Національний університет «Полтавська політехніка імені Юрія Кондратюка»

Навчально науковий інститут інформаційних технологій та робототехніки

Кафедра комп’ютерних та інформаційних технологій і систем

Пояснювальна записка

до дипломної роботи

магістра

на тему:

Розробка програмного забезпечення тензометричної станції для потреб лабораторій Полтавської політехніки

Виконав: студент _2___ курсу, групи __д6ТН__ спеціальності

123 Комп’ютерна інженерія

Кириченко Володимир Анатолійович

Керівник к.т.н., доцент Фесенко Т.М.

Рецензент _____

Полтава – 2024 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

спеціальність 123 «Комп'ютерна інженерія »

на тему:

«Розробка програмного забезпечення тензометричної станції для потреб лабораторій Полтавської політехніки»

Студента групи д6ТН Кириченко Володимира Анатолійовича

Керівник роботи
кандидат технічних наук,
доцент Фесенко Т.М.
Завідувач кафедри
кандидат фізико-математичних
наук, доцент Двірна О.А.

Полтава – 2024

РЕФЕРАТ

Кваліфікаційна робота магістра: 53 с., 16 рисунків, 8 додатків, 21 джерело.

Об’єкт дослідження: автоматизація збору та обробки показань тензометричних датчиків.

Мета роботи: розроблення програмного забезпечення сучасної тензометричної станції для проведення лабораторних робіт у Полтавській політехніці.

Методи: проектування та розробка прошивки мікроконтролера, створення інтерфейсу та програмування вебдодатку.

Ключові слова: тензометрія, автоматизація, мікроконтролер, аналогово-цифровий перетворювач, прошивка, вебдодаток, інтерфейс.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АІС – автоматизована інформаційна система

ІС – інформаційна система

КІ – комп’ютерна інженерія.

ПЗ – програмне забезпечення.

ОС – операційна система.

АЦП – аналогово-цифровий перетворювач.

API – прикладний програмний інтерфейс.

SPS – частота вибірки даних.

DOUT (Data Out) – цифровий вихід.

SCK (Serial Clock) –лінія серійного таймера.

MCU (Microcontroller Unit) – мікроконтролер.

USB (Universal Serial Bus) – універсальна серійна шина.

VCC (Voltage Common Collector) – напруга живлення.

GND (Ground) – земля.

ВСТУП

Метою роботи є створення програмного забезпечення сучасної тензометричної станції для проведення лабораторних робіт у Полтавській політехніці. Дана станція сприятиме підвищенню якості навчального процесу та забезпечить можливість для студентів різних спеціальностей здобувати практичні навички роботи з вимірювальним обладнанням. Це особливо актуально в умовах, коли існуючі тензометричні станції морально та фізично застарілі [4], а їхня обмежена функціональність не дозволяє проводити всі необхідні експерименти.

Проєкт ґрунтується на ідеї інтеграції сучасних апаратних і програмних рішень для забезпечення простого у використанні, точного й надійного інструменту для навчальних потреб. Вибір сучасного контролера разом із модулями для вимірювання навантажень та деформацій дозволяє створити гнучку платформу, яка може бути легко адаптована під різні лабораторні роботи та експерименти [8 - 10]. Відмінною особливістю цього проєкту є також розробка програмного забезпечення з інтуїтивним інтерфейсом для збору, обробки та візуалізації даних у режимі реального часу.

Завдяки можливості інтеграції з комп'ютером та налаштуванням різних параметрів експериментів, тензометрична станція стане важливим доповненням до лабораторій Полтавської політехніки, дозволяючи викладачам і студентам ефективніше організовувати процес навчання та проводити дослідження з більшою точністю й контролем.

РОЗДІЛ 1
СУЧАСНІ РІШЕННЯ В ТЕНЗОМЕТРІЇ. АНАЛІЗ ПОТРЕБ ЛАБОРАТОРІЙ ПОЛТАВСЬКОЇ ПОЛІТЕХНІКИ

1.1 Призначення та основні напрямки застосування тензометрії

Тензометрія — це науково-технічна дисципліна, яка вивчає механічні напруження і деформації в матеріалах та конструкціях [1]. Основне її призначення — отримання точних даних про розподіл і величину деформацій та напружень, що дозволяє оцінити надійність конструкцій, діагностувати їхній стан та прогнозувати поведінку під впливом різних навантажень. Це є важливим для забезпечення безпеки та довговічності технічних об'єктів.

Основні напрямки застосування тензометрії:

– будівництво та інженерні споруди: моніторинг напружень у несучих елементах мостів, дамб, будівель, що дозволяє запобігати аваріям і забезпечувати надійність споруд;

– авіація та космонавтика: дослідження напружень у корпусах літаків, ракет і космічних апаратів, що піддаються екстремальним навантаженням під час польоту;

– машинобудування: контроль міцності та надійності деталей машин, транспортних засобів та іншого обладнання;

– лабораторні дослідження та освітня сфера: використовується для дослідження властивостей матеріалів, вивчення їхньої реакції на різні типи навантажень, що є важливим для інженерної освіти та розвитку студентських навичок [3].

В освітній сфері тензометрія відіграє важливу роль у підготовці майбутніх інженерів, техніків та науковців. Завдяки тензометричним вимірюванням студенти отримують практичний досвід роботи з вимірювальним обладнанням, вчаться аналізувати напруження і деформації в матеріалах, а також оцінювати міцність конструкцій. Це дозволяє їм глибше розуміти теоретичні принципи

механіки, опору матеріалів, а також здобувати навички, необхідні для проектування будівельних конструкцій та деталей машин.

Основні напрямки застосування тензометрії в навчальному процесі:

– дослідження фізико-механічних характеристик матеріалів (металів, бетону, дерева, композитів, полімерів) під дією навантажень;

– аналіз роботи конструкцій: моделювання реальних умов для оцінювання напружень у макетах будівельних конструкцій і деталей машин;

– лабораторні роботи та експерименти: проведення лабораторних занять, де студенти вчаться налаштовувати тензодатчики, проводити вимірювання, обробляти дані;

– науково-дослідна діяльність аспірантів та докторантів з технічних спеціальностей потребують широкого використання тензометрії для експериментальної перевірки нових теоретичних гіпотез, положень та моделей, оцінювання результатів отриманих теоретичних методик [4].

1.2 Теоретичні основи тензометрії

Визначення механічних напружень і деформацій у твердих тілах є основним завданням при оцінюванні міцності та жорсткості конструкцій [1]. Безпосереднє вимірювання напружень в матеріалах практично неможливе, тому доводиться обмежуватися експериментальним визначенням деформацій.

Поняття деформації в твердих тілах відображає зміну форми чи розмірів матеріалу під дією навантаження. Деформація може бути пружною, коли матеріал повертається до початкової форми після зняття навантаження, або пластичною, коли матеріал залишається деформованим після зняття навантаження.

Рівняння, що зв'язують напруження і деформацію, описуються законом Гука для пружних матеріалів, згідно з яким напруження σ пропорційне до деформації ϵ.

$$\sigma = E * \epsilon$$

Модуль Юнга E, модуль зсуву G та коефіцієнт Пуассона ν є ключовими фізико-механічними характеристиками, що визначають реакцію матеріалу на зовнішнє навантаження [3].

Вивчення напружень і деформацій у складних конструкціях потребує розгляду внутрішніх силових взаємодій. У рамках тензометрії використовують концепції нормальних та дотичних напружень, які допомагають оцінити, які внутрішні сили виникають в матеріалі. Особливого значення набуває тензор напружень, який описує розподіл напружень у кожній точці тіла. Це дозволяє побудувати детальну їх картину і передбачити можливі зони руйнування. За допомогою рівнянь рівноваги і граничних умов визначають напруження у будь-якій точці матеріалу, що є основою для інженерних розрахунків і проєктування конструкцій.

Основним інструментом вимірювання деформацій є тензодатчики — сенсори, які перетворюють механічну деформацію на електричний сигнал [3]. Найбільш поширеними є тензорезистивні датчики, що змінюють свій електричний опір під дією деформації. Опір таких датчиків змінюється пропорційно лінійним деформаціям внаслідок зміни довжини чутливого елементу датчика та зміни площі його перерізу. Це дозволяє фіксувати мінімальні зміни розмірів об'єкта. Ці зміни перетворюються на електричний сигнал, який далі обробляється і аналізується. Тензодатчики встановлюються на поверхню матеріалу в зонах, де необхідно вимірювати деформації, і працюють у складі вимірювальних схем [2], таких як мостова схема Уітстона. Для точного вимірювання змін опору тензодатчика використовуються підсилювачі, фільтри та аналого-цифрові перетворювачі.

1.3 Вплив температури та корекційні методи

Однією з важливих проблем у тензометрії є вплив температури на точність вимірювань. Оскільки температурні коливання призводять до зміни опору тензодатчика, потрібні спеціальні методи корекції. Застосовуються

температурно-компенсуючі тензодатчики або ж використовуються диференціальні схеми, де два датчики розташовують так, щоб один з них компенсував температурні зміни іншого. Це дозволяє зменшити похибку вимірювань, забезпечуючи більшу точність даних.

Корекція впливу температури не є актуальна у тих випадках коли випробування матеріалу чи конструкції здійснюється за короткий проміжок часу в декілька секунд чи декілька хвилин, але при більших тривалостях випробувань вплив температури може давати суттєві похибки [5].

1.4 Мостові схеми

Основним методом зчитування малих змін опору в тензометрії, є мостові схеми. Найбільш поширеною схемою є мостова схема Уїтстона (рисунок 1.1), що дозволяє більш точно виміряти зміну опору тензодатчика і перетворити її в електричний сигнал, зручний для подальшого аналізу. Ця схема включає чотири резистора, два з яких часто є тензодатчиками, розміщеними у відповідних гілках мосту, а інші два виступають як опорні резистори [2].

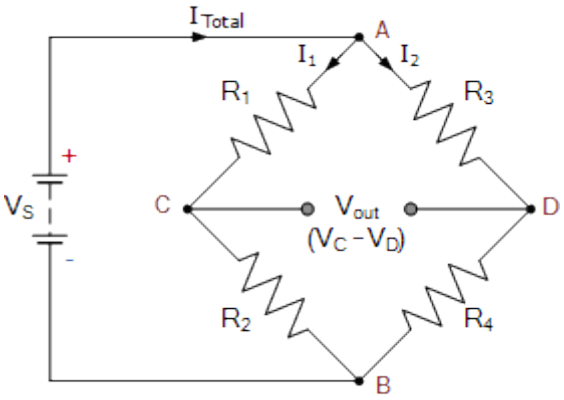

Рисунок 1.1. Мостова схема для вимірювання малої зміни опору

Коли всі резистори мають однаковий опір, напруга V_{CD}=0. У момент деформації, коли опір тензодатчиків змінюється, баланс порушується, і між середніми точками мосту виникає невелика напруга, яка пропорційна зміні

опору. Ця напруга підсилюється і використовується для визначення величини деформації.

У тензометрії застосовуються різні варіації мостової схеми Уїтстона [3] залежно від кількості тензодатчиків та їхнього розташування.

Чвертьмостова схема: використовується один тензодатчик. Така схема більш чутлива до зовнішніх впливів і температурних змін, але є найпростішою.

Напівмостова схема: включає два тензодатчики, що дозволяє краще компенсувати температурні ефекти.

Повномостова схема: застосовує чотири тензодатчики, які розміщують таким чином, щоб одні датчики реагували на стискання, а інші — на розтягнення. Це забезпечує найвищу точність і чутливість до механічної деформації та найбільш тесператрно скомпенсована.

Мостові схеми дозволяють отримувати високоточні вимірювання навіть при мінімальних змінах опору тензодатчиків. Крім того, їх легко інтегрувати в сучасні тензометричні системи з підсилювачами і аналого-цифровими перетворювачами, що робить їх зручними для автоматизації процесів збору і обробки даних.

Сучасна тензометрія також включає алгоритми цифрової обробки сигналів для підвищення точності та стабільності вимірювань. Дані, отримані з тензодатчиків, проходять кілька етапів обробки, серед яких фільтрація шумів, підсилення сигналу, а також калібрування. Калібрування тензометричної системи — це важливий процес, що дозволяє точно налаштувати вимірювальні прилади, визначити параметри перетворення механічних деформацій у електричний сигнал і врахувати вплив зовнішніх чинників у кожному конкретному випадку [3].

Таким чином, теоретичні основи тензометрії охоплюють широке коло понять і методів, від механічних характеристик матеріалів до електричних перетворень у тензодатчиках. Це дозволяє отримати високоточні дані про

напруження і деформації, що критично важливо для інженерних досліджень, оцінювання безпеки конструкцій і створення нових матеріалів.

1.5 Наявні технічні рішення для вимірювання деформацій

Лабораторії Полтавської політехніки оснащені багатьма випробувальними машинами [4], що дозволяють проводити комплексні дослідження та вимірювання для різних дисциплін і спеціальностей. Ці машини використовуються як у навчальному процесі, так і в науково-дослідній діяльності. Завдяки великій кількості лабораторних установок студенти мають змогу виконувати лабораторні роботи різноманітного спрямування з дисциплін «опір матеріалів», «будівельна механіка», «будівельні конструкції».

Лабораторні роботи проводяться на різних курсах і для студентів багатьох спеціальностей, включаючи будівельну інженерію, машинобудування, матеріалознавство та інші. Кожна з цих спеціальностей потребує застосування тензометрії для вимірювання деформацій різного роду конструкцій.

Важливість отримання ефективної тензометричної системи полягає в можливості забезпечення високої точності вимірювань та водночас в інформативності отриманих результатів. Відповідно, потреба в універсальній тензометричній системі, яка була б сумісною з різними випробувальними установками, є нагальною для лабораторій Полтавської політехніки. Це дозволить оптимізувати навчальний процес, підвищити ефективність та інформативність лабораторних занять і забезпечити студентів необхідними навичками роботи з сучасними технологіями.

Наявне тензометричне обладнання в лабораторіях Полтавської політехніки вже тривалий час перебуває в експлуатації і застаріло як морально, так і фізично. Існуючі вимірювачі деформацій використовують ручний спосіб перемикання каналів, що значно ускладнює і сповільнює процес вимірювання, особливо при проведенні багатоканальних тестів. Замість автоматизованого зняття даних,

оператор змушений вручну фіксувати показання кожного з датчиків, що може призводити до похибок через людський фактор.

Ще однією проблемою є те, що електронні схеми, які використовуються в цих станціях, фізично застаріли й уже не відповідають сучасним вимогам щодо точності і стабільності. Ці системи побудовані на базі аналогових компонентів, що унеможливлює інтеграцію з комп'ютерними технологіями для автоматизованої обробки і аналізу даних. Через це відсутня можливість швидкого збору, збереження та аналізу даних у цифровому вигляді, що є важливою вимогою в сучасних навчальних і дослідницьких умовах [1].

Відсутність можливості комп’ютеризації також обмежує застосування таких станцій у навчальних і наукових проєктах, які потребують комплексної обробки великих масивів даних, а також інтеграції з програмним забезпеченням для моделювання і аналізу. Потреба у сучасній тензометричній системі, що дозволяє автоматизоване зняття даних та інтеграцію з комп’ютерними системами, є надзвичайно актуальною для забезпечення якісного навчального процесу і розширення можливостей лабораторних досліджень.

