Lämpötilaa mittaava Internet of Things -rele

Santeri Valkama

Ohjaaja: Juha Häkkinen

SÄHKÖTEKNIIKAN TUTKINTO -OHJELMA

2016

Sisällysluettelo

Tiivistelmä 3
Abstract 4
1. Johdanto 5
2. Digistump Oak ja IoT -ohjelmistoa 6
2.1 Yleistä 6
2.2 Digistump Oak- kehitysalusta 6
2.3 Moderneja IoT- ohjelmia 7
2.4 Käytettävien ohjelmien valinta 8
3. Lämpötilaa mittaavan IoT- releen tekeminen 10
3.1 Arduino -kehitysympäristön ja Oakin alustus 10
3.2 Tulevan ja lähtevän yhteyden muodostaminen Oakille 10
3.3 Oakin rele ja lämpösensori 14
3.4 IoT -verkon laajentaminen IFTTT:llä ja Android -käyttöliittymällä 17
3.5 Työn lopputulos 21
4 Pohdinta 22
4.1 Jälkimietteitä 22
4.2 Kehitys- ja jatkomahdollisuudet 22
5 Yhteenveto 23
Lähteet 24
Liitteet 26

Tiivistelmä

Tässä kandidaatintyössä esitetään internetin välityksellä ohjattavan ja releellä ja lämpösensorilla varustetun Internet of Things -yksikön rakentaminen Digistump Oak- kehitysalustalle ja todistetaan sen toimivuus. Työssä esitetään myös muutamia tapoja todeta internet -pyyntöjen toimivuus.

Lisäksi tarkastellaan Internet of Things -tarkoituksiin soveltuvia ohjelmistoja työhön sopivuuden näkökulmasta.

Asiasanat: Internet of Things, Digistump Oak, REST API

Abstract

The purpose of this bachelor's work is to present and prove functionality of a basic Internet of Things unit built on the Digistump Oak -development board capable of connections through internet and equipped with a relay and a temperature sensor. A couple of ways to prove the functionality of web requests are also presented.

In addition, a few Internet of Things -applicable software were reviewed regarding suitability in this work.

Keywords: Internet of Things, Digistump Oak, REST API

1. Johdanto

Internet of Things terminä kuulostaa tulevaisuuden järjestelmältä, jonka yritykset ovat tuoneet tavallisten kuluttajien tietoon viime vuosina varsinkin tulevaisuuden mieleen tuovan älykodin muodossa. Vaikka käsite kuulostaa oudolta, määritelmä IoT:lle on yksinkertaisesti sähköiset fyysiset esineet, jotka ovat yhteydessä toisiinsa ja kykenevät vaihtamaan tietoa keskenään itsenäisesti[1]. Tähän määritelmään sopii vaikkapa nykyaikainen automatisoitu tuotantolinja, jossa on useita toisistaan riippuvia vaiheita sekä erilaisia robotteja, jotka taas kommunikoivat konenäköjensä kanssa. Ajotietokoneella varustettu auto, jossa useissa osissa on sensori, täyttää kriteerit. Myös kodin LAN ja erityisesti WLAN -verkko ovat hyviä kaikille tuttuja esimerkkejä IoT -verkosta ja verkkoon kytkettävä tulostin IoT -laitteesta. Tulostin tulostaa kun se saa käskyn verkossa olevalta tietokoneelta ja lähettää tietokoneelle ilmoituksen vähäisestä musteesta tai loppuneesta paperista omien sensorien tai käytönaikaisen diagnosoinnin ongelman huomatessa, mikä helpottaa hieman käyttäjän elämää vähentämällä omatoimisen ongelmanselvittelyn tarvetta.

Työn aiheen takana oleva motivaatio tulee käyttäjämukavuuden parantamisesta ja mahdollisesta sähkönkulutksen vähentämisestä puhelimella ohjattavan, tai automaattisesti toimivan releen avulla. Jos projektia vie pidemmälle, voi mukaan lisätä myös kalenteri- ja ajastustoimintoja IFTTT -palvelun tai laajemman koodin avulla(kuva 1). Lopputuloksena voisi silloin olla ohjelmoitava kytkin, jonka voi kytkeä releeseen, joka taas on kytkettävissä ohjaamaan vaikkapa auton lämmitystä talvisin. Tällaiselle IoT -laitteelle on helppo keksiä eri käyttötarkoituksia, jotka voivat helpottaa elämää.

2. Digistump Oak ja IoT -ohjelmistoa

2.1 Yleistä

Työn suorittamiseen tarvitaan useampi ohjelma ja laite. Alustava suunnitelma on tehdä käyttöliittymäksi Android -sovellus, jolla ohjataan Oak- kehitysalustaa käyttäen koodaukseen joko Digistumpin omaa Oakin kontrollointisovellusta, MIT:n App inventor- ohjelmaa, tai Blynk inc. tekemää Blynk- sovellusta. Oakille täytyy tehdä sitten luoda API, jolla pystyy ohjaamaan tai lukemaan laitetta tarpeen mukaan mielivaltaisesti jatkokehityksen mahdollistamiseksi ja mahdollisesti myös muiden projektien pohjaksi, eli tarkoituksena on toteuttaa mahdollisimman universaali ratkaisu.

API eli application programmable interface on laitteiden välinen käyttöliittymä, ja toimii kuten tavallinen ihmisen ja laitteen välinen käyttöliittymä. API tehdään vastaanottamaan ja lähettämään vain halutut muuttujien arvot, joita on tarve lähettää tai vastaanottaa.

2.2 Digistump Oak- kehitysalusta

Työssä käytettävä kehitysalusta on Digistumpin luoma Oak -kehitysalusta, jonka ympärille loppu projektista rakentuu. Merkittäviä ominaisuuksia tällä alustalla on sisäänrakennettu WiFi -yhteys ja kehitysalustalle hyvä prosessorin suoritusteho. Rahoitusta kehitykseen Oak on saanut onnistuneesta Kickstarter -kampanjasta. Alusta on ominaisuuksiinsa nähden halpa. Lisäksi jokaisella alustalla on ilmainen salattu yhteys Particle Cloud -pilvipalveluun, jossa on monipuolisesti ominaisuuksia laitteen tai laitteiden kontrollointiin ja ohjelmointiin. Kehitysalustan ohjelmointiin voi käyttää laitteen omia kirjastoja, mutta lisäksi se on Arduino -yhteensopiva, joten kaikki Arduino -kirjastotkin ovat ohjelmoijan käytössä. Laitteen kehittäjän aikaisempien tuotteiden Spark -moduulit ovat suoraan yhteensopivia Oakin kanssa. Kehitysalustan RF- yksikölle Acornille haetaan lisäksi FCC ja CE hyväksyntää, jotka

edellyttävät laitteen tuottamien haitallisten ja häiritsevien radioaaltojen minimointia. [2]

Syy Oakin käyttöön muiden kehitysalustojen sijaan ovat sisäänrakennettu WiFi ja Particle Cloudin API -mahdollisuudet. Nämä asiat helpottavat projektin tärkeiden yhteyksien luomista.

2.3 Moderneja IoT- ohjelmia

Oak on nyt 2016 -keväällä melko uusi kehitysalusta. ensimmäiset laitteet toimitettiin ostajille vuoden 2015 loppupuolella, eli laitteen käytöstä ei ole vielä kovin paljon käyttäjäkokemuksia[3]. Myöskään kehitysalustaan liittyvät ohjelmistot eivät ole vielä valmiita, tai julkaistu ollenkaan.

