanottaisi viisi volttia. Tähän tarkoitukseen tulee käyttää PWM merkittyjä digitaalisia pinnejä. PWM digitaalisten pinnien avulla vastaanottavan toimilaitteen toimintaa voidaan muuttaa, lähettämällä sille nopeaan määriteltyyn tahtiin peräkkäin viisi ja nolla volttia. PMW:n avulla saadaan toimilaite olemaan osan ajasta päällä ja osan ajasta pois päältä, toimien näin ollen vajaalla teholla. (Arduino b; Society of Robots.)

Toisin kuin digitaaliset pinnit, analogiset pinnit pystyvät lukemaan signaaleja joiden voimakkuus on nollan ja viiden voltin välillä. Analogisia pinnejä käytetään useimmiten analogisten anturien kanssa. Analogiset pinnit ottavat vastaan analogisia signaaleja nollan ja viiden voltin väliltä, jotka analoginen pinni muuntaa digitaaliseen muotoon, jota mikrokontrolleri pystyy lukea. Analogisia pinnejä voidaan käyttää myös digitaalisina pinneinä. (Society of Robots.)

2.2.2 Muut pinnit ja painikkeet

Arduino kehitysalustoissa on erilaisia virran siirtoon tarkoitettuja pinnejä. Yleisimmät näistä ovat 5V, 3.3V ja Vin pinnit. Useimmissa Arduino kehitysalustoissa toimintajännitteenä on viisi volttia, mutta joissain tapauksissa ne toimivat 3.3 voltin jännitteellä. 5V pinnillä voidaan siis antaa virtaa esimerkiksi antureille tai toimilaitteelle, jolloin käytetään viiden voltin jännitettä. 3.3V pinnillä voidaan taas luoda 3.3 voltin jännite toimilaitteelle tai anturille, vaikka kehitysalustan toimintajännite olisi viisi volttia. Vin pinniä käytetään silloin, kun Arduino kehitysalustalle halutaan antaa virtaa ulkoisesta lähteestä, kuten patterista tai pistorasiasta. Maadoitus pinnit yksinkertaisesti saattavat virtapiirin päätökseen. (Arduino c; Fitzgerald & Shiloh 2013, 22.)

Arduinon aref pinniä käytetään joissain tapauksissa muun muassa analogisiin pinneihin liitettyjen antureiden kanssa. Arduino kehitysalustat käyttävät oletuksena niiden omaa toimintajännitettä vertausarvona, kun ne saavat signaaleja analogisilta pinneiltä. Useim-

missa kehitysalustoissa vertausarvon oletuksena on viisi volttia, mutta joissain kehitysalustoissa se on 3.3 volttia. Joskus antureita halutaan käyttää kuitenkin toisella jännitteellä, esimerkiksi 3.3V pinnin avulla. Tässä tapauksessa 3.3V pinni pitää liittää aref pinniin. Aref pinnin avulla voidaan siis määritellä jännitteen vertausarvo toiseksi, kuin se on oletuksena. Ioref pinniä käytetään Arduinossa usein miten, kun siihen liitetään lisälaitteita. Ioref pinni antaa jännitteen, joka on sama kuin kyseisen kehitysalustan toimintajännite. (Arduino c; Boxall 2013.)

Arduino kehitysalustoissa on myös reset pinni, jonka avulla kehitysalusta voidaan palauttaa oletusarvoihin. Arduino kehitysalustoissa on myös erilliset painikkeet oletusarvojen palauttamiseen. Lisäksi joissain kehitysalustoissa voi olla reset-painikkeet jonkin toiminnon, kuten WiFin oletusarvojen palauttamiseksi. (Arduino c; Arduino g.)

2.3 Temboo

Temboo on ohjelmointikoodien virtualisointipalvelu. Yksinkertaistettuna Temboo suorittaa pilvessä lähtökohtaisesti monimutkaisia ohjelmointiprosesseja käyttäjän puolesta. Temboolla on suuri arkisto ohjelmointikoodeja, joita kutsutaan Choreoiksi. Temboon Choreot tukevat useita ohjelmointikieliä. Choreot lisätään ohjelmakoodiin ja kun ne suoritetaan ne ottavat yhteyttä Temboon sovellusalustaan, jossa varsinainen ja huomattavasti monimutkaisempi ohjelmakoodi suoritetaan. Temboo palvelu on suunniteltu helpottamaan ulkoisten internetpalveluiden, kuten Facebookin, Twitterin ja Gmailin kanssa kommunikoimista sekä työskentelyä. Arduino tekee yhteistyötä Temboon kanssa. (Arduino d; Temboo.)

3 Arduino ohjelmointikieli ja- rakenne

Arduino ohjelmointikieli juontuu Wiring ohjelmointikielestä, joka on myöskin vapaan lähdekoodin ohjelmointikehys mikrokontrollien ohjelmointiin. Arduino ohjelmakoodien kirjoittamiseen käytetään C++ tai C ohjelmointikieliä. (Arduino a.)

Jokaisessa Arduino ohjelmakoodissa on kaksi pääfunktiota. Funktiot ovat osa ohjelmointikoodia ja ne suorittavat määriteltyjä komentoja. Arduinossa pääfunktiot ovat setup() ja loop(). Setup() funktio suoritetaan vain kerran, kun Arduino laite laitetaan päälle, joten sitä käytetään ohjelman asetuksien luomiseen. Loop() funktion toistaminen aloitetaan setup() funktion jälkeen, kunnes Arduino laite sammutetaan. Arduino ohjelmiin liittyvät komennot voidaan jakaa kolmeen osaan, jotka liittyvät rakenteeseen, arvoihin ja funktioihin. (Arduino q; Fitzgerald & Shiloh 2013, 36.)

Arduino ohjelmointi mahdollisuuksien laajentamiseen ja helpottamiseen voidaan käyttää Arduino ohjelmakirjastoja. Ohjelmakirjastot parantavat Arduino ohjelmointikielen toiminnollisuutta esimerkiksi muiden laitteiden kanssa tai parantavat tiedonkäsittelyn mahdollisuuksia. Ohjelmakirjastoja voidaan luoda itse, mutta Arduino ohjelmointiympäristön mukana tulee joukko ohjelmakirjastoja valmiiksi asennettuna. (Arduino o.)

3.1 Arduino ohjelmointiympäristö

Arduino ohjelmointiympäristö ladataan tietokoneelle Arduinon virallisilta verkkosivuilta ja sillä on monia toimintoja. Arduino ohjelmointiympäristö toimii tekstinkäsittelyohjelmana ohjelmakoodin kirjoittamiselle, jonka lisäksi se kommunikoi Arduino laitteiden kanssa ja hoitaa Arduino ohjelmakoodin lataamisen laitteelle. Arduino ohjelmointiympäristöön kuuluu ilmoituskenttä, tekstikonsoli, useita valikoita ja työkalupalkki. (Arduino h.)

Arduino ohjelmakoodeja talletettaessa ne saavat .ino päätteen. Ohjelmointiympäristön ilmoituskenttä antaa palautetta, kun ohjelmakoodeja tallennetaan tai siirretään, jonka lisäksi siinä ilmoitetaan tapahtuneista virheistä. Tekstikonsolista voidaan esimerkiksi lukea virheilmoitukset kokonaisuudessaan. Työkalupalkin painikkeilla voidaan luoda uusia, avata vanhoja ja tallentaa ohjelmakoodeja. Lisäksi painikkeilla voidaan tarkastaa ohjelmakoodi virheiden varalta, ladata ohjelmakoodi laitteelle ja käyttää serial monitoria. Serial monitoria voidaan käyttää, kun halutaan tarkastella kehitysalustan ja ohjelmointiympäristön välistä kommunikointia. Ohjelmointiympäristöön sisältyy myös valikot: File, Edit, Sketch,

Tools ja Help, joiden avulla voidaan muun muassa valita kommunikoimiseen käytettävä portti sekä käytettävä Arduino kehitysalusta. (Arduino h.)

Arduinosta on saatavilla kaksi versiota, jotka ovat kirjoittamishetkellä Arduino 1.0.6 ja Arduino 1.5.8 BETA. Tämän projektin kannalta on huomioitava, että Arduino Yúnille löytyy tuki vain 1.5.8 BETA versiosta, joten Arduino 1.0.6 ohjelmointiympäristöä ei voida käyttää tässä projektissa. (Arduino i.) Kokeilujeni perusteella ohjelmointiympäristöt ovat hyvin samankaltaisia, eikä niiden käytössä ilmene suuria eroja. Lisäksi Arduino 1.5.8 BETA on toiminut moitteettomasti beta vaiheesta huolimatta, eikä ongelmia ole ilmennyt.

3.2 Työssä käytettävät komennot

Arduino ohjelmissa käytettävät komennot on jaoteltavissa kolmeen osaan, jotka liittyvät rakenteeseen, arvoihin ja funktioihin. Komentoja on erittäin paljon, joten tässä opinnäytetyössä käsitellään vain komennot, joita laitteen valmistuksessa on käytetty. Liitteeseen yksi on koottu ja selitetty Arduinon verkkosivujen oppaan (Arduino q) perusteella lähes kaikki monitorointilaitteen valmistamiseen tarvittavat ohjelmointi komennot. Osa käytetyistä komennoista on myös selitetty alla olevissa kappaleissa.

Oleellisimmat rakenteeseen liittyvät komennot ovat setup() ja loop(), joita kutsutaan myös Arduino ohjelmointikielen pääfunktioiksi. Pääfunktioiden toiminta käsiteltiin lyhyesti jo edellä. Setup() funktio suoritetaan, kun Arduino ohjelmakoodi alkaa ja sitä käytetään asetuksien luomiseen. Setup() funktion suoritus tapahtuu vain kerran, kun Arduino laite laitetaan päälle. Loop() funktion suorittaminen alkaa setup() funktion jälkeen ja sitä toistetaan jatkuvasti. Loop() funktio on Arduino ohjelmakoodien ydin ja sitä käytetään Arduino kehitysalustojen aktiiviseen ohjaamiseen. (Arduino j; Fitzgerald & Shiloh 2013, 36.)

Rakenteen hallintaan tässä opinnäytetyössä tarvitaan if, if / else ja while komentoja. If komennolla voidaan tarkastaa, mikäli jokin tietty tila Arduino ohjelmakoodissa on saavutettu. Mikäli tarkkailtava tila on saavutettu, suoritetaan if komennolle määritellyt toiminnot, mutta jos tarkkailtavaa tilaa ei ole saavutettu, if komennolle määritellyt toiminnot jätetään väliin. (Arduino k.)

If / else komento mahdollistaa useampien tilojen tarkkailujen ryhmittämisen yhteen. If / else toimii samankaltaisesti kuin if komento, mutta sillä erolla, jos tarkkailtavaa tilaa ei ole saavutettu, suoritetaan else osalle määritellyt toiminnot. Mikäli tarkkailtava tila saavute-

taan, suoritetaan if / else komennon if osalle määritellyt toiminnot ja else osalle määritellyt toiminnot jätetään huomiotta. (Arduino l.)

While komentoa suoritetaan niin kauan, kunnes sille määritellystä lausekkeesta tulee epätosi. While komennolle määritellyn tarkkailtavan muuttujan on vaihduttava jollain tavalla ohjelmakoodissa tai while komentoa toistetaan ikuisesti. (Arduino m.)

