МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра електроніки і комп'ютерної техніки

«До захисту допущено»

Завідувач кафедри ЕКТ

_____ Анатолій ОПАНАСЮК
(підпис) (Ім'я та ПРІЗВИЩЕ)

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «магістр»

зі спеціальності 171 «Електроніка»

освітньо-професійної програми «Електронні системи та компоненти»

на тему:

АВТОМАТИЗОВАНА ЕЛЕКТРОННА СИСТЕМА ЗБОРУ ДАНИХ ІЗ ЗАСТОСУВАННЯМ SMART ХАБУ

Здобувача групи ЕС.м – 21 Кривоноса Юрія Ігоровича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ _____
(підпис) (Ім'я та ПРІЗВИЩЕ)

Керівник, доцент, к.т.н., доцент Ольга БЕРЕЖНА _____
(підпис)

Консультант з техніко-економічної частини,
доцент, к.е.н., доцент Олександр МАЦЕНКО _____
(підпис)

Суми – 2023

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет електроніки та інформаційних технологій

Кафедра електроніки і комп'ютерної техніки

Напрям підготовки 171 Електроніка

Освітня програма Електронні системи та компоненти

ЗАТВЕРДЖУЮ

Зав. кафедрою Опанасюк А.С.

"____"_____ 2023 р.

З А В Д А Н Н Я

на кваліфікаційну роботу магістра

Кривоноса Юрія Ігоровича

1 Тема роботи: Автоматизована електронна система збору даних із застосуванням Smart хабу
затверджена наказом по університету "06" листопада 2023р. №1233-VI
2 Термін здачі студентом закінченої роботи: 14.12.2023
3 Вхідні дані до роботи: Smart хаб повинен забезпечити керування чотирма GSM модемами із USB портами та забезпечити CSD з'єднання із модемами на віддалених об'єктах.
4 Зміст розрахунково-пояснювальної записки (перелік питань, що належить розробити): 1. Огляд літератури та постановка задачі. 2. Науково-дослідна частина. 3. Розроблення електронної системи з використанням отриманих результатів дослідження. 4. Техніко-економічна частина.
5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): 1. Схема алгоритму. 2. Схема електрична структурна. 3. Схема електрична функціональна. 4. Схема електрична принципова.

6 Консультанти з кваліфікаційної роботи

Розділи	Консультанти	Завдання видав	Завдання прийняв
Техніко-економічна частина	Маценко О.М.		

7 Дата видачі завдання 06.11.2023

8 Керівник _____ Бережна О.В.

9 Завдання прийняв до виконання Кривоніс Ю.І.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів виконання дипломного проекту	Термін виконання	Примітки
1	Огляд літератури	10.11.23	
2	Науково-дослідна частина	24.11.23	
3	Розроблення алгоритму роботи системи	25.11.23	
4	Розроблення структурної схеми системи	26.11.23	
5	Розроблення функціональної схеми пристрою	01.12.23	
6	Розроблення принципової схеми пристрою	04.12.23	
7	Техніко-економічна частина	12.12.23	
8	Оформлення пояснювальної записки	13.12.23	
9	Розроблення та оформлення графічної частини	14.12.23	
10	Представлення роботи керівнику і отримання відгуку	15.12.23	
11	Представлення роботи кафедрі для отримання рецензії	19.12.23	

Керівник кваліфікаційної роботи: Бережна О.В.

Студент: Кривоніс Ю.І.

"____"_____ 2023 р.

РЕФЕРАТ

Пояснювальна записка містить: 91 аркуш, 42 рисунка, 13 таблиць, 16 джерел літератури.

Графічна частина роботи містить: схему алгоритму роботи пристрою, структурну, функціональну та принципову електричні схеми.

Пояснювальна записка містить сім розділів: огляд літератури і постановку завдання, науково-дослідну частину, розроблення структурної схеми системи та алгоритму її функціонування, розроблення функціональної схеми пристрою, розроблення принципової схем пристрою, розроблення програмного забезпечення, техніко-економічну частину.

Перший розділ містить загальну інформацію про автоматизовані системи та огляд дротових та бездротових технологій передачі даних, а також постановку завдання.

Другий розділ присвячений дослідженню характеристик алгоритмів стиснення даних і вибору алгоритму стиснення даних для застосування в автоматизованої системі збору даних.

У третьому розділі розроблено алгоритм функціонування та структурну схеми проектованої системи.

Четвертий розділ присвячений розробленню функціональної схеми пристрою.

В п’ятому розділі розроблення принципіальної схеми пристрою.

Шостий розділ присвячено розробленню програмного забезпечення пристрою.

У сьомому розділі наведено розрахунок собівартості та ціни Smart хабу.

ЗМІСТ

ВСТУП

У нашому сучасному світі, де обсяги інформації безперервно збільшуються, прогрес технологій вимагає наявності високоефективних та інноваційних систем для збору та аналізу цих даних. Однією з найбільш значущих галузей є розробка автоматизованих електронних систем, спрямованих на ефективний збір даних. У цьому контексті виникає вимога до нових рішень, які поєднують в собі принципи "розумних" технологій та ефективного збору інформації.

Шлях у сферу інноваційних технологій та автоматизації передбачає об'єднання передових підходів з ефективними засобами обробки даних. Використання Smart хабу в системі збору даних стає ключовим елементом, забезпечуючи не лише точність та надійність збору даних, а й розширюючи можливості оптимізації зберігання та передачі цієї інформації.

Об'єктом даного дослідження є процес інтеграції Smart хабу в автоматизовану систему та аналіз процесу стиснення зібраних даних. Мета полягає в вивченні технічних аспектів цього процесу та встановленні його впливу на оптимізацію зберігання та використання інформації.

Отримані результати цього дослідження мають велике значення для подальшого розвитку систем збору та обробки даних. Вони сприятимуть поліпшенню процесів зберігання великих обсягів інформації в умовах швидко змінюваних технологічних реалій сьогодення та визначать напрямок для майбутніх інновацій в цій області.

1 ОГЛЯД ЛІТЕРАТУРИ

Одне з основних завдань для будь-якого підприємства – це забезпечення ефективного використання енергії, що дозволить зберігати конкурентоспроможність у зростаючому середовищі вартості енергоресурсів. Для досягнення цієї мети важливо точно відстежувати споживання ресурсів. Для забезпечення можливості аналізу споживання електроенергії та його управління необхідно, щоб у кожного користувача був встановлений власний електролічильник. Звичайний процес зчитування лічильників, який виконується операторами, є неефективним через такі проблеми:

– потреба регулярно відвідувати всі об'єкти з лічильниками, що вимагає багато часу та коштів (заробітна плата та соціальні витрати оператора, транспортні витрати при великій відстані між об'єктами споживання енергії);

– ймовірність помилок на етапі, де бере участь людина. Таким чином, для якісного підвищення рівня контролю над споживанням та обліком енергоресурсів, щоб покращити ефективність енергоощадження, створюються спеціальні автоматизовані системи обліку енергоспоживання.

Головні завдання, які вирішує автоматизована система:

– забезпечення автоматизації отримання, обробки та аналізу поточних даних щодо споживання енергоресурсів;

– інтеграція даних про споживання з різних місць;

– підвищення ефективності обробки поточних даних;

– забезпечення резервування даних обліку енергоресурсів;

– формування звітів, що включають:

a) цінні профілі навантажень;

b) програми навантажень;

c) звіти за різними періодами (денні, тижневі, місячні, щорічні);

d) точність вимірювань;

e) платіжні дані.

В залежності від технологій та середовища передавання даних, спеціалізовані автоматизовані електронні системи можна умовно розділити на ті, що використовують дротові та бездротові засоби передачі інформації.

1.1 Огляд дротових технологій передачі даних

Важливо відзначити, що лічильник, як завершальний пристрій, має можливість не лише реєструвати виміри, а й передавати їх за запитом. Для цього він має в собі вузли або окремі додаткові пристрої, які забезпечують обмін інформацією через провідну або бездротову системи передачі даних. У процесі розвитку лічильників електроенергії від електромеханічних до електронних "розумних" пристроїв, вони отримують нові функції. Також зростає їх чутливість та надійність. Сучасні електронні лічильники можуть, наприклад, обліковувати енергію за різними тарифними зонами, що мотивує споживачів використовувати енергію під час "непікових" годин за економічними умовами. Деякі моделі лічильників дозволяють віддалено відключати споживачів від електроенергії за командою керуючого пристрою системи.

Історично автоматизоване зчитування показань з лічильників спочатку розпочалося з використання дротових систем. У цьому контексті використовувалися послідовні інтерфейси, такі як RS-232 та RS-485. Вибір технології передачі даних залежить від кількості приладів обліку на об’єкті та їхніх особливостей використання.

Послідовний інтерфейс RS-232 призначений для з'єднання пристроїв, що передають або приймають дані (наприклад, лічильники енергоресурсів), з кінцевим пристроєм, таким як персональний комп’ютер [1].

Зазвичай, пристрої обліку енергоресурсів оснащені цифровим послідовним інтерфейсом RS-232 у формі оптично-інфрачервоного інтерфейсу IrDA. Ця технологія дозволяє з'єднувати комп'ютер з лічильником для зчитування поточних або архівних значень, а також для налаштування параметрів.

Послідовний інтерфейс RS-485 є апаратним варіантом інтерфейсу RS-232. У відміну від останнього, для його роботи потрібна двохдротова лінія, до якої всі пристрої підключені паралельно. Максимальна довжина цієї лінії може сягати до 1000 метрів. Кількість підключених пристроїв, враховуючи узгоджувальні резистори, обмежується 32 пристроями в мережі [1].

Для підключення лічильників енергоресурсів до мережі RS-485 вони повинні бути обладнані відповідним інтерфейсом.

Переваги такої системи включають:

– стабільну роботу;

– безпечний обмін даними;

– високу швидкість передачі даних;
– високу стійкість до перешкод.

Основними недоліками цієї системи є:

– складність та тривалість прокладення дротових каналів;
– висока вартість;
– складність масштабування.

Незважаючи на це, через високу стійкість до перешкод та захищеність даних, дротові технології все ще застосовуються для створення систем дистанційного зчитування показань від приладів обліку.

1.2 Огляд бездротових технологій передачі даних

Серед багатьох можливих бездротових технологій, які використовуються для передачі даних від приладів обліку енергії, популярні наступні варіанти: GSM / GPRS, різні системи, створені виробниками лічильників і які працюють на неліцензованих частотах, Wireless M-Bus, LoRaWAN та NB-IoT.

Технологія CSD використовувалася довгий час для отримання інформації від вимірювальних приладів через аналогові модеми, що були підключені до телефонних ліній. Проте цей спосіб став незручним у багатьох системах через тривалість процесу — потрібно було чекати довгих інтервалів часу на встановлення зв'язку та опитування всіх пристроїв системи через послідовне опитування вузлів обліку [2].

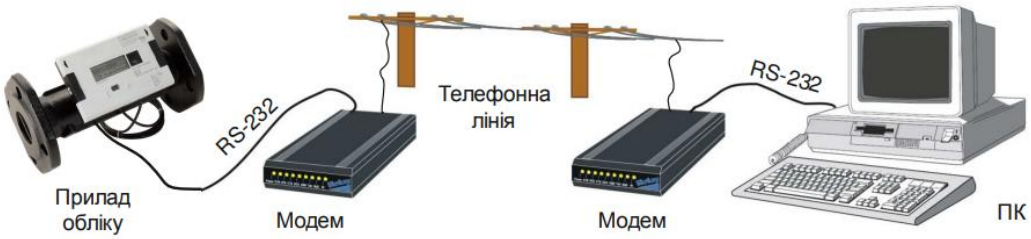


Рисунок 1.1 – Структура із аналоговим модемом

Зокрема, лише певні моделі теплолічильників та інших електронних вимірювальних приладів, які мали вбудоване чи зовнішнє джерело живлення, мали можливість прямого підключення до модему через інтерфейс RS-232. Проте у

більшості випадків підключення приладу обліку до модему було складним і вимагало використання додаткових інтерфейсних модулів.

Більш сучасна версія цього методу полягає у використанні GSM-модемів (технологія CSD) для передачі даних. Ця технологія передбачає створення голосового зв'язку між модемами через телефонний номер для передачі даних, коли один з модемів ініціює голосовий дзвінок іншому модему. Однак цей процес займає певний час і не дозволяє одночасно опитувати кілька приладів.

Рисунок 1.2 – Застосування GSM модему

Системи такого типу не стали популярними через кілька важливих недоліків:

- Великий час відклику та необхідність попереднього установлення зв'язку;
- Низька швидкість передачі даних (до 9,6 кбіт/с);
- Високі витрати на зв'язок, які враховують час спілкування, а не обсяг переданих даних.

GPRS (General Packet Radio Service) став більш поширеним завдяки використанню цього каналу зв'язку у GSM-модемах. Ця технологія передачі даних у формі пакетів у мережах GSM операторів дозволяє пристроям з'єднуватися через мережу Інтернет за протоколом TCP/IP [2].

Зазвичай така система включає в себе наступні компоненти: пристрій обліку з портом RS232 для виводу даних, GSM/GPRS-модем з активною SIM-картою, на якій є достатній баланс та активований режим передачі даних у формі пакетів.

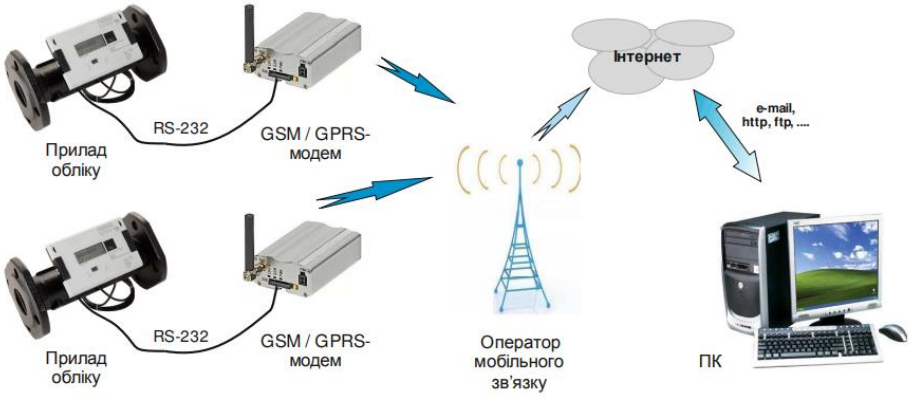


Рисунок 1.3 – Узагальнена структурна схема передачі даних за допомогою технології GPRS

Використання GPRS-модемів дозволило відмовитись від телефонних ліній та, у певних випадках, навіть від мережевого живлення. Однак вимоги до підключення приладів обліку і використовувані інтерфейси залишилися без змін.

При використанні цієї технології передача пакетів даних здійснюється за протоколом TCP у режимі "клієнт – сервер", оскільки для адресації використовується IP-адреса, а не номер телефону.

