TARTU ÜLIKOOL
Arvutiteaduse Instituut
Informaatika õppekava

Ott Eric Ottender

Elektrihinna kuvaseadme arendamine kasutades ESP-32 süsteemikiipi

Bakalaureusetöö (9 EAP)

Juhendaja: Alo Peets

Tartu 2025

Elektrihinna kuvaseadme arendamine kasutades ESP-32 süsteemikiipi

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärgiks oli arendada ESP-32 süsteemikiibil põhinev seade, mis kuvab reaalajas elektrihindu ning aitab kasutajatel oma tarbimist ajastada odavamatele tundidele. Seadme arendamisel lähtuti tingimustest, et see peab olema odav, kasutajasõbralik ning kasutamine ei nõua digipädevust. Valminud lahendus koosneb ESP-32 süsteemikiibist ja ILI9341 või ILI9488 tüüpi LCD-ekraanist. Tarkvara loodi vabavaralises Arduino IDE keskkonnas ning hinnainfo hangitakse automaatselt Eleringi API kaudu. Töö käigus valmisid kaks prototüüpi erinevate ekraanisuurustega, mis said positiivset tagasisidet test-kasutajatelt. Seadme tugevusteks on selle madal hind, lihtne kasutamine ja sõltumatus nutiseadmetest, mistõttu sobib see eriti hästi näiteks eakatele kasutajatele.

Võtmesõnad: IoT, ESP-32, ILI9341, ILI9488, API, Arduino, elektrihind, kuvaseade

CERCS: T170 Elektroonika; T180 Telekommunikatsioonitehnoloogia, P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Electricity price monitoring device based on ESP-32 development board

Abstract:

The aim of this bachelor's thesis was to develop a device based on the ESP-32 system-on-chip that displays real-time electricity prices and helps users schedule their consumption for cheaper hours. The design requirements emphasized affordability, user-friendliness, and minimal digital skills from the user. The final solution consists of an ESP-32 microcontroller and an LCD display (either ILI9341 or ILI9488 model). The software was developed using the open-source Arduino IDE, and electricity price data is automatically fetched via the Elering API. During the project, two prototypes with different screen sizes were created, which were tested in a pilot project with potential customers. The strengths of the device include its low cost, ease of use, and independence from smartphones or computers, making it particularly suitable for elderly users.

Keywords: IoT, ESP-32, ILI9341, ILI9488, API, Arduino, electricity price, monitoring device

CERCS: T170 Electronics; T180 Telecommunication engineering; P170 Computer science, numerical analysis, systems, control

Sisukord

1. Sissejuhatus

Alates 2010-ndast aastast on Eesti majapidamistel olnud võimalus lisaks fikseeritud hinnale osta elektrit ka turuhinnaga, mis määratakse Nord Pool Groupi poolt. Hinnad tekivad 24-tunnise etteteatamisega ning muutuvad iga tunni tagant [1]. Sarnaselt muude vaba turu kaupadega, kujuneb ka elektri hind nõudluse ja tootluse suhte tulemusel. Üheltpoolt üritatakse ennustada inimeste potentsiaalset tarbimist, teisalt arvestatakse näiteks järgmise päeva päikesepaistet või tuulekiirust toodetava elektri koguse ennustamiseks. Seda põhjusel, et elektri salvestamiseks pole turul täna piisavalt lahendusi. Seetõttu on oluline, et tarbimine ja pakkumine oleksid võimalikult tasakaalus. Vähese tarbimise ja rohke tootmise korral läheb elekter raisku, vastupidisel juhul tekivad elektri puuduse tõttu elektrikatkestused [2].

Elektrihind võib kõikuda 24 tunni jooksul väga palju – mõnel päeval lausa 300% – mistõttu võib elektri kasutamise ajastamine olla inimesele suureks kokkuhoiu allikaks. Näiteks üks kuivatustsükkel läheks 4 kWh elektrit tarbiva kuivati puhul maksma odavamal tunnil vähem kui üks sent, kõrgemal tunnil aga 0.92 senti. Näiteks Norra kahes maakonnas läbiviidud uuringu tulemusena säästeti 2022. aastal targema elektrikasutuse tulemusena kokku 9.8 millionit Norra krooni ehk pisut alla miljoni euro [3]. Kusjuures, kõrgeimat ja madalaimat hinda võivad eraldada ainult mõni tund. Siinkohal on aga oluline meeles pidada, et ajastamise aluseks on elektrihindadega kursis olemine. Hetkel on võimalik elektrihinda jälgida Nord Pooli[1] või ka Eleringi[2] veebilehel, lisaks ka äppides nagu Elektrihind[3] või Enefit Eesti[4]. Nendest keskkondadest elektrihinna kätte saamine nõuab aga inimeselt digipädevust ja on enamasti tülikas. Teisalt on võimalik paigaldada koju nutipistikud, mis võtavad iseseisvalt arvesse elektrihinna kõikumist, kuid ka nende haldamine nõuab äppide olemasolu ja kasutamisoskust.

[1] https://data.nordpoolgroup.com (03.03.2025)

[2] https://dashboard.elering.ee/et/nps/price (03.05.2025)

[3] https://apps.apple.com/lt/app/elektrihind/id1601718250 (03.03.2025)

[4] https://apps.apple.com/ee/app/enefit-eesti/id898742562 (03.05.2025)

Kõige suurem potentsiaal on elektri hinda manuaalselt ajastada inimestel, kes viibivad peamiselt kodus ja kellel on rohkem vaba aega – pensionäridel. Siin aga kujuneb probleemiks asjaolu, et vanematel inimestel pole tihti nutitelefoni ja/või arvutit. Olemasolu korral on aga sageli probleemiks vähene kasutusoskus, mistõttu võib elektrihinna kättesaamine olla neile raske.

Seetõttu oleks vaja lahendust, mis teeks elektrihinna neile lihtsasti kättesaadavaks ning ei nõuaks kasutajalt digipädevust. Elektrihinda kuvav seade aitaks eelkõige pensionäridel, aga ka teistel vabalt turult elektrit ostvatel inimestel tarbimise ajastamisega elektrikuludelt kokku hoida. Käesolev töö kirjeldabki kuidas tehniliselt kompetentne isik saab hästivarustatud elektroonikapoes leiduvatest komponentides ise kokku panna elektri hinda kuvava seadme.

Esmalt tutvustan loodavale seadmele kehtivaid riist- ja tarkvaralisi ning füüsilisi nõudeid, lisaks olemasolevaid alternatiivseid lahendusi. Seejärel põhjendan valitud riistvaralisi komponente ning nende eeliseid samalaadsete toodete ees. Järgneb kirjeldus ning analüüs loodud tarkvaralistest protseduuridest. Viimaks tutvustan loodud seadme seadistamist, kasutamist ning testimist. Lisades on leitav ligipääs seadme täispikale lähtekoodile.