Arduino ohjelmointikielen arvoja käsiteltäessä puhutaan muuttujista ja vakioista. Muuttujiin talletetaan tietoa ja niillä on nimi, arvo sekä tietotyyppi. Vakiot toimivat samoin kuin muuttujat, mutta vakioiden arvot eivät voi muuttua. Muuttujien arvot taas voivat muuttua Arduino ohjelman aikana. Arduino ohjelmointikielessä on myös joitain ennalta määriteltyjä vakioita, jotka on tehty ohjelmoinnin helpottamiseksi. Oleellisia ennalta määriteltyjä vakioita tämän opinnäytetyön kannalta ovat HIGH, LOW, INPUT ja OUTPUT. (Arduino n; Fitzgerald & Shiloh 2013, 46.)

Arduino ohjelmointiympäristön mukana tulee joukko ennalta määriteltyjä funktioita, jotka yksinkertaistavat ja selkeyttävät ohjelmointia. Funktiot ovat ohjelmointikoodin osia, jotka suorittavat niille määritellyn toiminnon. Toiminnon suorittamisen jälkeen ohjelmakoodissa palataan takaisin kohtaan, josta funktion suorittaminen käynnistettiin. Ohjelmointiympäristössä olevien valmiiden funktioiden lisäksi, funktioita voidaan määritellä itse ohjelmakoodiin ja niitä voidaan kutsua ohjelmakoodin muista osista. (Arduino r.)

4 Arduino monitorointilaitteen suunnittelu

Tämän opinnäytetyön tarkoituksena on valmistaa Arduino kehitysalustaa hyväksi käyttäen monitorointilaite, joka tarkkailee lämpötilaa ja liikettä. Tarkkailun lisäksi laite lähettää sähköpostilla varoitusilmoituksen, kun se havaitsee liikettä tai lämpötila nousee liian korkeaksi. Laitteen valmistus tehdään open source projektina, jotta laitteen jatkokehitys on mahdollista myös opinnäytetyö projektin ulkopuolisille henkilöille.

Laitetta valmistettaessa ja suunniteltaessa on pyritty yksinkertaisuuteen. Yksinkertaisella ja helpolla ratkaisulla halutaan pitää työn lopputuloksen kohderyhmä mahdollisimman laajana, mutta samalla on pyritty siihen, että laite toimii moitteetta ja sille on useita käyttötarkoituksia. Lisäksi laitteen valmistamiseen menevät kulut on pyritty minimoimaan. Opinnäytetyön tavoitteena on mahdollistaa laitteen valmistus myös niille, joilla ei ole paljoa aiempaa tietoteknistä kokemusta. Kohderyhmänä toimii siis kaikki, jotka ovat kiinnostuneita valmistamaan itse helppotekoisen ja halvan monitorointilaitteen. Open source projektin ansiosta myös tietoteknisesti kokeneemmat käyttäjät voivat jatkokehittää ja muokata laitetta omien tarpeidensa mukaan.

Opinnäytetyössä valmistuvalle laitteelle löytyy useita käyttötarkoituksia. Yleisellä tasolla laitetta voidaan käyttää edulliseen kulunvalvontaan ja lämpötilan tarkkailuun, jonka tarkkailuarvot ja toimintaominaisuudet ovat helposti muokattavissa. Yksityiset henkilöt voivat käyttää laitetta esimerkiksi kodin murtohälyttimenä ja mökin tai kasvihuoneen lämpötilan tarkkailuun. Esimerkki tilanteena voidaan käyttää mökkiä, jossa lämmitys pitää kytkeä päälle, mikäli lämpötila laskee halutun rajan alle. Pienille yrityksille laite voi myös toimia murtohälyttimenä tai jonkin pienen alueen liikehdinnän tarkkailuvälineenä. Lämpötilan tarkkailua pienet yritykset voivat tarvita esimerkiksi omien palvelinhuoneiden tarkkailuun. Jos palvelinhuoneesta erimerkiksi katkeaa ilmastointi ja siellä on useita palvelimia, lämpötila huoneessa nousee korkeaksi, joka saattaa aiheuttaa palvelinkoneiden sammumisen ja sitä myöten palvelukatkoksen.

Arduino monitorointilaitteen suunnittelussa, tavoitteiden saavuttamisen kannalta tärkeimmät tekijät olivat menetelmien ja osien valinnat. Laitteen valmistuksessa käytettävät osat olivat usein riippuvaisia käytettävistä menetelmistä etenkin kehitysalustojen kohdalla. Menetelmiä valittaessa eniten vaikutusta oli varoitusviestin lähettämiseen käytettävällä menetelmällä. Muita vaikuttavia tekijöitä olivat kehitysalustan ja anturien valinnat.

4.1.1 Varoitusviestin lähettämiseen käytettävän menetelmän valinta

Varoitusviestin menetelmää valittaessa lähtökohtana oli saada laite lähettämään varoitusviesti joko tekstiviestillä tai sähköpostilla. Menetelmää valittaessa alettiin tutkia mahdollisia ratkaisu vaihtoehtoja varoitusviestin toteuttamiseksi. Ratkaisu vaihtoehtojen tutkimisessa käytettiin internethakuja. Suurin painoarvo ratkaisu vaihtoehtojen tutkimisessa oli muiden Arduino käyttäjien tekemissä samankaltaisissa projekteissa.

Vanhoista open source projekteista ja muista internet lähteistä löytyi useita potentiaalisia ratkaisu vaihtoehtoja. Suurimmassa osassa ratkaisu vaihtoehdoista oli kuitenkin puutteita, sillä useimmat niistä vaativat erillisen palvelimen ja kolmannen osapuolen palvelun toimiakseen. Useiden ratkaisu vaihtoehtojen joukosta käytettäväksi valittiin ohjelmointikoodien virtualisointipalvelu Temboo. Temboon palveluista valittiin käytettäväksi gmail sähköpostiviestin lähettämisen suorittava palvelu, jota käytetään laitteen varoitusviestinä.

Temboon käyttöön varoitusviestin menetelmänä oli useita syitä. Temboo on kolmannen osapuolen palvelu, mutta se tekee yhteistyötä Arduinon kanssa, joten sen käyttäminen Arduinon kanssa on helppoa ja yksinkertaista. Temboo ratkaisua käytettäessä monitorointilaite voi toimia ilman ulkoisia laitteita, kuten palvelimia tai muita tietokoneita. Lisäksi ohjelmointikoodin virtualisointipalvelu hoitaa haastavimman vaiheen, eli sähköpostin lähettämisen. Mahdollisuus käyttää gmail-tiliä varoitusviestin lähettämiseen, tekee laitteesta myös helpommin lähestyttävän, koska gmail on niin laajasti käytetty ja tunnettu palvelu. Temboo palvelussa on myös rajoituksia. Temboo palvelun ilmaiskäyttäjä voi ajaa ohjelmointikoodin Temboon palvelimilta tuhat kertaa kuukaudessa ja tiedonsiirron määrä on rajoitettu 512 megabittiin. Ilmaiskäyttäjälle sallitut käyttömäärät riittävät kuitenkin helposti monitorointilaitteen käyttötarkoituksiin.

4.1.2 Kehitysalustan ja anturien valinta

Monitorointilaitteessa käytettävien kehitysalustojen ja anturien valintaan vaikuttavia tekijöitä olivat helppokäyttöisyys, yksinkertaisuus, toimivuus ja hinta. Lisäksi käytettävän kehitysalustan valintaan vaikutti paljon varoitusviestin lähettämiseen valittu menetelmä. Mikrokontrolleriksi valittiin Arduino Yún, lämpötila-anturiksi Analog Devicesin tmp36 ja liikkeen tarkkailuun Parallaxin Rev B PIR anturi.

Valittu varoitusmenetelmä mahdollisti kaksi eri vaihtoehtoa kehitysalustaa valittaessa. Vaihtoehtoina oli käyttää Arduino Yúnia tai jotain toista Arduino kehitysalustaa, johon voi yhdistää Ethernet lisälaitteen. Edellä mainituista vaihtoehdoista monitorointityökalun kehitysalustaksi valittiin Arduino Yún. Arduino Yún valittiin kehitysalustaksi, koska se on yksin-

kertaisempi integroida toimimaan Temboo palvelun kanssa ja se on ominaisuuksiltaan monipuolisempi. Arduino Yúnin monipuolisuus mahdollistaa monitorointilaitteen jatkokehittäjille huomattavasti enemmän vaihtoehtoja kehittämiseen. Arduino Yúnin valinnan ansiosta kehitysalustaan ei tarvitse liittää ylimääräisiä lisälaitteita, joiden asentaminen vaatisi ylimääräistä työtä. Lisäksi Arduino Yún tukee SSL:lää, toisin kuin useimmat muut Arduino kehitysalustat, joten Arduino Yúnia käytettäessä tietoturva on parempi. Arduino Yún on hieman kalliimpi ratkaisu, kuin halvimpien Arduino kehitysalustojen ja Ethernet lisälaitteiden käyttö, mutta ero ei kuitenkaan ole suuri.

Antureiden valinnassa pääpaino oli yksinkertaisuudessa. Molempien anturien valintaan vaikutti se, ettei niitä tarvitse erikseen kalibroida. Valittu Parallax Rev B anturi kalibroituu automaattisesti 20 – 60 sekunnin aikana, kun sille annetaan virtaa. PIR anturi sopii hyvin tässä opinnäytetyössä valmistuvan monitorointilaitteen käyttötarkoituksiin, koska PIR anturi tarkkailee muutoksia infrapunan määrässä. Useimmat mahdolliset liikkeen tarkkailua vaativat käyttötarkoitukset perustuvat ihmisen liikkeen tarkkailuun, joten infrapunan tarkkailu on hyvä menetelmä siihen.

Valitun tmp36 lämpötila-anturin käyttämä tarkkailumenetelmä ei vaadi kalibrointia, koska sen käyttämä menetelmä muuttaa annettavaa jännitettä ympäröivän lämpötilan perusteella. Tmp36 lämpötila-anturin käyttämä menetelmä on myös hyvä, koska sen antama jännite muuttuu tunnetussa suhteessa ympäröivään lämpötilaan, joten jännite voidaan muuttaa tarkasti mihin lämpötilayksikköön tahansa. Lisäksi tmp36 lämpötila-anturi on edullinen.

Anturien liittämiseen kehitysalustaan käytetään koekytkentälevyä ja kahdenlaisia kytkentäjohtoja. Koekytkentälevyn käytön ansiosta laitteen valmistuksessa liitäntöjä ei tarvitse juottaa. Koekytkentälevyn käyttö on myös muuten kaikin puolin helpompaa ja se helpottaa erilaisten huomattavasti laitteessa käytettävien piirien testaamista. Lisäksi PIR anturin liittämiseen käytetään kolmea uros – naaras johtoa, jotta laite voidaan kytkeä suoraan kehitysalustaan. Koekytkentälevyn, lämpötila-anturin ja muiden liitäntöjen toteuttamiseksi tarvitaan kuusi kappaletta uros – uros johtoja.

5 Arduino monitorointilaitteen valmistaminen

Arduino monitorointilaitteen valmistaminen toteutetaan useassa vaiheessa, jotta laitteessa olevat virheet on helpompi havaita. Molemmat laitteessa käytettävät anturit laitetaan toimimaan yksitellen. Kun molemmat anturit on saatu toimimaan erikseen, kootaan laite, jossa on molemmat anturit, jonka jälkeen molemmissa laitteissa käytetyt ohjelmakoodit yhdistetään sekä muokataan toimivaksi. Myös varoitusviestin lähettämiseen tarvittava ohjelmakoodi tehdään erikseen ja sen toteuttamiseen tarvittavat toimenpiteet esitellään. Lopussa varoitusviestiin käytettävä ohjelmakoodi yhdistettään antureita kontrolloivaan ohjelmakoodiin ja fyysiseen laitteeseen tehdään tarvittavat muutokset.