Blynk

Blynk on Android- sovellus, joka lupailee helppoa kehitysalustojen kontrollointia pienellä vaivalla. Sovelluksella luodaan Android- sovelluksia lisäämällä kaikenlaisia virtuaalisia kytkimiä, mittareita ja näyttöjä sivuille, jotka liitetään virtuaalisesti kehitysalustaan tai asetetaan tekemään jotakin toimintoa perinteisen koodauksen sijaan. Ohjelma syntetisoi kaikki luodut rakennuspalikat koodiksi uuteen ohjelmaan ja hoitaa yhteyden haluttuun laitteeseen automaattisesti. Ohjelma tukee monia kehitysalustoja, mutta Oak ei ole yksi niistä. Yhteensopivuus Oakin kanssa on kyllä suunniteltujen Blynk- päivitysten listalla, mutta tässäkään tapauksessa mitään päivämääriä tai tietoja päivityksen julkaisemisesta ei ole ilmoitettu. Onnistuneita yhteensopivuuden sijaisratkaisujakaan ei ole tiedossa. [4,5]

App Inventor

App Inventor on selaimessa toimiva MIT:n kehittämä Android- koodausohjelma, joka pyrkii tekemään koodauksesta helpompaa. Ohjelmassa käytetään palapelin palasiksi visualisoituja funktioita ja niiden argumentteja. Palaset on tehty sopimaan vain sellaisiin toisiin palasiin, joihin ne tavallisessakin koodissa sopisivat. Kyseessä on siis uusi tapa koodata perinteisen kirjoittamisen sijaan. Ohjelmasta löytyy kategorisoituna kaikki funktiot ja muut palat mitä koodissa voi tarvita, niin back- kuin frontend- koodiinkin. Ohjelma ei siis tunnu olevan mikään laimennettu versio perinteisestä koodin kirjoituksesta, vaan käyttäjäystävällisempi ja nopeampikin tapa koodata, koska kirjoittamisen tarvetta ei juurikaan ole. Myöskään koodikielen dokumentointeja ei tarvitse selata etsien jotakin tiettyä funktiota, kun kaikki palaset ovat jo valmiina hyvin kategorisoituna ohjelmointiruudun sivussa, ja antavat käyttövinkkejä helposti tarvittaessa. Myös itselle uudet koodikielen ja funktioiden ominaisuudet saa helpommin ja virhevapaammin toimimaan, koska yhteensopimattomat koodin osat on hankalampaa laittaa yhteen. Tämä tekee koodauksesta myös erittäin aloittelijaystävällistä, hukkaamatta kuitenkaan koodauskielen käytettävyyttä.[6]

IFTTT

Lisäykseksi työhön voi ottaa IoT- palvelu IFTTT:n, joka nimensä "If this then that" mukaan tekee jotakin, kun jokin määritelty tapahtuma tapahtuu. Palvelulla on paljon yhteistyökumppaneita, joiden omat palvelut ovat käytettävissä ja automatisoitavissa laukaisemaan tapahtumia tai olemaan tapahtuman kohteina IFTTT:ssä. Palvelua voi hyödyntää työssä käyttämällä palvelun kalenteritoimintoa. Kalenteriin voidaan määrittää ajat, jolloin rele halutaan pitää päällä.[7]

2.4 Käytettävien ohjelmien valinta

App inventorissa ei ole työtä helpottavia erityis- tai valmisominaisuuksia IoT -yhteyksille kuten muissa mainituissa ohjelmissa. App inventor on suunniteltu yleiseen koodaukseen eikä erikoistu IoT -yhteyksiin. Internet -yhteydet ovat

mahdollisia, joten App inventor on täysin työhön sopiva vaihtoehto, sillä Oakille tehtävä API on käytettävissä niiden avulla. App Inventor on hyvä valinta, jos tulee tarve laajentaa Androidille tulevaa osaa työstä esimerkiksi hyödyntämään joitakin puhelimen ominaisuuksia, kuten GPS -paikannusta.

Toinen mahdollisuus on luoda ohjelma Blynkillä ja yrittää hyödyntää Oakin Arduino -yhteensopivuutta kokeilemalla muiden kehitysalustojen ohjelman rakentamista Blynkissä. Jotkin Oakin ominaisuudet voivat jäädä toimimattomiksi laitteiden fyysisten eroavaisuuksien takia, jos tavalla edes saa yhteyden.

Työssä tullaan käyttämään Android -käyttöliittymän koodaukseen App Inventoria, muiden vaihtoehtojen kärsiessä yhteensopimattomuudesta tai tuntemattomista julkaisuajankohdista. Oakin koodi kirjoitetaan Arduinon kehitysympäristössä. IFTTT otetaan mukaan kalenteritoimintoa varten.

3. Lämpötilaa mittaavan IoT- releen tekeminen

3.1 Arduino -kehitysympäristön ja Oakin alustus

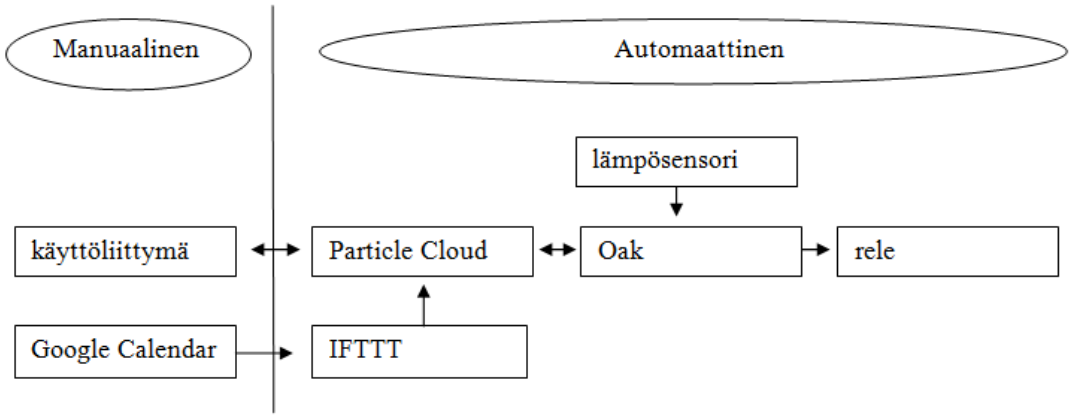
Ohjeiden mukaan Oakin yhteys Particleen muodostetaan yhdistämällä jokin toinen laite Oakin luomaan WiFi- verkkoon, ja määrittämällä tätä kautta Oakille WiFi -verkko ja sen salasana, jota kautta se pääsee yhdistymään verkkoon ja Particle Cloudiin. Tämä onnistuu menemällä osoitteeseen http://digistu.mp/oakconfig yhdistyneenä Oakin verkkoon. Sivun ohjeita seuraten Particle -tilin ja WiFi –verkon määrityksen jälkeen Oak lataa päivitykset ja on tämän jälkeen käyttövalmis.[8]

Koodi kirjoitetaan tavallisessa Arduinon kehitysympäristössä Oakin yhteensopivuuspaketti asennettuna[9]. Oakin paketti vaati vielä Particle- tilin tiedot, johon kehitysalusta on yhdistetty ja yhteydessä, jonka määrittämiseksi asennetaan Oak-cli ohjelma määrittämään halutun kehitysalustan tiedot kehitysympäristölle koodin lähettämistä varten[9]. Tämän jälkeen on hyvä testata kaiken toimivuus valitsemalla kehitysympäristöstön valmis testikoodi, vaikkapa Oakin sisäänrakennettua LED -valoa vilkuttava koodi ja lähettää se Oakille. Kun Oakin LED -valo vilkkuu koodissa määritettyyn tahtiin, Oak on saanut koodin onnistuneesti.

