Всеукраїнській конкурс студентських наукових робіт
з природничих, технічних та гуманітарних наук

Спеціальність: Інженерія вбудованих систем

Конкурсна робота

Шифр роботи «Автономна метеостанція»

“Розробка автономної метеостанції”

2020-2021 навчальний рік

Зміст

Анотація

Актуальність роботи зумовлена тим, що існує практична потреба у тому, щоб була можливість проводити моніторинг погодних умов в екстремальних умовах та мати представлення про приблизний прогноз погоди на наступний день.

Метою наукової роботи є полегшення життя людини, що знаходиться в екстремальних умовах, а саме забезпечення її інформацією про стан навколишнього середовища, а саме – про погоду, та забезпечення її прогнозом погоди без використання будь-яких засобів зв'язку.

Об'єкт досліджень – погодні умови навколишнього середовища, процес зміни їхнього стану.

Предмет досліджень – різноманітні характеристики погодних умов, а саме: температура, тиск, концентрація CO2, концентрація легких органічних речовин в повітрі, залежність одних характеристик погодних умов від інших.

Завдання дослідження, які дозволяють досягти поставленої мети:

1. Аналіз існуючих методів дослідження погодних умов, методів розробки прогнозів погоди

2. Розробка власних інноваційних методів дослідження та аналізу погодних умов.

3. Розробка та створення міні-комп'ютера з використанням набору сенсорів для збору, аналізу та прогнозуванню змін стану повітря.

4. Розробка відповідного ПЗ, що буде працювати на міні-комп'ютері, встановлення цього ПЗ

При виконанні досліджень та прийнятті рішень використовувались такі **методи та підходи**: побудова нейромереж, методи побудови міні-комп'ютерів на базі Arduino, теорія алгоритмів

Наукова новизна:

Був створений новий аналітичний метод для аналізу стану погодних умов та процесу їхніх змін, який виділяється тим, що він використовує нейронну

мережу, що дає можливість робити йому постійні вдосконалення для забезпечення дуже хорошої точності.

Практичне значення:

Було створено новий пристрій для збору та аналізу даних про погодні умови, який дозволяє отримати відносно точні прогнози погоди, без великої витрати обчислювальної потужності. Це дозволяє отримати прогнози з прийнятною для використання точністю у багатьох випадках.

Стислий опис результатів дослідження: була визначена архітектура міні-комп'ютера, обрани основні компоненти та, власне, була проведена збірка цього пристрою. Сам пристрій був доволі успішно протестований та вже знаходиться в робочому стані. В процесі виконання роботи була створена доволі проста нейронна мережа, що видає непогані за точністю результати. Також, було зроблене порівняння результативності мережі порівняно з іншим способом побудови статистичних передбачень – побудова лінійної регресії.

Робота має обсяг 29 аркушів, має 20 малюнків, 2 таблиці, 10 джерел посилань, додатки на 7 аркушах.

Ключові слова: Мікрокомп`ютер, С – розробка, Arduino, нейромережа, прогноз

Вступ

Доволі часто при невеликих та невеликих походах або при поїздках на віддалені від розвиненої цивілізації місцях є потреба у тому, щоб знати які зараз погодні умови та якими приблизно вони будуть на наступний день. Через те, що далеко не завжди є доступ до інтернету, в таких випадках можна лише орієнтуватися на передчасно перевірені дані, які в будь-якому випадку не будуть коригуватися та оновлюватися, а поточну температуру можна буде на 100% дізнатися лише по власних почуттях, що не є точним та навіть може призвести до поганих наслідків.

Найбільш близька аналогія до пристрою, що розроблюється, яку можна знайти – це домашні метеостанції, проте вони є доволі обмеженими в своєму використанні. Вони не є автономними та мобільними, їх не можна взяти з собою через потребу у постійному струмі та доволі великі розміри. Завдяки інноваційним методам дослідження погодних умов, а також розширеним можливостям міні-комп'ютера та його програмного забезпечення є можливим робити більш комплексне дослідження зміну стану погоди протягом певного часу, і за рахунок цього робити базований на науковому ґрунті прогноз. Ще один аналогічний пристрій – портативна метеостанція, проте вона не має можливості дати прогноз на наступний день.

Ще одна особливість рішення, що створюється це наявність можливості постійно покращувати та добудовувати головний пристрій для збору більшої кількості даних різних типів та робити все складніший аналіз над цими даними власне в цьому ж пристрої.

1. Аналіз стану питання та постановка задачі

Питання того, як дізнаватися погоду у поточний час не маючи доступу до інформаційних мереж стоїть доволі давно. На ринку портативних метеостанцій існує вже певна кількість відомих фірм, що випускають моделі своєї власної збірки. Проте, точно не можна сказати, що цей ринок є переповненим та повністю окупованим монополістами. У напрямі автономних метеостанцій точно є куди ще розвинутися та цей проект може стати хорошим поштовхом для цього.

Чому взагалі потрібні портативні метеостанції? Далеко не завжди є можливість дізнатися поточну погоду через екстремальні умови в яких знаходиться людина – це може бути якийсь похід, відпустка на дачі, просто проблемне місце без доступу до інформаційних мереж, в яке людина випадково потрапила. Інформація про поточну погоду завжди є доволі актуальною для людини, що знаходиться в таких ситуаціях. Наприклад, маючи певний досвід у віддалених подорожах мандрівник базуючись на своєму досвіді зможе зробити певні висновки щодо наступних дій у своєї ситуації, також, знов на основі власного досвіду, зробити власний прогноз погоди на наступний день. Для випадків коли ж досвіду може не вистачати, прогноз погодних умов буде на сто відсотків корисною річчю.

Ще одна актуальна проблема сучасного світу – це забрудненість повітря. Кожен день різноманітний транспорт та промислові виробництва викидають велику кількість газових відходів, що є основною причиною глобального потепління. Але, ще одна важлива проблема шкідливих викидів це безпосередня шкода здоров’ю. Доволі велика кількість людей живе значно менший час, ніж могла б через хронічні хвороби, що виникають на ґрунті забрудненого повітря. Через це є важливим слідкувати за концентрацією $CO2$ та шкідливих для людини речовин, щоб вона могла одразу якось змінити навколишнє оточення, поки ще не стало пізно. Міні-комп’ютер матиме можливість вести необхідні спостереження у будь-який час, навіть у домашній обстановці.

1.1 Аналіз стану в прикладної галузі

На даний момент існує доволі велика кількість портативних аналізаторів повітря та портативних метеостанцій. Усі вони мають схожий принцип роботи та виконують схожу функції. Головна особливість нашої розробки – це створення короткочасних прогнозів на основі попередніх даних, що робиться в автономному режимі. Також, відмічу певне різноманіття задач, що виконує пристрій – далеко не кожна портативна метеостанція має можливості аналізатора повітря та далеко не кожен аналізатор повітря має можливості метеостанції. Як правило, такі пристрої коштують великих грошей.

1.2 Аналіз теоретичних та методичних розробок

Способи побудови автономного пристрою для спостереження за станом погодних умов

Існує багато мікроконтролерів, що дозволяють реалізовувати ідеї в життя: Arduino, STM32, Raspberry Pi, Orange Pi та інші. Для проекту нам потрібен енергоефективний мікрокомп’ютер, тому ми обрали Arduino, а саме Arduino Mega на чіпі ATMega2560 із обсягом Флеш пам'яті у 256 Кб, що дозволить нам реалізувати велику функціональну базу, наявність EEPROM дозволяє нам зберегти конфігурацію на базі мікрокомп’ютера. Далі потрібно вибрати сенсори. Доволі непоганим та збалансованим варіантом по енергопотребі таякістю є CJMCU-811 та для суперекорежиму було додано модуль SHT31 із температурним сенсором та сенсором вологості

Рис. 1.1 Сенсорна апаратна складова

Способи розрахунку прогнозів погоди

На даний момент погода може бути спрогнозована з неймовірно великою точністю порівняно з тим, як це було лише декілька десятиріч потому. Прогнози погоди робляться двома шляхами: перший – за допомогою вмінь синоптика, що спеціально для цього навчався, другий – за допомогою доволі складних обчислень на потужних машинах [1]. Само собою перед цим необхідні дані для прогнозу (зміни тиску, напряму вітру, температур тощо). Так як прогнози робляться на далекій відстані від місць збору інформації, то дізнатися про такі передбачення можна лише якщо в людини є доступ до Всесвітньої мережі або інший засіб зв'язку. Так відбувається через те, що теорія створення прогнозів вимагає повної картини щодо стану атмосфери на великій площі, а обробка такої кількості даних вимагає великих потужностей, через що усі обчислення завжди відбуваються на боці серверу. Але з практичної точки зору далеко не завжди є можливість мати дані про погодне оточення. В таких випадках є доречним розробити власний метод прогнозування погодних умов, проте який буде базуватися лише на доступних даних.

Здебільшого такі варіанти будуть мати статистичне коріння в своїй реалізації, що не є дуже хорошою річчю у випадку з прогнозуванням погоди, проте якщо врахувати особливості предметної області, то розроблені методики матимуть окрім статистичного, ще й логічне підґрунтя (тобто вони будуть мати під собою логіку природного, або навіть усього оточуючого світу. Статистичні методи передбачення, що могли б використовуватися для цього проекту:

1. Побудова лінійної регресії [2].

Цей метод є доволі поширеним у використанні, наприклад, для вирішення задач класифікації та створенні певних прогнозів, що є лінійно змінюваними. Головний недолік цього методу – це те, що він є ефективним лише з лінійно залежними даними, з нелінійними прогнози майже не будуть збуватися.

2. Побудова поліноміальної регресії [3]

Також відомий спосіб побудови лінії тренду для поданих даних. Головний плюс – це ефективність там, де звичайна лінійна регресія не допомагає. Проте для

випадку з передбаченням погоди поліноміальний спосіб побудови залежності не показує достатню ефективність.

