TARTU ÜLIKOOL
Arvutiteaduse instituut
Infotehnoloogia mitteinformaatikutele õppekava

Enn Ehrlich

ESP32 süsteemikiibil põhineva kaughallatava seireseadme välja töötamine

Magistritöö (15 EAP)

Juhendaja: Alo Peets

Tartu 2024

ESP32 süsteemikiibil põhineva kaughallatava seireseadme välja töötamine

Lühikokkuvõte:

COVID-19 kriisi kontekstis on Euroopas ja Eestis oluliselt kasvanud pakiautomaatide kasutamine. Seadmeid hooldaval ettevõttel oli tekkinud vajadus kliendi seadmetele ligipääsuks elektrikatkestuste ja sidekatkestuste puhuks. Lõputöö eesmärgiks oli projekteerida ja valmis ehitada temperatuuri, õhuniiskust ja elektrivõrgu parameetreid edastav kaughalduse seade. Töö tulemusena valmis seade, mis põhineb ESP32 süsteemikiibil ja kasutab GPRS standardile vastavat mobiilset andmeside. Seade kasutab temperatuuri ja õhuniiskuse mõõtmiseks DHT22 andurit. Vahelduvvoolu, pinget ja elektrivõrgu sagedust mõõdetakse Peacefair PZEM-004T seadmega. Kogutud andmetele lisatakse riistvaralisest reaalajakellast ajatempel ja andmed edastatakse kesksesse serverisse krüpteeritud MQTT protokolli kaudu. Kliendi seadmete juhtimiseks on seadmel kaks väljundreleed. Ühenduse katkemisel keskse serveriga salvestatakse andmed Micro SD kaardile. Seadme sisemine UPS tagab autonoomse töö 10 tunni jooksul. ESP32 süsteemikiibi püsivara lähtekood on kirjutatud C-keeles. Valminud seade vastab esitatud nõuetele ja seda on edukalt testitud reaalsetes olukordades.

Võtmesõnad:

IoT, MQTT, PKI, ESP32, GPRS , DHT 22, PZEM-004T, Arduino, kaughaldus, süsteemikiip

CERCS: T180 Telekommunikatsioonitehnoloogia; P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Remotely Managed Monitoring Device Based on ESP32

Abstract:

Due to the COVID-19 crisis, the use of parcel machines has increased significantly in Europe and Estonia. The company that maintains the equipment has developed a need for access to the client's equipment in the event of power outages and communications outages. The aim of the thesis is to design and build a remote management device that enables remote monitoring. The resulting device is created on the ESP32 System on Chip and uses a mobile connection according to the GPRS standard. The device uses a DHT 22 sensor to measure temperature and humidity. The alternating current, voltage and power network frequency are measured with the Peacefair PZEM-004T device. The resulting data shall be accompanied by a time stamp from a hardware real-time clock and transmitted to the central server via an encrypted MQTT protocol. To control the client's devices, the device has two output relays. In the absence of connection to the central server, the data is stored on the Micro SD card. The internal UPS of the device ensures autonomous operation within 10 hours. The ESP32 microcontroller firmware source code is written in C-language.

Keywords: IoT, MQTT, PKI, ESP32, GPRS, DHT 22, PZEM-004T, Arduino, Remote monitoring, SoC

CERCS: T180 Telecommunication engineering; P170 Computer science, numerical analysis, systems, control

Sisukord

1. Sissejuhatus

Peale Covid-19 epideemiat on Euroopas ja eriti Eestis pakiautomaatide paigaldamine ja kasutamine oluliselt suurenenud. Üleeuroopalisest uuringust[1] selgub, et Eesti on 2023. aasta alguse seisuga 9. kohal pakiautomaatide arvult elaniku kohta. Oluliselt on suurenenud ka pakiautomaatide hooldusmahud. Eestis pakiautomaate hooldava ettevõte nimekirjas oli 2024 aasta alguse seisuga 850 automaati. Paljud hoolduses olevad seadmed asuvad keskustest kaugel ja raskesti ligipääsetavates kohtades. Teede hoolduseks kulutatavate vahendite vähendamine, sagedased elektrikatkestused ja tõsised puudujäägid Elektrilevi töös[2] on muutnud seadmete reaalajas jälgimise ettevõtte jaoks oluliseks teemaks. Iga kaugteel lahendatud intsident tähendab olulist ajalist ja materiaalset kokkuhoidu. Eeltoodut arvesse võttes tekkis ettevõttel vajadus kaughallatavate seadmete järele. Ajendiks ettevõttesiseselt kaughalduseseade projekteerida oli tehnilistele nõudmistele vastava tööstuslikult toodetud seadme puudumine. Seadme olulisteks omadusteks pidid olema autonoomsus kliendi seadmetest, sõltumatu ühendus IoT (*Internet of Things*) serveriga ja avatud lähtekoodiga tarkvara kasutamine. Kuna kliente puudutav informatsioon on konfidentsiaalne, oli oluline tingimus andmete krüpteerimine. Tarkvaralise lahenduse poolest oli üheks nõudeks IoT keskserveri haldamine oma ettevõtte siseselt. Kaasaegsed nn pilvelahendused ei sobinud küsitava turvalisuse ja ettearvamatu hinnastamise pärast[1]. Turvalisuse ja andmete kaitstuse seisukohalt ei sobinud kasutamiseks paljud turul saadaolevad sobiva hinnaklassiga tööstuslikud seadmed[3]. Peatükis 3 on toodud täpsed nõuded riist- ja tarkvarale. Lisaks riistvaralisele poolele valmis lõputöö käigus süsteemi kasutamist kirjeldav tehniline dokumentatsioon. Laiemale sihtgrupile mõeldes on osad teemad põhjalikumalt kirjeldatud ja lisatud lähtekoodi selgitavad näited. Avatud lähtekoodi kasutades on seadme tööpõhimõtet ja lähtekoodi lihtsam selgitada klientidele, kelle seadmeid jälgitakse.

Töö lõpptulemusena valmis riist- ja tarkvarast koosnev lahendus. Riistvara keskse seadmena on kasutusel ESP32 süsteemikiip ja SIM800 GPRS modemil põhinev arendusplaat, millele on lisatud erinevad moodulid. Seadmete haldamiseks ja andmete salvestamiseks sai valitud ThingsBoard[4] IoT serveri tarkvara. Andmebaasi salvestatakse kliendi seadme voolutarve, elektrivõrgu nimipinge, tarbitav võimsus, elektrivõrgu sagedus. Lisaks veel seadet ümbritsev temperatuur ja niiskus. Kaugteel on võimalik ka kliendi seadmeid releemooduli abil alglaadida. Seadmete poolt edastatud andmed salvestatakse PostgreSQL andmebaasi.

Andmebaas võimaldab:

- planeerida paremini tegevusi, kiiremini reageerida
- esitada kliendile selgitusi ja aruandeid
- andmete analüüsi masinõppe meetoditega
- rakendada ennetavat hooldust

[1] https://lastmileexperts.com/wp-content/uploads/2023/06/Out-of-home-delivery-in-Europe-2023_PUDOs-and-automated-parcel-machines-report_v1_1.pdf (03.03.2024)
[2] https://www.konkurentsiamet.ee/uudised/konkurentsiamet-tegi-elektrilevile-ettekirjutuse-seoses-ulatuslike-elektrikatkestustega (02.03.2024)
[3] https://www.valisluureamet.ee/doc/raport/2024-et.pdf (02.03.2024)
[4] https://thingsboard.io/ (03.03.2024)

Valminud lõputöö algab tehniliste tingimuste kirjeldamisest loodavale seadmele. Sellele järgneb loetelu sarnase funktsionaalsuse ja hinnaklassiga tööstuslikult toodetud seadmetest. Enamuse osa tööst haarab ülevaade seadme loomiseks kasutatud riist- ja tarkvarast. Kokkuvõttes on kirjeldatud valminud seadme vastavust esitatud nõuetele ja edasisi tegevusi. Lisadest leiab IoT serveri paigaldamise ja seadme kasutamise materjalid.

2. Mõisted, terminid ja lühendid

API	*Application Programming Interface,* rakendusliides
API token	API pääse, rakendusepääse, ühene identifikaator juurdepääsu taotleva rakenduse autentimiseks, parooli analoog
Arduino	Avatud lähtekoodiga programeerimise platvorm
CoAP	*Constrained Application Protocol*
Debugging	Tõrkeanalüüs, arvutisüsteemide tõrgete analüüsimine
Firmware	Püsivara, programmikoodi või andmeid sisaldav riistvara, mille sisu ei saa lõppkasutaja keskkonnas muuta
GPIO	*General Purpose Input/Output*
GPRS	*General Packet Radio Service*, GSM võrgu kaudu toimuv andmeedastus.
HTTP	*HyperText Transfer Protocol*, protokoll teabe edastamiseks arvutivõrkudes
I2C	*Inter Integrated Circuit*
IDE	*Integrated Development Environment* , programmeerimise keskkond, lugemis- ja kirjutusvõimeline rakendus andmeüksuste loomiseks ja muutmiseks
IoT	*Internet of Things*, asjade Internet, värkvõrk
JSON	*JavaScript Object Notation*
LAN	Local Area Network, kohtvõrk
LPWAN	*Low Power Wide Area Network*, väikese energiatarbega aeglane traadita laivõrk.
LTE-M	*Long-Term Evolution Machine Type Communication,* energiasäästlik LTE võrgu alamliik.
LwM2M	*Lightweight Machine to Machine*
Micro SD	Välise mälukaardi standard
MQTT	*Message Queuing Telemetry Transport*, sõnumijärjekorraga telemeetriatransport
NoSQL	*Not only SQL,* "mitte ainult SQL" keelel põhinevad andmebaasid
OS	*Operating System*, operatsioonisüsteem
OTA	*Over-the-air update,* tarkavara uuendamine üle juhtmeta ühenduse
PKI	*Public Key Infrastructure*, avaliku võtme taristu
Publish-Subscribe	Teemade tellimisel põhinev MQTT protokoll
RPC	*Remote Procedure Call*
RTC	*Real Time Clock,* riistavaraline reaalaja kell
SDK	*Software Development Kit,* tarkvaraarenduse komplekt
SoC	*System on Chip,* süsteemikiip
SPI	*Serial Peripheral Interface*
SQL	*Structured Query Language,* struktureeritud päringute keel
SSL	*Secure Sockets Layer*, vanem X.509 versioon

TLS	*Transport Layer Security*, uuem ja parandatud X509 versioon
Token	Rakendusepääse
UART	*Universal Asynchronous Receiver Transmitter*
UI	*User Inteface,* kasutajaliides
Webhook	Veebihaak, kasutaja määratletud HTTP-tagasikutse veebilehe või veebirakenduse käitumise muutmiseks
VPS	Virtuaalne privaatserver. Virtuaalmasin, mis kasutab osa füüsilise serveri ressurssidest ja mille puhul saab valida meelepärase operatsioonisüsteemi ning seda ise juurkasutaja õigustes hallata, uuendada ja turvata.
VSC	*Visual Studio Code,* redaktor
X.509,X.509	rahvusvaheline (*International Telecommunication Union)* standard, mis määrab avaliku võtme krüptograafia formaadi.

3. Nõuded seadmele ja analoogsed seadmed

Järgnevalt on kirjeldatud tehnilised nõuded seadmele, mis said aluseks riist ja tarkvaralistele valikutele. Seadme projekteerimisel sai valitud nn ülalt-alla lähenemine. Kõigepealt kirjeldati süsteemile vajalikud nõuded ja prioriteedid, seejärel valiti välja sobivad tarkvaralised lahendused ja pärast seda riistvara ja komponendid, mis valitud tarkvaralist lahendust toetasid. Kõige olulisemaks nõudeks oli avatud lähtekoodi kasutamine ja avaliku võtme standardil põhinev krüpteeritud mobiilne ühendus IoT kesksserveriga.