Ennen laitteiden valmistamisen aloittamista tulee huomioida muutamia asioita. Arduino kehitysalustaan ei saa olla kytkettynä virtaa, kun laitteita kootaan. Arduino ohjelmakoodin kirjoitetaan Arduino ohjelmointiympäristöllä, joka on ladattavissa osoitteesta http://arduino.cc/en/Main/Software. Ohjelmointiympäristöä ladattaessa tulee varmistaa, että ladattava versio tukee Arduino Yún kehitysalustaa.

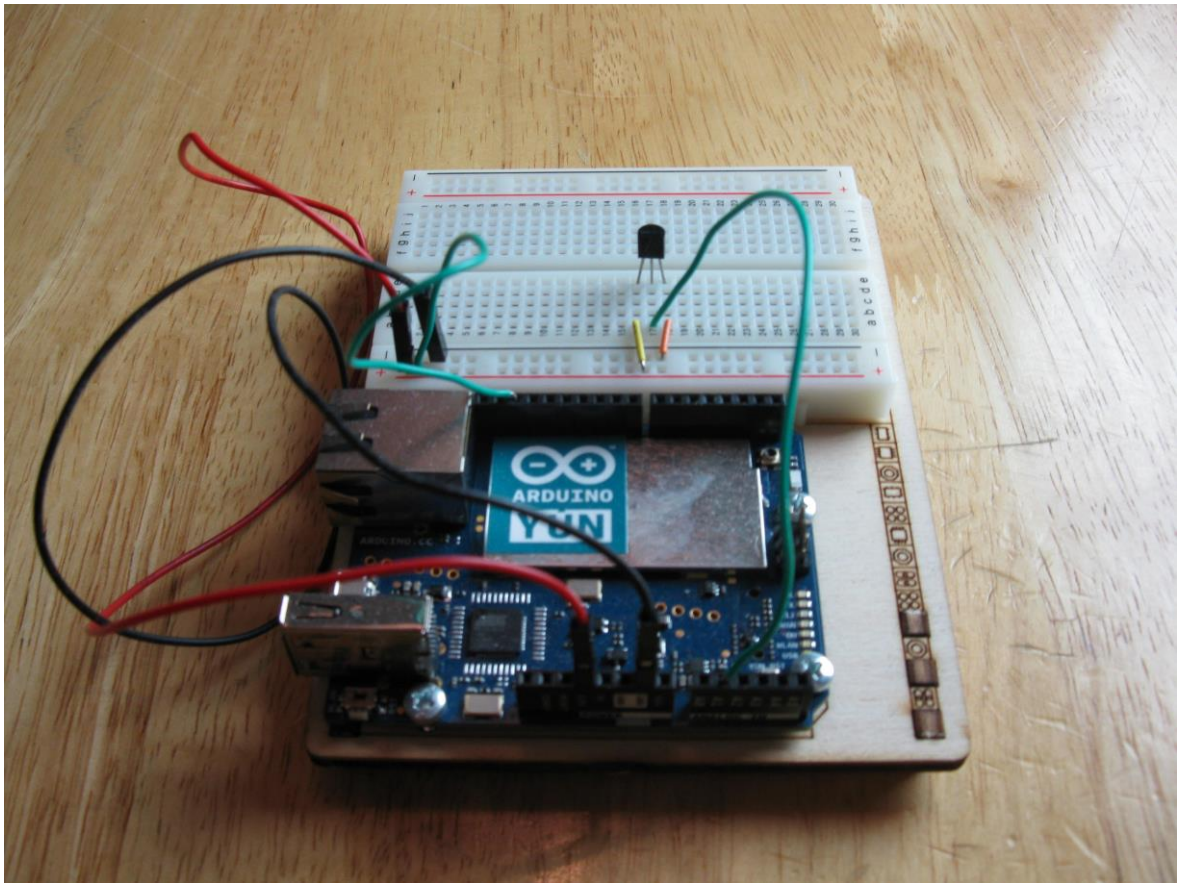
Viimeistään ennen ohjelmakoodin lataamista laitteelle, tulee Arduino ohjelmointiympäristössä valita laitteessa käytettävä Arduino kehitysalusta ja laitteiden väliseen kommunikointiin käytettävä portti. Valinta tehdään ohjelmointiympäristön ”Tools”-valikosta, josta löytyy valikot ”Board” ja ”Port”. Arduino kehityslevyn tulee olla kytkettynä tietokoneeseen, jotta käytettävän portin voi valita. Arduino ohjelmakoodin lataaminen laitteelle voidaan toteuttaa työkalupalkin ”Upload”-painikkeella ja ohjelmakoodin tallentaminen tietokoneelle työkalupalkin ”Save”-painikkeella.

5.1 Lämpötila-anturin ohjelmointi ja valmistaminen

Monitorointilaitteen valmistaminen aloitetaan valmistamalla laite, joka sytyttää Arduino Yúnin sisäänrakennetun LED-valon, kun lämpötila nousee määriteltyä rajaa korkeammaksi. Ensiksi kootaan laite eli siihen yhdistettään tarvittava anturi, jonka jälkeen aloitetaan ohjelmakoodin kirjoittaminen. Lämpötilan tarkkailuun käytettävä laite kootaan käyttäen tmp36 lämpötila-anturia, koekytkentälevyä, Arduino Yúnia ja yhdistämiseen tarvittavia johtoja.

Kytkentä aloitetaan yhdistämällä 3.3V pinni koekytkentälevyn virtaliitäntään ja maadoitus pinni koekytkentälevyn maadoitusliitäntään. Myös aref pinni tulee liittää koekytkentälevyn virtaliitäntään. Lämpötila-anturi kiinnitetään koekytkentälevyyn ja sen keskimmäinen jalka

liitetään koekytkentälevyn avulla analogiseen pinniin A1. Lämpötila-anturin vasemmanpuoleinen jalka liitetään koekytkentälevyn virtaliitäntään ja oikeanpuoleinen jalka liitetään koekytkentälevyn maadoitusliitäntään. Tässä vaiheessa on tärkeää huomioida, että lämpötilasensorin vasemman- ja oikeanpuoleiset jalat määritellään katsottaessa tmp36 lämpötila-anturin tasaista puolta. Väärin kytkettynä maadoitus ja virta jalat menevät väärinpäin, joka saattaa aiheuttaa lämpötila-anturin hajoamisen, eikä lämpötila-anturi toimi. Kun laite on koottu, sen tulisi näyttää kuvan yksi mukaiselta.



Kuva 1. Lämpötila-anturia hyödyntävä laite

Tämän osion ohjelmakoodi koostuu setup() ja loop() funktioista, sekä niiden ulkopuolelle määritellyistä vakioista, muuttujista ja muista funktioista. Käytettävät komennot selitetään siinä järjestyksessä, kun ne esiintyvät ohjelmakoodissa. Tämän osion ohjelmakoodi löytyy liitteestä kaksi ja ohjelmakoodi selitetään siinä esiintyvässä järjestyksessä.

Ensimmäisenä ohjelmakoodissa on määritelty vakiot. Vakiota ovat tempsensorPin, ledPin, warningTemp ja arfeVoltage. Muuttujat määritellään vakoiksi käyttämällä const etuliitettä ennen tietotyyppiä. Lämpötila-anturiin viitataan vakiossa tempsensorPin, jossa määritellään pinni, johon lämpötila-anturi on kytketty. Lämpötila-anturi on kytketty analogiseen pinniin yksi, joten tempsensorPin vakiolle annetaan arvo A1. Vakiolla ledPin viitataan

LED-valoon, joka sytytetään, kun lämpötila nousee annetun raja-arvon yläpuolelle. Arduino kehitysalustojen sisäänrakennetulle LED-valolle on useimmissa tapauksissa varattu digitaalinen pinni kolmetoista, joten ledPin vakiolle annetaan arvo 13. Koska ledPin ja tempsensorPin vakioiden arvot ovat tasalukuja, niissä käytetään tietotyyppiä int.

Murtolukuja tarvitaan vakioissa warningTemp ja arfeVoltage, joten niiden tietotyyppinä toimii float. Lämpötila, joka toimii raja-arvona LED-valon sytyttämiselle, määritellään vakiossa warningTemp. Lämpötilan raja-arvoksi on tässä tapauksessa määritelty 28.01, näin ollen warningTemp vakiolle annetaan arvo 28.01. Vakiota arefVoltage käytetään myöhemmin, kun halutaan muuntaa analogisen pinnin saama arvo volteiksi. Vakiolle arefVoltage annetaan arvo 3.3, joka on sama kuin jännitemäärä jolla lämpötilasensoria käytetään.

Vakioiden määrittelyn jälkeen määritellään void setup() funktio ja sen sisään tulevat funktiot. Ensimmäisenä setup() funktion sisällä on Serial.begin(), joka aloittaa tietokoneen ja kehitysalustan välisen kommunikoinnin, sekä asettaa kommunikoinnin tiedonsiirtonopeudeksi 9600 baudia. Seuraavaksi määritellään pinMode() funktion avulla lämpötila-anturia käyttävä pinni tuloliitännäksi ja sisäisen LED-valon pinni lähtöliitännäksi. Tulo- ja lähtöliitäntöjen jälkeen tulee funktio analogReference(), koska tässä laitteessa analogisten pinnien vertausarvona halutaan käyttää 3.3 voltin jännitettä, mutta oletuksena Arduino Yún käyttää sen omaa viiden voltin toimintajännitettä vertausarvona. Vaihtoehtoa EXTERNAL käytetään funktion analogReference() kanssa, koska laitteessa halutaan käyttää vertausarvona samaa jännitettä, jolla lämpötila-anturi toimii. Toisin sanoen vertausarvona käytetään jännitettä, joka aref pinnille syötetään.

