RĪGAS VALSTS 3. ĢIMNĀZIJA

Projekta darbs

Programmēšana II

Automatizēta meteoroloģiskā stacija ar ESP32 un datu uzglabāšanu

12. C1 klases skolnieks

Artūrs Nelsons

Rīga 2025

SATURS

1. IEVADS 1
2. TEORIJA 2
2.1. Meteoroloģija 2
2.2. IoT un tā pielietojums meteoroloģijā 2
2.3. Eksistējoši risinājumi un mikrokontrolleru izvēle projektam 3
2.4. REST API principi un projekta API arhitektūra 3
2.5. Izmantotā elektronika un tās specifikācija 5
2.6. Datu glabāšanas un servera infrastruktūras risinājumi 6
3. PRAKTISKĀ DAĻA 8
3.1. Programmu izstrāde 8
3.2. Tīkla pieejamība un domēna strukturēšana 10
3.3. Meteostacijas salikšana 10
4. SECINĀJUMI UN PRIEKŠLIKUMI 12
4.1. Secinājumi 12
4.2. Priekšlikumi 13
5. ATSAUCES 14

1. IEVADS

Mūsdienu tehnoloģijas ļauj izveidot dažādu veidu sistēmas, kas ir savienotas ar internetu, piemēram, noliktavas uzskaiti, sabiedriskā transportlīdzekļa atrašanās vietas noteikšanu un paredzēto ierašanās laiku aprēķināšanu u.c. Šādas sistēmas ir daļa no *Internet of Things* (IoT) jeb lietu interneta. IoT pazemina izmaksas, jo tiek samazināts manuālo darbību skaits un tiek automatizētas periodiskas darbības [11].

Meteoroloģiskās stacijas izveide ir aktuāla, jo šobrīd daži esošie risinājumi, piemēram, OpenWeatherAPI, nepadara skaidri saprotamu, kāda ir datu izcelsme [18]. Vēl viens iemesls ir tāds, ka iepriekš izveidotas meteoroloģiskās stacijas ir dārgākas nekā paštaisītas meteoroloģiskās stacijas [1]. Turklāt, stacijas izstrādāšana arī popularizēs meteoroloģiju, kā rezultātā dati kļūs brīvāk pieejami un būs zināms to avots.

Šī projekta darba mērķis ir izveidot meteoroloģisko staciju, kuras dati tiek augšupielādēti mākoņa datubāzē un vēlāk tiem dod iespēju piekļūt reāllaikā, izmantojot lietojumprogrammu saskarni (API).

Projekta darba uzdevumi:

- Veikt literatūras izpēti par mikrokontrolleri ESP32 un sensoriem, kuri tiks izmantoti;
- Analizēt jau eksistējošus risinājumus entuziastu meteorloģiskajām stacijām;
- Iegūt zināšanas par IoT un datu pārraides principiem, īpaši pievēršot uzmanību datu sūtīšanu uz “MariaDB” datubāzi, izmantojot Wi-Fi;
- Izveidot API, kas ir savienota ar “MariaDB” datubāzi, izmantojot FastAPI platformu programmēšanas valodā “Python”, lai nodrošinātu datu pieprasījumu un apstrādi;
- Izveidot meteoroloģisko staciju ar sensoriem, kas savienota ar ESP32 mikrokontrolleri, kas savienota ar API;
- Izveidot mājaslapu, kura periodiski ievāc datus no API un parāda tos kartē;
- Testēšana un validācija, secinājumi.

2. TEORIJA

2.1. Meteoroloģija

Meteoroloģija ir definēta kā zinātne par atmosfēru, atmosfēras parādībām un atmosfēras ietekmi uz mūsu laikapstākļiem. Meteoroloģijai ir arī vairāki mērogi: mikromēroga (parādības, kam ir mazs izmērs un ir īss dzīves ilgums), mezomēroga (parādības, kuru izmērs ir no dažiem kilometriem līdz aptuveni 1000 kilometriem) un sinoptiskā mēroga meteoroloģija (parādības, kuru izmērs ir no vairākiem simtiem līdz vairākiem tūkstošiem kilometriem) [19]. Atmosfēra ir gāzveida slānis, kas galvenokārt satur slāpekli, skābekli, argonu un ogļskābo gāzi. Tās biezums ir aptuveni 100 kilometri. [4].

Dati, kas tiek vākti meteoroloģijā, īpaši mezomēroga meteoroloģijā, ietver temperatūru, atmosfēras spiedienu, vēja ātrumu, relatīvo mitrumu u.c. Kombinējot šos datus, ir iespējams prognozēt tuvāko laikapstākļu attīstību.