3.1 Nõuded seadmele üldiselt

- Interneti teel keskselt hallatav ja jälgitav
- jälgitavast seadmest sõltumatu andmeside ühendus
- autonoomne töö tundides vähemalt 10 tundi
- soovitav IP45 tolmu ja veekaitse
- staatuse indikaator

3.2 Nõuded tarkvarale

- avatud lähtekoodiga
- vastavus tunnustatud ja avatud standarditele
- kliendiseadme ühendus serveriga põhineb PKI(*Public Key Infrastructure*) standardil
- täielik ligipääs edastatud andmete andmebaasile
- hallatav ettevõtte poolt, ilma kolmandate osapoolteta
- soovitav on püsivara uuendused „üle õhu“ (OTA-*Over The Air)*

3.3 Nõuded riistvarale

- põhineb tuntud ja lihtsalt kättesaadavatel komponentidel
- kogu seadme hind võiks jääda 75 euro juurde
- keskne mikrokontroller on programmeeritav Python või C-keeles
- laienduspesa või ühendus lisaseadmete ühendamiseks

Nõuetele vastava tööstusliku seadme olemasolul oleks mõttekas seda kasutada. Sobiva tööstusliku seadme kasutamine teeks lihtsamaks seadmete asendamise ja juurde tellimise. Samuti võiks loota tootjapoolsetele püsivara uuendustele ja turvariskide jälgimisele. Ise projekteeritud seadme puhul tuleb nende teemadega ettevõtte siseselt tegeleda.

3.4 Analoogsed seadmed

Järgnevalt on kirjeldatud tuntumaid sobiva hinnaklassiga tööstuslikud seadmeid. Välja on toodud seadmete põhilised omadused ja puudused. Tootjateks on Itead Studio (SonOff kaubamärk) , Allterco Robotics (Shelly kaubamärk) ja PiKVM kaughallatava seade.

Sonoff POW Elite

Hiina tooja Itead Studio[5] toodab laias valikus kaughallatavaid seadmeid. Valikust kõige sobivam võiks olla mudel Sonoff POW Elite[2] hinnaga 16 eurot. Seade põhineb ESP32

[5] https://itead.cc (03.03.2024)

süsteemikiibil ja võimaldab pinge, voolutugevuse ja võimsuse mõõtmist. Võimalik on ühe väljundrelee juhtimine. Seade ühendub läbi WiFI võrgu internetiga.

Seadme puudused:

- suhtlusprotokoll on ametlikult dokumenteerimata
- puudub ülevaade OTA(*Over the Air*) uuendustest, mis loob eeldused seadme häkkimiseks[3]
- kasutab ühenduseks ainult WiFI võrku. Puudub nii Etherneti kui ka mobiilse interneti võimalus
- andmed salvestatakse Iteadi omandis olevasse suletud serverisse
- andmetele ligipääs toimib läbi Android või Iphone tarkvara. Tarkvara on suletud koodiga, ebaturvaline ja arvatavasti kogub taustal kasutaja andmeid[4]
- puudub ametlik API (*Application Programming Interface*) integreerimiseks kasutaja andmebaasiga

Shelly Pro 1PM

Bulgaaria taustaga Shelly Group[6] seadmetest võiks kasutada Shelly Pro 1PM mudelit[5] hinnaga 72 eurot. Seade on ehitatud ESP32 süsteemikiibile ja võimaldab lisaks voolutugevusele ja pingele mõõta ka seadme temperatuuri. Lisaks on olemas Etherneti ühendus. Shelly Pro seeria tooted toetavad standardset MQTT protokoll, HTTPS veebihaake ja on olemas korralikult dokumenteeritud API[6].

Seadme puudused:

- Androidi ja Iphone'i tarkvara on suletud lähtekoodiga
- puudub ülevaade OTA uuendustest
- puudub mobiilse andmeside võimalus
- API kasutamiseks on vajalik iga seade registreerida. Registreerimine on esialgu tasuta, kuid hoiatatakse võimaliku edaspidise hinnastamise eest[7]. Tänu kontrollimata OTA uuendustele on edaspidine hinnastamine võimalik
- andmete hoidmine Shelly serveris on piirangutega ja kuumaksuga[8]

PiKVM

PiKVM[7](Keyboard Video Mouse) on RaspberryPi miniarvutil toimiv kaughalduse seade. Tarkvara on spetsiaalseks otstarbeks kohandatud(PiKVM OS) avatud lähtekoodiga Linuxi operatsioonisüsteem Seade emuleerib USB ühenduse kaudu monitooritava arvuti jaoks klaviatuuri, hiirt ja USB välimälu. Monitooritava arvuti väljastatav videosignaal hõivatakse PiKVM poolt välise USB videosignaali digitaliseeria poolt ja edastatakse MPEG4 formaadis. PiKVM võimaldab distantsilt sekkuda jälgitava arvuti alglaadimisse ja muuta BIOS sätteid. Võimalik on laadida emuleeritava USB välimälu pealt vajadusel uus operatsioonisüsteem. PiKVM seade on võimalik osta või valmistada ise Raspberry Pi miniarvutil. Puudusena tuleb välja tuua Raspberry Pi suurt voolutarvet (kuni 24W) ja pikaks autonoomseks tööks see lahendus ei sobi. Seade on suhteliselt kõrge hinnaga (350 dollarit PiKVM V4 Plus). PiKVM koos väikese voolutarbega ESP32 kontrolleril baseeruva seadmega moodustab aga väga mõistliku lahenduse erijuhtude tarbeks. Tarkvara litsents on GNU General Public License v3.0 seega on lubatud äriline kasutus ettevõtete poolt.

---

[6] https://corporate.shelly.com/ (05.03.2024)

[7] https://pikvm.org/ (05.03.2024)

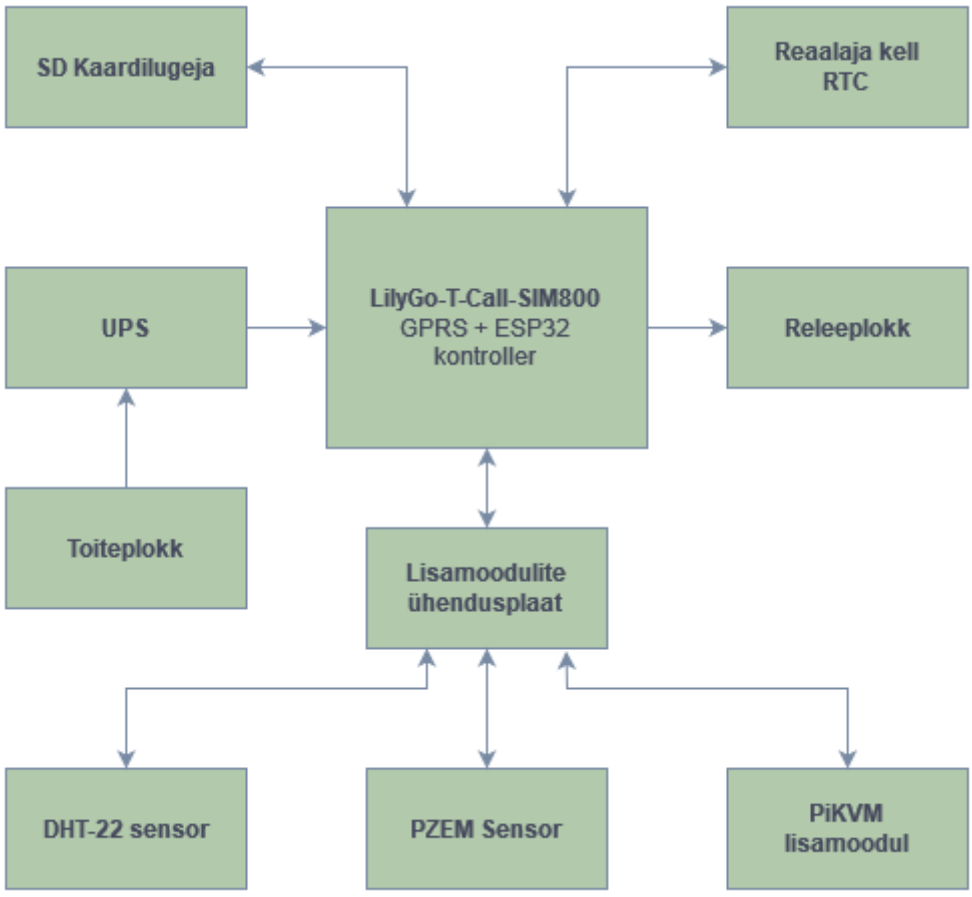
Sobivate tööstuslikult toodetud seadmete valiku puhul olid olulisteks puudusteks: küsitav turvalisus, liialt kõrge hind või suur energiatarve. Sobiva valmis seadme puudumine sai määravaks otsusele iseseisvalt seadme projekteerimisega alustada.

4. Riistavara ülevaade

Riistvara peatükis on kirjeldatud seadme tööd riistavara tasemel. Täpsemalt on kirjeldatud komponentide omadused ja antud ülevaade suhtlusprotokollidest. Peatüki lõpus on loetletud valminud seadme üldised parameetrid.

4.1 Riistvara plokkskeem ja kirjeldus

Joonisel 1 on kujutatud seadme plokkskeem. Seadme keskseks elemendiks on LilyGo T-Call arendusplaat, mis põhineb ESP32 süsteemikiibil ja SIM800 GPRS (General Packet Radio Service) modemil. Jälgitava seadme pinge, voolu ja sageduse väärtused saadakse PZEM-004 sensori abil. Temperatuuri ja õhuniiskust mõõdab DHT 22 andur. UPS(*Uninterruptible Power Supply*), reaalaja kell ja Micro-SD kaardilugeja kindlustavad andmete salvestamise ja säilimise elektrikatkestuse ajal. Lisamoodulite ühenduspesale on välja toodud 3,3V ja 5V toitepinged. Sama ühenduspesa kaudu saab lisada ka UART, I2C ja SPI protokolli toetavaid lisaseadmeid. Toiteplokiks on väline, 5V toitepinge ja 2A nimivooluga adapter.



Joonis 1. Riistavara plokkskeem

4.2 Ülevaade ESP32 süsteemikiibi ja lisaseadmete andmevahetuseks kasutatavatest protokollidest

ESP32 süsteemikiibil on seadmete vaheliseks suhtluseks kolm riistavaralist UART(*Universal Asynchronous Receiver/Transmitter*) liidest[8]. Vajadusel on võimalik tekitada juurde ka tarkvaralisi. UART liidesed on tarkvaraliselt seadistatavad ja nende kaudu on võimalik suhelda seadmetega RS232, RS422 ja RS485 protokollide vahendusel.

SPI(*Serial Peripheral Interface***)** protokoll[9] võimaldab süsteemikiibil lisaseadmetega suhelda. Kasutusel on *master-slave* arhitektuur ja seadmed ühenduvad omavahel :

- SCLK(*Clock*)- signaal sünkroniseerimiseks
- MOSI(*Master Out, Slave In*)- andmed Master seadmest
- MISO(*Master IN, Slave Out*)- andmed Slave seadmest
- SS(*Select Signal*)- Slave seadme valik

Andmevahetuse kiirus jääb vahemikku 10-100 Mbits/s. Iga Slave seade vajab ühte Select Signal ühendust, seega on lisatavate seadmete arv piiratud kontrolleri vabade ühendustega.

I2C(*Inter-Integrated Circuit***)**[9] protokoll loodi 1982 aastal Philipsi poolt eesmärgiga vähendada seadmete omavaheliste andmeühenduste arvu. Kasutusel on signaalid:

- SDA(*Serial Data*)
- SCL(*Serial Clock*)

I2C protokoll võimaldab seadmete adresseerimist. Seadmeid võib olla kuni 112(7 bitine aadress). Andmevahetuse kiiruseks on võimalik valida 100 kbits/s, 400 kbits/s või 3.4Mbits/s.

Modbus RTU[10] on sageli tööstuslikes seadmetes kasutatav protokoll. Töötab üle RS232 ja RS485 liideste. Protokoll on adresseeritav ja oluliseks plussiks on sisseehitatud veakontroll.

4.3 Riistavaraliste komponentide ülevaade

Riistavaralised komponendid valiti lähtuvalt projekti tehnilistes tingimustes kirjeldatud nõuetest. Prioriteediks olid lihtsalt kättesaadavad, soodsad ja levinud komponendid. Oluline tingimus oli sobivate vabavaraliste teekide olemasolu.