3.2 Tulevan ja lähtevän yhteyden muodostaminen Oakille

Vaadittavat yhteydet

Verkon kautta tulevia yhteyksiä tarvitaan kaksi, yksi käyttöliittymältä Oakille ja takaisin ja toinen kalenterilta Oakille IFTTT:n välityksellä(kuva 1). Kaikki Oakin ulkopuolelta tuleva ja lähtevä tieto ei mene suoraan Oakille vaan kulkee Particle Cloudin kautta, vaikkakin lähetettävät ja vastaanotettavat tiedot määritetään Oakin koodissa. Käytännössä Oakin koodia kirjoittaessa Particle Cloudin välissä oloa ei tarvitse miettiä.



Kuva 1. Vaikutuskaavio, jossa esitettynä tiedon liikkumissuunnat ja kaikki järjestelmän osat.

Oakin tunnus ja salasana

Ensimmäinen vaihe yhteyden muodostamisessa on selvittää Oakin tunnus ja salasanana toimiva token. Molemmat löytyvät kirjautumalla sisälle Particlen kehitysympäristöön [10]. Oakin laitekohtainen tunnus löytyy Devices- valikosta ja token on settings valikon alla. Molemmat tarvitaan tietojen vastaanottamiselle tai lähettämiselle internetin välityksellä Particle Cloudin kautta Oakille.

Oakin testikoodi

Ennen yhteyksien muodostamista on hyvä olla koodi, jonka avulla on helppo nähdä, saavuttiko viesti määränpäänsä. Oakin sisäänrakennetun LED -valon vilkuttaminen on hyvä väline tähän tarkoitukseen, ja onnistuu yksinkertaisesti kuten muidenkin pinnien kohdalla alustamalla ledin pinni ulostuloksi funktiolla pinMode(1, OUTPUT); jonka jälkeen pinnin tilan voi muuttaa vaikkapa loogiseksi ykköseksi tai nollaksi komennoilla digitalWrite(1,HIGH); tai digitalWrite(1,LOW);. Testikoodiin siis tarvitaan muuttuja, jonka muuttumista voi seurata ledin vilkkumisesta, ja kyseiseen muuttujaan liitetyt valmiit funktiot tarvittavien tietojen välittämiseksi Particle cloudiin. Aikaisemman testikoodin delay() -funktioiden argumentiksi eli käytännössä viiveen pituudeksi laitetaan muuttuja, jota voi lukea tai kirjoittaa ulkoisesti. (liite 1)

Particle.function

Funktiolla Particle.function("stoppause", funcstop); Oak voi vastaanottaa tietoja. Tässä tapauksessa stoppause on nimi, johon lähetettäessä käskyä viitataan ja funcstop() on Oakin koodiin tuleva käynnistettävä funktio, johon saa mukaan funktion käynnistävän paketin mukana tulleen argumentin määrittämällä sen tyypin ja käyttämällä arg- muuttujaa argumenttina kuten funcstop(String arg).(liite 1)[11]

Particle.variable

Funktio Particle.variable("outtest", pause); palauttaa muuttujan arvon. Tässä tapauksessa Oakin pause- muuttujan arvo lähetetään takaisin pyynnön tehneelle laitteelle. Lähetettävälle muuttujalle ei tarvitse tehdä mitään erityistä, kunhan se vaan on jossain kohti koodia määriteltynä. Oak hoitaa keskeytykset automaattisesti ja lähettää muuttujan arvon sillä hetkellä, kun se saa pyynnön tehdä niin. Kuten aiemminkin, funktio käynnistetään viittauksella outtest.(liite 1)[11]

API

API muodostuu edellä mainituista funktioista. Particle.function -funktio toimii virtuaalisena inputtina ja Particle.variable -funktion toimii virtuaalisena outputtina. Yhteyksien testausvaiheessa molemmat funktiot laitetaan viittaamaan samaan muuttujaan, joka laitetaan argumentiksi delay() -funktioihin, jotka määrittävät ledin vilkkumistahdin.(liite 1)

URL- pyynnön formaatti

Particle Cloud käyttää REST API -tyyppistä formaattia vastaanottaessaan Oakille menevät käskyt. Ensimmäisen testin voi tehdä jo selaimen URL -kentällä formaatilla ”https://api.particle.io/v1/devices/d957040012b83a8b0c878e80/outtest?access_token =asdfghjkl123, jossa ”https://api.particle.io/v1/devices/” on yleinen viittaus Particle cloudiin, joka on mukana jokaisessa viittauksessa sen alaisuudessa toimivan laitteen kanssa tapahtuvassa kommunikaatiossa. d957 -alkuinen tunnus on laitekohtainen,

jolla määritetään se tietty laite, jolle viesti on menossa. Kuten aiemmissa kappaleissa, outtest on kutsuttavan funktion nimi. ?access_token= on salasana, johon tulee Particlesta selvitetty token. Kyseinen URL -osoite käynnistää siis ”outtest” nimisen Particle.variable funktion Oakilla, ja palauttaa funktion määrittämän muuttujan arvon JSON -formaatissa muiden laitteen tietojen seassa.[12]

REST API -pyyntöjen metodeja

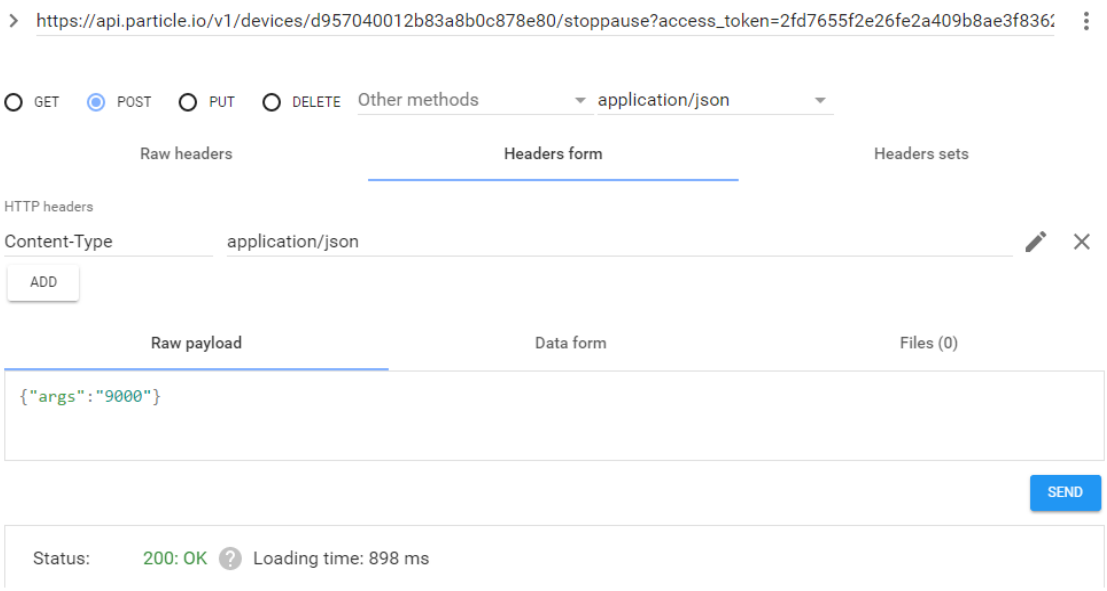
REST API käyttää metodeja, kuten tässä työssä käytettävät GET tai POST, pyyntöjen tarkoituksen määrittämiseksi. URL -kenttään metodia ei voi lisätä, vaan pyyntö tulkataan automaattisesti GET -tyyppiseksi. GET metodi nimensä mukaan palauttaa jonkin muuttujan tai muuttujia. POST on myös nimensä mukaan muuttujan lähettämistä varten.[13]

REST API- pyyntöjen koostumus

REST API -pyyntöjä varten tarvitaan ohjelma, joka pystyy kokoamaan ja lähettämään pyynnöt. Tässä projektissa on käytetty Advanced Rest Client -lisäosaa Google Chromelle, josta löytyvät kaikki tarvittavat kentät pyyntöjen toteuttamiseen: kenttä URL -osoitteelle, johon tulevat samat asiat kuin pelkässä URL -pyynnössä, tietoa lähettäviä metodeja varten kenttä lähetettävälle tiedolle, eli lastille ja sen tyypille, joka on tässä projektissa aina JSON, sekä metodivalikko.