Модем автоматично узгоджується з розкладом для встановлення зв'язку, передачі даних та переходу в режим енергозбереження для збереження заряду батареї і ресурсів. GPRS-модеми можуть працювати в будь-якому місці з достатнім рівнем сигналу від базової станції мобільного оператора (що перевіряється за допомогою мобільного телефону) [2].

Основними особливостями технології GPRS є:

– Постійне з'єднання з приймальним обладнанням;
– Велика швидкість передачі даних (85,6 кбіт/с);
– Простота створення каналу зв'язку (легкість налаштування обладнання);
– Можливість одночасної передачі даних з усіх вузлів обліку системи.

Одним із важливих недоліків цієї передавальної технології є велике споживання енергії модемом. Найкращі моделі можуть працювати на одному заряді батареї до двох років, але це відбувається за умови рідкісної передачі даних (не більше одного разу на тиждень).

Системи, які виготовляються виробниками приладів обліку, використовують в більшості країн світу один або кілька неліцензійних частотних діапазонів. Це означає, що використання цих діапазонів не потребує отримання дозволів як для

виробників обладнання, так і для кінцевих користувачів. Наприклад, в країнах Європи, зокрема в Україні, доступні два таких діапазони: 433 МГц (з передавальною потужністю до 10 мВт) та 868 МГц (з передавальною потужністю до 25 мВт). Від початку 2000-х років і до сьогодення більшість виробників вимірювального обладнання активно створювали та впроваджували власні радіосистеми передачі показань, що операційно працюють в одному з вказаних неліцензійних частотних діапазонів. Оскільки не існувало загального стандарту для розробки таких систем, кожен виробник створював свою систему з власним протоколом обміну. Це призвело до появи широкого спектру несумісного обладнання, яке, хоча мало схожі функціональні можливості, не могло взаємодіяти разом у межах однієї системи [2].

Системи, що працюють у одному з двох режимів передачі, стали найпопулярнішими:

– Односторонній режим, де радіомодуль лічильника регулярно передає показання та іншу інформацію через фіксовані інтервали часу. Після передачі модуль переходить у сплячий режим для економії енергії батареї. Довгий термін служби батареї (понад 5 років) досягається завдяки великим інтервалам між передачами, що можуть тривати від кількох хвилин до десятків годин;

– Двосторонній режим, що працює за принципом "запит-відповідь". У цьому режимі радіомодуль лічильника переважно вимкнений, але його приймач постійно активний. Коли приймається запит від приймального пристрою, модуль активує передавач і надсилає поточні показання лічильника. Без запиту модуль не передає жодних даних у ефір.

Дальність передачі даних значно залежить від умов, в яких встановлений радіомодуль лічильника та від матеріалів та типу перешкод між ним і приймальним пристроєм. Якщо лічильники встановлені у підвалах житлових будинків, середня дальність передачі зазвичай складає 100–200 метрів.

Система також надає можливість зчитувати показання, проходячи повз місця, де встановлені прилади обліку, або заїжджаючи за маршрутом, де вони розташовані.

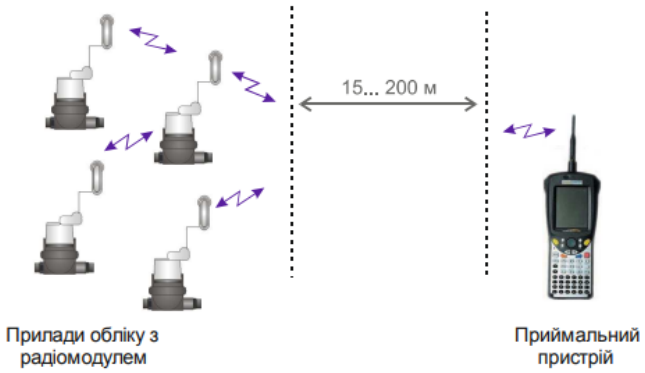


Рисунок 1.4 – Передача показань по радіоканалу

Крім того, це обладнання має можливість встановлення стаціонарної радіомережі, яка може бути розширена за допомогою додавання певної кількості ретрансляторів сигналу до радіомодулів. У цьому випадку збір показань відбувається з однієї центральної точки, до якої передаються дані від кожного лічильника — безпосередньо або через послідовний ряд ретрансляторів.

Рисунок 1.5 – Передача по радіоканалу із застосуванням ретрансляторів

Окрім тривалого терміну служби елементів живлення радіомодулів та можливості експлуатації обладнання без необхідності отримання дозволів на використання частотного ресурсу, важливими недоліками таких систем є:

– Несумісність обладнання від різних виробників, що ускладнює їх взаємодію.

– Обмежений радіус дії радіомодулів, особливо при встановленні у складних умовах, наприклад, у колодязях чи підвалах.

– Значні затримки у передачі даних через використання ретрансляторів.

Це призводить до великих затримок у відповіді від лічильника після запиту, особливо у випадку наявності багатьох ретрансляторів на шляху сигналу від лічильника до приймального пристрою.

Wireless M-Bus, що базується на M-Bus (Meter Bus), є стандартом комунікації у Європі для створення систем збору даних з комерційних вузлів обліку. Цей протокол використовує стандартну архітектуру "клієнт-сервер". Він є одним із найпоширеніших для передачі даних від різноманітних лічильників, таких яклічильники електроенергії, тепла, води, газу, а також деяких виконавчих пристроїв [2].

Інформація може передаватись безпосередньо на сервер або через концентратори шини M-Bus та підсилювачі-повторювачі сигналу. Основними особливостями цього протоколу є мінімальні технічні вимоги до обладнання та ліній зв’язку, а також простота та швидкість у впровадженні та монтажі.

Цей протокол має декілька фізичних рівнів: вита пара, оптична пара і радіоканал. Протокол Wireless M-Bus (WM-Bus) є варіантом M-Bus і використовує радіоканал у неліцензованих частотних діапазонах 169, 434 або 868 МГц. Обладнання, що працює за цим протоколом, розділяється на лічильники та приймальні пристрої (концентратори), які утворюють канал зв’язку типу "точка-точка". Лічильники передають інформацію про споживання, а концентратори збирають дані і можуть керувати лічильниками.

Середня дальність передачі показань від лічильника до приймача зазвичай складає 100–200 метрів для пристроїв, які використовують частотний діапазон 868 МГц з потужністю випромінювання 25 мВт. Ця відстань значно залежить від умов установки та типу перешкод (завад), які можуть бути між ними.

Протокол Wireless M-Bus визначає кілька режимів роботи, які включають:

- S (стаціонарний)
- T (часта передача)
- R2 (частий прийом)
- C (компактний)

У режимі T (часта передача), лічильник періодично відправляє дані. Цей режим має два підтипи: T1 і T2.

У режимі T1, лічильник надсилає дані незалежно від того, чи є концентратор. Після передачі даних пристрій виходить з режиму енергозбереження, передає дані і знову переходить у сплячий режим.

У режимі T2, лічильник передає дані і залишається активним протягом короткого інтервалу часу після передачі для очікування можливої відповіді. Якщо відповідь отримана, встановлюється двонаправлене з’єднання між лічильником і концентратором, в іншому випадку пристрій повертається до режиму енергозбереження.

Технологія, яка використовує протокол WMBus, подібна до раніше розглянутих систем. Її використання та можливості розгортання є схожими, але головна відмінність полягає в застосуванні стандартизованого протоколу обміну даними.

– Останнім часом прилади обліку, що вбудовані з радіомодулем і підтримують протокол WMBus, отримали значне розповсюдження в країнах Європи як серед виробників, так і серед користувачів. Це пояснюється кількома перевагами:

– Використання неліцензованого частотного діапазону 868 МГц дозволяє використовувати обладнання без необхідності отримання спеціальних дозволів на його експлуатацію.

– Енергоефективна технологія забезпечує довгий термін служби батареї в приладі обліку.

– Стандарт передбачає можливість шифрування передаваних даних, що забезпечує їх захист від несанкціонованого доступу.

– Ця технологія є придатною як для збору показань окремих лічильників через обхід або об’їзд, так і для створення постійних стаціонарних мереж за допомогою проміжних підсилювачів сигналу.

– Можливість використання у одній мережі обладнання різних виробників створює гнучкість та універсальність системи.

– Конкуренція серед виробників обладнання сприяла значному зниженню цін, особливо на приймальні пристрої. Це робить цю технологію ефективною для збору показань навіть з кількох приладів обліку.

Технологія Інтернету Речей (IoT) стає все більш поширеною - це мережа об'єктів, які унікально ідентифікуються і можуть спілкуватися між собою через IP-з'єднання без прямого втручання людини. У сфері передачі даних від приладів обліку енергоносіїв зараз активно використовуються два стандарти IoT: LoRaWAN і NB-IoT [2].

LoRaWAN (Long Range Wide Area Network) – це стандарт, який надає можливість передавати дані на великі відстані і призначений для обслуговування великої кількості малопотужних пристроїв, таких як прилади обліку.

Зазвичай мережа LoRaWAN складається з кількох елементів, які працюють у форматі "точка-точка": абонентські пристрої (сенсори, датчики), які часто перебувають у режимі енергозбереження при автономному живленні; базові станції або шлюзи, мережевий сервер та сервери застосунків [2].

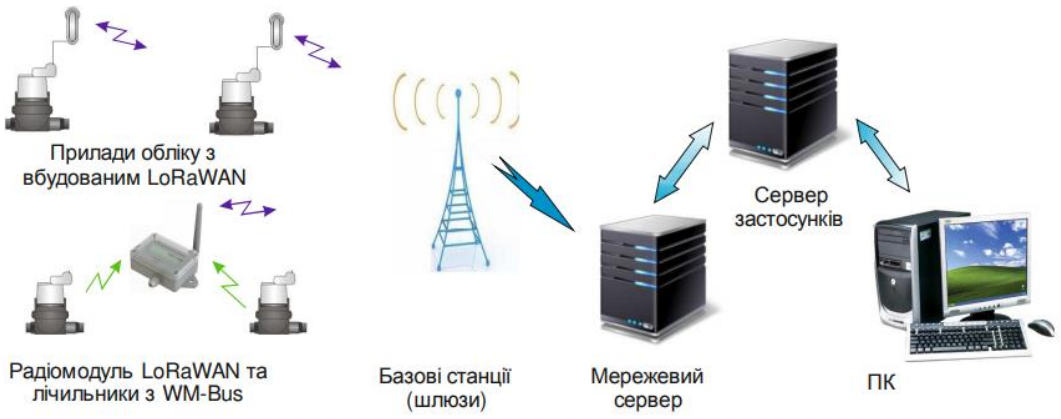


Рисунок 1.6 – Структура мережі із застосуванням технології LoRaWAN

Важливим аспектом мережі є можливість використання обладнання різних виробників. Всі компоненти цієї системи діють у неліцензованому частотному діапазоні 868 МГц.

Підключення лічильників до LoRaWAN виконується за допомогою спеціальних радіомодулів з автономним живленням. Ці модулі отримують дані від приладів обліку бездротовим шляхом за протоколом WM-Bus або безпосередньо підключаються до лічильника, що має імпульсний вихід.

Багато виробників вимірювального обладнання випускають лічильники з вбудованим радіомодулем, який дозволяє безпосереднє підключення приладу до мережі LoRaWAN.

Обладнання LoRaWAN відрізняється від WM-Bus у способі передачі даних: використовується вузька смуга частот та низька швидкість передачі, що дозволяє передавати дані на великі відстані (до 10 км на відкритій місцевості та 1-2 км у міських умовах). Якість передачі даних значно залежить від прямого огляду між абонентом та базовою станцією та відстані між ними.

Кожна базова станція має постійне з’єднання з мережею Інтернет, тому постійно отримує дані від радіомодулів лічильників та передає їх на мережевий сервер без перебоїв. Програмне забезпечення верхнього рівня через сервер застосунків отримує потрібну інформацію по обраним лічильникам, яка зберігається на мережевому сервері.

NB-IoT (Narrow Band Internet of Things) – це стандарт мобільного зв'язку, спеціально розроблений для телеметричних пристроїв, які передають невеликі обсяги даних. Цей стандарт був створений консорціумом 3GPP в рамках розробки стандартів нового покоління мобільних мереж. Перша версія специфікації була вперше представлена у середині 2016 року. NB-IoT є одним із трьох стандартів IoT, розроблених 3GPP для використання в мобільних мережах зв'язку [2].

Мережа NB-IoT може бути впроваджена як частина мобільних мереж LTE або окремо, навіть на базі GSM. Оператор мобільного зв'язку, який використовує технологію NB-IoT, забезпечує покриття радіосигналом по всій території країни, використовуючи існуючі базові станції 4G. Не потрібно встановлювати додаткове обладнання, оскільки базова станція оператора мобільного зв'язку автоматично ідентифікує пристрій та починає з ним взаємодіяти.

Підключення приладів обліку до мережі NB-IoT потребує спеціальних модемів з SIM-карткою відповідного оператора мобільного зв’язку. Це з'єднання може відбуватися через радіоканал (протокол WM-Bus) або через імпульсний вихід лічильника, іншого обладнання не потрібно. Оператор мобільного зв’язку виступає як транспортна інфраструктура, що забезпечує передачу показань від приладу обліку на сервер постачальника енергоресурсів.

Рисунок 1.7 – Структура мережі із застосуванням технології NB-IoT

Переваги NB-IoT включають:

– Низьке енергоспоживання: Довготривале використання батареї (5-10 років), проте час роботи пристрою може змінюватися в залежності від обсягу передачі даних та кількості зв'язкових сеансів.

– Велика ємність мережі: Можливість підключення десятків і навіть сотень тисяч пристроїв до однієї базової станції.

– Низька вартість обладнання: Економічні модеми дозволяють знизити витрати на кінцеве обладнання.

Так, використання технологій IoT у цій галузі дійсно знаходиться на ранньому етапі впровадження. Для успішного розгортання таких мереж потрібна глибока співпраця між виробниками обладнання, зокрема приладів обліку, та операторами мобільного зв'язку. Це дозволить створити ефективну інфраструктуру, що відповідає потребам галузі і забезпечить надійний обмін даними у системі IoT.

1.3 Постановка задачі

Метою роботи є розробка автоматизованої електронної системи збору даних із застосуванням Smart хабу, який повинен забезпечити підвищення стабільності роботи модемного пулу шляхом автоматичного контролю працездатності модемів та відновлення їх роботи після збоїв та відмов шляхом перезавантаження модемів програмно та по живленню.

Для розроблюваної системи передачі даних необхідно виконати наступне:

1. Визначити завдання та основні функції, які буде виконувати автоматизована електронна система збору даних із застосуванням Smart хабу.
2. Виконати дослідження характеристик алгоритмів стиснення даних без втрат, здійснити порівняльний аналіз характеристик алгоритмів стиснення без втрат. Спираючись на результати досліджень обрати алгоритм стиснення даних, який доцільно застосовувати в системі збору даних.
3. Розробити схему електричну структурну та схему алгоритму роботи системи збору даних, Smart хабу і пристрою стиснення даних.
4. Розробити схему електричну функціональну та схему електричну принципову Smart хабу.
5. Розрахувати собівартість та ціну Smart хабу.