3. Розробка нейромережі [4]

Один з найсучасніших методів розробки прогнозів, що розвивається з кожним днем неймовірними кроками. Через свою гнучкість є дуже корисним методом і використовується у великій кількості областей, починаючи від обробки зображень до розпізнавання голосових повідомлень. Головний мінус цього методу – потреба у великій потужності від машини, що обчислює поведінку нейромережі.

1.3 Постановка задачі

Для даного проекту потрібно сконструювати пристрій, що матиме можливості невеликого комп’ютера в собі з такими обчислювальними можливостями, що них буде достатньо для розрахунку потрібних прогнозів за допомогою нейронної мережі. Для процесу вибору компонентів збірки та її архітектури треба підходити максимально обережно та розумно (щоб не було зайвих витрат на компоненти, але й пристрій повністю би справлявся зі своїми задачами).

Також, треба розробити власну нейронну мережу на основі теоретичних відомостей щодо створення нейронних мереж та щодо того, як влаштовані зміни погодних умов у навколишньому середовищі.

2. Дослідження способів розробки автономного пристрою-міні-комп’ютеру для подальшого застосування як портативної метеостанції

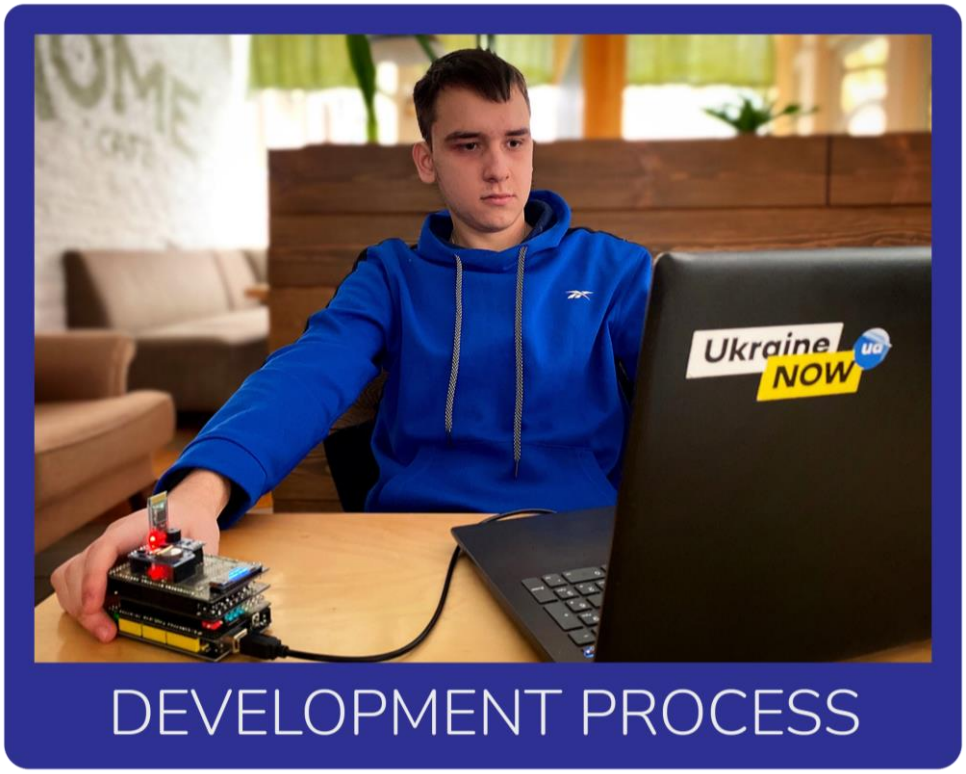


Рис. 2.1 Процес розробки

2.1 Центральний процесор – мозок системи

Перш за все нам потрібно було виділити основні компоненти для керування процесом отримання, зберігання та обробки даних. Для підвищення автономності ми вирішили обрати енергоефективний 8-бітний контролер. І зупинилися на мікроконтролері ATmega2560 [5] компанії Atmel на архітектурі AVR з тактовою частотою 16 МГц та 1 МГц в режимі енергозбереження. Температурні норми були підібрані для вуличних умов від -30 до +30, даний контролер працює в діапазоні від -40 до 85 згідно документації Atmel.

Мікроконтролер обладнаний 54 цифровими виходами, 15 з них підтримують технологію ШИМ, яку ми використовуємо для регулювання напруги.

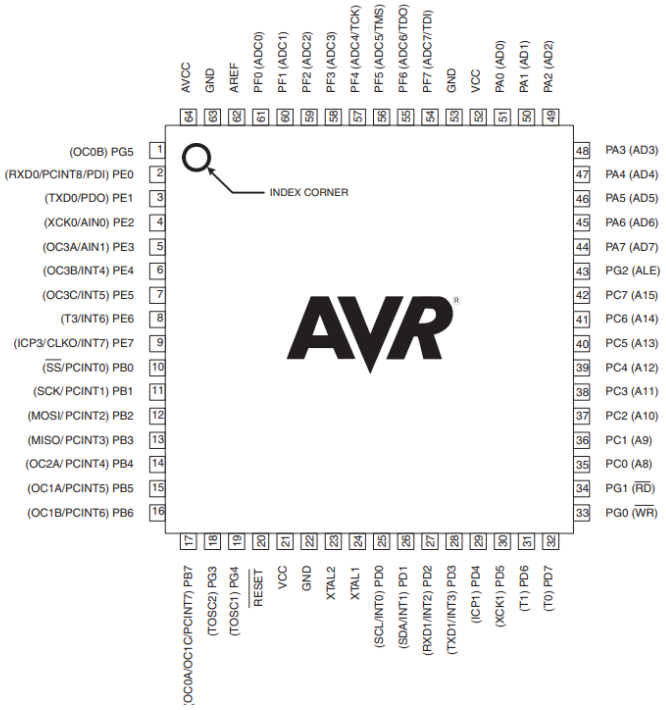


Рис. 2.2 Схема мікроконтролера

Ми вибрали два основні інтерфейси обміну даних між сенсорами та контролером: I²C, SPI.

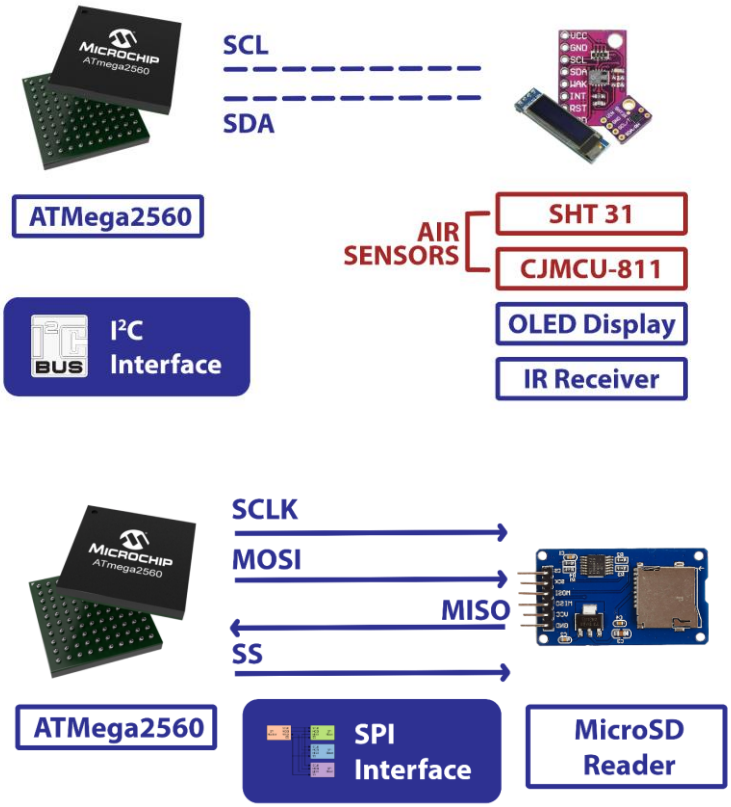


Рис. 2.3 I²C та SPI Інтерфейси в проекті

I²C [6] — послідовна шина даних для зв'язку інтегральних схем, розроблена фірмою Philips на початку 1980-х як проста шина внутрішнього зв'язку для створення керуючої електроніки.

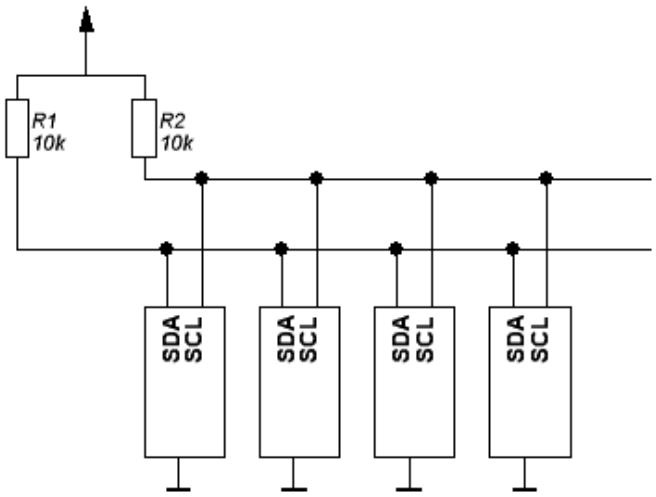


Рис. 2.4 Схема I²C Інтерфейсу

Переваги I²C:

• необхідний всього один мікроконтролер для керування набором пристроїв;

• використовується всього дві лінії введення-виведення загального призначення;

• стандарт передбачає «гаряче» підключення та відключення пристроїв в процесі роботи системи

• вбудований в мікросхеми фільтр придушує сплески, забезпечуючи цілісність даних.

Serial Peripheral Interface [7] — фактичний послідовний синхронний повно дуплексний стандарт передачі даних, розроблений фірмою Motorola для забезпечення простого сполучення мікроконтролерів та периферії. SPI також називають чотирьох-провідним інтерфейсом.

