TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ilja Kuznetsov 205866IACB

ESP32 veebiserver

Bakalaureusetöö

Juhendaja: Vladimir Viies

PhD

Tallinn 2023

BAKALAUREUSETÖÖ ÜLESANDEPÜSTITUS

Kuupäev: 26.02.2023

Üliõpilase ees- ja perekonnanimi: Ilja Kuznetsov
Üliõpilaskood: 205866IACB

Lõputöö teema: ESP32 nagu veebiserverina
Juhendaja: Vladimir Viies

Lahendatavad küsimused ning lähtetingimused:
Mikrokontrolleri ESP32 põhjal universaalse veebiserveri loomine, mis sisaldab järgmisi võimalusi:

1. Toetab mitu erinevat veebirakendusi:
1.1. Töölaud, kus kõik liigipääsetav rakendused asuvad.
1.2. Rakendus SD-kaardile salvestatud failidega töötamiseks (vaatamine, kustutamine, avamine).
1.3. Rakendus, mis näitab teoreetilist võimet suhelda väliskeskkonnaga, näiteks lambipirni sisse/välja lülitada või helisid teha.
1.4. Rakendus, mis näitab teoreetilist võimet lugeda andmeid väliskeskkonnast, näiteks temperatuuri mõõtmiseks.
2. Lihtne ja kiire võmalus uue veebirakenduste käivitamiseks.

Projekti loomisel on peamised arenduskeeled:
- C++. kuna ESP32 mikrokontroller on programmeeritud selles keeles.
- JavaScript / TypeScript veebirakenduste sisemise loogika programmeerimiseks, kuna see keel on selles valdkonnas kõige levinum ja toetatud.
- HTML ja CSS graafiliste rakenduskomponentide loomiseks ja nende kujundamiseks.

Kasutatakse ka abivahendeid, näiteks:

- Vue, üks levinumaid ja mugavamaid veebirakenduste arendusraamistikke.
- Vite, uus kiire ja mugav veebirakenduste paketihaldur, mis käsitleb projekti sõltuvusi ja koodi, optimeerides neid oluliselt ning pakendab need kokku, et rakendusi kiiremini käivitada ja kliendipoolset jõudlust saavutada.

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Ilja Kuznetsov

19.05.2022

Annotatsioon

Antud lõputöös uuritakse teoreetilisi ja praktilisi võimalusi kasutada ESP32 veebiserverina, mis on WiFi kaudu ühendatud Internetiga, võimaldades seda kasutada erinevates valdkondades, nagu väliskeskkonna andmete lugemine, mikrokontrolleriga ühendatud seadmetega suhtlemine, kaugjuurdepääs ja SD-kaardil olevate failide juhtimine.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 15 peatükki, 34 joonist.

Abstract

ESP32 web server

This thesis explores the theoretical and practical possibilities of using ESP32 as a web server connected to the Internet via WiFi, with potential applications in various fields, such as reading data on the external environment, interacting with equipment connected to the microcontroller, as well as remote access and management of files stored on an SD card.

The thesis is in estonian and contains 40 pages of text, 15 chapters, 34 figures.

Lühendite ja mõistete sõnastik

ESP32	Mikrokontroller, mida kasutatakse veebiserveri ehitamiseks.
JavaScript	Populaarne ja levinum programmeerimiskeel, mida kasutatakse veebilehtede arendamiseks.
TypeScript	JavaScriptist välja töötatud keel. Erineb rangema trükkimise poolest. Seda ei käivitata otse kuskil, see kompileeritakse täitmiseks JavaScripti.
Vue	Üks levinumaid ja mugavamaid veebirakenduste arendusraamistikke.
Vite	Vite on tööriist JS frontend projektide koostamiseks, mis muudab arenduse palju lihtsamaks.
HTML	Dokumendi hüperteksti märgistuskeel veebilehtede vaatamiseks brauseris.
CSS	Keel HTML-i abil kirjutatud veebilehe välimuse kirjeldamiseks
Veebirakendus / moodul	Mikrokontrolleri SD-kaardile salvestatud veebileht, millega kasutaja saab suhelda.
React	Teine üks levinumaid ja mugavamaid veebirakenduste arendusraamistikke.
Visual Studio Code (VS Code)	Visual Studio Code on kaasaegne kerge modulaarne IDE erinevates keeltes koodi kirjutamiseks.

Sisukord

Sissejuhatus

Hetkel allub üha suurem osa meie elust informatsioonitehnoloogia mõjule. Arvutustehnika muutub üha kättesaadavamaks, tarkvarad üha keerukamaks ning inimeste vajadused ja soovid kasvavad lõputult nagu trepp, millele ilmub kümmekond lisatrepiastet hetkel, kui jõuate trepi otsani.

Eriti selgelt avaldub see Interneti valdkonnas. Kunagi USA eriteenistuste projekt, mille eesmärk oli tagada võrgus sisemine side isegi siis, kui selle osa kaob, on kasvanud ja muutunud ülemaailmseks nähtuseks, mis ühendab kõiki mandreid ja linnu ning millest tänapäeval ilma mõelda on kui mitte võimatu, siis väga-väga keeruline.

Nagu juba mainitud, tehnoloogia arengu tulemusena saavad arvutiseadmed, eriti mikrokontrollerid, mis on varustatud laia funktsionaalsuse ja head jõudlusega, endale lubada kõik. Ilmneb soov kasutada selliseid ressursse oma vajaduste rahuldamiseks.

Kättesaadavate vahendite abil on võimalik teha äärmiselt palju. Erinevate parameetrite mõõtmine, seadmete sisse- ja väljalülitamine ja paljud muud tegevused - kõike seda saab automatiseerida ja / või teha mugavamaks nende kahe asja - Interneti ja mikrokontrollerite - abil. Sellist funktsionaalsust saab kasutada näiteks nutikodu süsteemis või kaugjuhitava sensori mõõtmiste jaoks.

Projekti eesmärk on veenduda süsteemi loomise võimaluses, mis võib kasutada ESP32 mikrokontrollerit, mis toimib veebiserverina kohalikus võrgus ja võimaldab mugavalt juhtida sellega ühendatud seadmeid kaugjuhtimisega Interneti võrgu standardsete võimaluste abil, kasutades selleks välisliidest kohalikus võrgus asuva veebilehe kujul.

Projekti tulemusena saavutatakse edukas süsteemi loomine ja selle praktiline rakendamine.

Lõputöö koostamisel kasutati alusena projekti, mis oli välja töötatud "IXX1530 Arvutisüsteemide projekt" raames.

1 Kasutatud tööriistad

Antud peatükis vaadatakse läbi kasutatavad tööriistad, seadmed ja antakse ülevaade lõpliku lahenduse üldisest struktuurist.

1.1 Projekti struktuur

Antud projekt võib jagada kaheks osaks - serveripoolseks ja klientipoolseks. Serveripoolne osa koosneb ESP32 mikrokontrollerist, mis kasutab sisseehitatud WiFi moodulit kasutaja poolt saadetud HTTP päringute vastuvõtmiseks ja töötlemiseks, mis saabuvad tema IP-aadressile kohalikus võrgus. Päringute töötlemine hõlmab järgmisi tegevusi:

- Failide saatmine, mis on salvestatud mikrokontrolleriga ühendatud SD-kaardile, samuti nende muutmine/kustutamine vastava päringu kaudu.

- Suhtlemine mikrokontrolleriga ühendatud seadmetega, nagu temperatuuriandurid/LED-tuled.

Klientipoolne osa koosneb rakendustest (veebilehtedest), mis on salvestatud SD-kaardi vastavasse kausta, mis on ühendatud mikrokontrolleriga. Need rakendused võivad omada erinevat funktsionaalsust ja neid võib arendada erinevate JavaScripti raamistike abil, näiteks Vue või React. HTTP päringute abil interneti kaudu suhtlevad need rakendused mikrokontrolleriga, saades/saatades erinevat teavet.

1.2 Varustus

Järgmine seadmed olid kasutatud selle projekti arendamisel:

1. Mikrokontroller ESP32. See mikrokontroller omab märkimisväärseid eeliseid võrreldes kõigi teiste saadaolevate lahendustega, mis olid saadaval, ning sobib ideaalselt selle ülesande jaoks. Nimelt:

- Omab märkimisväärset hulka mälu (4 MB Flashi ja 320 KB operatiivmälu), samas kui näiteks Arduino UNO-l on ainult 256 KB Flashi ja 32 KB operatiivmälu.

- Omab kahte südamikku, mis töötavad sagedusel 240 MHz, samas kui Arduino UNO-l on ainult 16 MHz.

- Omab sisseehitatud WiFi moodulit, samas kui Arduino UNO nõuab välise mooduli ühendamist.

2. 32 GB mälumahtu SD-kaart. Seda SD-kaarti kasutatakse veebilehtede ja muude failide salvestamiseks, mis võivad töö käigus tekkida (näiteks mõõtmiste ajal salvestamiseks).

3. Moodul SD-kaardi ühendamiseks ESP32-ga.

1.3 Tarkvaraosa

Antud lahenduse arendamisel kasutati erinevaid programmeerimiskeeli, mis tulenes tehnilistest piirangutest antud ülesannete lahendamisel.

Seega kasutatakse kahte keelt loogika kirjeldamiseks (C++ ja JavaScript/TypeScript), ühte keelt veebilehe struktuuri kirjeldamiseks (HTML) ja ühte keelt veebilehe stiliseerimiseks (CSS).

C++ kasutatakse serveripoolsel (mikrokontrolleri) poolel, kuna ESP32 mikrokontrolleri programmeerimine toimub peamiselt selle keele abil. Teiste keelte (C#, LUA, JavaScript, Python) kasutamine selle programmeerimiseks on võimalik, kuid see on seotud kas tõsiste seadistusprobleemidega (näiteks spetsiaalse IDE kasutamine ja mikrokontrolleri uuesti programmeerimine) või tõsise jõudluskaduga.

Lisaks tuleb märkida, et arendus toimus peamiselt Wokwi simulatsioonikeskkonnas, mis praegu ei toeta teisi programmeerimiskeeli peale C/C++ piisavalt hästi.

JavaScript/TypeScript kasutatakse kliendi poolel (veebilehel), et lisada sellele interaktiivsust ja on selles valdkonnas hetkel valitsev. Arenduse lihtsustamiseks ja kiirendamiseks kasutatakse JavaScripti raamistikke nagu Vue ja React, näiteks. Nende kasutamine tuleneb asjaolust, et nad pakuvad mugavalt neid võimalusi, mida JavaScriptis oleks ilma täiendava arendusvahendita märkimisväärselt keeruline rakendada (olekute haldamine, teatud elementide atribuutide väärtuste sidumine kindla muutujaga).

HTML (Hüperteksti märgistuskeel) on klientide poolel kasutatav keel GUI struktuuriliste blokkide kirjeldamiseks, mis kuvatakse veebilehel. Seda kasutatakse ainult, kuna see on kõige levinum ja universaalsem lahendus sellistele ülesannetele. Sellel põhjusel kasutatakse koos CSS-iga, et kujundada neid struktuurseid blokke.

2 Ülevaade koodistruktuurist projektis

See osa käsitleb projekti koodibaasi direktoriumide kausta jaotust, selle põhjuseid ning seda, kuidas projekti funktsionaalsust laiendada ja täiendada.

Nagu varem mainitud, on projekt jaotatud kaheks osaks: serveriosa (ESP32) ja kasutajaliidese osa (mida töödeldakse kasutaja seadmes brauseri abil).

2.1 Tagaplaan

Serveripool kasutab Visual Studio Code PlatformIO raamistikku, mida kasutatakse mikrokontrollerite programmeerimise lihtsustamiseks. PlatformIO nõuetele vastavalt asuvad projekti juurkaustas kataloogid:

- **include** - C++ päisefailide (.h) salvestamiseks;
- **lib** - C++ lisandmoodulite salvestamiseks;
- **src** - projekti põhikoodi salvestamiseks.

Projekti põhikood asub "src/main.cpp" failis. Just seal asub kood, mis initsialiseerib ja käivitab veebiserveri ning sellega seotud päringute töötlemise funktsioonid.

Veebiserverile võivad esitada kahte tüüpi päringuid. Esimene tüüp on lihtsad GET-päringud, mis tehakse aadressidel "{IP}/" ja "{IP}/modules/". Neid kasutatakse rakenduste lehtede ja nendega seotud ressursside failide saamiseks.

Teine tüüp on API-kõned. API-kõned toimuvad aadressil "{IP}/api/" ja võivad olla erinevat tüüpi päringud, nagu GET, POST, DELETE. Edaspidi on rakenduste API-de kirjelduses sellel aadressil baas. Näiteks, **rmDir** API-kõne aadress on "{IP}/api/rmDir".

2.2 Kasutaja osa

Kood, mis puudutab seda, mida kasutaja näeb, on jaotatud kolme kausta vahel:

- **x-frontend-modules** - sisaldab projekti eraldi mooduleid, mis asuvad erinevates kaustades, samuti projekti konfiguratsioonifaili ja faili, mis sisaldab erinevaid kategooriaid, millele konkreetne moodul võib sobida.

- **x-frontend-common** - sisaldab elemente, mida saab kasutada mitmes moodulis, näiteks põhifailid, lehe malli komponendid, JavaScripti fail funktsioonidega API kutsumiseks jne.

- **SD** - sisaldab kaustu "modules" ja "modulesIcons", mis tuleb kopeerida muutmata SD-kaardile, mis on ühendatud mikrokontrolleriga. Nendes kaustades asuvad moodulite töötlusega failid ("modules") ja pildid, mida kasutatakse erinevate moodulite ikoonide kuvamiseks ("modulesIcons").

2.2.1 Projekti konfiguratsioonifail

Projekti konfiguratsioonifail asub kaustas "x-frontend-modules" ja kannab nime "project.config.js".

Failis on eksporditud JavaScript objekt, mis sisaldab moodulite loendit ja teatud projektide kaustade teid. Objektil on järgmised omadused:

- **projektid** - loend, mis sisaldab eraldi mooduleid kirjeldavaid objekte.
(Omadused: 2.2.1.1)

- **modulesDirRelativePath** - suhteline tee iga eraldi projekti kaustast kausta, kus iga eraldi mooduli failid peaksid pärast lähtekoodi töötlemist laaduriga asuma.

- **modulesDirFullPath** - absoluutne tee kausta, kus iga eraldi mooduli failid peaksid pärast lähtekoodi töötlemist laaduriga asuma.