Riistavara valikul osutus kõige keerulisemaks sobiva arendusplaadi valik. Tulevikku silmas pidades oli algselt plaanis kasutada LPWA (*Low Power Wide Area*)[11] tehnoloogiat. Täpsemalt pakkus huvi LTE-M standard[12], mis peaks võimaldama IoT seadmetel olulist energiakokkuhoidu[13]. 2024 aasta alguses Telia, Elisa ja Tele2 operaatorite esindajatega suheldes selgus, et ametlikke teenusepakette veel pole. Osad operaatorid pakkusid küll võimalust osaleda test projektides. Valikus oli ka LTE mobiilside standardit toetav LilyGO-T-SIM7000G[9] arendusplaat. Määravaks sai aga nimetatud arendusplaadi mitu korda kõrgem hind, suurem voolutarve ja vähene tarkvaraline tugi. Lõpuks sai seireseadme esimeseks versiooniks valitud GPRS ühendusel põhinev arendusplaat. Oluline on märkida, et tänu püsivara programmis kasutatud universaalsele TinyGSM teegile peaks olema hilisem üleminek LTE-M modemile lihtne (lähtekoodis kasutatavad funktsioonid jäävad samaks).

[8] https://github.com/espressif/esp-idf/blob/b90dfe075923870a8d1f8d85a75318781455f57a/docs/en/api-reference/peripherals/uart.rst (05.02.2024)
[9] https://github.com/Xinyuan-LilyGO/LilyGO-T-SIM7000G (05.02.2024)

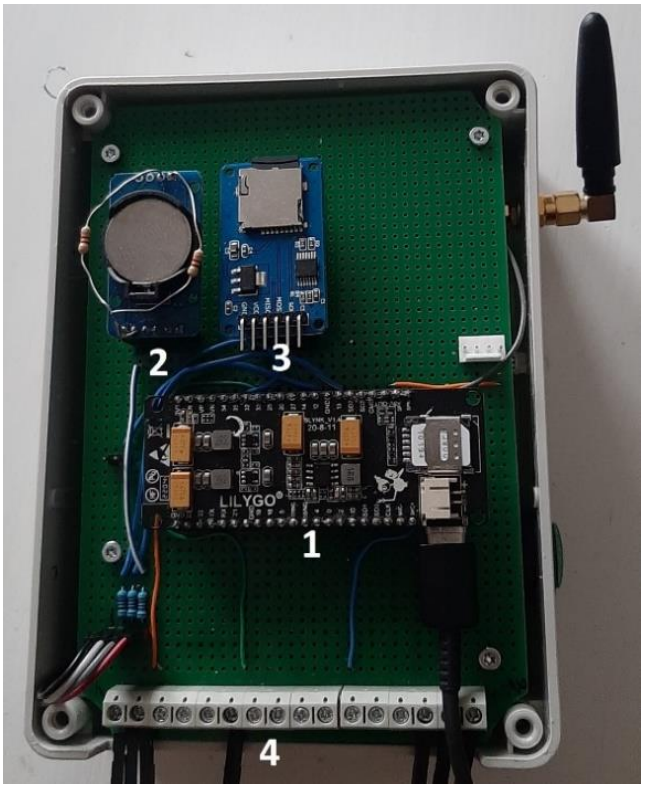
Tabel 1 Riistvara ülevaade

Nimi	Funktsioon	Toitepinge	Ühendus	Hind €
LilyGo T-Call[10]	Arendusplaat	5V	UART, SPI, I2C	25
Peacefair PZEM-004T[11]	Pinge, voolu ja sageduse andur	3.3V	UART	15
Micro SD seade	Micro SD kaardi lugeja/kirjutaja	5V	SPI	1
DHT 22[12]	Temperatuuri ja niiskuse andur	5V	Isiklik	2
DS3231SN	Reaalaja kell	3.3-5V	I2C	3
Akuplokk LX-2BUPS	Katkematu toite allikas	5V		5
Samsung INR18650-30Q[13]	Akud 2 tk	3.6V		6
Makettplaat[14]				10
Korpus				5

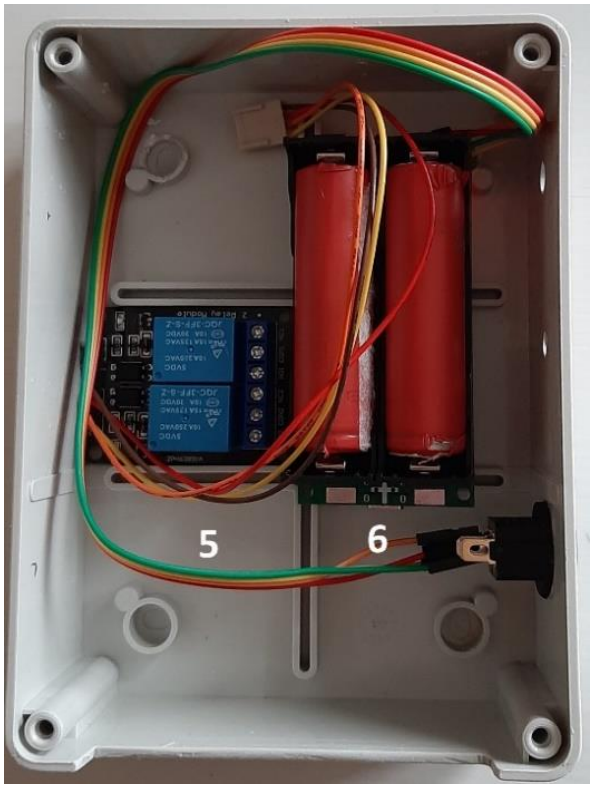
Komponentide ligikaudseks hinnaks tuli kokku 72 eurot (tabel 1). Sarnase hinnaga on saadaval peatükis 3.4 kirjeldatud seade Shelly Pro 1PM, millel on üks releeväljund ja puudub mobiilne andmeside keskserveriga. Seega ise komponentidest seadet ehitades olulist kokkuhoidu ei saavuta, hind jääb tööstuslike seadmetega samasse suurusjärku. Oluline erinevus tööstuslikust seadmest on võimalus valida sobiv andmeside liik keskserveriga ja luua ettevõtte vajadustele vastav funktsionaalsus.

[10] https://www.lilygo.cc/products/t-call-v1-4 (05.02.2024)
[11] http://en.peacefair.cn/products/3487.html (05.02.2024)
[12] http://www.aosong.com/en/products-22.html (05.02.2024)
[13] https://eu.nkon.nl/rechargeable/li-ion/18650-size/samsung-inr-18650-30q-3000mah-15a-2017.html (05.02.2024)
[14] https://www.oomipood.ee/product/ecs1_makettplaat_100_160mm_eraldi (05.02.2024)

Pildid seadmest



Joonis 2 Sisevaade 1



Joonis 3 Sisevaade 2

Joonis 4 Välisvaade koos anduritega

1. LilyGo T-Call arendusplaat
2. Reaalaja kell(RTC)
3. Micro SD seade
4. Ühendus välistele seadmetele
5. Releemoodul
6. Akuplokk
7. DHT 22 sensor
8. PZEM-004T moodul

LilyGo T-Call v1.3 arendusplaat.

Loodud seadme keskseks komponendiks sai valitud arendusplaat Hiina tootjalt LilyGo. Tegemist on tootega aastast 2019, kuid sellele on lisatud pidevalt riistvaralisi ja tarkvaralisi täiendusi. Toote paariaastane vanus on pigem boonuseks, tarkvaralisi vigu on vähem. Plaadi täielik nimi on ESP32 LilyGo-T-Call-SIM800 SIM800L_IP5306_(v1.3).15 Arendusplaat võimaldab kasutada erinevaid sidekanaleid suhtlemiseks IoT serveriga. Kogu plaadi tööd koordineerib 40 nm tehnoloogial põhinev kombineeritud süsteemikiip ESP32-WROVER[16]. Mis sisaldab muudetava kiirusega (80 MHz-240 MHz) madala voolutarbega kahetuumalist protsessorit. Sidelahendustest on süsteemikiibile integreeritud Bluetooth®, Bluetooth LE ja WiFi 802.11 b/g/n (kiirus kuni 150 Mbps). ESP32 süsteemikiibi eelisteks on:

- sisend/väljund ühenduste arvukus(34 programmeeritavat ühendust)
- standardsete suhtlusprotokollide tugi (UART, SPI, I2C)
- madal voolutarve (töös: 80 mA, uinunud olekus < 5 μA)
- soodne hind, hea kättesaadavus ja vabavaraline tugi.

T-Call arendusplaat kasutab lisaks WiFI standardile andmeside ühenduse loomiseks SIM800L GPRS modemit[17]. Modem toetab 850/900/1800/1900MHz sagedusi ja võimaldab maksimaalselt 85.6 kbps andmeedastust. Kogu plaadi toitepinge on 5V, regulaatorina on kasutuses SY8089 toitepinge regulaator. Arendusplaat võimaldab kasutada välise toiteallikana 3.7V 18650 Li-ion akusid laadimis/ tühjenemisvooluga kuni 2A.

Peacefair PZEM-004T V3.0 lisaseade

Lisaseade[18] võimaldab jälgida tarbimiskoha vahelduvvoolu järgmisi parameetreid: voolu, pinget ja sagedust. Lisaseade kasutab arendusplaadiga suhtlemiseks UART liidest ja Modbus-RTU standardset protokolli. Tootja andmetel suudab seade mõõta vahelduvpinget vahemikus 80V−260V ja vahelduvvoolu 0,1−10A, täpsusega 0.5%. Toitepingeks sobib 3.3-5V. PZEM-004T eelised on mõõtmiste jaoks piisav täpsusklass[14]. Seadmest on kasutusel kolmas riistvaraline versioon. Leidub arvukalt PZEM -004T põhinevaid projekte[15]. PZEM-004T suhtlusprotokoll on tootja kodulehel hästi dokumenteeritud[19] ja tarkvaraline tugi erinevates keeltes on hea. Seade on soodsa hinnaga. Sarnases hinnaklassis (15 eurot) arvestatavaid alternatiive pole.

Micro SD seade

Laialt levinud seade Micro SD mälukaardilt lugemiseks ja kirjutamiseks ühenduse puudumise režiimis. Kasutab ühenduseks SPI standardit. Toitepinge on 5V.

---

[15] https://github.com/Xinyuan-LilyGO/LilyGo-T-Call-SIM800/blob/master/schematic/LilyGo-SIM800L-IP5306-20190610.pdf (05.02.2024)

[16] https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf (06.02.2024)

[17] https://simcom.ee/documents/SIM800L/SIM800L%20SPEC170914.pdf (06.02.2024)

[18] http://en.peacefair.cn/products/3487.html (06.02.2024)

[19] https://innovatorsguru.com/wp-content/uploads/2019/06/PZEM-004T-V3.0-Datasheet-User-Manual.pdf (06.02.2024)

Aosong Electronics DHT22(AM2302) temperatuuri ja õhuniiskuse sensor

Sensor[20] on kasutuses jälgitavat seadet ümbritseva keskkonna jälgimiseks. Tegemist on suhteliselt kvaliteetse ja täpse anduriga. Valitud sai see hea kättesaadavuse ja stabiilse töö pärast. Andmeedastus toimub ühe väljundi kaudu, kasutuses on mittestandardne tootjapoolne protokoll. Protokoll on tootja dokumentatsioonis põhjalikult kirjeldatud. Andur on tootmises olnud kaua aega, seega on olemas sobivad teegid kõikides enamlevinud programmeerimiskeeltes. Andur mõõdab temperatuuri vahemikus -40 kuni +80 C ja õhuniiskust 0-100% RH. Toitepingeks sobib 3.3-6V

DS3231SN RTC

Reaalaja kella[21] RTC*(Real Time Clock*) DS3231SN kasutatakse seadmes andmetele ajatempli lisamiseks. Andmeside olemasolul sünkroniseeritakse reaalaja kella ajaserveri (NTP protokoll) kaudu. DS3231SN täpsus on ±2ppm (parts per millions)(0°C to +40°C). Suhtluseks on standardne I²C(400kHz) protokoll ja toitepingeks sobib 3.3V.

Releemoodul ME114

Hiina tootja moodul[16] on lisatud väliste seadmete alglaadimiseks ja nende toite juhtimiseks. Moodulil on kaks väljundreleed alalisvoolu (30V/10A) või vahelduvvoolu (250V/10A) juhtimiseks. Relee sisendid on optiliselt eraldatud ja neid saab juhtida 3,3V või 5SV pingega. Relee toide on 5V.

Akuplokk LX-2BUPS

Tagab seadme autonoomse töö. Plokk[22] kasutab standardseid liitium-ioon 18650 mõõdus akusid. Sisendpinge 2.7V-5V, Väljund 5V, Nimivool 3A. Kasutegurit lubatakse 96%. Komponentidest kasutatakse XR2981 ja TC4056A mikroskeeme.