REST API- pyyntöjen testaus

On helpompaa testata ja hioa pyynnöt toimiviksi kuvassa 2 olevalla Advanced Rest Clientillä ennen kuin sisällyttää ne koodiin.



Kuva 2. Onnistunut POST -pyyntö, jossa lastina luku 9000. Käytetty URL näkyy yläkulmassa.

Pyyntö on onnistuneesti mennyt perille, kun palautettu tila on 200. GET- metodin tapauksessa palautteena tulee myös pyydetty muuttuja.

3.3 Oakin rele ja lämpösensori

Oakin pinnit

Oakilla on useita pinnejä, joita voi käyttää johonkin erityiseen käyttötarkoitukseen, tai vaihtoehtoisesti tavallisina ohjattavina input tai analogisina tai digitaalisina output pinneinä. Ainut projektissa tarvittava erityispinni on A0, joka kykenee mittaamaan jännitteen ja palauttamaan arvon väliltä 0-1024, jossa 1024 vastaa käyttöjännitettä eli 3.3 volttia. P5 antaa muita pinnejä alhaisemman jännitteen, mitä ei tämän projektin tarkoituksiin kannata käyttää.[14]

Lämpösensorin kytkeminen

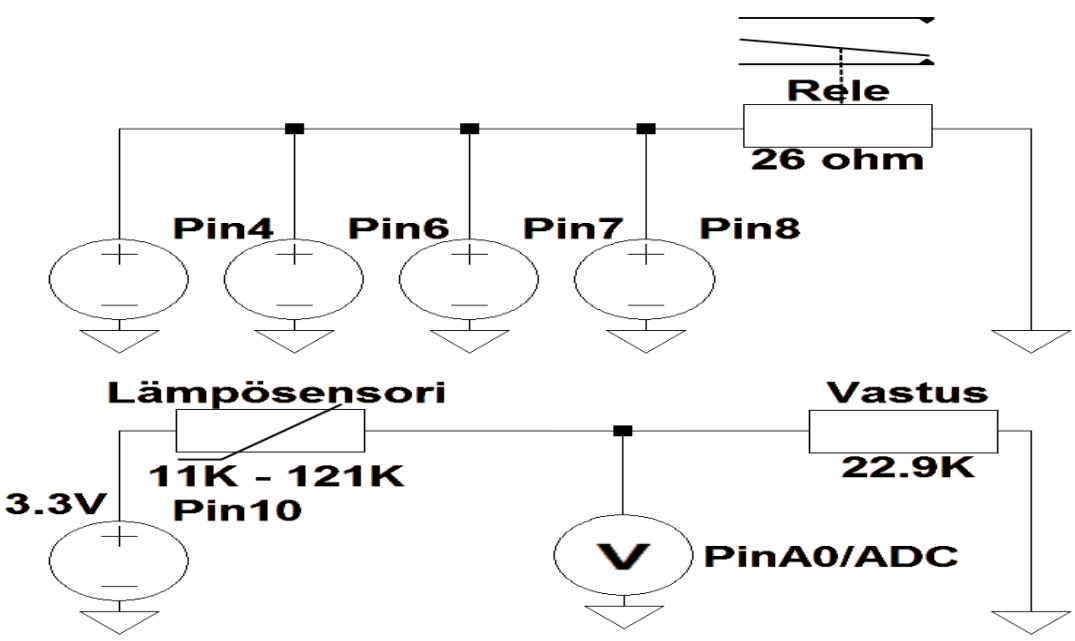
Lämpösensorin resistanssi muuttuu lämmön mukaan, mutta mittapinni A0 tarvitsee jännitteen mitattavaksi, joten on käytettävä jännitejakoa resistanssin jännitteeksi muuttamiseksi. Jännitejakoa varten liitetään vastus käyttöjännitteestä tai muusta 3.3 voltin pinnistä mittapinniin ja sensori mittapinnistä maahan tai muuhun nollan voltin

pinniin. Komponenttien paikat voi vaihtaa keskenään, jolloin saa käänteisen lukeman, kuten tässä projektissa on tehty (kuva 3).

Releen kytkeminen

Oakin pohjana olevan ESP8266EX kehitysalustan I/O pinnit datalehden mukaan antavat 12mA virtaa, mutta pystyvät ottamaan 20mA[15]. Kolmelle voltille tarkoitettu rele sisältää 25 ohmin kelan ja vaatii minimissään 0.27 wattia tehoa[16]. Käyttöjännitepinnit antavat tarpeeksi tehoa, mutta ne eivät ole ohjattavissa. Vaihtoehtoina on käyttää transistoria kytkimenä käyttöjännitteen ja releen välillä, tai ottaa käyttöön useampi ohjattava I/O pinni. Useamman pinnin käyttö on kuitenkin yksinkertaisempi vaihtoehto(kuva 3).

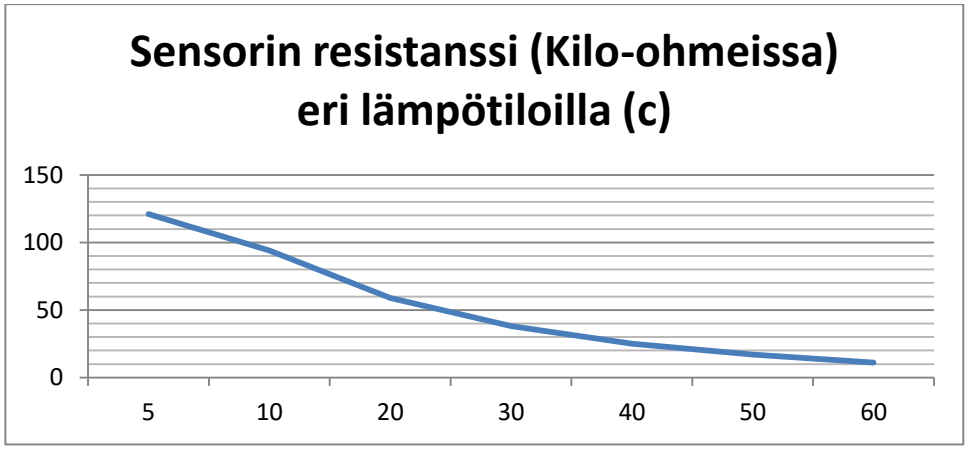
Käytetyn releen kelan resistanssi oli 26 ohmia. Yleismittarimittausten mukaan Oakin I/O pinnit pystyvät antamaan enemmän kuin tavalliset 12mA, tosin matalammalla jännitteellä. Jo neljällä pinnillä jännitettä releen yli saadaan 2.53 volttia, mikä tarkoittaa 26 ohmin kelan kanssa 97mA ja 0.245 wattia, joka testien mukaan riittää releen aktivoimiseen. Releen tila testataan johtavuuden varmistamiseksi myös yleismittarilla.