Серед відносно ефективних систем, з допомогою яких проводяться, у тому числі, наукові дослідження, є прилад АИД-4 (рисунок 1.2) — це тензометрична станція, призначена для вимірювання деформацій за допомогою повних тензорезисторних мостів і напівмостів. Він забезпечує точність вимірювань на рівні ±0,1% від вимірюваного значення, дозволяючи фіксувати деформації в діапазоні від ±0,01 до ±0,50 мм/м. Частота вимірювань складає до 1 вимірювання на секунду, що відповідає ручному режиму фіксації даних [4].

Рисунок 1.2. Автоматичний вимірювач дефоомацій АИД-4

Прилад працює від мережі 220 В з частотою 50 Гц, що забезпечує стабільність роботи, але через обмежену частоту вимірювань і відсутність цифрової обробки даних він не відповідає сучасним вимогам для швидких і автоматизованих тестів.

Сучасний ринок тензометричних станцій характеризується невеликим обсягом виробництва, орієнтованим на специфічні потреби вузькопрофільних галузей. Окремі виробники створюють спеціалізовані станції на замовлення для конкретних завдань, таких як тестування матеріалів у аерокосмічній, автомобільній, будівельній індустрії, але універсальні рішення зустрічаються рідко. Це пояснюється тим, що різні типи вимірювань потребують специфічних вимог до точності, частоти збору даних, діапазону вимірювань і програмного забезпечення, що ускладнює створення універсального пристрою.

Попит на багатофункціональні, адаптивні системи є, однак, невеликий. У зв’язку з цим, ринок орієнтується на невеликі серії пристроїв з вузькою спеціалізацією, які є відносно вартісними. Це створює певний бар’єр для навчальних закладів і лабораторій, які часто змушені використовувати застаріле обладнання через високу вартість сучасних вузькоспеціалізованих рішень.

Сучасне програмне забезпечення для тензометричних станцій відіграє ключову роль у зборі, обробці та аналізі даних, але його можливості та доступність значно залежать від типу станції та виробника. Більшість виробників

надають спеціалізоване ПЗ разом зі своїми пристроями, яке дозволяє автоматично зчитувати дані з тензодатчиків, проводити калібрування, забезпечувати температурну компенсацію і виконувати розширений аналіз. Це програмне забезпечення, як правило, розроблене для роботи з конкретними моделями станцій, що обмежує можливість його застосування з обладнанням інших виробників.

Потужні програми можуть підтримувати інтеграцію з іншими науковими та інженерними програмами (наприклад, MATLAB), а також з системами для обробки великих масивів даних. Деякі системи передбачають функції реального часу для моніторингу деформаційних процесів і автоматичного сповіщення при перевищенні встановлених порогів.

Однак для бюджетного обладнання програмне забезпечення часто обмежене базовими функціями, такими як ручний запис даних і простий аналіз. Відсутність універсального програмного забезпечення, яке б забезпечувало сумісність з різними типами тензометричних станцій, залишається проблемою, особливо для освітніх і наукових закладів, де використовуються тензометричні системи різних поколінь та виробників.

1.6 Аналіз потреб лабораторій в апаратному та програмному забезпеченні

Лабораторії навчальних закладів мають особливі вимоги до апаратного та програмного забезпечення для тензометричних станцій, оскільки воно повинно забезпечувати не тільки достатню точність і зручність вимірювань, а й мати навчально-демонстраційні можливості та бути універсальним для проведення різного роду лабораторних робіт. Студенти різних спеціальностей потребують доступного і зрозумілого інтерфейсу для збору даних, а також інструментів для аналізу і візуалізації результатів, що сприяє їхньому розумінню процесів деформування конструкцій та навичкам роботи з реальним обладнанням [4].

Програмне забезпечення для навчальних лабораторій повинно включати інтерактивні функції, які дозволяють легко налаштовувати параметри вимірювання, автоматично зчитувати дані з різних каналів, представляти результати як у числовому так і у графічному вигляді та створювати звіти з детальною статистикою. Іншою ключовою потребою є можливість роботи в багатоканальному режимі, що дає змогу проводити комплексні дослідження матеріалів, конструкцій і вимірювати множинні параметри в реальному часі.

Також важливою є підтримка інтеграції з іншими системами для обробки даних, такими як MATLAB або Excel, що дозволяє студентам обробляти результати в знайомому середовищі. Крім того, навчальні заклади часто потребують доступного ліцензування ПЗ, щоб використовувати його на багатьох пристроях одночасно, та регулярних оновлень для підтримки новітніх технологій і можливостей.

Програмне забезпечення має забезпечувати інтуїтивно зрозумілий інтерфейс, який полегшує навігацію та розуміння основних функцій без попереднього навчання. Інтуїтивний дизайн інтерфейсу передбачає просту структуру меню з чіткими іконками та підказками, які допомагають студентам швидко налаштувати параметри вимірювань, перемикатися між каналами, зчитувати та аналізувати результати в реальному часі. Це особливо важливо для нових користувачів, яким важко освоїти нові системи у короткий період лабораторної роботи. Функція зберігання налаштувань дозволили б викладачам і студентам створювати та зберігати профілі для різних типів вимірювань або лабораторних робіт, що забезпечує послідовність та економить час під час повторного використання обладнання. Кожна лабораторна робота може мати власний набір попередньо налаштованих параметрів, таких як частота зчитування, кількість активних каналів, тип мостової схеми, коефецієнти чутливості кожного з датчиків, які необхідно легко завантажувати у відповідності до завдання.

Програмне забезпечення має забезпечувати автоматичне збереження даних у процесі експерименту і надійне зберігання результатів, щоб уникнути їх втрати через технічні несправності, вихід обладнання з ладу, втрату живлення комп’ютера чи станції, втрату електричного сигналу з датчиків, тощо.

Важливим є й можливість детального опису кожної лабораторної роботи безпосередньо у програмі: для кожного експерименту можна вказати інструкції, параметри налаштувань, очікувані результати, а також методику аналізу даних. Такий опис дозволяє студентам орієнтуватися у процесі роботи з мінімальною участю викладача, підвищуючи самостійність та розуміння експериментальних завдань.

Проведений аналіз лабораторних робіт, що виконуються в лабораторіях кафедри будівельних конструкцій та наявного обладнання [4] показав очікувані параметри тензометричної станції, яка повинна відповідати наступним кількісним показникам.

Діапазон вимірювань відносних деформацій $(-500…500)\times10^{-5}$, що дозволяє охоплювати весь спектр допустимих деформацій тензодатчиків та можливих деформацій у сталі, бетоні та дереві.

Кількість каналів – у лабораторних роботах, що проводяться, використовуються одночасно від 4 до 16 тензодатчиків. Багатоканальність дає можливість проводити комплексні дослідження об'єктів, де необхідно фіксувати деформації одночасно у різних місцях.

Частота вимірювань не є критичною. В лабораторія не проводяться випробування на динамічні навантаження. Тому достатньою є навіть частота в 1 вимірювання за секунду (1 Гц). Така частота дозволить проводити моніторинг деформацій в реальному часі при проведенні статичних випробувань.

Точність вимірювання повинна бути не нижче 0,1% від максимальної шкали вимірювань. Це важливо при вимірюваннях малих деформацій, особливо у випадку проведеня наукових досліджень, де незначні похибки можуть істотно вплинути на підсумкові дані.

Електричний опір тензометричний датчиків, що використовуються в наявних лабораторних роботах, може бути різний – від 100 Ом до 400 Ом залежно від їх бази, типу та призначення датчиків [3].

Температурна компенсація – наявні лабораторні роботи виконуються при стабільній температурі, тому функція для температурної корекції вимірювань не є обов'язковою.

Інтерфейс для підключення до ПК – кращим варіантом буде підключення по USB для передачі даних на комп'ютер. Це спростить використання апаратного забезпечення при виконанні лабораторних робіт.

Тензометричний міст потребує живлення 3…5В, причому, вихідні параметри напруги пропорційні до напруги живлення, тобто стабільність напруги живлення повинна бути забезпечена, як мінімум в межах 0.1%. При цьому слід врахувати що при різній кількості датчиків струм живлення може суттєво відрізнятися тому апаратне забезпечення потребує стабілізації напруги в широких межах зміни струму живлення.

Тензометричні станції можуть працювати як з постійним, так і зі змінним струмом, кожен з яких має свої переваги і сфери застосування. Використання постійного струму є традиційним і найбільш підходящим для простих лабораторних установок, що часто характерно для навчальних лабораторій.

Перевагою використання постійного струму є менші вимоги до обладнання, DC-системи простіші у підключенні, а вимірювання постійної напруги значно простіше [1].

Застосування змінного струму (AC), у свою чергу, є доцільним для промислових та складних вимірювальних установок, де можливе виникнення значних перешкод або коли тензометричні кабелі довші. AC дозволяє зменшити вплив шумових зсувів у великих системах, але його використання вимагає додаткових заходів для компенсації та обробки сигналів.

Таким чином, для навчальних лабораторних робіт з короткими кабелями й відсутністю сильних перешкод застосування постійного струму є оптимальним і

економічно виправданим варіантом, забезпечуючи точність і спрощуючи проектування тензометричних станцій.

Такі показники є базовими для тензометричної станції, яка призначена для навчальних і дослідницьких лабораторій, забезпечуючи ефективність, точність і зручність використання.

РОЗДІЛ 2
АПАРАТНА ЧАСТИНА ТА ПРОГРАМУВАННЯ КОНТРОЛЕРА

2.1 Вибір апаратної платформи для тензометричної станції

У сучасних лабораторних дослідженнях важливо мати доступ до надійних і доступних технологій для вимірювання деформацій і навантажень. Вибір апаратної платформи для тензометричної станції є ключовим етапом, що вплине на простоту використання та можливість подальшого розвитку системи іншими користувачами.

Конфігурація тензометричної станції складається з кількох ключових компонентів. Основні елементи системи включають мікроконтролер, вимірювальну схему, а також мультиплексор для перемикання між кількома каналами.

Функціонал мікроконтролера полягає в управлінні всіма компонентами системи, обробці даних з тензодатчиків і здійсненні обчислень для отримання необхідних вимірювань. Вибір контролера, такого як Arduino, забезпечує:

– зчитування даних від вимірювальної схеми, перетворення їх у цифрові значення та обробка цих даних;

– управління роботою мультиплексора, вибираючи активний канал для зчитування даних з конкретного тензодатчика;

– вивід даних в порт комп'ютера для подальшого аналізу.

Однією з найбільш популярних і широко використовуваних платформ у цій сфері є мікроконтролери сімейства Arduino. Вони отримали визнання завдяки своїй доступності, простоті програмування та активній спільноті, що підтримує розробників різного рівня. Arduino є ідеальним вибором для навчальних лабораторій, де студенти можуть швидко освоїти основи програмування і електроніки [9].

Серед численних мікроконтролерів, доступних на ринку, слід виділити такі платформи, як Raspberry Pi, ESP8266 та ESP32, а також Arduino. Кожен з них має свої переваги та недоліки:

– Raspberry Pi – потужний мікрокомп'ютер з можливістю запуску повноцінної операційної системи. Проте його складність у налаштуванні та вищі вимоги до електроживлення а також суттєво вища ціна можуть бути проблемою для простих лабораторних установок;

– ESP8266 та ESP32 – ці мікроконтролери мають вбудований Wi-Fi, що дозволяє створювати бездротові системи для збору даних. Однак їх програмування є складнішим, ніж у Arduino, і може вимагати додаткових знань про мережеві протоколи;

– Arduino– це платформа, яка поєднує в собі простоту використання з достатньою потужністю для виконання більшості завдань у вимірювальних системах. Вона є популярною завдяки численним бібліотекам і підтримці різних модулів, що робить її ідеальним вибором для проектів у навчальних закладах [9].

Остаточно для реалізації проекту був вибраний мікроконтролер Arduino, що обумовлено кількома ключовими факторами:

– простота навчання – Arduino має простий та зрозумілий синтаксис програмування, що робить його більш доступним. Користувачі системи можуть швидко вивчити основи написання коду, що дозволяє їм зосередитися на вимірюваннях, а не на технічних аспектах програмування;

– доступність компонентів – багато компонентів для Arduino, включаючи датчики, модулі та плати, доступні на ринку. Це дозволяє легко знайти необхідні матеріали для побудови та налаштування системи;

– активна спільнота – Arduino має величезну спільноту користувачів, яка постійно ділиться своїми напрацюваннями, проектами та бібліотеками. Це забезпечує доступ до великої кількості ресурсів, що дозволяє розробникам швидше знаходити рішення для своїх завдань;

– гнучкість і масштабованість – Arduino має бібліотеки для різноманітних модулів, що дозволяє створити адаптивну систему в залежності від потреб лабораторії. Можливість підключення додаткових датчиків і модулів розширює функціональність станції.

– можливість реалізації реального часу – завдяки своїм характеристикам Arduino може обробляти дані в реальному часі, що є важливим для вимірювань деформацій.

Завдяки цим перевагам Arduino виступає оптимальним вибором для розробки тензометричної станції, що відповідає вимогам навчальних закладів, де необхідно поєднати точність вимірювань, простоту використання та економічність. Для проекту вибираємо Arduino Nano V3 на базі ATmega328p (рисунок 2.1).

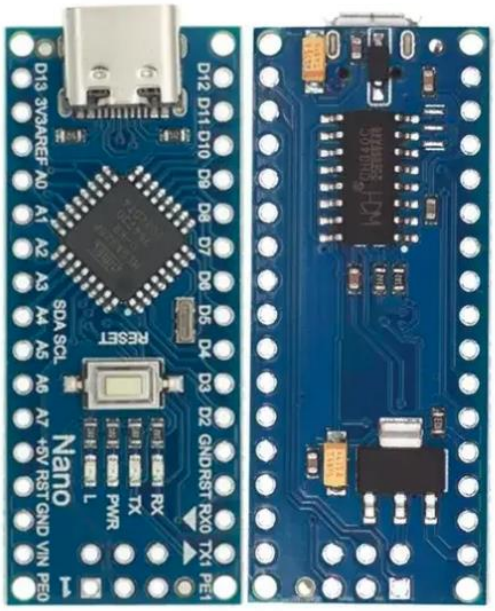

Рисунок 2.1. Загальний вигляд плати на базі мікроконтролера Arduino

Вимірювальна схема, є критично важливим елементом системи, оскільки вона виконує роль перетворювача аналогових сигналів з тензодатчиків на цифрові. Основні функції вимірювальної схеми включають:

– посилення сигналу, що надходить з тензодатчиків, забезпечуючи їх достатню чутливість для точного зчитування;

– цифрове перетворення аналогового сигналу у цифрові дані з достатньою точністю;

– фільтрацію даних – вимірювальна схема має вбудовані функції для фільтрації даних, що дозволяє зменшити вплив шумів і покращити точність і стабільність вимірювань.