- **base()** - funktsioon, mida kasutatakse iga ressursi tee prefiksi määramiseks Vite laaduriga eksporditud koodis. Prefiks on vajalik, kuna moodulifailid asuvad SD kaardil eraldi kaustades, mitte selle juurkataloogis. Ilma selleta poleks moodulstruktuur

võimalik, kuna laaduri poolt määratud teed tugineksid SD kaardi juurkataloogile, mitte kaustale "modules".

- **mode** - sisse lülitatud moodulite koostamise režiim. Kokku on olemas kaks moodulite koostamise režiimi - "BUILD" ja "SINGLE_MODULE_LOCAL_DEBUG". Esimeses töötleb laadur faile eeldusega, et neid laaditakse SD kaardilt mikrokontrolleri WiFi kaudu, teises režiimis aga eeldusega, et neid laaditakse kaustast, millele viitab modulesDirFullPath. See režiim on vajalik eraldi moodulite kasutajaliidese silumiseks, ilma et kogu koodi SD kaardile täielikult laadimata.

2.2.1.1 Mooduli omadusi kirjeldava objekti struktuur

Objekt, mis kirjeldab iga eraldi moodulit, omab järgmist struktuuri:

- **name** - string, ainulaadne projekti nimi. See peaks kattuma kausta nimega, kus asub selle projekti failid. Näiteks "measurement-example".

- **fullName** - string, projekti nimi, mis kuvatakse kasutajaliideses. Näiteks "Mõõtmise näide".

- **categories** - nimekiri, mis sisaldab ühte või mitut kategooriat, millele see moodul kuulub. Näiteks kasutades "[Categories.SENSOR, Categories.GAME]" kuulub moodul ESP32-le ühendatud andurite kasutamise moodulite kategooriasse ja mänguliste moodulite kategooriasse.

- **roundIconName** - string, ikooni nimi, mis asub kaustas "modulesIcons" ja mida kasutatakse mooduli ringikujulise ikoonina. Näiteks "desktop.png".

- **iconName** - string, ikooni nimi, mis asub kaustas "modulesIcons" ja mida kasutatakse mooduli põhikuvana.

- **hideInDesktop** - boolean väärtus, mis määrab, kas mooduli ikoon kuvatakse töölaua ("desktop") moodulis.

2.2.2 Plugin

Iga eraldi projektile lisatud moodul asub eraldi kaustas "x-frontend-modules" sees.

Moodul võib olla korraldatud kuidas iganes soovitakse, kuid peab järgima kahte reeglit:

1. Iga mooduli laadimine peab olema seadistatud nii, et lõppfailid eksporditakse SD kausta ja kõik viited koodis kasutatud ressurssidele omavad eesliite, mis tagastab base() funktsioon, nagu kirjeldatud "project.config.js".
2. Iga mooduli kaustas peab olema index.html fail. See on ainus fail, mida kuvatakse, kui üritate minna mooduli lehele.

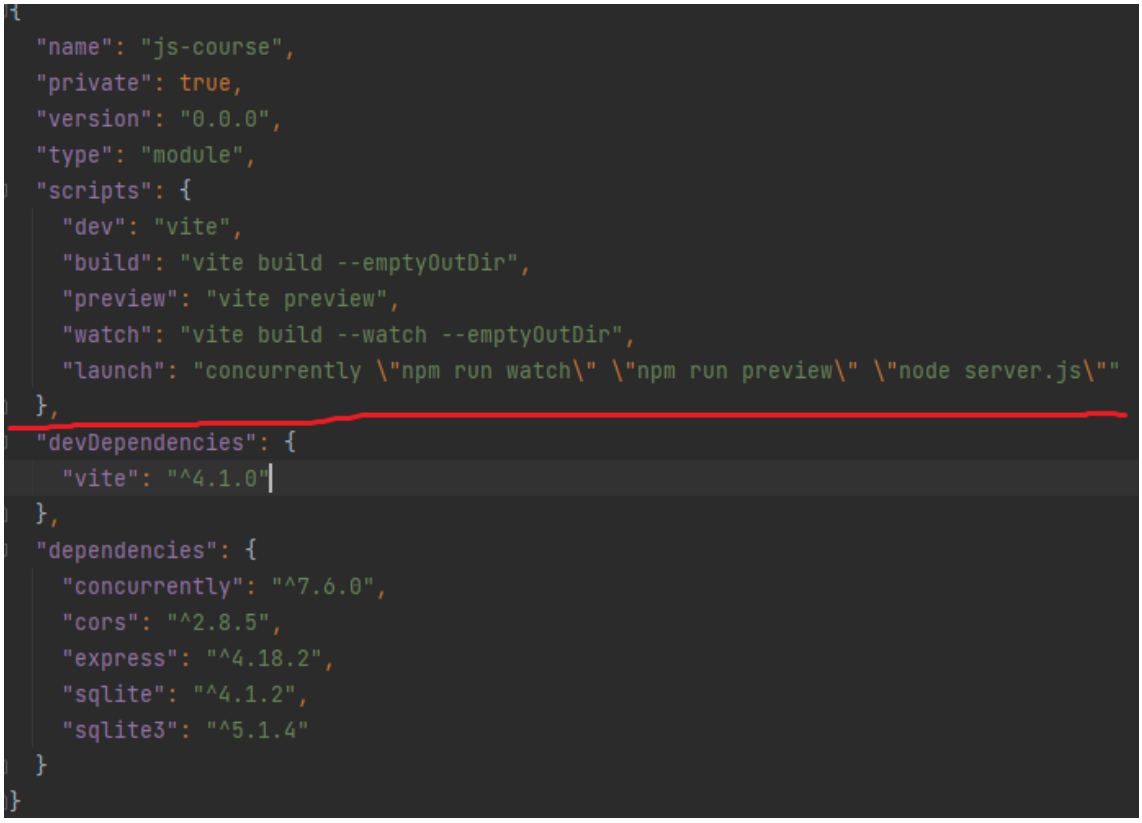
2.2.3 Aja säästmine mitme mooduli kasutamisel koos laaduritega

Tavaliselt tuleb projekti kompileerimiseks kasutada laadurit (näiteks Vite) ja käivitada konsoolis skripti sõlme pakihalduris.

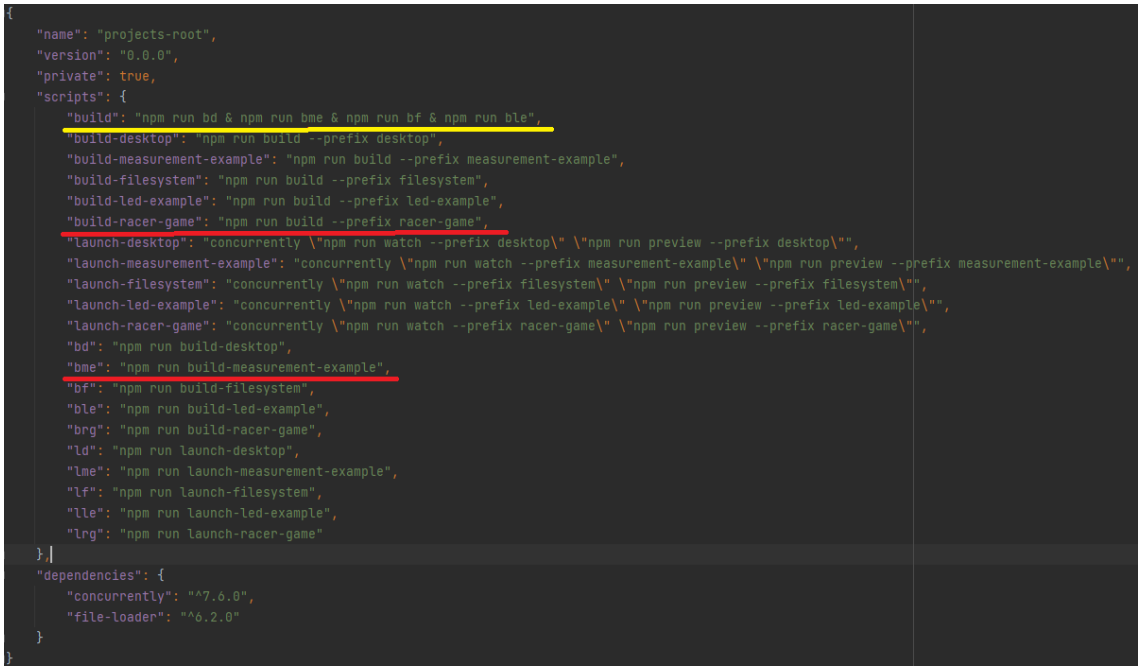
Selle käsu mall on "npm run <skripti nimi>".

Kui teil on üks projekt, saate selle käsu käivitada käsitsi. Kuid mida teha, kui on palju projekte? Selleks, et mitte korrata seda toimingut iga eraldi projekti jaoks, saate üks kord lisada vajaliku käsu "x-frontend-modules/package.json" skripti. Seejärel saate ühe

käsu abil automaatselt koguda kõik projektid, käivitades selle skripti. See näeb välja selline:



Joonis 1. Package.json mooduli kataloogis



Joonis 2. Skript, mis alustab selle mooduli koostamist failist package.json kuni x-frontend-modules (punane) ja skript, mis alustab kõigi moodulite loomist (kollane)

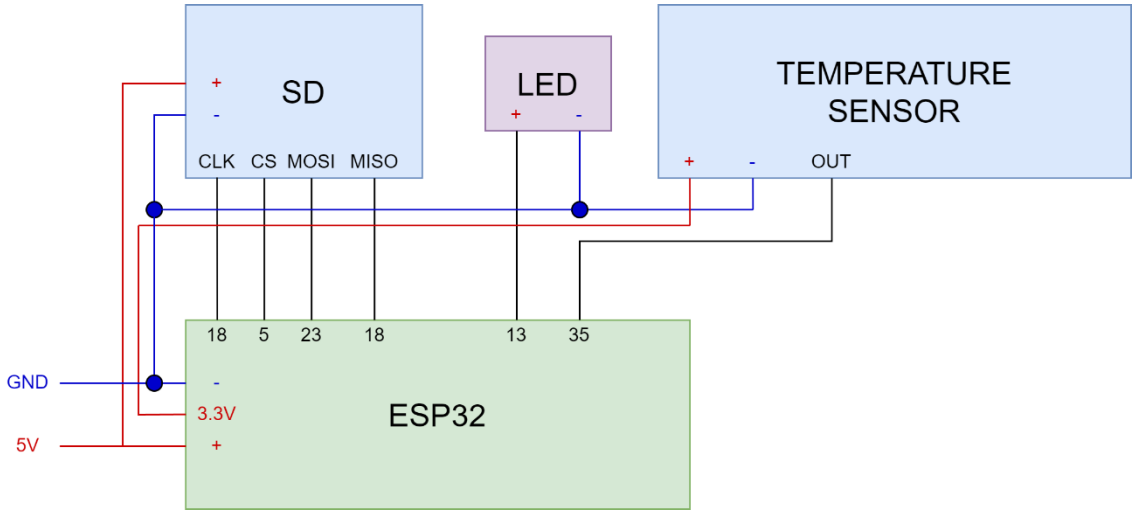
3 Elektroonikaseadmete ühendusskeem

See osa näitab elektrikomponentide omavahelise ühendamise skeemi.

3.1 IXX1530 Arvutisüsteemide projekt

Lõputöö osas kasutatakse järgmisi omavahel ühendamist vajavaid elektroonilisi seadmeid:

1. ESP32
2. SD
3. LED
4. Temperatuur sensor.

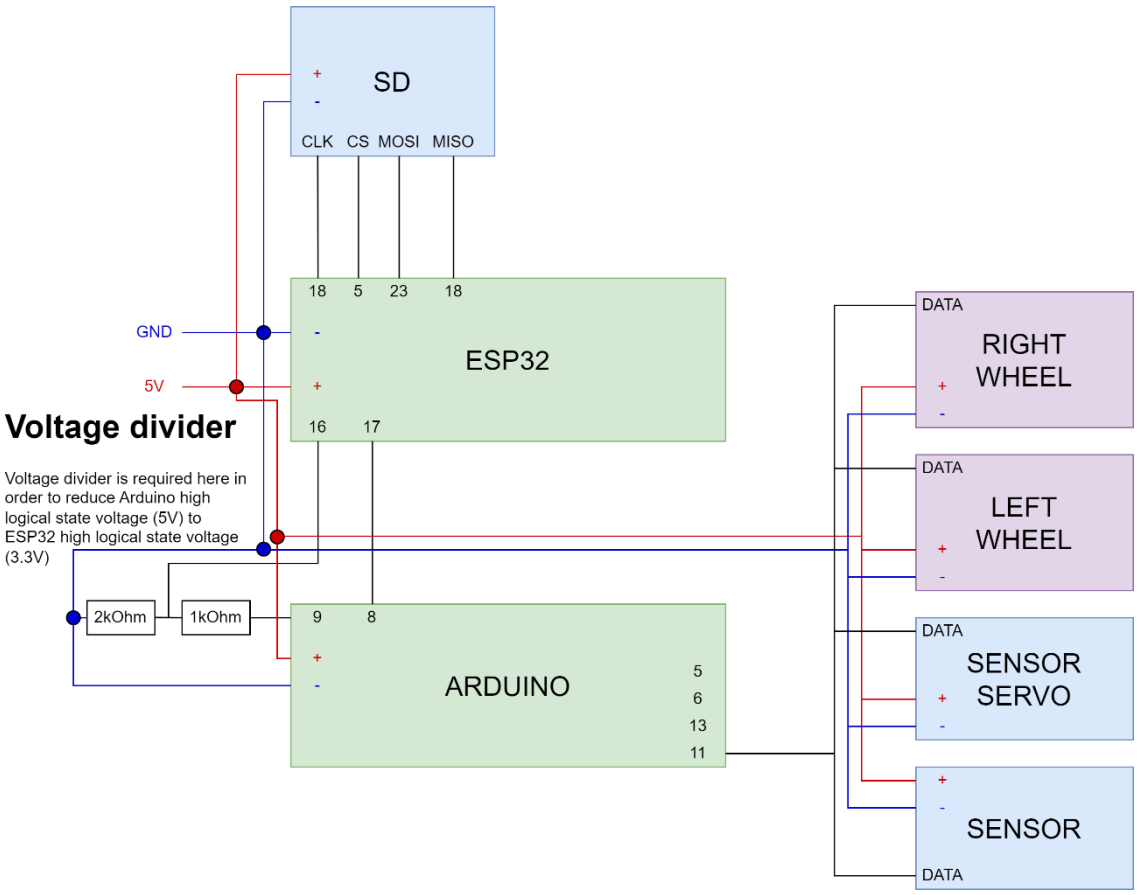


Joonis 3. IXX1530 osa lihtsustatud ühendusskeem.