2 НАУКОВО-ДОСЛІДНА ЧАСТИНА

2.1 Поняття про стиснення даних

Більшість видів даних відзначається надмірністю, яка становить одну з їхніх основних характеристик. Звісно, ступінь цієї надмірності може варіюватися залежно від типу даних. Наприклад, в порівнянні з графічними даними, відеодані зазвичай виявляють велику надмірність, тобто займають більше місця, та знаходяться на кілька порядків вище за надмірність графічних даних. З свого боку, надмірність графічних даних може бути значно вищою, ніж надмірність текстових даних. Важливим фактором, який впливає на ступінь надмірності, є система кодування, що застосовується для подання даних. Зокрема, системи кодування, такі як звичайні мови спілкування, фактично слугують системами кодування понять та ідей для передачі думок.

У конкретному випадку, кодування текстових даних з використанням засобів української мови призводить до збільшення надмірності на приблизно 20-25% у порівнянні з кодуванням схожих даних англійською мовою [3].

Для людини, надмірність даних зазвичай визначає співвідношення між кількістю інформації та її якістю. В багатьох випадках надмірність сприяє кращому розумінню та сприйняттю інформації. Проте, коли мова йде про зберігання та передачу даних за допомогою комп'ютерної техніки, надмірність може мати негативний вплив, так як вона призводить до збільшення витрат на зберігання та передачу інформації. Ця проблема особливо актуальна, коли необхідно обробляти величезні обсяги даних при обмежених ресурсах для зберігання і передачі.

Тим самим, виникає необхідність у вирішенні проблеми надмірності даних або їх стисненні.

Стиснення даних – це процес зменшення обсягу інформації шляхом зміни, кодування чи перетворення даних. Суть цього процесу полягає в перекодуванні інформації з використанням меншої кількості бітів, ніж у початковому представленні. Щоб це досягти, використовуються різноманітні алгоритми та методи, які ефективно визначають, як зменшити обсяг даних. Наприклад, алгоритм може замінити послідовність бітів меншою послідовністю, використовуючи довідковий словник для перетворення між ними [3].

Стиснення тексту часто досягається за допомогою видалення зайвих символів і заміни їх одним символом, який посилається на повторення

послідовностей символів, та заміною менших рядків бітів на більші. За допомогою правильних методів стискування даних можна зменшити розмір текстового файлу на 50% та більше, суттєво скоротивши його загальний обсяг.

При передачі даних може бути використане стиснення для оптимізації як обсягу передачі, так і змісту. Це може стосуватися як окремих файлів, так і всієї передачі. При використанні Інтернету файли великого обсягу, які передаються як окремі об'єкти або як частина архівних файлів, можуть бути переведені у компактні формати стиснення, такі як ZIP, RAR, 7z або MP3, для ефективнішої передачі через мережу та зменшення обсягу займаного місця на пристрої отримувача.

Стиснення даних використовує різні підходи та методи в залежності від призначення та характеристик даних. Наприклад, алгоритми стиснення тексту можуть виявити і вилучити надлишкові прогалини, символи та інші зайві елементи тексту, щоб зменшити його розмір. Стиснення зображень може використовувати методи компресії, які редукують кількість інформації, представленої в зображенні, при цьому намагаючись зберегти його якість. Для звукових файлів, стиснення може включати в себе видалення зайвої аудіо інформації або використання аудіо кодеків, які зменшують бітовий обсяг аудіо сигналу.

Стиснення даних важливе в різних сферах, включаючи зберігання і передачу інформації, оптимізацію роботи програм та зменшення витрат ресурсів. Також воно грає важливу роль в забезпеченні якості обслуговування користувачів, зменшуючи час передачі та завантаження мережі. Розуміння процесу стиснення даних та вибір відповідних методів інструментів є ключовими аспектами у сучасному інформаційному суспільстві.

2.2 Огляд методів стиснення даних

В сфері стиснення даних існує багато практичних методів, і всі вони опираються на два основних способи зниження надлишковості. Перший метод включає в себе зміну вмісту даних, тоді як другий змінює структуру даних [3].

Якщо при стисненні даних відбувається модифікація їхнього вмісту, то метод стиснення є незворотнім, іншими словами, при відновленні даних не вдається повністю відновити початкову інформацію. Ці методи часто називаються методами стиснення з втратами інформації. Зрозуміло, що їх можна використовувати лише для типів даних, де втрата частини вмісту не призводить до значущого спотворення інформації. До таких типів даних відносяться відео-, аудіо- та графічні дані.

Методи стиснення з втратами інформації забезпечують значно вищий ступінь стиснення, але не підходять для текстових даних. Прикладами таких форматів стиснення є JPEG (Joint Photographic Experts Group) для графічних даних, MPG для відеоданих та MP3 для аудіоданих.

У випадку, коли при стисненні даних змінюється лише їхня структура, метод стиснення є зворотнім. У цьому випадку з архіву можна повністю відновити інформацію. Зворотні методи стиснення можна застосовувати до будь-яких типів даних, але вони надають менший ступінь стиснення порівняно з незворотними методами стиснення. Прикладами форматів стиснення без втрати інформації є GIF (Graphics Interchange Format) та TIFF (Tagged Image File Format) для графічних даних, AVI для відеоданих і різні формати стиснення файлів, такі як ZIP, ARJ, RAR, CAB, LH, для різних типів даних.

Обрання методу стиснення даних без втрат може бути обґрунтованим різними причинами, особливо коли мова йде про стиснення текстових даних з лічильників. Основна перевага цього підходу полягає в тому, що він дозволяє зберегти всю початкову інформацію без будь-яких втрат чи спотворень. Для текстових даних, особливо якщо це статистичні або числові дані з лічильників, точність та цілісність інформації є критичними факторами. Стиснення без втрат забезпечує збереження цієї точності і надає можливість подальшого аналізу та використання даних без втрати якості.

Для стиску без втрат доведено такі теореми.

– Для будь-якої послідовності даних існує теоретична межа стиснення, яка не може бути перевищена без втрати частини інформації.

– Для будь-якого алгоритму стиснення можна знайти таку послідовність даних, для якої він забезпечить кращий ступінь стиснення, ніж інші алгоритми.

– Для будь-якого алгоритму стиснення можна знайти таку послідовність даних, для якої даний алгоритм взагалі не дозволить отримати стиснення.

Зі сформульованих теорем випливає, що найвищу ефективність алгоритми стиснення демонструють для різних типів даних та різних обсягів даних. Тому розроблено нині досить багато алгоритмів стиснення без втрат [3].

Найбільш поширені алгоритми стиснення без втрат наведено на рисунку 2.1.

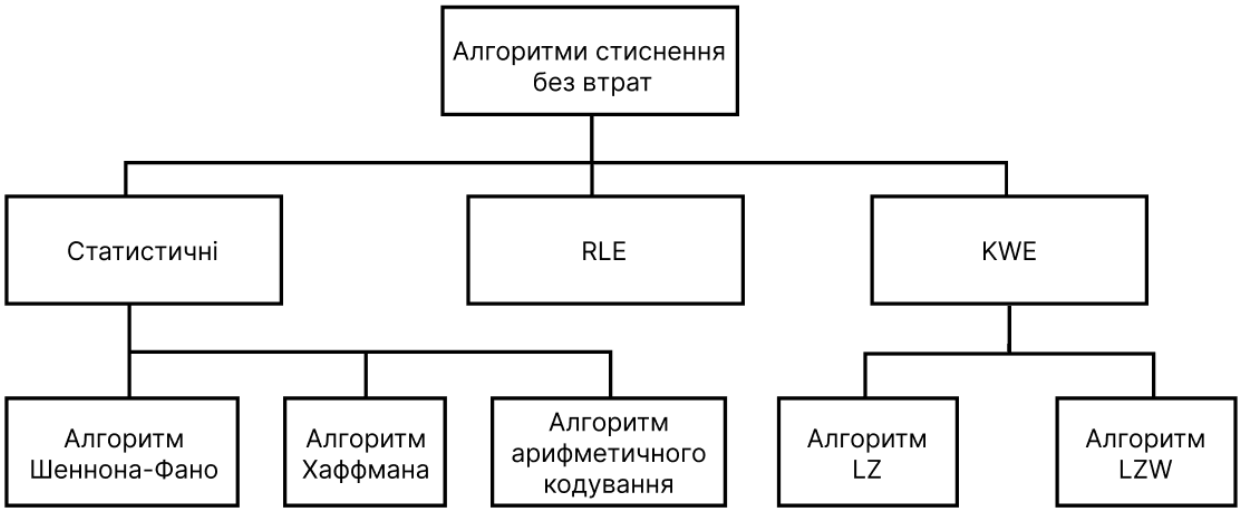


Рисунок 2.1 – Алгоритми стиснення без втрат

2.3 Характеристики алгоритмів стиснення даних

У процесі порівняння різних методів стиснення даних виникає необхідність оцінки їхньої ефективності за конкретними критеріями. Для цього використовуються чисельні показники, як критерії оцінки програмних та апаратних реалізацій алгоритмів стиснення.

Коефіцієнт стиснення R обчислюється як співвідношення об'єму вихідних нестислих даних до об'єму стислих даних за формулою:

$$R = \frac{S_{\text{вих}}}{S_{\text{сж}}}$$

де R – коефіцієнт стиснення;

$S_{\text{вих}}$ – обсяг вихідних даних;

$S_{\text{сж}}$ – обсяг стислих даних.

Ступінь стиснення r характеризує відносне зменшення обсягу даних:

$$r = \frac{S_{\text{вих}} - S_{\text{сж}}}{S_{\text{вих}}} \cdot 100\%$$

Коефіцієнт стиснення R та ступінь стиснення r характеризують один і той самий критерій ефективності, але дають різний порядок цифр. Наприклад, обсяг

вихідних даних дорівнює 100 Кбайт, обсяг стиснутих архіватором цих даних дорівнює 10 Кбайт. В цьому випадку:

$$R = \frac{100\ \text{Кбайт}}{10\ \text{Кбайт}} = 10$$

$$r = \frac{(100 - 10)\text{Кбайт}}{100\ \text{Кбайт}} \cdot 100\% = 90\%$$

Швидкість стиснення $V_{\text{ст}}$ – це параметр, який обчислюється, як співвідношення об'єму вихідних нестислих даних до часу стиснення вихідних даних за формулою:

$$V_{\text{ст}} = \frac{S_{\text{вих}}}{t_{\text{ст}}}$$

де $t_{\text{ст}}$ – час стиснення вихідних нестислих даних.

Швидкість розпакування $V_{\text{р}}$ – це параметр, який визначає з якою швидкість відбувається розпакування даних. Визначається за формулою:

$$V_{\text{р}} = \frac{S_{\text{сж}}}{t_{\text{р}}}$$

де $t_{\text{р}}$ – час розпакування стислих даних.

Очевидно, що час стиснення та розпакування істотно залежить від продуктивності апаратних засобів, що використовуються. Тому при порівнянні швидкості роботи реалізацій алгоритмів має вказуватись обладнання, використане для тестування.

Важливою характеристикою алгоритму стиснення є симетричність у часі C – відношення часу стиснення вихідних даних до часу розпакування:

$$C = \frac{t_{\text{ст}}}{t_{\text{р}}}$$

Деякі алгоритми (наприклад, фрактальний алгоритм стиснення зображень) забезпечують дуже великий коефіцієнт стиснення, але витрачають багато часу на стиснення зображення. Однак розпакування стислих зображень здійснюється дуже швидко, тобто алгоритм має високу несиметричність. Такий алгоритм застосовувати доцільно для архівації зображень, тому що в цьому випадку стиск здійснюється один раз, а при багаторазовому використанні буде проводитися досить швидко. Таким чином, симетричність показує сферу застосування алгоритму.

2.4 Алгоритми стиснення даних без втрат

Алгоритм Шеннона-Фано

Першим статистичним алгоритмом стиснення був алгоритм Шеннона-Фано. Коди Шеннона-Фано префіксні, тобто жодне кодове слово не є префіксом будь-якого іншого. Ця властивість дозволяє однозначно декодувати будь-яку послідовність кодових слів.

Алгоритм Шеннона-Фано використовує надмірність повідомлення, укладену в неоднорідному розподілі частот символів його (первинного) алфавіту, тобто замінює коди частіших символів короткими двійковими послідовностями, а коди більш рідкісних символів — довшими двійковими послідовностями.

Техніка отримання кодів така: Символи первинного алфавіту виписують у порядку зменшення ймовірностей. Символи отриманого алфавіту ділять на дві частини, сумарні ймовірності символів яких максимально близькі один до одного.

У префіксному коді першій частині алфавіту присвоюється двійкова цифра «0», другої частини — «1». Отримані частини рекурсивно діляться та його частинам призначаються відповідні двійкові цифри у префіксному коді. Коли розмір подалфавіту дорівнює нулю або одиниці, то подальшого подовження префіксного коду для відповідних йому символів первинного алфавіту не відбувається, таким чином, алгоритм присвоює різним символам префіксні коди різної довжини. На кроці поділу алфавіту існує неоднозначність, так як різниця сумарних ймовірностей може бути однаковою для двох варіантів поділу (враховуючи, що всі символи первинного алфавіту мають ймовірність більше за нуль).

Програмна реалізація алгоритму Шеннона-Фано за часом для 11 символів (цифри та перенос рядка) алфавіту ASCII:

Таблиця 2.1 – Процес кодування алгоритмом Шеннона-Фано з розміром алфавітом 11 символів

Кількість символів в тесті	Кодування Шеннона-Фано, мс
500	1
1000	2
5000	5
10000	8

Таблиця 2.2 – Процес декодування алгоритмом Шеннона-Фано з розміром алфавітом 11 символів

Кількість символів в тесті	Кодування Шеннона-Фано, мс
500	1
1000	2
5000	6
10000	11

Графік залежності часу роботи на кодування та декодування алгоритмом Шеннона-Фано від розміру файлу у символах:

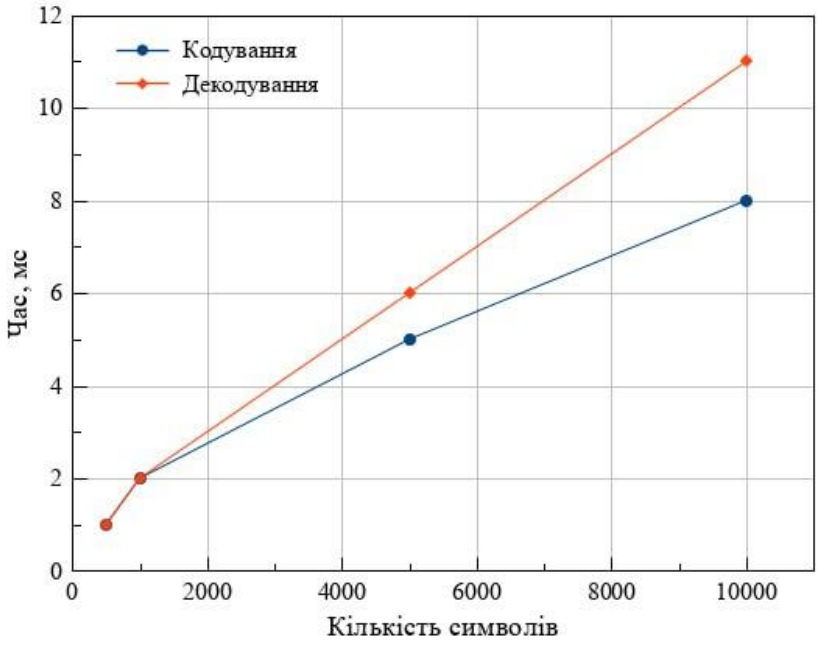


Рисунок 2.2 – Графік залежності часу на кодування та декодування від кількості символів з розміром алфавітом 11 символів

Алгоритм Хаффмана

Алгоритм Хаффмана заснований на принципі присвоєння символам унікальних кодів змінної довжини, що складаються з цілих кількостей бітів. Довжина коду визначається частотою зустрічі символу у вихідних даних: рідше зустрічається символ - довший його код. Крім того, процес побудови кодів забезпечує унікальність та однозначність кодів завдяки властивості префіксності [4].