Аналіз відомих сучасних аналого-цифрових перетворювачів показує що оптимальним варіантом для даного проекту є АЦП HX711 (рисунок 2.2) – це високоточний аналого-цифровий перетворювач, спеціально розроблений для роботи з тензодатчиками та іншими вимірювальними пристроями. HX711 забезпечує високу точність зчитування (заявлено 24 біт, хоча реальна роздільна здатність біля 20 біт), має 64-кратний підсилювач сигналу, що робить його ідеальним для застосувань, де потрібні точні вимірювання сили, ваги або деформації, підтримує два канали вимірювання, легко підключається до популярних мікроконтролерів, таких як Arduino, завдяки простим протоколам зв’язку та має низьке енергоспоживання. HX711 часто використовується в таких системах, як ваги, системи контролю навантаження та подбні установки для вимірювання механічних параметрів.

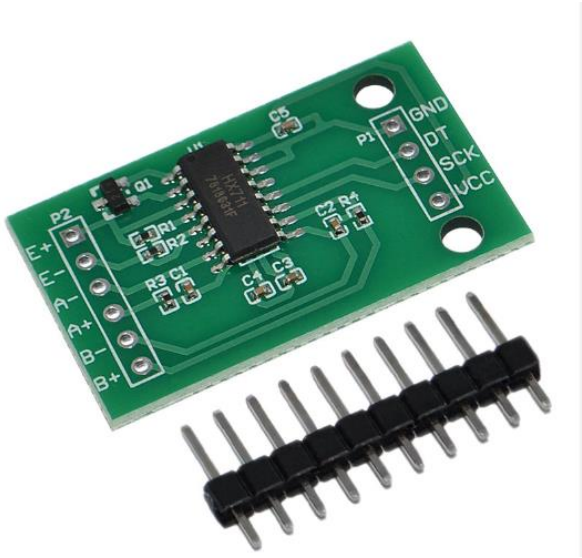

Рисунок 2.2. АЦП з вбудованим підсилювачем - HX711

Мультиплексор є додатковим компонентом, який дозволяє системі переключатися між кількома тензодатчиками. Основні функції мультиплексора включають перемикання каналів між кількома тензодатчиками і дозволяє вибрати, з якого з них зчитувати дані в даний момент.

Для даного проекту вибраний 16-канальний мультиплексор CD74HC4067 (рисунок 2.3), який дозволяє вибирати один з кількох вхідних або вихідних сигналів [10].

Основні його характеристики та особливості наступні:

– кількість каналів – 16;

– призначений для роботи як з аналоговими так і з цифровими сигналами;

– використовує 4 адресні лінії для вибору активного каналу, що забезпечує простоту у підключенні і управлінні;

– забезпечує високу швидкість перемикання 6нс;

– широкий діапазон живлення – від 2 В до 6 В, що робить його гнучким для використання з різними мікроконтролерами.

Рисунок 2.3. Мультиплексор CD74HC406

Конфігурація тензометричної станції, що складається з мікроконтролера, вимірювальної схеми та мультиплексора, забезпечує ефективне вимірювання деформацій. Кожен компонент виконує специфічні функції, які в сукупності

дозволяють реалізувати надійну та гнучку програмно налаштовувану систему для проведення навчальних лабораторних досліджень та наукових експериментів.

2.2 Електрична схема

Класичні тензометричні станції використовують послідовне перемикання каналів (низькоомним механічним реле) і зчитування показань, при якому живлення робочого і компенсаційного тензодатчиків подається лише на активний канал. Така схема має певний недолік у нашому випадку: автоматичне електронне перемикання каналів не забезпечує стабільного низького (практично нульового) опору ключа, що може призвести до суттєвих коливань у зчитуванні показань [2, 3].

У цьому проєкті було вирішено використовувати постійне живлення всіх тензодатчиків, а перемикання каналів здійснювати лише в колі вимірювання (рисунок 2.4). Такий підхід має свої переваги та недоліки.

Перевагами такого методу є те, що постійний струм через тензодатчики знижує ризик збоїв, пов’язаних із початковими змінами температурного режиму датчиків, оскільки після певного часу після увімкнення схеми їхній температурний стан стабілізується. Це позитивно впливає на точність вимірювань і знижує вплив теплових коливань внаслідок нагріву та охолодження датчиків на їх показання.

Основним недоліком є підвищене енергоспоживання, яке вимагає реалізації блоку живлення значно більшої потужності. АЦП HX711 має вбудований стабілізатор живлення мостової схеми, проте його потужності недостатньо для живлення багатьох датчиків одночасно. Тому в проєкті вирішено живити систему від стабілізатора із стабілізатора плати Arduino, здатного забезпечити достатній струм живлення, що відповідає потребам системи [5].

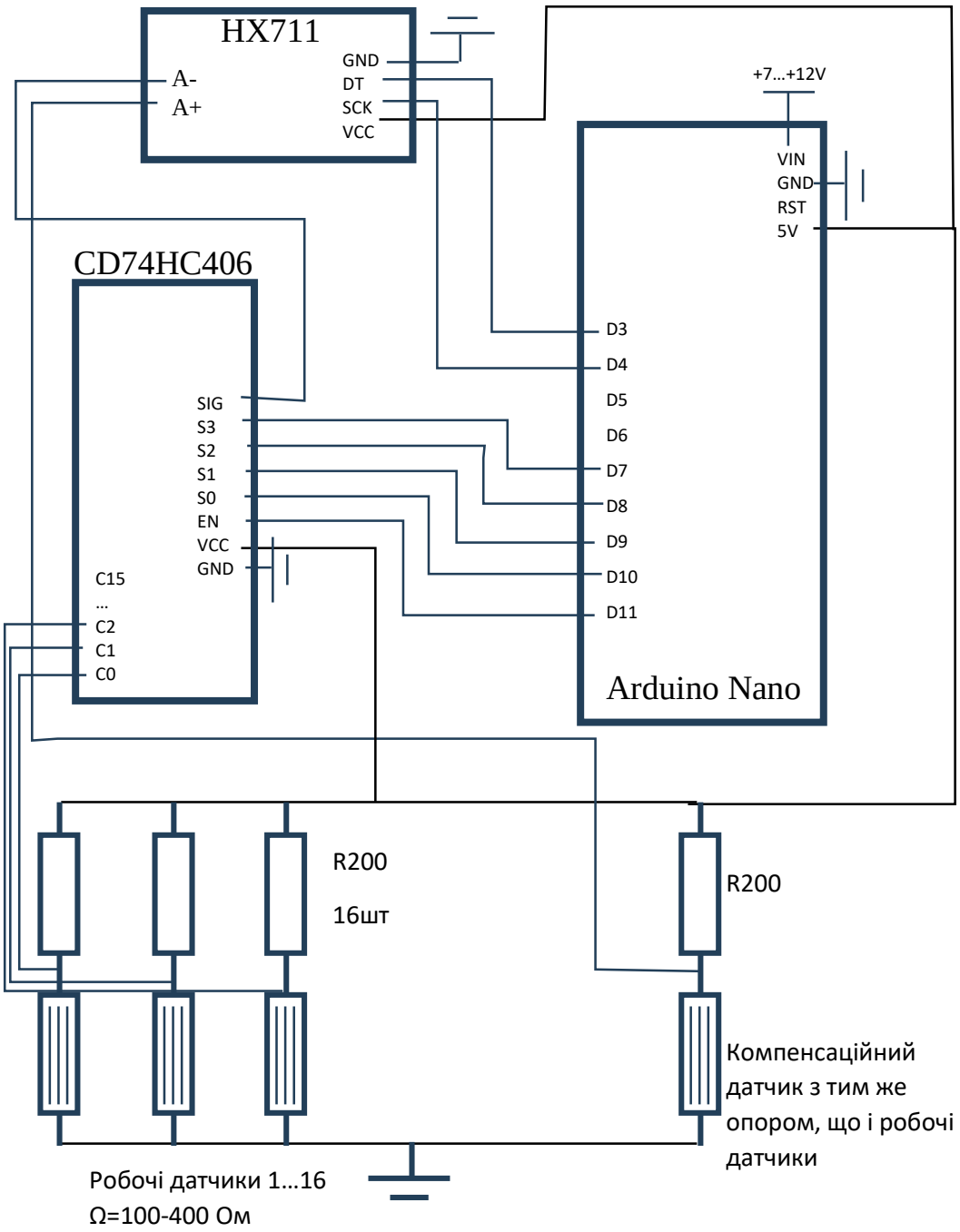

Рисунок 2.4. Електрична схема тензометричної станції

Як правило, у мостових схемах електричний опір всіх віток моста повинен бути однаковий для забезпечення максимальної точності результатів. Проте реальні потреби лабораторних досліджень вимагають застосування тензометричних датчиків різного електричного опору, як правило від 100 до 400 Ом, причому у кожній лабораторній установці як правило електричний опір

всіх тензодатчиків є однаковим. Тому прийнято рішення встановлення незмінних резисторів (200 Ом) для однієї половини мостової схеми та використання компенсаційного датчика, що має такий же опір як і робочі для іншої половини. такий підхід дозволяє суттєво спростити електричну схему хоча зменшує вихідну напругу і відповідно точність на кілька відсотків та вимагає корекції тензометричного коефіцієнта використовуючи значення вказаних електричних опорів.

2.3 Вибір середовища для програмування контролера

Існують різні платформи для програмування мікроконтролерів, які надають додаткові можливості для професійного використання. Однією з таких є PlatformIO [6] — мультиплатформенне середовище, що підтримує широкий спектр мікроконтролерів, зокрема й Arduino. PlatformIO пропонує розширені функції відладки та інтеграцію з Visual Studio Code [7], що є дуже корисним для професіоналів. Однак, PlatformIO може виявитися складнішим для початківців через свою багатофункціональність.

Іншим варіантом є Atmel Studio, розроблене спеціально для мікроконтролерів AVR та SAM, на яких базуються більшість плат Arduino. Atmel Studio забезпечує глибоку підтримку апаратного забезпечення та надає додаткові інструменти для відлагодження й профілювання, що робить його ідеальним вибором для складніших проєктів, хоча воно й складніше для освоєння. Ще однією популярною альтернативою є MPLAB X IDE, що підходить для програмування мікроконтролерів Microchip, як-от PIC та AVR, але не підтримує Arduino, що робить його менш універсальним у цьому контексті.

Попри це, Arduino IDE залишається оптимальним вибором, особливо для навчальних проєктів і початкових етапів розробки. Його основна перевага полягає в простоті та доступності, що робить його ідеальним інструментом для студентів і новачків. Arduino IDE пропонує широку підтримку бібліотек і прикладів, зокрема для таких компонентів, як HX711 та CD74HC4067, що

суттєво прискорює роботу та полегшує вирішення типових задач. Велика кількість доступної документації та активне ком'юніті допомагають швидко освоїти основи програмування мікроконтролерів і знаходити рішення для розповсюджених проблем. Arduino IDE легко налаштувати, і воно підходить для швидкого створення невеликих проєктів, що робить його незамінним у лабораторних роботах та студентських проєктах, де особливо важливі легкість і швидкість старту.

2.4 Взаємодія з HX711

Програмування взаємодії з HX711 потребує налаштування вручну протоколу зв'язку. HX711 використовує простий інтерфейс із двома лініями: DOUT (дані) і SCK (синхронізація), де дані передаються по 24 біти за раз з використанням імпульсів на піні SCK.

Згідно з документацією [7], основні кроки для взаємодії з HX711 наступні:

– налаштування пінів – DOUT як вхід для зчитування даних, і SCK як вихід для формування тактових імпульсів;

– зчитування 24-бітного значення шляхом зчитування імпульсів на DOUT і передавання імпульсів на SCK.

– налаштування коефіцієнта посилення для наступного зчитування – HX711 дозволяє налаштувати коефіцієнт посилення (128, 64, або 32) шляхом додаткових імпульсів SCK, що подаються після основних імпульсів.

Оскільки у кожному конкретному випадку (для різних датчиків) калібрувальні коефіцієнти будуть різні, прийнято рішення проводити обробку даних безпосередньо на боці комп'ютера, тому з мікроконтролера прийнято рішення зчитувати сирі дані (24 бітне ціле число для кожного датчика).

За результатами попередніх експериментів, у ряді випадків АЦП HX711 демонструє значні відхилення, надаючи явно некоректні дані, які виходять за допустимі межі. Це може бути пов'язано з шумом або іншими зовнішніми факторами, що впливають на стабільність показань. У зв'язку з цим було

прийнято рішення використовувати медіанний фільтр [5] для обробки даних. Застосування медіанного фільтра дозволяє відсіяти аномальні значення, забезпечуючи більш надійні результати і підвищуючи загальну точність вимірювань.

Медіанний фільтр — це алгоритм обробки сигналів, який використовується для видалення шуму та аномальних значень з вимірювань. Основна ідея цього методу полягає у заміні кожного значення на середнє (медіанне) значення з групи сусідніх значень, щоб згладити сигнал, відсіявши аномальні пікові значення, що можуть виникати через перешкоди або коливання в вимірювальній системі (рисунок 2.5). При цьому для кожного зчитування формується "вікно" з кількох сусідніх значень (зазвичай непарного розміру, наприклад, 3, 5 або 7 значень), включаючи поточне зчитування. Дані всередині вікна сортуються в порядку зростання. Це дозволяє виділити середнє значення, яке менше піддається впливу крайніх значень, яке і буде остаточним значенням для поточного вимірювання. Перевагами медіанного фільтра є стійкість до аномалій (медіанний фільтр ефективно видаляє імпульсні завади або викиди, оскільки лише одне "аномальне" значення не може вплинути на медіану. Також алгоритм не вимагає складних обчислень, а сортування невеликих вікон (наприклад, із 3 або 5 елементів) виконується швидко.

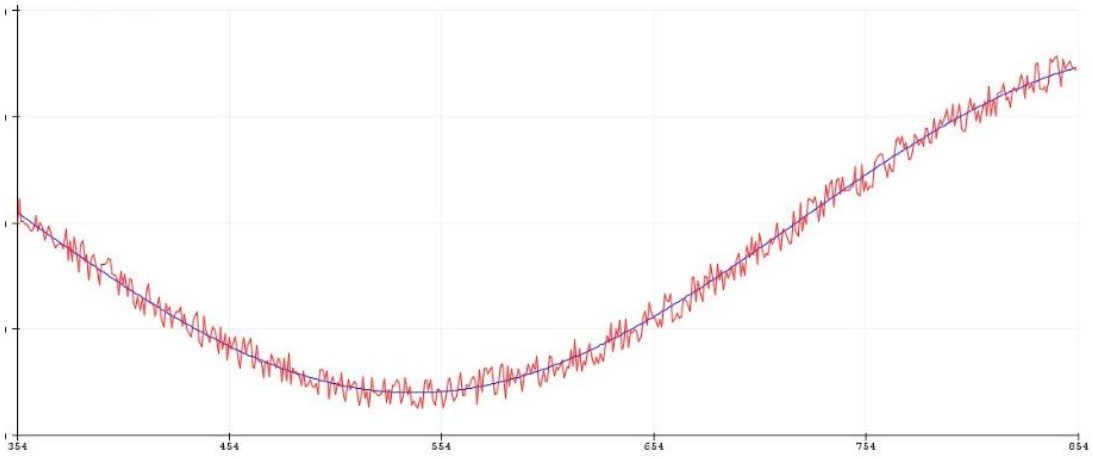

Рисунок 2.5. Приклад застосованого медіанного фільтру

У контексті вимірювань, проведених АЦП HX711, медіанний фільтр дозволяє згладити раптові відхилення показань, які не відповідають реальним даним, наприклад, через електричні перешкоди чи короткочасні збої. Фільтрацію даних вирішено реалізувати на стороні контролера, при цьому використовуючи налаштовуваний рівень фільтрації (від одного до дев'яти зчитувань на показання).