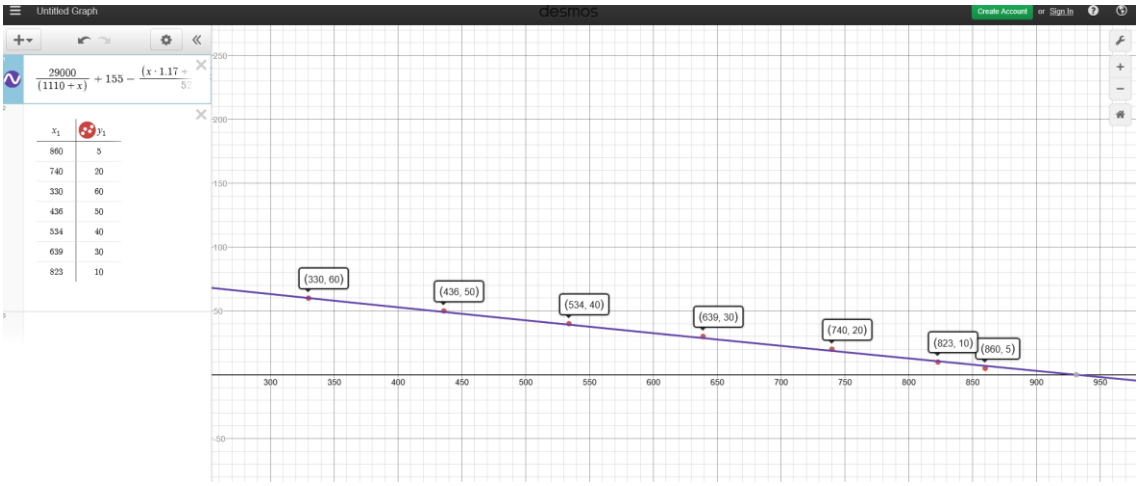
Kuva 3. Releen ja lämpösensorin kytkentäkaaviot.

Lämpösensorin koodi

Lämpösensorina toimii lattialämmityksen termostaatin sensori, jonka lämpötila-resistanssikäyrä ei ole lineaarinen. Laaditaan funktio datalehdessä olevien pisteiden mukaiseksi muuttamaan pinnin mittaama arvo ihmisen ymmärtämäksi lämpötilaksi(kuva 4, kuva 5)[17,18].



Kuva 4. Mitta alue on välillä 5 - 60 celsiusta, ja resistanssi välillä 121 - 11 Kilo-ohmia.



Kuva 5. Sensorin jännitejaon tulokset lämpötilaksi muuttava funktio seuraa annettuja pisteitä.

Käytettään funktiota 29000/(1110+x)+155-(x*1.17+1110)*4/52=y, jossa x on mittapinnin lukema ja y lämpötila celsiuksissa. Muunnosfunktio vääristää tuloksia hiukan, heittäen maksimissaan kaksi astetta viiden asteen lämpötilassa. Projektin

tarkoitus ei ole tarkka lämpömittari, joten muunnosfunktio on tarpeeksi hyvä tarkoitukseensa. Muunnosfunktio on vain ihmisen ymmärtämää lukemaa varten, joten koodissa käytetään mittapinnin antamaa vääristymätöntä arvoa sen sijaan. Tämän järjestelyn antamat tulokset olivat tarpeeksi lähellä todellisia arvoja, joten kalibrointia ei tarvitse suorittaa.(liite 2)

Releen koodi

Rele on hyödytön, jos sitä ei pysty ohjaamaan. Ohjauksen voi tehdä vaikkapa if -rakenteella, jossa argumenttina POST -pyyntöjen argumentilla hallittava muuttuja etäohjausta varten päälle ja pois kytkemiseksi. Koska rele tarvitsee neljä pinniä virtaa varten, käytettyjen pinnien tilan pitää vaihtua yhdessä.(liite 2)

Lämpötilaohjattu rele

Kun molemmat komponentin ovat kytketty, ne laitetaan toimimaan yhdessä. Laitetaan rele menemään päälle kun lämpötila nousee yli 30 asteen, joka on 639 mittapinnin antamana lukemana. Vain yhden rajapisteen ohjelmaa ei kannata laittaa tarkastamaan lukemaa lyhyin väliajoin, sillä se voi johtaa releen jatkuvaan tilanvaihteluun, varsinkin jos rele ohjaa jotakin lämpötilaan vaikuttavaa konetta. (liite 2)

3.4 IoT -verkon laajentaminen IFTTT:llä ja Android -käyttöliittymällä

Hyödyllisiä IFTTT kanavia

IFTTT:llä on monia IoT -aiheisia kanavia, joista ainakin kolme sopii hyvin projektiin. Maker -kanava mahdollistaa reseptien aktivoinnin internet -pyynnöillä, ja antaa käyttöön samantyylisen kaavakkeen pyyntöjen tekemiseen kuin Advanced Rest Client. Google Calendar -kanavalla saa aktivoitua reseptejä omien kalenterimerkintöjen avulla, kunhan liittää Google -tilin kanavaan kalenterin käyttöoikeuksia varten. Particle on myös projektille luonteva kanava, tarjoten valmiit pyynnöt Particle Cloudiin, jotka säästävät käyttäjän tavallisten pyyntöjen luomiselta.

IFTTT Do Button

Projektissa käytettävällä IFTTT:llä on myös Android- sovelluksia, joista yksi on IFTTT Do Button, joka aktivoi valitun IFTTT:n reseptin käyttäjän painalluksesta. Yksinkertainen ohjelma, joka on täydellinen testailuun, sillä reseptejä aktivoivia tekijöitä ei tarvitse keinotekoisesti järjestää Do Buttonia käytettäessä. Do Buttonilla voi testata vaikkapa pyynnöt, jotka todettiin toimiviksi Advanced Rest Clientissä valitsemalla aktivoitavaksi kanavaksi Maker -kanavan ja syöttämällä tiedot ruutuihin.