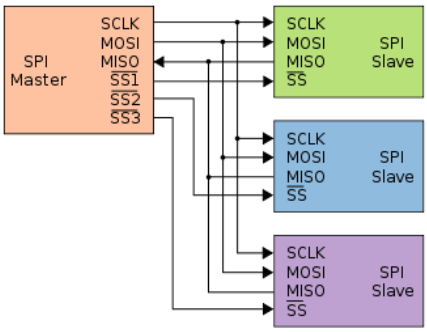


Рис. 2.5 Схема SPI Інтерфейсу

2.2 Допоміжний процесор – USB-Контроль

Для кращого розподілення навантаження на ЦП для прийому/передачі даних по USB інтерфейсу використовується мікроконтролер ATmega16U2 [8].

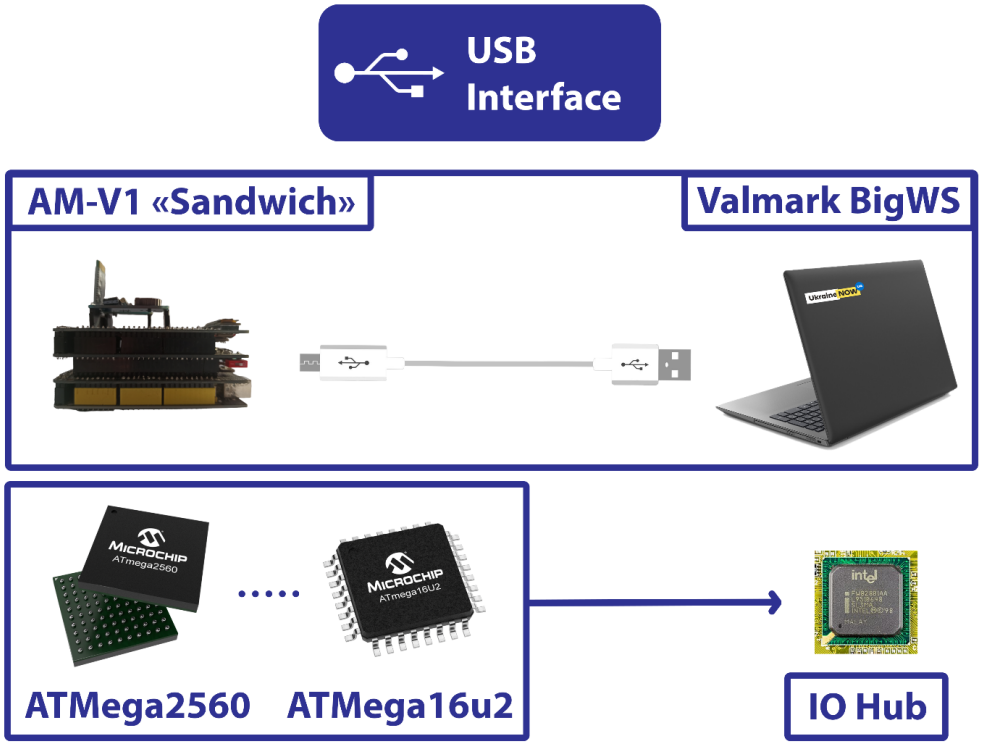


Рис. 2.6 USB Інтерфейс у проекті

2.3 Отримання та збереження даних

Головне завдання пристрою – отримання та обробка атмосферних даних. Для збору даних використовується апаратний комплекс CJMCU-811, що забезпечує нам збір температури та вологості повітря, атмосферного тиску, концентрації CO2, а також таких речовин:

• Ефіри (Бутилгликоль - використовується в авіа та космічної промисловості для очищення поверхні) - дуже висока чутливість.

• Перманентний маркер - дуже висока чутливість.

• Спирти - дуже висока чутливість.

• Толуол - дуже висока чутливість.

• Ацетон - дуже висока чутливість.

• Бутанол - відчуває

• Бутилацетат - відчуває

• Бутан - виявляє, але слабо

• Хлороформ - практично не відчуває

• Ацетальдегід - слабо відчуває

Отримані дані зберігаються до бази даних (СУБД - SQLite) на microSD карті у зашифрованому вигляді.

2.4 Керування даними та пристроєм

Для керування всіма процесами було прийнято рішення створити Веб-програму з можливістю запуску на трьох операційних системах (Windows, Linux, Mac OS).

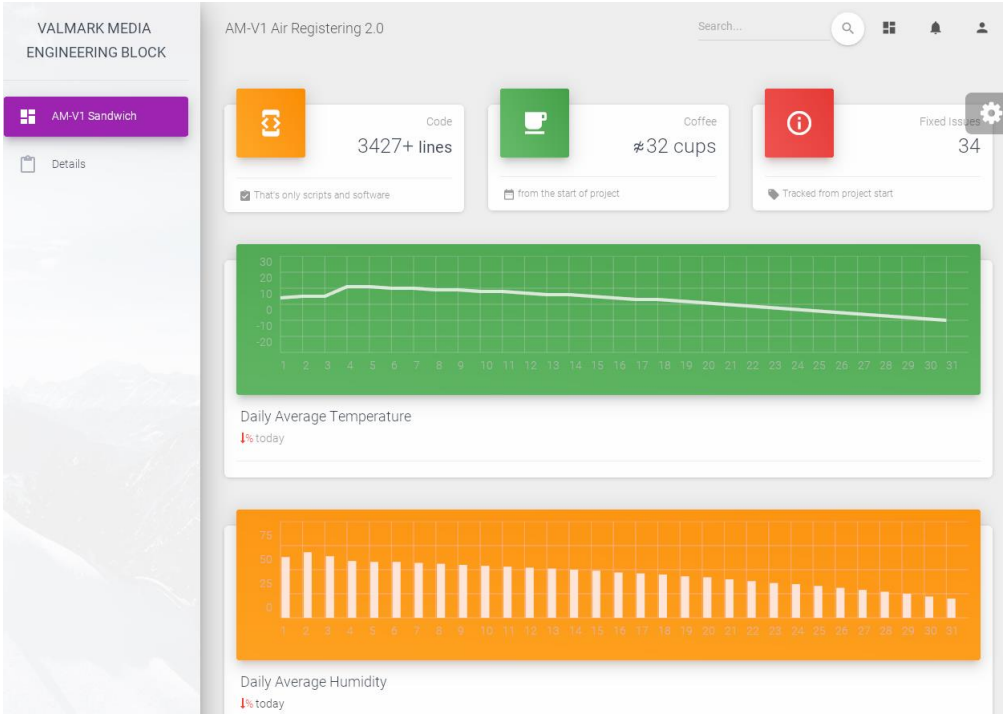


Рис. 2.7 Веб-інтерфейс створеної метеостанції

Веб-інтерфейс (Додаток В.1) побудований на Ultralight SDK, розроблений для кращого використання ресурсів комп’ютера, із серверною мовою розробки C++. Ми розробили зручний для користувача інтерфейс, використовуючи Material Design, що був розроблений компанією Google, що став найкращим стилем 2020 року. На серверній стороні розроблено два потоки: Головний та Прослуховуючий, поки програма працює комп’ютер підтримує постійний зв’язок із зовнішнім пристроєм. Для підтримки та захищення з’єднання використовується наша пропрієтарна система Valmark Bridge. Отриманий пакет даних програма розподіляє в словники та будує графіки по категоріям (Додаток В.2). Пакет даних додатково несе в собі сервісну інформацію про статус елементів пристрою, ця інформація передається до окремого блоку (Додаток В.3). Переглянути всі дані з бази мікрокомп’ютера можливо у вкладці “Деталі” (Додаток В.3).

2.5 Безпека – понад усе

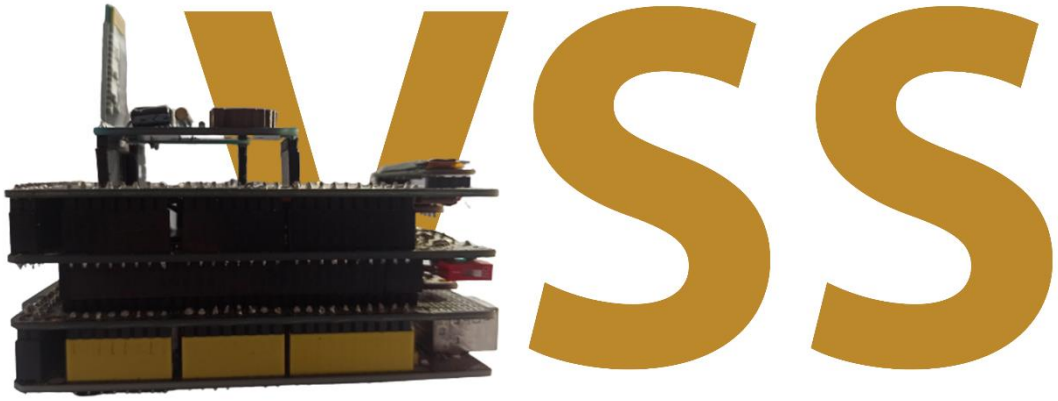


Рис. 2.8 Valmark Secure Service (Компонент безпеки)

Ми дбаємо про безпеку даних, з якими працюємо. Тому для забезпечення цього створили окремий компонент Valmark Secure Service та інтегрували політику обробки даних перед відправкою у системі Valmark Bridge.

Шифрування

RSA (абревіатура від прізвищ Rivest, Shamir та Adleman) — криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел.

Алгоритм RSA складається з 4 етапів: генерації ключів, шифрування, розшифрування та розповсюдження ключів.

Безпека алгоритму RSA побудована на принципі складності факторизації цілих чисел. Алгоритм використовує два ключі — відкритий (public) і секретний (private), разом відкритий і відповідний йому секретний ключі утворюють пари ключів (keypair). Відкритий ключ не потрібно зберігати в таємниці, він використовується для шифрування даних. Якщо повідомлення було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним секретним ключем.

Генерація ключів:

Для того, щоб згенерувати пари ключів виконуються такі дії:

1) Вибираються два великі прості числа p і q 512 біт завдовжки кожне.

2) Обчислюється їх добуток n=pq.

