TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Tartu kolledž

ENOCEAN RAADIOLIIKLUSE JÄLGIMISE VÕIMEKUSE LOOMINE TARTU KOLLEDŽI ELUSLABORATOORIUMILE

ESTABLISHING ENOCEAN RADIO TRAFFIC MONITORING IN TALTECH TARTU COLLEGE LIVING LAB

BAKALAUREUSETÖÖ

Üliõpilane: Gregor Randla

Üliõpilaskood 178411EDTR

Juhendaja: Ago Rootsi, lektor

Tartu 2020

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.
Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

jaanuar 2021.a
Autor: Gregor Randla */allkirjastatud digitaalselt/*

Töö vastab bakalaureusetöö esitatud nõuetele

jaanuar 2021.a
Juhendaja: Ago Rootsi */allkirjastatud digitaalselt/*

Kaitsmisele lubatud

jaanuar 2021.a

Kaitsmiskomisjoni esimees: Helle Hallik */allkirjastatud digitaalselt/*

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Gregor Randla (sünnikuupäev: 20.01.1997)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „ENOCEAN RAADIOLIIKLUSE JÄLGIMISE VÕIMEKUSE LOOMINE TARTU KOLLEDŽI ELUSLABORATOORIUMILE“,

mille juhendaja on Ago Rootsi,

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

---

[1] *Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil.*

/allkirjastatud digitaalselt/

jaanuar 2021.a

TALTECH TARTU KOLLEDŽ
LÕPUTÖÖ ÜLESANNE

Üliõpilane: Gregor Randla 178411EDTR

Õppekava, peaeriala: EDTR17/17 - Telemaatika ja arukad süsteemid

Juhendaja(d): Lektor, Ago Rootsi, +372 56629821

Lõputöö teema:

(eesti keeles) EnOcean raadioliikluse jälgimise võimekuse loomine Tartu kolledži eluslaboratooriumile

(inglise keeles) Establishing EnOcean radio traffic monitoring in TalTech Tartu college Living Lab

Lõputöö põhieesmärgid:

1. Mõista EnOcean pakettide loogikat
2. Leida riistvara EnOcean andmete pakettide vastuvõtmiseks ja MeiePilv andmebaasi saatmiseks
3. Ühendada riistvara, teha vajalikud muutused tarkvarale ja katsetada valmis lahendust

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.		
2.		
3.		

Töö keel: Eesti keel

Lõputöö esitamise tähtaeg: 11. jaanuar 2021.a

Üliõpilane: Gregor Randla */allkirjastatud digitaalselt/* jaanuar 2021.a

Juhendaja: Ago Rootsi */allkirjastatud digitaalselt/* jaanuar 2021.a

Programmijuht: Helle Hallik */allkirjastatud digitaalselt/* jaanuar 2021.a

SISUKORD

EESSÕNA 7
Lühendite ja tähiste loetelu 8
SISSEJUHATUS 9
1. Lähteülesanne 10
1.1 Eluslabor ja selle tähtsus TalTech Tartu kolledžis 10
1.2 Ülesande sisu 10
2 Ülesande jaotus alamülesanneteks 12
2.1.1 EnOcean andmesidega tutvumine 12
2.1.2 MeiePilv API iseärasused 12
2.1.3 Pakettide vastuvõtmise ja edasisaatmise lahenduse valimine - seadmevalik 12
2.1.4 Tarkvara kirjutamine/muutmine 12
2.1.5 Katsetamine ja optimeerimine 13
3 Meiepilv 14
3.1 MeiePilv ülesehitus 14
3.2 OmaWiFi 14
4 Enocean alliance 17
4.1 EnOceani toodete põhimõtted 17
4.2 EnOceani protokoll 18
4.3 EnOcean andurid eluslaboris 19
4.3.1 EnOcean lüliti PTM210 DB 19
4.3.2 Thermokon SR06 LCD ruumikontroller 20
4.3.3 Schneider Electric LSS10020032 magnetkontaktandur 21
4.4 EnOcean pakettide vastuvõtmine 21
5 Vastuvõtuvärava lahenduse valik 23
5.1 Pressac EnOcean Gateway 23
5.2 EnOcean Pi + Raspberry Pi 23
5.3 Thermokon EnOcean Gateway 24
5.4 WinShine LD-GateWay/A 25
5.5 ESP32-WROOM-32/ESP8266-12E + EnOcean Pi 25
5.6 Lõplik valik 26
6 Prototüübi arendus 28
6.1 Elektroonika ühendused 28
6.2 EnOcean pakettide esialgne lugemine 29
6.3 EnOcean pakettide vastuvõtmise tarkvara kirjutamine 29

6.4 Andmete saatmine MeiePilv andmebaasi 31
6.5 Lõpliku koodi selgitus 32
7 Võimalikud arengusuunad 37
7.1 Andurite andmete talletamine püsimälusse 37
KOKKUVÕTE 38
SUMMARY 39
KASUTATUD KIRJANDUSE LOETELU 40
LISAD 43

EESSÕNA

Käesoleva lõputöö ülesanne on sõnastatud TalTech Tartu kolledži õppejõu Ago Rootsi poolt. Temalt pärinevad ka esmased viited seonduvale materjalile ja olemasolevatele süsteemidele.

EnOcean, juhtmevaba, eluslabor, bakalaureusetöö.

LÜHENDITE JA TÄHISTE LOETELU

API – rakendustarkvara liides

GND – maa, ühisklemm, *ground*

HTTP - HyperText Transfer Protocol, võrguprotokoll andmete edastamiseks arvutivõrkudes

JSON - JavaScript Object Notation, universaalne andmeedastuse protokoll, mis on üheks võimalikuks ühendavaks formaadiks erinevate rakenduste vaheliseks andmevahetuseks olles seejuures ka inimloetav

kbit/s – kilobitti sekundis

mAh – milliampertund

MHz – megaherts

Rx – ressiiver, andmeside sisendklemm UART andmesides

Tx – transmitter, andmeside väljundklemm UART andmesides.

UART – transistor – transistor loogika (TTL) pingenivoodel töötav kahesuunaline järjestikside liides.

WiFi - Wi-Fi ehk raadiokohtvõrk võimaldab seadetel nagu lauaarvutitel, mängukonsoolidel, nutitelefonidel või digitaalsetel muusikamängijatel luua ühenduse internetiga, kui nad asuvad Wi-Fi-levialades.

SISSEJUHATUS

2015 aastal lisandus tollasesse Tallinna Tehnikaülikooli Tartu kolledžisse uus õppekava nimega Telemaatika ja arukad süsteemid. Õpetatavaks peaerialaks oli küberfüüsikaline süsteemitehnika (hiljem küberfüüsikalised süsteemid), mille raames tehti algust innovaatilise õpikeskkonna, Eluslaboratiooriumi, rajamist. Eluslabori jaoks loodi ka lokaalne andmebaas nimega MeiePilv. Eluslabori põhieesmärgiks on rikastada tudengite õpitööd läbi praktiliste kogemuste. Lisaks paljudele teistele hüvedele võimaldab eluslabor kasutada tehisintellekti, mis omakorda võimaldaks näiteks tarka tehnosüsteemide kasutamist. See aitaks kaasa omakorda väga aktuaalsele teemale – energia kokkuhoid.

Ruumiandmete kogumiseks eluslaboratooriumis valiti välja raadiolahendused ja üheks neist raadiolahendusteks on EnOcean. EnOcean on ligi 20 aastat vana raadiotehnoloogia, mis toodab hooldusvabasid ja madala energiatarbega juhtmevabu andureid ja lüliteid [1]. 2008. aastal loodi ka EnOcean Alliance, mille eesmärgiks on koondada firmasid, kes töötavad hooneautomaatikaga, sealhulgas ka nn targa kodu lahendustega. Praeguse seisuga on selles ühenduses üle 400 liikme ja Alliance kuulsamateks edendajateks on näiteks Microsoft, IBM ja Honeywell [2].

Lõputöö eesmärgiks on luua TalTech Tartu kolledži ruumidesse EnOceani raadioliikluse jälgimise süsteem. See koguks sisekliimaautomaatika seisundi- ja tagasisideandmeid MeiePilv andmebaasi, et neid tagantjärgi analüüsida ja seeläbi automaatika toimimist optimeerida väiksema energiakulu ja parema sisekliima suunas. Kuna automaatika areneb järjepidevalt, võib ka automaatika töös seisakuid toimuda. Selle eest töötavad EnOceani andurid edasi ja lõputöös loodud süsteem saadab andmeid katkestustest sõltumatult andmebaasi. MeiePilve andmebaasil on oma iseärasused, mistõttu kõik valmislahendused EnOceani vastuvõtuvärava (gateway) jaoks ei pruugi sobida. Veendumaks, et vastuvõtuvärav sobib, oli esmalt vaja tutvuda EnOcean andmesidega. Seejärel ka täpsemalt MeiePilve ja selle iseärasustega. Tuleb tutvuda olemasolevate lahendustega ja kontrollida nende sobivust.

1. LÄHTEÜLESANNE

1.1 Eluslabor ja selle tähtsus TalTech Tartu kolledžis

Praeguses arenevas maailmas on loodud võimalusi väga paljudeks erinevateks innovaatilisteks arenguteks. Suur muutus on ka toimunud tarbimises – väga jõudsalt liigutakse toodetepõhiselt majandusest teenustepõhisele majandusmudelile [3]. Innovatiivsete teenuste väljaarendamine, mida lõppkasutajad tahaksid kasutada ja näeksid ka selles potentsiaali, on raske. Sellisel arendusel on mõistlik kasutada eluslaborit. Eluslabor on avatud innovatsiooniprotsesside korraldaja, mis keskendub päris elulisele koosloomele kaasates erinevaid huvirühmasid arendusel ja keskendudes eelkõige lõppkasutajale [4]. Eluslaboratoorium kui süsteemide käitumise uurimise meetod, on laialdaselt kasutuses sedalaadi probleemide lahendamisel. Siin ei eraldata uurimistööks tervikust mingit osa vaid uuritakse süsteemi kui tervikut. TalTech Tartu kolledži eluslabor on eelkõige õpikeskkond.

TalTech Tartu kolledži üks eesmärkidest on eluslabori abil rikastada õppetööd eelkõige süsteemitehnika valdkonnas. Esialgu alustati sisekliima juhtimise automaatika valdkonnast, kus tagasiside tehtule on ka otseselt tajutav. Protsesse jälgides ja kogutuid andmeid läbi töötades saavad üliõpilased praktilisi kogemusi nii hooneautomaatika kui ka hoone sisekliimat tagava muu tehnosüsteemi valdkonnas. Samuti on praeguses ajas väga aktuaalne ka energia kokkuhoid ja sisekliima juhtimise uurimine aitab selle valdkonna arengule kaasa. Nimelt on sisekliima optimeeritud juhtimine üks olulisemaid meetodeid hoonete parema energiatõhususe tagamisel. Sisekliima targa juhtimise eelduseks on palju eelnevaid andmeid, mille peale saab hiljem baseeruda näiteks tehisintellekti süsteemid.

1.2 Ülesande sisu

Ruumiandmete kogumiseks valiti enne Eluslabori rajamisel välja raadioprotokolliga töötavad seadmed, et vältida kaablite tõmbamist kõikidesse ruumidesse ja samuti vältida probleeme muinsuskaitsega. Nimelt muinsuskaitsest tulenevalt on Puiestee 80A hoones, millest eluslaboratooriumi rajamist alustati, komplitseeritud täiendavate ühenduskaablite paigaldamine. Sellest tulenevalt valiti hooneautomaatika kontrolleriteks Schneider Electric GMBH MPM seeria, mis ongi ette nähtud raadioside kasutamiseks. Üheks kasutatavaks raadioprotokolliks valiti EnOcean, millel on palju

eeliseid – madala energiatarbega standard, võrdlemisi hea levi siseruumides (kuni 30 meetrit) ja EnOcean-i andurid omavad uuenduslikke süsteeme [5].