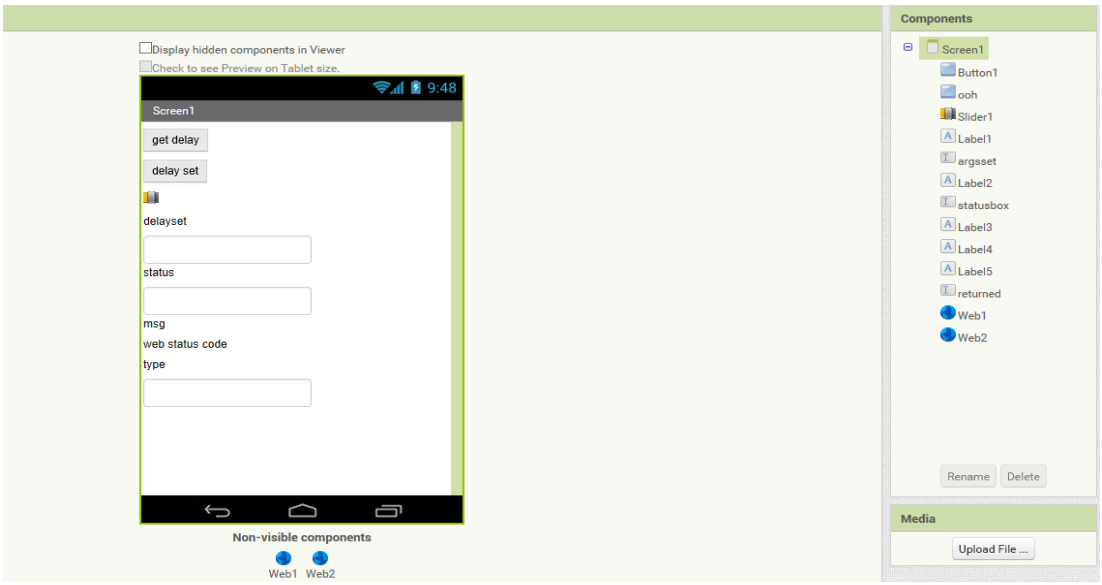
Kalenteriohjaus releelle

Koska releelle on jo tehty POST- pyynnöllä vaihdettavat tilat, enää tarvitsee vain lisätä IFTTT:ssä Google Calendar -kanava aktivoimaan joko Maker tai Particle kanava vaihtamaan Oakin tilan. Tällaisia reseptejä tarvitaan kaksi, toinen päälle ja toinen pois päältä laittamiseen. Google Calendar -kanavan aktivoivista tapahtumista sopivin on reseptin aktivointi 15 minuuttia ennen kalenterin tapahtuman alkua, jonka nimestä tai sisällöstä löytyy määritetty hakusana. Hakusanoiksi asetetaan ”relon” ja ”reloff” ja samoin tapahtumien nimiin tai sisältöihin. IFTTT ei tarkasta reseptien aktivointikriteerejä kovin usein, mahdollisesti kerran viidessä tai kymmenessä minuutissa, joten vaikkapa minuutin päähän testausmielessä asetetut aktivointitapahtumat todennäköisesti eivät tule toimimaan. Maker- tai Particle -kanavan pyynnöt ovat kuitenkin tässä vaiheessa jo testattu, joten niiden toimivuus varmistuu kopioimalla ja liittämällä edellisestä testikohteesta. Vastaanottavan Oakin pitää olla pyyntöjen lähetyksen aikana päällä, tai muuten reseptit eivät anna edes virheilmoitusta pyynnön matkan katkeamisesta.

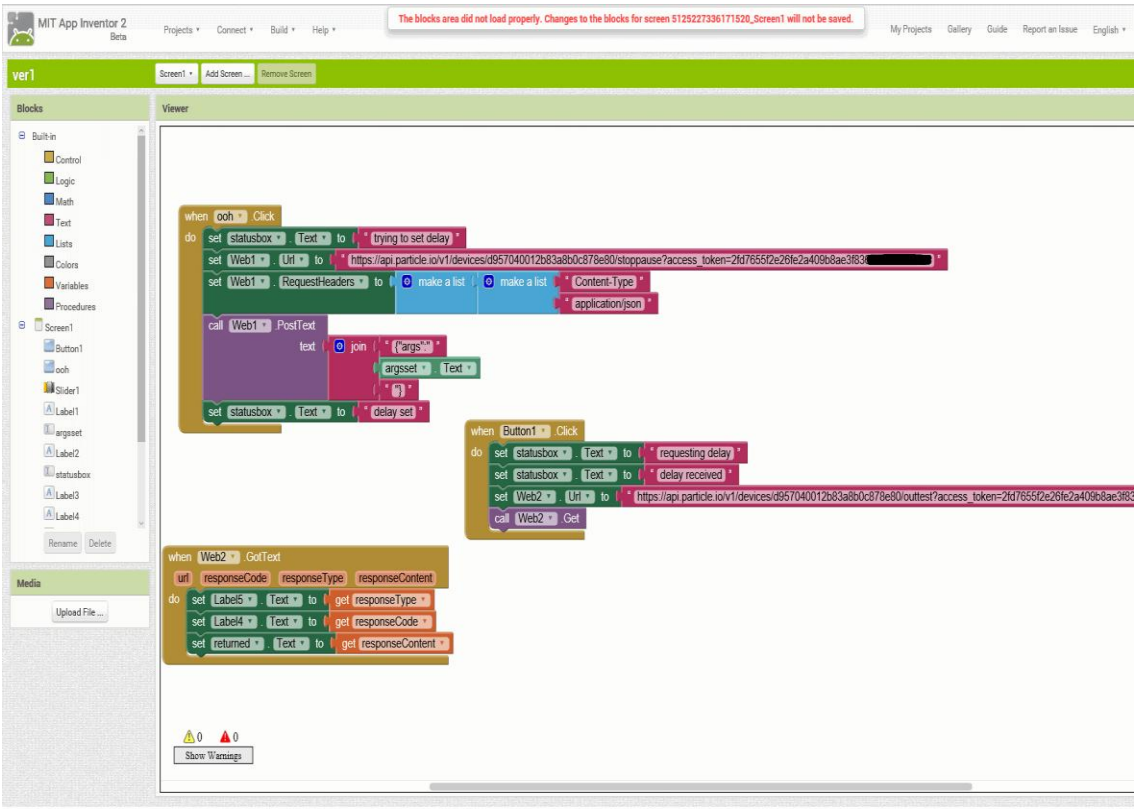
Android -käyttöliittymä App Inventorilla

Kaiken tarvittavan voisi tehdä erillisillä Do Buttoneilla, mutta kätevämpää ja käyttäjäystävällisempää on tehdä yhdelle sivulle puhelimen ruutuun mahtuva käyttöliittymä. Android -sovellukset onnistuvat helposti käyttöjärjestelmän koodia tuntemattomaltakin App Inventorilla. Aluksi vedetään näytölle käyttäjälle näkyvät nappulat ja tekstikentät, sekä verkkoelementit yhteyksiä varten(kuva 6). Sitten

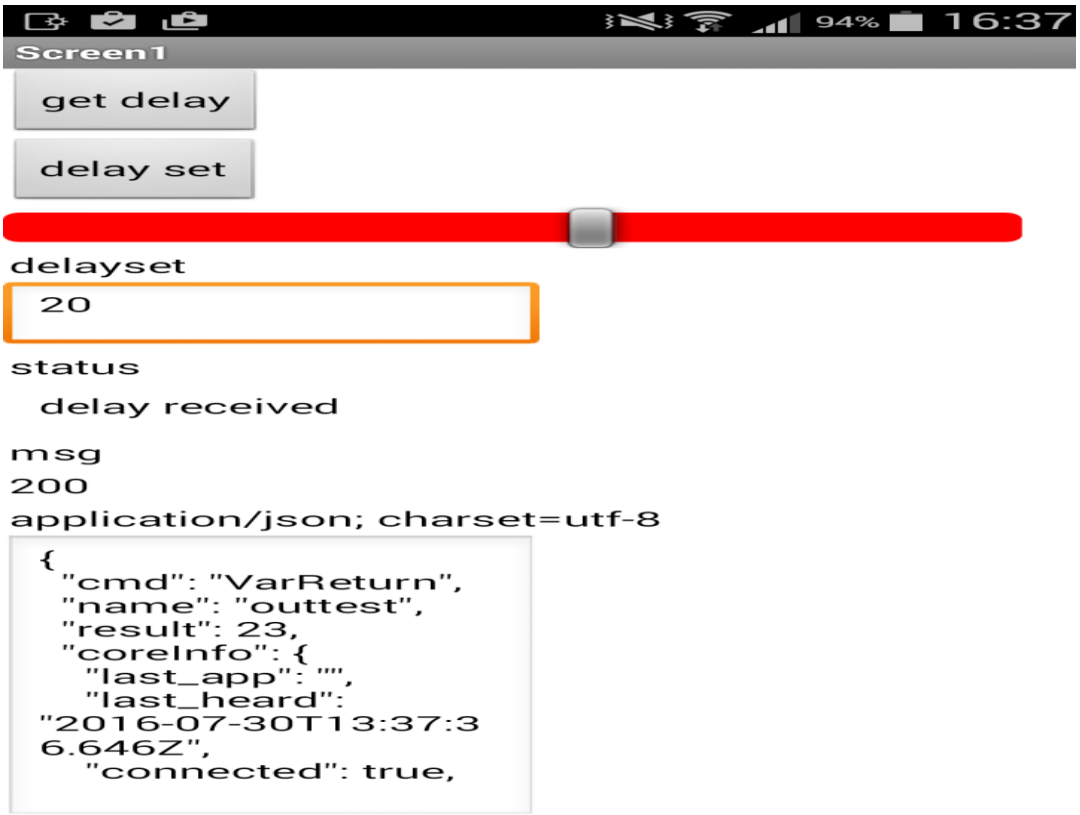
siirrytään koodin backendiin ja luodaan tarvittavat pyynnöt, jotka yhdistetään aktivoiviin nappuloihin ja tekstikenttiin palauttamaan käyttäjälle tarpeellista tietoa, kuten GET- pyynnön noutama muuttuja(kuva 7).Sitten ladataan ja asennetaan koodin apk -tiedosto Androidille ja ohjelma on käyttövalmis(kuva 8).