Швидкодія вибраного АЦП обмежується максимальною частотою вибірки, яка становить 10 або 80 Гц (SPS), а також часом перемикання між каналами (output settling time), який дорівнює 400 або 50 мс. Таким чином, для мінімальний інтервал між послідовними зчитуваннями одного датчика має становити щонайменше 12,5 мс, а при перемиканні датчика – як мінімум, 50 мс. Таким чином, на зчитування показань 16-ти датчиків із застосуванням медіанного фільтру між 5-ма значеннями необхідно близько $16 * (5 - 1) * 12.5 + (16 - 1) * 50 = 1550$ мс. Така швидкодія є більш ніж достатньою у випадку проведення статичних випробувань. Для більшої стабільності показань ці інтервали можна збільшити. В даному проекті прийнято рішення зробити ці інтервали програмно налаштовуваними.

2.5 Взаємодія з мультиплексором CD74HC4067

Мультиплексор CD74HC4067 [10] забезпечує зчитування сигналів з 16-ти каналів, передаючи аналоговий сигнал на вхід АЦП. Перемикання каналів керується мікроконтролером. Для вибору потрібного каналу мікроконтролер надсилає на адресні пін-коди мультиплексора певну комбінацію сигналів. Ця комбінація вибирає один з каналів, дозволяючи сигналу з нього потрапляти на вихід. У вибраного мультиплексора чотири адресні пін-коди (S0, S1, S2, S3) контролюють вибір каналу, подаючи двійкове значення, що відповідає номеру каналу (рисунок 2.6).

S0	S1	S2	S3	$\overline{E}$	SELECTED CHANNEL
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

Рисунок 2.6. Таблиця істинності CD74HC4067

Мікроконтролер послідовно перемикає канали, зчитуючи сигнал з кожного з них. Після вибору каналу мікроконтролер витримує необхідну паузу, здійснює зчитування даних з АЦП та зберігає отримане значення. Цикл повторюється для кожного каналу. Оскільки швидкість перемикання мультиплексора висока, ця схема дозволяє ефективно працювати з кількома датчиками, використовуючи один канал АЦП.

Електричний опір закритих каналів мультиплексора є досить високим, що практично усуває вплив сусідніх датчиків на той, який у даний момент часу перебуває у робочому стані. Це забезпечує точність вимірювань, оскільки ізоляція між каналами дозволяє зчитувати лише сигнал активного датчика без спотворень від сусідніх елементів системи.

2.6 Формат даних для обміну з ПК

Обмін з комп’ютером здійснюється в текстовому форматі для зручності обробки та аналізу. Після кожного циклу зчитування датчиків набір даних

передається до вихідного буферу. Кожен набір даних містить кілька значень: виміряну деформацію на кожному датчику, величину навантаження, що відповідає цій деформації, та напругу після контролера. Значення стану контролера включає інформацію про поточну напругу живлення, що дозволяє контролювати стабільність системи під час роботи. Виведення інформації в текстовому форматі дозволяє легко інтегрувати дані в інші програми для подальшого аналізу чи візуалізації, а також зберігати результати у файлі для архівування.

Для передавання команд комп’ютер надсилає на контролер вхідні налаштування, які можуть включати вибір частоти зчитування, налаштування каналів та режими вимірювання. Це дозволяє здійснювати гнучке налаштування системи, адаптуючи її роботу під конкретні експериментальні задачі. Такий формат обміну є універсальним і зручним для використання в лабораторних умовах.

2.7 Програмовані налаштування контролера

У процесі розробки системи було визначено доцільність використання таких налаштувань: номери активних каналів, кількість зчитувань для реалізації фільтрації даних, тривалість пауз після завершення зчитувань з усіх датчиків, інтервал паузи між послідовними зчитуваннями одного датчика, тривалість паузи після перемикання на інший датчик, а також тривалість паузи між послідовними зчитуваннями значень напруги, що надходить на плату, та значення сили.

Ці налаштування дозволяють оптимізувати роботу системи відповідно до специфічних вимог вимірювань, зокрема забезпечують стабільність показань шляхом коректного встановлення інтервалів між зчитуваннями, що мінімізує можливі помилки та спотворення сигналу.

В наступній таблиці показані значення цих параметрів та їх орієнтовні межі.

Налаштування	min	max
Активні канали (1-16)	0	2^{16}-1 = 65535
Кількість послідовних зчитувань одного каналу перед фільтрацією	1	9
Пауза після завершення зчитувань всіх каналів, мс.	0	5000
Пауза після перемикання каналу	50	500
Пауза між послідовними зчитуваннями одного каналу, мс.	15	200
Пауза між зчитуваннями значення робочої напруги та значення навантаження	200	5000

Номери активних каналів позначаються за допомогою двобайтового числа, використовуючи побітову маску. Це означає, що кожен із 16 каналів відповідає одному біту у двох байтах, де кожен біт вказує на стан конкретного каналу (увімкнений або вимкнений).

Наприклад, перший біт відповідає першому каналу, другий біт — другому каналу і так далі до 16-го біту, який відповідає останньому каналу. Якщо біт у масці встановлений в "1", відповідний канал активний і буде враховуватися при зчитуванні; якщо біт встановлений в "0", цей канал пропускається. Такий підхід дозволяє гнучко вибирати комбінацію активних каналів, ефективно керуючи зчитуванням даних з множини датчиків через один двобайтовий параметр а також мінімізує об’єм даних, що передаються в контролер при зміні налаштувань.

Передбачено збереження останніх налаштувань у внутрішній пам'яті контролера EEPROM (Electrically Erasable Programmable Read-Only Memory) для можливості подальшого зчитування при наступному включенні приладу. Налаштування зберігаються у фіксованих адресах у вигляді двобайтних чисел, при зміні налаштувань система автоматично записує їх у внутрішню пам'ять.

Після ініціалізації контролера збережені налаштування передаються в порт для зчитування комп'ютером. Це дає змогу контролювати ці налаштування користувачем перед здійсненням вимірювань.

РОЗДІЛ 3
КЛІЄНТСЬКА ЧАСТИНА: ЗЧИТУВАННЯ, ЗБЕРІГАННЯ, ОБРОБКА ТА ВІЗУАЛІЗАЦІЯ ДАНИХ

3.1 Вимоги до ПЗ

Програмне забезпечення повинно бути сумісним із найбільш поширеними операційними системами, такими як Windows, macOS та Linux, і вимагати мінімальних апаратних ресурсів для стабільної роботи навіть на застарілих комп'ютерах. Ідеально, щоб програма не потребувала встановлення додаткових компонентів або сторонніх бібліотек, забезпечуючи повну функціональність "із коробки" для користувача. Це дозволить легко розгортати та експлуатувати програмне забезпечення у різних навчальних лабораторіях без необхідності додаткової технічної підтримки.

Зчитування даних та зв’язок із контролером. Програма повинна коректно приймати та передавати дані від контролер в текстовому форматі через USB, включаючи зчитування показань у реальному часі та обмін налаштуваннями. Команди, що передаються на контролер, забезпечують збереження конфігурацій під специфічні вимоги до зчитування даних.

Оскільки дані надходять серіями, програма повинна обробляти їх у режимі реального часу, застосовуючи коефіцієнти тензометрії, щоб забезпечити коректне перетворення в значення деформації. Особливу увагу слід приділити відстеженню можливих помилок під час зчитування, таких як відхилення показань або відсутність сигналу, а також можливості корекції цих помилок.

Зберігання та експорт. Необхідно зберігати всі отримані дані для їх подальшого перегляду. Програма повинна мати функцію конвертації даних у зручні формати для інтеграції з іншими програмними продуктами. Зважаючи на те, що протягом проведення лабораторного експерименту можливі збої, пов'язані із різними факторами, корисним функціоналом буде збереження даних в режимі реального часу по мірі їх надходження їх із контролера. При цьому потрібно

врахувати можливі збої як на стороні контролера так і на стороні комп’ютера, наприклад, випадкові відключення контролера, розірвання зв’язку або ж вимкнення комп'ютера чи вихід із програми.

Візуалізація. Програма має надавати візуалізацію результатів у вигляді таблиць та графіків у режимі реального часу, щоб користувачі могли відстежувати процес та результати випробувань інтерактивно. При цьому важливо мати можливість встановлення «нульових» показів, тобто показів за відсутності навантаження на конструкцію.

3.2 Вибір платформи

Розглядаючи варіанти для створення програми, доцільно розглянути платформи, які забезпечують швидке виконання коду та легкість розгортання. Серед таких варіантів можна виділити десктопні програми та веб-платформу [12-14].

Розробка на **Python** забезпечує кросплатформеність і мінімальні системні вимоги. Python має широку підтримку бібліотек для роботи з даними, а також забезпечує легкий доступ до функцій графічного інтерфейсу. Перевагами є простота реалізації; доступ до широкого спектру бібліотек, кросплатформеність. Недоліком може бути необхідність у встановленні Python-інтерпретатора, складність розгортання оновлень для користувачів.

C# на .NET — оптимальний варіант для Windows-середовища, забезпечує гарну сумісність з апаратним забезпеченням та можливості інтеграції з USB-пристроями. Переваги – висока швидкодія. розширені можливості взаємодії з системним обладнанням. Недоліки – обмежена кросплатформеність, вимагає налаштування .NET середовища, що може обмежити універсальність рішення.

Java — стабільна кросплатформенна мова, яка дозволяє створювати потужні десктопні застосунки з підтримкою відображення графіків та таблиць. Переваги – сумісність з різними ОС, можливість налаштування графічного інтерфейсу для відображення даних у таблицях та графіках. Недоліки – потреба

у Java Runtime Environment, складне розгортання для користувачів, які не мають Java.

Веб-платформа — HTML, CSS, JavaScript (можливо, із використанням фреймворків React або Vue) для frontend, Node.js, PHP або Python для backend, що працює через веб-браузер. Суттєвою первагою такого рішення є доступність та легкість розгортання – використовується лише браузер, що знімає потребу у встановленні додаткових програм, оновлення здійснюється «автоматично» при завантаженні сторінки, що знижує технічну підтримку. Важливим фактором є також кросплатформеність – доступність на всіх ОС без додаткових компонентів. Гнучкість інтерфейсу дасть можливість створення адаптивного інтерфейсу та налаштування його під різні потреби користувачів. Недоліком може бути обмежений доступ до системних функцій – деякі операції, наприклад, робота з USB, можуть вимагати додаткових рішень. Також аке рішення залежне від підключення – хоча основні функції можуть бути локальними, вебдодаток потребує з’єднання для інтерактивності й доступу до оновлень [12-15].

Для даного проекту вибрана веб-платформа, яка забезпечує всі потреби ПЗ та не вимагає додаткових рішень. Доступ до програми, навіть для користувачів з обмеженими технічними знаннями, буде легким та зрозумілим. Веб-сторінка забезпечить легкий доступ та оновлення програмного забезпечення, а також дозволить гнучко інтегрувати необхідний функціонал.

3.3 Вибір програмних рішень

Вибір програмних рішень для заданої задачі потребує наступний функціонал – зчитування та передача даних через USB та зберігання даних, що надходять з контролера в реальному часі.

При розробці програми для взаємодії з USB є кілька можливих способів, серед яких:

– **Serial API** (браузерний) — дозволяє браузеру зчитувати та передавати дані через USB-порт, підходить для простих завдань без складного апаратного налаштування [16];

– **WebUSB API** — більш потужний API для роботи з USB-пристроями [17] і має більше налаштувань та більш підтримуваний [11] (рисунок 3.1).

Рисунок 3.1. Підтримка браузерами Serial API та Web USB API

Щодо зберігання даних, розглянемо кілька підходів:

– **localStorage** — простий метод зберігання невеликого обсягу даних в браузері, не підходить для великих або структурованих даних. Недоліком є те, що дані зберігаються у вигляді рядків тобто для зчитування та зберігання даних потрібно використовувати функції серіалізації наборів даних, також LocalStorage має обмеженн по об'єму зберігання даних (до 5 МБ);

– **IndexedDB** — браузерна база даних, яка дозволяє зберігати структуровані дані, ідеально підходить для автономної роботи з більшими даними, проте трохи складніша у використанні;

– **серверна база даних** (наприклад, MySQL, PostgreSQL) — підійде для обробки значних обсягів даних і забезпечує доступ з різних клієнтських пристроїв, але вимагає підтримки бекенду;

– **файлове сховище** (JSON, CSV) — дані можна зберігати у вигляді файлів на сервері, підходить для архівування або переносу даних, просте у використанні, але вимагає підтримки бекенду.

Серед перерахованих рішень приймаємо вибір на користь **LocalStorage,** який для невеликих даних без постійного підключення до мережі спрощує розробку і не потребує бекенду та підтримується всіма популярними браузерами (рисунок 3.2). Для складніших застосунків рекомендовано IndexedDB або серверну базу даних.

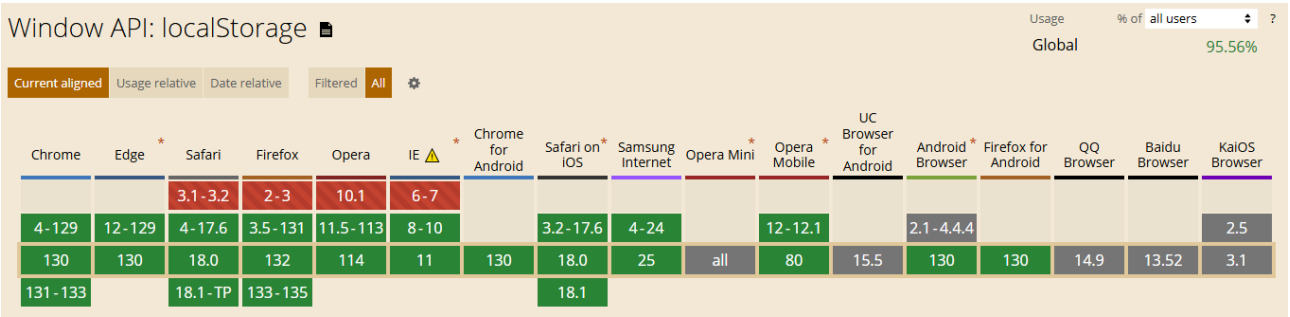

Рисунок 3.2. Підтримка localStorage браузерами

3.4 Вибір технологічного підходу для розробки клієнтської частини

Для розробки вебдодатка можна обрати різні підходи, включаючи написання на чистих HTML, JavaScript, і CSS або ж використання фреймворків, таких як React, Vue.js чи Angular. Кожен варіант має свої переваги і недоліки. Використання HTML, JavaScript і CSS забезпечує простоту і легкість старту без встановлення додаткових бібліотек чи фреймворків. Код залишається максимально гнучким і не залежить від сторонніх компонентів, що дозволяє

зберігати повний контроль над проєктом. Проте такий підхід підходить здебільшого для невеликих проєктів, оскільки при масштабуванні код стає громіздким і менш зручним для підтримки. Крім того, відсутність компонентної структури ускладнює організацію коду.