2. Mõisted, terminid ja lühendid

ADC	*Analog-to-Digital Converter,* analoog-digitaal konverter
API	*Application Programming Interface*, rakendusliides
CS	*Chip Select*, kiibi valiku viik.
DAC	*Digital-to-Analog Converter,* digitaal-analoog konverter
DC	*Data/Command*, andmete ja käskude vahetamise viik
ESP-32	Espressif Systemsi mikrokontrollerite seeria
GND	*Ground*, maandus
GPIO	*General Purpose Input/Output,* üldotstarbeline sisend/väljund
GUI	*Graphical User Interface*, graafiline kasutajaliides
IDE	*Integrated Development Interface*, programmeerimise keskkond
IoT	*Internet of Things*, asjade internet
JSON	*JavaScript object notation*, JavaScripti objektide notatsioon
LCD	*Liquid Crystal Display*, vedelkristallekraan
LED	*Light Emitting Diode*, valgusdiood
MISO	*Master In Slave Out*
MOSI	*Master Out Slave In*
NTP	*Network Time Protocol*, võrguaja protokoll
RGB	*Red, Green, Blue (display)*, puna-rohe-sinine ekraan.
RST	*Reset,* lähtestusviik
SCK	*Serial Clock,* seerianäidik/kellaviik
SD	*Secure Digital (memory card)*, turvaline digitaalkaart
SPI	*Serial Peripheral Interface*
SSID	*Service Set IDentifier; network name*, võrgu nimi.
TFT	*Thin-Film Transistor*
UART	*universal asynchronous receiver/transmitter,* universaalne asünkroonne vastuvõtja/saatja
VCC	*Voltage Common Collector,* toide/pingesisend

3. Seadme nõuded ja analoogsed seadmed

Peatükis käsitletakse loodava seadme toimimiseks vajalikke riistvaralisi, tarkvaralisi ning füüsilisi nõudeid, mis tulenevad nii tehnilistest eesmärkidest kui ka kasutajamugavuse kaalutlustest. Lisaks antakse ülevaade analoogsetest seadmetest turul, et hinnata olemasolevaid lahendusi ning tuua välja nende tugevused ja kitsaskohad võrreldes käesolevas töös arendatava lahendusega.

3.1. Riistvaralised nõuded

- Wi-Fi ühendus. See on oluline, sest seade saab kogu informatsiooni internetist. Kui internetiga ühendust pole, ei ole mingit võimalust hinnainfo saamiseks.
- Protsessor peaks olema piisavalt võimekas, et kogu programmi otsast peale iga tund täita. See nõuab muuseas ka API-st pärit JSON andmete töötlemist, mis võib kujuneda üsna ressursimahukaks.
- Toiteallikaks võiks olla patareid. See on oluline, kuna muidu peab seadme paigaldamiseks sinna juurde ka elektrijuhtmed tooma, mis võib tekitada mõttetut lisakulu. Patareidega seadet saab aga ükskõik kuhu kinnitada.
- Kogu riistvara maksumus võiks olla võimalikult madal, et hoida hind võimalikult madalal. Odav hind on tähtis selleks, et kõik inimesed saaksid seda endale lubada ning et toode teeniks enda hinna säästetud elektriarve pealt võimalikult kiiresti tagasi.

3.2. Tarkvaralised nõuded

- Avatud lähtekoodiga.
- Koodil ei ole peamise kasutamisvoo täitmisega probleeme.
- Seadme seadistamisel on võimalik valida õige riik, kus elektrit tarbitakse. Kuna elektrit saab turuhinnaga osta ka teistes Baltiriikides ja Skandinaavias, peaks olema võimalik ka vastav riik valida, kus parasjagu viibitakse. Sellega koos peaks muutuma ka ajatsoon.
- Seadme kuvatud hinnad on õiged. Juhul, kui seadme kuvatud hinnad on valed, on tegemist mõttetu ja kahjuliku vidinaga.
- Hinnainfot värskendatakse iga tunni järel. On oluline, et kasutajad saaksid võimalikult varakult uue hinnainfo teada, et oma elektrikasutust paremini ajastada.

3.3. Füüsilised nõuded

Seadme dimensioonid oleksid kasutajasõbraliku suurusega. Seade ei tohiks olla liiga väike, et piiratud nägemisvõimega inimesed ei näeks sealt vajalikku hinnainfot lugeda. Samas, ei tohi olla seade ka liiga suur, võttes seinal mõttetult palju ruumi, oleks visuaalselt häiriv või raske.

3.4. Analoogsed lahendused

Hetkel olemasolevatest alternatiividest ei tuvastanud ma ühtegi, mis vastaks täpselt minu loodud seadmele. Esimeseks kaudseks alternatiiviks on näiteks elektrihinda kuvavad äpid ja veebilehed, kuid neid ma laiemalt analüüsima ei hakka. Järgnevalt on välja toodud peamised alternatiivid.

TarkHoone HeatAdapt börsimoodul

TarkHoone[5] on Eesti firma, mis tegeleb nutikodu tehnoloogiate paigaldamisega. Nende HeatAdapt börsimoodul on loodud spetsiaalselt erinevatele soojuspumpadele, mis võtab arvesse nii elektrihinda, võrgutasusid kui ka ilmaprognoosi [4]. Seade paigaldatakse soojuspumba ja toiteallika vahele, mis vastavalt tarkvara poolt tehtud arvutustele kas edastab soojuspumbale voolu või mitte. Ühtlasi on võimalik kogu tööd hallata nende äpist ning seejuures jälgida ka elektrihinna kõikumist.

Siiski on antud lahendusel mõned puudujäägid:

- Loodud spetsiaalselt soojuspumpadele, mistõttu teiste seadmete kontrollimine ei pruugi olla mugav.
- Nõuab kasutajalt digipädevust ja nutitelefoni/arvuti olemasolu ja kasutamisoskust.
- Olgugi, et täpne hind oleneb soojuspumba tüübist, tootjast ja hoone asukohast võib arvestada vähemalt 3-kohalise väljaminekuga, mis teeb seadme üsna kalliks.

[5] https://tarkhoone.ee/ (20.04.2025)

Celeon Miniatuurne tark pistikupesa

Nagu TarkHoone, on ka Celeon Eesti firma. Nende peamiseks tegevusalaks kaugjuhitavate seadmete müük, mis võtavad arvesse ka elektrihinda. Celeon pakub mitmekülgset lahendust, alustades tarkadest pistikupesadest, mis sobib enamusele kodutehnikale, lõpetades spetsiaalselt kütteseadmetele loodud lahendustega[6]. Siinkohal keskendun eelkõige nende miniatuursele targale pistikupesale, kuna see on hinnaklassilt ja otstarbelt kõige sarnasem minu loodava seadmega.

Seade paigaldatakse pistikupessa, kuhu seejärel ühendatakse kodutehnika. Tänu seadmes olevale Wi-Fi releele on võimalik seadet Celeoni äpist kontrollida ja seadistada vastavalt kliendi soovile. Ühtlasi saab seadistada seadme töötama kindlate intervallidega [5].

Siiski on seadmel ka olulised puudujäägid:

- Seadme kontrollimine nõuab kasutajalt digipädevust ja nutitelefoni/arvuti olemasolu ja kasutamisoskust.
- Seadme hind on üsna kallis: €99,00. See võib muuta seadme potentsiaalsetele kasutajatele kättesaamatuks ning ei pruugi ennast ära tasuda.

Alternatiivide kokkuvõte

Kokkuvõttes on alternatiivide puhul peamiseks probleemiks nende hind ning digipädevuse nõudmine. Seetõttu valin enda seadmesse töökindlad, kuid odavad komponendid ning loon disaini, mis ei eelda kasutajalt nutiseadmete kasutamise oskust. Lisaks kommertstoodetele üritasin leida ka teisi kodus meisterdatud elektrihinna jälgimisseadmeid, kuid leitud variandid tegelesid pigem isetoodetava taastuvenergia kasutamise planeerimisega või elektritarbimise analüüsiga, mitte planeerimisega.