Для створення коду символу потрібно мати таблицю частоти появи символів у вхідних даних або їхні імовірності. На основі цієї таблиці створюється дерево Хаффмана за наступними кроками:

– Створення списку вільних вузлів із символів, які зустрічаються у вхідних даних. Кожен вузол має вагу – імовірність появи символу у даних.

– Вибір двох вільних вузлів з мінімальною вагою і створення для них батьківського вузла з вагою, що дорівнює сумі ваг його дітей.

– Додавання батьківського вузла до списку вільних вузлів і видалення його дітей зі списку. Додатково встановлюються 0 та 1 для дуг, які з'єднують батьківський вузол із його дітьми.

– Повторення попередніх кроків до того моменту, поки залишиться лише один вузол у списку вільних вузлів – корінь дерева.

З використанням дерева Хаффмана можна легко отримати код будь-якого символу. Для цього необхідно пройти шлях від кореня дерева до вузла, що відповідає цьому символу. Отримана послідовність нулів та одиниць і буде кодом Хаффмана.

Однак оригінальний алгоритм має дві важливі недоліки. По-перше, для розпакування даних розпаковувачу потрібно мати таблицю кодів, створену пакувальником, що може збільшити обсяг переданої інформації. По-друге, необхідно двічі прочитати дані – один раз для збору статистики символів, а потім для стиснення [4].

Програмна реалізація алгоритму Хаффмана за часом для 11 символів (цифри та перенос рядка) алфавіту ASCII:

Таблиця 2.3 – Процес кодування алгоритмом Хаффмана з розміром алфавітом 11 символів

Кількість символів в тесті	Кодування Шеннона-Фано, мс
500	1
1000	2
5000	4
10000	8

Таблиця 2.4 – Процес декодування алгоритмом Хаффмана з розміром алфавітом 11 символів

Кількість символів в тесті	Кодування Шеннона-Фано, мс
500	1
1000	2
5000	5
10000	8

Графік залежності часу роботи на кодування та декодування алгоритмом Хаффмана від розміру файлу у символах:

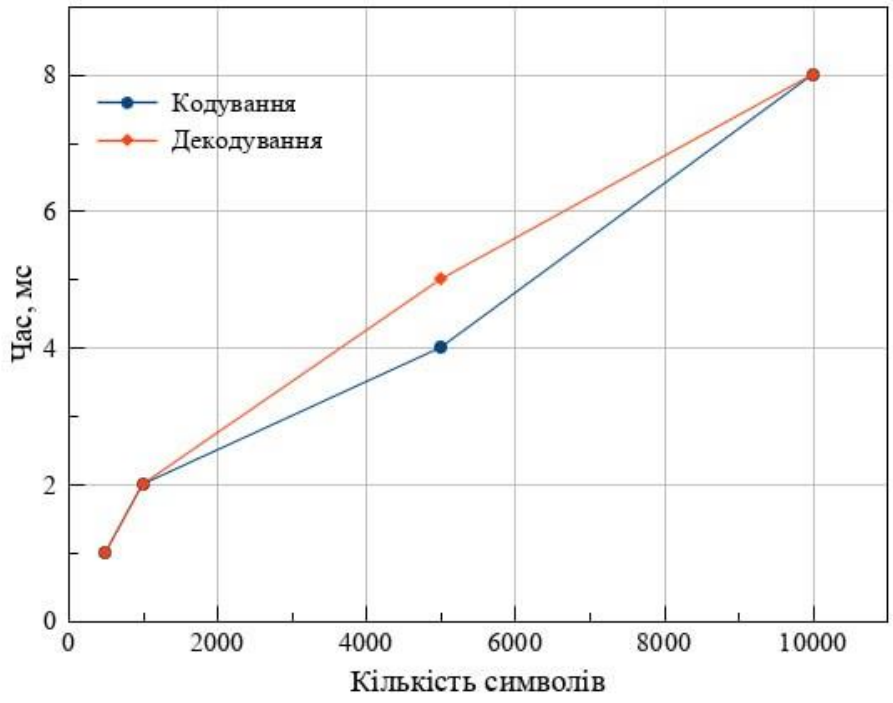


Рисунок 2.3 – Графік залежності часу на кодування та декодування від кількості символів з розміром алфавітом 11 символів

Адаптивний алгоритм Хаффмана

Основна ідея адаптивного алгоритму Хаффмана полягає в тому, що як кодер, так і декодер починають з порожнього дерева Хаффмана, а потім змінюють його під час читання та обробки сигналів. Кодер і декодер повинні змінювати дерево однаковим чином, щоб постійно використовувати один і той же код, який може змінюватися під час процесу. Якщо стискувач і розпаковувач синхронізовані, це означає, що їхні роботи виконуються згідно з жорстко встановленим порядком (хоча не обов'язково в один і той же час). Спочатку кодер створює порожнє дерево Хаффмана, куди ще не призначено жодного символу. Перший символ просто записується у вихідний файл у непакованому вигляді. Потім цей символ додається до дерева, і йому присвоюється код. Якщо він зустрінеться знову, його поточний код буде записаний у файл, а його частота збільшиться на одиницю. Оскільки ця операція змінює дерево, його потрібно перевірити на те, чи є воно деревом Хаффмана (чи дає найкращі коди). Якщо ні, це вимагає перебудови дерева та зміни кодів.

Декомпресор діє зворотним чином. Коли він читає непакований символ, він додає його до дерева і надає йому код. Якщо він читає упакований код (змінної довжини), він використовує поточне дерево для визначення, якому символу відповідає цей код, після чого змінює дерево таким же чином, як і кодер.

Перевірка дії проводиться при кожному входженні нового вхідного символу. Якщо дерево не є деревом Хаффмана, його слід виправити. Процес змін, що показаний на рисунку 2.4, описує модифікацію дерева. Дерево на рисунку 2.4а складається з п'яти символів - A, B, C, D, E. У круглих дужках для кожного символу вказані його частоти появи. Властивість Хаффмана передбачає, що частоти, при проходженні по дереву зліва направо і знизу вгору (від листя до кореня), будуть впорядковані за зростанням або не спаданням. Отже, лівий нижній вузол (A) має найменшу частоту, а верхній правий (корінь) – найбільшу. Це властивість конкуренції, оскільки символи з високими частотами мають коротші коди і розташовані на вищих рівнях у дереві. Впорядкованість частот на одному рівні - це вимога для визначеності, що спрощує процес побудови дерева.

Цикл модифікації дерева починається з вузла, що відповідає новому вхідному символу. Позначимо цей вузол X, а його частоту - F. На кожній ітерації циклу потрібно виконати три дії:

1. Порівняти X з його найближчими сусідами в дереві (праворуч і зверху). Якщо сусід має частоту F+1 або більше, вузли залишаються впорядкованими, і

нічого не змінюється. Якщо сусід має частоту F або менше, X і останній вузол в цій групі міняються місцями (крім випадку, коли X - батьківський вузол).

2. Збільшити частоту X з F до F+1. Збільшити частоту всіх його батьків на одиницю.

3. Якщо X - корінь, то цикл завершується. Інакше він повторюється до вузла, який є батьківським для X.

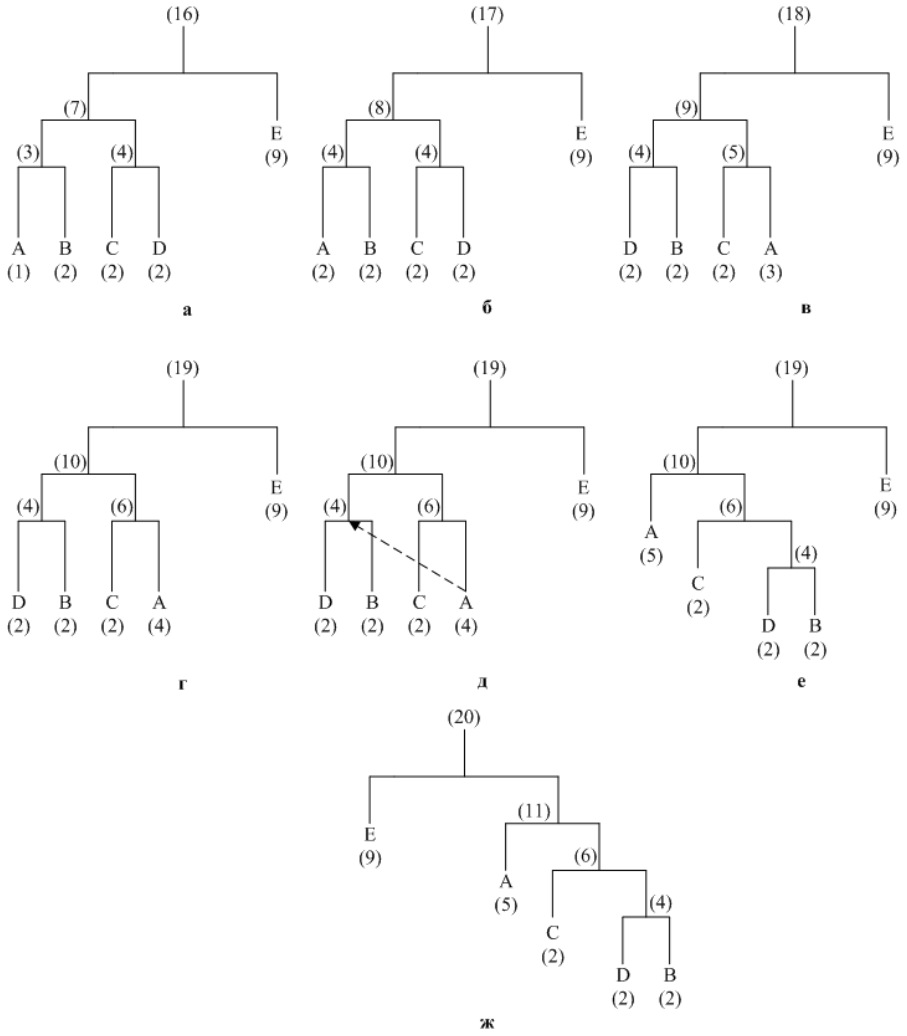


Рисунок 2.4 – Модифікація дерева Хаффмана

Алгоритм арифметичного кодування

Алгоритм арифметичного кодування, так само як і метод Хаффмана для стискання, використовує різницю у частоті появи символів, щоб зменшити обсяг початкових даних. Під час стискання символи, які зустрічаються рідко, кодуються більш довгими кодами порівняно з символами, які зустрічаються частіше. Але відмінністю від алгоритму Хаффмана, символи можуть бути закодовані не

обов'язково цілим числом бітів, тобто один стиснутий біт даних може відноситися до декількох символів вихідних даних, що стискаються [5].

Для даних, де частоти зустрічі символів майже рівні, арифметичне кодування демонструє результати, схожі на ті, що отримуються від алгоритму Хаффмана. Проте, якщо частоти символів різко відрізняються при їх невеликій кількості, арифметичне кодування видає кращі результати порівняно з алгоритмом Хаффмана. У більшості реальних сценаріїв арифметичне кодування забезпечує більший коефіцієнт стиснення.

Під час арифметичного стиснення повідомлень алфавіту, числова вісь [0,1) використовується для відображення діапазонів, де кожен символ алфавіту займає свій власний відрізок. Розмір кожного відрізку на цій вісі залежить від ймовірності (частоти) зустрічі символу у повідомленні.

Розглядаючи, як приклад, алфавіт з шести символів A, B, C, D, E, F, де ймовірності зустрічі відповідно дорівнюють 0.1, 0.1, 0.3, 0.2, 0.2 і 0.1. Тоді відрізок [0,1) можна поділити на такі частини:

A: [0, 0,1); B: [0,1, 0,2);
C: [0,2, 0,5); D: [0,5, 0,7);
E: [0,7, 0,9); F: [0,9, 1,0).

Розподіл числової осі ілюструється на рисунку 5. Як видно з прикладу, поділ одиничного інтервалу зручно здійснюється підсумовуванням імовірності кожного з границею попереднього [5].

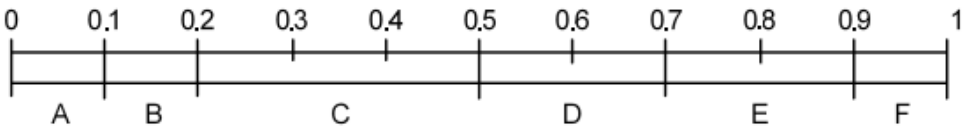


Рисунок 2.5 – Розподіл відрізків числової осі між символами при арифметичному кодуванні

Під час стиснення текстового повідомлення кожен символ відображається на числовому інтервалі в межах [0, 1]. Починаючи з інтервалу [0, 1], перший символ виділяє свій власний діапазон, пропорційний його ймовірності в повідомленні. При надходженні наступних символів цей діапазон звужується, а нові межі визначаються ймовірністю кожного наступного символу. Цей процес триває з кожним новим символом, стискаючи інтервал. Теоретично його можна звужувати

нескінченно, але на практиці обмежують технічні обмеження кодувальних пристроїв, такі як розмір слова.

Кількість бітів, потрібних для представлення ширини інтервалу s, обчислюється за допомогою $-\log(s)$ з основою логарифму 2. Ширина s останнього інтервалу у повідомленні з N_C символами обчислюється як добуток ймовірностей символів P_{jC} повідомлення.

$$S = \prod_{i=1}^{N_C} P_{jC}$$

Враховуючи це, можна сформулювати, що:

$$-\log_2(s) = -\sum \log P_i = -\sum_{i=1}^{m} P(a_i) \log P(a_i)$$

де m – кількість різних символів повідомлення a_i, (i = 1,2…m)

Отже, кількість бітів, використаних для кодування повідомлення арифметичним методом, точно відповідає ентропії джерела [5].