3.2 Bakaureuse töö osa

Lõputöö osas kasutatakse järgmisi omavahel ühendamist vajavaid elektroonilisi seadmeid:

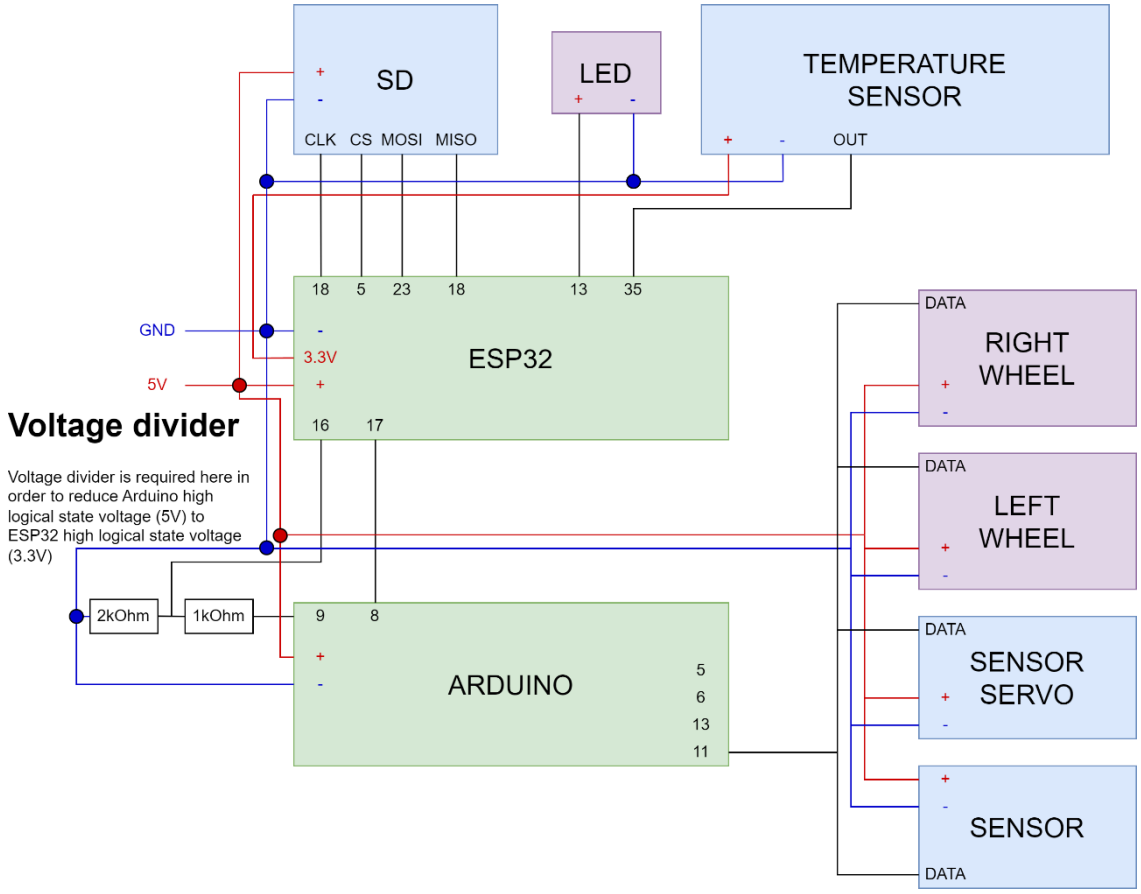
1. ESP32
2. SD
3. Arduino
4. 2x Servo wheel
5. Sensor servo
6. Ultrasonic sensor



Joonis 4. Lõputöö osa lihtsustatud ühendusskeem

3.3 Üldine ühendusskeem

Kui kasutatakse kõiki veebirakendusi ja ühendatakse kõik elektroonilised seadmed, näeb ühendusskeem välja selline:



Joonis 5. Üldine lihtsustatud elektriskeem.

4 Moodulite kirjeldus
(IXX1530 Arvutisüsteemide projekt osa)

See osa kirjeldab IXX1530 raames loodud moodulite struktuuri ja funktsionaalsust. Kokku loodi 5 moodulit:

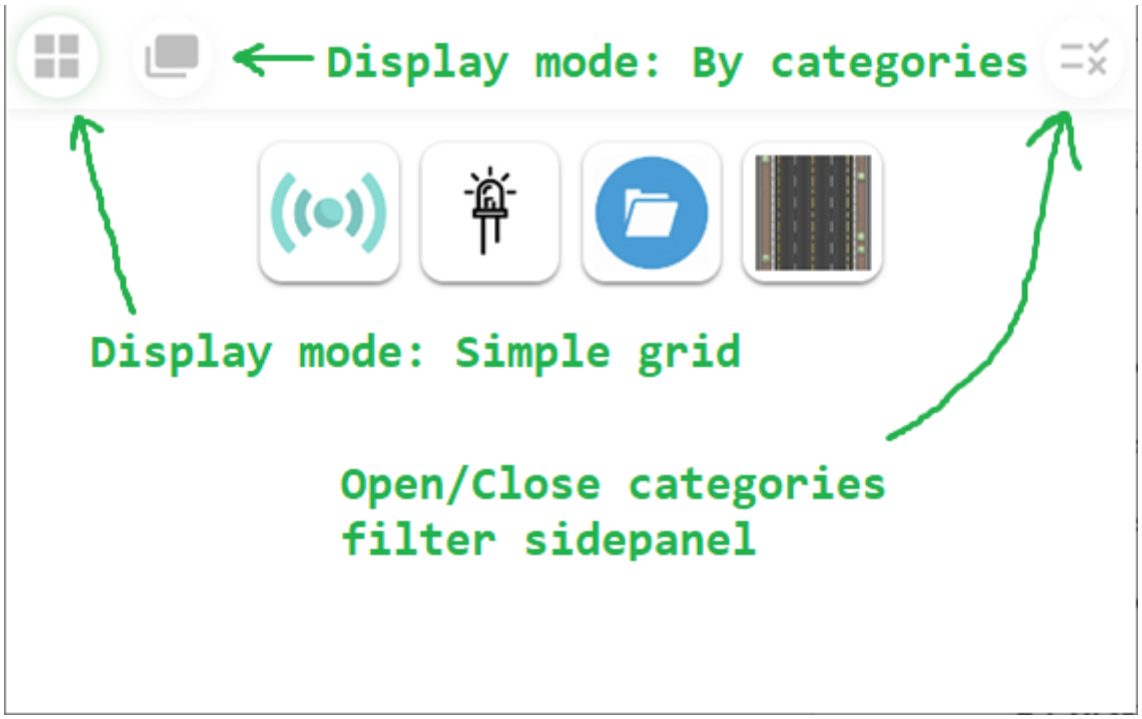
- Desktop app
- Filesystem app
- Led Example app
- Measurement Example app
- Racer Game app

Need moodulid võimaldavad kasutajal järgmist:

1. Näha olemasolevaid veebirakendusi ja käivitada neid mugava graafilise kasutajaliidese abil.
2. Näha SD-kaardil olemasolevaid faile ja kaustu ning nendega suhelda.
3. Suhelda väliskeskkonnaga mikrokontrolleriga ühendatud seadmete, näiteks LED-ide abil.
4. Andmete lugemine väliskeskkonnast temperatuurianduri abil ning nende andmete kuvamine graafikuna.
5. Mängida JavaScriptis kirjutatud mänge.

4.1 Desktop app

Antud rakendus on mõeldud kasutamiseks vahekaardi paneelina kõigile ühendatud moodulitele. Selle abil saate näha nende loendit kahe erineva valikuna ja filtreerida rakendusi nende kategooriate järgi, millele need kuuluvad.



Joonis 6. Rakenduse ülevaatepaneel, lihtne kuvarežiim.

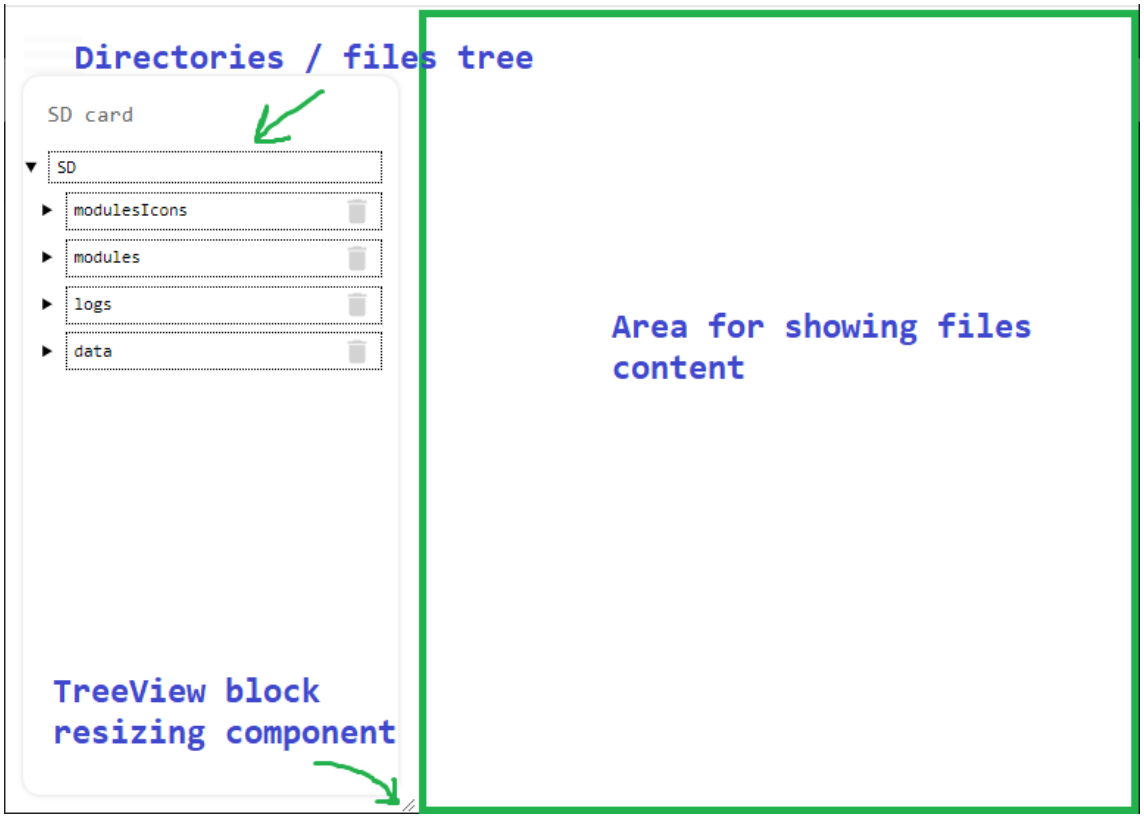
Kursori liigutamisel rakenduse ikooni kohal kuvatakse selle nimi ja klõpsamisel suunatakse kasutaja selle rakenduse lehele.

4.2 Filesystem app

Antud rakendus on mõeldud kataloogide ja failide vaatamiseks, mis on salvestatud SD-kaardile, mis on ühendatud ESP32-ga. Siin saab faile vaadata, kustutada ja alla laadida.

4.2.1 Üldine struktuur

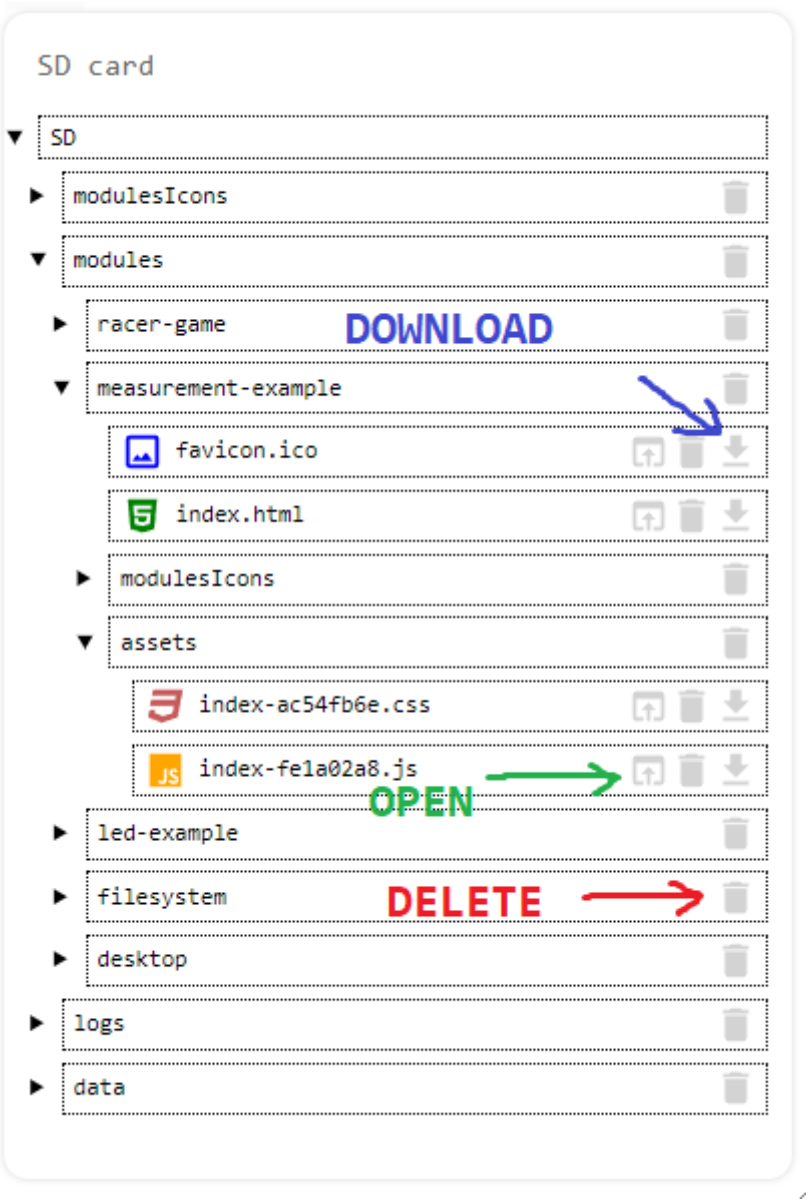
Kokkuvõtlikult koosneb leht kahest põhiosast - kaustade/failide puu nimekirjast ja alast, kus kuvatakse failide sisu. Lisaks on kaustade/failide puud sisaldaval blokis kasutajal võimalik ise muuta bloki laiust.