У цьому проєкті важливою вимогою є можливість налаштовувати та відображати різні параметри системи, які взаємопов'язані між собою. Так, будь-які зміни одного з налаштувань можуть впливати на інші параметри, а також змінювати спосіб виведення або обробки даних. Цей високий рівень взаємозалежності та необхідність динамічного оновлення даних у реальному часі потребують реактивної роботи компонентів користувацького інтерфейсу. Для такої інтерактивності й швидкого оновлення вмісту без перевантаження сторінки доцільно обрати сучасний фронтенд-фреймворк, що забезпечує реактивність. Це дозволяє створювати комплексний і гнучкий інтерфейс для зручного налаштування, візуалізації даних, а також спрощує підтримку та масштабування застосунку.

Серед відомих сучасних фреймворків можна виділити React, Vue.js та Angular.

Фреймворк React [13] відзначається високою продуктивністю завдяки віртуальному DOM і надає розвинену екосистему інструментів для розробки великих додатків. Компонентний підхід дозволяє створювати незалежні компоненти, які можна багаторазово використовувати, але для новачків він може здатися складним через специфічний синтаксис JSX.

Vue.js [12] є більш доступним і має пологішу криву навчання, ніж React або Angular, що робить його хорошим вибором для швидкого старту. Vue також підтримує вбудовані інструменти для маршрутизації і управління станом, але менш популярний серед великих корпорацій, тож кількість ресурсів і підтримки може бути обмеженою.

Angular [14] є всеосяжним рішенням для корпоративних проєктів, де важлива висока організованість і підтримка великого обсягу функціональності, але цей фреймворк вимагає значного навчання і конфігурації.

Розробка вебдодатку на базі Vue.js пропонує зручний баланс між швидкістю розробки і гнучкістю, тому у даному проекті прийнято рішення його використання.

3.5 Інтерфейс

Інтерфейс користувача вирішено розробляти на базі фреймворку Quasar [15] в основі якого лежить використання реактивного фреймворку Vue.js. Фреймворк Quasar є популярним серед розробників та має велику кількість інтуітивно зрозумілих та гнучко налаштовуваних візуальних компонентів, які підійдуть для більшості елементів інтерфейсу.

Вебдодаток включає три основні сторінки:

– головна сторінка (рисунок 3.3) вміщує список посилань на описання доступних лабораторних робіт;

– сторінка описання лабораторної роботи (рисунок 3.4) містить матеріал, який підвантажується в форматі html з сервера при відкритті чи оновленні сторінки;

– сторінка виконання лабораторної роботи (рисунок 3.5) містить елементи для управління процесом зчитування даних а також налаштування та візуальне супроводження даних.

Кожна сторінка вміщується в контейнер, що вже містить меню та верхню панель навігації. Функціонал також дозволяє показувати спливаючі повідомлення в зручному для перегляду форматі без блокування сторінки (як це роблять стандартні діалогові вікна).

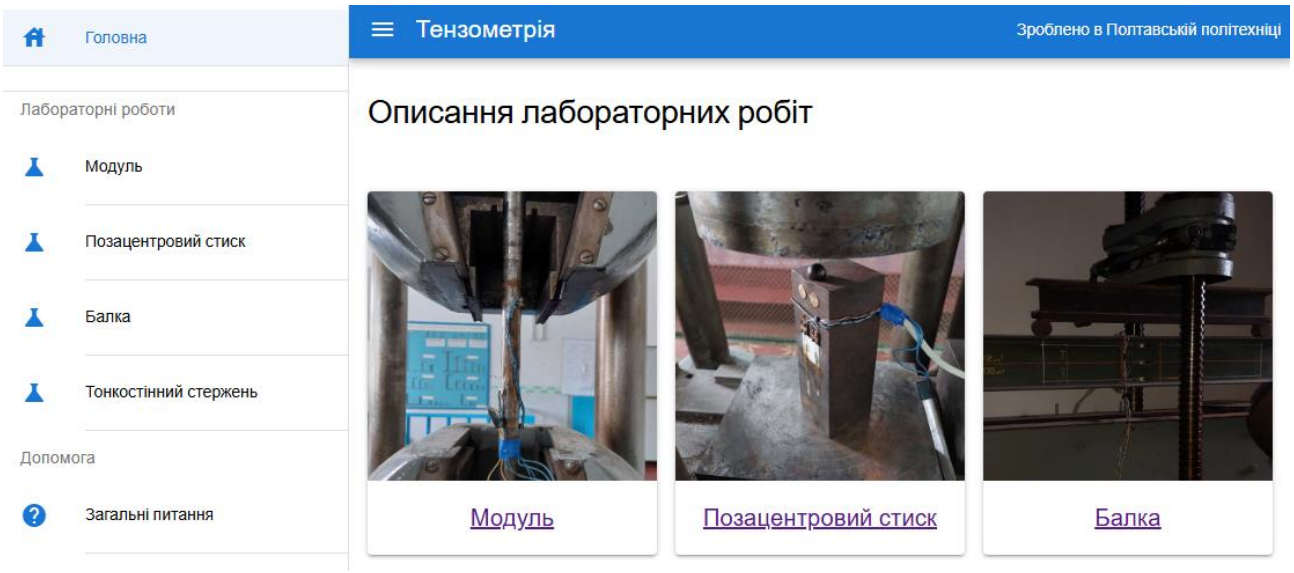

Рисунок 3.3. Головна сторінка

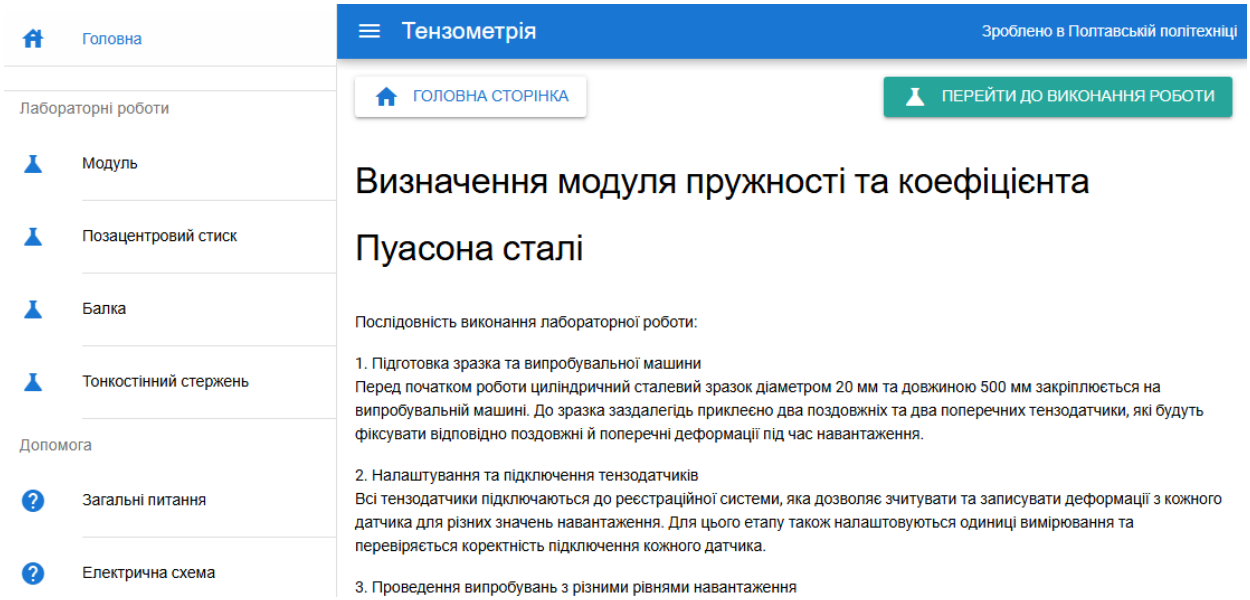

Рисунок 3.4. Сторінка описання лабораторної роботи

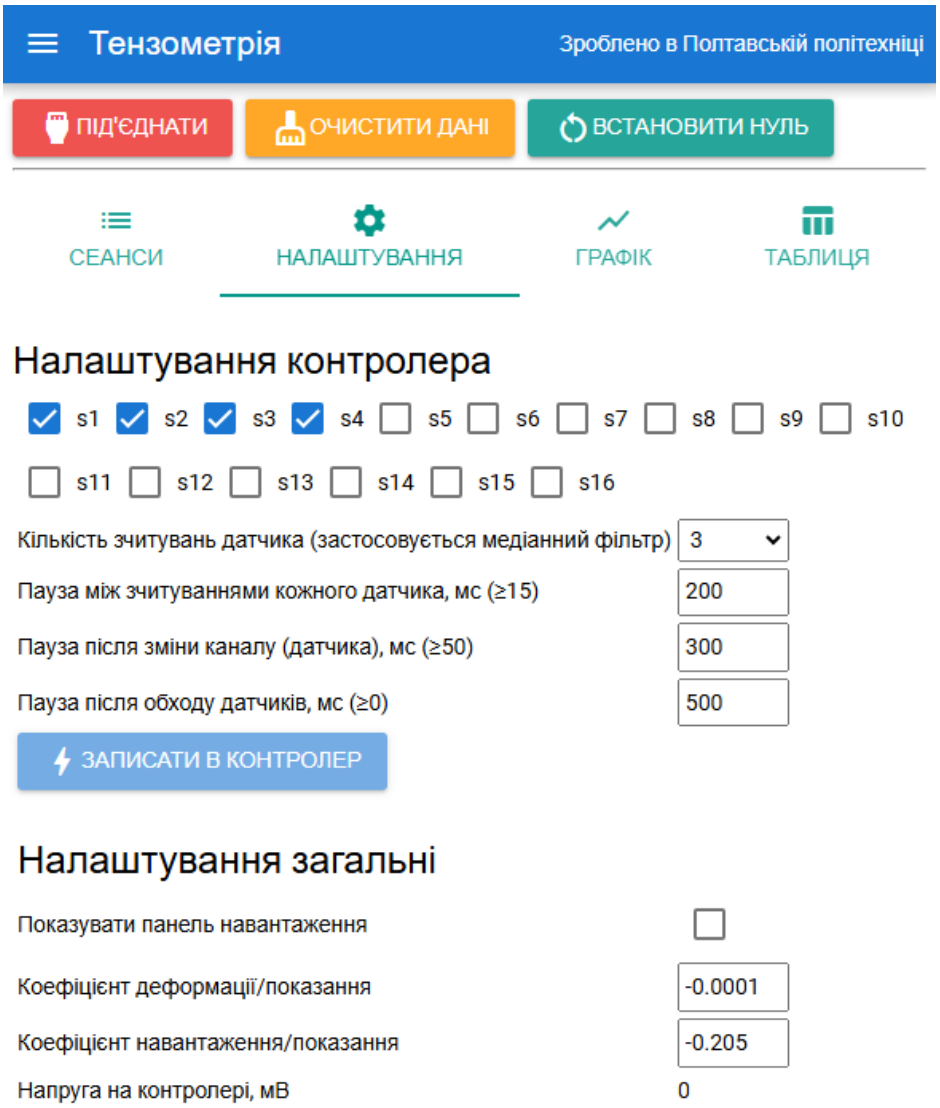

Рисунок 3.5. Сторінка виконання лабораторної роботи (налаштування)

На рисунку 3.5 представлено інтерфейс сторінки для управління параметрами контролера, зчитуванням показань датчиків і налаштуванням загальних параметрів.

Верхня панель кнопок активна завжди, вони реалізують наступні дії.

Під'єднати - ця кнопка ініціює з'єднання з контролером через USB використовуючи Web Serial API [16], який має деякі особливості в плані безпеки та взаємодії з користувачем. Після натискання кнопки браузер ініціює вікно вибору порту. Це зроблено для безпеки під час роботи з інтерфейсом: підключення до порту не може відбутися програмно, а лише після чіткої дії

користувача (механізм User Activation [19]). Браузер відкриває діалогове вікно, де користувач повинен вручну вибрати доступний порт для під'єднання. Ця процедура гарантує, що програмне забезпечення не отримує доступ до апаратного забезпечення без дозволу, знижуючи ризик несанкціонованого втручання або автоматичних підключень до потенційно небезпечних пристроїв.

Очистити дані - призначена для видалення усіх збережених даних, забезпечуючи "чистий" старт.

Встановити нуль - дозволяє обнулити показання, щоб відштовхуватися від нової базової точки для вимірювань. Базовими можуть бути не лише початкові показання, а й показання після кількох циклів зчитувань.

Зчитати 1 показання - зчитує одне наступне значення з датчика, без необхідності увімкнення автоматичного режиму зчитування.

Автоматичне зчитування (чекбокс) - вмикає режим автоматичного збору даних з датчиків. Через певний проміжок часу первіряється наповнення буфера порту і, при наявності даних, вони зчитуються у буфер програми.

Панель налаштувань контролера місьтить налаштування, які можуть бути записані в контролер шляхом натискання кнопки **"Записати в контролер".**

Вибір каналів (s1–s16) – набір чекблоксів, що дозволяє користувачеві вибирати один або кілька каналів для зчитування даних з датчиків.

Кількість зчитувань датчика - вказує, скільки зчитувань потрібно зробити для одного датчика (реалізується медіанний фільтр для підвищення точності).

Пауза між зчитуваннями кожного **датчика** - затримка між послідовними зчитуваннями одного датчика.

Пауза після зміни каналу (датчика) - час, необхідний для стабілізації сигналу після перемикання на інший датчик.

Пауза після обходу датчиків - час паузи перед повторним циклом зчитування всіх активних датчиків.

Панель загальних налаштувань стосується суто програми, на контролер ніякого впливу немає.

Показувати панель навантаження (чекбокс) - дозволяє відобразити або приховати додаткову панель, що показує поточне навантаження.

Коефіцієнт деформації/показання та **Коефіцієнт навантаження/показання** - ці поля налаштовують коефіцієнти для масштабування даних, які приходять з датчиків, для коректного відображення деформації та навантаження.

Напруга на контролері, мВ - поле для відображення поточної напруги на контролері, що необхідно для відслідковування коректної його роботи.

Блок "Повідомлення від контролера" показує останні повідомлення, що надійшли від контролера, допомагаючи відслідковувати стан системи та можливі помилки.

Таким чином, інтерфейс забезпечує користувача необхідними засобами для повного контролю за зчитуванням і налаштуванням параметрів під час лабораторних експериментів.

Вкладка "Сеанси" (рисунок 3.6) дає змогу запам’ятовувати набори зчитаних даних у внутрішньому сховищі браузера LocalStorage.