Алгоритм RLE

Алгоритм RLE (Run-Length Encoding) – це простий метод стиснення, спрямований на швидке скорочення об'єму даних, коли у послідовності зустрічаються багато однакових символів, що йдуть один за одним.

Алгоритм кодування довжин серії ґрунтується на ідентифікації послідовностей однакових символів, які йдуть один за одним, і їх заміні на структуру, що включає код символу та кількість повторень у цій послідовності. Іншими словами, група однакових символів, які зустрічаються поряд, замінюється на пару кодів у формі <кількість повторень; код символу>. Максимальна кількість однакових символів, яку можна закодувати у такій парі, визначається довжиною коду кількості повторень [3].

Проблему при застосуванні алгоритму RLE складають дані, в яких зустрічається обмежена кількість однакових символів підряд. Під час стискання, навіть для одиночних символів, доводиться додавати лічильник повторень, що призводить до збільшення розміру даних, а не їх стискання. У практичних

реалізаціях алгоритм RLE частково модифікується для уникнення збільшення обсягу стиснутих даних у випадку не найкращих для нього даних.

Якщо алфавіт складається лише з двох символів (наприклад, для чорно-білих зображень), тоді для кодування достатньо створювати послідовності змінних довжин, що чергуються одна за одною.

Наприклад, для закодування рисунка 6 потрібно 64 біти, якщо не використовувати жодні методи стиснення.

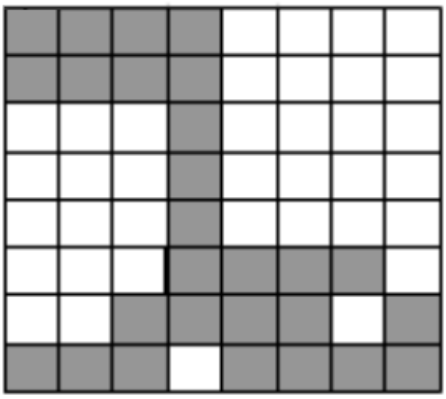


Рисунок 2.6 – Приклад чорно-білого зображення для кодування

Необхідні 64 біти:

1111000111100000001000000010000000111100011110111101111

При кодуванні довжин послідовностей нулів та одиниць отримуємо наступне:

4, 4, 4, 7, 1, 7, 1, 7, 1, 7, 4, 3, 4, 1, 4, 1, 4

Якщо застосовується рівномірне кодування (три біти на символ), довжина стиснутого повідомлення буде 51 біт, що менше за довжину вихідного повідомлення приблизно на 20% [3].

Якщо маємо алфавіт із більш ніж двох символів, розглядаємо кілька варіантів відповідно до того, які символи зустрічаються та як часто.

1. Якщо окремі символи та послідовності різних символів зустрічаються рідко, тоді кожна послідовність замінюється на два параметри: кількість символів у послідовності та сам символ, який повторюється (кожен окремий символ також кодується як послідовність довжиною 1).

2. Якщо послідовності різних символів зустрічаються достатньо часто, тоді кожна з таких послідовностей замінюється трьома складовими:

– префікс, який вказує на необхідність розкодування наступних двох символів як частини послідовності;

– довжина самої послідовності;

– символ, який повторюється у цій послідовності.

Решта символів передаються без будь-якого додаткового кодування. Якщо довжина послідовності менша за 4, кодувати її у префіксному вигляді неефективно, оскільки це не призводить до зменшення обсягу даних.

Як префікс можна використовувати символ, який ніколи не з'являється в тексті. Якщо такий символ відсутній, то префіксом може стати символ, що має найменшу ймовірність. Якщо цей символ з'явиться у тексті, він кодується за префіксною системою.

3. Якщо окремі символи зустрічаються рідко, а послідовності різних символів зустрічаються ще рідше, але можуть мати велику довжину, у цьому випадку можна закодувати саме ці послідовності у префіксному вигляді. Після префіксу необхідно вказати довжину таких послідовностей різних символів. Для префіксу варто використовувати 0, оскільки для послідовностей довжина 0 неможлива.

4. Якщо виникають як довгі послідовності однакових символів, так і довгі ряди різних символів, і довжина ряду досягає певної межі, всі наступні символи ряду замінюються кодом їх кількості (а якщо довжина рівна межі, обов'язково передається код 0 для правильного розкодування).

Найкращими об'єктами даного алгоритму є графічні файли, у яких великі одноколірні ділянки зображення кодуються послідовністю однакових байтів. Цей алгоритм може давати помітне стиснення деяких типах файлів баз даних, що мають таблиці з фіксованою довжиною полів. RLE використовується як етап стиснення алгоритмів стиснення зображень. Слід зазначити, що даний алгоритм стиснення є неефективними для стиснення текстових даних.

Алгоритм групи KWE

Алгоритм кодування за ключовими словами (KWE = Keyword Encoding) базується на тому, щоб перетворювати частини тексту у вихідному документі у фіксовані групи байтів. Наприклад, ці частини можуть бути словами (послідовність символів, які обмежені пробілами чи символами кінця рядка). Результат перетворення утворює таблицю, яка є словником, і додається до стисненого коду.

Зазвичай для англомовних текстів використовується двобайтове кодування слів. Пари байтів, що утворюються таким чином, отримують назву "токени".

Ефективність цих алгоритмів сильно залежить від розміру документа, оскільки для коротких документів потрібно додатково включати словник, що призводить до збільшення обсягу і не завжди полегшує стиснення [3].

Найефективніший алгоритм для англомовних текстів та баз даних. Однак для документів написаних українською, де слова можуть бути довшими і містити багато афіксів і закінчень, використання двобайтових токенів не завжди ефективно, і продуктивність алгоритму падає.

Алгоритми стиснення цієї групи найкраще працюють з великими обсягами текстових даних, але менш ефективні для невеликих файлів, особливо тоді, коли потрібно зберігати словник.

Існує чимало реалізацій цього методу, серед яких найпоширенішими є алгоритм Лемпеля-Зіва (LZ) та його вдосконалена версія – алгоритм Лемпеля-Зіва-Велча (LZW). Алгоритми LZ враховують зв'язки між символами, що дозволяє збільшити стиснення даних

Алгоритм Лемпеля-Зіва (LZ)

Основа цього алгоритму полягає у заміні послідовностей символів, що повторюються на покажчики тих місць, де вони вже зустрічалися у тексті. Словник цієї системи складається з потенційно нескінченного списку фраз. Початково словник майже порожній і містить лише одну закодовану фразу - рядок NULL.
Коли наступний символ у вихідній послідовності даних зчитується, він додається до поточного рядка. Цей процес триває, доки поточний рядок відповідає якій-небудь фразі у словнику. Проте часом поточний рядок не збігається з жодною фразою у словнику. У такий момент, коли поточний рядок стає останнім збігом зі словником і в ньому з'являється ще один символ, кодер видає код, який складається з індексу збігу та наступного символу, що порушив збіг. Нова фраза, яка складається з цього збігу та символу, додається до словника. Якщо ця фраза зустрінеться знову у тексті, вона використовується для побудови більшої фрази, що допомагає у стисненні інформації.

Алгоритм Лемпеля-Зіва-Велча (LZW)

Алгоритм LZW ґрунтується на таблиці фраз, яка замінює послідовності символів у повідомленні, що стискається, на фіксовані коди. Ця таблиця має

своєрідну особливість, відому як властивість випередження: якщо конкретна послідовність символів w та символ K вже є у таблиці, то саме w також додається до словника. Коли всі можливі комбінації у словнику вже присутні, процес кодування стає неадаптивним і базується на наявних фразах у словнику.

Процес роботи алгоритму включає наступні кроки:

1. Ініціалізація словника: У першому етапі словник заповнюється початковими символами або символьними фразами.
2. Кодування даних: Під час кодування тексту або послідовності символів алгоритм знаходить найбільш довгий фрагмент, який вже міститься у словнику. Він заміняє цей фрагмент відповідним кодом зі словника.
3. Розширення словника: Після кожного успішного знаходження фрагменту та його заміни кодом, алгоритм додає нову фразу у словник, поєднуючи попередній фрагмент з наступним символом у вихідному тексті.
4. Створення стисненого потоку даних: Коди, які представляють заміщені фрагменти, утворюють стиснений потік даних.

Однією з переваг LZW є те, що він не вимагає попередньої передачі словника. Це дозволяє створювати стислі коди без необхідності передачі повного словника спільно з стислим текстом. Алгоритм добре працює з текстовою інформацією, де є багато повторень або фрагментів з однаковою структурою.

2.5 Порівняння алгоритмів стиснення та вибір оптимального алгоритму

У даному розділі буде проведено порівняльний аналіз у вигляді таблиці кількох алгоритмів стиснення даних без втрат. Це дозволить виокремити сильні сторони кожного алгоритму та обрати оптимальний метод стиснення.

Таблиця 2.5 – Порівняння характеристик алгоритмів стиснення без втрат

Алгоритм	Ступінь стиснення	Швидкість роботи	Затрати пам’яті	Кількість проходів	Надмір ність
Шеннона-Фано	Середній	Середня	Середні	1	Рідко
Алгоритм Хаффмана	Вище середнього	Висока	Великі	2	Рідко
Адапт-ий алгоритм Хаффмана	Вище середнього	Вище середньої	Вище середнього	1	Рідко

Продовження таблиці 2.5

Алгоритм	Ступінь стиснення	Швидкість роботи	Затрати пам'яті	Кількість проходів	Надмірність
Арифм. кодування	Високий	Середня	Великі	2	Рідко
RLE	Низький	Висока	Середні	1	Можливо
LZ	Високий	Нижче середнього	Невеликі	1	Рідко
LZW	Середній	Вище середнього	Середні	1	Рідко

Зважаючи на дані представлені в табл. 2.5, робимо висновок, що адаптивний алгоритм Хаффмана має найбільш придатні для використання характеристики. Також адаптивний алгоритм Хаффмана має декілька переваг, які роблять його привабливим в багатьох сценаріях стиснення даних, особливо в області лічильників.

– Адаптивність: Цей алгоритм може змінювати свій словник під час процесу стиснення. Він підлаштовується під унікальні властивості конкретних даних з лічильника, що дозволяє йому ефективно пристосовуватися до змін в характеристиках даних та надає більш ефективне стиснення.

– Гнучкість: Адаптивний Хаффман дозволяє додавати нові символи або змінювати частоти символів без необхідності повторної побудови всього дерева кодування. Це зручно в ситуаціях, коли частота символів може змінюватись з часом.

– Добре справляється з текстовими даними: У випадку текстової інформації, де патерни повторюваності можуть бути помітними, Адаптивний Хаффман може ефективно використовувати ці патерни для стиснення.

– Ефективність для великих обсягів даних: Даний алгоритм часто проявляє себе добре при обробці великих обсягів даних, забезпечуючи хороший коефіцієнт стиснення та роблячи розпакування прийнятним за часом.

Таким чином, використання Адаптивного алгоритму Хаффмана для стиснення даних з лічильників може бути досить доцільним, оскільки він має можливість адаптуватися до змін в даних, ефективно використовувати їх патерни для стиснення та забезпечувати гнучкість управління словником кодування.

3 РОЗРОБЛЕННЯ АЛГОРИТМУ РОБОТИ ТА СХЕМИ ЕЛЕКТРИЧНОЇ СТРУКТУРНОЇ СИСТЕМИ ЗБОРУ ДАНИХ

3.1 Розроблення алгоритму роботи системи

Даний підрозділ містить у собі розроблену блок схему алгоритм роботи автоматизованої електронної системи.

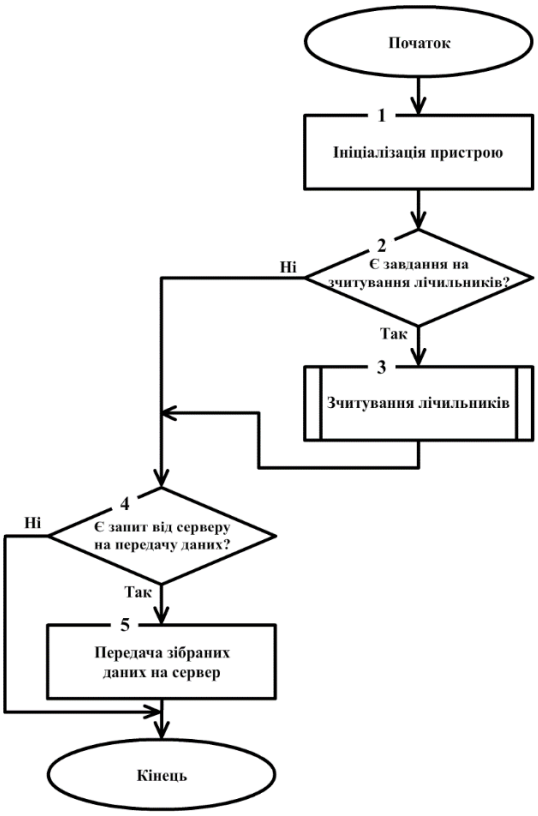


Рисунок 3.1 – Блок схема алгоритму роботи автоматизованої електронної системи

Алгоритм роботи автоматизованої електронної системи полягає в наступному:

Крок 1. Ініціалізація пристрою.

Крок 2. Перевірка, чи є завдання на зчитування лічильників. Якщо так, переходимо до наступний крок, якщо ні – до кроку 4.

Крок 3. Зчитування лічильників.

Крок 4. Перевірка, чи є запит від серверу на передачу даних. Якщо так, переходимо до наступного кроку, якщо ні – алгоритм завершується.

Крок 5. Передача зібраних даних на сервер.

3.2 Розроблення структурної схеми системи

В даному розділі розроблено структурну схему автоматизованої електронної системи збору даних із застосуванням Smart хабу. Структурна схема системи наведена на рисунку 3.2.