2.2. IoT un tā pielietojums meteoroloģijā

*Internet of thing*s (IoT), jeb lietu internets, ir fizisku ierīču, transportlīdzekļu un citu fizisku objektu tīkls, kas ir aprīkots ar sensoriem, programmatūru un interneta savienojumu, ļaujot tiem vākt un koplietot datus [11].

Ierīces, kas padara IoT iespējamu, sauc par viedajiem objektiem. Tās var būt tik vienkāršas kā viedās mājas ierīcēs, piemēram, termostats un tik advancētas kā valkājamas ierīces, piemēram, viedpulksteņi [11].

IoT ierīces lielākoties ievāc datus un augšupielādē tos mākonī, kur datus apstrādā jaudīgāks dators, ko sauc par serveriem. Ierīču programmēšana parasti notiek programmēšanas valodā “C++”, taču pati ierīce ir parasti Arduino vai ESP mikrokontrollers.

Datu aktualitāte katru dienu kļūst arvien svarīgāka. Šobrīd Latvijā atrodas 33 standartizētas meteostacijas, par kurām ir atbildīgs Latvijas vides, ģeoloģijas un meteoroloģijas centrs, kuru dati tiek izmantoti lai prognozētu laikapstākļus visai Latvijai. Kaut gan šis skaits ir pietiekams pēc Pasaules Meteoroloģijas organizācijas normatīviem [31], meteostaciju skaita palielināšana uzlabotu prognozes. Tā kā tas nav pārāk izdevīgi, tad hobijisti var izveidot paši savas meteostacijas, kuras ir izdevīgākas, kā rezultātā vispārēja laikapstākļu prognoze varētu kļūt akurātāka Latvijas teritorijā.

2.3. Eksistējoši risinājumi un mikrokontrolleru izvēle projektam

Viens no eksistējošiem risinājumiem, kas ir pieejami tiešsaistē, izmantoja spiediena sensoru BMP180, mitruma sensoru DHT22, anemometru, kas ir vēja ātruma mērītājs, vēja virziena noteicēju un nolijušā ūdens līmeņa noteicēju. Liela daļa detaļu tika drukātas ar 3D printeri. Dati tiek sūtīti uz "Home Assistant", vietne, kur var savienot IoT ierīces. [12]

Cits risinājums ir iegādāties meteostaciju no interneta veikaliem, piemēram, Amazon.com, taču to cenas ir ievērojami augstākas un mazāka iespēja modificēt.

Mikrokontrolleris ir maza ierīce ar zemu jaudu, kas satur centrālo procesoru (CPU), operatīvo atmiņu (RAM), programmu un datu lasāmatmiņu un vairākus ievades/izvades portus.

ESP32 ir izdevīgs mikrokontrolleris, kam ir iespēja savienoties ar Bluetooth un Wi-Fi. Tam ir 2 kodolu procesors ar ātrumu 240 MHz [26, 3. lpp]. Šo mikrokontrolleri ražo uzņēmums “Espressif Systems” Šanhajā, Ķīnā [7].

Vēl populāri mikrokontrolleru dizaini ir no uzņēmuma “Arduino”. Šie mikrokontrolleri parasti ir mazāk jaudīgi nekā to “Espressif Systems” līdzinieki un ir draudzīgāki jaunpienācējiem, taču tie parasti nenāk ar iespēju savienoties ar Wi-Fi vai Bluetooth [27].

2.4. REST API principi un projekta API arhitektūra

API ir kods, kas ļauj divām programmatūrām sazināties. API dizains nosaka pareizo veidu, kā izstrādātājs raksta programmu [2]. Ir ļoti svarīgi, lai API būtu drošs un nebūtu iespējas izmantot kādu ievainojamību kā arī dizainam ir jābūt saprotamam un intuitīvam.

RESTful API ir veids, kā var dizainēt API. REST API pamatā sastāv no 3 daļām: klients, serveris un resurss. Klients ir programmatūra, kas izmanto API. Serveris ir tas, kas dod pieeju šiem datiem un resurss ir dati vai saturs, ko serveris kontrolē un padara pieejamu klientam. Lai klients piekļūtu resursam, viņam ir nepieciešams aizsūtīt HTTP pieprasījumu serverim. Šis pieprasījums satur:

- HTTP metodi – metode nosaka, ko klients vēlēsies darīt. Tas parasti seko principam CRUD (**C**reate, **R**ead, **U**pdate, **D**elete):
    - CREATE (POST), mērķis ir kaut ko izveidot, piemēram, pievienot vēl vienu rindu datu bāzei;
    - READ (GET) mērķis ir, piemēram, iegūt informāciju no API;
    - UPDATE (PUT, PATCH) atjaunina esošos datus vai nu pilnīgi, vai nu daļēji;
    - DELETE (DELETE) izdzēš pieprasīto.