Kuva 6. Frontend -koodi App Inventorissa.



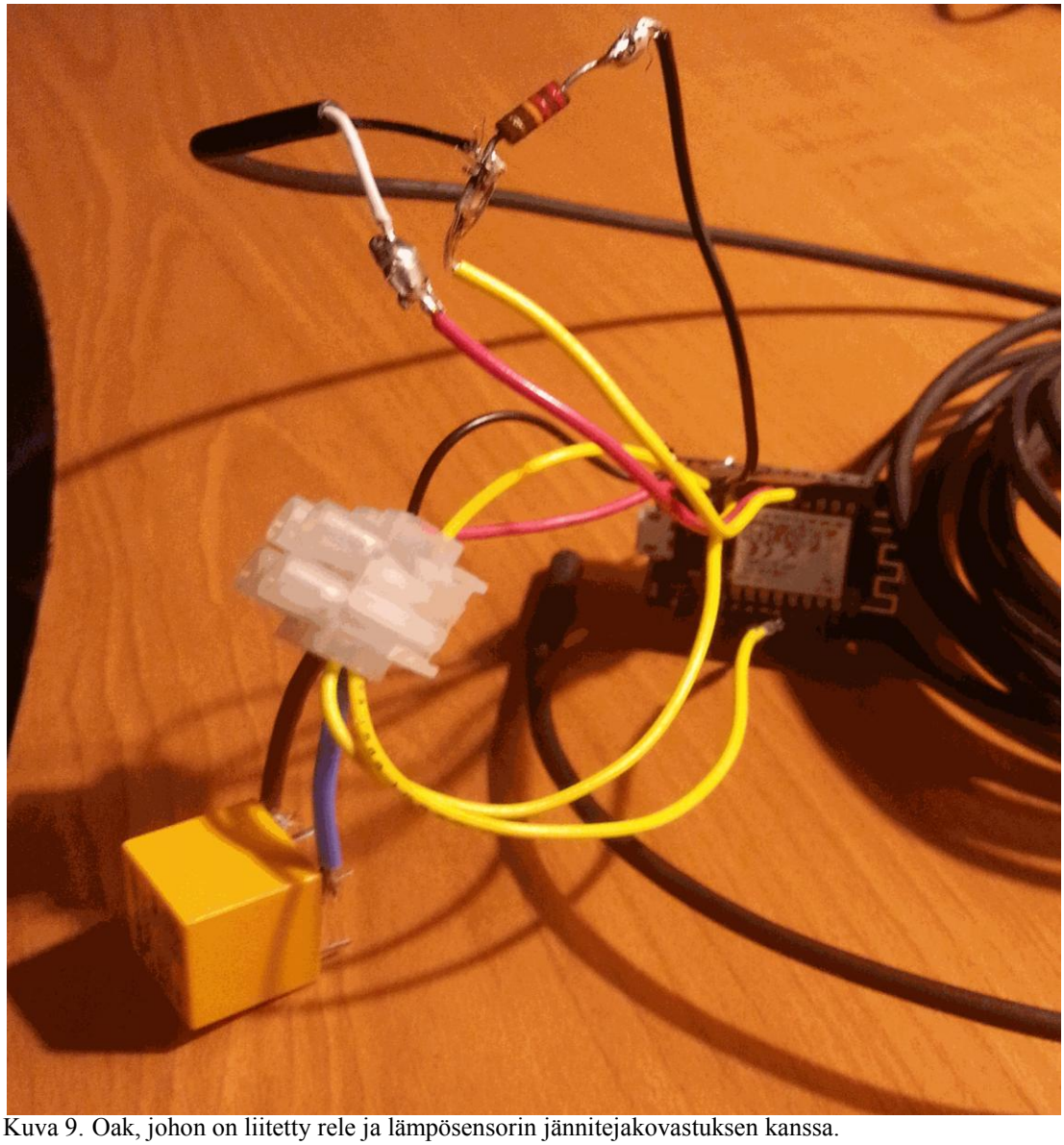
Kuva 7. Backend -koodi App Inventorissa.



Kuva 8.Valmis käyttöliittymä Android -laitteen näytöllä, jossa mukana testausta helpottavaa informaatiota.

3.5 Työn lopputulos

Lopputuloksena oli Oakille rakennettu IoT -yksikkö, jonka päätoiminto oli releen tilanvaihto, jota ohjasivat automaattisesti lämpösensori ja kalenterimerkinnät, sekä manuaalisesti Android -käyttöliittymä(liite 3,kuva 9). Kyseessä oli yksinkertainen toteutus, jossa pääasia oli saada kaikki toimimaan tarkoitetulla tavalla. Esimerkiksi palautettu lämpötila oli edelleen JSON -formaatissa muiden tietojen joukossa, eikä nappuloiden nimiä ole muutettu niiden tarkoituksen muututtua.



Kuva 9. Oak, johon on liitetty rele ja lämpösensorin jännitejakovastuksen kanssa.

4 Pohdinta

4.1 Jälkimietteitä

Suurin haaste projektissa oli yhteyksien muodostaminen ja oikeanlaisen muotoilun löytäminen, varsinkin kun aluksi ei ollut minkäänlaista käsitystä siitä, miten ylipäätään mitään saa siirrettyä tietoa internetin välityksellä. Lopuksi yrityksen ja erehdyksen ja systemaattisen ongelmanratkonnan päätteeksi kuitenkin yhteydet toimivat.

Työ on ollut opettavainen ja mielenkiintoinen kokemus, joka osoitti IoT:n olevan pohjimmiltaan yksinkertainen toteutus, joka ei kompastu monimutkaisuuteensa, kunhan on selvittänyt perusasiat.

4.2 Kehitys- ja jatkomahdollisuudet

IoT -verkon perusta on luotu. Sen esimerkin mukaisesti verkkoa voi laajentaa useammalla Oakilla tai haluamallansa internet -yhteyteen kykenevällä laitteella.

Käyttöliittymää voi parantaa ja viimeistellä enemmän käyttäjäystävälliseksi.

Jotta relettä voi käyttää verkkovirran katkaisijana, tarvitaan turvallisuuden vuoksi toinenkin rele toiselle johtimelle, turvallinen kotelointi sekä kunnollinen johdotus kosketushäiriöiden ja oikosulun vaaran poistamiseksi.

Tietoturva -asioita ei ole huomioitu projektin aikana, joten kehitysalusta ei saa olla kriittisessä käyttötarkoituksessa.

5 Yhteenveto

Työn aiheena oli tehdä yksinkertainen IoT -verkon yhteys, jonka välityksellä sai välitettyä esimerkiksi sensorin keräämää tietoa.

Erilaisten IoT -ohjelmien tarkastelun jälkeen projektissa käytettäväksi kehitysalustaksi valikoitui Digistump Oak ja ohjelmiksi MIT App Inventor ja IFTTT.

Aluksi selvitettiin REST API -internetpyyntöjen käyttö, sillä Particle Cloud, jonka kautta Oakin kanssa voi kommunikoida, käyttää tätä järjestelmää. Testaamisessa auttoivat Advanced Rest Client ja IFTTT Do Button.