Lõputöö eesmärgiks on luua paralleelne andmekogumiskanal, mis võimaldaks vastu võtta EnOcean-i signaale ja neid edasi saata lokaalsesse andmebaasi nimega MeiePilv. Sarnased seadmed on olemas, kuid tähtis osa andmebaasi saatmise juures on ka see, et antud seade toimiks autonoomselt, st ta jälgib automaatika andmeliiklust olemata ise automaatika osa. Kasutades MPM kontrollereid otse andmete lugemiseks võib tekida olukord, kus ümberseadistamiste ja/või muude töökatkestuste korral andmete logimine katkeb. Valitav või valmistatav seade aga jätkab raadioliikluse jälgimist tagades sellega andmete pideva kogumise.

2 ÜLESANDE JAOTUS ALAMÜLESANNETEKS

Lõputöö ülesande lahendamiseks jagasin selle alljärgnevateks omavahel seotud alamülesanneteks:

2.1.1 EnOcean andmesidega tutvumine

Selle lõputöö läbiv raadioprotokoll on EnOcean. Lõputöö eesmärgi täitmiseks on vaja tutvuda eelmainitud protokolliga ja tema tehniliste iseärasustega, et hiljem vastuvõetud pakette õiges vormingus sisse lugeda ja õiged andmed edasi andmebaasi saata. Nendeks iseärasusteks on:

- edastuskiirus
- paketi sisu – paketis asuvate andmete asukohad
- EnOcean andurite erisused andmete saatmisel

2.1.2 MeiePilv API iseärasused

Lähteülesannet saades mõistsin, et muutuste tegemine andmebaasis MeiePilv ei ole mõistlik. See tähendas seda, et pakett, mis tuleb saata andmebaasi, tuleb modifitseerida vastavalt andmebaasile.

2.1.3 Pakettide vastuvõtmise ja edasisaatmise lahenduse valimine - seadmevalik

Pakettide vastuvõtmise osa on üheselt lahendatav: EnOcean on raadioprotokoll ja selle vastuvõtmiseks tuleb leida õige seade. Pakettide edasi saatmiseks on vaja teha valik – kas andmed edastada samuti üle õhu või edastada need juhtmeühenduse kaudu. Vastavalt sellele valikule tuleb teha valik mikrokontrolleri osas. Pärast seadmete valikut tuleb teostada seadme kooste ja valida ka seadmele toitelahendus.

2.1.4 Tarkvara kirjutamine/muutmine

Pärast põhjalikku tutvumist EnOcean andmesidega, MeiePilve API-ga ja pärast elektroonika valikut, on võimalus koostada mikrokontrollerile tarkvara või muuta olemasoleva kontrolleri tarkvara endale sobivaks. See hakkab omakorda juhtima andmepakettide vastuvõtmist ja ka edasisaatmist, tehes paketile vajalikke muudatusi tulenevalt serveri ja lähteülesande iseärasustest.

2.1.5 Katsetamine ja optimeerimine

Komplekteerides kõik eelneva, saab hakata katsetama seade töötamist. Kokkupandud lahendus vajab katsetamist, et kontrollida prototüübi tööd, selgitada välja võimalikud mittevastavused ja optimeerida nii riist- kui ka tarkvara.

3 MEIEPILV

3.1 MeiePilv ülesehitus

MeiePilv on TalTech Tartu kolledžis C-majas asuv server, millel on MONGO DB andmebaasil rakendus mõõtmistulemuste ja süsteemi seisundite salvestamiseks ja haldamiseks. Salvestamine toimub korraga kahele kõvakettale (RAID1). Samuti on serveril olemas WAN IP, mis tähendab, et ta on laivõrgus kättesaadav. Andmeid on võimalik maha lugeda NetAtmo tootepõhisest pilvest. Andmebaas töötab juba 3 aastat ja selle aja jooksul on talle lisatud võimekust võtta vastu erinevate edastuse iseärasusega andmevoogusid - Schneider Electric GMBH MPM tüüpi automaatikakontrollerid toetavad MODBUS, EnOcean ja Zigbee ühendust anduritega. Samuti on võimalus vastu võtta LoRaNet ja oBIX pakette.

Erinevate andmete jaoks on pilvel oma vaheserverid, kas rakendusena kusagil arvutis või eraldi kontrollerina. Igat sellist vaheserverit nimetatakse MeiePilve kontekstis võrguks (network). Omaette võrguna on välja toodud ka selle andmebaasi otsesisend, millega saab eespool kirjeldatud vaheservereid liidestada.

3.2 OmaWiFi

Lõputöö käigus loodav EnOcean pakettide vastuvõtuvärav hakkab andmeid saatma vahevõrku nimega OmaWiFi. See suudab vastu võtta andmeid üle laivõrguühenduse, mis omakorda teeb selle pilve mistahes maailma punktist kättesaadavaks.

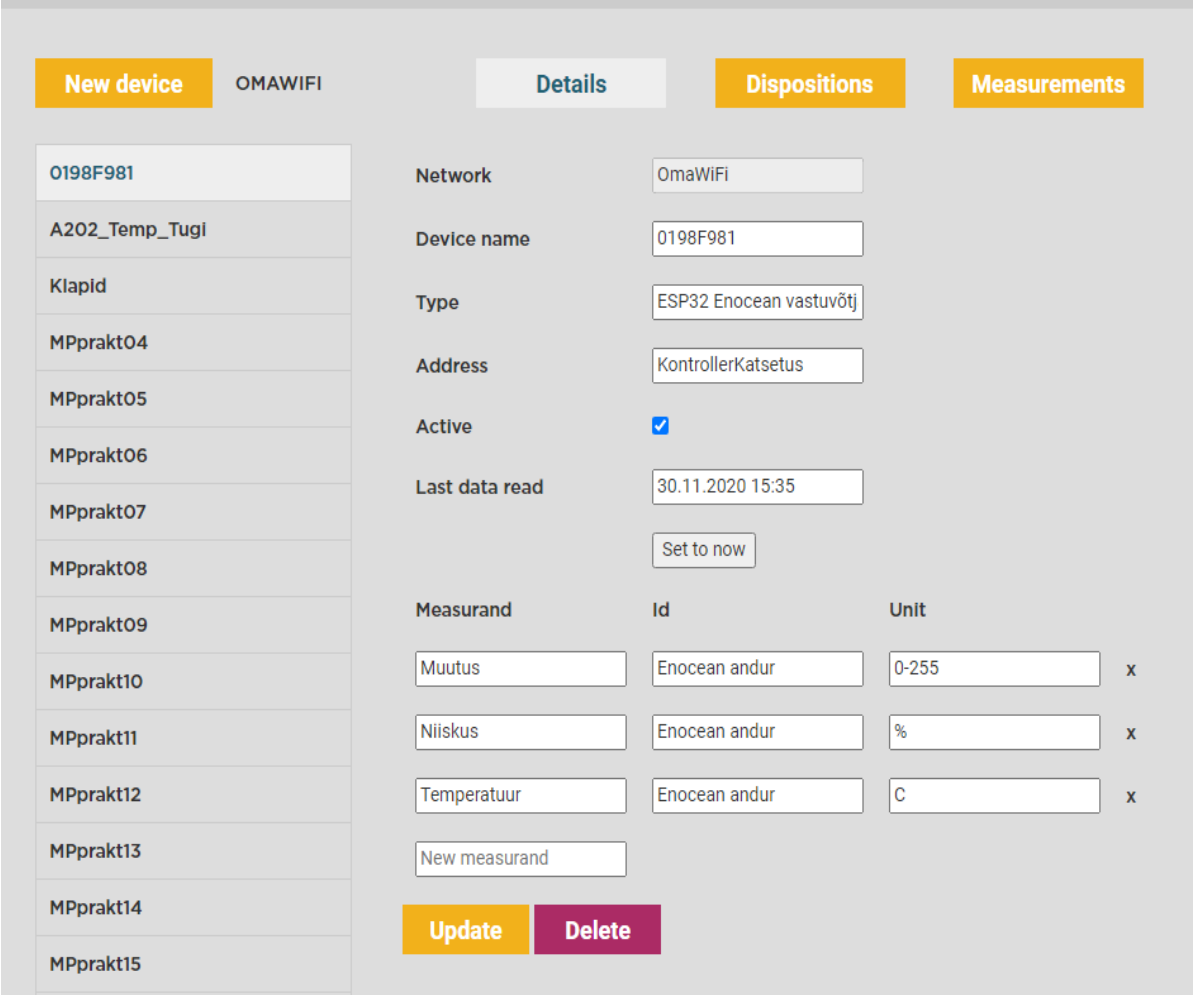
Pakett saadetakse OmaWiFi vaheserverisse JSON-formaadis. Paketi päis peab sisaldama kahte asja: *device name* ehk anduri nime ja *network name* ehk võrgu nime (Joonis 3-1).

```
{
"device_name": "0198F981",
"network_name": "OmaWiFi",
```

Joonis 3-1: Paketi päise näide

Igal EnOcean anduril on enda unikaalne identifitseerimiskood, mis on üldiselt kirjutatud ka selle anduri tagumisele paneelile. Seetõttu oli loogiline kasutada anduri ID-d tema tunnuseks serverisse saatmisel. Võrgu nimeks tuleb iga kord panna OmaWiFi, et paketid jõuaksid alati õigesse kohta. Selleks, et server võtaks vastu ja salvestaks andmed

OmaWiFi võrku, tuleb sinna teha iga anduri kohta *device* ehk üksuse nimi ja defineerida ära antud üksuse parameetrid, mis sellelt serverisse tulevad (Joonis 3-2).



Joonis 3-2: Serveris oleva anduri näide

Paketi päisele järgneb paketi kehand (Joonis 3-3). Kehandis sisalduvad *time* ehk ajamuutuseandmed alates paketi saatmisest ning *values* ehk mõõteandmed. Ajamuutuse andmeid on võimalik kasutada siis, kui on soov saata serverisse näiteks 10 andmepaketti korraga. See võimaldab lisada iga andmepaketi juurde aja, millal see sisse loeti. Kuna antud projektis on eesmärgiks saata pakett koheselt serverisse, ilma andmeid kogumata, siis igal meie saadetud paketil on ajamuutus saatmise hetke suhtes 0. Järgmisel real asetsevad anduri poolt saadetud andmed. Need on eraldatud komadega ja olulist rolli mängib andmete järjestus – järjekorras esimene andmepakett on serveri vaates ülevalt alla esimene *measurand* ehk mõõdetav suurus (Joonis 3-2), teine pakett - teine mõõdetav suurus jne.