- Galapunktu (endpoint) - nosaka vietu, kur atrodas pieprasītā lieta, piemēram, "api.example.com/**weather/name/meteostacija**".
- Galveni (header) - satur nepieciešamo informāciju, lai izpildītu pieprasījumu, piemēram, kāds datu tips tiek sūtīts (JSON, TXT, XML, u.c.).
- Ķermeni (body) - visi dati, kas tiek sūtīti uz/no servera.

[2]

Efektīvs veids, kā var izveidot RESTful API ir, izmantojot platformu "FastAPI" programmēšanas valodā "Python". "FastAPI" ir laba izvēle, jo tā izmanto salīdzinoši maz resursus un to ir viegli atkļūdot [9].

Lai uzglabātu meteostaciju datus, tiks izmantots datubāzes serveris "MariaDB". Šis serveris ir viens no populārākajiem pasaulē un tas tika atvasināts no "MySQL" [21]. Atšķirībā no "SQLite", ar “MariaDB” var veikt vairākas izmaiņas reizē, kamēr “SQLite” var veikt tikai viens uz tabulu – “SQLite” dara tabulas-līmeņa bloķēšanu, taču “MariaDB” dara rindiņu līmeņa bloķēšanu, kas to padara labāku priekš gadījumiem, kad datubāzei piekļūst vairāki klienti reizē [15].

Svarīgi ir izvairīties no t.s. "SQL injekcijām". SQL injekcijas ir tad, kad lietotājam ir iespēja ievadīt kādu SQL pieprasījumu kādā ievades laukā ļaunprātīgiem nolūkiem [20]. Piemēram, lietotājs lietotājvārda laukā ievada "abc OR 1=1". Tas, kas tiks pieprasīts datubāzei būs "SELECT * FROM Users WHERE username=**abc OR 1=1;**":

```
input = "abc OR 1=1"
sql = f"""SELECT * FROM Users WHERE username = {input};"""
cursor.execute(sql) # tā kā arī ielavījās OR 1=1, tad tas arī izpildīsies
```

Šāds pieprasījums atgrieztu visus reģistrētos lietotājus. Lai no šādiem incidentiem izvairītos, tiek izmantoti t.s. sagatavotie paziņojumi (*prepared statement*). Kad tiek izmantoti sagatavotie paziņojumi, tad visa ievade tiek uzskatīta kā viens liels lietotājvārds:

```
sql = """SELECT * FROM Users WHERE username = ?""";
cursor.execute(sql, ("abc OR 1=1",))
```

Šeit viss tiks uzskatīts kā tas pats lietotājvārds. Tā kā tāda visticamāk nav, tad nekas netiks atgriezts.

2.5. Izmantotā elektronika un tās specifikācija

Lai būtu meteostacija, ir nepieciešami vairāki sensori, kuri nolasa vērtības, tādēļ šajās (kopumā četrās) meteostacijās tiks izmantots:

- DHT22;
- BMP180;
- MQ135;
- YL-83.

Meteostacijai, kas atradīsies Carnikavā, būs arī anemometrs. Meteostacija izmantos mikrokontrolleri ESP32. Pirmā meteostacija, kas tiks izveidota, būs tā, kura atrodas Carnikavā, jo tur dzīvo darba autors un spēs nodrošināt, ka to nesabojās un neaiztiks neautorizētas personas. Pārējās tiks izveidotas 2025. gada vasarā.

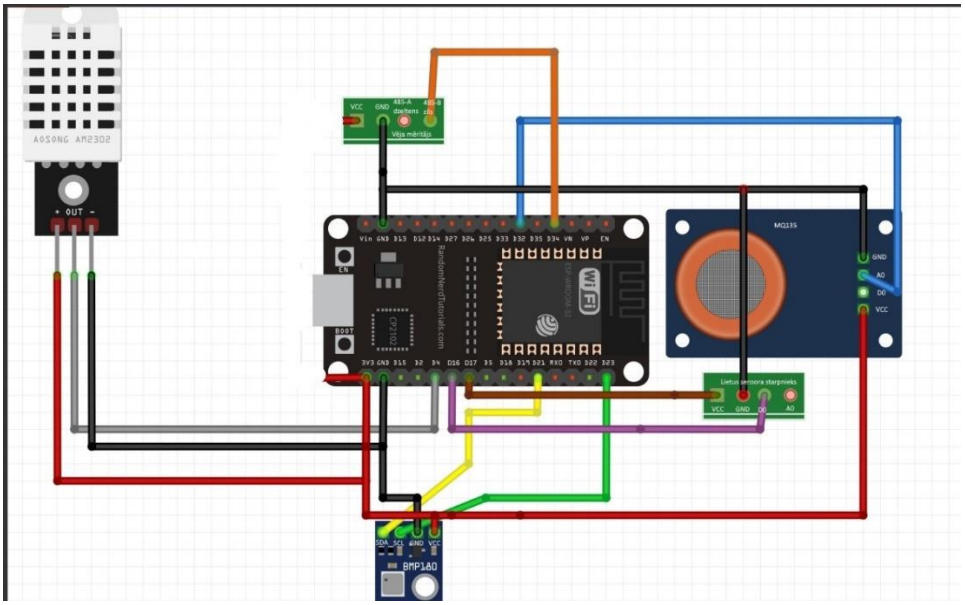
DHT22 ir digitāls sensors, kas mēra temperatūru un mitrumu. Tās mitruma mērīšanas precizitāte ir ± 2%, temperatūras precizitāte ir ±0.5°C un strādā spriegumā 3,3 – 5,5V [5]