Koodi Oakille kirjoitettiin Arduino kehitysympäristössä. Koodi sisälsi LEDin vilkuttamisen tapahtuvien toimintojen merkiksi, lämpösensorin lukemisen ja lukeman tulkkaamisen, releen tilanvaihdon, sekä yksinkertaisen application programmable interfacen. API sisälsi yhden Oakin muuttujan lukemisen ja lähettämisen, sekä toimintatilaa vaihtavan muuttujan vastaanoton. Lopullinen päätoimintatila sääti releen tilaa lämpösensorin lukeman ja Google Calendar -ohjelman merkintöjen mukaan itsenäisesti.

Projektin aikana testaukseen ja kehitysalustan hallintaan kirjoitettiin myös Android-käyttöliittymä, josta pystyi seuraamaan kehitysalustan muuttujan arvoa, sekä vaihtamaan toimintatilaa.

Lähteet

[1]Rouse M(Luettu 10.8.2016) Internet of Things. URL:
http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[2]Digistump(Luettu 10.8.2016) Oak by Digistump. URL:
http://digistump.com/oak/

[3]Digistump(Luettu 18.8.2016) Oak by Digistump: Wi-Fi for all things! (Arduino Compatible). URL:
https://www.kickstarter.com/projects/digistump/oak-by-digistump-wi-fi-for-all-things-arduino-comp

[4]Blynk (Luettu 10.8.2016) Blynk. URL:
https://play.google.com/store/apps/details?id=cc.blynk

[5]Blynk (Luettu 18.8.2016) URL:
http://www.blynk.cc/

[6]MIT (Luettu 18.8.2016) URL:
http://appinventor.mit.edu/explore/

[7]IFTTT (Luettu 18.8.2016) URL:
https://ifttt.com/

[8]Digistump (Luettu 8.8.2016) Connecting a new Oak. URL:
http://digistump.com/wiki/oak/tutorials/connecting

[9]Digistump (Luettu 18.8.2016) Connecting and Programming Your Oak Installation Instructions. URL:
http://digistump.com/wiki/oak/tutorials/arduino

[10]Particle (Luettu 18.8.2016) URL:
https://build.particle.io/build/

[11]Particle (Luettu 7.8.2016) Cloud functions.
https://docs.particle.io/reference/firmware/photon/#cloud-functions

[12]Digistump (Luettu 20.8.2016) Oak: Particle API basics. URL:
http://digistump.com/wiki/oak/tutorials/particle-id-token

[13] Jaansen G(Luettu 20.8.2016) Methods. URL:
http://restful-api-design.readthedocs.io/en/latest/methods.html

[14]Digistump (20.8.2016) Oak pinout diagram. URL:
https://digistump.com/wiki/oak/tutorials/pinout

[15]Espriff Systems(20.8.2016) ESP8266EX. URL:
http://download.arduino.org/products/UNOWIFI/0A-ESP8266-Datasheet-EN-v4.3.pdf

[16]Sun Hold Electric (20.8.2016) RAS series. URL:
http://www.sunhold.com/upload/prd1/21-3.pdf

[17]Desmos (20.8.2016) URL:
https://www.desmos.com/calculator

[18]Ensto (20.8.2016) Asennusohje. URL:
http://www.ensto.com/files/documents/ii/heat/ECO10_RAK40_UM2.pdf

Liitteet

Liite 1: Yhteyksien testaus

```
int pause = 1000; //muuttuja muokattavaksi ja luettavaksi

void setup() {

  pinMode(1, OUTPUT); //LED

  Particle.function("stoppause", funcstop); //saapuva yhteys
  Particle.variable("outtest", pause); //lähtevä yhteys

}

int funcstop(String arg) //tilanvahdot
{
  pause = arg.toInt();
}

void loop() {

   //pausesta riippuva välkkymisnopeus
 digitalWrite(1, HIGH);
 delay(pause);
 digitalWrite(1, LOW);
 delay(pause);

}
```

Liite 2: Sensorin ja releen testaus

```
int pause = 1400;

int tem = 500;
int calcu=0;
int calcu2=0;
int selection=0;
int rel=0;

void setup() {

  pinMode(1, OUTPUT); //LED
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(10, OUTPUT); //sensori 3v3
  pinMode(9, OUTPUT); //sensori 0v
  pinMode(4, OUTPUT);

  Particle.function("stoppause", funcstop); //saapuva yhteys
  Particle.variable("outtest", pause); //lähtevä yhteys

}

int funcstop(String arg) //tilanvahdot
{
  selection = arg.toInt();
}

void reon(){
                //rele päälle
    digitalWrite(4, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    digitalWrite(8, HIGH);
}
void reoff(){
                 //rele pois
    digitalWrite(4, LOW);
    digitalWrite(6, LOW);
    digitalWrite(7, LOW);
    digitalWrite(8, LOW);
}
void loop() {
                  //pääohjelma
digitalWrite(10, HIGH);
digitalWrite(9, LOW);
                  //tilanvalinta
if (selection==0){
                  // lämpötila celsiuksina
 tem = analogRead(A0);
 tem= 1024-tem;
 calcu=tem+1110;
 calcu2=tem*1.17+1110;
 pause=29000/calcu+155-calcu2*4/52;
 delay(2000);
}

else if(selection==1){
                  //lämpötila raakadatana
  pause = 1024 - analogRead(A0);
  delay(1000);
```

```
}
else if(selection==2){
                  //releen säätely lämpösensorin mukaan 30C
  pause = 1024 - analogRead(A0);
  delay(20000);
  if(pause<650){
    reon();
  }
  else{
    reoff();
  }
}

else if(selection==5){
                //releen tilanvaihtelu
  reon();
  delay(1000);
  reoff();
  delay(1000);
}
else if(selection==6){
  reon();
}
else if(selection==7){
  reoff();
}
else{
                //tila valitsematta
 digitalWrite(1, HIGH);
 delay(100);
 digitalWrite(1, LOW);
 delay(1200);
 digitalWrite(1, HIGH);
 delay(100);
 digitalWrite(1, LOW);
 delay(1200);
}
if(pause<1){
  pause=1500;
}

 digitalWrite(1, HIGH);
 delay(100);
 digitalWrite(1, LOW);
 delay(100);

}
```

Liite 3: Lopullinen koodi

```
int pause = 1400;
int selection=0;

void setup() {

  pinMode(1, OUTPUT); //LED
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(10, OUTPUT); //sensori 3v3
  pinMode(9, OUTPUT); //sensori 0v
  pinMode(4, OUTPUT);

  Particle.function("stoppause", funcstop); //saapuva yhteys
  Particle.variable("outtest", pause); //lähtevä yhteys

}

int funcstop(String arg) //tilanvahdot
{
  selection = arg.toInt();
}

void reon(){
                //rele päälle
    digitalWrite(4, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    digitalWrite(8, HIGH);
}
void reoff(){
                 //rele pois
    digitalWrite(4, LOW);
    digitalWrite(6, LOW);
    digitalWrite(7, LOW);
    digitalWrite(8, LOW);
}
void loop() {
                  //pääohjelma

digitalWrite(10, HIGH); //jännitteet sensorille
digitalWrite(9, LOW);

    //kun relon kalenterimerkintä alkaa, selection saa arvon 3, rele aktivoituu kun lämpötila on yli 30 astetta
if(selection==3){
  pause = 1024 - analogRead(A0);
  if(pause<639){
    reon();
  }
  else{
    reoff();
  }
  delay(10000);
}


    //kalenterimerkintä reloff vaihtaa releen pois päältä
else if(selection==4){
  reoff();
}

else{
```

```
                //tila valitsematta tai tila tuntematon
 digitalWrite(1, HIGH);
 delay(500);
 digitalWrite(1, LOW);
 delay(500);
}

 //Oakin aktiivisuuden ilmaisemista varten
 digitalWrite(1, HIGH);
 delay(1000);
 digitalWrite(1, LOW);
 delay(1000);
}
```