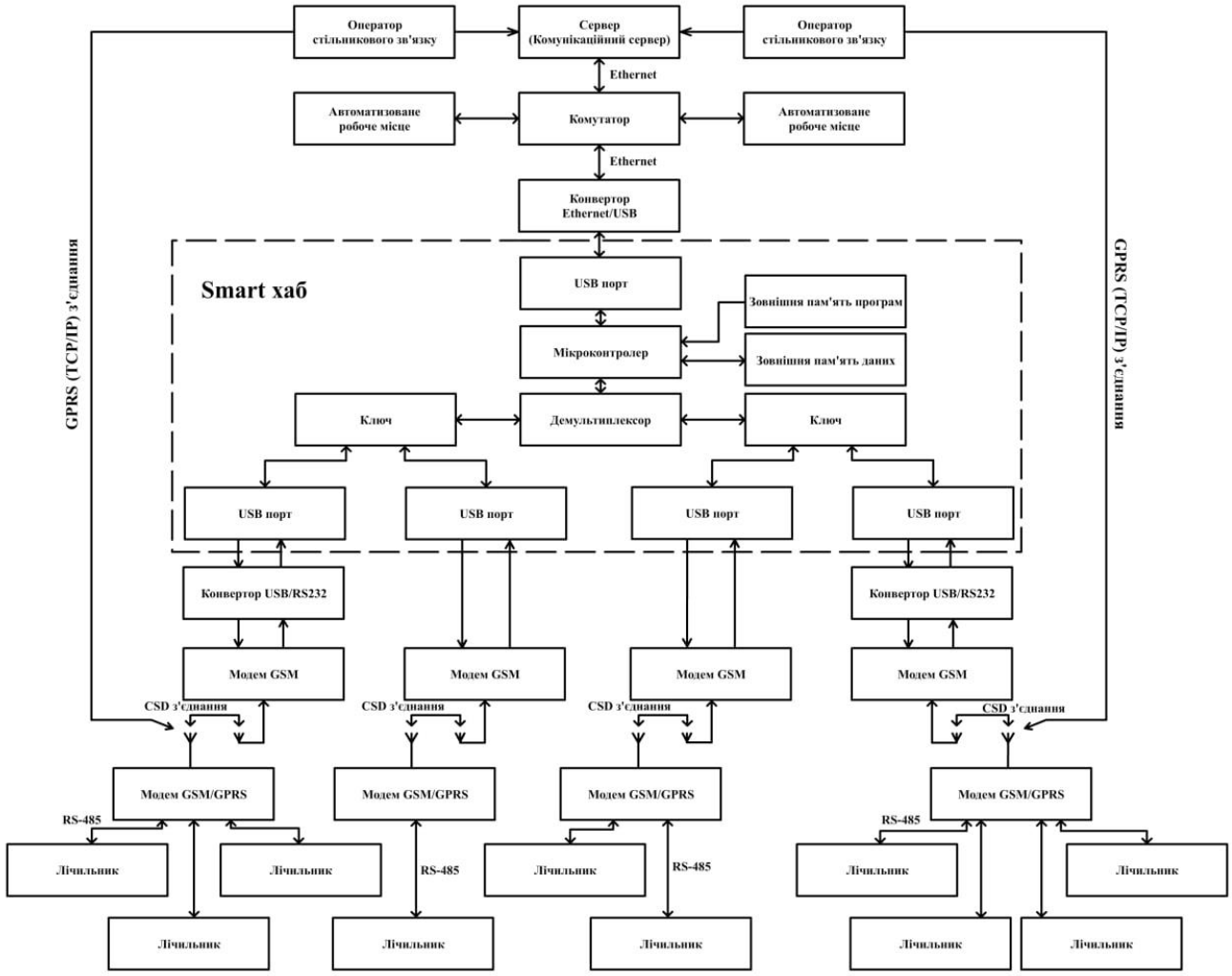


Рисунок 3.2 – Структурна схема автоматизованої електронної системи збору даних із застосуванням Smart хабу

Дана структурна схема складається з наступний блоків:

Блок серверу – в даному блоці зберігається дані з лічильників, які передає Smart хаб.

Блок оператора стільникового зв’язку – блок необхідний для виконання GPRS(TCP/IP) з’єднання.

Блок комутатора – використовується для передачі даних між пристроями всередині локальної мережі, визначаючи, куди направити інформацію на основі MAC-адрес пристроїв.

Блок автоматизоване робоче місце – дозволяє автоматизувати велику кількість рутинних завдань, отримувати швидкий доступ до даних про енергосистему, а також виконувати аналізи для прийняття рішень.

Блок конвертору Ethernet/USB – даний блок дозволяє підключати пристрої з портом USB до мережі Ethernet.

Блок Smart хабу – даний блок забезпечує автоматичний збір даних з лічильників, контроль стану лічильників, декодування зібраних даних, зберігання декодованих даних, індивідуальне живлення портів хабу, блокування модемів, блокування читання лічильників, передача даних на сервер.

Блок конвертору USB/RS-232 – блок дозволяє підключати пристрої з портом USB до пристроїв з портом RS-232.

Блок модему GSM – блок дозволяє виконати з’єднання з підстанційним модемом та передавати дані.

Блок модему GSM/GPRS – підстанційний модем, який дозволяє передавати дані по каналу зв’язку та кодувати дані з лічильника.

Блок лічильника – здійснює контроль витрат електроенергії та фіксує погодинні результати цього контролю.

3.3 Розроблення алгоритму роботи Smart хабу

Процес зчитування лічильників доволі тривалий. Тому є сенс надати алгоритм процедури зчитування на рисунку 3.3

Приклад роботи алгоритму:

Якщо програмному модулю хабу (надалі ПМХ) потрібно отримати дані з лічильника, який з'єднаний з підстанційним модемом, що має номер, скажімо, 0503735878, і цей модем використовується для зчитування через телефонний модем, який, у свою чергу, підключений до порту COM-3, то програмний модуль хабу звертається до порту COM-3.

Якщо з'єднання ПМХ до COM-3 успішно – здійснюється дзвінок з модему на підстанційний модем з номером 0503735878.

Якщо програмний модуль хабу при з'єднанні з портом COM-3 виявляє, що телефонуючий модем, який використовує цей порт, зайнятий (читає інший

лічильник), то програмний модуль хабу підключається до альтернативного засобу зв'язку. Якщо альтернативний модем, який прикріплений, наприклад, до порту COM-5, також зайнятий (читає свій власний лічильник), то програмний модуль хабу періодично спробує спочатку з'єднатися з COM-3, а потім з COM-5, до тих пір, поки один з модемів не вільний від попереднього зчитування. Лише після цього модулю хабу надсилається команда на зв'язок з модемом, що має номер 0503735878. Блокування підстанційного модему з номером 0503735878 при певній кількості невдалих спроб має сенс лише у разі відсутності зв'язку з підстанційним модемом.

Якщо програмний модуль хабу звертається до порту COM-3 або COM-5 і виявляє, що немає з'єднання, він спробує виконати тест модему шляхом відправки AT команди. Якщо від модему отримано позитивний відгук «ОК», програмно виконується перезавантаження модему. Після цього процес повертається до початку.

У випадку, якщо позитивного відгуку немає, потрібно перезавантажити живлення модему. Треба відправити SMS-повідомлення з працездатного телефонного модему, щоб інформувати оператора про це. Якщо є рестартер для пулу телефонних модемів, застосовується дистанційний рестарт рестартера за допомогою спеціальних команд живлення.

Якщо вдалося підключитися до порту COM і здійснити з'єднання з підстанційним модемом (далі - ПС-модем) за номером 0503735878, і виявлено, що цей модем вже використовується, раціонально спробувати здійснити дзвінок через хвилину.

У разі важкості з цим, розумно пропустити виклик і повернутися до спроби зчитування цього модему на наступному етапі.

Якщо ПС-модем не відповідає на дзвінок (немає можливості встановити TCP/IP з'єднання) через поганий сигнал, потрібно здійснити встановлену кількість дзвінків і тимчасово заблокувати дзвінки на цей GSM-модем на певний період.

Якщо ПС-модем є GSM/GPRS модемом (1 SIM-карта) і основним засобом зв'язку є TCP/IP з'єднання, що не може встановитися декілька разів підряд на запит від сервера, перед блокуванням звернення до цього модему, слід спробувати з'єднання з ПС-модемом за допомогою CSD-з'єднання. Якщо після встановленої кількості CSD з'єднань не вдасться з'єднатися, здійснюється блокування. Якщо ПС-модем є GSM/GPRS модемом (2 SIM-карти), після невдалої спроби встановити

зв'язок основною SIM-картою основного оператора мобільного зв'язку, слід здійснити CSD-дзвінок на другу резервну SIM-карту резервного оператора мобільного зв'язку, а в разі невдачі заблокувати звернення до цього 2-SIM-кового модему.

Якщо підстанційний модем встановив зв'язок із лічильником на необхідній швидкості, ПМХ звертається до лічильника згідно з планом завдань у планувальнику. Зчитувані дані проходять процес стиснення в ПС-модемі та розтискання в Smart хабі. Після успішного отримання даних з цього лічильника з'єднання із лічильником припиняється, а також перевіряється наявність інших лічильників, що підключені до цього ж модему. Якщо є ще лічильники, зв'язок з модемом не переривається, і проводиться поетапне зчитування всіх лічильників, які підключені до цього модему.

Якщо виникає випадок невдалого зчитування будь-якого з цих лічильників, цей лічильник буде продовжувати читатись встановлену кількість разів. Після цього звернення саме до цього лічильника буде тимчасово заблоковане, і зчитування переходить до наступного лічильника у послідовності. Після зчитування стиснених даних з усіх лічильників у цьому ланцюжку, деякі з яких можуть бути заблоковані, з'єднання з модемом буде розірване, а отримані дані будуть збережені.

При повторних зверненнях до модему, до лічильника, під час перемикання модему на резервні комунікації, під час переходу читання від лічильника до лічильника при відкритої комунікації з модемом необхідно вводити паузи. Паузи необхідно для коректного закриття поточних сеансів зв'язку лічильників та модемів. Паузи для лічильника 2 с, для модемів – 5 сек.

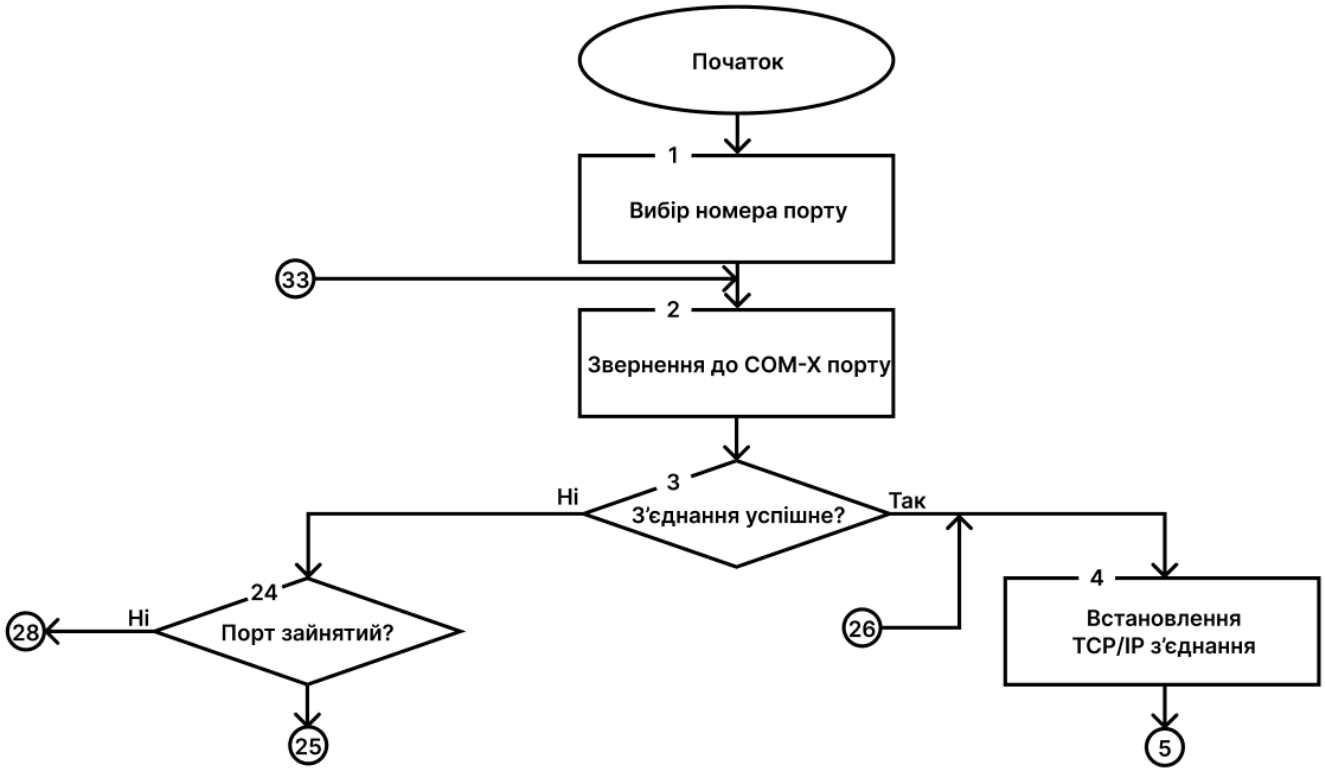


Рисунок 3.2 – Блок схема алгоритму процедури зчитування лічильників

Алгоритм процесу зчитування лічильників полягає в наступному:

Крок 1. Вибір номера порту.

Крок 2. Звернення до COM-X порту.

Крок 3. Перевіряємо, чи було звернення успішним. Якщо так, переходимо до кроку 4, якщо ні – до кроку 24.

Крок 4. Встановлення Комунікаційним сервером TCP/IP з'єднання.

Крок 5. Перевіряємо, чи встановлено TCP/IP з'єднання. Якщо так, виконуємо крок 6, якщо ні – переходимо до кроку 13.

Крок 6. Зчитуємо дані з лічильника.

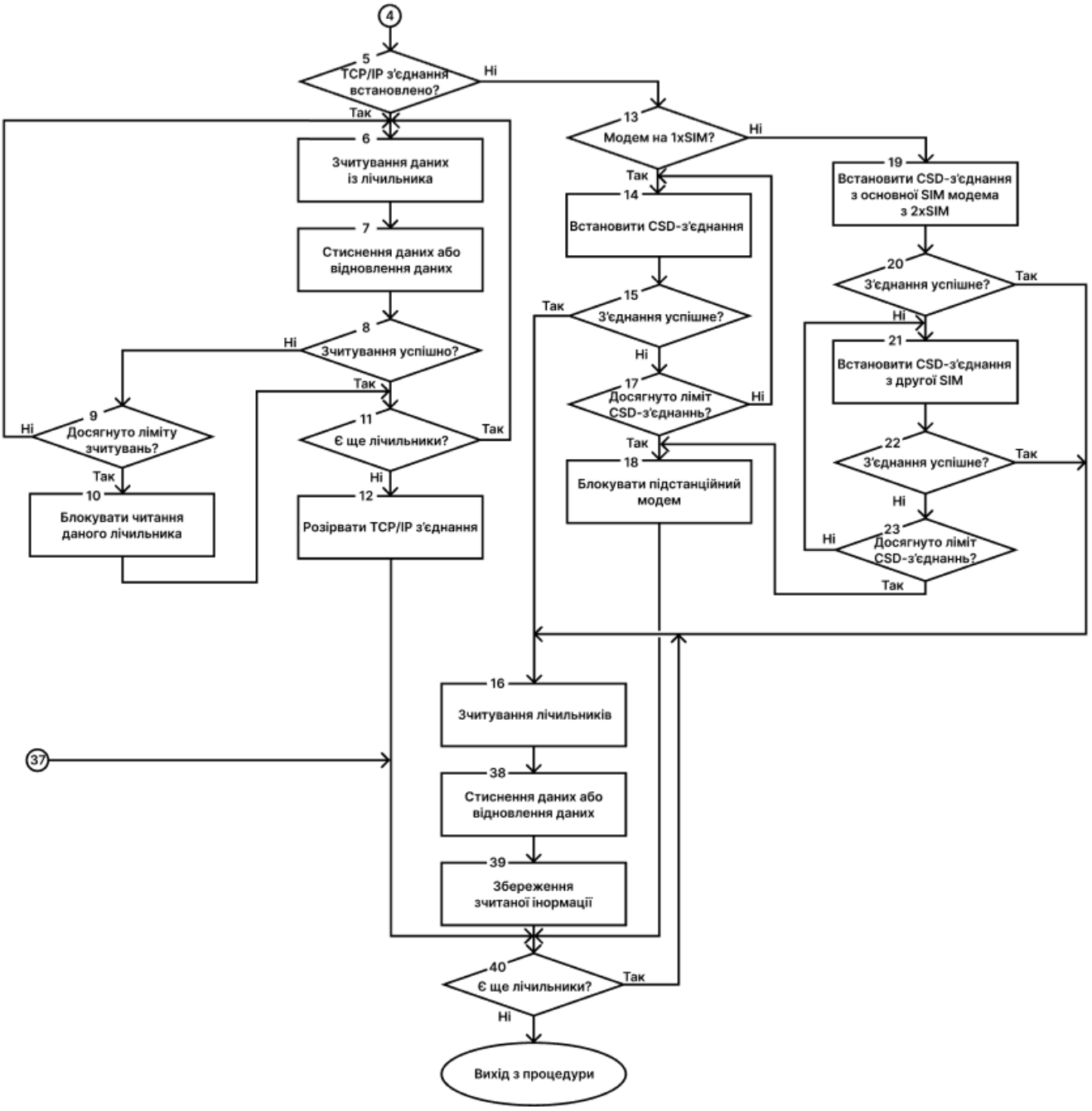
Крок 7. Стиснення даних або відновлення даних.