BMP180 ir barometrs un digitāls sensors, kas mēra atmosfēras spiedienu un temperatūru, tas izmanto I2C protokolu un tā strādā spriegumā 3,3 – 5,5V. Šis barometrs spēj nolasīt vērtības no 300 hPa (hektopaskāliem) līdz 1100 hPa un tā precizitāte ir ±0.12hPa [3].

MQ135 ir izdevīgs analogs gāzes sensors, kas, izmantojot alvas oksīdu (SnO2), nosaka gaisa kvalitāti. Alvas oksīdam tīrā gaisā ir augstāka pretestība nekā piesārņotā gaisā- tātad, jo zemāks izvades spriegums, jo kvalitatīvāks ir gaiss [16].

YL-83 ir analogs sensors, kas nosaka, vai līst lietus/snieg sniegs. Tas strādā no 3.3 līdz 5V. Tas ir savienots ar LM393 salīdzinātāju [22]. LM393 salīdzinātājs ir precizitātes sprieguma salīdzinātājs, kas spēj darboties ar vienu vai vairākiem barošanas avotiem [23].

Anemometrs ir sensors, kas mēra vēja ātrumu. Tas sastāv no 3 konusiem, kā rezultātā tas tiek iegriezts vēja dēļ. Šis sensors strādā no 10 līdz 30 voltiem, taču izvada 0 līdz 5V spriegumu un spēj izmērīt vēja ātrumus līdz 30 metriem sekundē.