Serverisse paketi saatmisel saadab server ka vastuse, kas serveriga on ühendust saadud. Server ei saada vastust selle kohta, kas samanimeline andur on juba serveris olemas. Kui peaks juhtuma, et serverisse hakkab andmeid saatma andur, mida ei ole veel võrguüksuse nimedes kirjas, ei salvestata neid andmeid serverisse.

```
"body":  [
 {
  "time": 0,
  "values": [127.00, 42.00, 23.04]
 }
 ]
```

Joonis 3-3: Paketi kehandi näide

4 ENOCEAN ALLIANCE

EnOcean Alliance on rahvusvaheline juhtivate ehitus- ja IT-ettevõtete ühing, mis asutati 2008. aastal. See mittetulundusühing on pühendunud nutikodude, nutihoonete ja nutiruumide koostalitlusvõimeliste ökosüsteemide võimaldamisele ja edendamisele, tuginedes hooldusvabale sidestandardile (ISO / IEC 14543-3-10 / 11). EnOceani ökosüsteem koosneb praegu 5000 tootevariandist, mis põhinevad 1500 põhitootel ja uusi tooteid arendatakse igapäevaselt. Tänu standardsele andurite profiilile saab uusi ja vanu tooteid omavahel integreerida vähese vaevaga. [2]

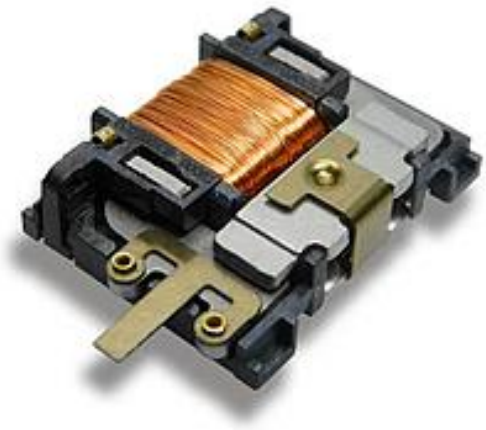
4.1 EnOceani toodete põhimõtted

Sidestandard, millele EnOceani seadmeid tehes tuginetakse, käsitleb juhtmevaba protokolli madala energiatarbega seadmetes. Tähtsateks aspektideks on ka:

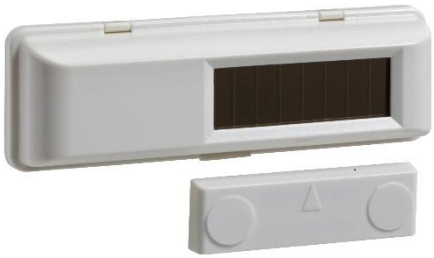
- kommunikatsiooni lühikeste pakettidena hoidmine
- tavaliselt ühesuunaline ja harv pakettide edastus
- selliste ülekandesageduste kasutamine, kus madala saatevõimsuse juures on suur edastusmaa
- energia ammutamine ümbritsevast keskkonnast [6]

Viimane punkt on ka väga päevakajaline teema. Nimelt alternatiivsete energiaallikatega andurite kasutamine võimaldab vähendada keemilisi toiteallikaid kasutavaid seadmeid. Keemilised toiteallikad on omakorda jäätmetena loodusevaenulikud ja kasutavad haruldasi leelismuldmetalle nagu näiteks liitium, mille varud on piiratud. Lisaks eelnevale on alternatiivsete energiaallikatega seadmed energiasõltumatud, st elektri puududes jäävad nemad siiski tööle.

EnOceani raadioandurite poolt kasutatavaid alternatiivseid energiaallikaid on mitmeid. Näiteks on lülititel võimekus ammutada energiat lüliti vajutamisest ja selle energiaga saata välja andmepakett (Joonis 4-1). Samuti on olemas andureid, millel on integreeritud väikesed päikesepaneelid, mis hakkavad alates 200 luksist piisavalt energiat tootma, et pakett välja saata (Joonis 4-2). On olemas ka andurid, mis toodavad energiat temperatuuri vahest alates 2°C. Viimane on tihedasti kasutuses küttetorustike klappide ja siibrite ajamites. Mõningatel juhtudel on ka neid allikaid kombineeritud. [5]



Joonis 4-1: Lüliti, mis toodab saatmiseks vajaliku energia vajutamisega [3]



Joonis 4-2: Päikesepaneeliga EnOceani magnetkontaktandur [7]

4.2 EnOceani protokoll

Euroopa regulatsioonidele vastab 868 Mhz ülekandesagedusel töötavad EnOcean saatjad ja vastuvõtjad. Teistes riikides on veel kasutuses ülekandesagedused 902 MHz (USA ja Kanada) ja 928 MHz (Jaapan). EnOcean on lühimaa raadioprotokoll. Leviala sõltub suuresti ümbritsevast keskkonnast, kuid tootja lubab kuni 30 meetrit siseruumides ja 300 meetrit vabas õhus (ilma takistusteta). Kaugematest kohtadest signaali püüdmiseks on võimalik kasutada raadiopikendit (*repeater*). Andmeid edastatakse samuti kiiresti – 125 kbit/s [5]. Katsetades ise andureid jõudsid kõik paketid kohale. Üheaegselt jõudnud paketid ei seganud üksteist ja edastati üksteise järel.

EnOceani pakettide mõistmiseks oli esialgu vaja mõista EnOcean Serial Protocol 3 (ESP3). ESP3 kommunikatsioon põhineb 3-juhtmelise UART ühendusel (Rx, Tx, GND) kasutades tarkvaralist kontrolli ja täisdupleks ühendust, sarnaselt RS-232 standardile [8].

Järgnevalt oli vaja mõista pakettide ülesehitusest. Esimene bait on EnOceani seadmetel, mis kasutavad ESP3 standardit, alati sama – 0x55. See juhtbait võimaldab valideerida paketti ja tuvastada paketi alguspunkti. Järgnevalt kahelt baidilt saab lugeda andmete paketi pikkust alustades esimesest baidist ja lõpetades sisulise infoga (Data). Sisulise info all on mõeldud andmeid, mida andur loeb, näiteks temperatuur, süsihappegaasi kontsentratsioon, niiskus jne. Sisulisele infole järgneb neli baiti infot, mis sisaldab anduri ID-koodi. (Joonis 4-3)

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	Serial synchronization byte; always set to 0x55
Header	1	2	Data Length	0xnnnn	Specifies how many bytes in DATA must be interpreted
	3	1	Optional Length	0xnn	Specifies how many bytes in OPTIONAL_DATA must be interpreted
	4	1	Packet Type	0xnn	Specifies the packet type of DATA, respectively OPTIONAL_DATA
-	5	1	CRC8H	0xnn	CRC8 Header byte; calculated checksum for bytes: DATA_LENGTH, OPTIONAL_LENGTH and TYPE
Data	6	x	... ...	... ...	Contains the actual data payload with topics: - RawData (e.g. 1:1 radio telegram) - Function codes + optional parameters - Return codes + optional parameters - Event codes x = variable length of DATA / byte number
Optional Data	6+x	y	... ...	... ...	Contains additional data that extends the field DATA; y = variable length of OPTIONAL_DATA
-	6+x+y	1	CRC8D	0xnn	CRC8 Data byte; calculated checksum for whole byte groups: DATA and OPTIONAL_DATA

Joonis 4-3: ESP3 paketi ülesehitus

4.3 EnOcean andurid eluslaboris

Enne arendustööd tutvusin kolme erinevat tüüpi anduritega, mis on plaanis Eluslaborisse paigutada.

4.3.1 EnOcean lüliti PTM210 DB

PTM210 DB on ESP3 protokollil töötav lüliti. Antud lüliti on õhukese disainiga, kergesti seinale monteeritav ja ammutab saatmise energia lüliti vajutamisest. Lülitil on kolm erinevat positsiooni - kaks mittepüsivat äärtes ning üks püsiv positsioon keskel. Vajutades lülitile saadab ta välja EnOceani paketi, mille sisulise info pikkus on üks bait. Erinevatesse äärtesse vajutades saadetakse välja erinevad paketid. Jõudes tagasi keskele, saadab lüliti samuti paketi. Hoides lülitit all mõnda aega on võimalik näiteks sujuvalt muuta valgustugevust ruumis. Muutes lüliti korpust on võimalik sama saatjat

kasutada ka kahe klahvilise lülitiga (Joonis 4-4). Sellisel juhul saadab iga klahvi iga asend eraldi andmepaketi välja ehk 4 erinevat baiti.

Joonis 4-4: PTM210 DB korpused kahe ja ühe klahviga [9]

4.3.2 Thermokon SR06 LCD ruumikontroller

Ruumi sisekliima kohta info saamiseks ja selle juhtimiseks on osades ruumides kasutusel Thermokoni ruumikontrollerid. Antud mudelinumbriga ruumikontrollereid on erinevaid tüüpe. Kolledži kasutuses on kahe nupuga kontrollerid mudelinumbriga 2T, mis lisaks temperatuuri ja suhtelise õhuniiskuse edastamisele võimaldavad kasutajal sisestada endale sobiva temperatuuri muutuse miinus kolmest kuni pluss kolmeni (-3 – 3). Sellele mudelile lisaks on olemas nelja nupuga seadmed mis võimaldavad kontrollida lisaks temperatuurile ka valgust, kardinaid, ventilatsiooni jne (Joonis 4-5). [10]

SR06 LCD anduri temperatuuri mõõtmisvahemik on 0 - 40 °C täpsusega ±0,4 K. Niiskuse mõõtmisvahemik on 0 – 100 % täpsusel ±5 % (vahemikus 30 – 70 %). Ruumikontroller saadab pakette välja iga 240 sekundi järel ehk iga 4 minuti järel või iga kasutaja valitud temperatuuri muutuse järel. See aeg on ka muudetav kasutades AirConfig programmi. Saadetud andmepakett sisaldab neli baiti sisulist infot:

- temperatuurimuutus (vahemikus 0-255)
- niiskus (vahemikus 0-250)
- temperatuur (vahemikus 0-250)
- LRN bit, mis saadab vastuvõtuväravale anduri baasinfo vajutades taga asuvat õpetamise nuppu [10]

SR06 saab enda toite esiküljel olevast päikesepatareist, mis laeb liitiumpolümeerakut parameetritega 3.7 V ja 60 mAh. Samuti on anduri alumises servas lisa laadimisvõimalus mikro-USB pordi kaudu. Soovi korral on võimalik lisada kontrollerisse CR1632 patarei. [10]

Joonis 4-5: Thermokon SR06 LCD 4T ja 2T ruumikontroller [10]

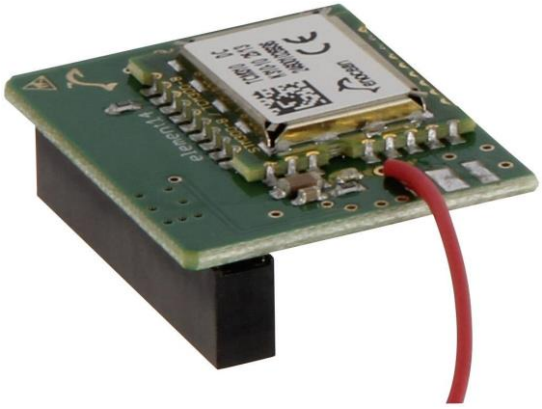
4.3.3 Schneider Electric LSS10020032 magnetkontaktandur

Viimaseks anduriks oli Schneider Electricu poolt valmistatud magnetkontaktandur. Ehituselt väike andur saab paketi saatmiseks vajaliku energia esiküljel asuvast päikesepaneelist (Joonis 4-2). Pakette saadab andur iga kord kui magnet viiakse anduri alumise osa juurde või magnet liigub sealt ära. Samuti saadetakse pakette teatud aja jooksul, kui muutusi magnetiga ei ole toimunud. Täpset ajavahemiku ei ole teada, kuid andureid kasutades selgus, et selleks ajaks võib lugeda umbes 22 minutit. Saadetud sisuliste andmete suurus on 1 bait, mis sisaldab infot selle kohta, kas andur on kontaktis magnetiga või mitte. Antud anduri kasutusvõimaluseks on näiteks aknakontakt - kütmisautomaatikat saab juhtida vastavalt sellele kas aken on avatud või suletud. Samuti on võimalik kasutada andurit ukse asendi jälgimiseks. [11]

4.4 EnOcean pakettide vastuvõtmine

TCM310 kiip on EnOcean pakette vastuvõttev kiip. Antud kiibile on tehtud laiendusplaat EnOcean Pi (Joonis 4-6), mille koosseisu kuulub kontroller TCM310. EnOcean Pi eeliseks on juhtmete ühendamise mugavdamine nii prototüüpimise faasis kui ka valmis toote jaoks.