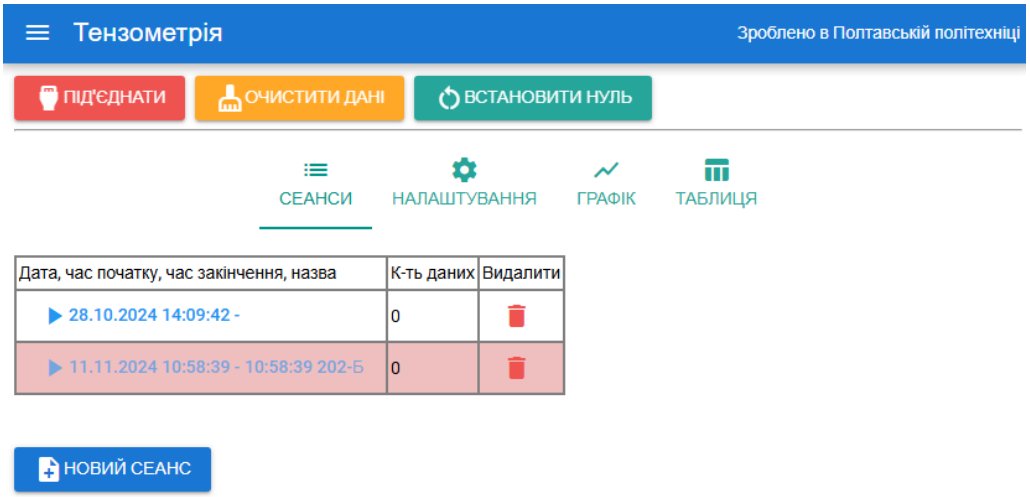

Рисунок 3.6. Вкладка "Сеанси"

Вкладка "Графік" (рисунок 3.7) містить графічне відображення отриманих даних. Реалізовано налаштування горизонтальної та вертикальної осей. При

цьому горизонтальну вісь можна показувати час випробування, у секундах, або ж номер вибірки. Також існує можливість цього налаштування у форматі навантаження. На вертикальній осі відображені деформації, але існує можливість показувати сирі дані, що надходять з контролера або ці ж дані відносно нульових показань. Показ сирих даних може бути корисним у тому випадку коли аналізується можливість виходу даних в процесі експерименту із допустимого діапазону вимірювань.

Рисунок 3.7. Вкладка "Графік"

Функціонал графіків реалізованого за допомогою JavaScript бібліотеки Chart.js [18].

Вкладка "Таблиця" відображує ті ж дані які подаються на графіку але в табличній формі. Це може бути корисно для подальшого аналізу результатів у сторонніх додатках.

3.6 Вибір механізмів для реалізації функціоналу вебдодатку

Для реалізації функціоналу у Vue 3 спочатку налаштовується основна структура компонентів та реактивність [12]. У Vue 3 використовується `Composition API`, що дозволяє більш гнучко керувати станом компонента та розподіляти логіку. Наприклад, можна створити функції для підключення та взаємодії з контролером, обробки даних і візуалізації, тощо.

Перевагою вибраного фреймворку є те, що при розробці маємо чітке розділення між HTML кодом і javascript кодом, що дозволяє легко читати код та змінювати функціонал вебдодатку.

Vue.js надає кілька зручних способів для створення подій і виклику функцій безпосередньо в HTML-шаблонах, що робить розробку інтуїтивною та швидкою. Основний спосіб — це використання **v-on** директиви або її скорочення **@**, яка дозволяє зв'язати події, такі як click, submit, input тощо, з методами або виразами.

Vue також підтримує передачу параметрів у функції безпосередньо в шаблоні. Наприклад, @click="handleClick(item)" викликає функцію handleClick і передає в неї об'єкт item. Це зручно для обробки динамічних даних. Крім цього, у Vue можна використовувати вбудовані модифікатори подій, такі як .prevent, .stop, .once, що дозволяють керувати поведінкою подій без необхідності написання додаткового коду. Наприклад, @submit.prevent="submitForm" автоматично зупинить стандартну поведінку події submit, запобігаючи перезавантаженню сторінки при надсиланні форми.

Vue.js також надає можливість умовного рендерингу та показу даних залежно від стану компоненту. Це можна зробити, використовуючи директиву v-if або v-show разом із функціями, що дозволяє більш гнучко управляти контентом у HTML-шаблоні.

Таким чином, Vue забезпечує багатий і зручний інструментарій для роботи з подіями та функціями безпосередньо у HTML, що значно спрощує процес розробки інтерактивних інтерфейсів.

У Vue.js для реалізації переходів між сторінками без фізичного перезавантаження використовується механізм Vue Router. Це спеціальний інструмент для маршрутизації в SPA (Single Page Application) [20] — додатках, де всі сторінки завантажуються динамічно, без перезавантаження всього документа. Vue Router [21] дозволяє визначати маршрути (шляхи) до різних компонентів і управляти переходами між ними, що значно покращує користувацький досвід і швидкість роботи додатку.

Перевагами використання Vue Router є швидкість переходів, оскільки сторінки не перезавантажуються, навігація між компонентами відбувається миттєво, забезпечуючи плавність інтерфейсу та збереження стану, тобто дані, зібрані чи змінені під час роботи з однією сторінкою, не втрачаються під час переходу на іншу сторінку, що особливо важливо для форм і багатокрокових процесів.

В даному проекті вирішено реалізувати динамічне підвантажування даних з сервера за потребою (налаштування окремих лабораторних робіт, їх описання), що має переваги. По-перше, такий підхід оптимізує швидкість завантаження вебдодатка, оскільки дозволяє підвантажувати лише потрібні дані за запитом, знижуючи навантаження на сервер, підвищуючи ефективність роботи. Це також забезпечує можливість оновлення й редагування змісту лабораторних робіт чи параметрів без оновлення всієї програми. По-друге, динамічне завантаження підвищує зручність для користувачів та створювати більш гнучкі інтерфейси: наприклад, користувач може перемикатися між налаштуваннями або переглядати описання лабораторій без повного перезавантаження сторінки, що робить роботу швидшою і зручнішою.

Загалом, динамічне підвантажування не лише покращує продуктивність, але й забезпечує кращу інтерактивність і можливість налаштування для кінцевих користувачів, що робить його важливим для інтуїтивних і швидкодіючих навчальних платформ.

3.7 Розробка макету вебдодатку та статичних сторінок

У процесі розробки вебдодатку ми дійшли до висновку, що серверна частина буде містити лише мінімальний набір статичних даних, які за потребою підвантажуються у клієнтську частину. Це такі дані, як опис лабораторних робіт, налаштування для кожної з них та їхній загальний перелік, зображення, тощо. При проєктуванні головної сторінки та загальної компоновки визначено, що під час завантаження будь-якої сторінки вебдодатку список лабораторних робіт має автоматично підвантажуватися в основному макеті. Це дозволяє забезпечити доступ до актуального переліку робіт на кожній сторінці, оптимізуючи загальну швидкість завантаження й підвищуючи зручність користувача. Всі налаштування маршрутів вебдодатку показані у додатку Б.

Список лабораторних робіт зберігається на сервері у форматі об'єктів JSON, який забезпечує структурування даних для зручного обміну між сервером та клієнтом. Також у форматі JSON зберігаються й налаштування кожної лабораторної роботи, що дозволяє легко обробляти їх у JavaScript. Опис кожної роботи зберігається в окремих HTML-файлах, що містять текст та розмітку, оптимально підготовлені для виведення на веб-сторінку.

Доступ до цих даних із веб-сторінки здійснюється за допомогою JavaScript та механізму XMLHttpRequest. Цей механізм дозволяє надсилати запити до сервера та отримувати відповіді без перезавантаження сторінки. XMLHttpRequest дозволяє вебдодатку асинхронно отримувати JSON-об'єкти з даними, обробляти їх у JavaScript, а також динамічно завантажувати HTML-контент із серверних файлів.

За такими підходами розроблені наступні сторніки:

- головна сторінка;
- макет сторінки з описаннями лабораторних робіт;
- сторінка «Help / Загальні запитання»;
- сторінка «Help / Електрична схема»;
- сторінка «Help / Скетч MCU».

Вихідний код макету вебдодатку та його статичних сторінок показаний у додатку В-Ж.

3.8 Розробка сторінки виконання лабораторної роботи

Сторінка для проведення лабораторної роботи містить чотири основні вкладки: "Сеанси", "Налаштування", "Графік", та "Таблиця". Також на сторінці розміщено набір кнопок для взаємодії з контролером та для відображення поточних зчитуваних даних. Код сторінки наведений у додатках Е, Є, Ж.

Під час завантаження сторінки виконується ініціалізація параметрів. Спочатку зчитуються налаштування базові налаштування даної лабораторної роботи, які оновлюються у вкладці "Налаштування" (рисунок 3.5).

Далі з локального сховища LocalStorage зчитуються дані про попередні проведені та збережені сеанси, після чого динамічно формується їх список (рисунок 3.6) у вкладці "Сеанси", вони активуються по кліку (функція activateSession, додаток З).

Налаштування кожного сеансу може бути різним, налаштування, отримані з сервера використовуються тільки як значеня за замовчуванням.

Після під'єднання до контролера активується кнопка "Записати в контролер", в результаті натискання якої налаштування передаються до контролера та автоматично ним застосовуються.

Веб-додаток надає змогу переглядати результати вже проведених лабораторних робіт без під'єднання до контролера, вкладки "Графік" та "Таблиця" (рисунок 3.6).

Взаємодія сторінки з контролером можлива лише після фізичного иа програмного (функція connect, додаток З) під'єднання контролера по натисканню кнопки "Під'єднати" та ініціалізації спливаючого модального вікна зі списком доступних портів.

Після успішної ініціалізації порту створюємо JavaScript об'єкт port та об'єкти для зчитування (reader) та запису (writer) даних у буфер вибраного порту.

При цьому активуються кнопка "Зчитати 1 показання" та галка "Автоматичне зчитування».

Мікроконтролер постійно пердає дані відповідно до налаштувань, вебдодаток зчитує ці дані з буфера порту, але не обов'язково додає їх у масив даних. Додавання даних відбувається лише за активності галки "Автоматичне зчитування» або після натискання кнопки "Зчитати 1 показання", яка відразу ж деактивується після надходження даних.

Для контролю періодичності та вигляду даних що надходять з контролера у вкладці налаштування передбачено візуалізація цих повідомлень (рисунок 3.8).

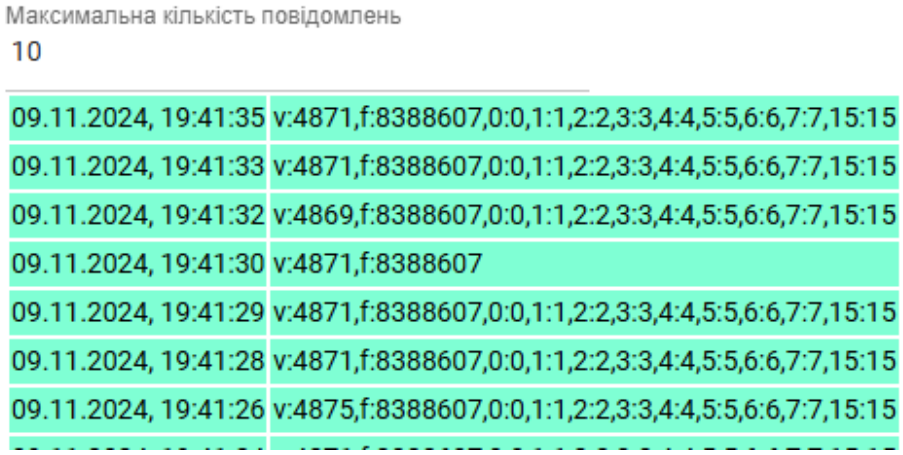

Рисунок 3.8. Повідомлення від контролера

При надходженні нових даних вони додаються до активного сеансу, в режимі реального часу оновлюється як таблиця з даними так і графік (фунція update, додаток З), що дуже зручно при виконанні експериментальних досліджень.

Дані зберігаються безпосередньо в тому ж форматі який приходить з мікроконтролера (всі показання - у вигляді цілочисельних даних). При цьому графік та таблиця даних будуються із врахуванням поточних налаштувань (параметрів осей X та Y), які можна змінювати в процесі випробування.

ВИСНОВОК

Макістерська кваліфікаційна робота на тему: «Розробка програмного забезпечення тензометричної станції для потреб лабораторій Полтавської політехніки» розроблена згідно із завданням.

Виконання даної роботи дозволило розробити програмне забезпечення тензометричної станції, яке відповідає потребам сучасних навчальних лабораторій. У процесі роботи було здійснено аналіз специфічних вимог до обладнання для навчальних закладів, зокрема необхідність адаптації до різних лабораторних робіт. У результаті було обрано відповідний мікроконтролер і вимірювальні компоненти, що забезпечують точність, надійність і швидкість зчитування даних.

Проєкт охоплює всі аспекти побудови тензометричної станції, включаючи апаратну частину, що складається з мікроконтролера Arduino, мультиплексора, вимірювальних схем і підключених датчиків. Описано процеси зчитування та способи фільтрації даних для підвищення точності, а також розроблено ефективний алгоритм обробки показань, який забезпечує сталість результатів.

Розробка програмного забезпечення була націлена на досягнення максимальної інтерактивності, інтуїтивності у використанні, а також можливості збереження даних у режимі реального часу. Зокрема, використання веб-платформи дозволяє легко інтегрувати систему в навчальні процеси, забезпечуючи доступ до станції через браузер і спрощуючи її обслуговування. Платформа забезпечує можливість візуалізації результатів експериментів у вигляді графіків і таблиць, що значно спрощує аналіз даних.

Таким чином, розроблена тензометрична станція відповідає вимогам науково-освітнього процесу та надає навчальним лабораторіям гнучке та зручне у використанні рішення, яке сприяє підвищенню якості проведення експериментів і поліпшенню навчального досвіду для студентів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Долгов М.А., Пискунов С.О. Міцність та руйнування елементів конструкцій. Частина 1. Фізичні основи міцності та використання тензометрії для визначення напруженого стану елементів конструкцій : Практикум. Навч. посіб. для здобувачів ступеня магістр за освітньою програмою «Динаміка і міцність машин» спеціальності 131 Прикладна механіка. – Київ: НТУУ «КПІ імені Ігоря Сікорського», 2022. – 44 с.

2. Основи метрології та електричних вимірювань. Частина І : конспект лекцій / В. В. Кухарчук – Вінниця : ВНТУ, 2020. – 148 с.

3. Метрологічне забезпечення вимірювань і контролю / [Є. Т. Володарський, В. В. Кухарчук, В. О. Поджаренко, Г. Б. Сердюк]. – Вінниця : ВДТУ, 2001. – 219 с.

4. Наявна матеріально-технічна база кафедри будівельних конструкцій https://nupp.edu.ua/page/materialno-tekhnichna-baza-kafedri-budivelnikh-konstruktsiy.html

5. Теорія сигналів та електричних кіл. Основи розрахунку електричних кіл: конспект лекцій / укладачі: О. М. Кобяков, О. В. Д'яченко, І. Є. Бражник. – Суми: Сумський державний університет, 2022. – 168 с.

6. PlatformIO. [Електронний ресурс]. – Режим доступу: https://platformio.org/. – Дата доступу: 01.12.2024.

7. Visual Studio Code. [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/. – Дата доступу: 01.12.2024.

8. Avia Semiconductor. Datasheet HX711 [Електронний ресурс]. – Режим доступу: https://www.alldatasheet.com/datasheet-pdf/view/1132222/AVIA/HX711.html. – Дата доступу: 01.11.2024.

9. Arduino. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.arduino.cc. – Дата доступу: 01.11.2024.

10. Texas Instruments. Datasheet CD74HC4067 [Електронний ресурс]. – Режим доступу: https://www.alldatasheet.com/datasheet-pdf/view/27059/

TI/CD74HC4067.html. – Дата доступу: 01.11.2024.

11. Can I use. Підтримка веб-функцій у браузерах [Електронний ресурс]. – Режим доступу: https://caniuse.com. – Дата доступу: 01.09.2024.