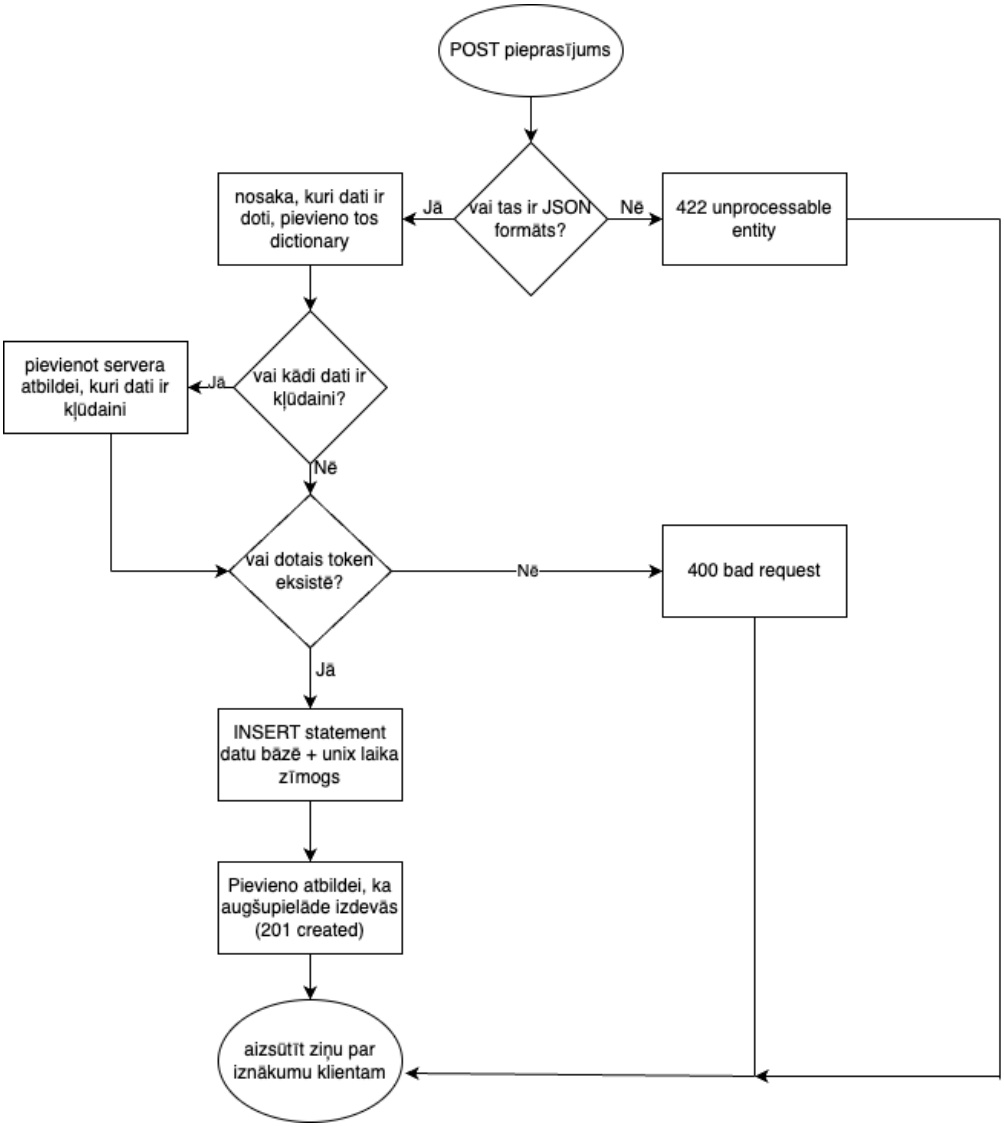


Attēls 1 - Meteostacijas shēma ar ESP32

2.6. Datu glabāšanas un servera infrastruktūras risinājumi

Lai būtu, kur uzglabāt meteostaciju datus, nepieciešams serveris, kur šos datus uzglabāt, tādēļ tiks izmantots mākoņa piedāvājums no uzņēmuma "Hetzner". Serverim būs 2 "Intel" procesora kodoli, 4GB brīvpiekļuves atmiņa un 40GB krātuves atmiņa. Operētājsistēma būs "Ubuntu" versija 20.04. Lai API būtu sasniedzams, tiks izmantota "Apache2" pakotne un, lai būtu iespējama mājaslapa ar drošu savienojumu (HTTPS), tiks izmantota pakotne "Certbot".

Lai meteostacija varētu augšupielādēt datus, ir nepieciešama saite, kur to var izdarīt. Tam būs paredzēts galapunkts "/upload". Kad stacija sagatavo lasījumus JSON formātā, tā izdara POST pieprasījumu, tad serveris to apstrādā un pievieno tos datubāzē. Skatīt 2. attēlu, kā strādās datu augšupielādēšana.



Attēls 2 - Laikapstākļu API blokshēma

3. PRAKTISKĀ DAĻA

3.1. Programmu izstrāde

Lai izveidotu programmatūru priekš ESP32, tiks izmantots Arduino IDE. Taču, lai izveidotu API, tiks izmantots PyCharm Professional IDE.

API izveidei tika izlemts veidot 5 galapunktus- 4 "GET" galapunkti:

1. /name/{name} - atgriež jaunākos datus no noteiktas meteostacijas
2. /country/{country}/town/{town} - atgriež jaunākos datus no nejaušas meteostacijas definētajā valstī un pilsētā
3. /name/{name}/amount/{amount} - atgriež pēdējos {amount} mērījumus no definētās stacijas.
4. /all-stations - atgriež visas definētās meteostacijas

un viens "POST" galapunkts, **/upload/**, kurš augšupielādē meteostacijas datus datubāzē.

Lai datus zinātniskiem un vispārējiem nolūkiem būtu vieglāk izmantot, ir arī sniegti pārrēķini SI sistēmā un citās izplatītās mērvienībās, piemēram, jūdzes stundā, mezgli, Fārenheita grādi, u.c. Turklāt API atbildes "info" sadaļā, tiek arī dots laiks ISO8601 formātā, kas laiku attēlo formātā `YYYY-MM-DD`, kur `YYYY`- gads, `MM` - mēnesis un `DD`- diena. Piemēram, lai attēlotu 2022. gada 27. septembri plkst. 18.00, raksta `2022-09-27 18:00:00.000` [24].

Lai programmatūra atbilstu drošības standartiem un lai projekts būtu mērogojams, tika izmantota bibliotēka "dotenv", lai failā ".env", varētu ātrāk nomainīt mainīgos un lai citi varētu uzstādīt līdzīgus API uz saviem serveriem, piemēram:

```
conn = mariadb.connect(
        user=os.getenv("DB_USER"),
        password=os.getenv("DB_PASSWORD"),
        host="localhost",
        port=3306,
        database=os.getenv("DB_NAME")
    )
```

, kur, piemēram, "DB_USER", failā ".env" būtu:

```
DB_USER="weather".
```

Īpaši svarīgi bija uzstādīt atbilstošu drošību serverim. Tas tika sasniegts, failā `/etc/sudoers/` pievienojot lietotāja "weather" atļauju darīt visu:

```
"weather ALL=(ALL:ALL)ALL"
```

Papildus tam, failā "`/etc/ssh/sshd_config`" tika izmainīts lauks no

```
PermitRootLogin yes
```

uz

```
PermitRootLogin no
```

Izmainot šo lauku, pievienošanās lietotājam "root", kam ir atļauja darīt sistēmā visu, caur SSH (protokols, lai savienotos ar serveri [28]) uzreiz nav iespējams. Tā vietā vajag pievienoties serverim kā lietotājam "weather" un, komandrindā, ievadot "sudo su", ievadīt lietotāja "weather" paroli vēlreiz.