Joonis 7. Failisüsteemi rakenudse lehe üldvaade ja struktuur.

4.2.2 Puuvaade

Puuvaatel loetletud esemete kaheks põhiliigiks on kaustad ja failid.



Joonis 8. Puuvaates kuvatavad kataloogid ja failid.

Kataloogid saab avada, sulgeda või kustutada. Kataloogid kuvatakse kolmnurgaga vasakul, mis näitab nende praegust olekut. Kui kolmnurk on külgsuunas, tähendab see, et kataloog on kokkutõmmatud. Kui kolmnurk osutab alla, tähendab see, et kataloog on avatud ja näitab selles sisalduvaid faile ja alamkaustu.

Faile saab kustutada, alla laadida või avada paremal pool puuvaates olevas piirkonnas. Failid kuvatakse ikooniga, mis näitab nende tüüpi, näiteks tekstifail, js / css koodifail, html, pilt jne. Kokku on enam kui 20 erinevat ikooni erinevate failitüüpide kuvamiseks. Ikonni ja selle värvi, mida tuleks kuvada, määratakse automaatselt iga eraldi komponendi põhjal faili laienduse alusel.

4.2.3 Failide avamine vaatamiseks

Faili vaatamiseks tuleb klõpsata ikoonil, millel on üles nooltega aken.

Joonis 24. Kuidas failide avamise ikoon välja näeb ja kus see asub.

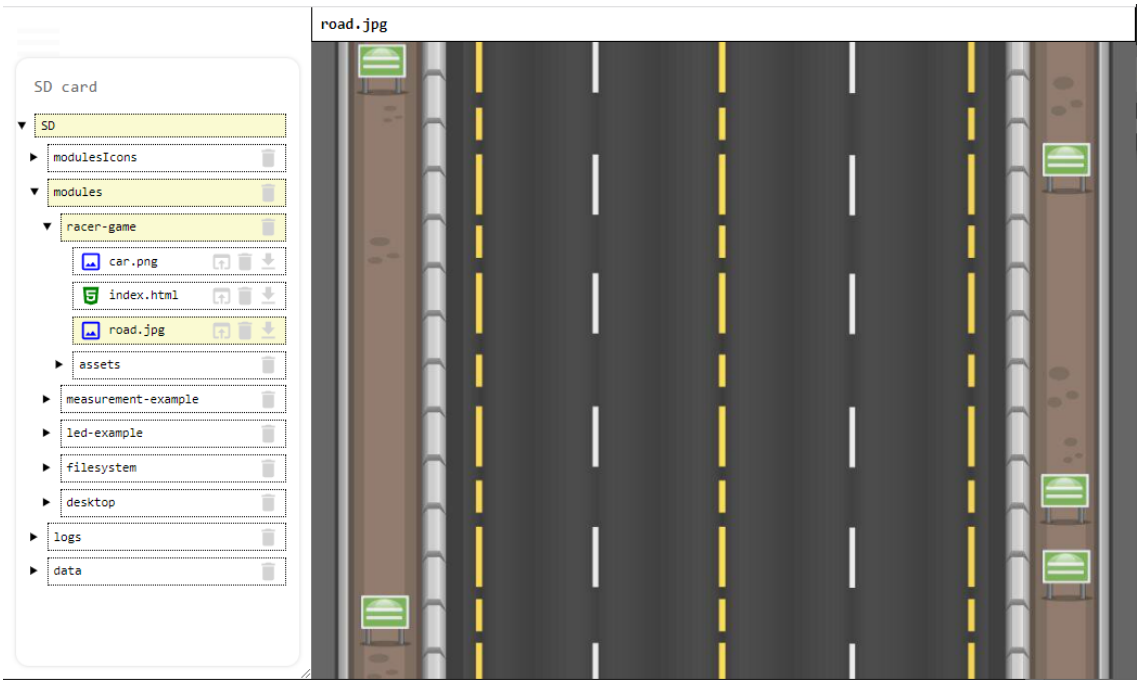
Vajutades nuppu avaneb dialoogaken, kus saab valida faili avamise režiimi:

- **TEXT** - Fail avatakse tavalise tekstina.

- **IMAGE** - Fail avatakse pildina.

JOONIS 25. FAILI AVAMISE DIALOOGIBOKS.

Joonis 9. Fail avatud tekstivormingus.



Joonis 10. Pildina avatud fail.

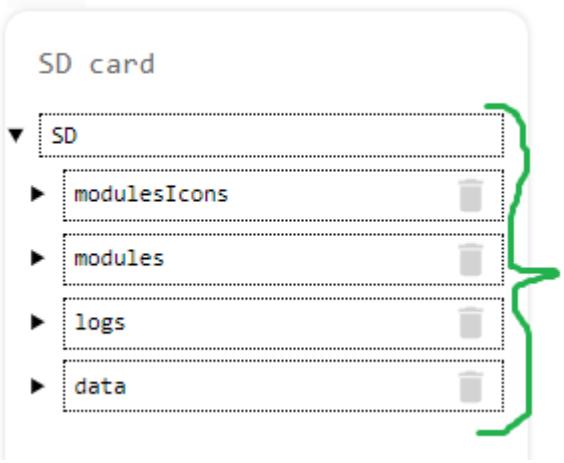
Kui fail on avatud, siis selle sisu kuvatakse parempoolses alaosas kaustapuus. Lisaks on fail ja kõik tema eelnevad kaustad esile tõstetud, et meeles pidada, milline fail on avatud kaustade avamisel / sulgemisel.

4.2.4 Puulisti elementide laadimise mõned omadused

ESP32 ja SD-kaardi suhtlus on võrreldes arvutiga ühendatud kõvakettaga üsna aeglane. Lisaks raskendab andmete edastamine WiFi kaudu protsessi veelgi. Nende põhjuste tõttu oleks väga ebamugav ja aeganõudev laadida kogu kõvaketta salvestatud kaustade/failide täielik struktuur korraga lehe laadimisel, kuna see võtaks märkimisväärse aja (üle viie-seitsme sekundi eeldusel, et SD-kaardil on ainult need rakendused, mis olid osa lõputööst). See on vastuvõetamatu.

Selliste viivituste vältimiseks, mis tekitavad kasutajale ebamugavust, kasutati teist süsteemi. Selle asemel, et laadida kogu kaustade/failide puu korraga, laaditakse seda osade kaupa. Mida see praktikas tähendab?

Lehe laadimisel laaditakse ainult juurkaust "SD" ja selle alamkaustad ning failid kuni üks tase allapoole.



Joonis 11. Leheküljel navigeerimisel laaditud kataloogid ja failid.

Seega edastatakse ainult teave SD-kaardi kataloogi ja selle alamkataloogide ning osaliselt ka vana-alade kohta, mis aitab aega säästa. Kui puuvaate kataloogi avamisel kontrollitakse, kas vana-alade teave on laaditud, ja kui seda pole, laaditakse see eraldi päringus. Kataloog ise muutub sel ajal passiivseks, et vältida veel laadimata alamkataloogide avamist.

JOONIS 26. PUUVAATE ELEMENDID, MILLE KOHTA LAADITAKSE PRAEGU TEAVET NENDE ALAMKATALOOGIDE JA FAILIDE KOHTA.

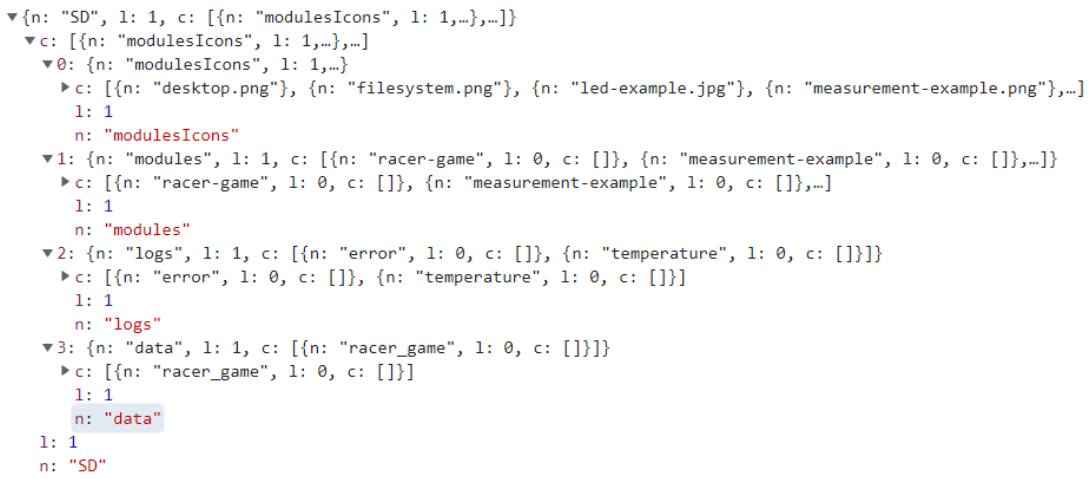
4.2.5 API, mida kasutatakse SD-kaardil olevate failidega suhtlemiseks

Kokku kasutab failisüsteemi moodul nelja API-punkti:

- **dirInfo**
- **download**
- **rmDir**
- **rmFile**

4.2.5.1 dirInfo

Antud API punkti kasutatakse alamkaustade kohta teabe saamiseks. API sisendiks on kausta tee, millele järgneb JSON objekt, mis kirjeldab antud kausta ja kõiki selle lapsi ja lapselapsi (lapselapsed ei ole täielikud ja vajavad täiendavat teabe laadimist, kuna neil ei ole teavet oma järglaste kohta).



Joonis 12. JSON objekt, mis kirjeldab SD-kataloogi (SD-kaardi juurkataloog) ja sisaldab teavet selle laste kohta.

Iga kataloog ja fail edastatakse objektina, millel on üks kohustuslik võti **n**, mille väärtus on faili või kataloogi nimi, ja kahe võtmega, mida kasutatakse ainult kataloogide puhul - **l** ja **c**.

Võtme c väärtus on objektide loend, mis asuvad kataloogis.

Võti **l** annab teada, kas edastatud info kataloogi alamkataloogide kohta vastab tegelikkusele või vajab täpsema info saamiseks täiendavat API kõnet.

4.2.5.2 download

Seda API-punkti kasutatakse SD-kaardile salvestatud failide allalaadimiseks.

4.2.5.3 rmDir

Seda API-t kasutatakse kataloogide eemaldamiseks SD-kaardilt.

4.2.5.4 rmFile

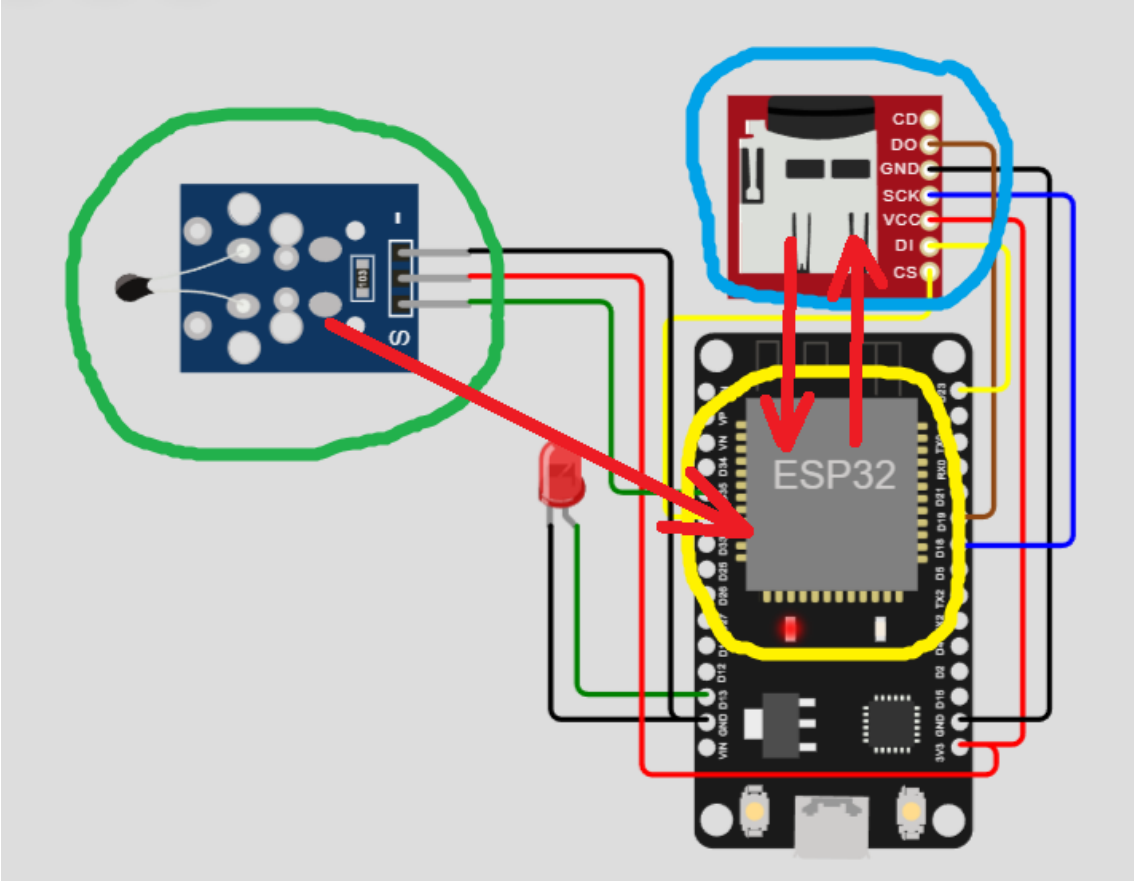
Seda API-t kasutatakse failide kustutamiseks SD-kaardilt.