3) Обчислюється функція Ейлера $\varphi(n) = (p-1)(q-1)$

4) Вибирається ціле число e таке, що $1 < e < \varphi(n)$ та e взаємно просте з $\varphi(n)$

5) За допомогою розширеного алгоритму Евкліда знаходиться число d таке, що $ed \equiv 1 \left(mod\ \varphi(n)\right)$

Число n називається модулем, а числа e і d — відкритою й секретною експонентами (англ. encryption and decryption exponents), відповідно. Пари чисел (n, e) є відкритою частиною ключа, а (n, d) — секретною. Числа $p\ i\ q$ після генерації пари ключів можуть бути знищені, але в жодному разі не повинні бути розкриті.

Швидкість роботи алгоритму RSA

Як при шифруванні і розшифруванні, так і при створенні і перевірці підпису, алгоритм RSA у своїй сутності складається з піднесення в степінь, яке виконується як ряд множень.

У практичних додатках для відкритого ключа зазвичай обирається відносно невеликий показник, а часто групи користувачів використовують один і той же відкритий показник, але кожен з різним модулем. Якщо відкритий показник незмінний, вводяться деякі обмеження на головні співмножники (фактори) модуля. При цьому шифрування даних йде швидше, ніж розшифрування, а перевірка підпису — швидше, ніж підписання.

Якщо k — кількість бітів у модулі, то зазвичай в алгоритмах, що використовуються для RSA, кількість кроків, необхідна для виконання операції з відкритим ключем, пропорційна другій степені k, кількість кроків для операцій секретного ключа — третій степені k, кількість кроків для операції створення ключів — четвертій степені k.

Методи «швидкого множення» — наприклад, методи засновані на швидкому перетворенні Фур'є — виконуються меншою кількістю кроків. Проте вони не набули широкого поширення через складність програмного забезпечення, а також тому, що з типовими розмірами ключів вони фактично працюють повільніше. Однак продуктивність та ефективність програм і обладнання, які реалізовують алгоритм RSA, швидко збільшується.

Алгоритм RSA набагато повільніший, ніж DES та інші алгоритми блокового шифрування. Програмна реалізація DES працює швидше принаймні в 100 разів і від 1,000 до 10,000 — в апаратній реалізації (в залежності від конкретного пристрою). Завдяки проведенню розробок, швидкість алгоритму RSA, ймовірно, прискориться, але одночасно прискориться і робота алгоритмів блокового шифрування.

Довжина ключа

Число n повинне мати розмір не менше 512 біт. На 2007 рік система шифрування на основі RSA вважалась надійною, починаючи з величини N в 1024 біти. І ми притримуємось зазначеної величини ключів у 1024 біти.

Наразі, ми інтегрували алгоритм шифрування AES-256, що забезпечує більший рівень безпеки передачі даних.

Advanced Encryption Standard (AES), також відомий під назвою Rijndael — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт), фіналіст конкурсу AES і прийнятий як американський стандарт шифрування урядом США. Вибір припав на AES з розрахуванням на широке використання і активний аналіз алгоритму, як це було із його попередником, DES. Державний інститут стандартів і технологій (англ. National Institute of Standards and Technology, NIST) США опублікував попередню специфікацію AES 26 жовтня 2001 року, після п'ятилітньої підготовки. 26 травня 2002 року AES оголошено стандартом шифрування. Станом на 2009 рік AES є одним із найпоширеніших алгоритмів симетричного шифрування.

В принципі, алгоритм, запропонований Рейменом і Дейцменом, і AES не одне і те ж. Алгоритм Рейндол підтримує широкий діапазон розміру блоку та ключа. AES має фіксовану довжину у 128 біт, а розмір ключа може приймати значення 128, 192 або 256 біт. В той час як Рейндол підтримує розмірність блоку та ключа із кроком 32 біт у діапазоні від 128 до 256. Через фіксований розмір блоку AES оперує із масивом 4×4 байт, що називається станом (версії алгоритму із більшим розміром блоку мають додаткові колонки).

3. Дослідження способів використання нейронних мереж для передбачення погоди

3.1 Властивості нейронних мереж

Нейронна або штучна нейронна мережа – програмна імплементація нейронних структур мозку живої істоти. Тобто програмний код імітує можливості нашого мозку за допомогою певних «нейронів», налаштування яких і є побудовою нейронної мережі. Проте, на відміну від людського мозку, «комп'ютерний мозок» має значно менше нейронів, їхня кількість залежить від задачі поставленої перед дослідниками.

Етапи для побудови нейронної мережі, які б ми виділили:

1. Визначення вхідних даних
2. Визначення даних за якими буде навчатися мережа
3. Визначення функції активації
4. Визначення структури нейронної мережі
5. Розробка тестів та тестування для визначення ефективності мережі
6. Переробка, якщо результати тестів не є позитивними

3.2. Розробка нейронної мережі. Перші етапи розробки

Визначення вхідних даних

Так як погода є доволі комплексним явищем, було вирішено використовувати різні її фактори для прогнозування температури на наступний день. У першій версій нейронної мережі вирушено було додати окрім максимальних та мінімальних показників температури, ще показник відносної вологості, так як він є одним з факторів, що залежить від температури на протязі усього дня та опадів, що також впливають на температуру. А саме такий фактор нам потрібен для збільшення точності прогнозу.

Визначення даних за якими буде навчатися мережа

Так як погода є доволі змінним явищем протягом року, або навіть років, нема сенсу навчати просту мережу на даних за ці роки, тим більш що це доволі сильно сказиться на швидкості навчання. Тому, як альтернатива, було вирішено використовувати дані лише за останні 8 днів (враховуючи поточний). Тобто,

для розробки тестів дані за останні дні розбиваються таким чином: вхідні дані – відносна вологість, максимальна та мінімальна температури за певний день, вихідні дані – середня температура за наступний день.

Визначення функції активації

Для побудови нейронної мережі треба мати певний алгоритм, за яким вона буде навчатися та власне тестові дані для цього процесу. Алгоритм має дві основні складові: функція активації та спосіб навчання. Визначення функції активації ставить певні умови на спосіб навчання, тому ці дві речі поєднані разом. Одна з найпопулярніших функцій активації нейромереж – функція сігмоїди або логістичної кривої. Ця функція має таку формулу:

$$f(x) = \frac{1}{1+e^{-x}} \qquad (1)$$

А її графік виглядає так:

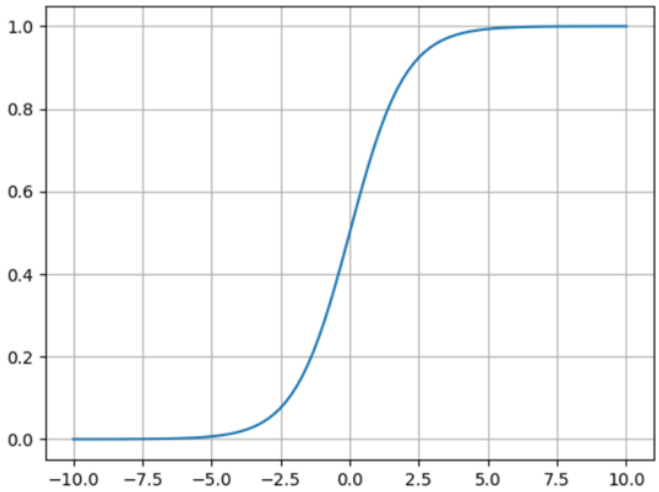


Рис. 3.1 Графік функції сігмоїди

Ця функція не без недоліків, один з них – це те, що вона має властивість прижиматися до країв свого діапазону при занадто великих значеннях вхідних даних. Проте, завдяки тому, що нейронна мережа буде постійно, а саме кожен день, за наявності нових даних, перевчатися, цей мінус не є критичним, а тому використання сігмоїди як функції активації є доречним.

За допомогою саме функції активації відбувається навчання нейрону та обчислення значення, його від нього очікують. Яким чином все це відбувається: на початку нейронна мережа має у якості ваг випадкові значення, вхідні дані

(тренувальний сет) нормалізуються для того, щоб діапазон входу відповідав діапазону виходу ([0,1]), потім нейронна мережа за допомогою ваг отримує певне значення, яке потім порівнює з очікуваним, далі за допомогою градієнту функції активації вираховується значення, на яке треба скорегувати ваги. І так робиться стільки разів, скільки запланує програміст (в нашому випадку це 10000 разів).

Визначення структури нейронної мережі

Для того, щоб на пристрої нейронна мережа могла працювати з прийнятною швидкістю, було вирішено занадто сильно не ускладнювати нейронну мережу і по суті зробити лише один нейрон. А у випадку, якщо така структура нейронної мережі буде причиною видачі поганих результатів прогнозу, то буде вже вирішено її доробити та ускладнити, проте, у розумних масштабах.

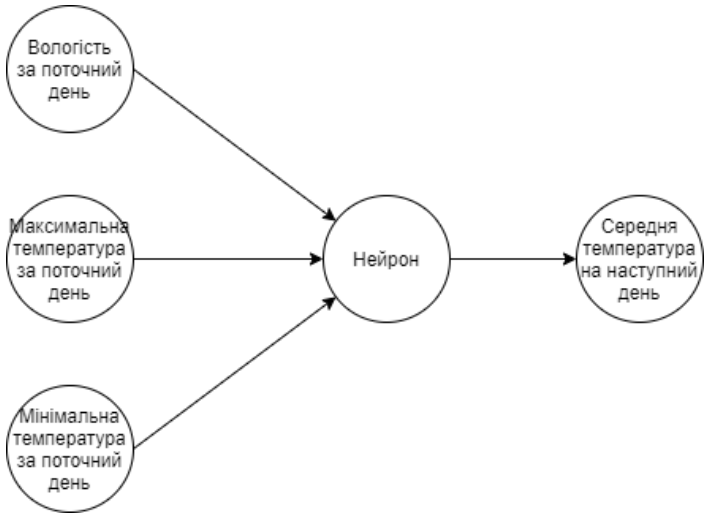


Рис. 3.2 Структура нейронної мережі

Також на цьому етапі варто визначити алгоритм навчання мережі. Так як структура є максимально простою, то й алгоритм навчання буде максимально простим, за допомогою градієнту (похідної) функції активації.

$$f`(x) = f(x) * (f(x) - 1), \text{ де} \quad (2)$$

f(x) – це фунція сігмоїди

Градієнт (похідна) сігмоїди

$$k_i = error * input[i] * f`(input[i]), \text{ де} \quad (3)$$

k_i – це значення, яке треба додати до і-тої ваги,

error – різниця між значенням мережі та очікуваним значенням,

input[i] – і-те значення вхідних даних,

f`(input[i]) – градієнт сігмоїди від і-того значення вхідних даних

На кожній ітерації циклу навчання буде визначатися похибка мережі та за допомогою та за допомогою формули для визначення значень, які треба буде додати для поточних ваг для їхньої кореляції (3). Код навчання нейронної мережі наведений в додатку (Д. Б.1).