Taču, lai izveidotu programmatūru priekš ESP32, bija nepieciešams izveidot kompromisu - mikrokontrolleris pats par sevi neatbalsta savienojumu ar HTTPS, tādēļ API atslēgas drošība var būt apdraudēta. Veids, kā HTTPS var būt atbalstīts, ir iekopēt meteostacijas kodā SSL sertifikātu, tomēr sertifikāti no "LetsEncrypt" mainās ik pēc 90 dienām [8], kas nozīmē, ka katrus 3 mēnešus vajadzēs ESP32 kodā manuāli izmainīt sertifikātu. Var iegūt sertifikātus, kas ir derīgi ilgāk par 3 mēnešiem, bet tas veidos papildus izmaksas. Turklāt šīs meteostacijas radītie dati nav kritiski. Tādēļ ESP32 strādā izmantojot HTTP, nevis HTTPS.

Vēl tika izveidota vienkārša mājaslapa ar HTML, JavaScript un Leaflet, kas parāda meteostacijas aktuālos datus, pieprasot jaunumus katru minūti. Leaflet ir JavaScript bibliotēka, kas atļauj izveidot interaktīvu karti [14]. Skatīt 3. attēlu, kā izskatās izveidotā mājaslapa. Tā kā darba autors negrib atklāt savu tiešo atrašanās vietu, tad punkts ir novietots Gaujā.

Attēls 3 - Izveidotā interaktīvā mājaslapa, kas saņem datus no API

Lai meteostacija varētu augšupielādēt datus, tika arī izveidota programmatūra priekš tās programmēšanas valodā “C++”. Meteostacija uzsāk savu dzīvesciklu, savienojoties ar Wi-Fi. Ja tas neizdodas, meteoroloģiskā stacija pārtrauc darbu. Tā arī pārbauda, vai sensori strādā, taču, ja kāds nestrādā, darbs netiek pārtraukts. Ja savienojums izdodas, tad meteostacija katras 5 minūtes nolasa vērtības no sensoriem, un JSON formātā tās aizsūta uz API caur galapunktu "/upload/".

3.2. Tīkla pieejamība un domēna strukturēšana

Lai savienošanās ar kādu mājaslapu, piemēram, "google.com", būtu iespējama, tiek izmantota t. s. domēnu nosaukumu sistēma, jeb DNS. DNS darbojas kā interneta infrastruktūras pamatelements, kas ir atbildīgs par domēnu vārdu tulkošanu atbilstošajās interneta protokola (IP) adresēs [30].

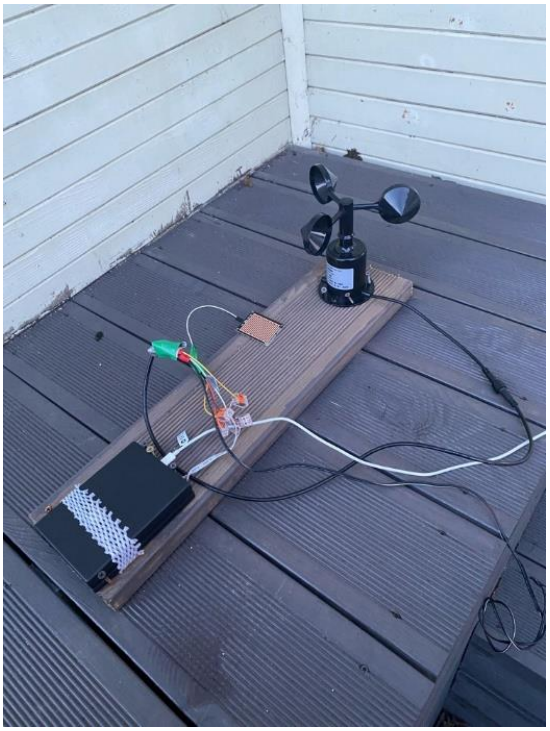
Taču, lai savienošanās ar mājaslapu būtu iespējama, lielākoties izmanto adreses (A) rekordu, kas satur apakšdomēna vārdu (piemēram, **maps**.google.com) un uz kuru IP adresi tas novirza, piemēram, 142.250.74.142 [29].