TCM310 suhtleb läbi UART ühenduse kasutades selle jaoks kiirust 57600 bitti sekundis [8]. Kiip võimaldab vastu võtta ESP3 (EnOcean Serial Protocol 3) pakette EnOceani anduritelt kui ka saata ESP3 pakette anduritele tagasi. Sisuliselt oleks selle kiibiga võimalik ehitada ka ESP3 pakettide raadiopikendi. EnOcean Pi eelisteks on see, et ta võimaldab ise luua vastuvõtuvärava, mis on enda tehtud tarkvaraga ja hiljem muudetav vastavalt vajadusele. [12]



Joonis 4-6: EnOcean Pi ehk TCM310 joodetud Raspberry Pi laiendusplaadile [25]

5 VASTUVÕTUVÄRAVA LAHENDUSE VALIK

EnOceani andurite kasutamiseks on vaja andmete vastuvõtjaid. Samuti on olemas ka erinevate tootjate poolt loodud vastuvõtuväravad (*gateway*). Vastuvõtuväravate valimi sain EnOcean Alliance leheküljelt. Kuna nende lahendused on töötava elektroonika ja tarkvara kooslusega, peaks esialgu leidma esmapilgul sobivad tooted ja kontrollima nende ühilduvust antud projektiga.

5.1 Pressac EnOcean Gateway

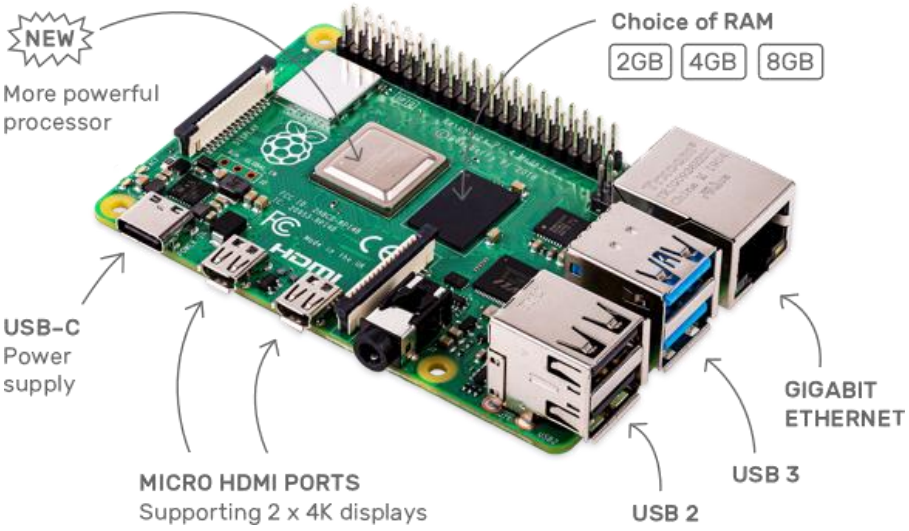
Antud seadmel on võimekus võtta vastu EnOceani pakette ja need saata WiFi või mobiilandmeside abil serverisse. Serverisse saadetakse andmeid JSON formaadis MQTT protokolliga. Samuti omab ta võimekust pakette serverisse saata juhtmega interneti abil. Olukorras, kus üks edastusviis ei tööta, on võimalik seadistada varusüsteem, mis paketi ära saadab. Näiteks kui paketti ei ole võimalik üle juhtme saata, saadetakse see hoopis üle 4G. [13]

Joonis 5-1: Pressac EnOcean Gateway [13]

5.2 EnOcean Pi + Raspberry Pi

EnOcean Pi on EnOceani pakettide vastuvõtmiseks mõeldud moodul (4.4). Raspberry Pi on võimekas mikroarvuti, millega on võimalus ühenduda võrku nii kaabli kui ka integreeritud WiFi kiibi abil. Samuti on tal olemas Bluetooth võimekus. Olenevalt Raspberry Pi versioonist, on tal erinevaid väljundeid, kuid valides näiteks neljanda versiooni, on võimalik kasutada USB liidesega seadmeid, kasutada ekraani (näiteks

liikluse jälgimiseks) ja ühendada Raspberry Pi internetti juhtme abil. EnOceani pakettide võrku saatmiseks, tuleb Raspberry Pi-le kirjutada tarkvara. [14]



Joonis 5-2: Raspberry Pi 4

5.3 Thermokon EnOcean Gateway

Thermokonil on 11 erinevat EnOcean vastuvõtuväravat. Välja saadetakse pakette kasutades erinevaid protokolle: modbus, EVC, BACnet, LON, KNX jpm. Üks vastuvõtuvärav võimaldab andmeid serverisse saata ka juhtmega ühenduse abil JSON formaadis. [15]

Joonis 5-3: Thermokon EnOcean Gateway RJ45 pesaga

5.4 WinShine LD-GateWay/A

WinShine LD-GateWay/A võtab vastu EnOcean pakette ja saadab need üle WiFi võrku. EnOceani andureid ja täitureid saab juhtida antud seadme põhisest mobiiltelefoni aplikatsioonist. Andmelehelt lugedes jääb selgusetuks, kas salvestamiseks on võimalik valida enda andmebaasi või toimib ainult seadmekeskne andmebaas. [16]

Joonis 5-4: WinShine LD-GateWay

5.5 ESP32-WROOM-32/ESP8266-12E + EnOcean Pi

Nii ESP32 kui ka ESP8266 on väikesed mikrokontrollerid. Oma olemuselt on nad üsnagi võimekad, WiFi ja Bluetooth toega ja madala energiatarbega. Kuna energiatarve on madal, pole tarvidust tegeleda jahutusprobleemidega. Mõlemad kontrollerid oleksid võimelised kasutama EnOcean Pi moodulit EnOcean pakettide vastuvõtmisel. Selle jaoks on võimalik kasutada mõlemal mikrokontrolleril UART porti. Mõlemad kontrollerid on vaja programmeerida vastavalt vajadusele Arduino IDE rakenduses. [17] [18]

ESP32 ja ESP8266 vahel valides langes valik ESP32 kontrollerile järgmistel põhjustel:

- ESP32 omab kahetuumalist protsessorit, mis töötab 160-240 MHz kiirusel, kuid ESP8266 omab ühetuumalist ning 80 MHz protsessorit
- ESP32 omab 3 UART porti, mis on 1 port rohkem kui ESP8266
- WiFi moodul ESP32 kiibil on uuem ning võimsam kui ESP8266-l
- ESP32 omab Bluetooth kiipi, mis avab võimalusi arendusteks

Kõigi nende lisade juures on ESP32 ja ESP8266 keskmine energiatarve sama suur. [17] [18]

Joonis 5-5: ESP32-WROOM-32 [19]

5.6 Lõplik valik

Võttes arvesse kolmandas, neljandas ja viiendas peatükis arutatud punkte, langes vastuvõtuvärava valik ESP32 mikrokontrollerile, mis töötab üheskoos EnOcean Pi kiibiga. Valiku põhjuseid on mitmeid:

1. Server MeiePilv pole minu poolt üles seatud ja seda muuta ei ole mõistlik, tuleb välja saadetavate andmete formaat kohandada vastavalt serverile. Valmis kujul olevad andmepakettide vastuvõtjad/saatjad sobivad küll väga hästi pakettide vastuvõtmiseks, kuid serveri iseärasuste tõttu oleks vaja lisada pärast *gateway*-d üks kontroller või mikroarvuti, mis muudaks saadetava koodi serverile sobivaks.

2. Valmis kujul maksavad EnOcean paketi vastuvõtjad/saatjad ligikaudu 500€ või rohkemgi. ESP32 baasil vastuvõtuvärava hind peaks tulema umbes 50€. See tähendaks 10-kordset rahalist võitu.

3. Eluslaboratoorium on pikema aja jooksul kujunev ja ümberkujunev üksus. Valides ESP32 ja kirjutades vastava tarkvara sinna ise peale, tähendaks seda, et hiljem on võimalik tarkvara muuta vastavalt uutele soovidele. Ainsana teistest võimaldab seda veel teha Raspberry Pi

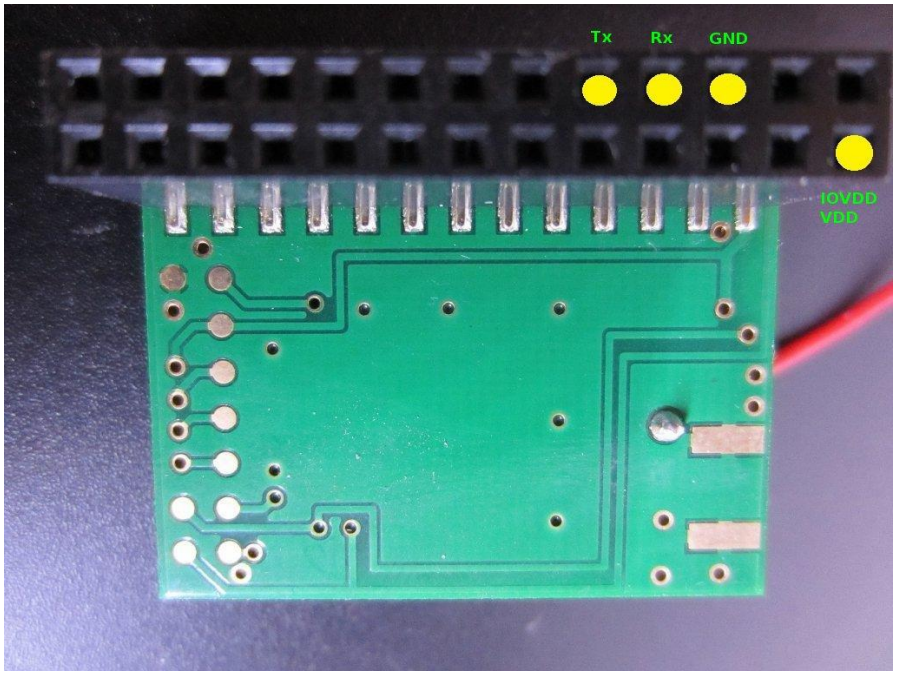
4. Raspberry Pi lahendusega võrreldes on ESP32 madalama energiatarbega ja väiksem mikrokontroller. See mugavdab ESP32 baasil lahendust integreerida näiteks olemasolevatesse automaatikakilpidesse lihtsa vaevaga ja samuti poleks probleeme mikrokontrolleri ülekuumenemisega.

5. Lisaks eelnevale on mõlemale mikrokontrollerile tarvilik luua tarkvara, et nad võtaksid vastu pakette ja ühilduksid OmaWiFi liidesega. Valides ESP32 tähendaks seda, et tarkvara koostamine toimuks Arduino IDE-s ja antud hetkel on see parem variant, kuna mul on sellega varasemalt kogemust võrreldes Raspberry Pi-le tarkvara kirjutamisega.