3.3 Розробка нейронної мережі. Розробка та проведення тестів

Структура тестового кейсу була визначена доволі легко та є сама по собі простою:

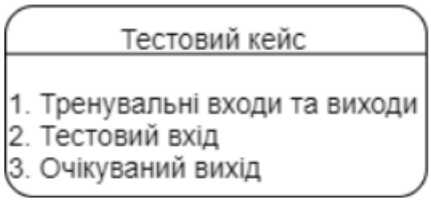


Рис. 3.3 Структура тесту

Яким чином були розроблені тести:

1. Були взяті дані про погоду за певний місяць (наприклад, для першого тесту – дані за січень)

2. Дані були оброблені таким чином: кожна сімка днів використовується як множина даних для навчання, коли як восьмий день – це «поточний день тесту», а середня температура дев’ятого дня – очікуваний результат нейронної мережі.

3. За допомогою циклу з кожного тесту береться множина тренувальних входів та виходів нормалізуються, та передаються до нейромережі. З даних тесту беруть дані для входу поточного тесту, та передаються до нейронної мережі для отримання передбачення. Це передбачення потім порівнюється з тим, що було у реальності, та робляться висновки.

4. Якщо на певному тесті ризниця температур за модулем більша ніж 3 градуси, то тест вважається не пройденим, якщо менше – то навпаки.

Код, що використався для створення тестів наведений в додатку (Д Б.2).

Для побудови тестів було вирішено використати дані за різні місяці 2020 року. Дані для тестів були узяті з доволі відомого сайту з архівом погоди багатьох країн та міст [9]. Перший тест був проведений за даними за січень 2020 року.

Рис. 3.4 Графік порівняння роботи нейромережі з реальними значеннями за січень 2020 року

Зазначимо, що так як дані бралися лише за січень, тому перше передбачення було зроблене на 8 січня, тоді як попередні дані за 1-7 січня можуть бути використані лише для тесту. Тобто 1 на графіку (Рис. 3.4) – це не 1 січня, а перший тест, 2 – другий тест і так далі. Як видно на графіку (Рис. 3.4), ефективність передбачення доволі чутлива до різких підвищень або зменшень температури, проте в цілому, якщо вважати, що вгаданий прогноз – це коли очікувана температура відрізняється від реальної менше ніж на 3 градуси за модулем, то нейромережа показала доволі приємну ефективність не вважаючи на її простоту (19/23 випадків, або 82,6% точності).

Відмічу, що так як погода доволі часто зберігає тенденцію у своїй зміні, а якщо різко змінюється, то зазвичай не більш одного разу на декілька днів. Гірший випадок для даної нейромережі – це коли температура різко змінюється кожен день, на щастя такого в Україні майже не буває, проте, якщо адаптувати мережу для умов інших країн, то її варто буде сильніше ускладнити, навіть, якщо прийдеться зазначити втрат у швидкості обчислень та часу роботи пристрою.

Були проведені тести з іншими місяцями, та отримані такі результати:

Рис. 3.5 Графік тесту за червень 2020

Рис. 3.6 Графік тесту за травень 2020

У червні, завдяки доволі стабільної погоди, та відсутності різких змін, результати мережі навіть кращі ніж були у січні того ж року – 20/22 вгаданих прогнозів, що складає приблизно 91% точності, що є дуже вдалим результатом.

А от у травні, як був взятий як раз як місяць з доволі нестабільною погодою нейромережа дала найгірший результат – 18/23 вгаданих прогнозів, або 78% точності (що не є сильно поганим результатом, проте виділяється у негативному плані порівняно з іншими).

Для візуалізації результативності мережі були зроблені кругові діаграми для кожного тесту:

Рис. 3.7 Кругові діаграми результативності тестів нейромережі

Середнє значення похибок та квадратичних відхилень разом із результатами тестів наведені у додатках (Д. А.1 – Д. А.3). У більш стислому форматі результати тестів можна виразити у такій таблиці:

Таблиця 3.1

Результати тестів нейромережі

Місяць	Січень	Червень	Травень
Відсоток вдалих прогнозів	82,60869565	90,90909091	78,26086957
Середня похибка	1,669316139	1,5035	1,665615217
Стандартне відхилення похибки	0,941783536	0,994472727	1,121753308

3.4 Розробка нейронної мережі. Порівняння з лінійною регресією

Побудова лінійної регресії методом найменших квадратів [10] є одним з доволі відомих та надзвичайно простих методів прогнозування та виявлення залежностей в багатьох сферах, пов'язаних з бізнесом та економікою. Проте, лінійну прогресію можна використати і в нашій сфері прогнозування погоди.

$$y = a * x + b \qquad (4)$$

Для того, щоб побудувати лінійну регресію такого вигляду, як вона записана формулою (4), єдине що нам потрібне – набір даних, а саме набір залежних змінних та відповідних незалежних змінних.

$$a = \frac{N\sum_i x_i y_i - \sum_i x_i \sum_i y_i}{N\sum_i x_i^2 - (\sum_i x_i)^2}, b = \frac{1}{N}\sum_i y_i - \frac{a}{N}\sum_i x_i \qquad (5)$$

Формула 5. Знаходження коефіцієнтів лінійної регресії

Лінійна регресія – це завжди графік і тому треба визначитися з тим, яка змінна будуть незалежними, а яка залежною. По аналогії з нейромережею, незалежною змінною буде середня температура в попередній день, а незалежною

– температура у наступний день. Робитися лінійна прогресія буде також на основі попередніх 7 днів. Множина даних, за допомогою яких розраховуються коефіцієнти – це середні температури попередніх 7 днів, відносно «поточного» для тесту. За допомогою середньої температури «поточного» дня тесту обчислюється середня температура наступного дня, використовуючи отриманні коефіцієнти на основі тестових даних. Тобто, прогноз робиться лише на один день уперед, як це було і в нейронній мережі.

Рис. 3.8. Тест лінійної регресії за січень 2020

На графіку (Рис. 3.8) видно, що лінійна регресія ще більш чутлива до різких змін температури. По виду графіка може скластися враження, що лінійна регресія немов трохи відстає за часом від актуальних значень – це пов'язано взагалі з тим, як саме працює лінійна регресія, а саме з тим, що вона дуже чутлива до певних тенденцій та через це видає передбачення доволі схожі на попередні, що використовувалися при визначенні коефіцієнтів. Точність лінійної регресії склала 18/23 вгаданих прогнозів або 78,3%, тоді як точність мережі складає 82,6%. Виграш у точності, використовуючи нейронну мережу, складає 4,3%, якщо порівняти середню похибку прогнозів, то різниця приблизно така ж – нейронна мережа має похибку, що на 3,4 % менша ніж в лінійної регресії. В цілому, різниця не сильно критична, проте, нейронна мережа дає можливість експериментувати та покращувати її, коли як лінійна регресія доволі швидко набуває свого максимального потенціалу, і через це вже не можна буде

покращити її результати. Порівняння двох способів передбачень наведено у таблиці:

Таблиця 3.2

Результати порівняння лінійної регресії та нейромережі

	Нейромережа	Лінійна регресія	Різність показників відносно мережі, в умовних одиницях	Різність показників відносно мережі, в відсотках
Кількість вгаданих прогнозів	19	18	1	4,347826087
Середня похибка	1,669316139	1,727439522	-0,058123383	-3,364713026
Стандартне відхилення похибки	0,941783536	0,97995276	-0,038169223	-3,895006474

3.5 Розробка нейронної мережі. Висновки

Як висновок щодо роботи з прогнозуванням погоди за допомогою нейромереж можу сказати, що результат є доволі вдалим, та точність прогнозів дозволяє використовувати цю нейронну мережу на практиці вже зараз. Також, завдяки доволі простій структурі мережі, інтегрування цієї мережі у пристрій не є неможливою річчю. У порівнянні з лінійною регресією, мережа дає більш точні результати, та показує більшу стабільність до різких перепадів температури, а головний її плюс – це можливість покращувати її, на відміну від регресії.

4. Зменшення об’єму даних, обмін якими відбувається між міні-комп’ютером та комп’ютером користувача

Одна з проблем, з якою довелося зіштовхнутися протягом розробки та тестування міні-комп’ютеру – це великий час передачі даних, що збираються протягом дня та певна похибка через погодні умови (тобто, наприклад, на температуру може вплинути різкий вітер, дощ, тощо) при записі даних. Обидві ці проблеми можна вирішити за допомогою одного з найпростіших та доволі ефективних методів згладжування – рухомого середнього. За допомогою цього методу можна отримати більш “чистий” та зрозуміліший графік або набір значень, які можна далі використовувати для тренування нейромережі та для відображення користувачу.

4.1 Зменшення об’єму даних для обміну. Рухоме середнє

Рухоме середнє [11] – один з найвідоміших методів згладжування даних, що має велику кількість сфер, в яких його можна застосувати:

1. В статистиці та економіці для згладжування числових, часових рядів.
2. В техніці, для обробки різноманітних сигналів, для аналізу систем
3. У якості індикатору в технічному аналізі

Цей метод можна застосувати і в нашому випадку. Як було зазначено вище, наш пристрій повинен мати інформацію про середню температуру, що була протягом дня. Також, він повинен передавати інформацію про температуру протягом дня до комп’ютеру для відображення вже в веб-інтерфейсі користувача.