Lai savienošanās ar API un karti būtu ērta, tad API tika piesaistīta adrese api.nelsons.lv, taču kartes mājaslapai tika piesaistīta adrese weather.nelsons.lv. Tas tika izdarīts, izmantojot mājaslapu nic.lv, kuri pārvalda .lv augstākā līmeņa domēnu [25].

3.3. Meteostacijas salikšana

Meteostacijas salikšanas laikā bija sastapšanās ar divām problēmām: padarīt staciju noturīgu pret samirkšanu lietū, kā arī, lai vadu savienojumi ir noturīgi. Lai atrisinātu pirmo problēmu, tika veikts pieprasījums nozares profesionālim to salikt tādā veidā, lai meteostacija būtu noturīga pret laikapstākļiem. Taču, lai savienojumi ar vadiem būtu uzticami, tika izmantoti Wago 221 savienotāji, kas ļauj ērtā veidā ar svirām savienot divus vadu galus. Skatīt 4. attēlu, kā izskatās meteostacijas gala versija.

Taču tad, kad meteostacija bija salikta, neliela problēma bija novietot to tādā vietā, kur mājas Wi-Fi to var visu laiku sasniegt. Tas tika atrisināts, pieliekot meteostaciju tuvāk logam.



Attēls 4 - Pabeigtā meteostacija

4. SECINĀJUMI UN PRIEKŠLIKUMI

4.1. Secinājumi

Šī projekta darba mērķis bija izveidot meteoroloģisko staciju ar zemām izmaksām, kas ir savienota ar internetu un tas arī izdevās. Šādas meteostacijas izveide ir viens no pirmajiem soļiem, lai dati par laikapstākļiem kļūtu daudz pieejamāki, kas varētu būt svarīgi mašīnmācīšanās modeļiem kļut akurātākiem. Hobijistiem šādu staciju izveidot ir ļoti reālistiski un labs veids, kā attīstīt gan meteoroloģijas prasmes, gan IT prasmes. Taču ir ļoti svarīgi nodrošināt datu precizitāti, liekot sensorus atklātā vietā.

Projekta darba izstrādes laikā tika:

- Izveidota meteoroloģiskā stacija ar ESP32 mikrokontrolleri, kura ir savienota ar internetu un sūta savus datus uz API katras 5 minūtes. Tas dod iespēju datiem būt vairāk atbilstošiem reālajam laikam;
- Izveidota lietojumprogrammas saskarne (API) api.nelsons.lv programmēšanas valodā "Python" platformā "FastAPI", kuras mērķis ir uzņemt datus no meteostacijām un padarīt tos publiskus jebkuram interneta lietotājam. Šis ļauj jebkuram hobijistam pievienot savus datus no savas meteostacijas;
- Izpētīta literatūra par serveru un datu bāzu drošību;
- Izveidota mājaslapa weather.nelsons.lv, kuras mērķis ir rādīt aktuālos datus par meteostacijām interaktīvā kartē. Šīs mājaslapas mērķis ir parādīt potenciālu veidu, kā izmantot datus, kā arī lai hobijisti varētu kopīgot datus vieglākā veidā;
- Izpētīta literatūra par esošām vaļasprieka līmeņa un patērētāju meteostacijām, kuru cenas ir no 70€ līdz aptuveni 90€. Darba autora izveidotā meteostacija izmaksāja aptuveni 40€, kas ir ievērojami zemāka cena;
- Pievērsta īpaša uzmanība standartiem, tai skaitā ISO8601 laika zīmoga standartam, SI sistēmas un citām plaši izmantotām mērvienībām. Tas dod iespēju šim projektam globalizēties.

4.2. Priekšlikumi

- Atvērt API publiskai izmantošanai 2025. gada vasaras laikā, vispirms uzlabojot tā drošību. Tas ir tādēļ, lai cilvēki no jebkurienes varētu izmantot uzticamu API saviem meteostaciju datiem;
- MQ135, DHT22, BMP180 sensoru vietā izmantot vienu sensoru BME680, kas mēra gan gaisa kvalitāti, gan relatīvo gaisa mitrumu, gan atmosfēras spiedienu, gan gaisa temperatūru [10, 1. lpp], tas ievērojami vienkāršotu elektroinstalāciju un samazinātu enerģijas patēriņu;
- Tā kā liela daļa sensoru atrodas melnā kastē, tas nozīmē, ka daži no mērījumiem, piemēram, gaisa temperatūra, nav pietiekami akurāti. Lai to atrisinātu, ir nepieciešama laba gaisa cirkulācija ar, piemēram, gaisa ventilatoriem;
- Izveidot atlikušās 3 meteostacijas vasaras laikā, lai varētu paplašināt tīklu un lai būtu pieejami dati arī no citām vietām.