12. Vue.js. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://vuejs.org. – Дата доступу: 01.10.2024.

13. React.js. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://reactjs.org. – Дата доступу: 01.10.2024.

14. Angular. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://angular.dev. – Дата доступу: 01.10.2024.

15. Quasar Framework. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://quasar.dev. – Дата доступу: 01.11.2024.

16. MDN Web Docs. Web Serial API [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Web_Serial_API. – Дата доступу: 01.12.2024.

17. MDN Web Docs. Web USB API [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/USB. – Дата доступу: 01.12.2024.

18. Chart.js. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.chartjs.org/. – Дата доступу: 01.12.2024.

19. MDN Web Docs. User Activation [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/UserActivation. – Дата доступу: 11.12.2024.

20. MDN Web Docs. Glossary: Single Page Application (SPA) [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/SPA. – Дата доступу: 01.12.2024.

21. Vue Router. Офіційна документація [Електронний ресурс]. – Режим доступу: https://router.vuejs.org/. – Дата доступу: 01.12.2024.

Додаток А
Вихідний код прошивки контролера

```
/*
// command list
ch=number - channels bit mask
ta=number - time after cycle of reading, ms
rn=number - read number for each channel
tb=number  - time between read of channels
on=number - 1 / 0 - read / pause
*/
#include <EEPROM.h>
#include <iarduino_VCC.h>

const int maxchannels=16;

uint32_t val[maxchannels];
const int muxPin[] = {4,5,6,7};
const int muxOnPin=3;

const int gain=1; // 3 - 64, channel A  // 1 - 128, channel A // 2 - 32, channel B

const int doutPin = 12;
const int sckPin = 11; //mcu > HX711 sck pin

const int doutPinF = 9;
const int sckPinF = 10; //mcu > HX711 sck pin

uint32_t channels=1; // bit mask
uint16_t readNumber=1;
uint16_t timeAfter=1000;
uint16_t timeCheck=500;
uint16_t timeBetween=120;
uint16_t timeForce=1000;

uint32_t A[9];
int ArrayCompare(const uint32_t *AFirst, const uint32_t *ASecond) {
   if (*AFirst < *ASecond) return -1;
   return (*AFirst == *ASecond) ? 0 : 1;
}

uint32_t medianFilter(unsigned long *AA,int rn){
  qsort(AA, readNumber, sizeof(AA[0]), [](const uint32_t *AFirst, const uint32_t *ASecond) ->
uint32_t{ return -ArrayCompare(AFirst, ASecond); });
 return AA[(int)floor(rn/2)];
}

String showSettings(){
 return "SETT:1,ch=" + String(channels) + ",rn=" + String(readNumber)+",ta=" + String(timeAfter)
+ ",tc="+String(timeCheck)+",tb="+String(timeBetween)+",tf="+String(timeForce)+";";
}
```

```
void setup(void)
{
  Serial.begin(9600);

  for(int i = 0; i < 4; i ++)pinMode(muxPin[i], OUTPUT);

  pinMode(doutPin, INPUT);
  pinMode(sckPin, OUTPUT);
  // HX711 powerUp
  digitalWrite(sckPin, LOW);

  pinMode(doutPinF, INPUT);
  pinMode(sckPinF, OUTPUT);
  // HX711 powerUp
  digitalWrite(sckPinF, LOW);

  pinMode(muxOnPin, OUTPUT);
  digitalWrite(muxOnPin, HIGH);

  uint16_t data[7];


  EEPROM.get(0, data);
  channels=data[0]+data[1]*65536;
  timeAfter=data[2];
  timeBetween=data[3];
  timeCheck=data[4];
  readNumber=data[5];
  timeForce=data[6];
  Serial.println(showSettings());
}


uint32_t read(int8_t gn,bool force=0)  //read 24 bit data, store in dataset and start the next conversion
{
  uint32_t data = 0;
  uint8_t dout;

  int sck = force ? sckPinF :  sckPin;
  int dt = force ? doutPinF :  doutPin;

  //if(SCK_DISABLE_INTERRUPTS)
  noInterrupts();

  for (uint8_t i = 0; i < (24 + gn); i++)
  {   //read 24 bit data + set gain and start next conversion
    digitalWrite(sck, 1);

    //if(SCK_DELAY)
    delayMicroseconds(1); // could be required for faster mcu's, set value in config.h
    digitalWrite(sck,0);
    if (i < (24))
```

```
    {
      dout = digitalRead(dt);
      data = (data << 1) | dout;
    } else {
      //if(SCK_DELAY)
      delayMicroseconds(1); // could be required for faster mcu's, set value in config.h
    }
  }

  ///if(SCK_DISABLE_INTERRUPTS)
  interrupts();


  /*
  The HX711 output range is min. 0x800000 and max. 0x7FFFFF (the value rolls over).
  In order to convert the range to min. 0x000000 and max. 0xFFFFFF,
  the 24th bit must be changed from 0 to 1 or from 1 to 0.
  */
  data = data ^ 0x800000; // flip the 24th bit

  return data;
}

void cmdInput(){

  String cmd = Serial.readString();
  cmd.trim();

    /*for(int i=0;i<10;i++){
      digitalWrite(13, i%2);delay(50); // LCD
    }*/
    while(cmd.length()){
      String c=cmd.substring(0,2);
      cmd=cmd.substring(3);
      int i=cmd.indexOf(";");
      uint16_t v=cmd.substring(0,i).toInt();

      //Serial.println(String(c)+"="+String(v));

      if(c=="ch"){
        channels=strtoul(cmd.substring(0,i).c_str(),NULL,0);
        EEPROM.put(0, channels);
      }
      if(c=="ta"){
        timeAfter=v;
        EEPROM.put(4, v);
      }
      if(c=="tb"){
        timeBetween=v;
        EEPROM.put(6, v);
      }
      if(c=="tc"){
```

```
      timeCheck=v;
      EEPROM.put(8, v);
     }
     if(c=="rn"){
      readNumber=v;
      EEPROM.put(10, v);
     }
     if(c=="tf"){
      timeForce=v;
      EEPROM.put(12, v);
     }
     if(c=="ss"){
      Serial.print(showSettings());
     }
     cmd=cmd.substring(i+1);
    }
    Serial.println(showSettings());
}

String showVF(){
  const int rn=5;
  uint32_t B[rn];
  for(int i=0;i<rn;i++){
    B[i]=read(1,1);
    delay(50);
  }
  return String("v:")+int(1000*analogRead_VCC())+",f:"+medianFilter(B,rn);
}

unsigned long tf=millis();
unsigned long ts=millis();
int j=0;
int channel = 0;
String REZ="";

void nextChannel(){
  for(channel++;channel<maxchannels && !bitRead(channels, channel);channel++);
  setChannel(channel);
}
void loop() {
  if (Serial.available() > 0){
    cmdInput();
    return;
  }

  // show voltage & force;
  if( millis() > tf){
    Serial.println(showVF()+";");
    tf = timeForce + millis();
    return;
  }
  if( millis() < ts ) return;
```

```
  if(channel>=maxchannels){
    Serial.println(showVF()+REZ+";");
    ts = timeAfter + millis();
    channel=0;
    REZ="";
    return;
  }

  if(j>=readNumber){ // next channel
      j=0;
      REZ+=","+String(channel)+":"+String(medianFilter(A,readNumber));
      ts = millis() + timeCheck;
      nextChannel();
      return;
  }
  // next read
  A[j]=channel;//read(gain);
  ts = millis() + timeBetween;
  j++;
  return;
}
int muxChannel[16][4]={
    {0,0,0,0}, //channel 0
    {1,0,0,0}, //channel 1
    {0,1,0,0}, //channel 2
    {1,1,0,0}, //channel 3
    {0,0,1,0}, //channel 4
    {1,0,1,0}, //channel 5
    {0,1,1,0}, //channel 6
    {1,1,1,0}, //channel 7
    {0,0,0,1}, //channel 8
    {1,0,0,1}, //channel 9
    {0,1,0,1}, //channel 10
    {1,1,0,1}, //channel 11
    {0,0,1,1}, //channel 12
    {1,0,1,1}, //channel 13
    {0,1,1,1}, //channel 14
    {1,1,1,1}  //channel 15
};
void setChannel(int channel){
  digitalWrite(muxOnPin, 1);
  delay(10);
  for(int i = 0; i < 4; i ++){
    digitalWrite(muxPin[i], muxChannel[channel][i]);
  }
  delay(10);
  digitalWrite(muxOnPin, 0);
  return;
}
```

Додаток Б
Маршрути вебдодатку

```
const routes = [
  {
    path: "/",
    component: () => import("layouts/MainLayout.vue"),
    children: [
      { path: "", component: () => import("pages/IndexPage.vue") },
      {
        path: "/:lab",
        component: () => import("pages/LabPage.vue"),
      },
      {
        path: "/description/:lab",
        component: () => import("pages/DescriptionPage.vue"),
      },
    ],
  },
  {
    path: "/:catchAll(.*)*",
    component: () => import("pages/ErrorNotFound.vue"),
  },
];

export default routes;
```

Додаток В
Код макету вебдодатку

```
<template>
  <q-layout view="lHh Lpr lFf">
    <q-header elevated>
      <q-toolbar>
        <q-btn
          flat
          dense
          round
          icon="menu"
          aria-label="Menu"
          @click="toggleLeftDrawer"
        />

        <q-toolbar-title
          ><router-link to="/">Тензометрія</router-link>
        </q-toolbar-title>

        <div>Зроблено в Полтавській політехніці</div>
      </q-toolbar>
    </q-header>

    <q-drawer v-model="leftDrawerOpen" show-if-above bordered>
      <q-list bordered>
        <q-item clickable v-ripple to="/">
          <q-item-section avatar>
            <q-icon color="primary" name="house" />
          </q-item-section>

          <q-item-section>Головна</q-item-section>
        </q-item>
        <q-separator />
        <q-item class="title">
          <q-item-section avatar>
            <q-icon color="primary" name="science" />
          </q-item-section>

          <q-item-section>Лабораторні роботи</q-item-section>
        </q-item>
        <q-separator />

        <q-item
          clickable
          v-ripple
          v-for="(lab, i) in lablist"
          :key="i"
          :to="'/' + lab.url"
        >
          <q-item-section avatar>
            <q-icon color="primary" name="info" />
```

```
          </q-item-section>
          <q-item-section>{{ lab.name }}</q-item-section>
        </q-item>
      </q-list>
    </q-drawer>

    <q-page-container>
      <router-view />
    </q-page-container>
  </q-layout>
</template>

<script setup>
import { ref, provide } from "vue";
import EssentialLink from "components/EssentialLink.vue";

defineOptions({
  name: "MainLayout",
});

const leftDrawerOpen = ref(false);

function toggleLeftDrawer() {
  leftDrawerOpen.value = !leftDrawerOpen.value;
}

const url = "/api/lablist.json";
const request = new XMLHttpRequest();
request.open("GET", url, false);
request.send(null);

const lablist = ref(JSON.parse(request.responseText));
provide("lablist", lablist);
</script>

<style>
.router-link-active {
  text-decoration: none;
  color: white;
}
.title {
  background-color: rgb(192, 192, 192);
}
</style>
```

Додаток Д
Код головної сторінки

```
<template>
  <q-page class="">
    <div class="q-pa-md row items-start q-gutter-md">
      <router-link
        v-for="(lab, i) in labs"
        :key="i"
        :to="'description/' + lab.url"
      >
        <q-card class="my-card">
          <img :src="`/img/` + lab.url +`.jpg`" />

          <q-card-section>
            <div class="text-h6">{{ lab.title }}</div>
            <div class="text-subtitle2">{{ lab.name }}</div>
          </q-card-section>
        </q-card>
      </router-link>
    </div>
  </q-page>
</template>

<script setup>
import { ref, onMounted, inject } from "vue";

const labs = inject("lablist");
</script>

<style lang="sass" scoped>
.my-card
  width: 100%
  max-width: 250px
</style>
```

Додаток Е
Код сторінки описання лабораторної роботи

```
<template>
  <q-page><div v-html="content"></div> </q-page>
</template>

<script setup>
import { ref, watch } from "vue";
import { useRoute } from "vue-router";

const content = ref("");

const route = useRoute();

function load() {
  const url ="/api/" +location.href.split("/").at(-1) +".html";
  const request = new XMLHttpRequest();
  request.open("GET", url, false);
  request.send(null);
  content.value = request.responseText;
}

load();

// Відстежуємо зміну URL
watch(
  () => route.fullPath,
  (newUrl) => {
    load();
  }
);
</script>
```