6 PROTOTÜÜBI ARENDUS

6.1 Elektroonika ühendused

Kui kõik vajalikud komponendid said valitud ja jõudsid ka minu kätte, oli esimeseks ülesandeks vaja teha vajalikud ühendused EnOcean Pi ja ESP32 vahel. Nendeks olid 2 toitejuhet ja 2 UART järjestikkommunikatsioon ühendusjuhet. TCM310 vajab töötamiseks 3,3 volti. ESP32 on võimeline enda 3,3 V siinilt välja andma kuni 1100 mA [20], mis on märkimisväärselt üle 33 mA, mida TCM310 maksimaalselt kasutab [21]. EnOcean Pi ühenduste kohta ei olnud infot ametlikel lehekülgedel. Vajamineva informatsiooni leidsin õiged ühendused Kerry D. Wongi blogist [22].



Joonis 6-1: TCM310 ühendused [23]

Ühendasin juhtmed laiendusmooduli poolt ära ja seejärel tegin ka ühendused ESP32 poolel. Toitejuhtmed tuli ühendada vastavalt 3,3 V ja GND viikude külge (Joonis 6-1), joonisel vastavalt pin1 ja pin2 ning UART ühenduse juhtmed TCM310 RX-pesast ESP32 TX2-pessa ja TCM310 TX-pesast ESP32 RX2-pesasse. ESP32 kiip omab kolme erinevat riistvaralist UART-liidest, ühte neist kasutame suhtluseks TCM310ga ning teist info kirjutamiseks Arduino IDE järjestikmonitori häälestamise faasis. Testperioodil saab ESP32 toite läbi USB kaabli arvutist, mis hiljem vahetatakse välja pingemuundur, mis teeb tööstusautomaatika standardsest 24 voldist ESP32-le sobiliku 5 volti.

6.2 EnOcean pakettide esialgne lugemine

Järgmisena katsetasin info kättesaamist EnOcean anduritelt. Selle jaoks kirjutasin lühikese koodi, kus kõik EnOcean Pi poolt saadetav info prinditakse Arudino IDE järjestikmonitori välja (Joonis 6-2). Selle info võrdlemiseks andis praktikajuhendaja kaasa USB300DB, mis on arvutiga ühilduv EnOcean pakette vastuvõttev seade. Pakettide vastuvõtmiseks kasutab see samuti TCM310 moodulit ja pakette saab lugeda programmist DolphinView Advanced [22].

Esialgses programmis olin sättinud ühenduse kiiruseks vale arvu, 9600 bit/s, mille olin ekslikult vaadanud ESP2 standardi järgi. Läbi selle jõudsid andmed küll kohale, kuid andmed ei olnud kuidagi sarnased DolphinView programmis nähtavate andmetega. Leides mõnda aega hiljem enda vea, muutsin ühenduskiiruse 57600 bit/s peale. Seejärel sain samu andmeid nii Arduino IDE kui ka DolphinView kaudu.

```
int Bait;

void setup() {
  Serial.begin(57600); // Alustab järjestikkommunikatsiooni kiirusega 57600 bit/s
  Serial2.begin(57600);
}

void loop() {
  if (Serial2.available() > 0) {
    Bait = Serial2.read();

    Serial.print("Lugesin: ");
    Serial.println(Bait, HEX);
  }
}
```

Joonis 6-2: Esialgne EnOcean pakettide kontrollkood

6.3 EnOcean pakettide vastuvõtmise tarkvara kirjutamine

Järgnevaks kirjutasin esmase tarkvara, mis loeks sisse EnOcean andurid ja näitaks õigel seda kujul Arduino IDEs. Suur väljakutse oli erineva suurustega andmepakettidest õigete andmete kättesaamine. Andmepakettide ajutiseks salvestamiseks kasutasin andmemassiivi nimega „pakett“. Andmemassiivi maksimaalseks pikkuseks seadsin 25 baiti, kuna selle lähteülesandes antud andurid suuremaid pakette ei edasta. Seejärel katsetasin EnOceani andmepakettide sisselugemist. Kasutades seinalülitit, mille

andmepaketi suurus oli 22 baiti, ei tulnud esimese vajutuse poolt saadetud andmed läbi. See oli põhjustatud koodireast, mis ei tuvastanud andmepakette, mis on väiksemad kui 25 baiti. Järgmise vajutusega salvestas see 3 esimest baiti massiivi viimasesse kolme kohta (22, 23, 24), kuvas paketti Arduino IDEs ja kirjutas info alates neljandast baidist massiivi „pakett" algusesse. Sellest tulenevalt tekkis vahepeal mitmeid pakette, mis tulid välja vigastena, kuna kontrollides esimest baiti massiivis, ei olnud see 0x55. Jõudes ringiga tagasi sellesse hetke, kus esimene pakett oli 0x55, oli võimalus jälle lugeda andurilt tulevat infot. Esialgu sai antud probleem lahendatud sellega, et massiivi suurus ja aktsepteeritava sissetuleva paketi suuruseks muuta 22 baiti. Seejärel oli võimalik väga edukalt sisse lugeda andurite andmepakette, mis olid suurusega 22 baiti. Võrreldes neid DolphinView programmist loetavate andmetega, olid andmed samad. Probleem tekkis aga siis, kui oli soov suuremaid ja väiksemaid andmepakette läbisegi lugeda. Näiteks Thermokon SR06 LCD ruumikontrolleri paketi pikkuseks oli 25 baiti. Saates neid pakette ennist kirjeldatud koodiga ESP32 mikrokontrollerile, tekkis samasugune uude massiivi kirjutamise probleem nagu varasemalt mainitud.

Tegin otsuse selle projekti raames lugeda sisse andmepakette suurusega 22-25 baiti. Selleks, et mikrokontroller saadaks edasi õiged andmed serverisse, pidin esialgu lugema sisse paketi bait number kahe alt, mitu baiti on andmepaketi algusest kuni sisuliste andmete lõpuni ja omistasin selle muutujale „n". Sisuliste andmete all mõtlen lüliti vajutust, temperatuuriandmeid, niiskusandmeid jne. Kuna ESP3 protokoll määratleb ära väga täpselt, mis andmed on enne ja pärast sisulisi andmeid ning mis kohtadel nad asuvad, sai muutujale „n" arve liites, kätte soovitud info. Näiteks anduri ID tuli koheselt pärast sisulisi andmeid ja oli 4 baiti pikk. See tähendas et anduri ID asus andmepaketis kohtadel n+1, n+2, n+3 ja n+4 ja need baidid omistasin ka muutujale ID. Anduri ID-d kontrollides, tuli välja see, et alati puudus selle algusest 0. Kõikide andurite, mida kasutasin arendamisel, ID-koodid algasid 0-iga. Lisasin iga ID algusesse ka numbri 0.

Kuna praeguse projekti raames olid kasutatavad andurid piiritletud, võimaldas see koodi kirjutada vastavalt anduritele. See tähendas seda, et määratlesin ära 4 erinevat juhtu: kui muutuja „n" suurus on 7, 8, 9 ja 10 ehk kui sisulisi andmeid oli vastavalt 1, 2, 3 või 4 baiti. Esimesel kolmel juhul kirjutasin sissetulnud andmed otse muutujasse „andmed" ilma neid muutmata. Ainukese erandi tegin Thermokon SR06 LCD ruumikontrolleriga, kus teisaldasin sissetulevad andmed inimloetavasse formaati ehk temperatuuri number on igapäevaselt kasutataval Celsiuse skaalal ja õhuniiskus muutus protsendilisele skaalale. Andmete teisendamise põhjuseks oli see, et MeiePilvel pole rakendust, mis töötleks sissetulevaid andmeid. Teisendamiseks lõin uue funktsiooni nimega „mapf()", mis lubab teisendada ka komakohtadega numbreid. Arduino IDE-sse sisseehitatud

funktsioon „map()" lubab teisendada ainult täisarve. Siinkohal võib tekkida probleeme teiste andurite andmepakettide edastusega, mis sisaldavad samuti 4 baiti sisulisi andmeid.

6.4 Andmete saatmine MeiePilv andmebaasi

Andmete saatmiseks MeiePilv võrgu OmaWiFi sisendliidesesse kasutasin TalTech Tartu kolledži õppeaine „Mikroprotsessorsüsteemid" materjale. Nimelt sealse õppeaine praktikumi raames saatsime samuti pakette MeiePilv andmebaasi.

Andmebaasi saatmine koosnes kahest tegevusest – esimene pool oli koodiosa nimega Byrokraat() (Joonis 6-3). See osa võttis sisse vajaliku info ja seejärel muutis selle õige ülesehitusega JSON-formaadis andmepaketiks (3.2).

```
String Byrokraat()
{
  String sI = "";                                  // Muutuja kogu teksti kogumiseks
  String rV = String('\n');                        // Abimuuutuja reavahetuse jaoks
  String jM = String('"');                         // Abimuutuja " jaoks, mis võib
                                                   //kompilaatoris erroreid tekitada
  // PÄIS
  sI = "{" + rV + jM + "device_name" + jM;         // jupp "device_name"
  sI += ": " + jM + ID + jM + "," + rV;          // lisab "<seadme ID>", rv
  sI += jM + "network_name" + jM + ": ";           // lisab "network_name":
  sI += jM + vork + jM + "," + rV + jM;            // lisab "<võrgu nimi>", rv"
  sI += "body" + jM + ":  [" + rV + " {" + rV; // lisab body":  [ rv  { rv

  // KEHAND
  sI += "  " + jM + "time" + jM + ": ";          // lisab "time":
  sI += String("0") + "," + rV;                  // lisab ajanihke (0)
  sI += "  " + jM + "values" + jM + ": ";        // lisab "values":
  sI += "[" + andmed + "]" + rV;   // lisab andmed muutuja
  sI += " }" + rV + " ]" + rV + "}" + rV;          // lõpetan kogu kirje

  return sI;
}
```

Joonis 6-3: Koodiosa „Byrokraat()"

Teine pool andmebaasi saatmisest oli koodiosa nimega Kirjatuvi(). Selle ülesandeks oli käivitada HTTP klient, määrata andmete saatmise sihtkoht, täpsustada saadetavate andmete sisu, need välja saata, kontrollida serveri vastust - kas pakett jõudis serverisse ja seejärel lõpetada HTTP klient (Joonis 6-4).

```
boolean Kirjatuvi(String info) {

  int httpVastuskood = 0;                    // muutuja http vastuskoodile
  boolean abi = false;                       // abimuutuja vaikimisi valeks

  HTTPClient http;                          // ... käivitan http kliendi
  http.begin(host, port, sUrl);             // Määran sihtkoha (oluline kirje)

  // Sean andmesisuks text/plain - edastatav info on tekst
  http.addHeader("Content-Type", "text/plain");
  httpVastuskood = http.POST(info);         // Muutujas "info" on tekst mida edastada
  Serial.println (httpVastuskood);
  if (httpVastuskood == 201)                // kui serveri vastuskood on 200, siis ...
  {
    abi = true; // saatmine õnnestus ja funktsioon saab olema true ...,
  }
  // ... muidu mitte
  http.end();                               // Vabastan serveri
  return abi;                               // funktsiooni väärtuseks abimuutuja väärtus
};                                           // funktsiooni lõpp
```