•

5. ATSAUCES

[1] Amazon.com : Weather stations. (n.d.). https://www.amazon.com/weather-stations/b?ie=UTF8&node=397435011

[2] Bigelow, S. J., & Gillis, A. S. (2024, May 10). *RESTful API*. Search App Architecture. https://www.techtarget.com/searchapparchitecture/definition/RESTful-API

[3] BMP180 - Atmospheric Pressure Sensor. (n.d.). Components101. https://components101.com/sensors/bmp180-atmospheric-pressure-sensor

[4] Cermak, A. (2024, October 22). *The Atmosphere: Getting a handle on carbon dioxide - NASA Science*. NASA Science. https://science.nasa.gov/earth/climate-change/greenhouse-gases/the-atmosphere-getting-a-handle-on-carbon-dioxide/

[5] DHT22 Temperature-Humidity Sensor - Waveshare Wiki. (n.d.). https://www.waveshare.com/wiki/DHT22_Temperature-Humidity_Sensor

[6] Earth atmosphere. (n.d.). https://www.grc.nasa.gov/www/k-12/airplane/atmosphere.html

[7] ESP32 Wi-Fi & Bluetooth SOC | Espressif Systems. (n.d.). https://www.espressif.com/en/products/socs/esp32

[8] FAQ. (n.d.). https://letsencrypt.org/docs/faq/

[9] FastAPI. (n.d.). https://fastapi.tiangolo.com/

[10] Gas Sensor BME680. (n.d.). Bosch Sensortec. https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors/bme680/

[11] IBM. (2025, April 16). Internet of Things. *IBM*. https://www.ibm.com/topics/internet-of-things

[12] Instructables. (2025, April 20). *DIY weather station with ESP32*. Instructables. https://www.instructables.com/DIY-Weather-Station-With-ESP32/

[13] John. (2022, March 25). *NodeMCU ESP8266 Specifications, Overview and setting up*. Make-It.ca. https://www.make-it.ca/nodemcu-details-specifications/

[14] Leaflet — an open-source JavaScript library for interactive maps. (n.d.). https://leafletjs.com/

[15] MariaDB vs SQLite | What are the differences? (2020, November 5). StackShare. https://stackshare.io/stackups/mariadb-vs-sqlite

[16] MQ-135 - Gas Sensor for Air quality. (n.d.). Components101. https://components101.com/sensors/mq135-gas-sensor-for-air-quality

[17] Oliynyk, K. (2025, April 15). Weather monitoring system using IoT: Benefits & importance. *webbylab*. https://webbylab.com/blog/weather-reporting-system-using-iot-benefits-use-cases/

[18] OpenWeatherMap.org. (n.d.). *Current weather data - OpenWeatherMap*. https://openweathermap.org/current

[19] The Science and Art of Meteorology. (n.d.). https://education.nationalgeographic.org/resource/science-art-meteorology/

[20] W3Schools.com. (n.d.). https://www.w3schools.com/sql/sql_injection.asp

[21] MariaDB.org. (2023, June 19). *MariaDB in brief - MariaDB.org*. https://mariadb.org/en/

[22] Snow & Raindrops Detection Sensor Module YL-83. (n.d.). Twins Chip. http://twinschip.com/Snow_Raindrops_Detection_Sensor_YL83

[23] onsemi. (n.d.). *Low offset voltage dual comparators* [Datasheet]. https://www.onsemi.com/download/data-sheet/pdf/lm393-d.pdf

[24] ISO - ISO 8601 — Date and time format. (2017, February 21). ISO. https://www.iso.org/iso-8601-date-and-time-format.html

[25] Lumii, L. U. M. U. I. I. T. R. D. (n.d.). *Par mums*. LUMII, Latvijas Universitātes Matemātikas Un Informātikas Institūts, Tīkla Risinājumu Daļa (NIC). https://www.nic.lv/lv/par-mums

[26] Espressif. (n.d.). *ESP32 Series* [Datasheet]. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[27] Agarwal, T. (2024, May 21). *ESP32 vs Arduino : Definition & the Main Differences*. ElProCus - Electronic Projects for Engineering Students. https://www.elprocus.com/difference-between-esp32-vs-arduino/

[28] Cloudflare. (n.d.). What is SSH. *Cloudflare*. https://www.cloudflare.com/learning/access-management/what-is-ssh/

[29] Cloudflare. (n.d.-a). DNS A record. *Cloudflare*. https://www.cloudflare.com/learning/dns/dns-records/dns-a-record/

[30] Cloudflare. (n.d.-b). What is DNS? *Cloudflare*. https://www.cloudflare.com/learning/dns/what-is-dns/

[31] *Latvijas Vides, ģeoloģijas un meteoroloģijas centrs*. (n.d.). https://videscentrs.lvgmc.lv/lapas/meteorologiskais-tikls