RGB LED indikaator

Standardne 3.3-5V juhitav RGB led seadme staatuse indikatsiooniks.

Ühendus välistele seadmetele

Võimaldab lisada lihtsalt väliseid seadmeid. Väljundid on SPI, I2C, UART, releede, 3.3V ja 5V toite jaoks (tabel 2). Välja on toodud ka osad ESP32 GPIO kasutamata väljundid, mida saab vastavalt vajadusele ümber programmeerida.

Tabel 2 Ühenduspesa kontaktide numeratsioon

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
GND	3.3V	_	_	_	DHT sensor	I2C SCL	TX	RX	I2C SDA	_	SPI MISO	SPI SCK	TX2	RX2	5V

[20] http://www.aosong.com/en/products-22.html (06.02.2024)

[21] https://www.analog.com/media/en/technical-documentation/data-sheets/DS3231.pdf (06.02.2024)

[22] https://cdn.hackaday.io/files/1829407826904960/Xysemi_XR2981.pdf (06.02.2024)

Toiteplokk

Väljundpinge: 5V.

Väljundvõimsus: 2W.

Seadme konstruktsioon

Elektroonilised komponendid monteeriti makettplaadile eelnevalt joodetud pesadesse. Makettplaat (Joonis 2) koos releede ja UPS mooduliga (Joonis 3) paigaldati sobivas mõõdus tehnoloogilisse karpi. Seadme korpusel (Joonis 4) on GPRS andmeside antenn, sisse-väljalülituse nupp, toitepesa ja seadme staatust näitavad Led indikaatorid.

Seadme parameetrid:

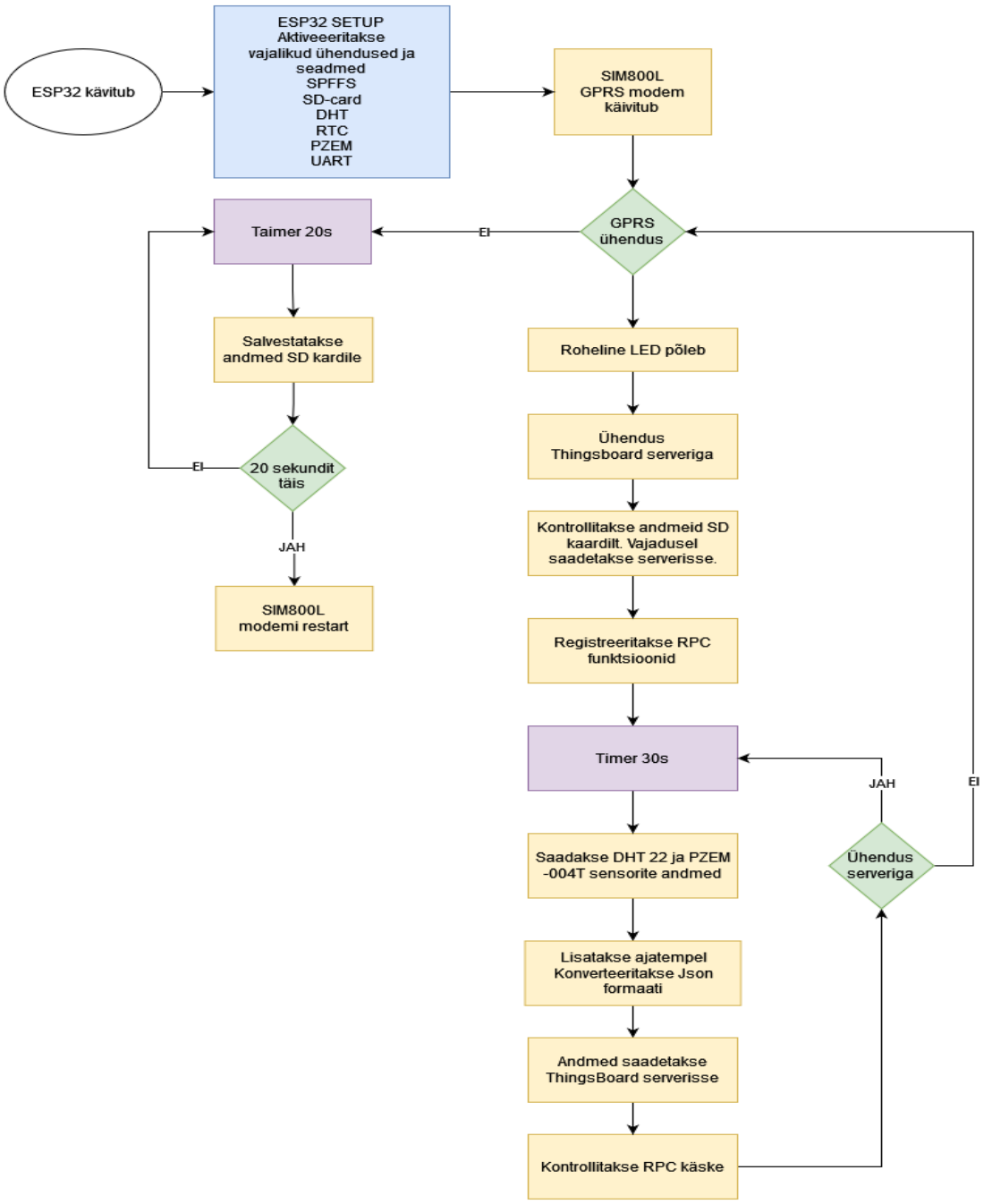
- seadme toitepinge: 5V;
- seadme voolutarve ooteajal: 80 mA;
- seadme voolutarve andmete saatmise ajal: 170 mA;
- seadme kaal: 480 g;
- seadme mõõtmed: laius 115 mm, pikkus 155 mm, kõrgus 85 mm.

Peale veendumist, et makettplaadil olevate seadmete omavahelised ühendused olid korrektsed ja süsteemikiip ESP32 suudab täita lihtsamaid funktsioone oli võimalik asuda tegelema keerukama püsivara programmeerimisega.

5. Tarkvara ülevaade

Ülevaade algab seadme töö üldise kirjeldusega ja ülevaatega ESP32 süsteemikiibi programmeerimisest. Järgneb alalõik IoT serveripoolsest tarkvarast-ThingsBoard IoT server. Antakse ülevaade selle omadustest ja tööpõhimõttest. Täpsemalt on kirjeldatud seadme ja serveri vahelist MQTT suhtlusprotokolli. Peatüki teine pool on tehnilisem. Sealt leiab komponentide programmeerimiseks kasutatud teekide kirjelduse koos lühikeste koodinäidetega. Peatüki lõpetab ülevaade lähtekoodist ja kirjeldus töö käigus tekkinud mõtetest.

5.1 Ülevaade programmi tööst



Joonis 5. Tarkvara plokkskeem.

Järgnevalt on kirjeldatud Joonis 5 oleva tarkvara plokkskeemi toimimist

1. ESP32 põhinev süsteemikiip käivitub.
2. Initsialiseeritakse lisaseseadmed: GPRS modem, reaalaja kell(RTC), SD kaardi lugeja, DHT22 ja PZEM-004t andurid.
3. GPRS ühenduse olemasolul võtab seade ühendust NTP(*Network Time Protocol*) serveriga ja uuendab reaalaaja kella kellaaja ja kuupäeva.
4. Luuakse MQTT-TLS protokolli kasutades turvaline ühendus ThingsBoard IoT serveriga. Seade autoriseeritakse ja registreeritakse unikaalse rakenduspääsme(token) abil.
5. Kontrollitakse SD kaardil olevat eelmistest sessioonidest säilinud infot. Info olemasolul saadetakse see ThingsBoard IoT serverisse ja luuakse andmete jaoks uus tühi fail.
6. Sensorite DHT22 ja PZEM-004 andmetest ning reaalaja kella ajatemplist(Unix formaadis) luuakse lihtne struktuur andmete saatmiseks ja vajadusel salvestamiseks.
7. Andmed saadetakse ThingsBoard serverisse.
8. ThingsBoard server tagastab vastuse andmete salvestamise õnnestumise kohta. Salvestamise luhtumise korral salvestatakse andmed SD kaardile, õnnestumise korral jätkatakse programmi täitmist.
9. Kontrollitakse RPC (*Remote Procedure Call*) käskude olemasolu. RPC käske kasutatakse seadme väljundreleede juhtimiseks. Käskude olemasolul need täidetakse.
10. Peale ajalist viidet tsükkel kordub alates andmete lugemine sensorilt.
11. GPRS ühenduse või ThingsBoard serveriga andmeside katkemise korral salvestatakse andmed SD kaardile kuniks ühendus taastub. Peale ühenduse taastumist saadetakse vahepeal kogutud andmed serverisse.

5.2 ESP32 süsteemikiibi programmeerimine

Seireseadme keskmeks oleva ESP32 süsteemikiibi programmeerimiseks sobiv tarkvara on olemas nii Windowsi, Linuxi kui ka Mac OS operatsioonisüsteemide jaoks. Levinumad programmeerimise võimalused on:

- C-keeles Arduino keskkonnas, kasutades tootja Espressif IoT Development raamistikku
- C-keeles tootja Espressif IoT Development raamistikku kasutades
- C-keeles kasutades Visual Studio Code IDE + PlatformIO lisamoodulit
- Python keeles Micropython keskkonnas

Arduino programmeerimise keskkond

Arduino keskkond on vabavaraline, Java keeles kirjutatud, erinevaid protsessoreid toetav, programeerimise keskkond[23]. Alustamise Arduino keskkonnas teeb lihtsaks just algajatele mõeldud juhendite ja näidete rohkus. Erinevate lisaseadmete ja andurite tugi on hea ja versiooniuuendused sagedased. Tegemist on õppimiseks sobiva standardse ja suhteliselt stabiilse keskkonnaga. Palju kasulikku infot leiab erinevatest foorumitest ja kasutajate kodulehtedelt. Mahukamate projektide jaoks jääb Arduino keskkond siiski natuke piiratuks just seadistuste ja mugavust võimaldavate tööriistade puudumise poolest. ESP32 prot-

[23] https://www.arduino.cc/en/software (07.02.2024)

sessori tootja Espressif Systems[24] arendab vabavaralist ESP32 protsessori tuge Arduino keskkonnale. Tarkvara nimi on Arduino Core for The ESP32[25].

Visual Studio Code + PlatformIO

Visual Studio Code (VSC) redaktor[26] on Microsofti poolt arendatud tarkvara. Sobib kasutamiseks Window, Linux ja MacOS operatsioonisüsteemidel. ESP32 tootja Espressif Systems on sarnaselt Arduino keskkonnale loonud ka Visual Studio Code redaktorile ESP32 tarkvaralise toe[27]. Tänu PlatformIO[28] lisamoodulile on selle kasutamine tehtud mugavaks ja lihtsaks. Visual Studio Code keskkonnas on suuremate projektide jaoks lisavahendeid ja hõlbustusi rohkem. Saadaval on palju programmeerimist lihtsustavaid lisasid. VSC keskkond toetab enamust programmeerimise keeli. Oluline lisa on ESP32 süsteemikiibi riistvaraline silumis funktsioon.

Espressif IoT Development Framework

On ESP32 tootja Espressif poolt loodud professionaalne programmeerimise raamistik Linuxi operatsioonisüsteemil[29]. Sihtgrupiks on professionaalsed programmeerijad. Kasutada saab Linuxi käsurea programme ja keelena on kasutusel C-keel. Eeliseks on kiired tootjapoolsed uuendused ja võimalus kasutada professionaalsetel programmeerijatel koodi optimeerimiseks mikroprotsessori kõige madalama taseme funktsioone.

Micropython

ESP32 süsteemikiipi on võimalik programmeerida ka Python keeles. Selleks on kasutusel Micropython[30] püsivara ESP32 protsessoritele. Väliste lisaseadmete ja andurite jaoks on vaja installeerida vajalikud teegid ja tarkvara. Micropythoni kasutamine teeb programeerimise oluliselt lihtsamaks, kuid takistuseks võib saada just mõne vajaliku lisaseadme tööks vajaliku tarkvaralise toe puudumine. Samuti langeb süsteemikiibi jõudlus ja suureneb mälukasutus.