Власне як працює алгоритм рухомого середнього:

1. Визначається певний період часу, на які буде ділитися потім графік. Наприклад, в нашому випадку – це 1 година, що буде охоплювати 4 записані значення в пам’яті пристрою.

2. Для кожний чотирьох значень (4, бо наш пристрій робить запис кожні 15 хвилин) шукається середнє арифметичне, і це буде наша нова точка, відповідає одній годині.

$$a_t = \frac{1}{n}\sum_{i=0}^{n-1} \quad p_{t-i}, \text{ де} \qquad (5)$$

a_t – t-те значення рухомого середнього,

n – кількість елементів для згладжування, яку ми обрали,

p_{t-i} – t-i-тий елемент початкової множини

Використовуючи формулу вище (5) можна порахувати рухоме середнє для будь –якого елементу. Трохи додам, що для кожного r елементу (r<n) елементів треба використовувати перші r елементів.

4.2 Зменшення об’єму даних для обміну. Тестування рухомого середнього

Рис. 4.1. Початкові дані, що зібрав міні-комп'ютер за 1 січня 2021 року

Як видно вище (Рис. 4.1), дані представлені у доволі незручному для читання вигляді, більш того, тут наявні дані за кожні 15 хвилин, що просто занадто багато для користувача. Обробивши дані методом рухомого середнього графік став виглядати так:

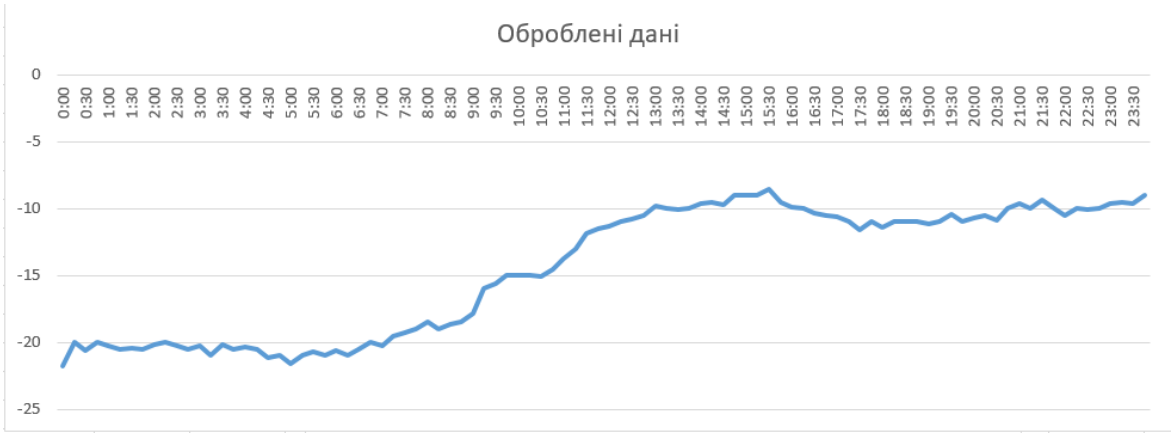


Рис. 4.2. Дані про температуру оброблені рухомим середнім

Як видно на графіку (Рис. 4.2), рухоме середнє доволі добре згладило увесь графік, на ньому тепер майже немає різких перепадів температури між двома

сусідніми значеннями. Проте, нам потрібні лише дані у точках, що відповідають певній годині, тому прибравши непотрібні значення можна отримати таку картину:

Рис. 4.3 Дані про поточні температуру за кожну годину після обробки рухомим середнім

Графік (Рис. 4.3) демонструє інформацію про зміну температури протягом дня у значно зручнішому вигляді для користувача. Ще один плюс такого вигляду даних – вони тепер займають у 4 рази менше місця, завдяки чому обмін даними між комп’ютером та нашим приладом займатиме значно менше часу.

5. Висновок

1. Був створений автономний пристрій, що дозволяє збирати та аналізувати дані про погодні умови та їхнього аналізу. Була створена нейронна мережа, що доволі результативно може прогнозувати погоду на наступний день.

2. Була створена доволі проста нейронна мережа (достатньо для того, щоб її можна було інтегрувати в міні-комп’ютер), що може прогнозувати середню температуру на наступний день з точністю близько 85%, що було доведено шляхом тестів. Також вона була зрівняна з іншою математичною моделлю побудови прогнозів — лінійна регресія.

3. Було створено інновацій пристрій для збору та аналізу даних про погодні умови, що може працювати автономно. Завдяки цьому пристрою будь-яка людина може отримати доволі точні прогнози середньої температури на наступний день.

4. Завдяки побудові нейронної мережі, точність прогнозів порівняно з лінійною регресією виросла на 4%, при тому середня похибка та її дисперсія є меншими трохи більше ніж на 3 %. І головна особливість цього алгоритму — це безмежний простір для його покращення, чого не можна сказати про лінійну регресію.

5. Продукт вже було використано в домашніх та вуличних умовах. В цілому, усі задачі, що були поставлені при розробці цього приладу, міні-комп’ютер виконує з очікуваними результатами.

6. Зараз продукт знаходиться у повністю працездатному стані. Але напрямками подальших досліджень та дороблень можуть бути такі:

- Ускладнення нейронної мережі для отримання більш точних результатів прогнозів, але у розумних масштабах, щоб міні-комп’ютер справлявся з навантаженням.

- Покращення алгоритму зберігання та обробки інформації

- Інтеграція криптографічного модулю

- Покращення алгоритму прийому/передачі пакетів між AM-V1 та комп’ютером

Література

1. Стаття про те, як робляться прогнози погоди – [Електронний ресурс]. Режим доступу: https://vodnyimir.ru/Na_kuhne_pogody.html

2. Стаття про властивості та побудову лінійної регресії – [Електронний ресурс]. Режим доступу: https://pidru4niki.com/15800119/statistika/regresiya

3. Стаття про властивості та побудову поліноміальної регресії – [Електронний ресурс]. Режим доступу: https://studfile.net/preview/5199989/page:6/

4. Стаття про властивості нейромереж – [Електронний ресурс]. Режим доступу: http://apeps.kpi.ua/neural-networks/en

5. Документація контролера ATmega2560 – [Електронний ресурс]. Режим доступу: http://wiki.amperka.ru/_media/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D1%8B:arduino-mega-2560:atmega2560_datasheet.pdf

6. Інформація про I^2C – [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/I%C2%B2C

7. Інформація про SPI – [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Serial_Peripheral_Interface

8. Документація мікроконтролера ATmega16U2 – [Електронний ресурс]. Режим доступу: http://wiki.amperka.ru/_media/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D1%8B:arduino-mega-2560:atmega16u2_datasheet.pdf

9. Ресурс, що містить інформацію про прогнози погоди – [Електронний ресурс]. Режим доступу: https://habr.com/ru/post/312450/https://www.wunderground.com/history

10. Інформація про метод найменших квадратів https://uk.wikipedia.org/wiki/Метод_найменших_квадратів

11. Опис рухомого середнього – [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Рухоме_середнє

Додатки

Додаток А — тести нейронної мережі

Тести за січень						
Результати нейронної мережі	Актуальні значення	Чи є прогноз вгаданим коректно	delta		Вгадані прогнози	19
-1,80	0,60	1	2,39996		Невгадані пронози	4
-0,76	1,30	1	2,059563			
0,22	4,30	0	4,075038		Середня похибка	1,669316
1,34	0,50	1	0,83871		середнє відхилення	0,941784
-1,02	2,20	0	3,21744			
0,26	1,50	1	1,241181			
1,22	1,10	1	0,11805			
1,54	0,70	1	0,83884			
1,76	-0,40	1	2,16139			
1,75	-1,60	0	3,35151			
-1,56	-2,00	1	0,4389			
-0,74	-0,40	1	0,338107			
0,63	2,80	1	2,172569			
5,51	2,30	0	3,21024			
1,88	0,70	1	1,17842			
2,01	1,00	1	1,01473			
1,68	1,00	1	0,68196			
1,22	1,80	1	0,58439			
1,36	-0,90	1	2,26453			
0,39	-1,10	1	1,489665			
-2,15	0,30	1	2,45341			
-0,03	1,80	1	1,8273382			
1,26	1,70	1	0,43833			

Д. А.1 Таблиця результатів тестів нейромережі за січень 2020

Тест за червень 2020						
Результати нейронної мережі	Актуальні значення	Чи є прогноз вгаданим коректно	delta		Вгадані прогнози	20
23,0962	25,6	1	2,5038		Невгадані пронози	2
26,3331	26,3	1	0,0331			
25,4294	27,2	1	1,7706		Середня похибка	1,5035
26,6081	27,6	1	0,9919			
26,5887	25,6	1	0,9887			
25,0544	20,6	0	4,4544			
17,144	22,8	0	5,656			
22,6867	23,8	1	1,1133			
24,0461	21,9	1	2,1461			
22,0562	22,4	1	0,3438			
22,2856	24	1	1,7144			
22,68	22,1	1	0,58			
22,2165	24,5	1	2,2835			
22,8839	23,6	1	0,7161			
23,538	22	1	1,538			
23,9039	21,5	1	2,4039			
22,2371	22,3	1	0,0629			
22,4949	21,9	1	0,5949			
21,8869	23,2	1	1,3131			
22,8191	23	1	0,1809			
22,5632	23,4	1	0,8368			
22,7508	21,9	1	0,8508			