[6] https://celeon.eu/ (20.04.2025)

4. Riistvara

Peatükis on kirjeldatud loodud seadme riistvaralist ülesehitust. Esmalt on võimalik lugeda valitud komponentide kohta ning miks valik just neile taandus. Viimaks saab tutvuda ka riistvara skeemiga.

4.1. Komponendid

Seadme toimimise aluseks on hoolikalt valitud riistvarakomponendid, mis tagavad süsteemi töökindluse, energiatõhususe ja kasutajamugavuse. Järgnevalt kirjeldatakse peamisi komponente, nende funktsiooni süsteemis ning valiku põhjendusi vastavalt seadme eesmärgile ja töötingimustele.

ESP-32 süsteemikiip

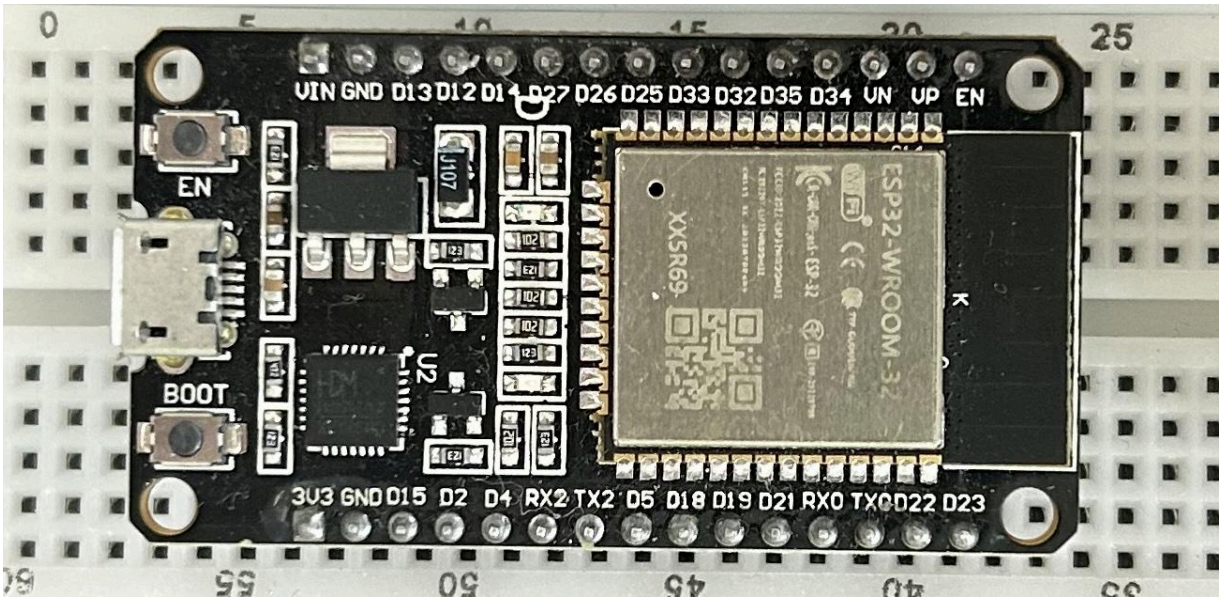
ESP-32 süsteemikiip on Espressif Systemsi poolt välja töötatud mikrokontrollerite seeria, mis on tuntud oma laia kasutuse poolest IoT projektide puhul. ESP-32 kiipi eelistatakse peamiselt selle odava hinna tõttu. Lisaks sellele langeb valik tihti ESP-32 mikrokontrollerile Wi-Fi ja Bluetooth-i ühenduse olemasolu, vähese energiatarbimise ja paljude sisend/väljund võimaluste tõttu [6]. ESP-32 kiipe on ka erinevaid vastavalt kasutaja vajadustele – on loodud eraldi kiip ESP32-S3, mis soodustab tehisintellektiga töötamist [7], samas ka kiip ESP32-C5, mis on suunatud suurema arvutusvõimsuse jaoks [8]. ESP-32 kiibid saavad anda edasi nii 3,3V kui ka 5V voolu ning tänu UART, ADC ja DAC välisseadmete toele on see väga hea valik IoT seadmete ehitamisel, kuna need võimaldavad erinevaid lisasid kiibiga ühendada, näiteks erinevaid sensoreid. Lisaks on sellel ka avatud lähtekood, mis teeb ka hobikorras arendajatele värkvõrgu projektid oluliselt lihtsamaks ja kättesaadavamaks.

Enda loodud seadmes otsustasin kasutada DORHEA ESP-32S (nähtav joonisel 1) kiipi eelkõige odava hinna ning sobiva(te) jõudluse ja lisade tõttu. Näiteks Amazonist[7] on 3 kaupa tellides ühe kiibi hinnaks keskmiselt 5-6 eurot. Hiinast[8] tellides langeb kiibi hind veel kahe-kolme euro võrra.

[7] https://tinyurl.com/ycxd2vy (20.04.2025)

[8] https://tinyurl.com/hputwuvs (03.05.2025)

Potentsiaalseteks alternatiivideks valitud kiibile oleks olnud ka teiste prototüüpimisel levinud firmade tehtud kiibid nagu Arduino Nano[9] või Raspberry Pi[10]. Nende tugevuseks oleks kindlasti nende toetussüsteemi – täpsemalt foorumite, abimaterjalide ja dokumentatsiooni – rohkus. Mõlema alternatiivi puhul oli minu jaoks määravaks hind. Nii Arduino Nano kui Raspberry Pi süsteemikiibid, mis oleksid samasuguste (jõudlus)näitajatega nagu valitud ESP-32S kiip, oleksid olnud vähemalt kaks korda kallimad.