Lõputöös oli algselt kavas kasutada Micropythoni tarkvara, kuna Python keeles programmeerimine on lihtsam ja kiirem. Töö käigus selgus, et Micropython kasutab rohkem ESP32 mikroskeemi mälu ja protsessori ressurssi. Lisaks vähendas iga lisatav teek oluliselt vaba mälu ja teekide lisandudes jäi õige pea süsteemikiibi 4MB mälumahust väheseks. Seega sai valminud töös kasutatud Arduino (Arduino IDE 2.3.2) keskkonda ja C-keelt. C-keele väiksem ressursinõudlikkus ja Arduino keskkonnas stabiilselt töötavate teekide olemasolu said lõpuks määravaks. Palju kasu oli teekidega kaasas olevad näidetest ja Arduino foorumitest saadud soovitustest.

---

[24] https://www.espressif.com (07.02.2024)
[25] https://github.com/espressif/arduino-esp32 (07.02.2024)
[26] https://code.visualstudio.com (07.02.2024)
[27] vscode-esp-idf-extension (07.02.2024)
[28] https://platformio.org/ (07.02.2024)

[29] https://github.com/espressif/esp-idf#espressif-IoT-development-framework (07.02.2024)

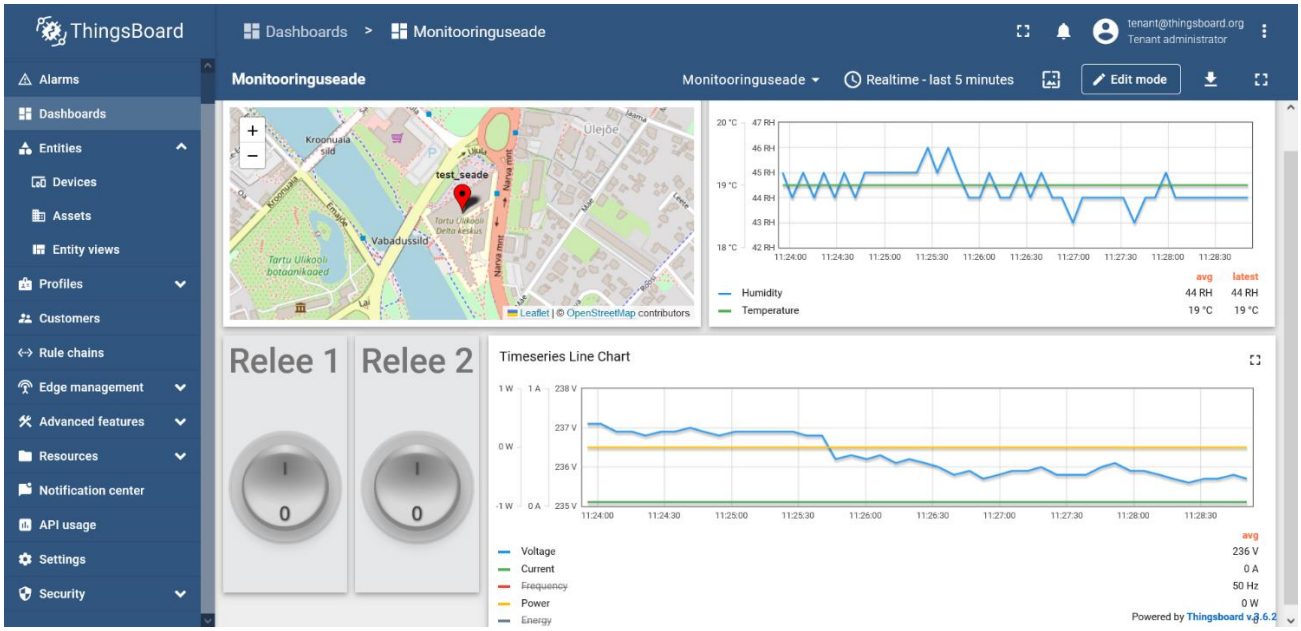
[30] https://docs.micropython.org/en/latest/esp32/tutorial/intro.html (07.02.2024)

5.3 ThingsBoard IoT platvormi ülevaade

Lisaks ESP32 süsteemikiibil töötavale andmeid edastavale seadmele (klient) oli vaja leida vastus ka IoT serveri näol. Iseseisvalt selle väljatöötamine oli liiga töömahukas ja valida oli võimalik mitmete valmis lahenduste vahel.

Olulised tingimused sobivale serveripoolsele IoT tarkvarale olid järgmised:

a) avatud lähtekoodiga;
b) vajadusel võimaluste ja jõudluse osas laiendatav;
c) võimaldab ligipääsu algandmete andmebaasile;
d) ühendustes kasutatakse standardseid avatud protokolle;
e) seadmete uuendamine „üle õhu“ (OTA);
f) seadmete keskne juhtimine;
g) võimalik ettevõtte poolt hallata.



Joonis 6. ThingsBoard IoT keskkond

Kirjeldatud tingimustele vastas kõige paremini ThingsBoard IoT platvorm[31]. Selles töös on kasutatud ThingsBoard CE(*Community Edition* 3.6.2) tasuta versiooni. Ettevõtte seisukohalt on oluline ThingsBoard PE[32] (*Professional Edition*) olemasolu. Tasuline versioon võimaldab lihtsat teenuse mahu suurendamist, tehnilist tuge ja lisafunktsioone. Ettevõtte jaoks üks olulisi lisasid on ThingsBoard Trendz[33] pakett. Trendz võimaldab masinõppe meetoditega andmete analüüsi, millest huvitavamad on anomaaliate tuvastamine ja ennetava hoolduse planeerimine. ThingsBoard server võimaldab ligipääsu ka andmebaasi toorandmetele, seega on võimalik kogutud andmeid iseseisvalt analüüsida. Ülevaade graafilisest liidesest on Joonis 6. Lisas 1 on toodud juhend ThingsBoard PostgreSQL andmebaasi ühendamiseks DBeaver[34] tarkvaraga.

[31] https://thingsboard.io/ (07.02.2024)

[32] https://thingsboard.io/products/thingsboard-pe/ (07.02.2024)
[33] https://thingsboard.io/products/trendz/ (07.02.2024)
[34] https://dbeaver.io/ (07.02.2024)

Thingsboard serveri tarkvara põhilised omadused.

1. Võimaldab reaalajas jälgida enamlevinud operatsioonisüsteemidel töötavaid seadmeid ehk kliente. Lihtne on kujunda sobivaid kontrollpaneele, hoiatusi ja graafikuid. Mikrokontrolleritel põhinevatele seadmetele on olemas Arduino teek[35]. Windows ja Linux operatsioonisüsteemidele on olemas tarkvaralised kliendid[36].
2. Telemeetrilised andmed salvestatakse edasise statistika tegemiseks andmebaasi. Andmebaasi valik on vaba, võimalik on valida SQL kui ka NonSQL andmebaasi vahel.
3. Andmete vahetuseks kasutatakse standardseid MQTT, HTTP(S), CoAP ja LwM2M protokolle.
4. Võimalik on klient seadmete tarkvara uuendamine „üle õhu“ (OTA).
5. ThingsBoard kasutab seadmete juhtimiseks RPC (*Remote Procedure Call)* meetodit. RPC algataja võib olla nii klient, kui ka server. Kliendi poolt algatatud ühendus võimaldab lihtsustatud suhtlust läbi tulemüüri[37].
6. Rule Engine on graafiline keskkond andmete töötluseks ja konverteerimiseks (sarnane laiemalt tuntud Node Red tarkvara tööpõhimõttega).
7. Thingsboard CE on kaetud Apache License 2.0 litsentsiga.
8. Põhjalik API koos dokumentatsiooniga.

ESP32 süsteemikiibi kasutamine koos ThingsBoard serveriga.

Klient seade saab andmeid vahetada Thingsboard serveriga kahel erineval viisil:

1. Kasutades standardseid HTTP või MQTT protokollide teeke ja dokumenteeritud API-t.
2. Kasutades spetsiaalset Arduino teeki ThingsBoard Client SDK[38] (*Software Development Kit*) suhtlemiseks Thingsboard IoT serveriga. Loodud on see MQTT protokollil, kuid võimaldab rohkem funktsioone.

Töös on kasutatud ThingsBoard Client SDK teeki versiooniga 0.11.1[39]. See oli ainuke teek, mis õnnestus ilma vigadeta suhteliselt stabiilselt tööle saada. Kasutusel olid järgmised funktsioonid:

- ThingsBoard.connect()- ühendab seadme Thingsboard serveriga;
- ThingsBoard.connected()- kontrollib ühendust, vajadusel ühendab uuesti;
- ThingsBoard.RPC_Subscribe()-registreerib seadme serveri poolt edastatud RPC käsklusi täitma:
- ThingsBoard.sendTelemetryData() -saadab lihtsa teksti kujul andmed serverile;
- Thingsboard.sendTelemetryJson() – saadab JSON formaadis andmed serverile;
- ThingsBoard.loop() -kontrollib serveri RPC käskluste olemasolu serveris.

Töö käigus sai andmete edastamiseks katsetatud erinevaid variante. Kõige sobivamaks osutus andmete edastamine JSON formaadis. Põhjuseks võimalus lisada andmetele stan-

[35] https://github.com/thingsboard/thingsboard-client-sdk (07.02.2024)

[36] https://thingsboard.io/docs/devices-library/ (08.02.2024)

[37] https://thingsboard.io/docs/user-guide/rpc/ (08.02.2024)

[38] https://github.com/thingsboard/thingsboard-client-sdk (08.02.2024)

[39] https://www.arduino.cc/reference/en/libraries/thingsboard/ (08.02.2024)

dardne Unix süsteemi ajatempel (millisekundites). Server tuvastab ajatempli DateTime formaadi ja tänu sellele astuvad ka eelmistest sessioonidest pärit andmed andmebaasis ja graafilisel aegteljel õigetesse kohtadesse. Näide edastatavate andmete formaadist JSON kujul:

```
{
  "ts": 1527863043000,
  "values": {
    "temperature": 20.2,
    "humidity": 50
  }
}.
```

Peale andmete vahetust tagastab server kliendile spetsiaalse tulemuskoodi. Andmevahetuse ebaõnnestumisel saadetakse andmed uuesti. Lisas 2 on kirjeldatud seadme püsivara seadistamist koostööks ThingsBoard serveriga.

5.4 MQTT protokolli ülevaade

Tehnilistes tingimustes oli seadme andmevahetuse eelduseks krüpteeritud ühendus IoT serveriga. Seega valiti seireseadme ühenduseks ThingsBoard serveriga krüpteeritud MQTT (*Message Queue Telemetry Transport*) protokoll. Järgneb ülevaade MQTT protokollist.

MQTT on standardiseeritud (v5.0 and v3.1.1 OASIS v3.1.1 ISO), avatud, masinalt-masinale, teemade tellimisel põhinev, kergekaaluline protokoll[40]. Loodi see 1999 aastal IBM inseneride poolt ja kasutati algselt tööstuslike seadmete vahel andmete vahetuseks.

MQTT protokolli eelised:

1. Efektiivne ja vähest ressurssi nõudev. Oli algselt mõeldud kaugel distantsil asuvate ja piiratud side ühendusega seadmete jälgimiseks ja juhtimiseks.
2. Protokoll kasutab spetsiaalseid QoS (*Quality of Service*) funktsioone nõrkade ja sageli katkevate sideühenduste jaoks.
3. Protokoll võimaldab kasutada *Transport Layer* Security (TLS) / *Secure Sockets Layer* (SSL) krüpteerimist. Võimalik on kasutajate autentimine salasõna ja/või kliendi sertifikaadi abil.
4. MQTT protokoll võimaldab seadmete ja keskse serveri kahepoolset suhtlust. Kliendid saavad nii postitada, kui ka lugeda teateid soovitud teemadest.
5. MQTT server jälgib klientidele teadete kättetoimetamist. Ühenduse katkemise järel salvestatakse kohale toimetamata teated ja peale ühenduse taastumist saadetakse need edasi. Seega pole pidev ühendus vajalik.
6. MQTT protokolli tarkvara on levinumates programmeerimise keeltes hästi toetatud.