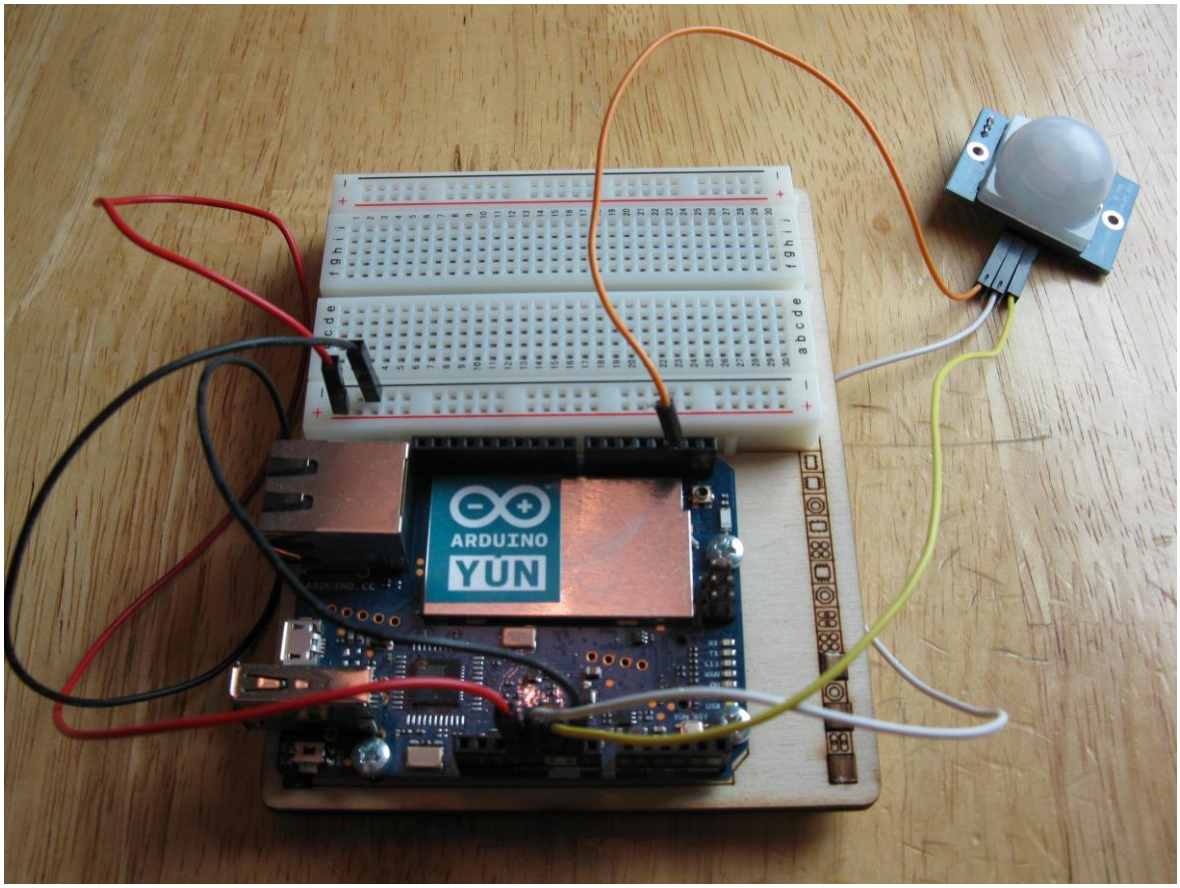
Setup() funktion jälkeen aloitetaan void loop() funktion suorittaminen. Ensimmäisenä loop() funktion sisällä määritellään muuttuja int sensorVal, joka saa arvon lämpötila-anturilta. Lämpötila-anturia luetaan funktiolla analogRead, joka saa arvon analogiselta pinniltä, johon lämpötila-anturi on liitetty. Lämpötila-anturin lukemisen jälkeen, tulostetaan serial monitorille lämpötila-anturin antama lukema. Tulostaminen tehdään funktiolla Serial.print(). Muutetaan lämpötila-anturin antama lukema volteiksi ja annetaan muuttujan float voltage arvoksi saatu volttilukema. Jännitteen määrä tulostetaan serial monitorille, jonka jälkeen muunnetaan jännitteen määrä celsiusasteiksi ja saatu lukema annetaan muuttujan float temperature arvoksi. Funktiolla Serial.println() tulostetaan celsiuslukema ja tehdään rivinvaihto serial monitorille. Tulostuksen jälkeen on asetettu 200 millisekunnin mittainen tauko analogisen pinnin muuntajan ruuhkautumisen välttämiseksi. Tauko toteutetaan komennolla delay().

Celsiuslukeman tulostuksen jälkeen tarkistetaan, onko saatu celsiuslukema suurempi kuin raja-arvoksi asetettu lukema. Tarkistus toteutetaan if / else komennon avulla. Mikäli celsiuslukema on suurempi kuin raja-arvo lukema, annetaan sisäisen LED-valon pinnille arvo HIGH digitalWrite funktion avulla. Jos celsiuslukema on pienempi kuin raja-arvon lukema, annetaan sisäisen LED-valon pinnille arvo LOW.

5.2 PIR anturin ohjelmointi ja valmistaminen

Lämpötilaa tarkkailevan laitteen valmistamisen jälkeen tehdään laite, joka tarkkailee liikettä. Liikkeen tarkkailu toteutetaan PIR sensorin avulla ja valmistuva laite sytyttää Arduino Yúnin sisäisen LED-valon kun se havaitsee liikettä. PIR anturilla toimivan laitteen valmistaminen aloitetaan kokoamalla fyysinen laite, jonka jälkeen sille kirjoitetaan tarvittava ohjelmointikoodi.

Lämpötilan tarkkailu toteutetaan käyttäen Parallax Rev B PIR anturia, joka kalibroituu automaattisesti. PIR anturin lisäksi laitteen valmistamiseen käytetään Arduino Yúnia ja yhdistämiseen tarvittavia johtoja. Tähän osioon ei tarvita koekytkentälevyä. Kokoamisen aikana Arduino Yún ei saa olla kytkettynä virtalähteeseen. Laitteen kokoaminen aloitetaan liittämällä 5V pinni PIR anturin keskimmäiseen eli VCC pinniin ja maadoituspinni PIR anturin oikeanpuoleiseen GND pinniin. PIR anturin jäljelle jäävä OUT pinni liitetään Arduino Yúnin digitaaliseen pinniin numero kaksi. Kokoamisen jälkeen laite näyttää kuvan kaksi mukaiselta.



Kuva 2. PIR anturia hyödyntävä laite

PIR anturin ohjelmakoodi löytyy liitteestä kolme. PIR anturin ohjelmakoodi aloitetaan määrittelemällä vakiot, jotka ovat pirsensorPin ja ledPin. Vakiolla pirsensorPin viitataan pinniin, johon PIR anturi on liitetty. PIR anturia käytetään digitaalisessa pinnissä numero kaksi, joten vakio pirsensorPin saa arvon 2. Vakio ledPin toimii tässä osiossa täsmälleen samalla tavalla, kuin lämpötilaa tarkkailevan laitteen kanssa, joten se saa myös tässä osiossa arvon 13.

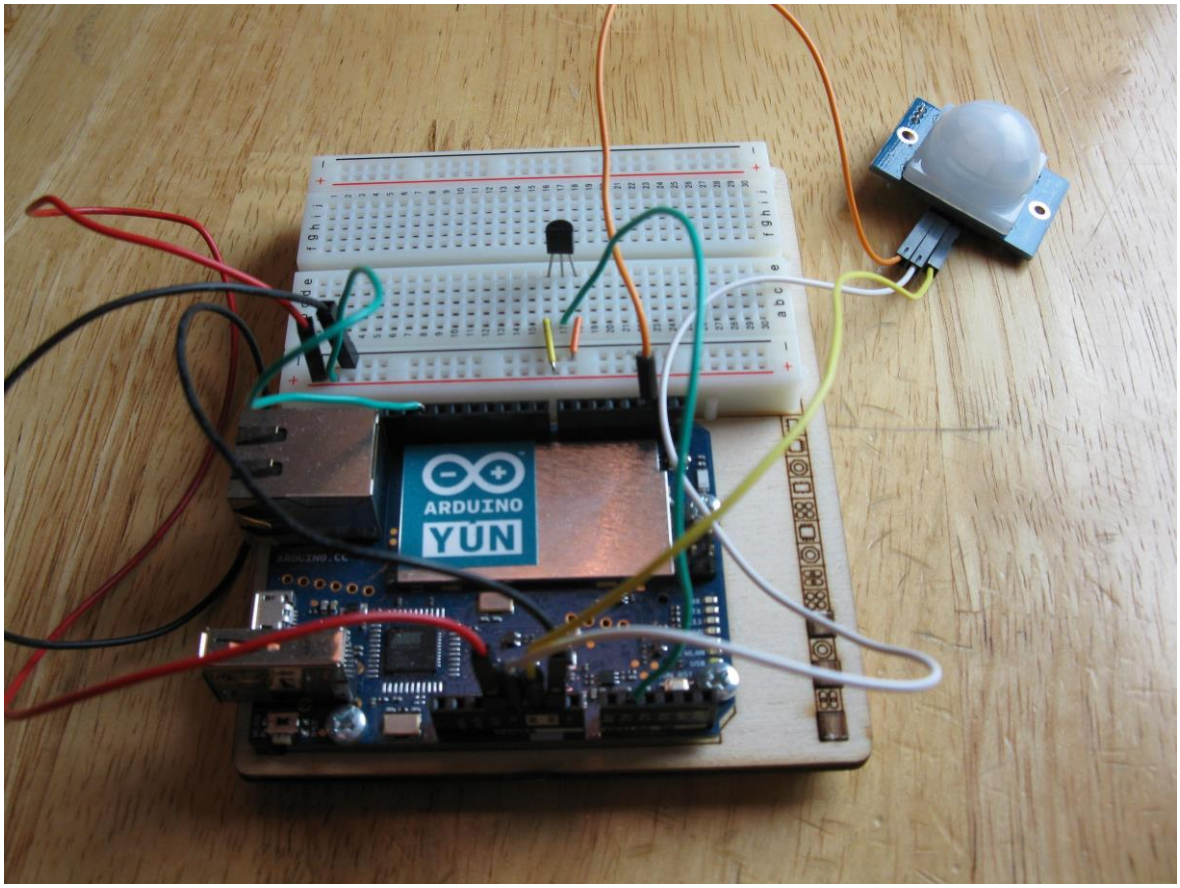
Funktion setup() sisälle määritellään lähtö- ja tuloliitännät samalla tavalla, kuin ne määriteltiin lämpötilaa tarkkailevaan laitteeseen. Funktioon loop() ei ole määritelty muuta kuin if / else lauseke, joka tarkkailee PIR anturin tilaa. Mikäli PIR anturi havaitsee liikettä, digitalRead funktio saa arvon HIGH ja LED-valo sytytetään. Kun PIR anturi ei havaitse liikettä, sisäisen LED-valon pinni saa arvon LOW.

5.3 PIR anturin ja lämpötila-anturin yhdistäminen

Molempien yksittäisiä antureja käyttävien laitteiden valmistamisen jälkeen, ne yhdistetään laitteeksi, joka tarkkailee liikettä ja lämpötilaa. Tässä osiossa valmistuva laite sytyttää Ar-

duino Yúnin sisäänrakennetun LED-valon, kun lämpötila nousee sallitun raja-arvon yli tai PIR anturi havaitsee liikettä. Laite toimii runkona lopullisille laitteelle, johon yhdistetään varoitusviestin lähettämiseen tarvittavat ohjelmakoodit. Ensimmäiseksi valmistetaan fyysinen laite, jonka jälkeen aiemmin tehdyt ohjelmakoodit yhdistetään ja niitä muokataan tarpeen mukaan.

Lämpötilaa ja liikettä tarkkailevan laitteen kokoaminen on yksinkertaista, jos laitteen kokoamisen aiemmat vaiheet on tehty. Laite kootaan yksinkertaisesti liittämällä molemmat anturit Arduino Yúniin samanaikaisesti. Laitteen kokoaminen voidaan aloittaa liittämällä PIR anturi Arduino kehitysalustaan, kuten kuvassa kaksi on tehty. PIR anturin liittämisen jälkeen liitetään lämpötila-anturi kuvan yksi tapaan. Tässä osiossa valmistuva fyysinen laite on nyt lopullisessa muodossaan, eikä siihen tule enää muutoksia varoitusviesti ominaisuuden liittämisen jälkeen. Lopullinen laite on esitetty kuvassa kolme.