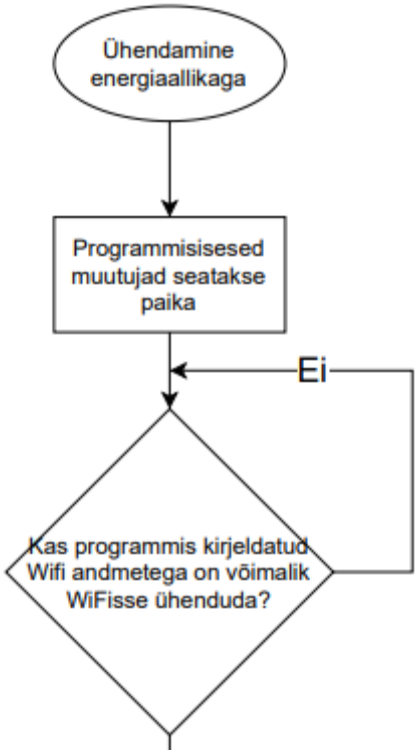
Joonis 6-4: Koodiosa „Kirjatuvi()"

6.5 Lõpliku koodi selgitus

Tarkvara tööle minemiseks on esialgu vaja riistvarale ühendada energiaallikas. Seejärel seatakse tarkvaraliselt paika programmisisesed muutujad ja integreeritakse vajaminevad teegid. Nendeks muutujateks on:

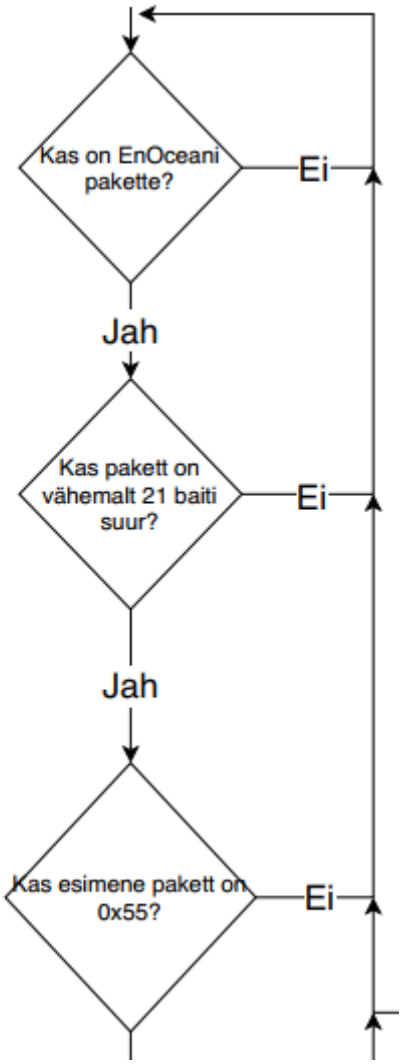
- paketi andmemassiiv
- paketi pikkuse muutuja
- muutuja niiskuse, temperatuuri ja temperatuurimuutuse jaoks (SR06 LCD jaoks)
- anduri ID-koodi muutuja
- konstandid ühendatava WiFi serveri nime ja parooli kohta
- MeiePilv andmebaasi võrgu muutuja
- saadetava serveri IP ja pordi muutujad
- muutujad „kiri" ja „ajutineAbimuutuja" serverisse saatmiseks
- andmete koondamiseks muutuja „andmed"

Seejärel alustatakse järjestikühendused kiirusega 57600 bit/s. Üks neist on andmete kättesaamiseks EnOcean Pi-lt ja teine Arduino IDE järjestikmonitori printimiseks. Samuti luuakse ühendus WiFiga. (Joonis 6-5)



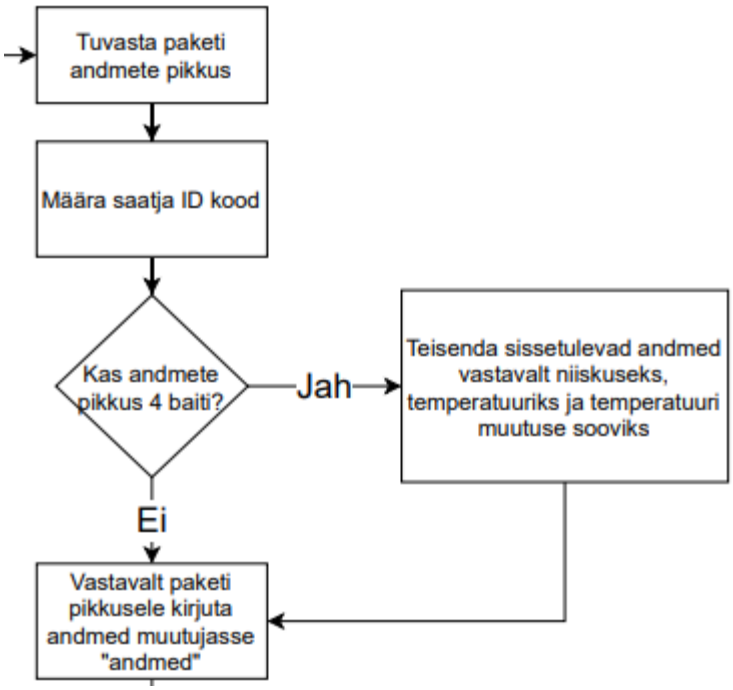
Joonis 6-5: Koodi esialgne käivitumine

Järgnevalt ootab kood sissetulevaid EnOcean pakette. Kui sissetulev pakett on EnOceani pakett, kontrollitakse tema pikkust, mis peab olema üle 21 baidi ja samuti esimest baiti, mis peab olema standarti järgi 0x55. Kui kumbki neist ei ole sobilik, liigub kood tagasi faasi, kus ta ootab uut EnOceani paketti. (Joonis 6-6)



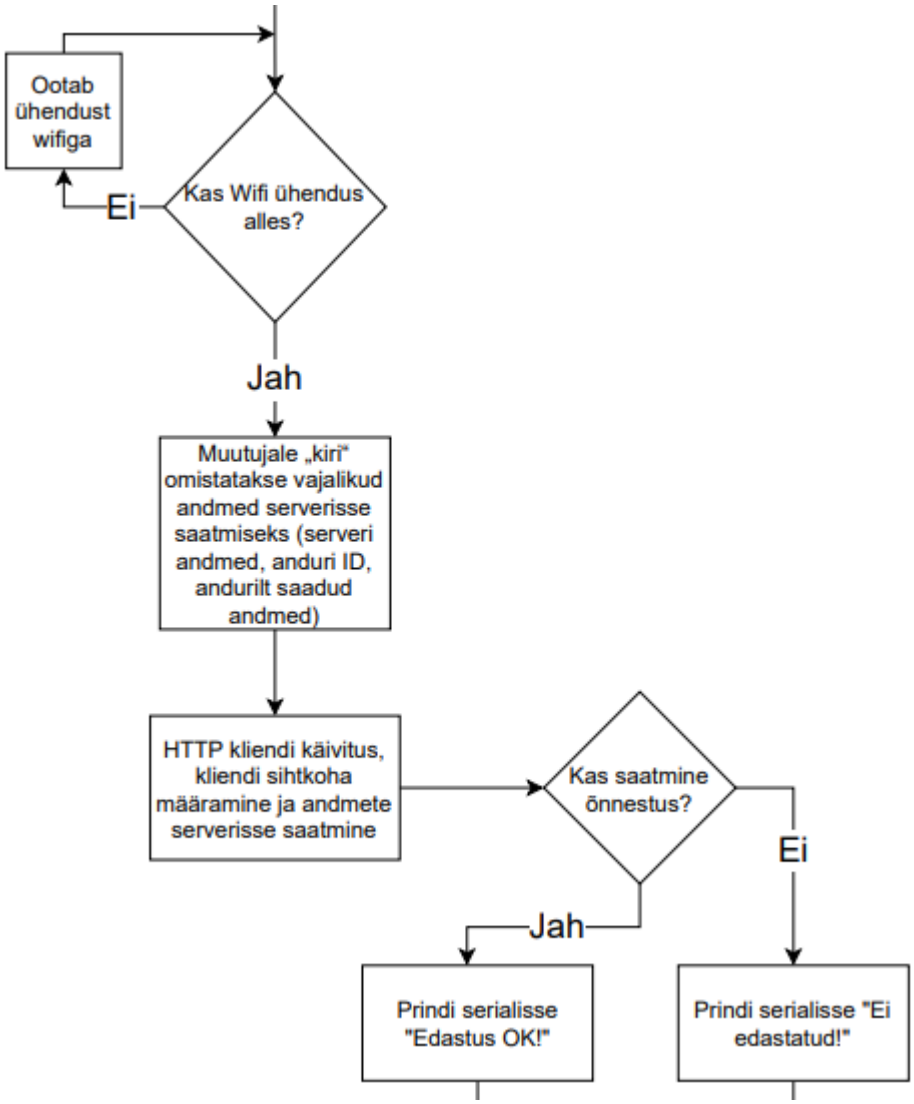
Joonis 6-6: Sissetuleva paketi kontroll

Kui pakett läbis eelneva kontrolli, tuvastatakse paketis sisalduvate andmete pikkus. Seejärel määratakse saatja ID-kood ja vastavalt paketi pikkusele kirjutatakse muutujasse „andmed" anduri poolt saadetavad info. Juhul, kui sisuliste andmete pikkuseks on 4 baiti, teisendatakse need andmed vastavalt niiskuseks, temperatuuriks ja temperatuurimuutuse sooviks. (Joonis 6-7)



Joonis 6-7: Sisuliste andmete kontroll ja kirjutamine muutujasse "andmed"

Järgnevalt kontrollib kood WiFi ühenduse olemasolu. Selle puudumisel ei liigu kood enne edasi, kuni WiFi ühendus on uuesti loodud. Ühenduse olemasolul omistatakse muutujale „kiri" vajalik info serverisse saatmiseks. Selleks infoks on serveri andmed, anduri ID-kood ja sellelt saadud sisulised andmed. Seejärel käivitatakse HTTP klient, määratakse kliendi sihtkoht ja saadetakse info serverisse. Õnnestunud saatmise puhul prinditakse järjestikmonitori „Edastus OK!" ja ebaõnnestunud saatmise korral „Ei edastatud!". Seejärel liigub kood tagasi algusesse ja ootab järgmist EnOcean paketti. (Joonis 6-8)



Joonis 6-8: Andmete saatmine serverisse

Koodi töötamise terviklik skeem asub lisades.

7 VÕIMALIKUD ARENGUSUUNAD

7.1 Andurite andmete talletamine püsimälusse

Vältimaks serveri ülekoormust, oleks võimalik lisada vastuvõtjale SD-kaardi moodul. See võimaldaks vastuvõtuväraval andureid ID-koodi järgi liigitada. See tähendaks, et näiteks varasemalt tehes valmis .csv-faili ja salvestades kõikide kasutatavate andurite ID-koodid ja andmete teisendused sinna faili, saadab vastuvõtuvärav serverisse alati ainult selles failis asuvate andurite andmed ja need teisaldatakse vastavalt anduri tüübile vastuvõtuväravas. Samuti saab muuta ka serveripoolseks tuvastamiseks mõeldud „device_name" andmevälja, kuhu saaks luua enda poolt väljavalitud süsteemi andurite nimede kohta. Need koodnimed võiksid sisaldada endas näiteks ruuminumbrit, kus nad asuvad, anduri järjekorranumbrit ja selle tüüpi. Edasine areng sellest oleks andmelehe muutmine interneti kaudu kasutades WiFi ühendust, mida ka ESP32 võimaldab.