[40] https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (11.02.2024)

MQTT protokolli tööpõhimõte

Põhilised üksused MQTT süsteemis on:

1. MQTT klient on iga seade, mis kasutab MQTT kliendi tarkvara ja mis on seadistatud tööks MQTT serveriga.
2. MQTT Broker ehk server tagab klientide ühendumise, autentimise ja klientide registreerimise teemadele ja teadete vahetusele.
3. Publish-Subscribe ehk avaldamine-registreerimine soovitud teemadele. Kliendid ei pea omavahel otse suhtlema, MQTT server tagab sõnumite kättetoimetamise.
4. Topic ehk teema millele kliendid saavad tellijaks asuda, näiteks “chat/room/1”. Teemad on hierarhilise ülesehitusega.
5. Quality of Service (QoS) ehk teenuse kvaliteedi kategooriad:
    a. QoS 0: Teade edastatakse 1 kord, kui klient seda kohe kätte ei saa, teade kaob. Tagasiside teate kättesaamise kohta puudub.
    b. QoS 1 :Teade edastatakse vähemalt korra ja teate saaja edastab info kättesaamise kohta. Kui saatja tagasisidet ei saa, teadet korratakse.
    c. QoS 2 :Teade edastatakse üks kord, kuid selle kättesaamist kontrollitakse 4-poolse käepigistusega.

MQTT kliendi ja serveri vaheline suhtlus:

- klient algatab ühenduse MQTT serveriga, kasutades TCP/IP protokolli. Vajadusel kasutab krüpteeritud ühendust ja identifitseerib enda;
- klient kas postitab teate või registreerib ennast soovitud teemadele;
- server edastab teate kõikidele tellijatele vastavalt klientide QoS seadetele.

MQTT protokolli turvalisus

Algselt oli MQTT protokolli eesmärk kanda üle ettevõte siseseid telemeetrilisi andmeid hoolimata suurtest vahemaadest ja alla keskmise võrguühendustest. Seega ei olnud turvalisus teemaks ja lihtne avatud tekstipõhine autentimine oli piisav. Tänapäeval läbib aga MQTT pakett mitmeid erinevaid võrke ja MiTM (*Man in The Middle*) rünnaku oht on reaalne. Kuna MQTT põhineb TCP (*Transmission Control Protocol*) protokollil, siis standardseks lahenduseks sarnaselt HTTP protokolliga on kasutada andmete krüpteerimiseks TLS/SSL (*Transport Layer Security*) protokolli. TLS/SSL protokolli toetab ka enamus MQTT servereid. Avatud MQTT protokoll on väga minimaalne, kuid koos TLS/SSL kasutamisega kasvab paketi suurus keskmiselt kaks korda[17]. Samuti pikeneb serveriga ühendumise aeg[18]. Olulise aspektina tuleb märkida, et kliendi andmed dekrüpteeritakse MQTT serveris. Seega on MQTT serveri turvalisus samuti väga oluline. Krüpteerimata MQTT serveri port on tavaliselt 1883 ja krüpteeritud ühenduse serveri port 8883.

MQTT protokolli turvalisus ThingsBoard platvormil

ThingsBoard platvormil on võimalik on kasutada järgnevaid viise[41] kasutajate autentimiseks:

1. Unikaalsed ligipääsu rakenduspääsmed
    a. Lihtne kasutada ja toetab ka vähese võimsusega seadmeid
    b. Saab kasutada koos TLS krüpteerimisega
    c. Puuduseks on võimalus lihtsalt krüpteerimata pakette pealt kuulata
2. Algne MQTT kasutajanime ja parooli kasutamine-sarnane esimese võimalusega.
    a. Lihtne kasutada ja toetab ka vähese võimsusega seadmeid
    b. Puuduseks on võimalus lihtsalt pakette pealt kuulata.
3. X.509 [42] sertifikaatide kasutamine -PKI, TLS standarditel
    a. Eeliseks on andmete täielik krüpteerimine.
    b. Puuduseks on lihtsamatel seadmetel puuduv TLS tarkvara tugi. Samuti on see eelmistes variantidest oluliselt energia ja ressursimahukam.

Seireseadmes kasutatud krüpteeritud MQTT ühendus

Seadmes kasutatakse MQTT kliendi ja serveri vahelise ühenduse krüpteerimiseks standardset TLS protokolli. Ühenduse testimiseks kasutati nn ise allkirjastatud sertifikaate (*šelf signed certificate*). ESP32 süsteemikiipide juures kasutatakse TLS protokolli enamasti üle WiFI ühenduse. Seega standardsed ESP32 teegid üle GPRS ühenduse ei tööta. Pärast paljude erinevate teekide katsetamist leidus teek SSL kliendi tarkvaraga[43], mis tegi ka GPRS andmeside krüpteerimise võimalikuks.

Lisatud näide SSL teegi kasutamises GPRS ühenduse krüpteerimiseks.

```
#include "SSLClient.h"

ENCRYPTED true;
//Layers stack
TinyGsm sim_modem(SerialAT);
TinyGsmClient gsm_transpor_layer(sim_modem);
SSLClient secure_presentation_layer(&gsm_transpor_layer);
//Add CA Certificate
 secure_presentation_layer.setCACert(root_cert);// root certificate
 secure_presentation_layer.setCertificate(client_cert);//x509 client Certificate
  secure_presentation_layer.setPrivateKey(client_key);//x509 client key
// ThingsBBoard init
#if ENCRYPTED
ThingsBoard tb(secure_presentation_layer, MAX_MESSAGE_SIZE);
#else
ThingsBoard tb(gsm_transpor_layer, MAX_MESSAGE_SIZE);
#endif
```

[41] https://thingsboard.io/docs/user-guide/mqtt-over-ssl/ (11.02.2024)

[42] https://thingsboard.io/docs/user-guide/certificates/ (14.02.2024)

[43] https://github.com/govorox/SSLClient (14.02.2024)

5.5 ESP32 süsteemikiibi ja lisaseadmete programmeerimine

Olulise osa tööst moodustas lisaseadmetele sobivate sobivate teekide leidmine, nende katsetamine ja omavahel tööle saamine. Järgnevad lühikesed väljavõtted algkoodist tutvustavad üldpõhimõtteid ja erinevate teekide kasutamist.

LilyGo-T-Call-SIM800 arendusplaadi programmeerimine

Arendusplaadi valikul sai oluliseks tarkvaraline toetus ja eeldatavasti töötavate näidisprogrammide olemasolu. Github[44] keskkonnast selgus, et neli aastat vana riistvara jaoks olid viimased tarkvaralised uuendused aasta tagused. Seega sai LilyGo-T-Call arendusplaadile panustatud. Seade telliti Aliexpress keskkonnast ja kauba kättesaamisel selgus millise riistvaralise versiooniga on täpselt tegemist.

Töös kasutati algselt Arduino 2.2.0 IDE keskkonda, hiljem projekti täienedes Visual Studio Code redaktorit koos PlatformIO laiendusega. Arendusplaadi programmeerimine toimus standardse USB-C pesa kaudu. Programmeerimise keeleks oli C-keel koos Arduino spetsiifiliste funktsioonidega.

Arendusplaadi programeerimise üldpõhimõte

Arendusplaadi keskmeks on ESP32-Wroom-32 süsteemikiip, mille kaudu suheldakse nii plaadil asuvate seadmetega kui ka väliste lisaseadmetega.

Tootja plaadile spetsiaalseid tarkvaralisi draivereid ei paku. Peale plaadi riistvaralise versiooni kindlaks tegemist tuleb leida sobiv konfiguratsioonifail, mis määrab plaadil asuvate seadmete parameetrid. Plaadil asub SIM800L GPRS modem ja alltoodud väljavõttes on toodud modemi tööks vajalik konfiguratsioon (modemiga suhtlus käib UART protokolli vahendusel ja kasutuses on ESP32 kontrolleri ühendused 26 ja 27).

Tasub meeles pidada, et arendusplaadi ühendusi, mis on juba sisemiselt kasutuses väliste seadmete jaoks enam kasutada ei saa. Kuna tootja kasutab plaadil ka ametlikult dokumenteerimata ühendusi, siis väliste seadmete ühendamisel võib kuluda kõvasti aega ja tekkida ebameeldivaid üllatusi. Peale õige riistvaralise versiooni ja sobiva konfiguratsioonifaili leidmist seade käivitus ja sai hakata tegelema lisaseadmete programmeerimisega.

Väljavõte konfiguratsioonifailist:

```
// ESP32 LilyGo-T-Call-SIM800 SIM800L_IP5306_VERSION_20190610 (v1.3) pins defi-
nition
#define MODEM_RST 5
#define MODEM_PWRKEY 4
#define MODEM_POWER_ON 23
#define MODEM_TX 27
#define MODEM_RX 26
#define I2C_SDA 21
#define I2C_SCL 22
#define LED_GPIO 13
#define LED_PIN 13
#define LED_ON HIGH
#define LED_OFF LOW
#define IP5306_ADDR 0x75
#define IP5306_REG_SYS_CTL0 0x00
```

[44] https://github.com/Xinyuan-LilyGO/LilyGo-T-Call-SIM800 (14.01.2024)

SIM800L GPRS modemi programmeerimine

LilyGo-T Call arendusplaadi oluline komponent on SIM800L GPRS modem[45]. Komponenti on toodetud kaua ja sobiv tarkvaraline tugi olemas. Suhtlus ESP32 kontrolleri ja modemi vahel käib UART protokolli vahendusel, lisafunktsioonide jaoks nagu näiteks "*deep sleep*" režiim kasutatakse I2C protokolli. Modem kasutab suhtlemiseks AT käsustikku (*AT command set*)[46], mille kasutamine on suhteliselt tülikas. Leidub TinyGsm[47] teek, mis toetab ka SIM800L modemi riistvara ja teeb modemi programeerimise oluliselt lihtsamaks. TinyGsm teegis on pikad ja kohmakad AT käsud asendatud samaväärsete C-keelsete funktsioonidega. Tasub märkida, et TinyGsm teeki saab kasutada koos StreamDebugger[48] teegiga, mis võimaldab erinevate vigade otsimist oluliselt kiirendada.

Näide AT käskude kasutamisest:

```
Connecting to APN: AT+CIPSHUT
AT+CGATT=0
+SAPBR 1: DEACT
AT+SAPBR=3,1,"Contype","GPRS"
OKAT+SAPBR=3,1,"APN",""
AT+CGDCONT=1,"IP",""
AT+CIPSTART=0,"TCP","demo.thingsboard.io",8883
```

ESP32 sisemise mälu programmeerimine

ESP32-Wroom süsteemikiibil on kokku 4MB SPI prokolli abil kasutatavat mälu. Arduino keskkonnas on võimalik valida, kui suur osa sellest jääb programmi koodi ja kui suur osa andmete jaoks. Need andmed säilivad peale alglaadimist. Käesolevas projektis kasutatakse SPI mälu mõningate parameetrite salvestamiseks.

Näide SPI mälu kasutamisest, kirjutamine mälusse:

```
#include "FS.h"
#include "SD.h"
#include "SPI.h"
#include "SPIFFS.h"
SPIFFS.begin()
File file = SPIFFS.open("/test.txt", FILE_WRITE);
file.print("TEST")
file.close();
```

PZEM-004T

Tegemist on seadmega, mis võimaldab mõõta vahelduvvoolu, pinget , sagedust, aktiivset energiat ja aktiivset tarbimist. ESP32 kontrolleriga toimub ühendus UART ühenduse kaudu kiirusega 9600 bits/s. Protokollina on kasutusel Modbus-RTU. Tootja otsest Esp32 süsteemikiibi tuge teekide näol ei paku, kuid dokumentatsioonis on seadme funktsioonid ja parameetrid täpselt kirjeldatud. Githubi keskkonnast leiab vabavaralise teegi[49] , mis teeb seadme kasutamise lihtsaks. Teek sisaldab korralikku dokumentatsiooni ja töötavaid näiteid.

[45] https://simcom.ee/documents/SIM800/SIM800%20SPEC_20170914.pdf (04.03.2024)
[46] https://www.itu.int/rec/T-REC-V.250/e (08.02.2024)
[47] https://github.com/vshymanskyy/TinyGSM (08.02.2024)
[48] https://github.com/vshymanskyy/StreamDebugger (08.02.2024)
[49] https://github.com/mandulaj/PZEM-004T-v30 (08.02.2024)

Näide kasutatavatest funktsioonidest:

```
//Initsialiseermine
#include <PZEM004Tv30.h>
PZEM004Tv30 pzem(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN);
//Andmete lugemine
 float voltage = pzem.voltage();
 float current = pzem.current();
 float power = pzem.power();
 float energy = pzem.energy();
 float frequency = pzem.frequency();
 float pf = pzem.pf();
```

DHT22 temperatuuri ja niiskuse andur

Tegemist on laialt tuntud ja kaua tootmises olnud anduriga[50]. Anduril on mitmed eelised: ühenduseks kasutab ta lisaks toitele ja maandusele ainult ühte kontakti ja võimalik on seadme kasutamine kuni 20 meetri kaugusel. Suhtluseks kontrolleriga on mittestandardset protokolli. Protokoll on tootja poolt hästi dokumenteeritud ja seega probleeme tarkvaralise toega pole. Valida on mitme variandi vahel. Selles töös on kasutatud ESP32 kohandatud Arduino teeki DHTesp[51].