4.3 Measurement Example app

Antud rakendus on näide süsteemi kasutamisest millegi mõõtmiseks andurite abil ning seejärel nende andmete kasutamisest graafiku koostamiseks.

4.3.1 Tagaosa: kuidas see töötab

Veebiserveri poolt lugestab mikrokontroller ESP32 temperatuuriandurilt väärtuse ajastatult ja salvestab selle SD-kaardile. Seejärel saab API abil need andmed lugeda ja saata kasutaja seadmele.



Joonis 13. Andmete liikumine süsteemis. Temperatuuriandur on tähistatud rohelisega, ESP32 on kollase värviga, SD-kaart on sinisega.

4.3.2 Andmete püsivus ja API

Salvestatud andmed salvestatakse tavalistes tekstifailides kaustas "SD / logs / temperatuur" formaadis "<tund>:<minut>:<sekund> <temperatuur>". Iga fail salvestab endasse ühe päeva andmed. Faili nimi on formaadis "<aasta><kuu><päev>.txt".

JOONIS 27. SALVESTATUD ANDMETE NÄIDE.

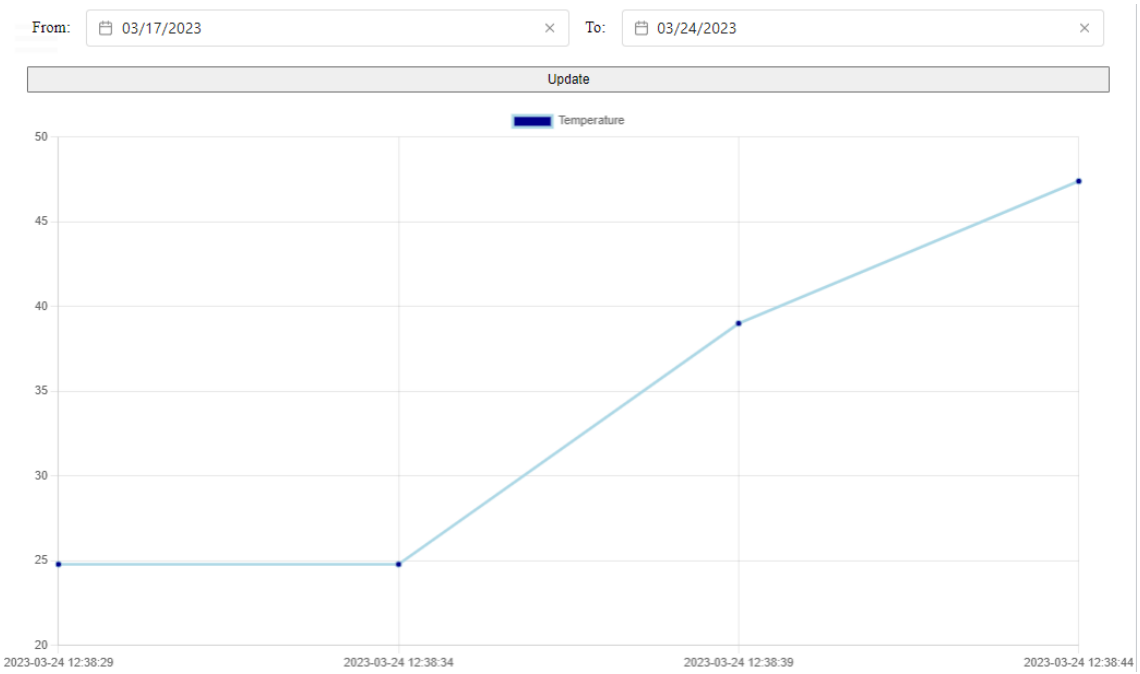
API **sensorData** punkti kaudu saab nimekirja sellistest failidest, mis sisaldavad teavet temperatuuri kohta erinevatel päevadel. Kui päring edastatakse ilma täiendavate parameetriteta, tagastatakse JSON nimekirjaga, mis sisaldab kõigi failide nimesid.

Kui lisaks saadetakse kuus argumenti, mis kirjeldavad kahe kuupäeva vahelist perioodi (yfrom, mfrom, dfrom, yto, mto, dto), siis tagastatakse ainult need failid, mis jäävad nende kahe kuupäeva vahele.

Kasutaja poolt saadud selle failide nimekirja korral teeb rakendus mitu päringut, saades igalt faililt sisu, pärast mida teisendab teksti graafikukomponendi töötlemiseks vajalikule kujule.

4.3.3 Kasutaja osa

Kasutajaliides rakenduses on väga lihtne. Ekraanil on ainult mõned elemendid - graafik, kaks kuupäeva valimise komponenti ja värskendamisnupp.



Joonis 14. Rakenduse leht.

Lehekülje laadimisel kuvatakse kõik viimase nädala andmed. Kuupäeva valimise komponentide kasutamisega saab reguleerida perioodi, millal temperatuuriandmed kuvatakse.

4.4 LED Example

Antud rakendust kasutatakse lihtsa näitena sellest, kuidas veebiserveri API kasutades saab suhelda sellega ühendatud elektriseadmetega ja mõjutada välist keskkonda.

JOONIS 28. RAKENDUSE LEHT, KUS LED ON VÄLJA LÜLITATUD.

"TOGGLE" nupp on rakenduse lehel ainus interaktiivne element. Seda vajutades saab sisse/välja lülitada LED-i, mis on ühendatud mikrokontrolleriga.

JOONIS 29. RAKENDUSE LEHT PÕLEVA LED-IGA.

Selle rakenduse töötamiseks kasutatakse kahte API-punkti:

- **ledState** - praeguse LED-i oleku saamiseks.

- **setLedState/{state}** - LED-i oleku seadmiseks. Selle asemel {state} edastatakse 0 või 1, mis tähendab LED-i sisse lülitamist (1) või väljalülitamist (0).

4.5 Racer Game app

Antud rakendus on ülaltvaates olev mäng, kus mängija juhib autot ja eesmärk on sõita võimalikult kaua, kokkupõrkeid teiste autodega vältides. Mäng muutub pidevalt raskemaks, suurendades sõiduki kiirust.

Selle rakenduse eesmärk on näidata praktikas, et projektis on võimalik kasutada mooduleid, mis on kirjutatud mis tahes JavaScript/TypeScript raamistikus või puhast JavaScript/TypeScript-is. Seda mängu kirjutati puhast JavaScript-i kasutades loodud mikro raamistikku Reactive, mis tagab kaasaegsete raamistike põhifunktsionaalsuse, nagu:

- Komponendi loomine mallist.

- Võimalus salvestada mis tahes objekte komponendi sisemisse salvestusse hilisemaks kasutamiseks.

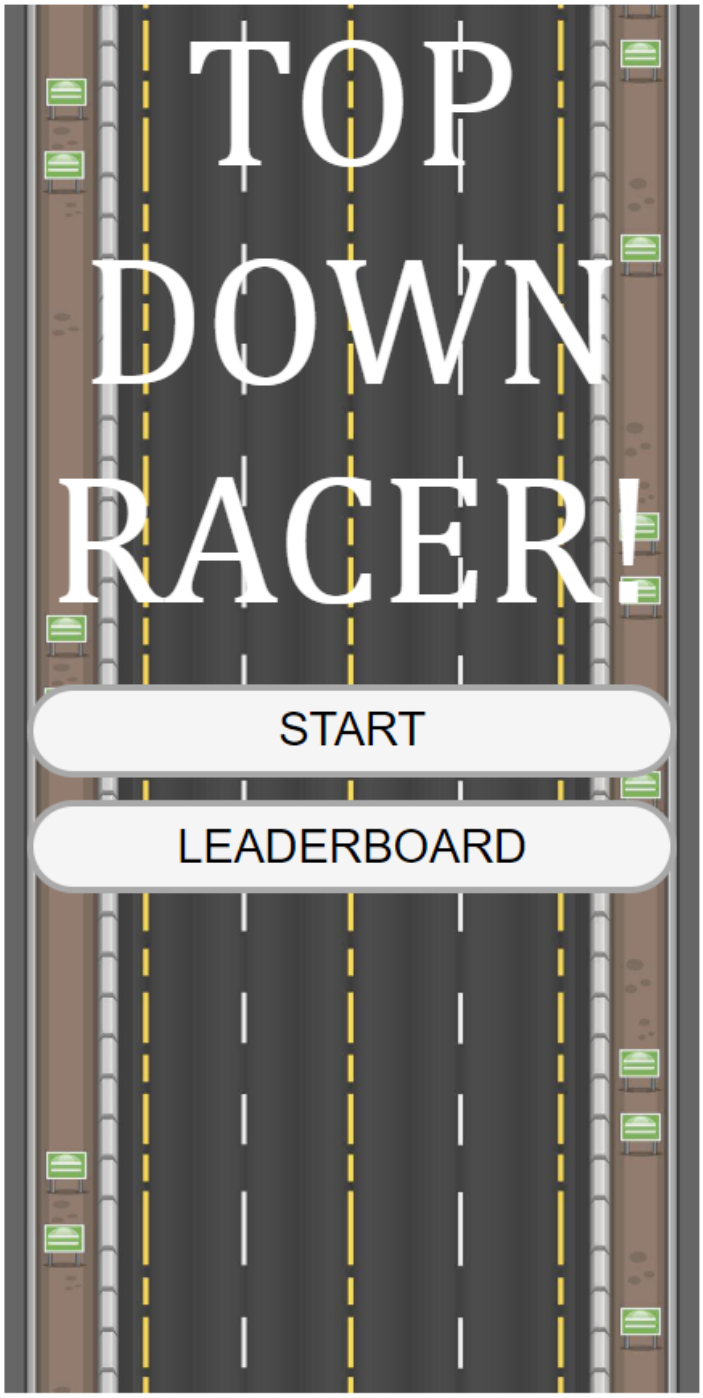
- Võimalus siduda komponendi sisemise salvestuse väärtusi HTML elementide omadustega.

- Arvutatavad omadused.

- Kohalikud CSS stiilid.

4.5.1 Mängu protsess

Käivitades rakendust, näeb mängija menüüd. Menüüs on kaks nuppu - "START" ja "LEADERBOARD". Esimene nupp käivitab mängu, teine avab tabeli mängijate skooridega.



Joonis 15. Mängu menüü.

JOONIS 30. TEE MÄNGIJAAUTOGA (PUNANE) JA VÄLDITAVATE AUTODEGA (KOLLANE).

Ülemises vasakus nurgas ekraanil on näha hetkeseisuga skoor ja paremal on nupp "X", millele vajutades saab lõpetada praeguse mängu.

Mängu ajal saab mängija oma autot juhtida klaviatuuri nooltega ning mängu pausile panna vajutades tühikuklahvi.

JOONIS 31. PAUS MÄNGUS.

Kui mängija põrkab vastu kellegi teise autot, siis mäng lõpeb.

JOONIS 32. MÄNG ON KAOTATUD.

Vajutades tühikuklahvi, saate minna menüüsse. Lisaks saate sisestada mängija nime sisendiväljale ja vajutada "SUBMIT". Seejärel salvestatakse mängija nimi ja tema skoor ning neid saab näha juhtide tabelis.

Menüüs "LEADERBOARD" saate avada tabeli salvestatud rekorditega.

JOONIS 33. REKORDITE TABEL.

4.5.2 API ja kirjete salvestamine

Mängijate kirjed salvestatakse tekstifaili "SD/data/racer_game/scores.txt". Nende kättesaamiseks kasutatakse kahte API-punkti - "scores" ja "addNewScore".

- **addNewScore** kasutatakse uue mängija-skoori paari lisamiseks tekstifaili.
- **scores** kasutatakse faili "SD/data/racer_game/scores.txt" saamiseks. Tabeli avamisel saab kasutaja seade faili sisu ja töötleb seda, muutes selle tekstivormingust objektide loendiks, mis on järjestatud punktide arvu järgi.

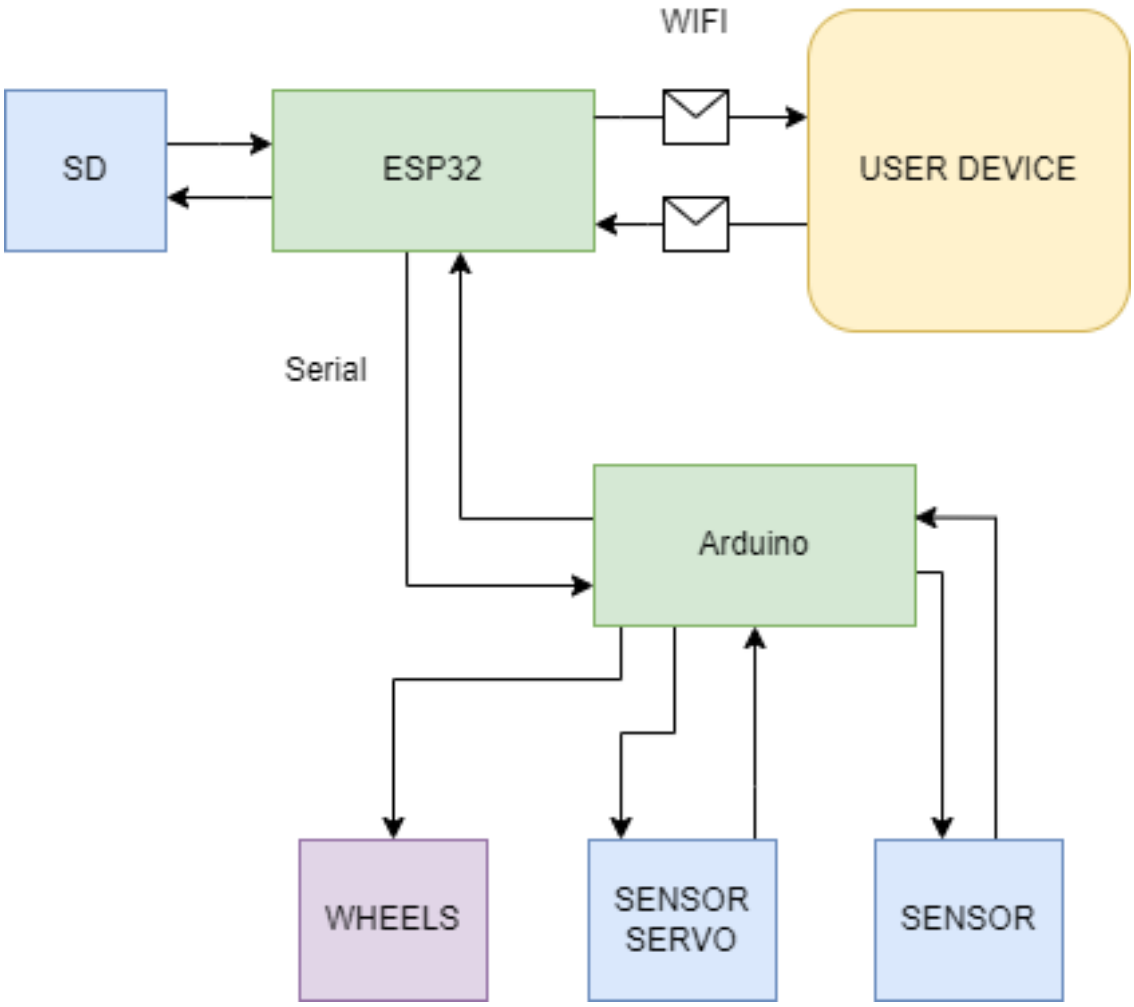
5 Robot (Bakalaureuse töö osa)

See osa kirjeldab lõputöö raames loodud roboti juhtimise rakenduse struktuuri ja funktsionaalsust, mis on IXX1530 projekti kohal olev pealeehitus.