Д. А.2 Таблиця результатів тестів нейромережі за червень 2020

Тест за травень 2020						
Результати нейронної мережі	Актуальні значення	Чи є прогноз вгаданим коректно	delta		Вгадані прогнози	18
14,7387	10,5	0	4,2387		Невгадані пронози	5
11,5809	15	0	3,4191			
15,2608	18,5	0	3,2392		Середня похибка	1,665615
16,955	13,5	0	3,455			
13,4458	9,3	0	4,1458			
12,8482	9,9	1	2,9482			
11,3879	9,9	1	1,4879			
10,7528	12	1	1,2472			
12,151	12,3	1	0,149			
11,2506	12,9	1	1,6494			
11,8965	11,4	1	0,4965			
11,4737	13,2	1	1,7263			
12,4796	10,6	1	1,8796			
11,0388	8,2	1	2,8388			
10,9511	9,3	1	1,6511			
10,4492	10,6	1	0,1508			
9,77645	10,7	1	0,92355			
13,0675	12	1	1,0675			
12,9565	12,8	1	0,1565			
12,3683	12,6	1	0,2317			
12,7048	13,7	1	0,9952			
13,3154	13,4	1	0,0846			
13,4275	13,3	1	0,1275			

Д. А.3 Таблиця результатів тестів нейромережі за травень 2020

Додаток Б — програмний код

```cpp
void NeuroNetwork::Train(const std::vector<std::vector<double>>& inputs, const std::vector<double>& outputs, unsigned int numIterations) {
    unsigned int i = 0;

    while (i < numIterations) {


        std::vector<double> errors(outputs.size());
        std::vector<double> networkOuts(outputs.size());

        std::vector<double> adjustments(outputs.size());

        for (size_t j = 0; j < outputs.size(); ++j) {
            networkOuts[j] = Think(inputs[j]);
            errors[j] = networkOuts[j] - outputs[j];

        }

        for (size_t j = 0; j < errors.size(); ++j) {
            for (size_t k = 0; k < inputs[j].size(); ++k) {
                this->weights[k] += errors[j] * inputs[j][k] * this->SigmoidDerivative(networkOuts[j]);
            }
        }

        ++i;
    }
}
```

Д. Б.1 Код з реалізацією алгоритму навчання мережі

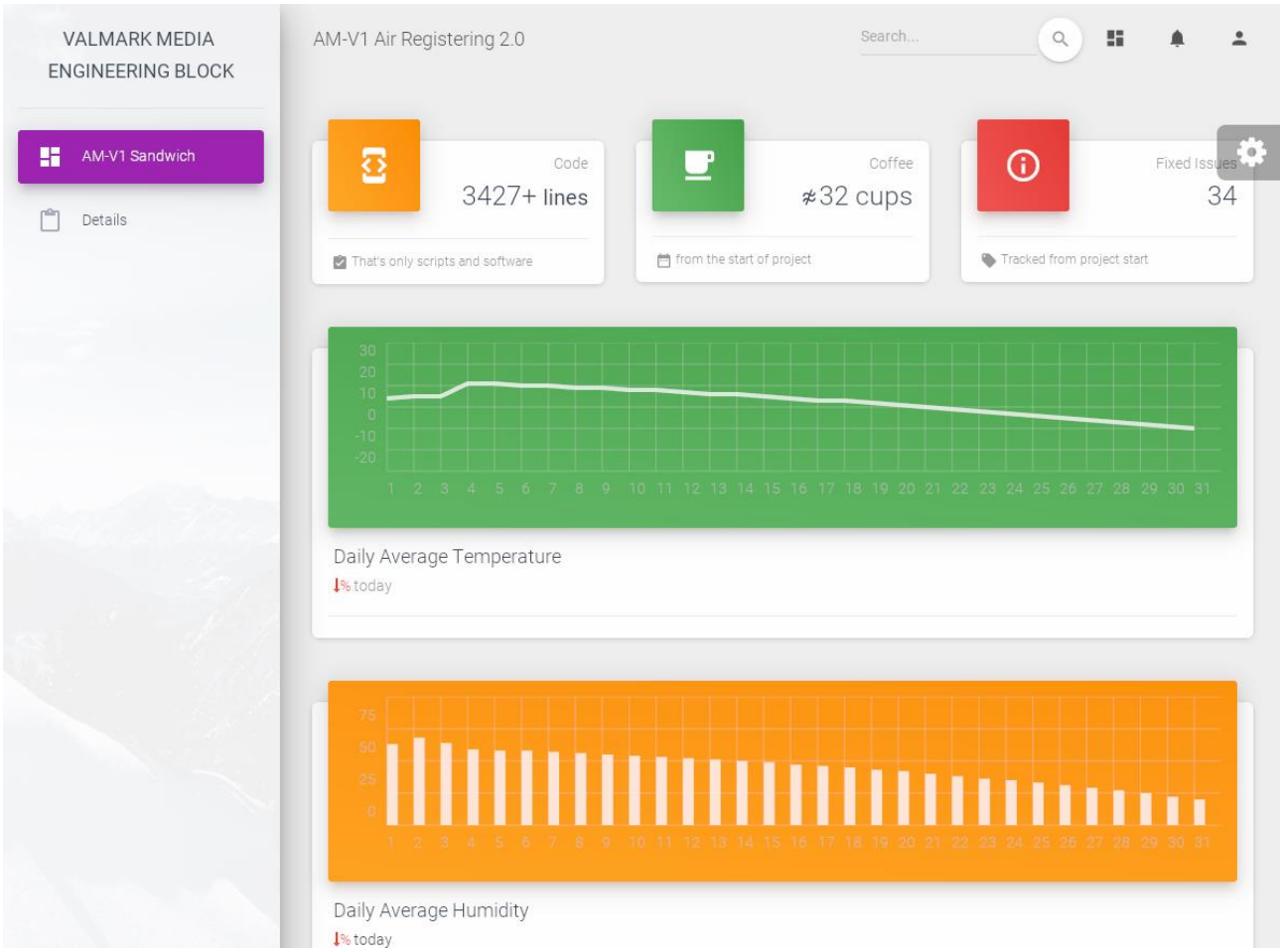
```cpp
struct TestSet {
    std::vector<std::vector<double>> train_set;
    std::vector<double> train_outs;

    std::vector<double> test_in;
    double expected_out;
};

std::vector<TestSet> getTests(std::string path) {
    std::fstream fs(path);
    std::vector<std::vector<double>> all_data;
    std::vector<TestSet> res;

    double min_temp,
        avg_temp,
        max_temp,
        humidity;
    int date;
    while (true) {

        if (!(fs >> date >> max_temp >> avg_temp >> min_temp >> humidity)) {
            break;
        }
        all_data.push_back({ max_temp, avg_temp, min_temp, humidity });
    }
    for (size_t i = 7; i < all_data.size() - 1; ++i) {
        std::vector<std::vector<double>> train_set;
        std::vector<double> train_outs;
        std::vector<double> test_in = { all_data[i][3], all_data[i][2], all_data[i][0]};
        double expected_out = all_data[i + 1][1];

        for (int j = 7; j >= 1; --j) {
            train_set.push_back({ all_data[i - j][3], all_data[i - j][2], all_data[i - j][0] });
            train_outs.push_back(all_data[i - j + 1][1]);
        }

        res.push_back({
            train_set,
            train_outs,
            test_in,
            expected_out
            });
    }
    return res;
}
```

Д. Б.2 Код функції, що створює тести з файлу з даними

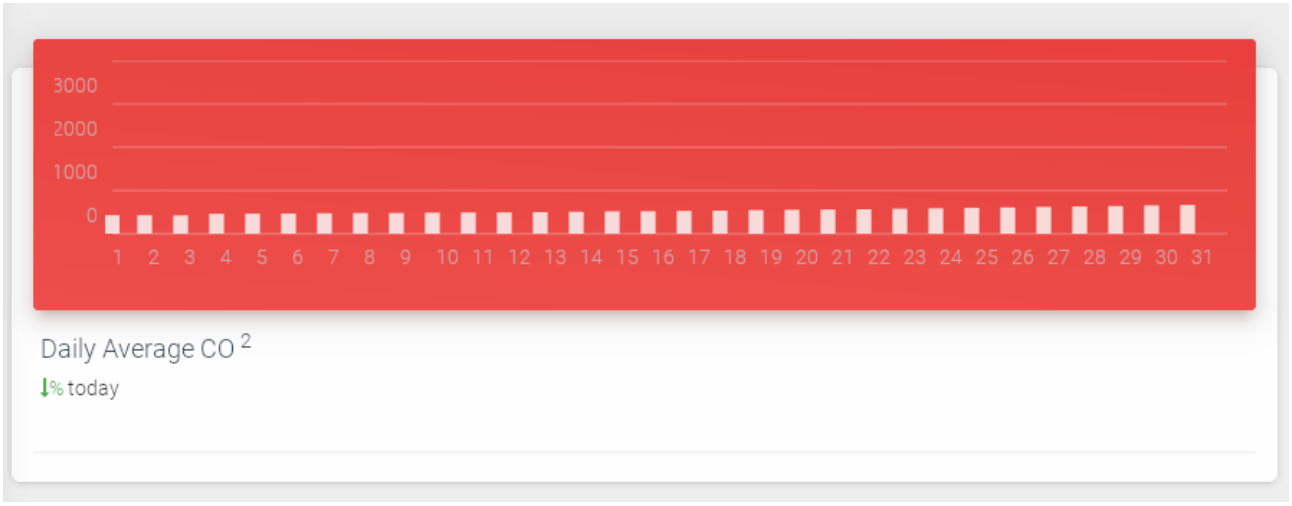
```
void commandHendler(char input[]){
    // Read each command pair
    char* command = strtok(input, "&");
    int commandCount = 0;
    // Get Commands
    while (command != NULL)
    {
        // Execute commands
        float dataArr[10];
        char* data = (char *) malloc(strlen(command)+1); /* +1 for '\0' */;
        strcpy(data, command);

        // Split the command and values
        data = strtok(data, ":");
        for (size_t i = 0; data != NULL; i++)
        {
            if (i != 0) dataArr[i] = atof(data);
            else command = data;
            //Serial.println("Debug: " + String(command) + ":" + String(dataArr[i]) + ":" + String(data));
            // Find the next command in input string
            data = strtok(NULL, ":");
        }



        /*
        Serial.println("Data: " + String(command) + ":" + String(dataArr[0]) + ":" + String(dataArr[1]));
        if (String(command) == "sum")
        {
            Serial.println("Data: " + String(command) + ":" + String(dataArr[0] + dataArr[1]));
        }
        else Serial.println("Wrong command input");*/

        // Find the next command in input string
        command = strtok(NULL, "&");
    }
}
```