Anduritele koodnimede määramine enne serverisse saatmist aitab kaasa andmebaasi järjepidevusele. Kui tulevikus peaks juhtuma, et mõni andur on vigane ja vajab väljavahetamist, võimaldab antud lahendus püsimälusse kirjutatud koodnimele vastava ID-koodi muutmist. Uuelt andurilt tulev info saadetakse vana anduri andmete järgi ja hiljem ei pea ühe ruumi andmeid otsima mitmest erinevast kohast.

KOKKUVÕTE

2015. aastal lisandus tollasesse Tallinna Tehnikaülikooli Tartu kolledžisse uus õppekava nimega Telemaatika ja arukad süsteemid. Õpetatavaks peaerialaks oli küberfüüsikaline süsteemitehnika (hiljem küberfüüsikalised süsteemid), mille raames tehti algust innovaatilise õpikeskkonna, Eluslaboratiooriumi, rajamist. Selle põhieesmärgiks on rikastada tudengite õpitööd läbi praktiliste kogemuste. Sellega seoses võeti ka kasutusele EnOcean andurid ja lülitid, mistõttu oli vaja EnOcean andmepakettidele vastuvõtuväravat. Vastuvõtuvärava ülesandeks oli saata EnOceani anduritelt ja lülititelt tulev info saata kohalikku kolledži andmebaasi nimega MeiePilv.

Vastuvõtuvärava valikul olid tähtsateks aspektideks selle andmebaasiga ühilduvus, suurus, hind ja võimalus teha muudatusi lähtekoodi. Neid tingimusi arvestades valisin lahenduseks mikrokontroller ESP32-WROOM-32 koostöös EnOcean pakettide vastuvõtjaga EnOcean Pi. See kooslus võimaldas kirjutada ise mikrokontrollerile lähtekoodi ja valida ka sellele sobiv toiteallikas. Samuti on antud kooslusel väga madal energiatarve.

Järgnevaks pidin ühendama omavahel ESP32 ning EnOcean Pi. Omavaheliseks suhtluseks kasutavad nad UART ühendust. Ühendades õiged viigud kontrolleri ja EnOcean Pi vahel, tuli esmalt kontrollida EnOceani pakettide kättesaamist. Pakettide eduka lugemise järel vajas mikrokontroller lähtekoodi eristamaks erinevaid andureid ja nende paketipikkuseid ja seejärel eraldamaks nendest pakettidest vajaliku info. Info andmebaasi saatmiseks lisasin tarkvarasse varasemalt kolledži praktikumis kasutatud koodiosa, mille abil esialgu loodi õige ülesehitusega pakett, siis avati ühendus serveriga ja saadeti andmed serverisse. Paketi eduka saatmise puhul tuli ka vastav vastus serverilt.

Antud töö tulemustega olen ise rahul. Olles varem kasutanud Arduino IDE programmeerimiskeskkonda, arendas antud koodi kirjutamine siiski minu koodikirjutamise loogikat. Samuti õppisin väga palju uut raadioühenduse EnOcean kohta, millest enne lähteülesande saamist polnud kuulnud. Kuigi antud vastuvõtuväravat saaks arendada edasi nii koodi, poolt kui ka lisades talle uusi võimalusi näen, et esialgne töö pakettide serverisse saatmiseks on tehtud ja siit edasi arenedada on juba lihtsam.

SUMMARY

In 2015, a new curriculum called Telematics and Intelligent Systems was added to the then Tallinn University of Technology Tartu College. The main specialty taught was cyberphysical systems engineering (later cyberphysical systems). At the same time an innovative learning environment, Eluslaboratoorium, was started. Its main goal is to enrich students' learning through practical experience. In this context, EnOcean sensors and switches were also introduced, necessitating a receiving gateway for EnOcean data packets. The task of the reception gate was to send the information from EnOcean's sensors and switches to a local college database called MeiePilv.

Important aspects when choosing a gateway were compatibility with its database, size, price, and the ability to make changes to the source code. Considering these conditions, I chose the microcontroller ESP32-WROOM-32 as a solution in cooperation with the EnOcean packet receiver EnOcean Pi. This combination made it possible to write the source code on the microcontroller itself and to select a suitable power supply for it. The community also has a very low energy consumption.

Next I had to connect ESP32 and EnOcean Pi. They use a UART connection to communicate with each other. By connecting the correct pins between the controller and EnOcean Pi, the receipt of EnOcean packets had to be checked first. After successfully reading the packets, the microcontroller needed source code to distinguish between the different sensors and their packet lengths, and then to extract the necessary information from those packets. To send the information to the database, I added the code snippet previously used in the college workshop, which was first used to create a packet with the correct structure, then the connection to the server was opened and the packet was sent. If the packet was sent successfully, the corresponding response was received from the server.

I am satisfied with the results of this work. Although having previously used the Arduino IDE programming environment, writing this code developed my code-writing logic. I also learned a lot about the radio connection EnOcean, which I hadn't heard about before getting the assignment. Although this gateway could be further developed both by the code and by adding new features to it, I see that the initial work of sending packets to the server has been done and it is easier to develop from here.

KASUTATUD KIRJANDUSE LOETELU

[1] „About EnOcean," [Võrgumaterjal]. Available: https://www.enocean.com/en/about-us/.

[2] „EnOcean Alliance," [Võrgumaterjal]. Available: https://www.enocean-alliance.org/about-us/.

[3] A. Ståhlbröst ja M. Holst, 2012. [Võrgumaterjal]. Available: https://www.ltu.se/cms_fs/1.101555!/file/LivingLabsMethodologyBook_web.pdf.

[4] P. Evans, D. Schuurman, A. Ståhlbröst ja K. Vervoort. [Võrgumaterjal]. Available: https://u4iot.eu/pdf/U4IoT_LivingLabMethodology_Handbook.pdf.

[5] „EnOcean Techhnology Energy Harvesting," [Võrgumaterjal]. Available: https://www.enocean.com/en/technology/energy-harvesting/.

[6] EVS, „EnOcean sidestandart," [Võrgumaterjal]. Available: https://www.evs.ee/et/iso-iec-14543-3-10-2020.

[7] „EnOcean andur päikesepatareiga," [Võrgumaterjal]. Available: https://download.schneider-electric.com/files?p_Doc_Ref=ODALSS10020032&p_File_Type=rendition_288_png&default_image=DefaultProductImage.png.

[8] EnOcean, „EnOcean Serial Protocol 3," [Võrgumaterjal]. Available: https://www.enocean.com/esp.

[9] EnOcean, „EnOcean lüliti PTM210 DB," [Võrgumaterjal]. Available: https://www.enocean.com/en/products/enocean_modules/ptm-210ptm-215/data-sheet-pdf/.

[10] Thermokon, „Thermokon SR06 LCD," [Võrgumaterjal]. Available: https://www.thermokon.de/en/products/easysensr-transmitter/room-operating-units/sr06-lcd/.

[11] „Schneider Electric magnetandur," [Võrgumaterjal]. Available: https://www.se.com/ee/et/product/LSS10020032/ecostruxure-building-expert-enocean-room-occupancy-sensor/.

[12] „EnOcean Pi," [Võrgumaterjal]. Available: https://www.element14.com/community/docs/DOC-55169/l/enocean-pi-transforms-raspberry-pi-into-a-wireless-gateway.

[13] „Pressac EnOcean Gateway," [Võrgumaterjal]. Available: https://www.enocean-alliance.org/product/smart-gateway/.

[14] „Raspberry Pi," [Võrgumaterjal]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/?resellerType=home.

[15] „Thermokon EnOcean Gateways," [Võrgumaterjal]. Available: https://www.thermokon.de/en/products/easysensr-receiver/gateways/.

[16] „WinShine LD-GateWay," [Võrgumaterjal]. Available: https://www.enocean-alliance.org/wp-content/uploads/2017/11/LD-GateWay_EN.pdf.

[17] A. Maier, A. Sharp ja Y. Vagapov, „ESP32," [Võrgumaterjal]. Available: https://ieeexplore.ieee.org/abstract/document/8101926?casa_token=5S_I9D0IEDYAAAAA:rHjDsQv9sLPNLG91dr6uh1VTqZQJ-JF7yk1hYX_qLDErrzL_4W9OtkJycjSt2Q0Qq5j9m-lmnA.

[18] „EspressIF ESP8266," [Võrgumaterjal]. Available: https://www.espressif.com/en/products/socs/esp8266.

[19] „ESP32 pilt," [Võrgumaterjal]. Available: https://images-na.ssl-images-amazon.com/images/I/61RixSStGBL._AC_SX466_.jpg.

[20] E. datasheet, „https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf," [Võrgumaterjal].

[21] T. datasheet. [Võrgumaterjal]. Available: https://www.enocean.com/en/products/enocean_modules/tcm-310/data-sheet-pdf/.

[22] „EnOcean USB300DB," [Võrgumaterjal]. Available: https://www.enocean.com/en/products/enocean_modules/usb-300/.

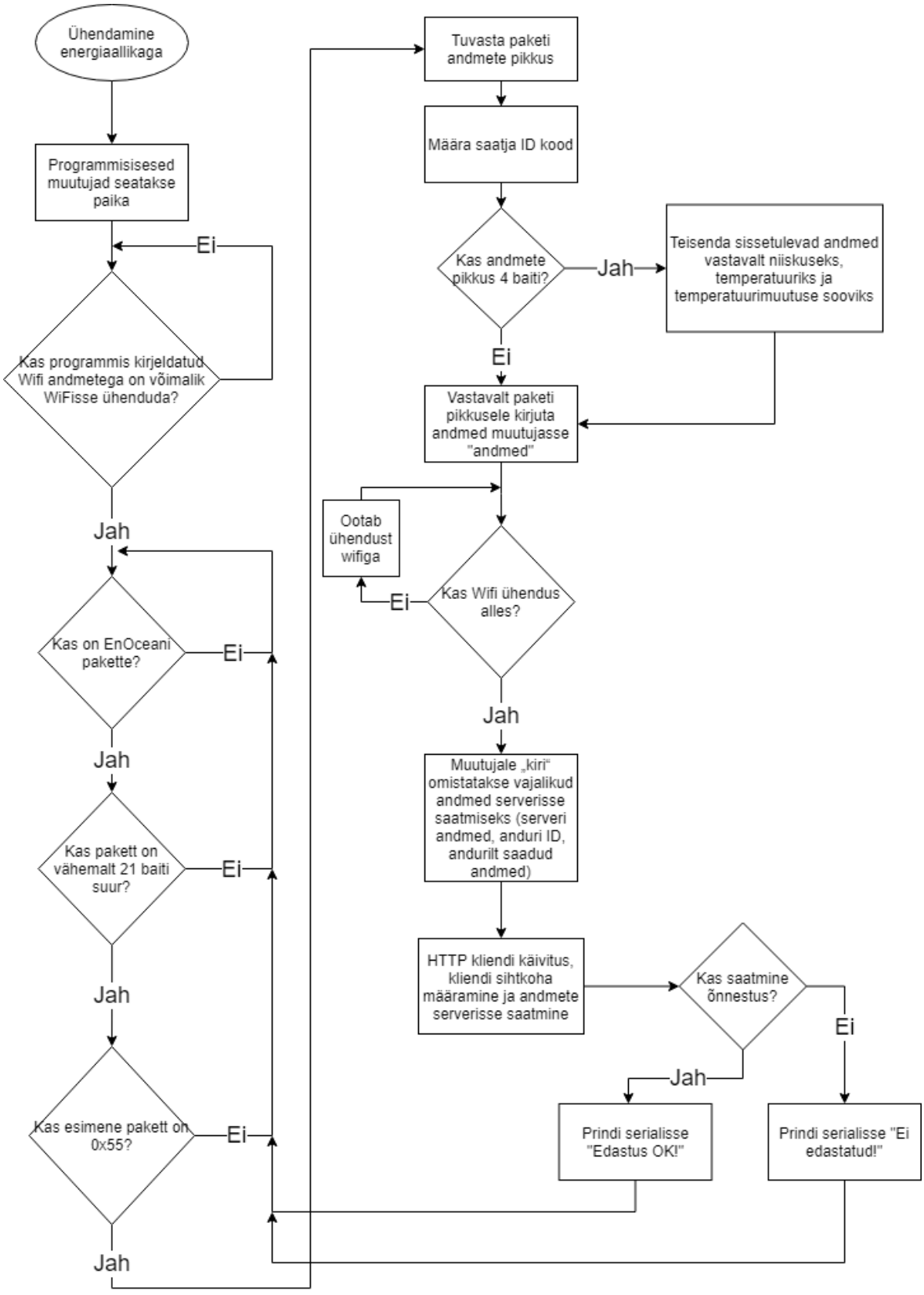
[23] K. Wong, 10 August 2014. [Võrgumaterjal]. Available: http://www.kerrywong.com/2014/08/10/a-quick-overview-of-the-enocean-pi-and-enocean-sensor-kit/.