Joonis 1. ESP-32 Süsteemikiip

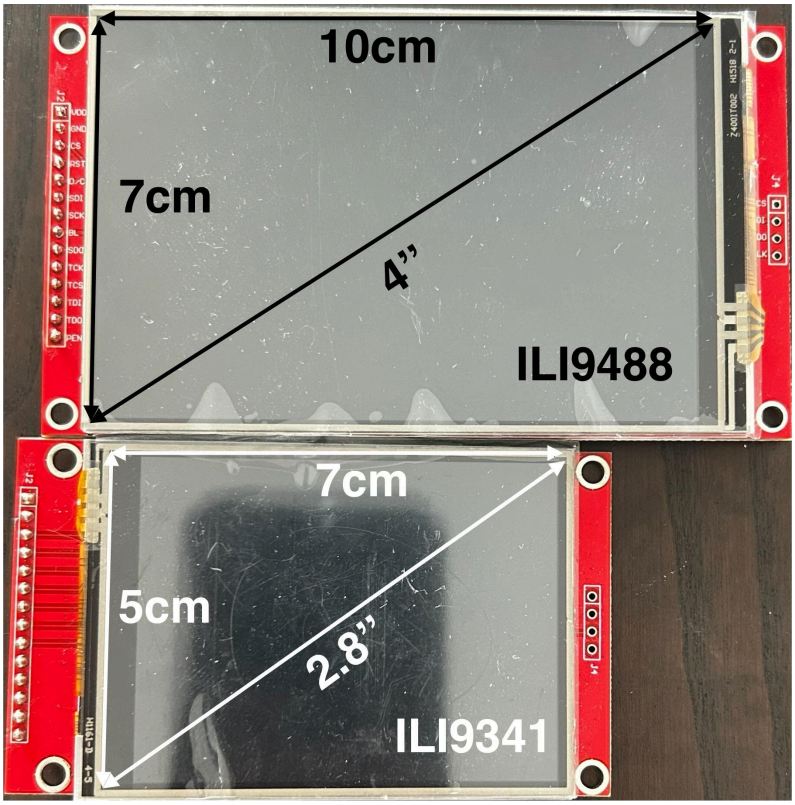
ILI9341/ILI9488 TFT LCD ekraan

Töö käigus valmis kaks erinevat seadet, millel on pisut erinevad ekraanid. ILI9341 ja ILI9488 on mõlemad LCD RGB ekraanid, mis töötavad 4-juhtmelise SPI ühendusega. Ühtlasi on mõlemal ka SD kaardi sisend ja võimalus muuta ekraan puutetundlikuks [9]. Mõlemad tüübid on populaarsed valikud IoT projektide puhul oma odava hinna ja lihtsa kasutuse tõttu. ILI9341 tüüpi ekraane kasutatakse pigem väiksemate, kuni 2,8 tolliste ekraanide puhul, mille resolutsioon on 240x320 pikslit. ILI9488 tüüpi ekraanid on muus osas sarnased ILI9341 ekraanidega, aga üldiselt on nad nii suurema füüsilise ekraani mõõtmetega kui ka ekraanilahutusega.

[9] https://tinyurl.com/chn69h74 (04.05.2025)

[10] https://www.raspberrypi.com/products/ (04.05.2025)

Enda loodud seadmes kasutasin väiskema seadme puhul 2,8-tollist 240x320 resolutsiooniga ILI9341 ekraani ning suurema seadme puhul 4-tollist 320x480 resolutsiooniga ILI9488 ekraani. Mõlemad ekraanid on toodetud WWZMDiB poolt, mis tegeleb IoT komponentide müügiga. Valisin väiksema ekraani puhul just 2,8-tollise ekraani, kuna leidsin, et see on piisavalt suur, et sellelt oleks kogu vajalik info lihtsalt kättesaadav, aga samas piisavalt väike, et ta ei võtaks mõttetult palju ruumi. Suuremaks ekraaniks valisin 4-tollise, kuna see võimaldab mul lisada ka näiteks hinnagraafiku ning suurem ekraan teeb hinna nägemise üleüldiselt lihtsamaks. Ühtlasi on antud ekraanid üsna odavad: hinnad jäävad mõlemal puhul alla 20 euro[11]. Mõlemad ekraanid on nähtavad joonisel 2.

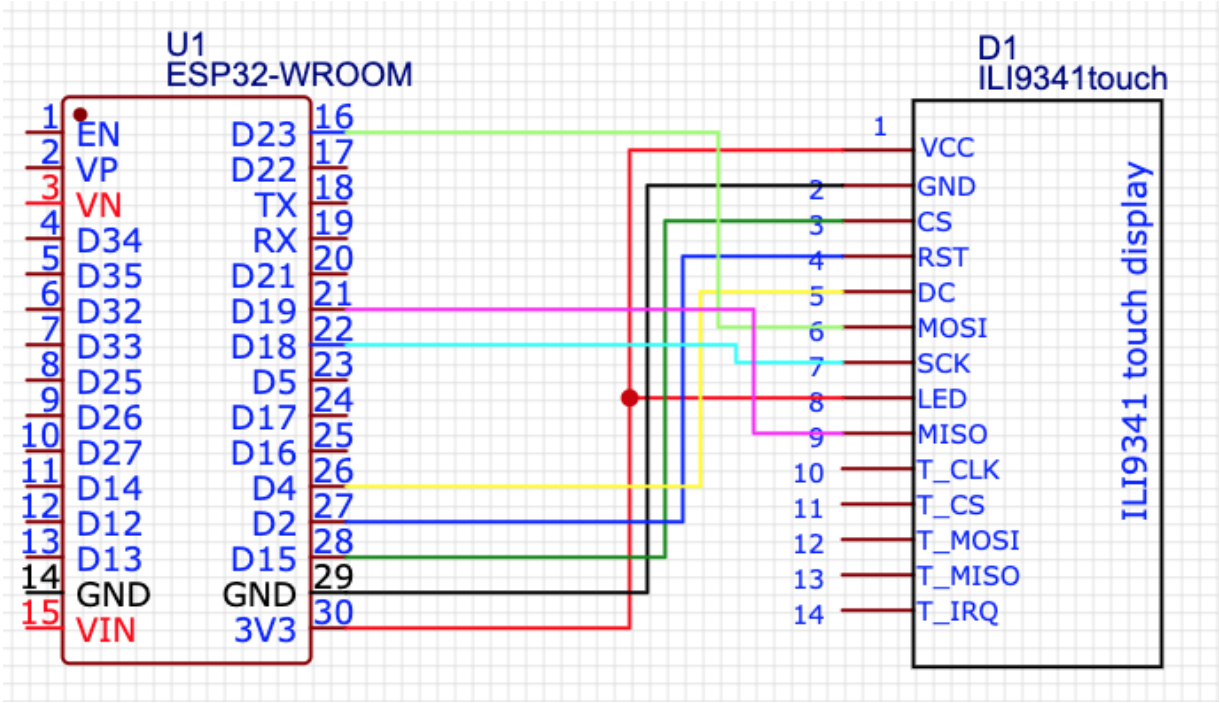


Joonis 2. ILI9488 ja ILI9341 ekraanid

[11]https://www.amazon.com/dp/B0CCHXX1Z9?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1 (20.04.2025)

Ekraanide puhul oleks peamiseks alternatiiviks olnud Raspberry Pi Touch Display 2[12]. Sarnaselt Raspberry Pi poolt pakutavale kiibile, oleks antud ekraani plussiks dokumentatsiooni parem olemasolu ja suurem kasutuskogemus üle maailma, aga taaskord sai määravaks hind. Nimelt algab Touch Display 2 hind kuuekümnest USA dollarist, mis on rohkem kui kolm korda kallim võrreldes valitud ekraaniga. Lisaks sellele oleks Raspberry Pi alternatiiv ka ilmselt liiga suur – ekraani diagonaal on 7 tolli.

4.2. Riistvara skeem



Joonis 3. Riistvara skeem

Ülaltoodud skeem näitab, kuidas seadmes ESP-32 mikrokontroller (joonisel vasakul) ja ILI9341 või ILI9488 ekraan (joonisel paremal) ühendatud on. Skeemi loomiseks kasutasin EasyEDA[13] keskkonda. Ühendatud viigid ja nende selgitused [10] on tabelis 1.

[12] https://www.raspberrypi.com/products/touch-display-2/ (04.05.2025)

[13] https://easyeda.com/ (04.05.2025)

Tabel 1. ESP-32 ja ILI9341/ILI9488 viikide ühendused.

ESP-32 viik	**ILI9341/ILI9488 viik**	**Selgitus**
3V3	VCC	Toide ehk pingesisend
3V3	LED	Vastutab ekraani valguse eest
GND	GND	Maandus
GPIO15	CS	Määrab, milline seade SPI-võrgus parasjagu suhtleb. *Low* = kuulab, *High* = ignoreerib.
GPIO2	RST	Lähtestusviik. Kasutatakse nt seadme taaskäivitamisel.
GPIO4	DC	Andmete ja käskude vahetamise viik. *Low* = käsud, *High* = andmed.
GPIO18	SCK	Vastutab andmeedastuskiiruse ja rütmi eest
GPIO19	MISO	Kasutatakse andmete lugemiseks ekraanilt kiibile.
GPIO23	MOSI	Kasutatakse andmete lugemiseks kiibilt ekraanile.