Kuva 3. Lämpötilaa ja liikettä tarkkaileva laite

Tämän osion ohjelmakoodi voidaan tehdä helposti yhdistämällä aiempien osioiden vakiot ja setup() funktion sisällä olevat funktiot. Vakioiden ja asetus funktioiden yhdistämisen lisäksi tehdään pieni muutos loop() funktion if / else lausekkeeseen. Vakioita määriteltäessä tulee huomioida, että vakio ledPin määritellään vain kerran ja setup() funktiossa sisäi-

sen LED-valon pinni määritellään vain kerran lähtöliitännäksi. Komentoa if /else muokataan siten, että se tarkkailee molempien anturien arvoa samanaikaisesti ja sytyttää LED-valon, mikäli edes toinen arvoista on tosi. Molempien anturien tarkkailu toteutetaan laittamalla molemmat tarkkailuarvot samaan lausekkeeseen ja niiden väliin looginen operaattori ||. Tämän osion ohjelmakoodi löytyy kokonaisuudessaan liitteestä neljä.

5.3.1 Varoitusviestin lähettämiseen vaaditut toimenpiteet

Varoitusviestin lähettäminen toteutetaan käyttäen Temboo ohjelmointikoodin virtualisointipalvelua. Varsinainen varoitusviestin lähettäminen tapahtuu Temboo palvelun toimesta, joka toteutetaan ohjelmakoodissa käyttämällä Temboo choreoita. Ennen varoitusviestin toteuttamista tulee tehdä muutamia toimenpiteitä, jotta varoitusviestin lähettäminen on mahdollista.

Temboo palvelun käyttäminen vaatii rekisteröitymisen, joten ensimmäiseksi rekisteröidytään palveluun osoitteessa www.temboo.com. Lisäksi varoitusviestin lähettäminen vaatii gmail sähköpostitilin, koska varoitusviesti lähetetään gmail sähköpostin kautta. Yhteys Temboo palveluun tapahtuu langattoman verkkoyhteyden välityksellä, joten Arduino Yúnin WiFi ominaisuus pitää asettaa toimivaksi.

Arduino Yúnin WiFi ominaisuuden asettamiseen tarvitaan itse kehitysalustan lisäksi tietokone, joka pystytään yhdistämään langattomaan verkkoon ja micro USB-kaapeli. Arduino Yúnin WiFi ominaisuuden asettaminen aloitetaan liittämällä Arduino Yún tietokoneeseen micro USB-kaapelin avulla.

Arduino Yún luo automaattisesti langattoman verkon, jonka nimi on muodossa Arduino-Yun-XXXXXXXXX, kun Arduino Yún käynnistetään ensimmäisen kerran. Arduino Yúnin liittämisen jälkeen, tietokone yhdistetään Arduino Yúnin luomaan langattomaan verkkoon. Tietokoneen yhdistettyä valittuun langattomaan verkkoon, avataan verkkoselain ja yhdistetään osoitteeseen http://arduino.local tai 192.168.240.1.

Annetun osoitteen syöttämisen jälkeen verkkoselaimeen aukeaa uusi verkkosivu, jonka password kenttään syötetään salasana ”arduino” ja klikataan painiketta “log in”. Kirjautumisen myötä auenneelta verkkosivulta klikataan painiketta “configure”, joka avaa jälleen uuden sivun. Auenneelta sivulta valitaan langatonverkko, jota Arduino Yúnin halutaan käyttävän. Lisäksi asetetaan muut asetukset, kuten salasana ja aikavyöhyke halutun lai-

siksi. Asetusten asettamisen jälkeen klikataan painiketta “configure & restart”. Painikkeen klikkaamisen jälkeen Arduino Yún käynnistyy uudelleen ja yhdistää määriteltyyn langattomaan verkkoon.

5.3.2 Varoitusviestin toteuttaminen

Varoitusviestin lähettämiseen käytettävän ohjelmakoodin pohjana on käytetty Temboo palvelun esimerkkiohjelmaa, jota voidaan käyttää gmail sähköpostin lähettämiseen Arduino Yúnilla. Esimerkkiohjelmaa on muokattu tämän monitorointilaitteen tarpeiden mukaiseksi. Muokkaamisen jälkeen varoitusviestiin tarvittava ohjelmakoodi on liitetty aiemmin luodun monitorointilaitteen ohjelmakoodiin. Lisäksi monitorointilaitteen ohjelmakoodia on muokattu toimimaan varoitusviestin lähettämiseen tarvittavan ohjelmakoodin kanssa.

Esimerkkiohjelma on löydettävissä Temboon verkkosivuilta osoitteesta https://www.temboo.com/arduino/yun/send-an-email. Esimerkkiohjelman käyttäminen vaatii Temboo palveluun kirjautumista ja se koostuu itse varoitusviestin lähettämiseen tarvittavasta ohjelmakoodista sekä header-tiedostosta. Header-tiedosto sisältää Temboo palveluiden käyttämiseen vaaditut tilitiedot ja siitä tehdään erillinen .h-päätteinen tiedosto, joka liitetään Arduino ohjelmakoodiin. Pohjana käytetty esimerkkiohjelma löytyy liitteestä viisi ja header-tiedosto liitteestä kuusi.

Lopullisen monitorointilaitteen varoitusviestin toteuttamiseksi, Temboon gmail sähköpostin lähettämiseen tarkoitetusta esimerkkiohjelmasta liitetään osia aiemmin luodun monitorointilaitteen ohjelmakoodiin. Aiemmin luodun monitorointilaitteen ohjelmakoodiin tehdään muutoksia setup() funktioon, loop() funktion if / else lausekkeeseen ja pääfunktioiden ulkopuolisiin määrittelyihin. Lisäksi monitorointilaitteen ja varoitusviestin toiminnan parantamiseksi tehdään lopulliseen ohjelmakoodiin muutamia esimerkkiohjelmasta riippumattomia muutoksia.

Esimerkkiohjelmasta liitetään pääfunktioiden ulkopuolisiin määrittelyihin kahden ohjelmakirjaston ja header-tiedoston liittämiseen tarvittavat määrittelyt. Liitettävät ohjelmakirjastot ovat Temboo ja Bridge. Temboo ohjelmakirjastoa tarvitaan, jotta Temboo palvelua voidaan käyttää ja Bridge ohjelmakirjastoa käytetään mahdollistamaan Arduino Yúnin mikrokontrollerin sekä toisen prosessorin välinen kommunikointi. Kirjastot määritellään #include komennolla. Samalla komennolla liitetään myös header-tiedosto. Lisäksi määritellään kolme const String vakiota, jotka sisältävät sähköpostien tilitietoja. GMAIL_USER_NAME vakioon määritellään gmail sähköpostiosoite, josta varoitusviesti halutaan lähettää. GMAIL_PASSWORD vakioon määritellään sähköpostiosoitteen salasana, josta varoitus-

viesti lähetetään. TO_EMAIL_ADDRESS vakioon voidaan määritellä mikä tahansa sähköpostiosoite, johon varoitusviestin halutaan saapuvan. Koska lopullisessa monitorointilaitteessa ei käytetä Arduino kehitysalustan sisäistä LED-valoa, poistetaan siihen liittyvät tiedot.

Funktion setup() sisälle lisätään uusi funktio, jota ei ole aiemmassa osiossa luodun monitorointilaitteen ohjelmointikoodissa. Lisättävä funktio on Bridge.begin(), joka aloittaa Arduino Yúnin varsinaisen mikrokontrollerin ja vaihtoehtoisen prosessorin välisen kommunikoinnin. Pääfunktion setup() sisältä poistetaan myös lähtöliitäntä määrittely Arduino kehitysalustan sisäiseltä LED-valolta.

Ohjelmakoodin sisältö pysyy lähes samana loop() funktion osalta. If / else lausekkeen sisälle kuitenkin liitetään esimerkkiohjelman if / else lausekkeen sisältö, jota muokataan myös hieman. Temboon esimerkkiohjelma on tarkoitettu pääsääntöisesti sähköpostin lähettämiseen käytettävän ominaisuuden testaamiseen. Esimerkkiohjelma lähettää sähköpostin vain kerran, mutta tämän monitorointilaitteen halutaan jatkavan toimintaansa myös sähköpostin lähettämisen jälkeen. Edellä mainitusta syystä vaihdetaan kokonaan if / else lausekkeen tarkkailtavat arvot ja käytetään siinä samoja tarkkailuarvoja kuin PIR anturin ja lämpötila-anturin sisältävässä laitteessa. Lisäksi poistetaan esimerkkiohjelman lopun if / else lausekkeesta muuttuja success. Myös lopussa oleva delay() funktio poistetaan. Viestikentät muokataan tarkoitukseen sopivimmiksi.

Varoitusviestin lisäämisen jälkeen, siihen tehdään vielä muutama käytännöllisyyttä parantava lisäys. Ohjelmakoodin lopussa olevaan if / else lausekkeen, joka tarkkailee, mikäli varoitusviesti on lähetetty, lisätään kaksi taukoa delay() funktioilla. Lisättäviä delay() funktioita käytetään muuttujien avulla, joten ohjelmakoodiin lisätään kaksi muuttujaa pääfunktioiden ulkopuolisiin määrittelyihin. Lisättävien muuttujien tietotyyppinä käytetään unsigned longia, jotta delay() funktion viive saadaan tarvittaessa mahdollisimman pitkäksi. Lisättävien muuttujien nimet ovat seconds ja minutes. Seconds muuttujalle määrillään arvoksi 1000, joka vastaa millisekunteina yhtä sekuntia. Minutes muuttujan arvoksi määritellään 60000, kertomalla seconds muuttuja kuudellakymmenellä. Lisätään ohjelmakoodin lopussa olevan if / else lausekkeen if osion loppuun viidentoista minuutin tauko, joka toteutetaan kertomalla minutes muuttuja viidellätoista. Saman if / else lausekkeen else osion loppuun lisätään kahdenkymmenen sekunnin tauko, joka toteutetaan kertomalla seconds muuttuja kahdellakymmenellä.

Varsinaisen ohjelmakoodin liittämisen lisäksi luodaan header-tiedosto. Header-tiedostosta haetaan tiedot, kun ohjelmakoodista viitataan siihen. Header-tiedosto luodaan Arduino

ohjelmointiympäristössä uudelle välilehdelle, varsinaisen ohjelmakoodin rinnalle. Uusi välilehti luodaan Arduino ohjelmointiympäristön oikeassa yläkulmassa olevasta vetovalikosta, josta valitaan vaihtoehto ”New Tab”. Uudelle välilehdelle liitetään esimerkkiohjelmassa käytetyt header-tiedoston tiedot, jotka löytyvät liitteestä kuusi.

5.4 Lopullisen monitorointilaitteen toimintakuvaus

Tässä kappaleessa kuvataan lopullisen monitorointilaitteen ohjelmakoodin eteneminen ja toiminta. Funktioiden setup() ja loop() ulkopuolelle on määritelty vakioita ja muuttujia, joille määritellään arvot. Määriteltyjä muuttujia ja vakioita käytetään myöhemmin ohjelmakoodissa setup() ja loop() funktioiden sisällä. Lisäksi pääfunktioiden ulkopuolella liitetään Temboo ja Bridge ohjelmakirjastot sekä header-tiedosto.

Pääfunktion setup() sisällä käynnistetään Arduino kehitysalustan ja ulkoisen tietokoneen, sekä kehitysalustan mikrokontrollerin ja toisen prosessorin välinen kommunikointi. Lisäksi setup() funktion sisällä määritellään antureita käyttävät pinnit tuloliitännöiksi. Asetetaan myös analogiset pinnit käyttämään vertausarvona ulkopuolista virtalähdettä.