Käesoleva töö raames:

1. Kirjutati Arduino mikrokontrolleri kood, mis juhib kahe rattaga robotit ning ultraheliandurit.
2. Täiendati ESP32 koodi, et ESP32 saaks järjestikpordi abil Arduino mikrokontrollerilt käskusid vastu võtta / saata.
3. Loodi eraldi veebirakendus, mis võimaldab juhtida roboti tegevust ning saada ümbritseva keskkonna teavet ultraheli kaugusmõõtja abil.

5.1 Üldine struktuur



Joonis 16. Roboti seadmete üldine skeem.

Töö skeem on järgmine:

- ESP32 kasutatakse veebiserveri majutamiseks, teabe saamiseks / kirjutamiseks SD-kaardile ning käskude saamiseks / saatmiseks Arduino'le.
- Arduino juhib ESP32 käskude alusel roboti ratast, mis asuvad külgedel, samuti servomootorit, millele on paigaldatud kaugusandur, ja kaugusandurit ennast. Ümbritseva keskkonna kohta kogutud teabe saadab Arduino ESP32-le.

5.2 Andmete edastamine Arduino ja ESP32 vahel

Andmete edastamine toimub järjestikpordi abil kiirusega 72800 bitti/s. Kuna mikrokontrollerite kõrged loogilised tasemed on erinevad (ESP32 puhul on see 3,3 V, Arduino puhul 5 V), ühendatakse Arduino TX-pistik ESP32 RX-pistikuga pingejaoturi abil, mis vähendab pinget 5-lt 3,3 volti. ESP32 TX-pistik ühendatakse Arduino RX-pistikuga otse.

5.2.1 Madala pingega probleem

Mikrokontrollerite põhiste RX- ja TX-pistikute kasutamine andmete edastamiseks ei õnnestunud, kuna Arduino RX-pistikule antud madala pinge tõttu andis mikrokontroller pidevalt veateate.

ESP32 ja Arduino vahelise kahepoolse andmeedastuse jaoks kasutati tarkvaralist järjestikpordi abil raamatukogu „SoftwareSerial“. Selle raamatukogu kasutamisel töötas kõik suhteliselt normaalselt.

5.2.2 Andmeside optimeerimine

Programmeerimise lihtsustamiseks ja koodi paremaks loetavuseks loodi spetsiaalne klass „RobotCMDSerial“, mis vastutab lihtsa ja usaldusväärse teabe edastamise eest järjestikuse (serial) pordi kaudu.

Kõigi andmete edastamine toimub käskude abil. Iga käsk koosneb kohustuslikust võtmest (käsu nimest) ja käsu sisust.

Võimalik on edastada:

A) Kehata käsk kujul: <käsk>;
Näide: „STOP;”

B) Kehaga käsk stringi kujul: <käsk>:<keha>;
Näide: „setDirection:IDLE;”

C) Kehaga käsk täisarvude massiivi kujul, edastatuna stringina:
<käsk>:<massiivi pikkus>:<komadega eraldatud arvude massiiv>;
Näide: „status:6:0,1,2,3,4,5;”

Saadud käskude lugemine ja töötlemine toimub tsüklis kahes etapis:

1. Saadud andmete lugemine tarkvaralise järjestikpordi (SoftwareSerial) puhvrist RobotCMDSerial klassi eksemplari puhvrisse.
2. Salvestatud saadud käskude hankimine ja nende käskudele vastavate toimingute teostamine.

Näide:

```
void loop() {
  temperatureLogger->Measure();
  cmdSerial->Receive();                                          1
  while(cmdSerial->GetLastCommandKey() != "") {                  2
    String key = cmdSerial->GetLastCommandKey();
    String body = cmdSerial->GetLastCommandBody();               3

    if (key == "directionConfirmed" && body == currentDirection) {   4
      directionConfirmed = true;
    } else if (key == "sensor") {                                5
      saveSensorData(body);
    } else if (key == "unknownCommand") {
      currentDirection = "IDLE";                                 6
      directionConfirmed = false;
    }
    cmdSerial->NextCommand();
  }
  if (!directionConfirmed && millis() - 100 > lastTimeDirectionUpdated) {
    lastTimeDirectionUpdated = millis();
    cmdSerial->Send("setDirection", currentDirection);
  } else if (!directionConfirmed && millis() < lastTimeDirectionUpdated) {
    lastTimeDirectionUpdated = millis();
  }
}
```

Joonis 17. RobotCMDSerial kasutamine

1. Andmete saamine järjestikpordist.
2. Tsükkel, mille käigus toimub kõigi saadud käskude järjestikune töötlemine.
3. Praeguse töödeldava käsu võtme ja keha hankimine.
4. Toiming 1.
5. Toiming 2.
6. Toiming 3.

6.3 Veebirakendus ning roboti juhtimise funktsionaalsus

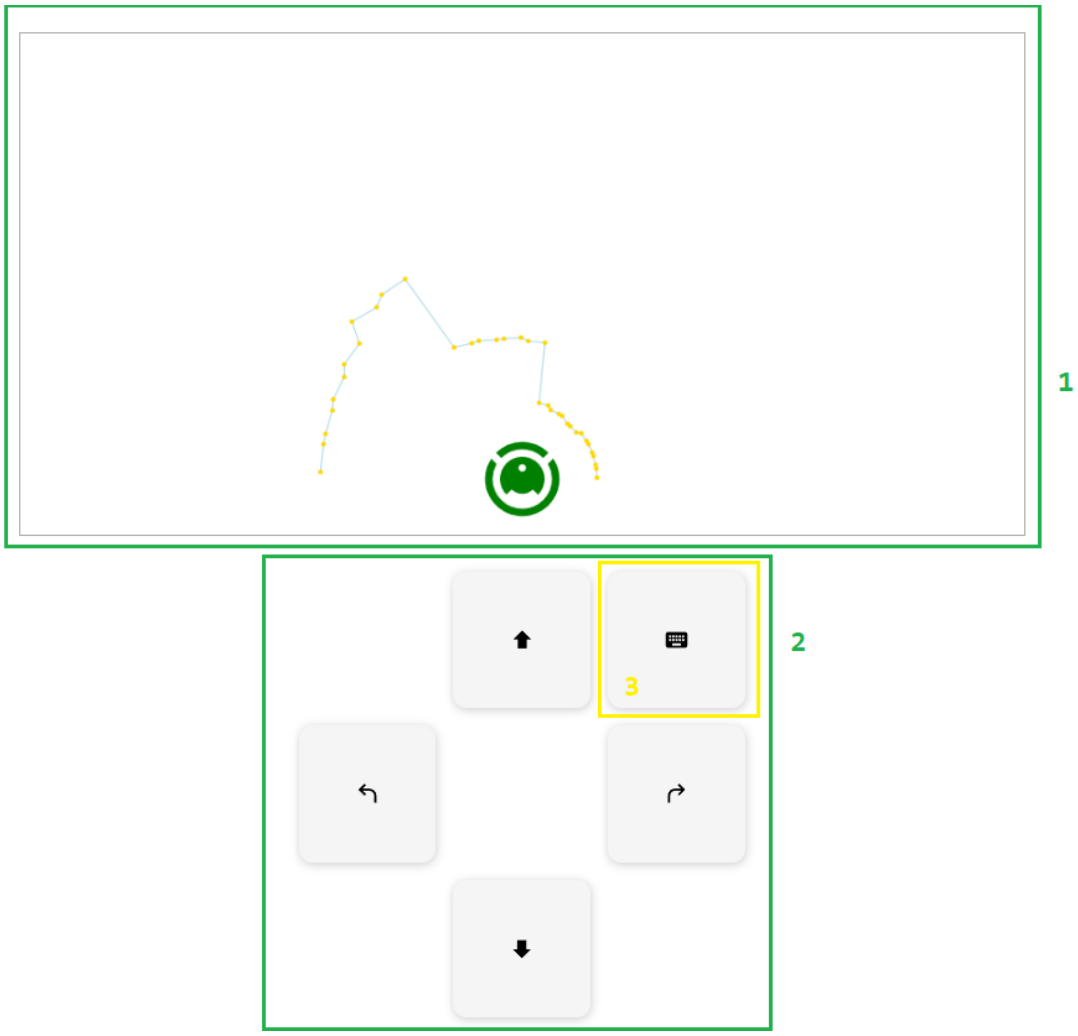
Veebirakenduse lehel, mis vastutab roboti juhtimise eest, on kaks põhielementi.

Esimene element on ekraan, kus saab näha ultraheliandurи näitu. Ekraan näitab kaugust robotist teiste objektideni 180-kraadise ala piires.

Teine element on klaviatuur. Klaviatuuri abil saab robotit juhtida, näidates talle, millises suunas liikuda. Klaviatuuri saab kasutada hiire abil või lülitades sisse klaviatuuri haarde režiimi nupu abil, mis asub paremas ülanurgas (3), kasutades füüsilisel klaviatuuril nooleklahve.

Eelkõige saab robotile anda järgmisi käske:

- Edasi
- Tagasi
- Pöörake vasakule
- Pöörake paremale



Joonis 18. Roboti kuvatavad veebilehe ja keskkonna andmed.

Joonis 19. Roboti tegelik keskkond.

6 Kokkuvõte

Kokkuvõttes tehtud töö ja saavutatud tulemuste põhjal võib kindlalt öelda, et kõik töö käigus seatud ülesanded on edukalt lahendatud ja töö eesmärk on saavutatud.

Eelkõige loodi ESP32-põhine veebiserver:

- Veebiserver toetab mitmeid erinevaid rakendusi.
- Loodi töölauarakendus, kus saab näha kõiki olemasolevaid rakendusi. (**DESKTOP APP**)
- Loodi rakendus, mis võimaldab vaadata, kustutada, avada ja alla laadida SD-kaardil olevaid faile. (**FILESYSTEM APP**)
- Loodi rakendus, mis demonstreerib teoreetilist võimet suhelda väliskeskkonnaga. (**LED EXAMPLE**)
- Loodi rakendus, mis demonstreerib teoreetilist võimet lugeda andmeid väliskeskkonnast. (**MEASUREMENT EXAMPLE APP**)
- Loodi rakendus, mis demonstreerib võimalust kasutada veebirakenduste arendamiseks mitte ainult Vue'd, vaid üldse mistahes JavaScripti raamistikku või puhast JavaScripti. See rakendus töötab nullist loodud mikro-raamistiku Reactive abil. (**RACER GAME APP**)
- Loodi keeruline rakendus, milles toimub üheaegne andmete lugemine väliskeskkonnast ja suhtlemine väliskeskkonnaga ning andmete edastamine kahe eraldi mikrokontrolleri vahel. (**ROBOT (BAKALAUREUSE TÖÖ OSA)**)
- Lihtne uute veebirakenduste lisamine. (**ÜLEVAADE KOODISTRUKTUURIST PROJEKTIS**)

Kasutatud kirjandus

JavaScripti teave:
https://learn.javascript.ru/ (29.04.2023)
https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide (29.04.2023)

Vue teave:
https://vuejs.org/ (29.04.2023)

C++ teave:
https://ravesli.com/uroki-cpp/ (29.04.2023)

99% kõigist koodiprobleemidest lahendamine:
https://stackoverflow.com/ (29.04.2023)

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Ilja Kuznetsov

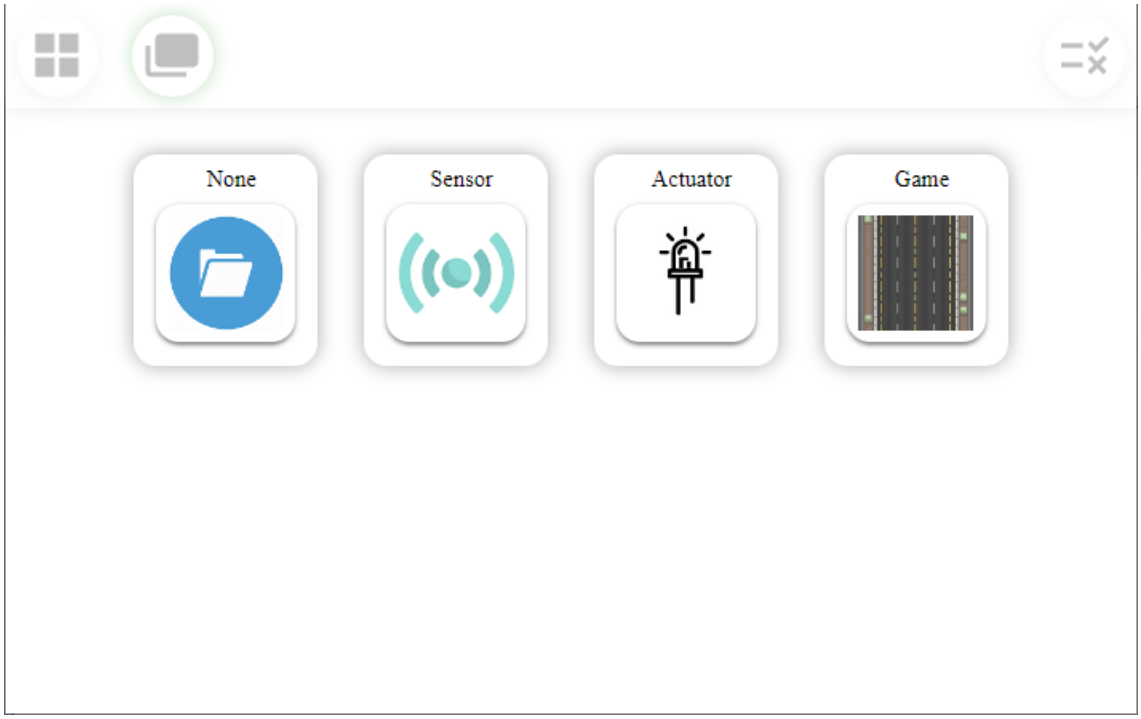
1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "ESP32 veebiserver", mille juhendaja on Vladimir Viies
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