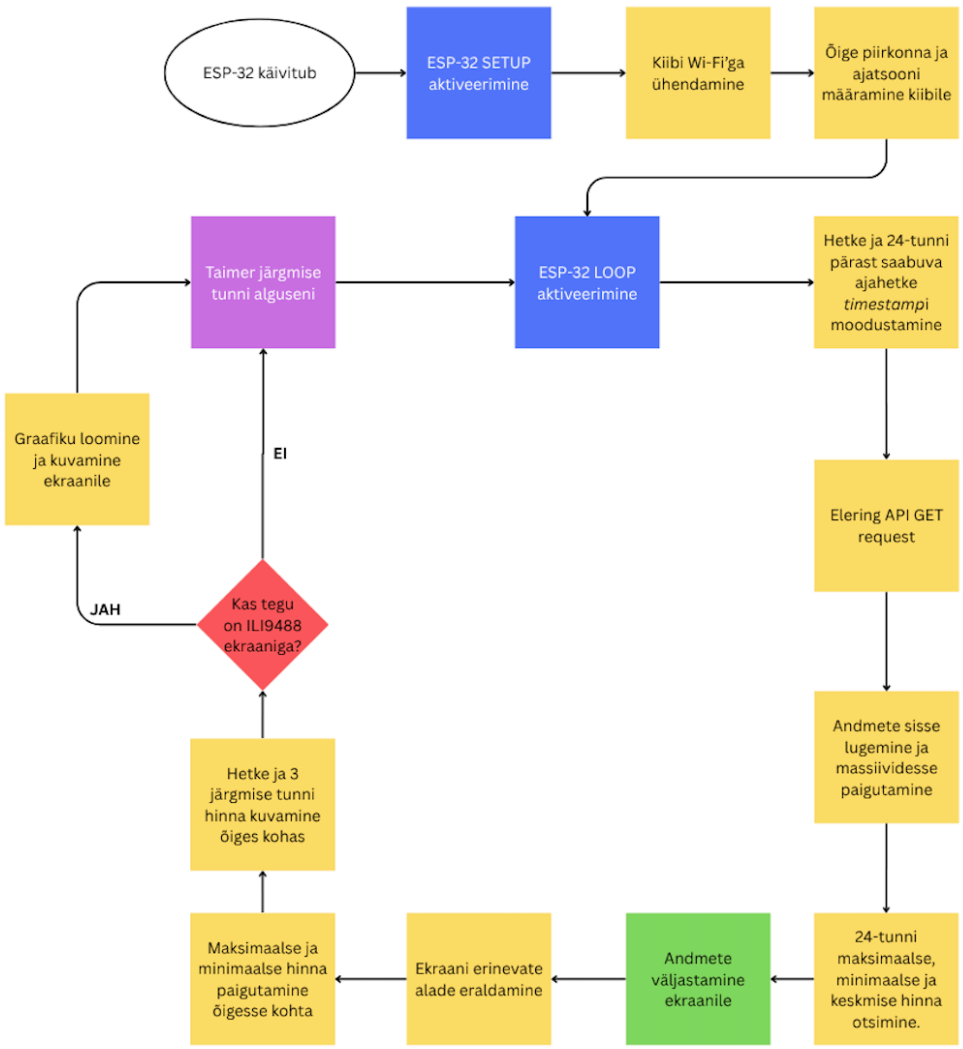
4.3. Riistvara kokkuvõte

Riistvara koosneb peamiselt kahest komponendist: DORHEA ESP-32S süsteemikiibist ja WWZMDiB toodetud ILI9341 või ILI9488 tüüpi ekraanist. Konkreetsed margid ja mudelid valisin alternatiivide ees peamiselt nende hinna ja sobivuse tõttu antud projektile seatud riistvaraliste nõuete tõttu. Ühendasin komponendid omavahel eeltoodud skeemi alusel ning kirjutasin juurde vajaliku tarkvara süsteemikiibi ja ekraani suhtluseks.

5. Tarkvara

Peatükis kirjeldan seadme tarkvaralist poolt. Esmalt annan üleüldise ülevaate programmi tööst kirjeldades lühidalt ka igat programmi sammu. Seejärel keskendun rohkem kasutatud API-le ning ESP-32 süsteemikiibi programmeerimisele. Viimaks toon välja vajalikud teegid ning seletan pikemalt, kuidas kogu hinnainfo ekraanile kuvatakse. Kogu lähtekoodile on võimalik ligi pääseda läbi lisas 1 väljatoodud Githubi lingile.

5.1. Ülevaade programmi tööst peale esmast seadistamist



Joonis 4. Tarkvara plokkskeem.

Järgnevalt on kirjeldatud joonisel 4 nähtava tarkvara plokkskeemi tööd.

1. Kiip ühendatakse vooluallikaga, mis käivitab automaatselt kiibi.
2. Käivitub koodi *SETUP* osa, kus kiip ühendab ennast Wi-Fi võrku. Selleks vajalikud SSID ja salasõna peavad olema koodis eraldi välja toodud.
3. Kiip seab ennast NTP serveriga ühendades koodis määratud õigesse ajatsooni.
4. Algab lähtekoodi *LOOP* osa. Esmalt luuakse API päringu jaoks praegune ja 24 tunni järgne ajatempel. Need lisatakse GET päringusse ja esitatakse veebilehele.
5. Kood loeb kokku, mitme tunni jagu hinnainfot tuli ja loob vastavalt sellele vajalike pikkustega massiivid ajatemplitele ja hindadele. Lähtekood filtreerib välja ainult valitud riigi hinnainfo ja loeb andmed massiividesse sisse.
6. Hinnamassiiv töötatakse läbi ja leitakse madalaim, kõrgeim ja keskmine hind.
7. Kõrgeim ja madalaim hind ning nende aeg, keskmine hind, ajamassiiv ja hinnamassiiv antakse *displayPrices* funktsioonile edasi, mis vastutab hindade kuvamise eest.
8. Funktsioon *displayPrices* alustab kasutajaliidese loomisega, keerates ekraani õigetpidi, värvides tausta mustaks ja jagades ekraani vajalikeks sektoriteks.
9. Esmalt paigutatakse minimaalne ja maksimaalne hind ja nende aeg õigetesse kohtadesse vastavalt *displayMinPrice* ja *displayMaxPrice* funktsioonidega.
10. Paigutatakse õigesse kohta praegune ja järgmise 2 või 3 tunni hind. Arv oleneb ekraani suurusest.
11. Kui tegemist on ILI9488 tüüpi ekraaniga, paigutatakse õigesse kohta ka hinnagraafik, mis kasutab tulba värvi valides keskmist hinda.
12. Programm ootab järgmise tunni alguseni, et naasta sammu nr 4.