Крок 8. Перевіряємо, чи було зчитування успішним. Якщо ні, переходимо до кроку 9, якщо так – до кроку 11.

Крок 9. Перевіряємо, чи досягли ліміту зчитувань. Якщо ні, повертаємося до кроку 6, якщо так – переходимо до наступного кроку.

Крок 10. Блокуємо читання опитуваного лічильника.

Крок 11. Перевіряємо, чи є ще лічильники. Якщо так, переходимо до кроку 6, якщо ні – переходимо до кроку 12.



Продовження блок схеми рисунка 3.2

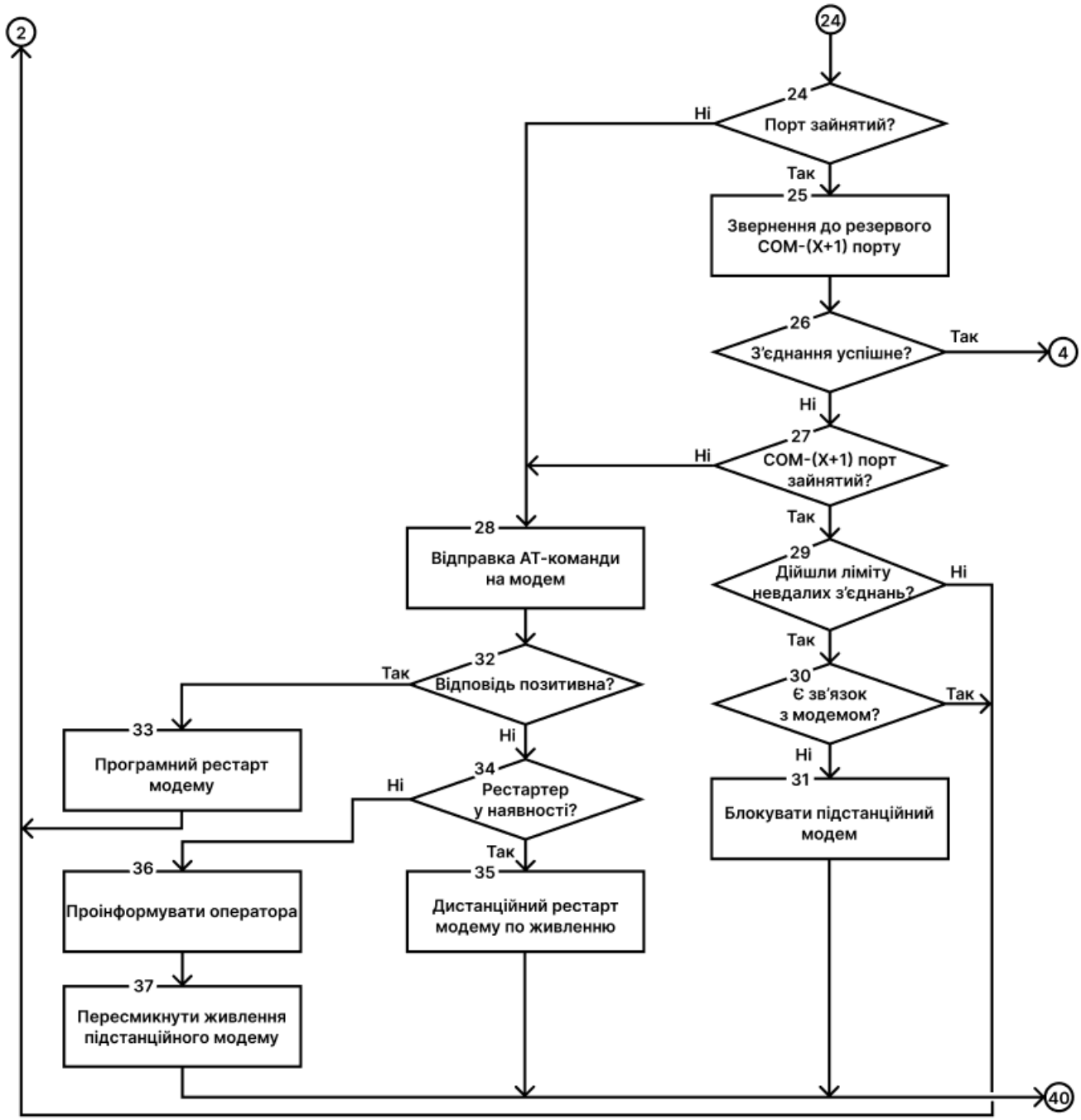
Крок 12. Розриваємо TCP/IP з’єднання.

Крок 13. Перевіряємо, чи використовуваний модем на 1 SIM. Якщо ні, переходимо до кроку 19, якщо так – до кроку 14.

Крок 14. Встановлюємо CSD-з’єднання.

Крок 15. Перевіряємо, чи було з’єднання успішним. Якщо так, то переходимо до кроку 16, якщо ні – до кроку 17.

Крок 16. Зчитуємо дані з лічильника.



Продовження блок схеми рисунка 3.2

Крок 17. Перевіряємо, чи досягли ліміту CSD-з'єднань. Якщо ні, то переходимо до кроку 14, якщо так – до кроку 18.

Крок 18. Блокувати підстанційний модем.

Крок 19. Встановлюємо CSD-з'єднання з основної SIM модему на 2 SIM.

Крок 20. Перевіряємо, чи було з'єднання успішним. Якщо так, то переходимо до кроку 16, якщо ні – до кроку 21.

Крок 21. Встановлюємо CSD-з'єднання з другої SIM.

Крок 22. Перевіряємо, чи було з’єднання успішним. Якщо так, то переходимо до кроку 16, якщо ні – до кроку 23.

Крок 23. Перевіряємо, чи досягли ліміту CSD-з’єднань. Якщо ні, то переходимо до кроку 21, якщо так – до кроку 18.

Крок 24. Перевіряємо, чи порт зайнятий. Якщо ні, то переходимо до кроку 28, якщо так – до кроку 25.

Крок 25. Звертаємося до резервного COM-(X+1) порту.

Крок 26. Перевіряємо, чи було звернення успішним. Якщо так, то переходимо до кроку 4, якщо ні – до кроку 27.

Крок 27. Перевіряємо, чи порт COM-(X+1) зайнятий. Якщо ні, то переходимо до кроку 28, якщо так – до кроку 29.

Крок 28. Відправка AT команди на модем

Крок 29. Перевіряємо, чи досягли ліміту невдалих з’єднань. Якщо ні, то переходимо до кроку 2, якщо так – до кроку 30.

Крок 30. Перевіряємо, чи є зв’язок з модемом. Якщо ні, то переходимо до кроку 31, якщо так – до кроку 2.

Крок 30. Блокувати підстанційний модем та перейти до кроку 40.

Крок 32. Перевіряємо, чи позитивна відповідь на AT команду. Якщо ні, то переходимо до кроку 34, якщо так – до кроку 33.

Крок 33. Виконуємо програмний рестарт модему та повертаємося до кроку 2.

Крок 34. Перевіряємо, чи у наявності рестартер. Якщо ні, то переходимо до кроку 36, якщо так – до кроку 35.

Крок 35. Виконуємо дистанційний рестарт модему по живленню та переходимо до кроку 40.

Крок 36. Повідомляємо оператора SMS-повідомленням.

Крок 37. Пересмикуємо живлення підстанційного модема та переходимо до кроку 40.

Крок 38. Стиснення даних або відновлення даних.

Крок 39. Збереження зчитаної інформації.

Крок 40. Перевіряємо, чи є ще лічильники. Якщо так, переходимо до 16 кроку, якщо ні – виходимо з підпрограми.

3.4 Розроблення структурної схеми Smart хабу

Розроблення структурної схеми для Smart хабу буде базуватися на вимогах, описаних у попередньому розділі.

Цей пристрій, згідно зі своєю назвою, має базуватися на USB-хабі і призначений для передачі інформації з вихідного порту на кілька вхідних портів. Цю роль чудово виконує демультиплексор, який має аналогічні функції.

Для відтворення певного Smart рівня необхідно, щоб USB-хаб мав змогу діяти як основний контролер. Це потребує встановлення у хаб спеціального контролера, який за допомогою програми, що знаходиться у зовнішній енергонезалежній пам'яті програм, буде забезпечувати роботу, яка була описана раніше. Цей контролер має включати в себе функції: збір даних з лічильників, контроль їх стану, кодування та декодування даних, збереження та передача цих даних, управління живленням портів хабу, а також можливість блокувати модеми та лічильники за необхідності.

Після того, як дані з лічильників будуть зібрані, їх потрібно зберегти. Оскільки пам'яті у мікроконтролері не вистачає, виникає необхідність встановлення додаткового зовнішнього сховища для цих даних.

Для забезпечення можливості контролювати живлення портів хабу, потрібно включити в схему блок ключа. Цей блок при отриманні певного рівня сигналу на свій вхід буде увімкати або вимикати живлення на модемі.

Виправдовуючи свою назву – USB хаб – треба додати один висхідний та декілька низхідних USB портів типу A.

Підсумовуючи все вище згадане, було розроблено структурну схему, яка наведена на рисунку 3.3.

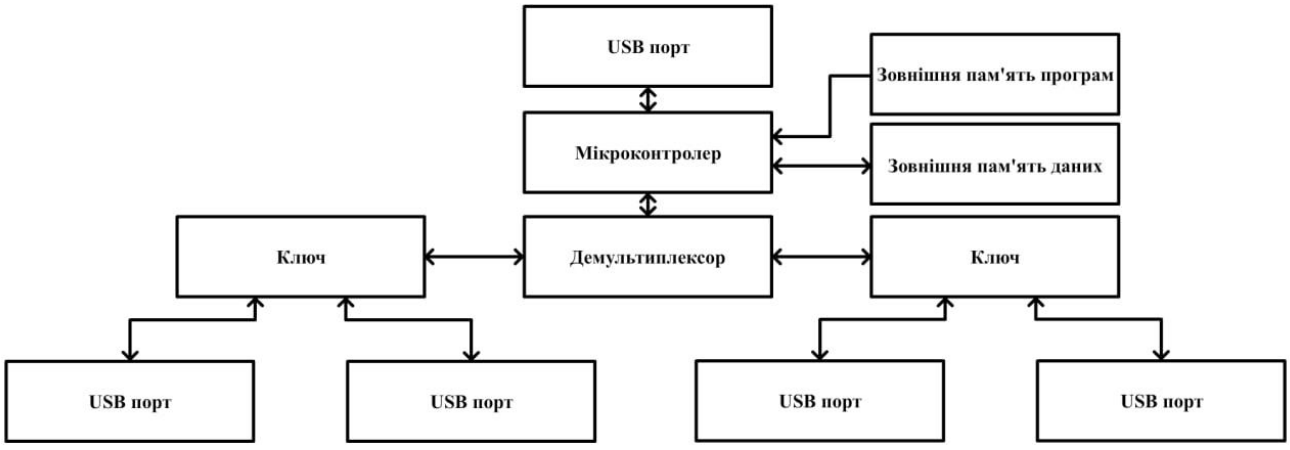


Рисунок 3.3 – Структурна схема Smart хабу

3.5 Розроблення алгоритму роботи пристрою стиснення

В даному підрозділі наведено розроблений алгоритм роботи пристрою стиснення.

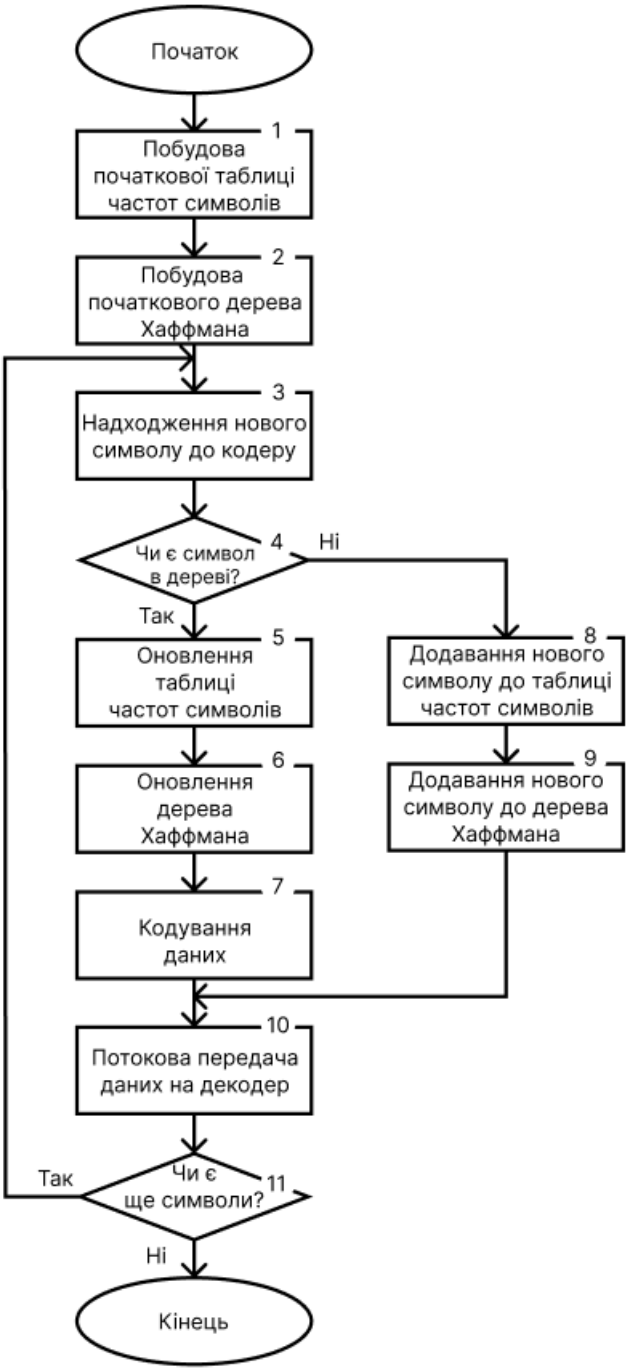


Рисунок 3.4 – Алгоритм роботи пристрою стиснення

Алгоритм роботи пристрою стиснення базується на алгоритмі Хаффмана та полягає в наступному:

Крок 1. Побудова початкової таблиці частот символів.