---

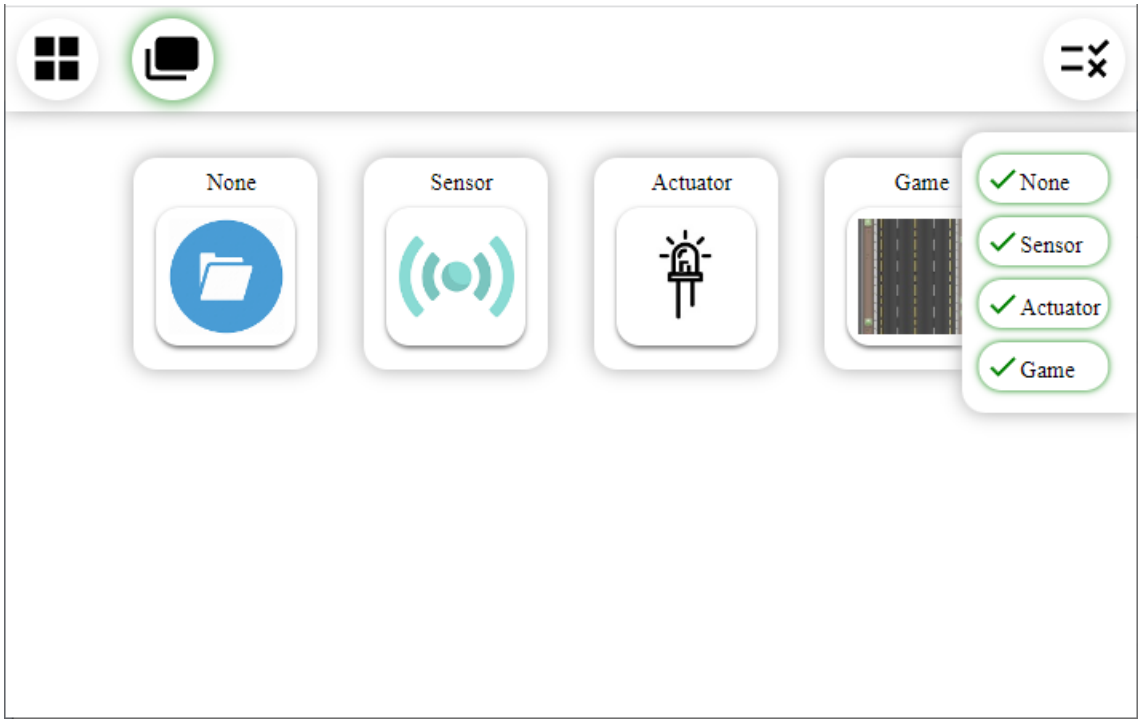
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

29.04.2023

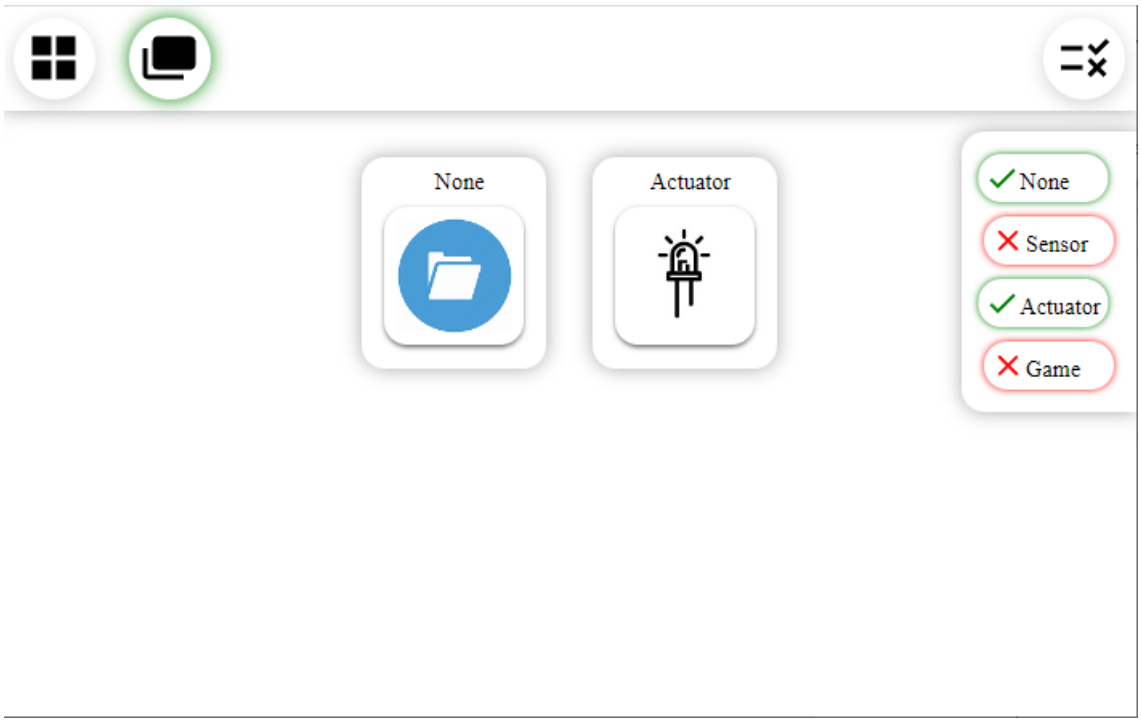
Lisa 2 – Ekraanipildid



Joonis 20. Rakenduse ülevaate paneel, kategooria kuvamise režiim.

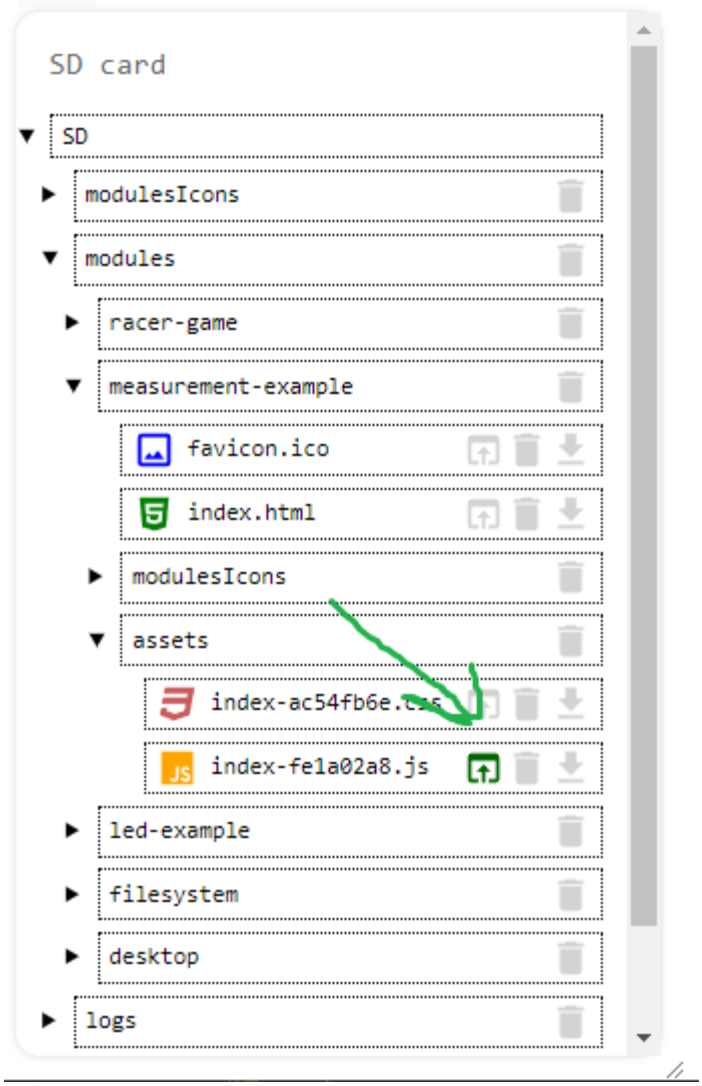


Joonis 21. Rakenduste ülevaatepaneel koos filtri külgribaga, kõik filtrid kaasas.



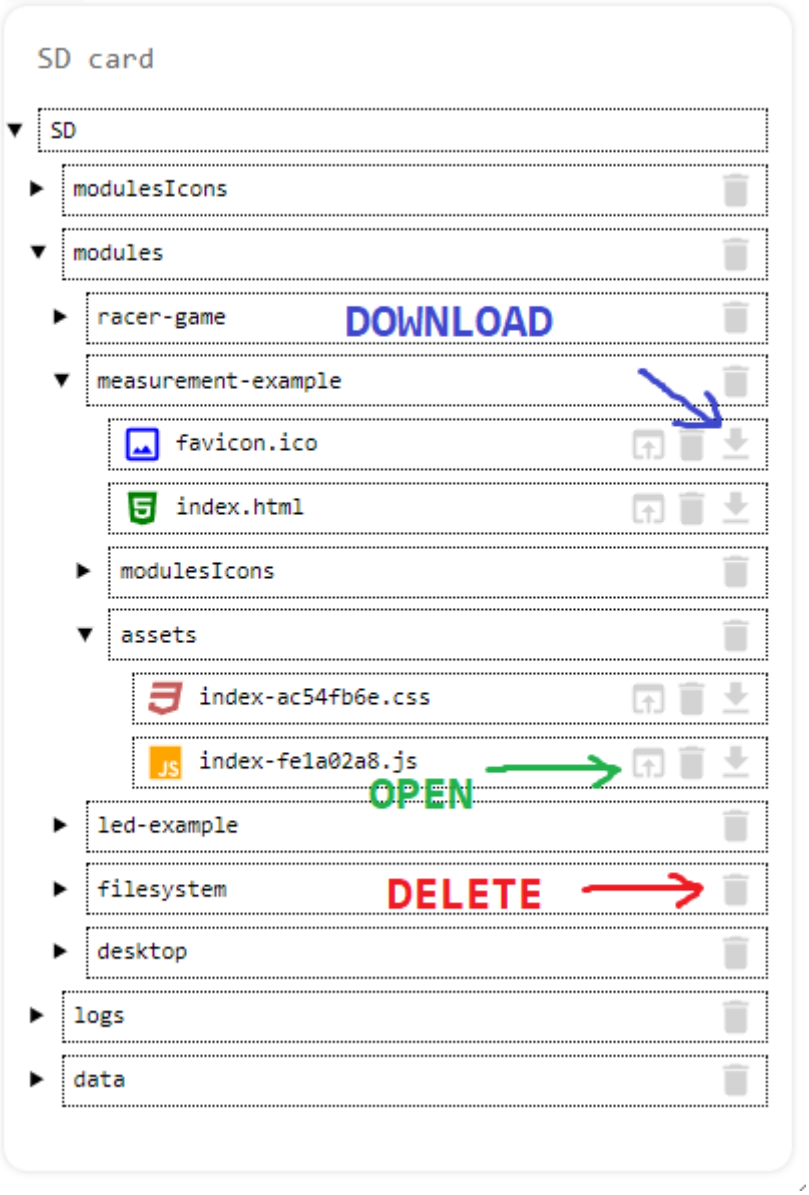
Joonis 22. Rakenduste ülevaatepaneel filtri külgribaga, blokeeritud rakenduste kuvamine kategooriates "Sensor" ja "Actuator".

Joonis 23. Näide selle kohta, kuidas rakenduse nime hõljutamisel kuvatakse.



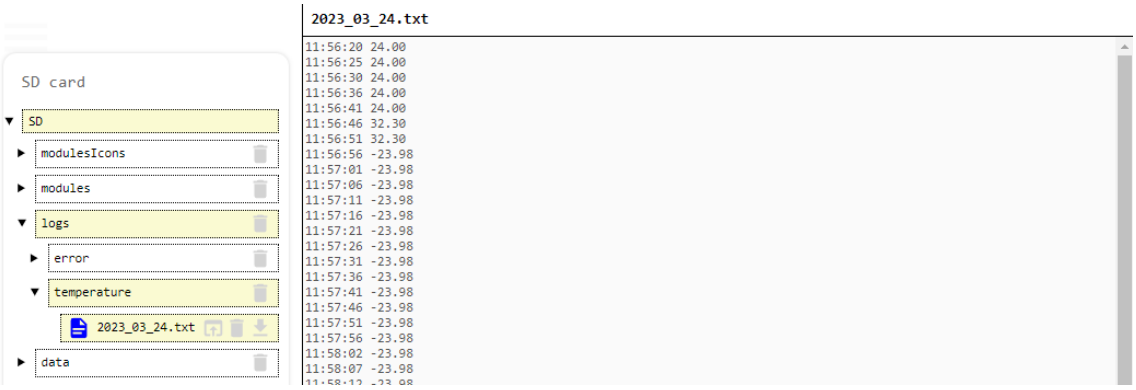
Joonis 24. Kuidas failide avamise ikoon välja näeb ja kus see asub.

Joonis 25. Faili avamise dialoogiboks.