5.2. Elering API

Loodud seadme kõige olulisem osa on õige elektrihinna näitamine. Selleks otsustasin kasutada Elering AS-i API-t [11], mis on leitav nende kodulehelt. API dokumentatsioonist valisin välja "/api/nps/price" GET päringu, mis koosneb ka päringu kellaajast ja soovitud lõpukellaajast. Seejärel tagastab päring JSON faili, mis koosneb sisestatud ajavahemikus Eesti, Läti, Leedu ja Soome hindasid. Iga tund on väljendatud UNIX ajatempli ja hinna paarina. API päring ning vastus on näha joonisel 5.

Request URL

```
https://dashboard.elering.ee/api/nps/price?start=2025-05-03T20%3A59%3A59.999Z&end=2025-05-04T20%3A59%3A59.999Z
```


```
{"success":true,"data":{"ee":[{"timestamp":1746306000,"price":64.1200},{"timestamp":1746309600,"price":60.6600},{"timestamp":1746313200,"price":45.2600},{"timestamp":1746316800,"price":32.0600},
{"timestamp":1746320400,"price":77.5000},{"timestamp":1746324000,"price":165.1900},{"timestamp":1746327600,"price":165.1900},{"timestamp":1746331200,"price":36.2300},
{"timestamp":1746334800,"price":33.0100},{"timestamp":1746338400,"price":5.4200},{"timestamp":1746342000,"price":1.0600},{"timestamp":1746345600,"price":7.2900},{"timestamp":1746349200,"price":5.6500},
{"timestamp":1746352800,"price":4.4900},{"timestamp":1746356400,"price":8.2200},{"timestamp":1746360000,"price":9.2000},{"timestamp":1746363600,"price":20.0700},{"timestamp":1746367200,"price":8.2400},
{"timestamp":1746370800,"price":10.2200},{"timestamp":1746374400,"price":61.1200},{"timestamp":1746378000,"price":96.9100},{"timestamp":1746381600,"price":109.6000},
{"timestamp":1746385200,"price":111.1700},{"timestamp":1746388800,"price":77.4900}],"fi":[{"timestamp":1746306000,"price":1.4300},{"timestamp":1746309600,"price":2.1700},
{"timestamp":1746313200,"price":1.7100},{"timestamp":1746316800,"price":0.4000},{"timestamp":1746320400,"price":0.7300},{"timestamp":1746324000,"price":0.4300},{"timestamp":1746327600,"price":0.4600},
{"timestamp":1746331200,"price":0.5000},{"timestamp":1746334800,"price":0.0000},{"timestamp":1746338400,"price":0.4700},{"timestamp":1746342000,"price":-0.0500},{"timestamp":1746345600,"price":-0.1100},
{"timestamp":1746349200,"price":-0.1800},{"timestamp":1746352800,"price":-0.1100},{"timestamp":1746356400,"price":-1.0100},{"timestamp":1746360000,"price":-2.8400},
{"timestamp":1746363600,"price":-0.6100},{"timestamp":1746367200,"price":0.4800},{"timestamp":1746370800,"price":3.1000},{"timestamp":1746374400,"price":3.1600},{"timestamp":1746378000,"price":4.1000},
{"timestamp":1746381600,"price":4.8300},{"timestamp":1746385200,"price":4.9900},{"timestamp":1746388800,"price":6.1200}],"lv":[{"timestamp":1746306000,"price":64.1200},
{"timestamp":1746309600,"price":60.6600},{"timestamp":1746313200,"price":45.2600},{"timestamp":1746316800,"price":32.0600},{"timestamp":1746320400,"price":77.5000},
{"timestamp":1746324000,"price":165.1900},{"timestamp":1746327600,"price":165.1900},{"timestamp":1746331200,"price":36.2300},{"timestamp":1746334800,"price":33.0100},
{"timestamp":1746338400,"price":5.4200},{"timestamp":1746342000,"price":1.0600},{"timestamp":1746345600,"price":7.2900},{"timestamp":1746349200,"price":5.6500},{"timestamp":1746352800,"price":4.4900},
{"timestamp":1746356400,"price":8.2200},{"timestamp":1746360000,"price":9.2000},{"timestamp":1746363600,"price":20.0700},{"timestamp":1746367200,"price":8.2400},{"timestamp":1746370800,"price":10.2200},
{"timestamp":1746374400,"price":61.1200},{"timestamp":1746378000,"price":96.9100},{"timestamp":1746381600,"price":109.6000},{"timestamp":1746385200,"price":111.1700},
{"timestamp":1746388800,"price":77.4900}],"lt":[{"timestamp":1746306000,"price":64.1200},{"timestamp":1746309600,"price":60.6600},{"timestamp":1746313200,"price":45.2600},
{"timestamp":1746316800,"price":32.0600},{"timestamp":1746320400,"price":77.5000},{"timestamp":1746324000,"price":165.1900},{"timestamp":1746327600,"price":165.1900},
{"timestamp":1746331200,"price":36.2300},{"timestamp":1746334800,"price":33.0100},{"timestamp":1746338400,"price":5.4200},{"timestamp":1746342000,"price":1.0600},{"timestamp":1746345600,"price":7.2900},
{"timestamp":1746349200,"price":5.6500},{"timestamp":1746352800,"price":4.4900},{"timestamp":1746356400,"price":8.2200},{"timestamp":1746360000,"price":9.2000},{"timestamp":1746363600,"price":20.0700},
{"timestamp":1746367200,"price":8.2400},{"timestamp":1746370800,"price":10.2200},{"timestamp":1746374400,"price":61.1200},{"timestamp":1746378000,"price":96.9100},
{"timestamp":1746381600,"price":109.6000},{"timestamp":1746385200,"price":111.1700},{"timestamp":1746388800,"price":77.4900}]}}
```


Joonis 5. Elering API päring ning vastus.

5.3. ESP-32 kiibi programmeerimine

Loodud seadme keskmes on ESP-32 süsteemikiip. Kiibi programmeerimiseks on erinevaid mooduseid, oma seadme loomisel kasutasin Arduino IDE-d. Lisaks sellele on võimalik kasutatada ka nt Micropythonit või Visual Studio Code koos Platform IO-ga.

Arduino IDE

Arduino IDE [12] on Arduino poolt loodud vabavaraline programmeerimiskeskkond, mis on tehtud IoT projektide arendamiseks. Programmeerimine käib programmeerimiskeeltele C ja C++ väga sarnase süntaksiga niiöelda Arduino keeles. Arduino IDE teeb veel eriliseks see, et ta on lisaks Arduino enda toodetud kiipidega ühilduv paljude teiste turult saadavate mikrokontrolleritega. Ühtlasi teeb Arduino IDE kasutamise mugavaks oluliste arendustööriistade – kompilaatori, programmeerimisliidese ja jadaliidese terminali – koondamine ühte programmi. See on tähtis, kuna see võimaldab kiibiga kiiret ja vahetut suhtlust. Tänu vabavaralisusele ja suurele kogukonnale on väga lihtne leida ka oma kiibile või lisaseadmetele näidiskoode, mis teevad õppimise oluliselt kergemaks. Näiteks loodud seadmes kasutatavale ESP-32 kiibile loob tarkvara mikrokontrolleri tootja Espressif Systems ise[14]. Ühtlasi on Arduino IDE-l rikkalik teekide loetelu, mis teeb näiteks andurite või ekraanide juhtimise mugavaks. Kui arenduskeskkonnast saadavast informatsioonist jääb puudu, on suure kasutajatehulga tõttu ka mahukaid netifoorumeid, kust saab probleemidele lahendusi leida.