Pääfunktion loop() toiminta aloitetaan lukemalla lämpötila-anturin antama arvo ja se tulostetaan serial monitorille. Seuraavaksi muutetaan lämpötila-anturilta saatu arvo jännitelukemaksi ja jännitelukema muutetaan celsiuslukemaksi, jonka lisäksi molemmat lukemat tulostetaan serial monitorille. Lämpötila-anturin lukemisen ja lukemien tulostamisen jälkeen ohjelman suorittamisessa pidetään 0,2 sekunnin tauko. Ohjelmakoodin seuraavassa vaiheessa luetaan PIR anturin antama lukema ja verrataan aiemmin saatua celsiuslukemaa lämpötilan raja-arvoon.

Ohjelman suorittamisessa jatketaan eteenpäin, mikäli celsiuslukema on suurempi kuin määritelty raja-arvo tai PIR anturia luettaessa saadaan arvo HIGH. Muissa tapauksissa palataan loop() funktion alkuun. Jos ohjelman suorittamisessa jatketaan eteenpäin, tulostetaan serial monitorille ilmoitus, että varoitusviestiä yritetään lähettää. Ilmoituksen jälkeen yhdistetään Temboo palveluun. Temboo palvelun käyttämiseen tarvittavat tilitiedot haetaan header-tiedostosta. Asetetaan choreoon gmail sähköpostitili, jota käytetään sähköpostin lähettämiseen, sähköpostitilin salasana ja sähköpostitili, johon varoitusviesti halutaan lähettää. Asetetaan myös varoitusviestissä käytettävä otsikko ja varoitusviestin sisältö.

Choreossa käytettävien tietojen asettamisen jälkeen choreo suoritetaan ja odotetaan, että se palauttaa arvon. Choreon palauttama arvo riippuu siitä onnistuiko vai epäonnistuiko

varoitusviestin lähettäminen. Choreon palauttama arvo asetetaan muuttujaan, jonka perusteella määritellään ohjelman jatkotoimenpiteet. Mikäli varoitusviestin lähetys onnistui, tulostetaan serial monitorille ilmoitus lähettämisen onnistumisesta ja ilmoitus ajasta, jonka jälkeen monitorointia jatketaan. Onnistuneen lähetyksen jälkeen ohjelma pitää viidentoista minuutin tauon, jonka jälkeen choreon suorittaminen lopetetaan ja siirrytään loop() funktion alkuun. Varoitusviestin lähetyksen epäonnistuessa tulostetaan serial monitorille virheilmoitus, joka riippuu choreon palauttamasta arvosta. Virheilmoituksen tulostamisen jälkeen ohjelma pitää kahdenkymmenen sekunnin tauon. Tauon jälkeen choreon suorittaminen lopetetaan ja siirrytään loop() funktion alkuun.

5.5 Monitorointilaitteen testaus

Jokaista monitorointilaitteen osakokonaisuutta testattiin, kunkin osakokonaisuuden edetessä ja osakokonaisuuden päätteeksi. Lämpötilan tarkkailun testaamiseen käytettiin serial monitorille tulostettavia lämpötila- ja volttilukemia. Serial monitorilla näkyvän lämpötilalukeman oikeellisuus tarkastettiin vertaamalla laitteen antamaa lukemaa ulkoisen lämpötilamittarin antamaan lukemaan. Lämpötilan tarkkailun varoitusominaisuutta testattiin yksinkertaisesti lämmittämällä lämpötila-anturia sormilla, samalla tarkkaillen serial monitorille tulostettavia lämpötilalukemia. Testauksessa tarkkailtiin, laukaiseeko laite varoituksen, kun lämpötila nousee yli annetun raja-arvon.

Liikkeentunnistuksen testauksessa käytettiin samaa tapaa, kuin lämpötilan testaamisessa, mutta serial monitoria ei käytetty. Laitteessa käytetyssä PIR anturissa on sisäinen LED-valo joka syttyy, kun PIR anturi havaitsee liikettä. Liikkeentunnistuksen testauksessa tarkkailtiin siis laukeaako varoitus, kun PIR anturin sisäinen LED-valo syttyy.

Varoitusviestin testauksessa käytettiin apuna myös serial monitoria, johon tulostettiin ilmoitukset esimerkiksi viestin lähettämisen onnistumisesta ja virheilmoitukset. Varoitusviestiä testattaessa, käytettiin yllämainittuja menetelmiä varoitusviestin laukaisemiseen, jonka jälkeen tarkistettiin varoitusviestin kohde sähköpostista saapuiko varoitusviesti.

Lopulliselle monitorointilaitteelle toteutettiin myös pidempiaikainen testaus. Pitkäaikaisella testauksella kokeiltiin laitteen toimintaa pidempiaikaisessa käytössä. Myös anturien häiriöherkkyyttä haluttiin testata. Pidempiaikaisessa testissä monitorointilaite oli kytketty päälle kolmen vuorokauden ajaksi. PIR anturi laitettiin suljettuun tilaan, jotta lämpötila-anturia voitiin testata helpommin. Kolmen vuorokauden ajan lämpötila-anturia ja PIR anturia testattiin satunnaisessa järjestyksessä, jonka lisäksi jokaisen kokeilun jälkeen tarkastettiin toimiko varoitusviesti. Varoitusviestin kohde sähköpostista pystyttiin helposti tarkastamaan

mahdolliset väärät varoitusviestit. Kolmen vuorokauden aikana PIR anturia testattiin yhteensä seitsemän kertaa ja lämpötila-anturia kuusi kertaa. Yhtään väärää varoitusviestiä ei kolmen vuorokauden aikana saapunut kohde sähköpostiin.

6 Pohdinta

Kokonaisuudessaan opinnäytetyölle asetetut tavoitteet saavutettiin. Opinnäytetyössä onnistuttiin valmistamaan monitorointilaite, joka tarkkailee määriteltyjä ympäröivän maailman tiloja. Valmistunut monitorointilaite myös lähettää varoitusviestin määriteltyjen raja-arvojen perusteella, joka oli tärkeä osa monitorointilaitteelle asetettuja tavoitteita. Monitorointilaitteen haluttiin lisäksi olevan yksinkertainen ja helposti toteutettavissa oleva.

Monitorointilaitteen tavoitteiden saavuttamisen kannalta tärkeitä asioita pohdittiin monitorointilaitteen suunnitteluvaiheessa. Monitorointilaitteen suunnittelussa onnistuttiin, sillä siinä onnistuttiin valitsemaan helposti toteutettavissa olevat menetelmät ja laitteessa käytettävät osat. Etenkin varoitusviestin lähettämiseen valittu ohjelmointikoodin virtualisointipalvelu menetelmä oli huomattavasti yksinkertaisempi, kuin muut suunnitteluvaiheessa tarkastellut menetelmät. Lisäksi valittu kehitysalusta oli helppo saada toimimaan valitun virtualisointipalvelun kanssa. Anturit olivat edullisia ja tarkkoja, eikä niitä tarvinnut kalibroida erikseen, joka helpotti laitteen ohjelmointia. Koekytkentälevyn käyttö teki laitteiden liittämisestä helppoa ja mahdollisti virheellisten kytkentöjen korjaamisen helposti.

Kaiken kaikkiaan monitorointilaitteen varsinaisessa toteutuksessa onnistuttiin hyvin. Monitorointilaitteen valmistus päätettiin toteuttaa osissa, joka helpottaa lopullisen monitorointilaitteen muokkausta siten, että siinä käytetään vain toista anturia ympäröivän maailma tarkkailuun. Lisäksi monitorointilaitteen valmistaminen osissa helpottaa laitteen toiminnan ymmärtämistä, koska siinä on eroteltu kuhunkin anturiin ja laitteen toimintaan vaikuttavat seikat.

Lopullisen monitorointilaitteen testauksessa laite todettiin toimivaksi. Pitkäaikainen testaus ei kuitenkaan ole täysin kattava. Pitkäaikaisessa testaamisessa ei ilmennyt virheellisiä varoituksia, mutta testausaika oli lyhyt. Testaus olisi pitänyt aloittaa aikaisemmin ja testausajan olisi pitänyt olla ainakin viikko anturia kohden. Testaus olisi voitu toteuttaa liittämällä toinen anturi pois laitteesta, jonka jälkeen laitetta olisi testattu viikon verran ilman toista anturia. Sama olisi toistettu toiselle anturille, jolloin kunkin testin aikana tiedettäisiin, kumpi anturi oli lähettänyt virheellisen varoituksen. Pidempiaikaisella testaamisella olisi voitu paremmin todeta esimerkiksi anturien herkkyys häiriölle.

Monitorointilaitteen raja-arvojen ja toiminnollisuuteen vaikuttavien tekijöiden muokkaamisesta on tehty yksinkertaista. Esimerkiksi tarkkailtava lämpötila ja varoitusviestien välissä pidettävä tauko, mahdollistavat laitteen muokkaamisen käyttötarkoitukseen sopivaksi. Varoitusviestien välissä pidettävä tauko estää sähköpostin lähettämisen liian nopealla

aikavälillä ja sitä voidaan muokata käyttötarkoituksen mukaan. Monitorointilaitteeseen valitut menetelmät ja osat myös laajentavat monitorointilaitteen mahdollisia käyttötarkoituksia. Monessa suunnitteluvaiheessa tarkkaillussa menetelmä vaihtoehdossa oli vaatimuksena ulkoisen laitteen, kuten palvelimen tai tietokoneen käyttö osana varoitusviestin lähettämistä. Tässä opinnäytetyössä valmistuvaa monitorointilaitetta voidaan kuitenkin tarvittaessa käyttää ilman ulkoisia laitteita, joka mahdollistaa laitteen käytön esimerkiksi käyttötarkoituksissa, joissa laite on saatava pieneen tilaan tai ulkoista laitetta ei voida muusta syystä käyttää.

Opinnäytetyön kohderyhmä haluttiin pitää mahdollisimman laajana. Monitorointilaitteen valmistus haluttiin mahdollistaa kaikille, jotka ovat kiinnostuneita valmistamaan itse monitorointilaitteen käyttäen Arduinoa, riippumatta aiemmasta tietoteknisestä kokemuksesta. Opinnäytetyön lopullisen monitorointilaitteen pystyy valmistamaan opinnäytetyön perusteella, vaikka aiempaa tietoteknistä kokemusta ei paljoa olekaan. Laitteen toiminnan ja periaatteiden täysi ymmärtäminen vaatii kuitenkin aiempaa kokemusta mikrokontrollerilaitteiden valmistamisesta tai perehtymistä muihin lähteisiin. Kohderyhmän rajaus laajaksi tuotti myös ongelmia opinnäytetyön kirjoitusprosessin aikana, koska asioiden selittäminen, käytettävän käsitteistön ja teoria aiheiden käsittelyn syvyys oli vaikea rajata siten, että siinä on kaikki tarpeellinen tieto, mutta se on samalla käsitettävissä ilman suurempaa tietoteknistä kokemusta.

Opinnäytetyössä valmistunutta monitorointilaitetta voidaan kehittää monella tavalla. Monitorointilaitteen lähettämät virheelliset varoitusviestit voivat osoittautua laitteen ongelmaksi. Virheelliset varoitusviestit voivat olla hyvin kiusallisia, riippuen monitorointilaitteen käyttötarkoituksesta. Monitorointilaitteen virheherkkyyttä voidaan parantaa etenkin anturien osalta. Esimerkiksi lämpötila-anturin antamista arvoista voidaan ottaa keskiarvo määritellyltä aikaväliltä, jolloin yksittäinen tai muutama virhelukema ei välttämättä aiheuta varoitusviestin lähettämistä. Toinen käytännöllinen jatkokehitys vaihtoehto on toteuttaa varoitusviestin lähettäminen siten, että varoitusviestistä on mahdollista lähettää kaksi eri versiota, sen perusteella kumpi anturi käynnistää varoitusviestin lähettämisen. Varoitusviesteihin voidaan myös lisätä ominaisuus, joka liittää esimerkiksi lämpötilalukeman varoitusviestiin.