Додаток Ж

Код дизайну сторінки виконання лабораторної роботи

```
<template>
  <q-page class="page">
    <div class="force" v-show="set.showForce">
      Навантаження <b>{{ Math.round(force) }}</b> Н
          <input v-model="notes" class="notes" />
      <hr />
    </div>

    <div>
      <div class="row q-gutter-sm">
        <q-btn @click="connect" icon="settings_input_hdmi" color="red-5"
          >Під'єднати</q-btn
        >
        <template v-if="portIsOpened">
          <q-checkbox v-model="reading">Автоматичне зчитування</q-checkbox>
          <q-btn
            @click="readingOne = !readingOne"
            :disable="reading"
            :color="readingOne ? 'red' : 'blue-5'"
          >
            <q-icon name="sensors" v-show="!readingOne" />
            <q-icon name="downloading" v-show="readingOne" />
            Зчитати 1 показання</q-btn
          >
        </template>
        <template v-if="sess.active.value != -1">
          <q-btn @click="clearData()" icon="cleaning_services" color="orange-5"
            >Очистити дані</q-btn
          >
          <q-btn
            icon="restart_alt"
            @click="setBase = !setBase"
            :color="setBase ? 'red' : 'teal-5'"
            >Встановити нуль</q-btn
          >
        </template>
      </div>
    </div>
    <hr />
    <q-tabs v-model="tab" class="text-teal">
      <q-tab name="sessions" icon="list" label="Сеанси" />
      <q-tab
        name="settings"
        icon="settings"
        label="Налаштування"
        v-show="sessOn"
      />
      <q-tab name="chart" icon="show_chart" label="Графік" v-show="sessOn" />
      <q-tab name="table" icon="table_chart" label="Таблиця" v-show="sessOn" />
```

```
</q-tabs>

<div v-show="tab == 'settings' && sessOn" class="settings">
 <div class="row">
  <div class="col">
   <h5>Налаштування контролера</h5>
  </div>
 </div>
 <div>
  <SelectSet v-model="sensorsBitmask" :n="sensorNumber" />
 </div>

 <div>
  <table>
   <tbody>
    <tr>
     <td>
      Кількість зчитувань датчика (застосовується медіанний фільтр)
     </td>
     <td>
      <select v-model.number="set.mcu.rn">
       <option v-for="i in 5" :key="i">{{ i * 2 - 1 }}</option>
      </select>
     </td>
    </tr>
    <tr>
     <td>Пауза між зчитуваннями кожного датчика, мс (&ge;15)</td>
     <td><input v-model.number="set.mcu.tb" /></td>
    </tr>
    <tr>
     <td>Пауза після зміни каналу (датчика), мс (&ge;50)</td>
     <td><input v-model.number="set.mcu.tc" /></td>
    </tr>
    <tr>
     <td>Пауза після обходу датчиків, мс (&ge;0)</td>
     <td><input v-model.number="set.mcu.ta" /></td>
    </tr>

    <tr>
     <td>
      <q-btn
       icon="bolt"
       color="primary"
       @click="write2controller"
       :disabled="!portIsOpened"
       >Записати в контролер</q-btn
      >
     </td>
     <td></td>
    </tr>

    <tr>
```

```
          <td colspan="2"><h5>Налаштування загальні</h5></td>
          <td></td>
        </tr>
        <tr>
          <td>Показувати панель навантаження</td>
          <td><q-checkbox v-model.number="set.showForce"></q-checkbox></td>
        </tr>
        <tr>
          <td>Коефіцієнт деформації/показання</td>
          <td><input v-model.number="set.strainFactor" /></td>
        </tr>
        <tr>
          <td>Коефіцієнт навантаження/показання</td>
          <td><input v-model.number="set.forceFactor" /></td>
        </tr>
        <tr>
          <td>Напруга на контролері, мВ</td>
          <td>{{ voltage }}</td>
        </tr>
      </tbody>
    </table>
  </div>
  <div class="col">
    <h5>Повідомлення від контролера</h5>
  </div>
  <div class="q-gutter-y-md column" style="max-width: 300px">
    <q-input
      v-model.number="msg.size"
      label="Максимальна кількість повідомлень"
      width="200px"
      :dense="dense"
    />
  </div>
  <table>
    <tbody>
      <tr v-for="(d, z) in msg.list" :key="z" :class="'msg-' + d[2]">
        <td>
          {{ d[0] }}
        </td>
        <td>
          {{ d[1] }}
        </td>
      </tr>
    </tbody>
  </table>
</div>

<div style="width: auto" v-show="tab == 'chart' && sessOn">
  <canvas id="myChart"></canvas>
  <div class="row">
    <div class="col">
      Вісь X
```

```
      <select v-model="set.axisX">
        <option value="index">Відлік</option>
        <option value="time">Час, сек</option>
        <option value="load">Навантаження</option>
      </select>
      <input size="4" v-model.number="margins[0]" />
      ...
      <input size="4" v-model.number="margins[1]" />
    </div>
    <div class="col">
      Вісь Y
      <select v-model="set.axisY">
        <option value="strain">Деформації</option>
        <option value="data">Дані датчика відносні</option>
        <option value="raw">Дані датчика абсолютні</option>
      </select>
      <input size="4" v-model.number="margins[2]" />
      ...
      <input size="4" v-model.number="margins[3]" />
    </div>
  </div>
</div>
<div v-show="tab == 'table' && sessOn">
  <table class="list-table">
    <thead>
      <tr>
        <td></td>
        <td>Час, сек</td>
        <td>Навантаження, Н</td>
        <td v-for="i in sensorNumber" v-show="sensors[i - 1]" :key="i">
          s{{ i }}
        </td>
      </tr>
    </thead>
    <tbody>
      <tr v-for="(d, z) in data" :key="z">
        <td>{{ d.note }}</td>
        <td>{{ resultTime(z.toFixed, d.time) }}</td>
        <td>{{ resultLoad(z) }}</td>
        <td v-for="i in sensorNumber" v-show="sensors[i - 1]" :key="i">
          {{ result(i - 1, d, z) }}
        </td>
      </tr>
    </tbody>
  </table>
</div>

<div v-show="tab == 'sessions'">
  <br />
  <table class="list-table">
    <tbody>
      <tr>
```

```
          <td>Дата, час початку, час закінчення, назва</td>
          <td>К-ть даних</td>
          <td>Видалити</td>
        </tr>
        <tr
          v-for="(d, z) in sess.val.value"
          :key="z"
          :class="'isactive-' + (z == sess.active.value)"
        >
          <td>
            <q-btn
              flat
              :disabled="z == sess.active.value"
              icon="play_arrow"
              @click="activateSession(z)"
              color="blue"
              >{{ sess.time(z) }} {{ d.name }}</q-btn
            >
          </td>
          <td>{{ d.data.length }}</td>

          <td>
            <q-btn
              flat
              icon="delete"
              @click="sess.remove(z)"
              color="red-5"
            ></q-btn>
          </td>
        </tr>
      </tbody>
    </table>
    <br />

    <q-banner class="bg-red text-white" v-show="sess.active.value == -1">
      Для початку роботи виберіть існуючий або створіть новий сеанс
    </q-banner>

    <br />
    <q-btn @click="createSession" icon="note_add" color="blue-8"
      >Новий сеанс</q-btn
    >
  </div>
</q-page>
</template>
```

Додаток З

JavaScript код сторінки виконання лабораторної роботи

```
<script setup>
import { ref, onMounted, inject, watch, computed } from "vue";
import { useQuasar } from "quasar";
import { useRoute } from "vue-router";
const $q = useQuasar();

const labs = inject("lablist");

import SelectSet from "components/SelectSet.vue";

const sensorsBitmask = ref(5);

const route = useRoute();

function notify(message, icon = "announcement", color = "black") {
  $q.notify({
    message,
    icon,
    color,
  });
}

// sessions
import { Sess } from "./sess.js";
const sess = new Sess();
sess.load();

function activateSession(z) {
  reading.value = false;
  sess.active.value = z;
  let d = sess.activeSess();
  data.value = d.data;
  baseIndex = d.baseIndex;
  set.value = d.sett;
  sensorsBitmask.value = d.sett.mcu.ch;
  if (d.data.length) update();
  notify("Відкрито збережений сеанс та його налаштування", "check", "blue");
}

function createSession() {
  let name = prompt("Назва нового сеансу", "");
  if (name === null) return;
  clearData(1);
  sess.create(set.value, [], name);
  notify("Створено новий сеанс", "check", "green");
}
//
```

```
const margins = ref([0, 0, 0, 0]);

import Chart from "chart.js/auto";

let chart;

let force = ref(0);
let voltage = ref(0);

const sensorNumber = 16;
const arr = [];
for (let i = 0; i < sensorNumber; i++) arr.push(0);

const data = ref([]);

const notes = ref("");

let baseIndex = 0;
const setBase = ref(false);

const set = ref({
  mcu: {
    ch: 5,
    rn: 3,
    tb: 50,
    tc: 150,
    ta: 1000,
  },
  strainFactor: 1,
  forceFactor: 1,
  axisX: "time", // index/time/force/
  axisY: "strain", // strain/raw
  showForce: true,
});

const tab = ref("sessions");

const sensors = ref([]);
const sensorNames = ref([]);

for (let i = 0; i < sensorNumber; i++) {
  sensors.value[i] = i < 3;
  sensorNames.value[i] = "sensor " + (i + 1);
}

//////////////////////

const clearData = (hard = 0) => {
  if (!hard) {
    if (!confirm("Дійсно бажаєте очистити всі дані поточного сеансу?")) return;
  }
  data.value = [];
```

```
  baseIndex = 0;
  sess.save({
    data: [],
    baseIndex: 0,
  });
  update();
};

function result(i, v, j) {
  let r = v.val[i];
  if (set.value.axisY != "raw") r -= data.value[baseIndex].val[i];
  if (set.value.axisY == "strain") r *= set.value.strainFactor;
  return r;
}
function resultTime(i, v) {
  if (!data.value[baseIndex]) return v;
  return Math.round((v - data.value[baseIndex].time) / 1000);
}
function resultLoadV(v) {
  if (!data.value[baseIndex]) return 0;
  return (v - data.value[baseIndex].load) * set.value.forceFactor;
}
function resultLoad(i) {
  return (
    (data.value[i].load - data.value[baseIndex].load) * set.value.forceFactor
  );
}

const update = function () {
  let b = margins.value[0] - margins.value[1] != 0;
  chart.options.scales.x.min = b ? margins.value[0] : undefined;
  chart.options.scales.x.max = b ? margins.value[1] : undefined;
  b = margins.value[2] - margins.value[3] != 0;
  chart.options.scales.y.min = b ? margins.value[2] : undefined;
  chart.options.scales.y.max = b ? margins.value[3] : undefined;
  //chart.options.scales.x.type = set.value.axisX == "time" ? "time" : "linear";

  chart.data.labels = data.value.map(
    (v, i) =>
      (set.value.axisX == "time") * resultTime(i, v.time) +
      (set.value.axisX == "load") * resultLoad(i) +
      (set.value.axisX == "index") * i
  );
  chart.data.datasets = arr.map((a, i) => {
    return {
      label: "s" + (i + 1),
      data: data.value.map((v, j) => result(i, v, j)),
      hidden: !sensors.value[i],
    };
  });
  chart.update();
};
```

```
let push = function (dt) {
  if (sess.active.value == -1) return;
  dt = dt.trim().split(",");
  let v = [...arr];
  for (let i = 1; i < dt.length; i++) {
    let t = dt[i].split(":");
    v[+t[0]] = +t[1];
    //v[i - 1] = +t[0];
  }

  data.value.push({
    time: Date.now(),
    load: +dt[0].split(":")[1],
    note: notes.value,
    val: v,
  });
  if (setBase.value) {
    baseIndex = data.value.length - 1;
    setBase.value = false;
  }
  sess.save({ data: data.value, baseIndex, sett: set.value });

  update();
};

onMounted(() => {
  chart = new Chart(document.getElementById("myChart"), {
    type: "line",
    options: {
      animation: false,
      plugins: {
        legend: {
          //display: false,
        },
        tooltip: {
          enabled: false,
        },
      },
      scales: {
        x: {
          type: "linear",
          //beginAtZero: true,
        },
        y: {
          //beginAtZero: true,
        },
      },
    },
    data: {
      labels: [],
      datasets: arr.map((a, i) => {
```

```
        return {
          label: "sens" + (i + 1),
          data: [2, 5, 4 + i],
        };
      }),
    },
  });

  window.chart = chart;

  if ("serial" in navigator) {
    // The Web Serial API is supported.
    //this.cycle();
  } else {
    alert("Web Serial API is NOT SUPPORTED in your browser");
  }
});

//////////////////  log
const msg = ref({ size: 10, list: [] });
const log3 = (t, status = true) => {
  msg.value.list.unshift([new Date().toLocaleString(), t, status]);
  if (msg.value.list.length > msg.value.size)
    msg.value.list.length = msg.value.size;
};

//////////////////////

watch(
  () =>
    [
      set.value.axisX,
      set.value.axisY,
      set.value.forceFactor,
      set.value.strainFactor,
    ].join(),
  (newset, oldset) => {
    update();
  }
);

watch(
  () => sensors.value.join(),
  (newset, oldset) => {
    update();
  }
);

watch(
  () => margins.value.join(","),
  (newset, oldset) => {
    update();
```

```
  }
);

function isBitOn(number, index) {
  return Boolean(number & (1 << index));
}

watch(
  () => sensorsBitmask.value,
  (newset, oldset) => {
    set.value.mcu.ch = sensorsBitmask.value;
    sensors.value = arr.map((v, i) => isBitOn(sensorsBitmask.value, i));
  }
);

watch(
  () => set,
  (newset, oldset) => {
    if (sess.active.value == -1) return;
    sess.save({
      sett: set.value,
    });
  },
  { deep: true }
);

watch(
  () => route.fullPath,
  (newUrl) => {
    sess.load();
  }
);

//////////////////////////////
const reading = ref(false);
const readingOne = ref(false);

const decoder = new TextDecoder();
const encoder = new TextEncoder();

let port,
  reader,
  writer,
  portIsOpened = ref(false);
let writeBuffer = "";
const buffer = ref("");

const connect = () => {
  notify("Запит на приєднання через USB", "check", "green");

  navigator.serial
    .requestPort()
```

```
    .then((rport) => {
      port = rport;
      log3("port is requested");
      rport
        .open({ baudRate: 9600 })
        .then(() => {
          log3("port is opened");
          notify("Порт відкрито", "check", "green");
          portIsOpened.value = true;
          cycle();
        })
        .catch((e) => {
          log3("port is NOT opened", false);
          notify("Помилка приєднання до порту", "error", "red");
        });
    })
    .catch((e) => {
      log3("port is NOT requested", false);
      notify("Приєднання до порту відхилено", "warning", "orange");
    });
};

let block = false;
async function cycle() {
  setTimeout(() => {
    cycle();
  }, 2000);

  if (!port) return;
  // write
  if (writeBuffer != "") {
    block = true;
    if (port.readable?.locked) {
      reader.cancel();
    }
    if (!port.writable?.locked) {
      // set write mode
      writer = port.writable.getWriter();
    } else {
      // write
      const data = new Uint8Array(encoder.encode(writeBuffer));
      log3("WRITE:" + writeBuffer);
      await writer.write(data);
      writeBuffer = "";
      writer.releaseLock();
    }
    block = false;
    return;
  }
  if (block) return;
  //check port
  if (!port.readable) {
```

```
    log3("port is not readable", false);
    return;
  }
  //read
  if (!port.readable?.locked) {
    // set read mode
    reader = port.readable.getReader();
  } else {
    await read();
  }
  block = false;
  return;
}

async function read() {
  try {
    while (true) {
      const { value, done } = await reader.read();
      if (done) {
        break;
      }
      buffer.value += decoder.decode(value);
      let c = buffer.value.split(";");
      while (c.length > 1) {
        answer(c.shift());
      }
      buffer.value = c[0];
    }
  } catch (error) {
    console.error(error);
  } finally {
    reader.releaseLock();
  }
}

function answer(str) {
  str = str.trim();
  log3(str);
  let ar = str.split(",");
  if (str.indexOf("SETT:") === 0) {
    let stt = {};
    ar.forEach((v) => {
      v = v.split("=");
      stt[v[0]] = +v[1];
    });
    set.value.mcu.rn = stt.rn;
    set.value.mcu.ta = stt.ta;
    set.value.mcu.tb = stt.tb;
    set.value.mcu.tc = stt.tc;
    sensorsBitmask.value = stt.ch;
    notify("Поточні налаштування зчитані з контролера", "check", "blue");
    return;
```

```
  }
  if (str.indexOf("v:") !== 0) return;
  force.value = resultLoadV(ar[1].split(":")[1]);
  voltage.value = ar[0].split(":")[1];
  ar.shift();
  if (ar.length < 2) return;
  if (reading.value) {
    push(ar.join(","));
  } else {
    if (readingOne.value) {
      push(ar.join(","));
      readingOne.value = false;
    }
  }
}

function write2controller() {
  writeBuffer = "";
  for (let k in set.value.mcu) {
    writeBuffer += k + "=" + set.value.mcu[k] + ";";
  }
}

let sessOn = computed(() => sess?.active?.value != -1);
</script>
```

Додаток Ж

Стилізація сторінки виконання лабораторної роботи

```
<style>
.page {
  padding: 10px 10px 10px 10px;
}
h5 {
  padding: 0;
  margin: 1em 0 0 0;
}
.settings input,
.settings select {
  width: 5em;
  padding: 0.2em;
}
.msg-true {
  background-color: aquamarine;
}
.msg-false {
  background-color: indianred;
}
.force {
  text-align: center;
  font-size: 3em;
}
.notes {
  text-align: center;
}
.md-input {
  padding: 0.2em;
  font-size: 1.4em;
}
.list-table {
  background-color: grey;
}
.list-table td {
  background-color: white;
  padding: 2px;
}
tr.isactive-true > td {
  background-color: #efbfbf;
}
</style>
```