[24] „https://duino4projects.com ESP32," [Võrgumaterjal]. Available: https://duino4projects.com/esp32-wroom-32-esp-wroom-32/#noreferrer%20noopener/8/.

[25] „EnOcean Pi pilt," [Võrgumaterjal]. Available: https://asset.conrad.com/media10/isa/160267/c1/-/en/1000221_RB_00_FB/image.jpg.

LISAD

Lisa 1: Koodi töötamise skeem



```
#include <HTTPClient.h>

#include <WiFi.h>

int pakett[24]; //vastuvõetava paketi massiiv EnOceani andurilt
int n; //paketi pikkuse muutuja

float niiskus; //muutuja niiskuse jaoks
float temperatuur; //muutuja temperatuuri jaoks
float muutus; //Temperatuuri muutuse number (vahemik 0-255), 127 tähendab et
muutust ei soovita
String ID; //Anduri ID-koodi muutuja

const char* ssid = "Lisa Wifi nimi"; //WiFi serveri nimi
const char* pass = "Lisa Wifi parool"; //WiFi serveri parool
const char* sUrl = "/measurements/objects"; //Serverisse kirjutamiseks vajalik rida

const char* vork = "OmaWiFi";// Võrguliigi nimi serveri jaoks.

const char * host = "193.40.13.86"; //Serveri, kuhu andmed saadetakse, IP
const uint16_t port = 82; //Serveri, kuhu andmed saadetakse, port

boolean ajutineAbimuutuja;
String kiri;

String andmed;

void setup()
{
  Serial.begin(57600); //Alustame serialit kuhu prinditakse infot töötamise ajal
  Serial2.begin(57600); //Alustame serialit sisselugemiseks
  Serial.println("EnOcean TCM310 lugeja ning MeiePilve serverisse saatja");
  delay(10);

  WiFi.begin(ssid, pass); // WiFiga ühenduse loomine
  Serial.print("Connecting.");
  while (WiFi.status() != WL_CONNECTED) { //Nii kaua kuni ühendust ei saavutata, ei
liigu kood edasi.
    Serial.print(".");
    delay(500);
  }

  Serial.print("WiFi ühendatud - IP aadress: "); //Annab märku ühendatud WiFist ja
kontrollerile omistatud IP aadressist
  Serial.println(WiFi.localIP());
  delay(500);
}

void loop() {
  while ( (Serial2.available() < 21)) //Ootab üle 21 baidi pikkust andmepaketti
  {}
  if (Serial2.available() < 21) //Juhul kui pakett peaks olema väiksem kui 21 baiti
  {
    Serial.println("ERROR - Ei saanud vähemalt 21 baidiga andmepaketti!");
  } else {
```

```
   for (int j = 0; j < 24; j++) {
     pakett[j] = Serial2.read(); // Loeb serialist baite
     if (pakett[0] == 0x55) { //Kontrollib esimese paketi järgi kas tegemist EnOceani
anduriga
       n = (pakett[2]); //Omistab muutujale n paketi suuruse algusest kuni anduri info
lõpuni
       ID = ("0" + String(pakett[n + 1], HEX) + String(pakett[n + 2], HEX) +
String(pakett[n + 3], HEX) + String(pakett[n + 4], HEX)); //Omistab muutujale ID
anduri ID-koodi
       ID.toUpperCase(); //Muudab anduri ID-koodis olevad tähed suurteks

       //Bait 0: SYNC
       //Bait 1-2: Length
       //Bait 3: OpLength
       //Bait 4: Type
       //Bait 5: CRC8H
       //Bait 6: RORG
       //Bait 7: andmed
       //Bait (n+1): Sender ID
       //Bait (n+5): Status
       //Bait (n+7): SubTel#
       //Bait (n+8): Dest ID
       //Bait (n+11): dBm
       //Bait (n+12): Security
       //Bait (n+13): CRC8D
     } else {
       Serial.println("Error: Not a valid sensor telegram"); //Kui paketi esimene bait ei
ole 0x55
     }
     Serial.println(String(j) + String(": ")+ String(pakett[j], HEX)); //Paketi andmete
kontrollimiseks
   }

   if (n == 7) { //Kui paketi pikkus algusest kuni info lõpuni on 7
     Serial.println(ID);//kontrolliks prindib välja
     Serial.print(pakett[7], HEX);
     Serial.println("");

     pakett[22] = 0; //Kuna antud pakett on ainult 21 ühikut pikk, muuda 3 viimast
muutujat 0
     pakett[23] = 0;
     pakett[24] = 0;

     andmed = (String(pakett[7], HEX)); //Omista muutujale andmed anduri poolt
saadetud info kuueteistkümnendiksüsteemis
   }

   if (n == 8) {
     Serial.println(ID);
     Serial.print(pakett[7], HEX);
     Serial.print(", ");

     Serial.print(pakett[8], HEX);
     Serial.println("");

     pakett[23] = 0;
     pakett[24] = 0;
```

```
    andmed = (String(pakett[7], HEX) + String(", ") + String(pakett[8], HEX));
   }

   if (n == 9) {
    Serial.println(ID);
    Serial.print(pakett[7], HEX);
    Serial.print(", ");

    Serial.print(pakett[8], HEX);
    Serial.print(", ");

    Serial.print(pakett[9], HEX);
    Serial.println("");

    pakett[24] = 0;

    andmed = (String(pakett[7], HEX) + String(", ") + String(pakett[8], HEX) +
String(", ") + String(pakett[9], HEX));
   }

   if (n == 10) {
    Serial.println(ID);

    niiskus = mapf(pakett[8], 0, 250, 0, 100); //muudab andurilt saadetavad
niiskusanduri andmed (0-250) õigele skaalale (0-100%)
    Serial.print(String("Niiskus: ") + String(niiskus)); //kontrolliks prindib välja
    Serial.print(", ");

    temperatuur = mapf(pakett[9], 0, 250, 0, 40); //muudab andurilt saadetavad
temperatuuri andmed (0-250) õigele skaalale (0-40°C)
    Serial.print(String("Temperatuur: ") + String(temperatuur));
    Serial.print(", ");

    muutus = (pakett[7]); //omistab muutujale muutus numbri vahemikus 0-255. Kui
muutust ei soovita, on arv 127
    Serial.print(String("Muutus: ") + String(muutus));

    andmed = (String(muutus) + String(", ") + String(niiskus) + String(", ") +
String(temperatuur)); //Omista muutujale andmed anduri poolt saadetud info
muudetud kujul
    Serial.println("");
   }

   if (WiFi.status() == WL_CONNECTED) {     // Kontrollib ühendust WiFiga
    kiri = Byrokraat();                  // Omistab muutujale "kiri" vormistatud kirje
saatmiseks
    Serial.println(kiri);                // Kontrolliks prinditakse serialisse
    ajutineAbimuutuja = Kirjatuvi(kiri);  // saadan info serverisse

    if (ajutineAbimuutuja) {
     Serial.println("Edastus OK!"); //Kui funktsioon on true, saatmine õnnestus
    } else {
     Serial.println("Ei edastatud!"); //Kui aga mitte, siis saatmine ebaõnnestus
    }
   } else { //Kui Wifi ühendus kadunud
    Serial.println("Wifi ühendus kadunud. Ühendus luuakse uuesti.");
```

```
    WiFi.begin(ssid, pass); //Ühenda uuesti

    Serial.print("Connecting.");
    while (WiFi.status() != WL_CONNECTED) {
      Serial.print(".");
      delay(500);
    }
    kiri = Byrokraat();                 //Saada pakett uuesti
    Serial.println(kiri);
    ajutineAbimuutuja = Kirjatuvi(kiri);

    if (ajutineAbimuutuja) {
      Serial.println("Edastus OK!"); //Kui funktsioon on true, saatmine õnnestus
    } else {
      Serial.println("Ei edastatud!"); //Kui aga mitte, siis saatmine ebaõnnestus
    }
   }
  }
}

String Byrokraat()
{
  String sI = "";                          // Muutuja kogu teksti kogumiseks
  String rV = String('\n');                  // Abimuuutuja reavahetuse jaoks
  String jM = String('"');                   // Abimuutuja " jaoks, mis võib kompilaatoris
erroreid tekitada
  // PÄIS
  sI = "{" + rV + jM + "device_name" + jM;     // jupp "device_name"
  sI += ": " + jM + ID + jM + "," + rV;      // lisab "<seadme ID>", rv
  sI += jM + "network_name" + jM + ": ";       // lisab "network_name":
  sI += jM + vork + jM + "," + rV + jM;        // lisab "<võrgu nimi>", rv"
  sI += "body" + jM + ":  [" + rV + " {" + rV; // lisab body":  [ rv  { rv

  // KEHAND
  sI += "  " + jM + "time" + jM + ": ";      // lisab "time":
  sI += String("0") + "," + rV;             // lisab ajanihke (0)
  sI += "  " + jM + "values" + jM + ": ";    // lisab "values":
  sI += "[" + andmed + "]" + rV;   // lisab andmed muutuja
  sI += " }" + rV + " ]" + rV + "}" + rV;      // lõpetan kogu kirje

  return sI;
}

boolean Kirjatuvi(String info) {

  int httpVastuskood = 0;                  // muutuja http vastuskoodile
  boolean abi = false;                    // abimuutuja vaikimisi valeks

  HTTPClient http;                       // ... käivitan http kliendi
  http.begin(host, port, sUrl);           // Määran sihtkoha (oluline kirje)

  // Sean andmesisuks text/plain - edastatav info on tekst
  http.addHeader("Content-Type", "text/plain");
  httpVastuskood = http.POST(info);       // Muutujas "info" on tekst mida edastada
  Serial.println (httpVastuskood);
  if (httpVastuskood == 201)              // kui serveri vastuskood on 200, siis ...
  {
```

```
    abi = true; // saatmine õnnestus ja funktsioon saab olema true ...,
  }
  // ... muidu mitte
  http.end();                          // Vabastan serveri
  return abi;                          // funktsiooni väärtuseks abimuutuja väärtus
};                                    // funktsiooni lõpp

float mapf(float x, float in_min, float in_max, float out_min, float out_max)
//Mappimis funktsioon float tüüpi muutujatele (standardis toimib map funktsioon
ainult int tüüpi muutujale)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```