Näide anduri kasutamisest:

```
#include <DHTesp.h>  // DHT for ESP32 library
// DHT object
DHTesp dht;
// ESP32 pin
#define DHT_PIN 32
// Initsialiseerimine
 dht.setup(DHT_PIN, DHTesp::DHT22);
 TempAndHumidity lastValues = dht.getTempAndHumidity();
```

Micro SD kaardilugeja

Kasutab suhtluseks süsteemikiibiga standardset SPI protokolli. Mälukaardiga suhtlemiseks on vajalikud järgmised teegid: SD Library for Arduino[52], failisüsteemi ja SPI protokolli tugi, mis sisaldub juba ESP32 tootja tarkvaras.

Näide SD kaardi kasutamisest:

```
#include "FS.h"
#include "SD.h"
#include "SPI.h"
!SD.begin()
File = SD.open("test.txt", FILE_WRITE);
File.println("TEST");
File.close();
```

RTC reaalaja kell

Tegemist on DS3231 mikroskeemil põhineva seadmega. Kell lisab ajatempli salvestatavatele andmetele. Tarkvaralise toe annab RTClib[53] teek. Ühendus süsteemikiibiga toimub I2C protokolli alusel.

---

[50] http://www.aosong.com/en/products-22.html (08.02.2024)
[51] https://github.com/beegee-tokyo/DHTesp (08.02.2024)
[52] https://github.com/arduino-libraries/SD (18.02.2024)
[53] https://github.com/adafruit/RTClib (18.02.2024)

Näide põhilistest funktsioonidest:

```
#include <Wire.h>
#include "RTClib.h"
RTC_DS3231 rtc;
rtc.begin()
rtc.adjust(DateTime(2023, 6, 18, (timeClient.getHours()),
(tmeClient.getMinutes()), (timeClient.getSeconds())));

DateTime now = rtc.now();
Serial.print(now.hour(), DEC);
Serial.print(':');
Serial.print(now.minute(), DEC);
Serial.print(':');
Serial.print(now.second(), DEC);
Serial.println();
```

Relee moodul väliste seadmete juhtimiseks

Mooduli juhtimine toimub ESP32 GPIO ühenduste kaudu ja spetsiaalseid teeke pole vaja.

Näide relee kasutamisest:

```
// Relee konfigureerimine
const int relay1 = 33;
pinMode(relay1, OUTPUT);
digitalWrite(relay1, LOW);
digitalWrite(relay1, HIGH)
```

5.6 Ülevaade lähtekoodist

Lähtekood asub Github keskkonnas. Koodile ligipääs tagatakse juhendajale ja retsensendile. Ligipääs lähtekoodile on piiratud ettevõtte turvanõuete kaitseks.

Lähtekood on suhteliselt mahukaks, seetõttu on sarnased funktsioonid jaotatud failide kaupa. Arduino keskkonnale tavapäraselt algab programmi töö sama nimega failist millise nimega kataloogis ta asub. Meie puhul siis *Monitoring_device.ino*. Kuna globaalseid muutujaid on palju, siis asuvad need eraldi failis *Utilities.h*, mis laetakse programmi käivitamise alguses. Järgnevalt käivitub *Setup.ino*, mis sisaldab muutujaid ja funktsioone mida käivitatakse programmi alguses üks kord. Seejärel võtab töö üle tsükliliselt töötav *Loop.ino*. Ülejäänud failide järjekord pole määratud, laetakse kõik kataloogis olevad failid. TLS krüpteerimiseks vajalikud sertifikaadid leiab failist *esp_certificates.h*.

Kommentaarid, funktsioonide nimed ja failinimed on inglise keelsed. Lähekoodi on mõõdukalt kommenteeritud.

Nimekiri failidest:

```
RTC_functions.ino
SD_card_functions.ino
SendData.ino
Setup.ino
SIM800L_modem.ino
Utilities.h
esp_certificates.h
LedBlinking.ino
Loop.ino
Monitoring_device.ino
OneButtonFunctions.ino
```

Lähtekoodis on kasutatud elemente Arduino teekidega kaasas olevatest vabavaralistest näidetest. Kasutatud on ka mõningaid funktsioone Random Nerd Tutorials veebilehel olevatest vabavaralistest näidetest [54]. Lisas 3 on püsivara lähtekoodis kasutatud teekide nimekiri koos versiooninumbrite ja tarkvara litsentsidega. Teekides kasutatavad litsentsid tarkvara kasutamist ärilistel eesmärkidel otseselt ei piira.

Kokkuvõtvalt moodustas tarkvara programmeerimine ja testimine kõige suurema osa tööst. Teekide erinevad versioonid ja automaatsed uuendused tekitasid töö alguses asjatut segadust. Näiteks koosneb Arduino ThingsBoard SDK neljast erinevast alamteegist ja iga automaatne uuendus rikkus kogu programmi. Lahenduseks oli automaatsete uuenduste keelamine ja teekide versioonide piinlikult täpne fikseerimine. Õigem oleks olnud kasutada HTTPS või MQTT algseid funktsioone ilma spetsiaalsete ThingsBoard SDK abstraktsioonideta. Lähtekoodi kasvades selgus, et Arduino redaktorist jääb suuremate projektide jaoks tööriistu vajaka. Mahukama lähtekoodi juures on kasulikum kohe algusest alustada VSC (Visual Studio Code) redaktori ja Platform IO tarkvaraga. Lisaks võimaldab VSC kasutada ESP32 süsteemikiibiga koos ka riistavaralist silurit. Hilisem programeerimise keskkonna või redaktori vahetamine on keeruline teekide väikeste dokumenteerimata erinevuste tõttu.

[54] https://randomnerdtutorials.com/ (05.03.2024)

6. Süsteemi seadistamine ja seadme kasutamine

Seireseadme testimiseks ThingsBoard IoT serveriga kasutati Docker platvormi ja VPS (*Virtual Private Server*) serverit. Loodud sai spetsiaalne *docker-compose.yml* fail, mille abil oli lihtne käivitada ThingsBoard MQTT-TLS turvalist ühendust kasutav versioon. ThingsBoard serveri käivitamine Docker keskkonnas on kirjeldatud lisas 4. Käivitunud server ootab MQTT protokolli päringuid TCP portides 1883 (krüpteerimata) ja 8883 (krüpteeritud). ThingsBoard serveri kasutajaliides on kättesaadav HTTPS protokolli kaudu pordis 8080. Uues seadme/kliendi lisamine ThingsBoard keskkonda on kirjeldatud lisas 5.

Seadme kasutamine on tehtud võimalikult lihtsaks. Peale sisselülitamist toimub GPRS ühenduse kontroll. Ühenduse olemasolul seadistatakse reaalaja kell kohaliku aja järgi õigeks. Seejärel toimub MQTT protokolli kaudu ühendumine ThingsBoard serveriga. Ühenduse olemasolul kontrollitakse andmete olemasolu SD kaardil. Vajadusel saadetakse eelmiste sessioonide andmed Micro SD kaardilt serverisse. Kordub ühenduse testi ja andmete saatmise tsükkel. ESP32 süsteemikiibi USB-C ühenduse kaudu on terminali programmi kasutades võimalik saada programmi tööst täpsemat informatsiooni. Ühenduse staatust näitab Led indikaator. Tuleb arvestada, et seadme töö autonoomses režiimis ja energiakulu sõltub GPRS ühenduse kvaliteedist. Testimisel erinevates asukohtades on olnud ühenduse kvaliteet piisav ja seade töötanud korrektselt. Testis olev seade on seni töötanud stabiilselt ja edastanud keskmiselt 10000 andmepunkti ööpäevas.

7. Kokkuvõte

Lõputöö eesmärk oli valmistada reaalselt toimiv seade kliendiseadme asukohas oleva temperatuuri, õhuniiskuse ja toitepinge parameetrite jälgimiseks.

Olulised riistvaralised tingimused olid: autonoomne töö elektrikatkestuse ajal, sõltumatu ühendus keskserveriga ja lisaseadmete ühendamise võimalus. Seadme hind pidi olema soodne ja lõputöö käigus leti lahendus toota seadet ligikaudu 75 eurose riistvarakuluga.

Lõputöö raames valminud seireseade saadab keskserverisse elektrivõrgu parameetrid, temperatuuri, õhuniiskuse ja ajatempli. Seade suhtleb IoT serveriga GPRS ühenduse kaudu. Seadme autonoomne töö on tagatud seadmesse lisatud UPS komponendi abil. Ühenduse puudumisel salvestatakse andmed Micro SD kaardile. Ühenduse taastumisel edastatakse salvestatud andmed koos Unix ajatempliga. Täpse ajatempli loomiseks kasutatakse riitvaralist reaalaja kella. Seadmes on kaks distantsilt juhitavat releed. Elektrikatkestuse korral töötab seade autonoomselt 10 tundi (tööaega on võimalik pikendada akupanga lisamisega). Seade vastab IP45 (tolmu, veekaitse) nõuetele ja seadme hind jääb ettenähtud 75 euro piiridesse. Seadme püsivara programmeerimisel kasutati avatud lähtekoodiga teeke ja C-keelt. Kasutati vabavaralist Arduino keskkonda ja redaktorit.

Olulised tarkvaralised tingimused olid: avatud lähekoodi ja protokollide kasutamine, avalikul võtme krüptograafial põhinev ühendus seadme ja serveri vahel. Üks nõue oli ligipääs edastatud toorandmete andmebaasile.

Töös kasutati IoT keskserverina ThingsBoard CE tarkvara. Tarkavara võimaldab lihtsate vahenditega luua paigaldatud seadmetest ülevaatlikke tabeleid, koostada andmetest kokkuvõtteid ja saata teavitusi. Keskselt on võimalik juhtida seadmete väljundreleesid. ThingsBoard IoT tarkavara on avatud lähtekoodiga, hästi dokumenteeritud. Tarkvara on ettevõtte poolt hallatav ja Docker keskkonnas lihtsalt paigaldatav. Tasulise teenusena pakub ThingsBoard keskkond erinevaid lisasid ja tehnilist tuge. Seadme ühendus IoT serveriga toimub MQTT protokolli kasutades. MQTT ühendus kasutab avaliku võtme krüpteerimist. Edastatud andmed salvestatakse PostgreSQL andmebaasi. Soovi korral on salvestatud andmetele olemas otsene ligipääs, mis võimaldab andmete analüüsiks kasutada näiteks masinõppe meetodeid.

Töö sisaldab tutvustavat ülevaadet kasutatud protokollidest ja kirjeldab süsteemi tööd. Eraldi on loodud failid ThingsBoard serveri installeerimiseks ja seadistamiseks Docker keskkonnas.

Edasiarendustest on plaanis lisada „üle õhu“ uuenduste võimalus ja vähendada seadme energiakulu. Kindlasti on seadme järgmises versioonis kavas kasutada energiasäästlikku LTE-M sidelahendust. Seadme mõõtmete vähendamiseks on edaspidi võimalik tellida tööstuslik kõiki elemente mahutav trükiplaat. Kaaluda võiks energiasäästliku ekraani lisamist.

Eeltoodud infole tuginedes võib väita, et lõputöö kõige olulisem eesmärk, luua seatud tingimustele vastav kasutatav seade, sai täidetud. Õnnestus luua töötav süsteem, mille väärtus on riistvara, tarkvara ja dokumentatsiooni terviklikkus. Hetkeseisuga on olemas reaalselt töötav riist- ja tarkvarast koosnev lahendus, mida klientidele pilootprojektina esitleda.