Valmistunutta monitorointilaitetta voidaan myös muokata käytettävien menetelmien osalta ja siihen voidaan lisätä toimintoja. Monitorointilaite voidaan esimerkiksi laittaa toimimaan palvelinten tarkkailuun suunnattujen ohjelmien kanssa, jolloin se toimii erinomaisesti palvelinhuoneen monitorointilaitteena. Toinen mahdollisuus on liittää monitorointilaitteeseen kamera tai käyttää tietokoneen webkameraa, jolloin monitorointilaite ja tietokone saadaan

lähettämään kuvaa alueelta, jossa monitorointi tapahtuu. Monissa jatkokehitys tilanteissa, joissa laitteeseen halutaan lisätä uusia toimintoja, täytyy varoitusviestin lähettämiseen käytettävää menetelmää vaihtaa, koska sen avulla voidaan lähettää vain tekstiä. Varoitusviestiä voidaan myös käyttää osana uutta laitetta, jonka rinnalle uusi toiminto lisätään.

Opinnäytetyön aihe oli entuudestaan tekijälle melko tuntematon, eikä aiempaa kokemusta ollut kuin Arduinon perusteista. Myös mikrokontrollereja käyttävien laitteiden valmistaminen oli uutta, eikä Haaga-Helian tietojenkäsittelyn koulutusohjelman pakollisessa kurssi suunnitelmassa ollut aiheeseen liittyviä kursseja. Ohjelmointikin oli suhteellisen tuntematon osa-alue opinnäytetyöntekijälle, koska tekijä oli suuntautunut järjestelmäasiantuntijaksi tietojenkäsittelyn koulutusohjelmassa. Etenkin C++ -ja C-ohjelmointikielet olivat tekijälle tuntemattomia, sillä tietojenkäsittelyn koulutusohjelman pakolliset ohjelmointiin liittyvät kurssit käsittelivät java-ohjelmointikieltä.

Opinnäytetyöprosessi syvensi tekijän tietämystä Arduino laitteista, niiden toiminnasta, ohjelmoinnista ja mikrokontrollereja käyttävien laitteiden valmistamisesta. Opinnäytetyössä valmistuvan laitteen osalta opinnäytetyön tekijä oppi ymmärtämään kehitysalustojen erilaisten liitäntöjen merkityksen ja ymmärtää nyt syvällisemmin syyt, miksi kutakin liitäntää käytetään ja miten liitännät toimivat teoriatasolla. Myös Arduino laitteissa käytettävien osien, kuten anturien ja kehitysalustojen ymmärrys parani. Etenkin varsinaisessa monitorointilaitteessa käytettyjen anturien ja kehitysalustan toiminnan ymmärtäminen parani. Esimerkiksi käytettyjen anturien toiminta periaatteet teoriatasolla tulivat selviksi, joka myös helpottaa anturien ja muiden kehitysalustaan liitettävien laitteiden käyttämistä käytännössä.

Arduino laitteiden ohjelmointitaidot kehittyivät työn edetessä ja sitä myöten myös yleiset ohjelmointitaidot kehittyivät. Tekijä oppi ymmärtämään paremmin eri funktioiden käyttötarkoitukset ja merkitykset, sekä muuttujien ja vakioiden tarpeellisuuden Arduino ohjelmoinnissa. Kyky ymmärtää ohjelmakoodin toimintaa ja etenemistä parani myös huomattavasti, joka ilmeni käytetyn ohjelmakoodin toiminnan selvittämistä kirjoitettaessa.

Lisäksi Arduinoon liittyvä käsitteistö ja lähteet tulivat tutuiksi opinnäytetyön tekijälle. Myös kyky selvittää Arduino laitteisiin liittyviä ongelmia parani, etenkin Arduinoon liittyvien lähteiden tuntemus helpottaa ongelmatilanteiden selvitystä. Arduinoon liittyvät lähteet olivat lähes kaikki englanninkielisiä, joten myös englannin kielen taidot paranivat etenkin Arduinoon liittyvien käsitteiden osalta. Englannin kielen oppimiseen vaikutti myös se, että varsinainen opinnäytetyö kirjoitettiin suomeksi, vaikka lähteet olivat englanniksi. Englannin

kieliset lähteet myös vaikeuttivat ajoittain opinnäytetyön kirjoittamisprosessia, koska joillekin Arduinon kanssa käytettäville käsitteille oli vaikeaa löytää suomenkielistä vastinetta.

Opinnäytetyön kirjoittamisprosessi paransi myös tekijän suomen kielen taitoa. Tekijän kirjallinen ilmaisutaito parani opinnäytetyön edetessä. Opinnäytetyön tekijän tietämys syveni myös kirjallisten raporttien kirjoittamisesta, esimerkiksi lähde viitteiden ja raporttien rakenteen osalta. Myös kyky tehdä ja selvittää erilaisia teoriatietoon perustuvia kirjoituksia kehittyi opinnäytetyön aikana.

Lähteet

Adafruit 2012. Overview. Luettavissa: https://learn.adafruit.com/tmp36-temperature-sensor. Luettu: 27.10.2014.

Adafruit 2014. How PIRs Work. Luettavissa: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work. Luettu: 27.10.2014.

Arduino a. What is Arduino? Luettavissa: http://www.arduino.cc/en/Guide/Introduction. Luettu: 21.10.2014.

Arduino b. PWM. Luettavissa: http://www.arduino.cc/en/Tutorial/PWM. Luettu: 23.10.2014.

Arduino c. Arduino Mega 2560. Luettavissa: http://arduino.cc/en/Main/ArduinoBoardMega2560. Luettu: 23.10.2014.

Arduino d. Guide to the Arduino Yún. Luettavissa: http://arduino.cc/en/Guide/ArduinoYun. Luettu: 22.10.2014.

Arduino e. Components. Luettavissa: http://playground.arduino.cc/Main/Components. Luettu: 26.10.2014.

Arduino f. Shields. Luettavissa: http://arduino.cc/en/Main/ArduinoShields. Luettu: 26.10.2014.

Arduino g. Arduino Yún. Luettavissa: http://arduino.cc/en/Main/ArduinoBoardYun. Luettu: 28.10.2014.

Arduino h. Arduino Development Enviroment. Luettavissa: http://arduino.cc/en/Guide/Environment. Luettu: 29.10.2014.

Arduino i. Download the Arduino Software. Luettavissa: http://arduino.cc/en/Main/Software. Luettu: 29.10.2014.

Arduino j. loop(). Luettavissa: http://arduino.cc/en/Reference/Loop. Luettu: 30.10.2014.

Arduino k. if (conditional) and ==, !=, <, > (comparison operators). Luettavissa: http://arduino.cc/en/Reference/If. Luettu: 30.10.2014.

Arduino l. if / else. Luettavissa: http://arduino.cc/en/Reference/Else. Luettu: 30.10.2014.

Arduino m. while loops. Luettavissa: http://arduino.cc/en/Reference/While. Luettu: 30.10.2014.

Arduino n. Variables. Luettavissa: http://arduino.cc/en/Tutorial/Variables. Luettu: 31.10.2014.

Arduino o. Libraries. Luettavissa: http://arduino.cc/en/Reference/Libraries. Luettu: 9.11.2014.

Arduino p. Hardware Index. Luettavissa: http://www.arduino.cc/en/Main/Boards. Luettu: 13.11.2014.

Arduino q. Language Reference. Luettavissa: http://arduino.cc/en/Reference/HomePage. Luettu: 13.10.2014.

Arduino r. Functions. Luettavissa: http://www.arduino.cc/en/Reference/FunctionDeclaration. Luettu: 13.11.2014.

Banzi, M. 2013. Send in the clones. Luettavissa: http://blog.arduino.cc/2013/07/10/send-in-the-clones/. Luettu: 21.10.2014.

Boxall, J. 2013. Arduino Tutorials – Chapter 22 – the AREF pin. Luettavissa: http://tronixstuff.com/2013/12/12/arduino-tutorials-chapter-22-aref-pin/. Luettu: 23.10.2014.

Castle, A. 2013. Know Your Arduino: A Practical Guide to The Most Common Boards. Luettavissa: http://www.tested.com/tech/robots/456466-know-your-arduino-guide-most-common-boards/. Luettu: 21.10.2014.

Fitzgerald, S. & Shiloh, M. 2013. Arduino projects book. Arduino. Torino.

Karvinen, T. & Karvinen, M. 2009. Sulautetut. Readme.fi. Helsinki.

Ladyada.net 2012. Sensors. Luettavissa: http://www.ladyada.net/learn/sensors/. Luettu: 24.10.2014.

Lahart, J. 2009. Taking an Open-Source Approach to Hardware. Luettavissa: http://online.wsj.com/news/articles/SB10001424052748703499404574559960271468066. Luettu: 21.10.2014.

Medea 2013. Ardiomp FAQ – With David Cuartilles. Luettavissa: http://medea.mah.se/2013/04/arduino-faq/. Luettu: 21.10.2014.

Society Of Robots. Introduction to microcontrollers. Luettavissa: http://www.societyofrobots.com/microcontroller_tutorial.shtml. Luettu: 23.10.2014.

Sparkfun. Analog vs. Digital. Luettavissa: https://learn.sparkfun.com/tutorials/analog-vs-digital. Luettu: 26.10.2014.

Temboo. Reprogramming Creativity, Re-Imagining Programming. Luettavissa: https://www.temboo.com/about. Luettu: 22.10.2014.

Wigmore, I. 2012. Sensor. Luettavissa: http://whatis.techtarget.com/definition/sensor. Luettu: 24.10.2014.

Liitteet

Liite 1. Muut laitteen valmistuksessa tarvittavat ohjelmointikomennot

Rakenteeseen liittyvät	
;	Käytetään lausekkeiden päättämiseen.
{}	Käytetään esimerkiksi if komentojen yhteydessä rajaamaan toiminnot jotka kuuluvat if komennon alaisuuteen.
//	Käytetään kommenttien lisäämiseen lähdekoodiin, mutta kommentit eivät suoraan vaikuta ohjelmakoodin toimintaan.
#define	Mahdollistaa vakioarvon nimeämisen ennen ohjelman kääntämistä.
#include	Käytetään ohjelmakirjastojen liittämiseen.
=	Tallettaa arvon yhtäläisyysmerkin oikealta puolelta muuttujaan, joka on yhtäläisyysmerkin vasemmalla puolella.
-	Käytetään laskuopillisissa lausekkeissa vähentämiseen.
*	Käytetään laskuopillisissa lausekkeissa kertomiseen.
/	Käytetään laskuopillisissa lausekkeissa jakamiseen.
==	Käytetään esimerkiksi if komentojen kanssa ja tarkoittaa jonkin olevan yhtä suuri kuin toinen.
>	Käytetään esimerkiksi if komentojen kanssa ja tarkoittaa jonkin olevan suurempi kuin toinen.
||	Käytetään esimerkiksi if komentojen kanssa ja käytetään kun if komennossa halutaan toisen kahdesta tarkkailtavasta arvosta olevan tosi.
!	Käytetään esimerkiksi if komentojen kanssa ja on tosi kun tarkkailtava arvo on eri kuin huutomerkin kanssa käytettävä arvo.
/=	Käytetään lyhentämään laskuopillisia lausekkeita sketchissä. Esimerkiksi x/=y; on sama kuin x=x/y;.
Muuttujat, vakiot ja tietotyypit	
INPUT	Käytetään silloin kun pinniä halutaan käyttää sisääntulona. Käytetään esimerkiksi sensorien tarkkailuun.