Д. Б.3 Обробник команд AM-V1 “Sandwich”

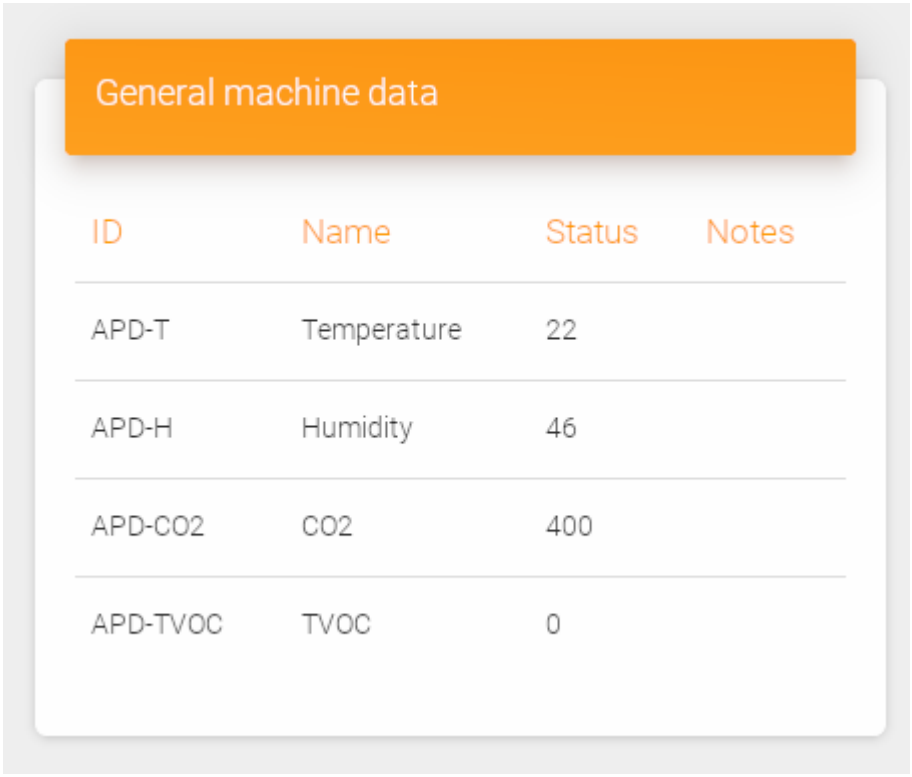
Додаток В — Знімки продукту



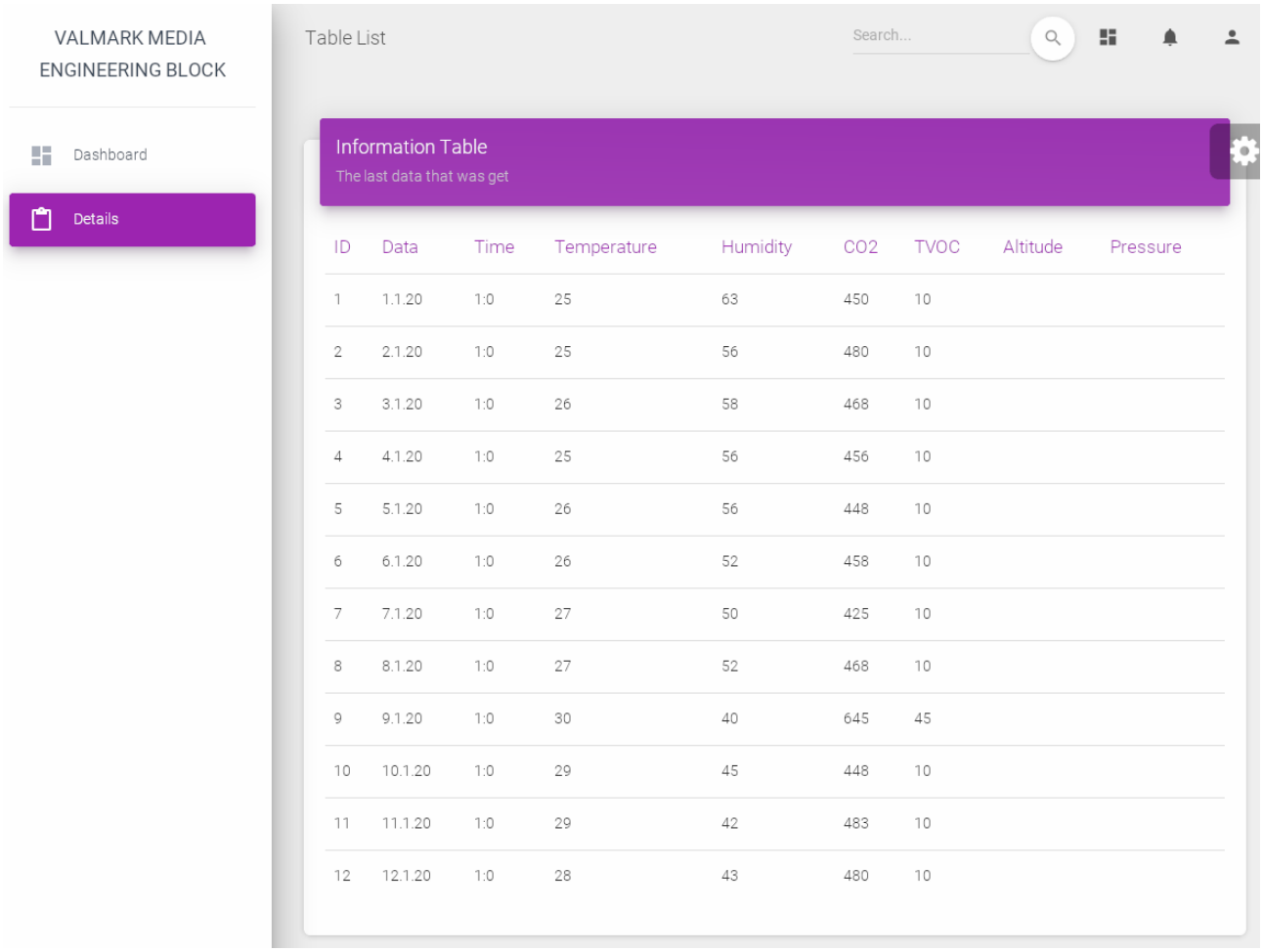
Д. В.1 Скріншот інтерфейсу програми



Д. В.2 Скріншот інтерфейсу програми.

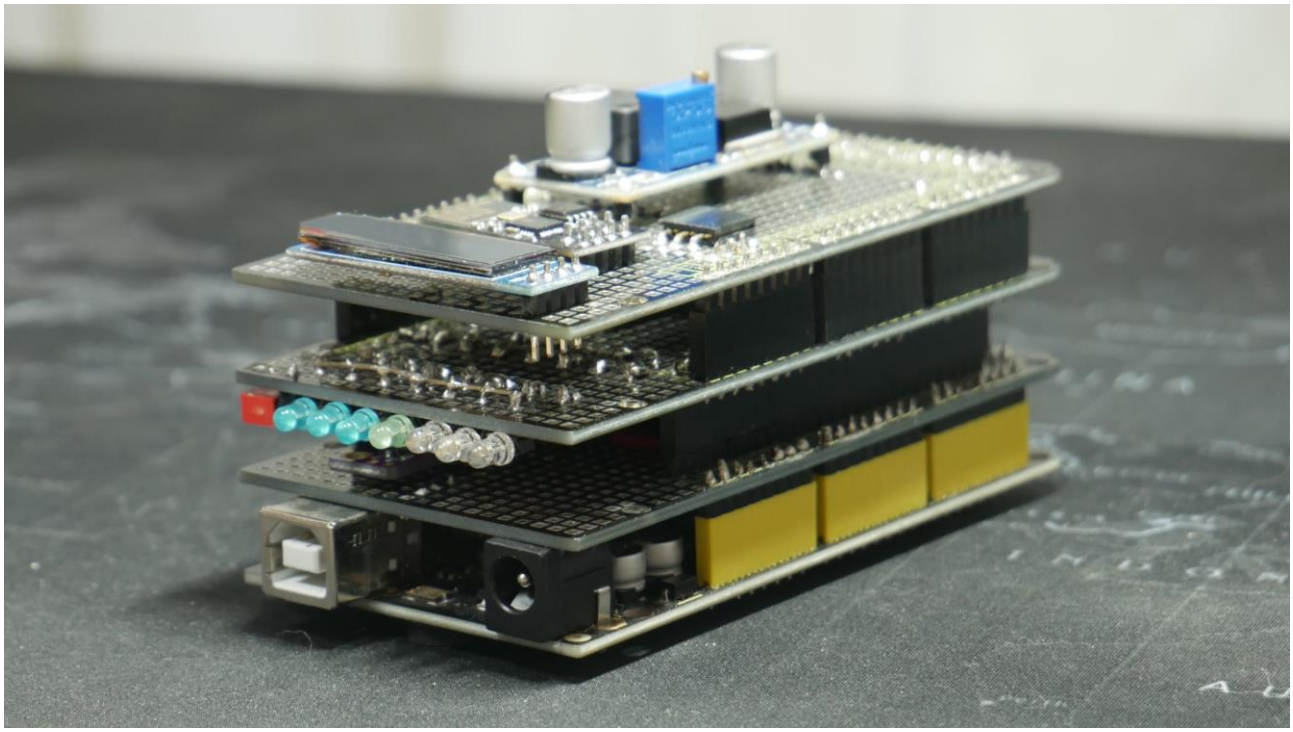


Д. В.3 Скріншот інтерфейсу програми.



Д. В.4 Скріншот інтерфейсу програми. Список даних, що зберігаються на пристрої

Д. В.5 Фото апаратно-програмного комплексу



Д. В.6 Фото міні-комп’ютеру