8. Viidatud kirjandus

1. Linthicum D. Cloud may be overpriced compared to on-premises systems [Internet]. InfoWorld. 2023 [tsiteeritud 3. veebruar 2024]. Available at: https://www.infoworld.com/article/3704228/cloud-may-be-overpriced-compared-to-on-premises-systems.html

2. POW Elite - SONOFF Official [Internet]. 2022 [tsiteeritud 22. jaanuar 2024]. Available at: https://sonoff.tech/product/diy-smart-switches/pow-elite/, https://sonoff.tech/product/diy-smart-switches/pow-elite/

3. Domus I. Sonoff switch complete hack without firmware upgrade [Internet]. Medium. 2017 [tsiteeritud 14. jaanuar 2024]. Available at: https://blog.ipsumdomus.com/sonoff-switch-complete-hack-without-firmware-upgrade-1b2d6632c01

4. Kupka M. Sonoff S26 Wi-Fi [Internet]. Hacking Lab. [tsiteeritud 21. jaanuar 2024]. Available at: https://hackinglab.cz/en/blog/sonoff-s26-wi-fi/

5. Shelly Pro 1PM [Internet]. Easy Smart Home Automation. [tsiteeritud 22. jaanuar 2024]. Available at: https://www.shelly.com/en/products/shop/shelly-pro-1pm

6. Welcome to Shelly Technical Documentation | Shelly Technical Documentation [Internet]. [tsiteeritud 22. jaanuar 2024]. Available at: https://shelly-api-docs.shelly.cloud/

7. Allterco Shelly API access requests [Internet]. Google Docs. [tsiteeritud 22. jaanuar 2024]. Available at: https://docs.google.com/forms/d/e/1FAIpQLSc5UWKugoJCKHVXgTZLs7jusAHF7BlVzO6AdIsSD_CZgUPbOw/viewform?usp=sf_link&usp=embed_facebook

8. Premium Service [Internet]. [tsiteeritud 22. jaanuar 2024]. Available at: https://kb.shelly.cloud/knowledge-base/premium-service

9. Introduction to I^2C and SPI protocols.pdf [Internet]. [tsiteeritud 8. november 2023]. Available at: https://web.eng.fiu.edu/watsonh/intromicros/M14-I2C/Introduction%20to%20I%C2%B2C%20and%20SPI%20protocols.pdf

10. Modbus protocol.pdf [Internet]. [tsiteeritud 8. november 2023]. Available at: https://www.modbus.org/docs/PI_MBUS_300.pdf

11. Tali K. LPWAN raadiovõrkude võrdlus ja kasutusjuhud Tartu näitel. Tartu 2020 [tsiteeritud 10. november 2023]

12. Gaddam SC, Rai MK. A Comparative Study on Various LPWAN and Cellular Communication Technologies for IoT Based Smart Applications. 2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR) [Internet]. 2018 [tsiteeritud 12. veebruar 2024]. lk 1–8. Available at: https://ieeexplore.ieee.org/document/8529060

13. Parmigiani A, Dettmar U. Comparison and Evaluation of LwM2M and MQTT in Low-Power Wide-Area Networks. 2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS) [Internet]. Bandung, Indonesia: IEEE; 2021 [tsiteeritud 8. november 2023]. lk 8–14. Available at: https://ieeexplore.ieee.org/document/9628463/

14. Khwanrit R, Kittipiyakul S, Kudtonagngam J, Fujita H. Accuracy Comparison of Present Low-cost Current Sensors for Building Energy Monitoring. 2018 International Conference on Embedded Systems and Intelligent Technology & International Conference on Information and Communication Technology for Embedded Systems (ICESIT-ICICTES) [Internet]. 2018 [tsiteeritud 14. jaanuar 2024]. lk 1–6. Available at: https://ieeexplore.ieee.org/document/8442066

15. Salazar R, Valencia R, Catota P, Pozo CT, Andagoya-Alba LD. Real-Time Monitoring and Measurement of Electrical Variables Using IoT. 2023 9th International Conference on Computer and Communication Engineering (ICCCE) [Internet]. 2023 [tsiteeritud 25. detsember 2023]. lk 41–5. Available at: https://ieeexplore.ieee.org/document/10246080/authors#authors

16. ME114.pdf [Internet]. 2 Channel 5V Relay Module Technical Specification. [tsiteeritud 4. märts 2024]. Available at: https://cdn-reichelt.de/documents/datenblatt/B300/ME114.pdf

17. Ahmad MZ, Adenan AR, Rohmad MS, Yussoff YM. Performance Analysis of Secure MQTT Communication Protocol. 2023 19th IEEE International Colloquium on Signal Processing & Its Applications (CSPA) [Internet]. Kedah, Malaysia: IEEE; 2023 [tsiteeritud 8. november 2023]. lk 225–9. Available at: https://ieeexplore.ieee.org/document/10087603/

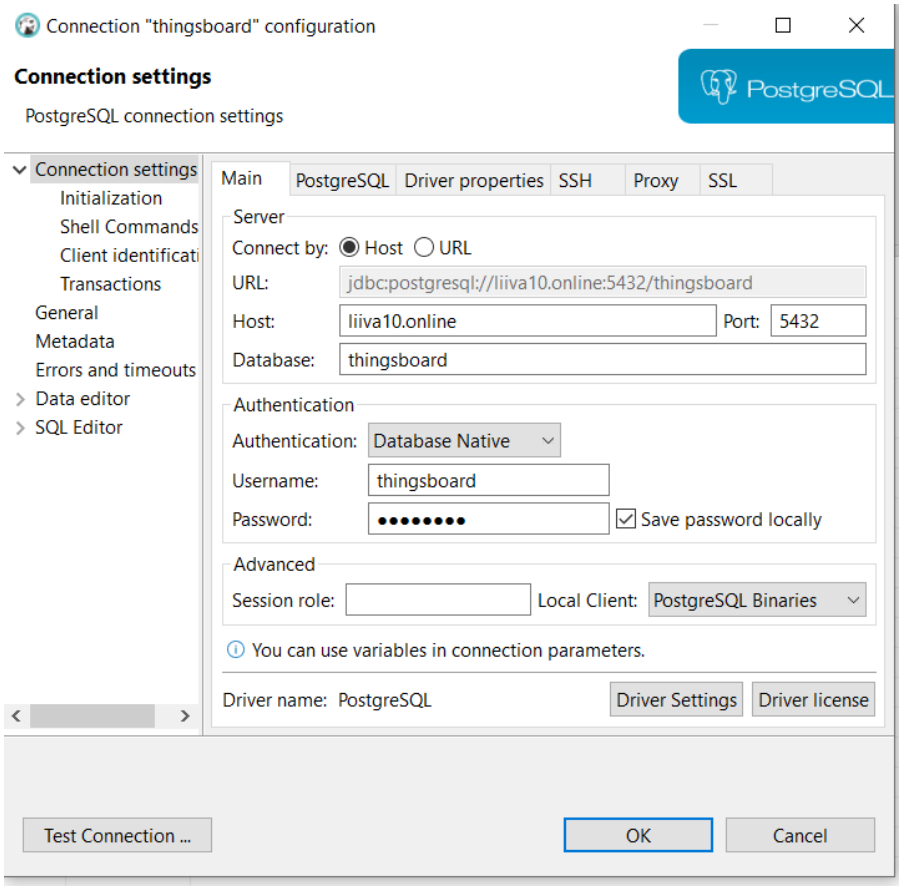
18. Jara Ochoa HJ, Peña R, Ledo Mezquita Y, Gonzalez E, Camacho-Leon S. Comparative Analysis of Power Consumption between MQTT and HTTP Protocols in an IoT Platform Designed and Implemented for Remote Real-Time Monitoring of Long-Term Cold Chain Transport Operations. Sensors [Internet]. 19. mai 2023 [tsiteeritud 8. november 2023];23(10):4896. Available at: https://www.mdpi.com/1424-8220/23/10/4896

9. Lisad

I. DBeaver tarkvara ühendamine PostgreSQL andmebaasiga

docker-compose.yml faili lisada:
Lisada admin port "5432:5432"
Linux serveri ~/.mytb-data/db kataloogis teha muutused:
Failis pg_hba.conf lisada
#IPv4 local connections: alalõik `host all all 0.0.0.0/0 trust`
Failis postgresql.conf lisada
#Connections settings: alalõik `listen_addresses = '*'`

DBeaver ühenduse seadistus



II. Seadme püsivara seadistamine tööks koos ThingsBoard serveriga

Seireseadme koostööks ThingsBoard serveriga peame muutma mõningaid parameetreid programmis.

// Autoriseerimise token, siia läheb eelnevalt genereeritud token

constexpr char TOKEN[] = " wmx5c9y6ol0wn407fi6o ";

// ThingsBoard serveri nimi

//constexpr char THINGSBOARD_SERVER[] = "demo.thingsboard.io";

// Kui kasutame turvalist ühendust

#define ENCRYPTED true

Fail esp_cerificates.h sisaldab ThingsBoard serveri avalikku võtit. Eelnevalt genereeritud failist serveri avaliku võtme failist server.pem tuleb see kopeerida faili esp_cerificates.h muutujasse const char root_ca[] (NB! Järgida realõpu sümboleid).

III. Püsivara teekide versioonid ja litsentsid

Teek	Versioon	Litsents
AdafruitBusIO	1.14.5	MIT License
ArduinoJson	6.21.3	MIT License
DHTsensorlibraryforESPx	1.19	GNU General Public License v3.0
FS	2.0.0	GNU Lesser General Public License v2.1
GovoroxSSLClient	1.1.6	GNU General Public License v3.0
NTPClient	3.2.1	BSD 3-Clause License
OneButton	2.5.0	Software License Agreement (BSD License
PZEM004Tv30	1.1.2	MIT License
RTClib	2.1.1	MIT License
SD	2.0.0	GNU Lesser General Public License v2.1
SPI	2.0.0	GNU Lesser General Public License v2.1
SPIFFS	2.0.0	GNU Lesser General Public License v2.1
TBPubSubClient	2.9.2	MIT License
ThingsBoard	0.11.1	MIT License
Ticker	2.0.0	MIT License
TinyGSM	0.11.7	GNU Lesser General Public License v3.0
Update	2.0.0	GNU Lesser General Public License v2.1
WiFi	2.0.0	Apache License 2.0
Wire	2.0.0	Free Software Foundation 2.1 License,

IV. ThingsBoard serveri installeerimine Docker keskkonnas

ThingsBoard serveri installeerimiseks on palju erinevaid variante. Selles töös kasutatakse serverit Dockeri keskkonnas. Lähtekoodiga samas kataloogis asub selleks otstarbeks loodud spetsiaalne docker-compose.yml fail. Enne faili docker-compose.yaml käivitamist tuleb täita mõned lihtsad eeldused.

Järgnev juhend sobib kasutamiseks Ubuntu Linux serveriga

- Installeerida kasutatavale serverile sobiv Docker ja Docker Compose tarkvara .
- Tekitada Ubuntu serveris kaks kataloogi ThingsBoard serveri andmete jaoks

mkdir -p ~/.mytb-data && sudo chown -R 799:799 ~/.mytb-data

mkdir -p ~/.mytb-logs && sudo chown -R 799:799 ~/.mytb-logs

- Genereerida turvalise TLS ühenduse jaoks vajalikud võtmed

openssl ecparam -out server_key.pem -name secp256r1 -genkey

openssl req -new -key server_key.pem -x509 -nodes -days 365 -out server.pem

- Anda võtmetele õigused

chmod 754 server.pem server_key.pem

- Kopeerida loodud failid server.pem ja server_key.pem kataloogi

~/.mytb-data:/data.

- Käivitada konteiner(docker-compose.yml failiga samas kataloogis):

docker compose up

Docker-compose.yml failis on võimalik määrata MQTT ja HTTPS kasutatavaid porte. Algsed standardsed pordid on MQTT 1883, Secure-MQTT 8883, HTTPS 443. Tasub tähele panna, et tulemüüri tarkvara Ufw ei tööta koos Docker keskkonnaga. On vaja kasutada spetsiaalset versiooni Ufw-Docker.

V. Uue seadme lisamine ja seadistamineThingsBoard keskkonnas

ThingsBoard keskkond pakub ülimalt palju võimalusi seadmete jälgimiseks ja andmete esitamiseks. Järgnevalt lihtne näide uue seadme lisamise kohta.

Peale keskkonda sisselogimist valime Entites-> Devices

„+“ märgiga valime uue seadme, paneme sellele nime ja valime Credentials.

Valime Access token , genereeritakse token ja ongi uus seade lisatud.

Salvestame edasiseks kasutamiseks genereeritud tokeni.

10. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Enn Ehrlich

1. Annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

ESP32 süsteemikiibil põhineva kaughallatava seireseadme välja töötamine ,

mille juhendaja on Alo Peets,

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Enn Ehrlich

07.03.2024