OUTPUT	Käytetään silloin kun pinniä halutaan käyttää ulostulona. Käytetään esimerkiksi LED-valojen kontrollointiin.
HIGH	Pinniä käytettäessä sisääntulona pinMode ja digitalRead komentojen kanssa mikrokontrolleri antaa arvon HIGH, mikäli pinni havaitsee yli kolmen voltin jännitteen. Jos pinni on asetettu ulostuloksi ja sille on asetettu HIGH arvo, se lähettää viiden voltin jännitteen.
LOW	Pinniä käytettäessä sisääntulona pinMode ja digitalRead komentojen kanssa mikrokontrolleri antaa arvon LOW, mikäli pinni havaitsee alle kahden voltin jännitteen. Jos pinni on asetettu ulostuloksi ja sille on asetettu LOW arvo, se lähettää nollan voltin jännitteen.
void	Käytetään funktioiden kanssa ja se ilmentää että funktion ei ole tarkoitus palauttaa arvoa sitä kutsuneelle funktiolle.
char	Tallentaa kirjaimia ja numeroita ASCII muodossa.
int	Tallettaa lukuja -32,768 ja 32,767 väliltä.
unsigned int	Tallettaa positiivisia lukuja 65,535 asti.
unsigned long	Tallettaa positiivisia lukuja 4,294,967,295 asti.
float	Käytetään lukujen kanssa jotka sisältävät desimaalipilkun.
String	Käytetään tekstiä sisältävien arvojen kanssa.
const	Ei ole varsinaisesti arvo, mutta constia käytettäessä, sen kanssa käytettävän muuttujan arvo ei voi muuttua.
Funktiot	
PinMode()	Funktion avulla voidaan määritellä pinni ulostuloksi tai sisääntuloksi.
digitalRead()	Lukee digitaalisen pinnin tilan, joka on joko HIGH tai LOW.
analogRead()	Lukee analogisen pinnin tilan, jonka jälkeen muuntaa saadun jännite arvon nollan sekä viiden voltin väliltä kokonaisluku arvoiksi nollan ja 1023 väliltä.
analogReference()	Asettaa jännitteen vertausarvon analogiselle sisääntulolle.
delay()	Pysäyttää ohjelman toiminnan millisekunteina määritellyksi ajaksi.

Serial	Käytetään kommunikoimiseen Arduino kehitysalustan ja muiden laitteiden välillä.
Serial.print()	Tulostaa tiedon sarjaportille luettavassa ASCII muodossa.
Serial.println()	Tulostaa tiedon sarjaportille luettavassa ASCII muodossa ja luo rivinvaihdon.
Serial.begin()	Asettaa tiedon siirtonopeuden sarjaportissa tapahtuvalle kommunikoinnille.
Bridge.begin()	Arduino Yúnissa luo siltauksen Atmega32U4 mikrokontrollerin ja AR9331 prosessorin välille. Funktion käyttö vaatii Bridge ohjelmakirjaston käyttämistä ohjelmointikoodissa.
Choreo.begin()	Käytetään Temboo choreoiden käynnistämiseen.

Liite 2. Lämpötilaa tarkkailevan laitteen ohjelmakoodi

```
const int tempsensorPin = A1;
const int ledPin = 13;
const float warningTemp = 28.01;
const float arefVoltage = 3.3;

void setup() {
  Serial.begin(9600);
  pinMode(tempsensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
  analogReference(EXTERNAL);

}
void loop() {
  int sensorVal = analogRead(tempsensorPin);
  Serial.print("Sensor Value: ");
  Serial.print(sensorVal);
  float voltage = sensorVal * arefVoltage;
  voltage /= 1024.0;
  Serial.print(", Volts: ");
  Serial.print(voltage);
  Serial.print(", degrees C: ");
  float temperature = (voltage - 0.5) * 100;
  Serial.println(temperature);
  delay(200);

  if (temperature > warningTemp) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

}
```

Liite 3. Liikettä tarkkailevan laitteen ohjelmakoodi

```
const int pirsensorPin = 2;
const int ledPin = 13;

void setup() {
  pinMode(pirsensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  if (digitalRead(pirsensorPin) == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Liite 4. Liikettä ja lämpötilaa tarkkailevan laitteen ohjelmakoodi

```
const int pirsensorPin = 2;
const int tempsensorPin = A1;
const int ledPin = 13;
const float warningTemp = 28.01;
const float arefVoltage = 3.3;

void setup() {
  Serial.begin(9600);
  pinMode(pirsensorPin, INPUT);
  pinMode(tempsensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
  analogReference(EXTERNAL);

}
void loop() {
  int sensorVal = analogRead(tempsensorPin);
  Serial.print("Sensor Value: ");
  Serial.print(sensorVal);
  float voltage = sensorVal * arefVoltage;
  voltage /= 1024.0;
  Serial.print(", Volts: ");
  Serial.print(voltage);
  Serial.print(", degrees C: ");
  float temperature = (voltage - .5) * 100;
  Serial.println(temperature);
  delay(200);

  if (digitalRead(pirsensorPin) == HIGH || temperature > warningTemp) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Liite 5. Temboon esimerkkiohjelman ohjelmakoodi

```
/*
  SendAnEmail

  Demonstrates sending an email via a Google Gmail account using the Temboo Arduino
Yun SDK.

  This example code is in the public domain.
*/
#include <Bridge.h>
#include <Temboo.h>
#include "TembooAccount.h" // contains Temboo account information
/*** SUBSTITUTE YOUR VALUES BELOW: ***/
// Note that for additional security and reusability, you could
// use #define statements to specify these values in a .h file.
// your Gmail username, formatted as a complete email address, eg
"bob.smith@gmail.com"
const String GMAIL_USER_NAME = "xxxxxxxxxx";
// your Gmail password
const String GMAIL_PASSWORD = "xxxxxxxxxx";
// the email address you want to send the email to, eg "jane.doe@temboo.com"
const String TO_EMAIL_ADDRESS = "xxxxxxxxxx";
boolean success = false; // a flag to indicate whether we've sent the email yet or not

void setup() {
  Serial.begin(9600);
  // for debugging, wait until a serial console is connected
  delay(4000);
  while (!Serial);
  Bridge.begin();
}

void loop()
{
  // only try to send the email if we haven't already sent it successfully
  if (!success) {
    Serial.println("Running SendAnEmail...");
```

```
TembooChoreo SendEmailChoreo;
// invoke the Temboo client
// NOTE that the client must be reinvoked, and repopulated with
// appropriate arguments, each time its run() method is called.
SendEmailChoreo.begin();
// set Temboo account credentials
SendEmailChoreo.setAccountName(TEMBOO_ACCOUNT);
SendEmailChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
SendEmailChoreo.setAppKey(TEMBOO_APP_KEY);
// identify the Temboo Library choreo to run (Google > Gmail > SendEmail)
SendEmailChoreo.setChoreo("/Library/Google/Gmail/SendEmail");
// set the required choreo inputs
// see https://www.temboo.com/library/Library/Google/Gmail/SendEmail/
// for complete details about the inputs for this Choreo
// the first input is your Gmail email address
SendEmailChoreo.addInput("Username", GMAIL_USER_NAME);
// next is your Gmail password.
SendEmailChoreo.addInput("Password", GMAIL_PASSWORD);
// who to send the email to
SendEmailChoreo.addInput("ToAddress", TO_EMAIL_ADDRESS);
// then a subject line
SendEmailChoreo.addInput("Subject", "ALERT: Greenhouse Temperature");
// next comes the message body, the main content of the email
SendEmailChoreo.addInput("MessageBody", "Hey! The greenhouse is too cold!");
// tell the Choreo to run and wait for the results. The
// return code (returnCode) will tell us whether the Temboo client
// was able to send our request to the Temboo servers
unsigned int returnCode = SendEmailChoreo.run();
// a return code of zero (0) means everything worked
if (returnCode == 0) {
  Serial.println("Success! Email sent!");
  success = true;
} else {
  // a non-zero return code means there was an error
  // read and print the error message
  while (SendEmailChoreo.available()) {
    char c = SendEmailChoreo.read();
    Serial.print(c);
```

```
      }
    }
    SendEmailChoreo.close();
    // do nothing for the next 60 seconds
    delay(60000);
  }
}
```

Liite 6. Temboon esimerkkiohjelmassa ja lopullisessa monitorointilaitteessa käytetty header-tiedosto

```
#define TEMBOO_ACCOUNT "xxxxxxxxx"  // your Temboo account name
#define TEMBOO_APP_KEY_NAME "myFirstApp"  // your Temboo app key name
#define TEMBOO_APP_KEY  "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"  // your Temboo app key
```

Liite 7. Lopullisen monitorointilaitteen ohjelmakoodi

```
const int pirsensorPin = 2;
const int tempsensorPin = A1;
const float warningTemp = 28.01;
const float arefVoltage = 3.3;
unsigned long seconds = 1000L;
unsigned long minutes = seconds * 60;
#include <Bridge.h>
#include <Temboo.h>
#include "TembooAccount.h"
const String GMAIL_USER_NAME = "xxxxxxx@gmail.com"; ////
const String GMAIL_PASSWORD = "xxxxxxxx";
const String TO_EMAIL_ADDRESS = "xxxxx@email.domain";

void setup() {
  Serial.begin(9600);
  Bridge.begin();
  pinMode(pirsensorPin, INPUT);
  pinMode(tempsensorPin, INPUT);
  analogReference(EXTERNAL);
}

void loop() {
  int sensorVal = analogRead(tempsensorPin);
  Serial.print("Sensor Value: ");
  Serial.print(sensorVal);
  float voltage = sensorVal * arefVoltage;
  voltage /= 1024.0;
  Serial.print(", Volts: ");
  Serial.print(voltage);
  Serial.print(", degrees C: ");
  float temperature = (voltage - .5) * 100;
  Serial.println(temperature);
  delay(200);

  if (digitalRead(pirsensorPin) == HIGH || temperature > warningTemp) {
    Serial.println("Running SendAnEmail...");
```

```
    TembooChoreo SendEmailChoreo;
    SendEmailChoreo.begin();
    SendEmailChoreo.setAccountName(TEMBOO_ACCOUNT);
    SendEmailChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    SendEmailChoreo.setAppKey(TEMBOO_APP_KEY);
    SendEmailChoreo.setChoreo("/Library/Google/Gmail/SendEmail");
    SendEmailChoreo.addInput("Username", GMAIL_USER_NAME);
    SendEmailChoreo.addInput("Password", GMAIL_PASSWORD);
    SendEmailChoreo.addInput("ToAddress", TO_EMAIL_ADDRESS);
    SendEmailChoreo.addInput("Subject", "Subject for the warning");
    SendEmailChoreo.addInput("MessageBody", "Message body for the warning");
    unsigned int returnCode = SendEmailChoreo.run();

    if (returnCode == 0) {
      Serial.println("Success! Email sent!");
      Serial.println("Monitoring continues aprroximately  in 15 minutes");
      delay(15 * minutes);

    } else {
      while (SendEmailChoreo.available()) {
        char c = SendEmailChoreo.read();
        Serial.print(c);
      }
      delay(20 * seconds);
    }
    SendEmailChoreo.close();
  }
}
```