[14] https://github.com/espressif/arduino-esp32 (21.04.2025)

Võrreldes teiste suurte IDE-dega on Arduino loodud keskkond pisut kohmakas ja vanamoodne. Kasutajaliidese kohandamine enda soovidele on üsna piiratud ja pole ka lisafunktsioone, mida leidub nt Visual Studio Code-is või JetBrainsi IDE-des.

Hoolimata Arduino IDE kasutamisele saab koodi soovi korral avada ka teistes uuemates arenduskeskkondades. Töö arenduse otsustasin teha just Arduino IDE keskkonnas, selle algajatele suunatud kiire ja lihtsa kasutamiskogemuse, kuid ka piisava funktsionaalsuse tõttu.

5.4. Vajalikud teegid

Seadme loomisel ja programmeerimisel on kesksel kohal ka kiibi suhtlus ekraani ja internetiga. Selle jaoks kasutasin erinevaid teeke. Esmalt tutvustan internetiühenduse ja API tööga ning seejärel ekraani suhtlusega seotud teeke ja nende implementeerimist.

Ühendus interneti ja API-ga

API ühenduse aluseks on internetiühendus. Selleks kasutan *WiFi.h* ja *HTTPClient.h* teeke. Esmalt loon internetiühenduse *WiFi.begin()* funktsiooniga, mis võtab parameetriteks programmi päises ära märgitud WiFi SSID ja salasõna.

```
#include <WiFi.h>
#include <HTTPClient.h>
const char * ssid = "WiFiSSID";
const char * wifipw = "WiFiPassword";

WiFi.begin(ssid, wifipw);
while (WiFi.status() != WL_CONNECTED) {
   Serial.print(".");
   delay(500);
 }
 Serial.println("\nWifi OK");
```

Peale Wi-Fi ühenduse loomist seadistab kiip ennast õigesse ajatsooni kasutades *TimeLib.h* ja *time.h* teeke. Alloleval väljavõttel on toodud näide aja määramisest.

```
#include <ArduinoJson.h>
#include <TimeLib.h>
```

```
#include <time.h>

configTime(0, 0, "1.pool.ntp.org")
struct tm timeinfo;

if (!getLocalTime(&timeinfo),5) {
   Serial.println("Failed to obtain time");
   return;
 }
 Serial.println("OK, Got the time from NTP");
 setTimezone(timezone); // then set your timezone
}
```

Järgmisena tuleb luua hetke ajatempel ning 24 tunni järgne ajatempel API päringu jaoks. GET päring tagastab seejärel JSON formaadis info ning edasi filtreeritakse ning paigutatakse õige info vastavatesse andmestruktuuridesse. Allpool on toodud väljavõte Elering API-ga ühendamisest ja tagastatud info talletamisest.

```
HTTPClient client;
client.begin("https://dashboard.elering.ee/api/nps/price?start="+makeTodayTimestamp()+
"&end="+makeTomorrowTimestamp());
//https://dashboard.elering.ee/assets/api-doc.html#/nps-controller/getPriceUsingGET
int httpCode = client.GET();
if (httpCode > 0){
   String input = client.getString();
   JsonDocument doc;
   DeserializationError error = deserializeJson(doc, input);
   if (error) {
       Serial.println(error.c_str());
   return;
   }
JsonObject data = doc["data"];
for (JsonObject data_ee_item : data[country].as<JsonArray>()) {
     String data_hour = hour;
     float data_price = data_ee_item["price"];
     prices[i]=data_price;
     hours[i]=data_hour;

     i++;
     hour=nextHour(hour);}
```

Ühendus ekraaniga

Ekraaniga suhtlus käib peamiselt läbi *Adafruit_GFX.h* teegi, mis kontrollib ekraani tööd sarnaselt TKinteri kontrollimisele. Esmalt tuleb aga luua ILI9341 või ILI9488 ekraani isend kasutades ESP-32'ga ühendatud viikide numbreid[15]. Järgnevalt on võimalik hakata ekraanile käsklusi andma, näiteks joonte tõmbamine, teksti kuvamine või kujundite joonistamine. Allpool on toodud näide ekraani kontrollimisest ILI9341 tüüpi ekraani näitel kui soovime kuvada minimaalset hinda.

```
#include "SPI.h"
#include "Adafruit_GFX.h"
#include "ILI9486_SPI.h"
#include "Adafruit_ILI9341.h"

#define TFT_DC 4
#define TFT_CS 15
#define TFT_RST 2
#define TFT_MISO 19
#define TFT_MOSI 23
#define TFT_CLK 18

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_MOSI, TFT_CLK, TFT_RST,
TFT_MISO);
tft.begin();

 tft.setRotation(1); //Sets the direction to horizontal
 tft.fillScreen(ILI9341_BLACK);
 tft.drawLine(200,0,200,240, ILI9341_WHITE); //Drawing the sections
 tft.drawLine(200,120,320,120, ILI9341_WHITE);

 //Displaying minimum price info
 tft.setCursor(255,25);
 tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
 tft.println(min_time); //Display minimum price time

 tft.setCursor(210+calc_y_offset(min_price),50);
 tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
 tft.println(min_price); //Display minimum price
```

[15] https://lastminuteengineers.com/esp32-pinout-reference/ (27.04.2025)

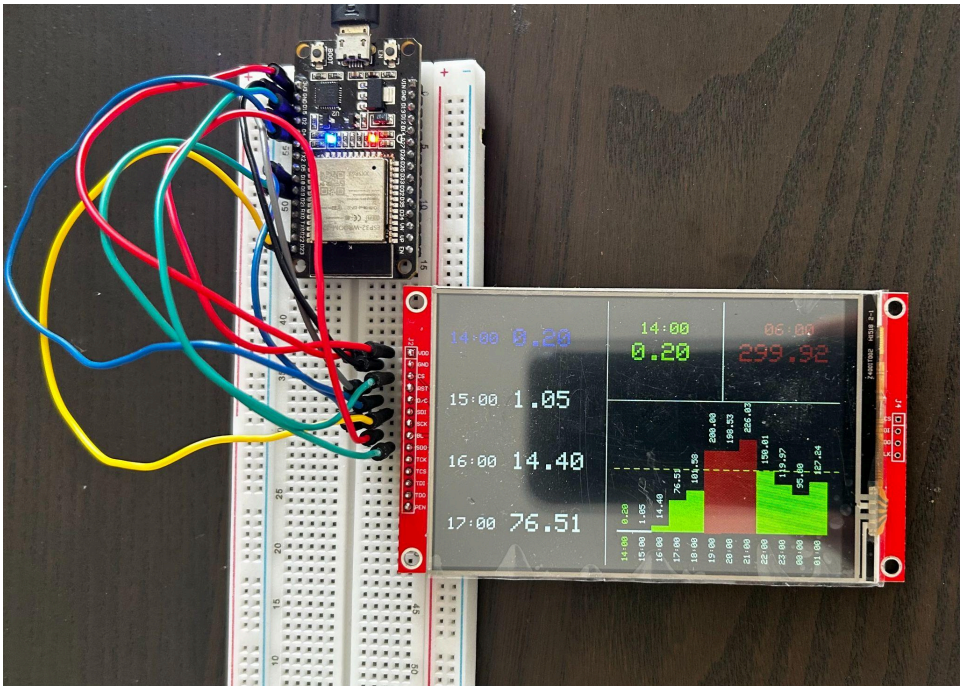
6. Seadme seadistamine, kasutamine ja testimine

Seadme esmaseks seadistamiseks tuleb arvutis avada seadme lähtekood. Seejärel tuleb sisestada koodi päises olevatele väljadele kasutatava Wi-Fi SSID ja parool ning riik (joonis 6), mille elektrihindu näha soovitakse. Peale koodi muutmist tuleb kiip arvutiga ühendada ning kood kiibile laadida. Sellega ongi seadistus tehtud.

```
37    const char * ssid = "ssid";
38    const char * wifipw = "password";
39    String country= "ee"; // "ee" - Eesti, "lv" - Läti, lt - "Leedu", "fi" - Soome
```

Joonis 6. Seadme Wi-Fi andmete ning riigi määramine.