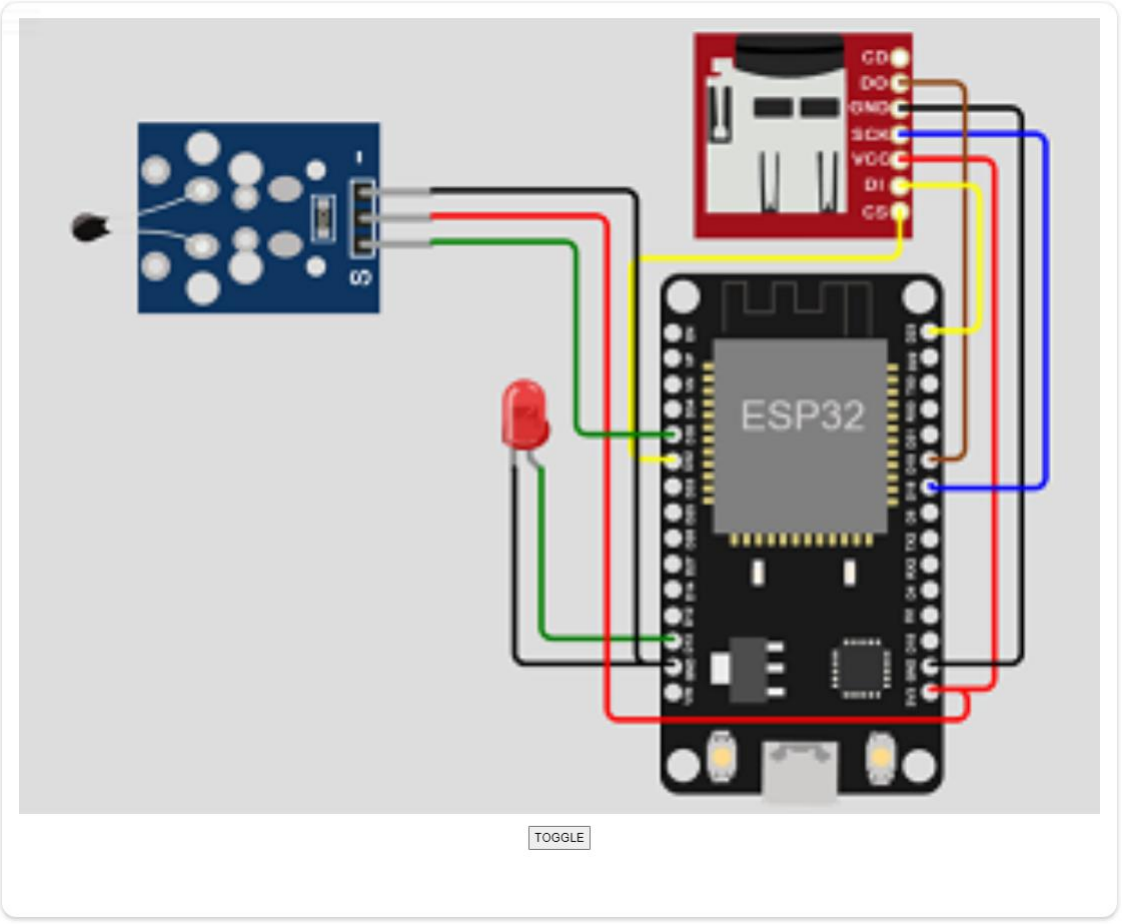


Pilt 1. Puuvaates kuvatavad kataloogid ja failid.

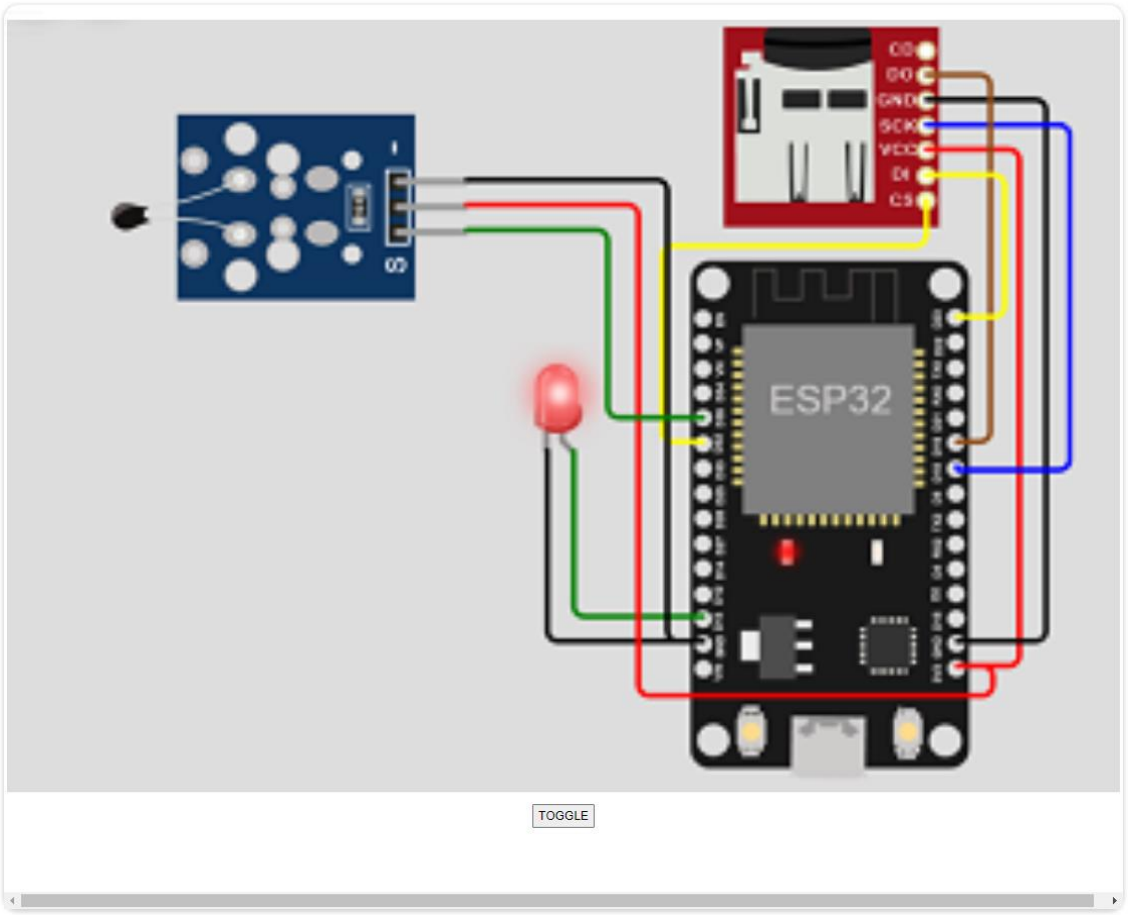
Joonis 26. Puuvaate elemendid, mille kohta laaditakse praegu teavet nende alamkataloogide ja failide kohta.



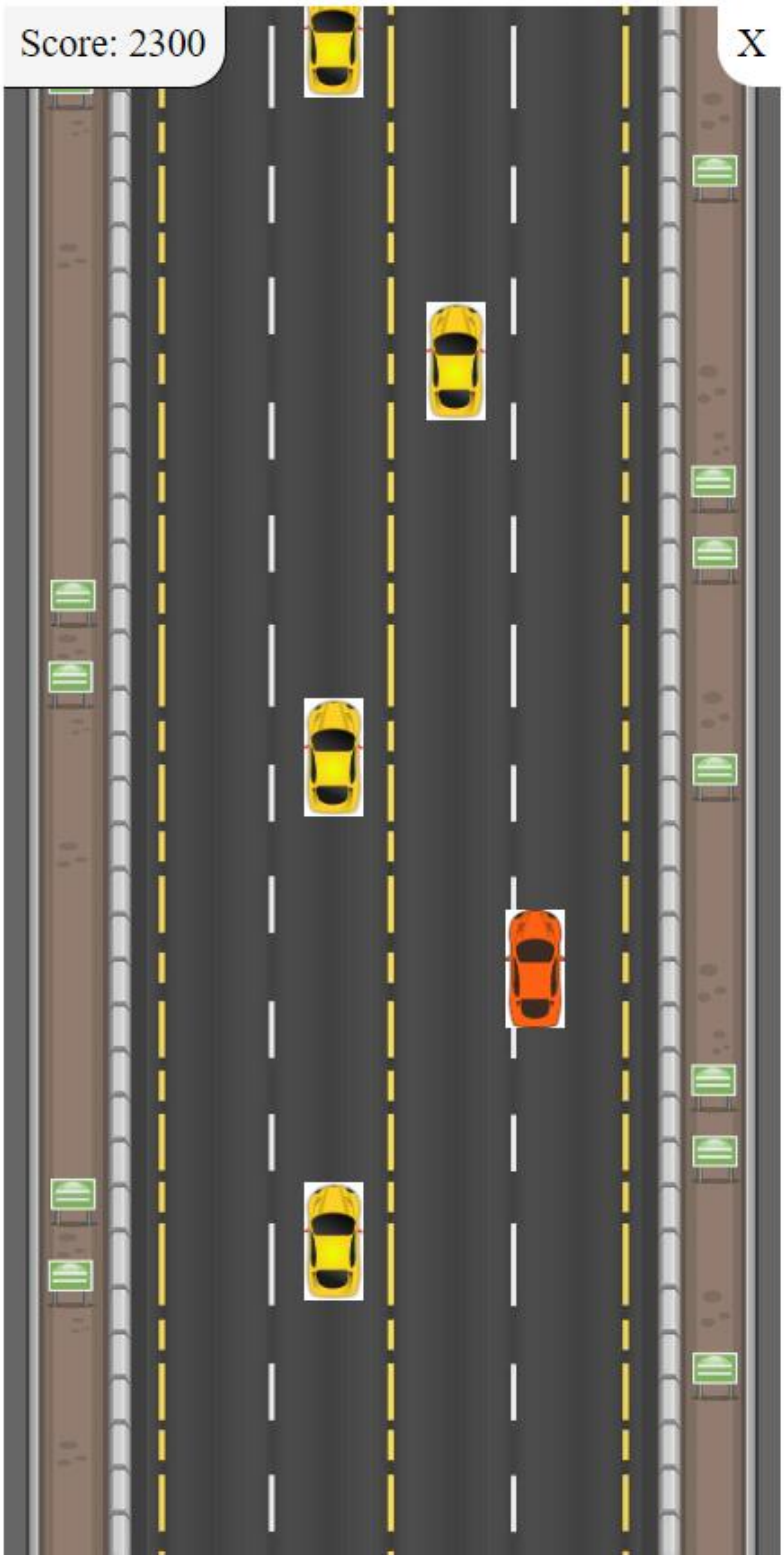
Joonis 27. Salvestatud andmete näide.



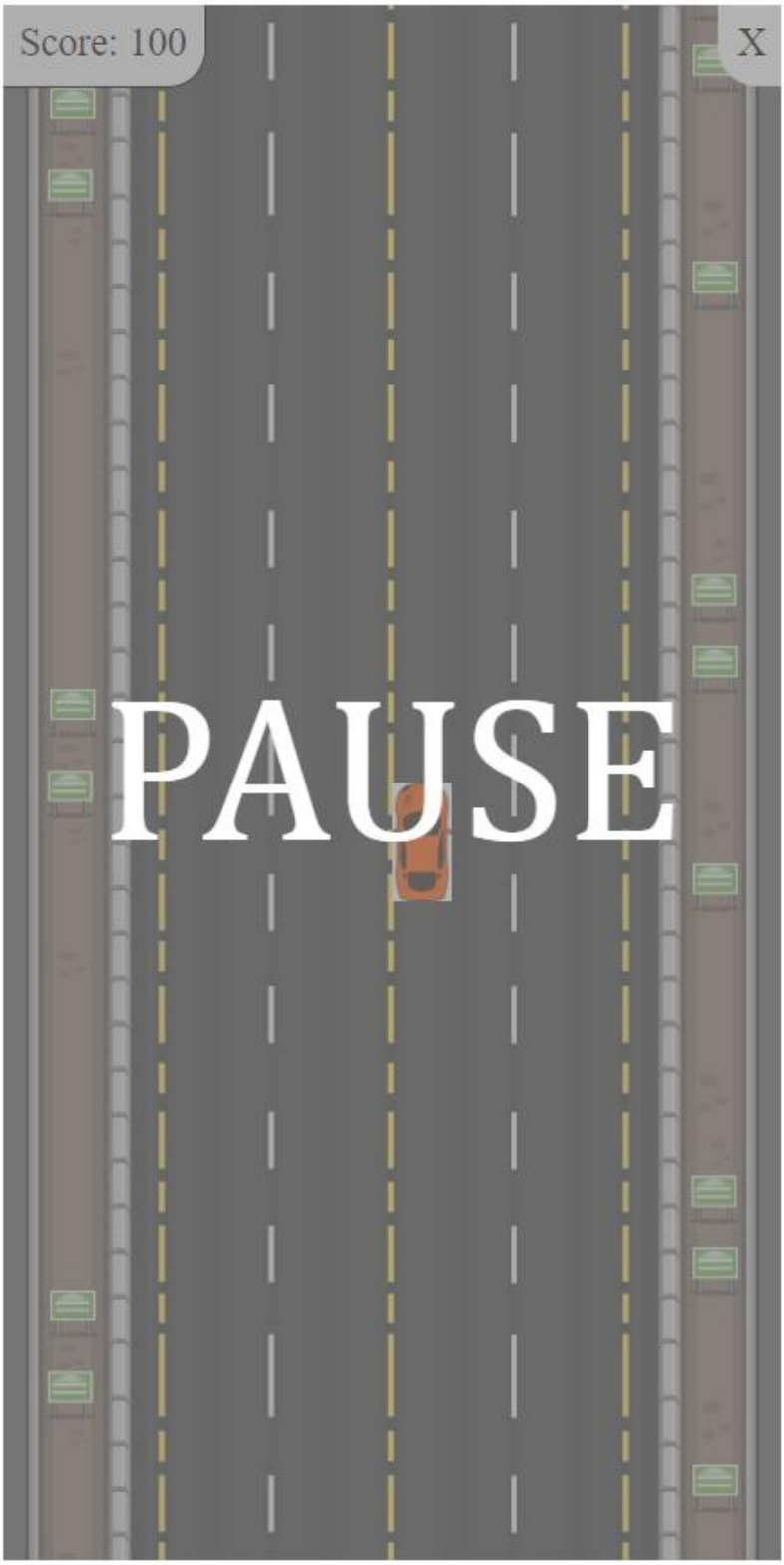
Joonis 28. Rakenduse leht, kus LED on välja lülitatud.



Joonis 29. Rakenduse leht põleva LED-iga.



Joonis 30. Tee mängijaautoga (punane) ja välditavate autodega (kollane).



Joonis 31. Paus mängus.

Joonis 32. Mäng on kaotatud.

USERNAME	SCORE
Steve	4335
Mr_Just_Do_It	2135
FirstTry	1750
MegaPro	850
HyperExpert_9999999	0

EXIT

Joonis 33. Rekordite tabel.