Seadme kasutamiseks tuleb seade vooluvõrku ühendada ja oodata kuni seade tööle hakkab. Seade töötab seni kuni ta on vooluvõrku ühendatud ning rohkem kasutaja poolset abi/sisendit ei vaja. Valminud lahendusest saab pilti näha joonisel 7.



Joonis 7. Valminud seade.

Seadme testimisel minu naabri poolt suuremaid probleeme ei ilmnenud. Esimesel pikemal katsetusel tekkis viga API päringu loomisel, kuna olin lähtekoodis jätnud kahe silma vahele loogikavea. Parandasin vea ning nädalase testimisperioodi jooksul rohkem probleeme ei tekkinud. Suusõnalises tagasisides märgiti, et seade aitas elektrit vabadel päevadel teadlikumalt tarbida ning ajastamine polnud keeruline. Seadme füüsiline olemasolu ja paigutamine pesuruumi tuletas tarbimise ajastamist paremini meelde. Lisaks tuli testijale kohati üllatusena, kui palju võib ühe päeva jooksul elektrihind muutuda. Reaalset kasu seadmest tingitud elektri tarbimise ajastamist on aga raske hinnata, kuna töö kirjutamise ajaks pole veel saabunud uue kuu elektriarvet.

Edasiarendusena sooviksin luua seadmele 3D prinditud ümbrise, et seda saaks ka päriselt seinale kinnitada ning kasutada. Lisaks sellele sooviksin luua riigi ning Wi-Fi SSID ja parooli seadistusvõimaluse ekraanile, et selleks ei peaks koodi eraldi avama. See eeldab samas puutetundlikku ekraani ning keerulisema kasutajaliidese kasutusele võtmist, mis ei mahtunud käesoleva töö skoopi.

7. Kokkuvõte

Lõputöö eesmärk oli valmistada toimiv seade, mis hoiaks kasutajat kursis elektrihinna ja selle kõikumisega näidates jooksva tunni ja kolme järgneva tunni hinda, 24 tunni maksimum- ja miinimumhinda ning suurema ekraani puhul ka 12 tunni hinnagraafikut.

Olulised riistvaralised tingimused olid: Wi-Fi ühendus; piisavalt võimas protsessor; odav riistvara hind.

Lõputöö raames valminud seade koosneb kahest peamisest komponendist: ESP-32 süsteemikiibist ja ILI9341 (diagonaal 2.8 tolli) või ILI9488 (diagonaal 4 tolli) tüüpi LCD ekraanist. ESP-32 ühendub WiFi-ga, saab API päringu tulemusena internetist hinna ning edastab vajaliku info ekraanile kuvamiseks.

Olulised tarkvaralised tingimused olid: avatud lähtekood; seadme seadistamisel on võimalik valida õige riik, kus elektrit tarbitakse; seade töötab 99.9% ajast ning kuvatud hinnad on õiged; hinnainfot värskendatakse iga tunni järel.

Kogu tarkvaraarenduse jooksul kasutati vabavaralist Arduino IDE keskkonda. Tarkvara kõige olulisemaks komponendiks on hinnainfo hankimine Elering API kaudu. ESP-32 ühendub esmalt kohaliku Wi-Fi võrguga, saab hetke ajatempli ning kasutab seda API päringu tegemiseks. GET päringu tulemusena tagastatakse JSON fail, kust loetakse ka vajalik hinnainfo. Seejärel töödeldakse hinnainfo ning ajatemplid vajalikesse andmestruktuuridesse. Viimaks kuvatakse vajalik hinnainfo ekraanil.

Käesolevas bakalaureusetöös antakse ülevaade kasutatud teekidest (põhilisteks *WiFi.h*, *ArduinoJSON.h* ja *Adafruit_GFX.h*) ja selgitatakse süvitsi programmi tööd. Ühtlasi kirjeldatakse ka programmeerimiskeskkonna Arduino IDE tööd ja põhjendatakse selle valikut.

Tuginedes ülaltoodud infole võib järeldada, et lõputöö peamine eesmärk – luua seatud tingimustele vastav ja kasutatav seade, mis suudab kuvada elektrihinda – on edukalt saavutatud. Hetkeseisuga on olemas praktiliselt toimiv lahendus, mis ühendab endas nii riist- kui tarkvara ning mis on peale korpuse loomist valmis ka pikemaajaliseks kasutuseks sihtgrupi poolt.

8. Viidatud kirjandus

[1] Nord Pool Group. https://www.nordpoolgroup.com/en/About-us/History/ (05.03.2025).

[2] Elektrum Eesti OÜ. https://www.elektrum.ee/ee/eraklient/elekter/elektrihinnad (17.04.2025).

[3] Liang, E.Z., Paramonov, S., Madessa, H.B. (2025). Reducing Energy Consumption Through Energy Monitoring Systems: A Case Study in Norway. In: Kioumarsi, M., Shafei, B. (eds) The 1st International Conference on Net-Zero Built Environment. NTZR 2024. Lecture Notes in Civil Engineering, vol 237. Springer, Cham. https://doi.org/10.1007/978-3-031-69626-8_96 (04.05.2025)

[4] TarkHoone. https://tarkhoone.ee/heatadapt/ (05.03.2025).

[5] Celeon. https://celeon.eu/toode/pwo-s/ (20.04.2025).

[6] Byte Pardigm. “Introduction to I^2C and SPI protocols.pdf”. https://web.eng.fiu.edu/watsonh/intromicros/M14-I2C/Introduction%20to%20I%C2%B2C%20and%20SPI%20protocols.pdf (29.04.2025).

[7] Espressif Systems. https://www.espressif.com/en/products/socs/esp32 (05.03.2025).

[8] Espressif Systems. https://www.espressif.com/en/products/socs/esp32-s3 (05.03.2025).

[9] Espressif Systems. https://www.espressif.com/en/products/socs/esp32-c5 (05.03.2025).

[10] LCDWiki. “2.8inch SPI Module MSP2807 User Manual”. http://www.lcdwiki.com/res/MSP2807/2.8inch_SPI_Module_MSP2807_User_Manual_EN.pdf (05.03.2025).

[11] Elering AS. https://dashboard.elering.ee/assets/api-doc.html (20.05.2025).

[12] Arduino. https://www.arduino.cc/en/software/ (21.04.2025).

9. Lisad

9.1. Lisa 1. Valminud tarkvara Github repositoorium

Käesoleva töö raames valminud kood on leitav järgnevas Github repositooriumis: https://github.com/OttEricOttender/StockWatt.

10. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Ott Eric Ottender

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

ESP-32 süsteemikiibil põhineva elektrihinna vaatlusseadme arendamine,

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 4.0, 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.

Ott Eric Ottender

15.05.2025