Крок 2. Побудова початкового дерева Хаффмана.

Крок 3. Надходження нового символу до кодеру.

Крок 4. Перевірка, чи є символ в дереві. Якщо так, то переходимо до кроку 5, якщо ні – до кроку 8.

Крок 5. Оновлення таблиці частот символів

Крок 6. Оновлення дерева Хаффмана.

Крок 7. Кодування даних.

Крок 8. Додавання нового символу до таблиці частот символів.

Крок 9. Додавання нового символу до дерева Хаффмана.

Крок 10. Потокова передача даних на декодер.

Крок 11. Перевірка, чи є ще нові символи. Якщо так, то переходимо до кроку 3, а якщо ні – алгоритм роботи завершується.

3.6 Розроблення структурної схеми пристрою стиснення

Згідно алгоритму роботи та поставленої задачі було розроблено структурну схему пристрою стиснення. Структурна схема наведена на рисунку 3.5.

До розробленої структурної схеми входять такі блоки, як:

- Блок джерела інформації – лічильник з якого знімають данні.
- Блок кодеру – блок, який кодує дані адаптивним алгоритмом Хаффмана;
- Блок декодеру – блок, який декодує повідомлення;
- Блок пам’яті дерева – блок, в якому зберігається дерево Хаффмана;
- Блок модулятора – блок, який модулює повідомлення перед каналом зв’язку;
- Блок демодулятора – блок, який демодулює повідомлення з каналу зв’язку;
- Блок каналу зв’язку – канал зв’язку через який передаються дані;
- Блок модему – блок, через який приймає та з якого відправляє дані хаб;
- Блок приймача інформації – блок, який є кінцевим отримувачем даних з лічильників.

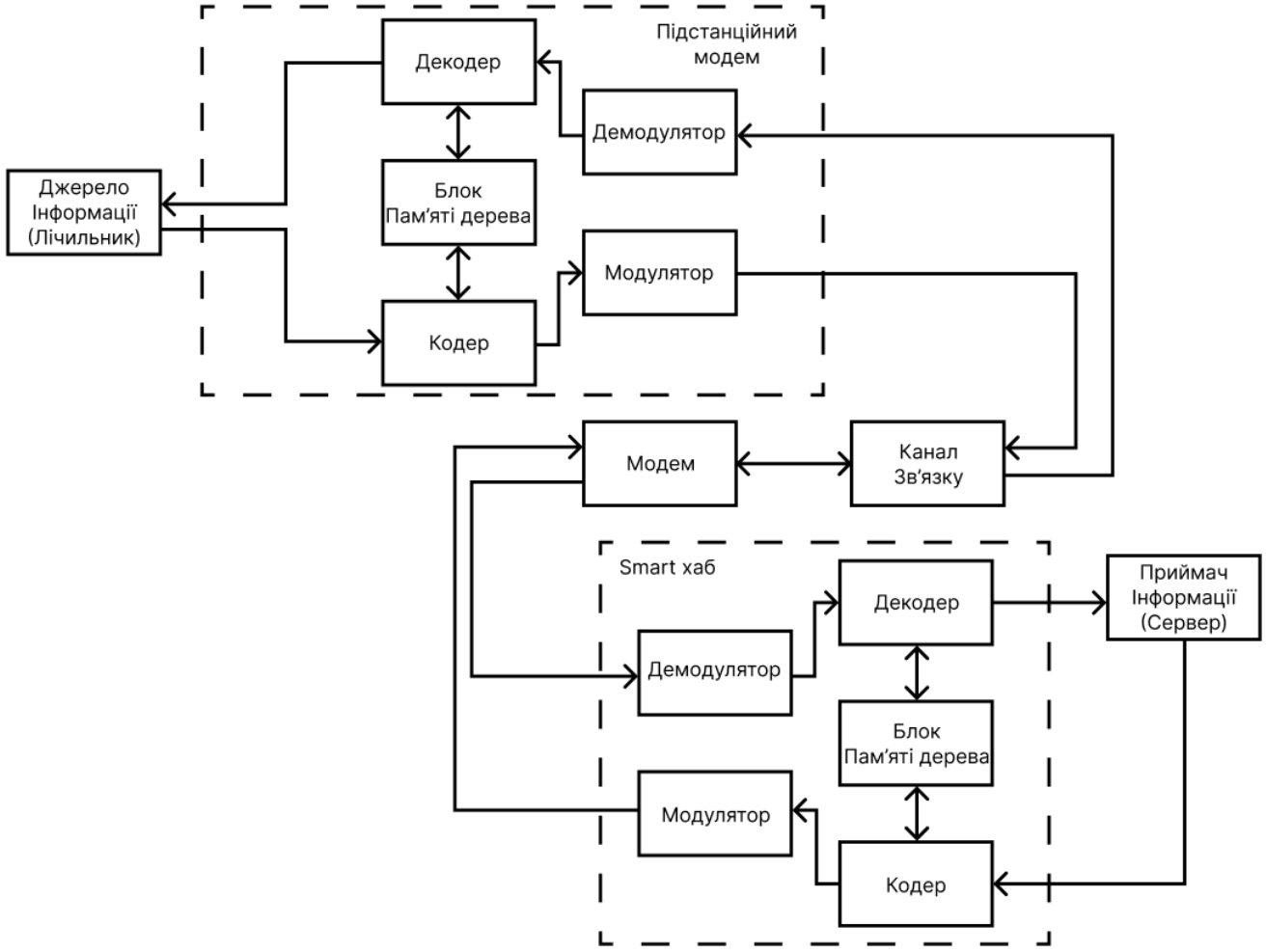


Рисунок 3.5 – Структурна схема пристрою стиснення

4 РОЗРОБЛЕННЯ СХЕМИ ЕЛЕКТРИЧНОЇ ФУНКЦІОНАЛЬНОЇ ПРИСТРОЮ

Мета цього розділу полягає в детальному дослідженні та розробці функціональної схеми Smart хабу. Цей етап відіграє важливу роль у забезпеченні сумісності, ефективності та функціональності пристрою, дозволяючи йому стати центральним елементом у системі розумного управління.

Розробка функціональної схеми Smart хабу – це не лише технічне завдання, а й ключовий стратегічний крок у створенні пристрою.

Спираючись на схему електричну структурну, що була розроблена в минулому розділі, було розроблено схему електричну функціональну.

Функція, яку повинні виконувати блоки USB – передача інформації в обидві сторони. З'єднання висхідного блоку USB з мікроконтролером відбувається через порти DM та DP.

До функцій мікроконтролера можна віднести: автоматичний збір даних з лічильників, передача даних на сервер, блокування одного або декількох модемів, блокування читання одного або декількох лічильників, декодування зібраних даних. Мікроконтролер з'єднується з пам'яттю EERAM контактами PA2 та PA3, а з пам'яттю EEPROM – контактами PA0, PA1. Використовуючи порти DM2, DP2 мікроконтролер з'єднується з демультиплексором.

Демультиплексор відслідковує сигнали з блоку ключа через порти $\overline{\text{PWR}}$ та $\overline{\text{OVL}}$ та передає цю інформацію до мікроконтролера через контакти DM0, DP0. Використовуючи контакти DM1, DP1, DM2, DP2, DM3, DP3, DM4, DP4 демультиплексор пропускає через себе команди мікроконтролера на декілька низхідних портів.

Основною функцією ключів у схемі є відслідковування перегріву чи короткого замикання на низхідних портах USB та можливість вимкнути живлення цих портів після надходження команди.

Схема електрична функціональна наведена на рисунку 4.1.

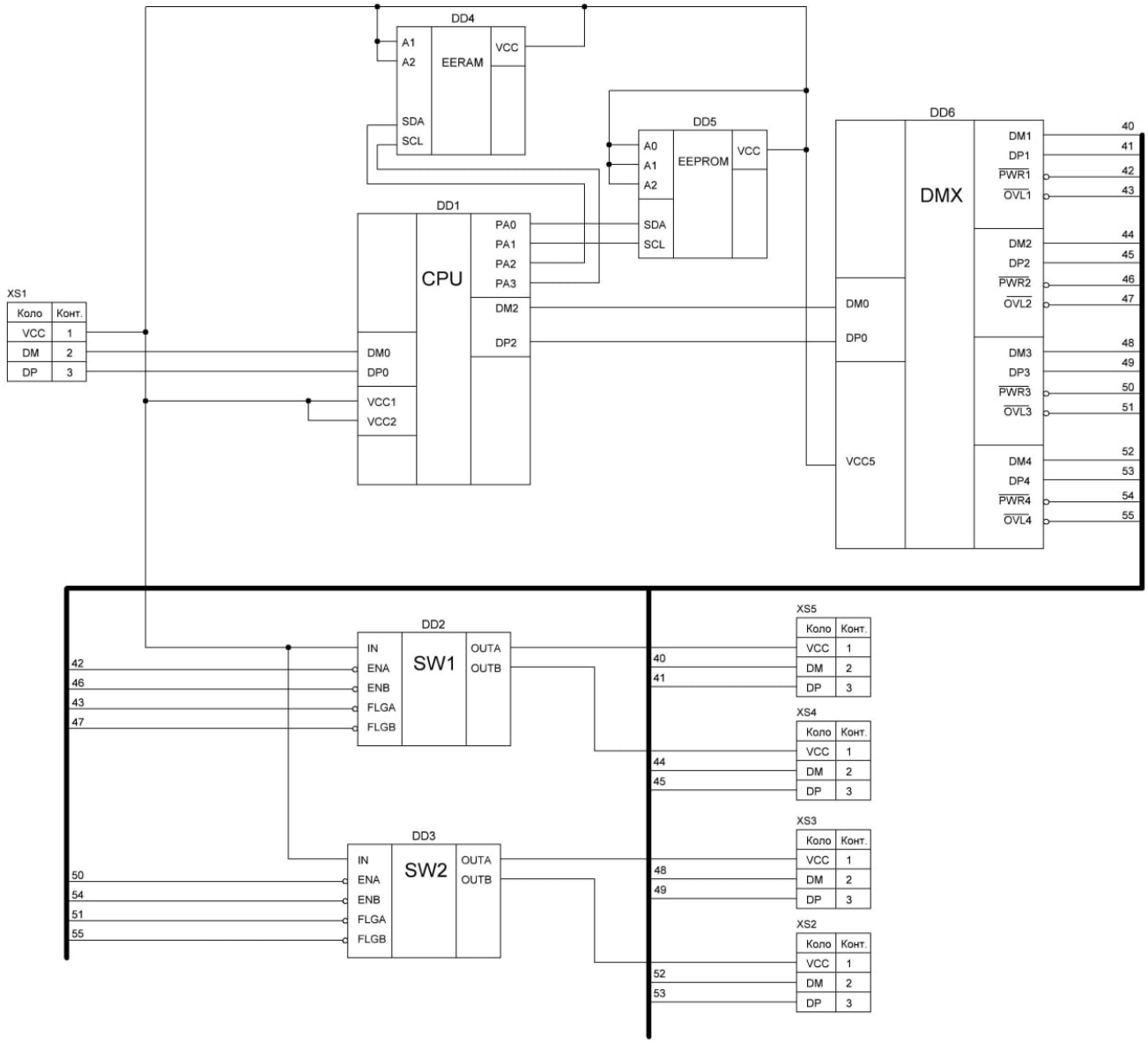


Рисунок 4.1 – Функціональна схема Smart хабу

5 РОЗРОБЛЕННЯ СХЕМИ ЕЛЕКТРИЧНОЇ ПРИНЦИПОВОЇ ПРИСТРОЮ

5.1 Вибір елементної бази

Зважаючи на умови побудови пристрою необхідно обрати елементну базу та розробити схему електричну принципову Smart хабу модемного пулу автоматизованої системи з параметрами:

- розрядність 8 біт;
- напруга живлення від 4.4 до 5.25 В;
- струм живлення 3А;
- робоча температура пристрою від -40 до 85 °С;
- максимальна кількість опитувальних лічильників – 128.

Smart хаб повинен забезпечувати функції відповідних блоків у структурній схемі, а саме:

- можливість керування живленням портів;
- збереження зібраної інформації;
- передача зібраної інформації;
- збір інформації с низхідних портів.

Щоб побудувати пристрій Smart хаб модемного пулу автоматизованої системи, слід вибрати декілька мікросхем, на яких буде реалізовані блоки пристрою.

Аналізуючи вищезазначені умови та функціонал приладу підібрані такі мікросхеми:

- мікроконтролер – AT43USB353M-Atmel;
- зовнішня пам’ять – 24AA512-Microchip Technology;
- зовнішня пам’ять – 47C16-Microchip Technology;
- демультиплексор – AT43312A-Atmel;
- ключ – MIC2526-2BM-Micrel.

5.2 Мікроконтролер – AT43USB353M-Atmel

AT43USB353M – це USB-мікроконтролер, який використовує архітектуру AVR та має вбудований розмножувач, сумісний з USB 2.0. В нього вбудовані три USB-порти: два зовнішні низхідні USB-порти, які можуть бути підключені до

різних пристроїв, таких як модулі флеш-пам'яті або інші USB-пристрої, і один висхідний порт для з'єднання з іншими пристроями через USB інтерфейс.

Даний мікроконтролер представляє собою високопродуктивний 8-розрядний AVR RISC мікроконтролер, що працює на частоті 12 МГц або 24 МГц. Він має програмну пам'ять розміром 24 Кбайт і область даних об'ємом 1 Кбайт. АЦП має мінімальний час перетворення 12 мкс та включає 12 вхідних каналів. Його 15 програмованих вводів-виводів загального призначення забезпечують достатню кількість для зчитування кнопок і перемикачів, а також керування світлодіодними індикаторами. [6].

Мікроконтролер AT43USB353M оптимально відповідає умовам та функціональним можливостям. Перевагами даного мікроконтролера серед інших аналізованих мікроконтролерів є:

- невеликі розміри;
- низька ціна;
- наявність низхідних USB портів.

Мікроконтролер має 48 виводи, які зображені у корпусі LQFP на рисунку 5.1.

На рисунку 5.2 приведено умовне позначення на принциповій схемі мікроконтролера з портами, які використовуються.

Напруга, яка живить вхідні порти, повинна бути не більше 5.25 В, згідно з технічною документацією для AT43USB353M. [6]

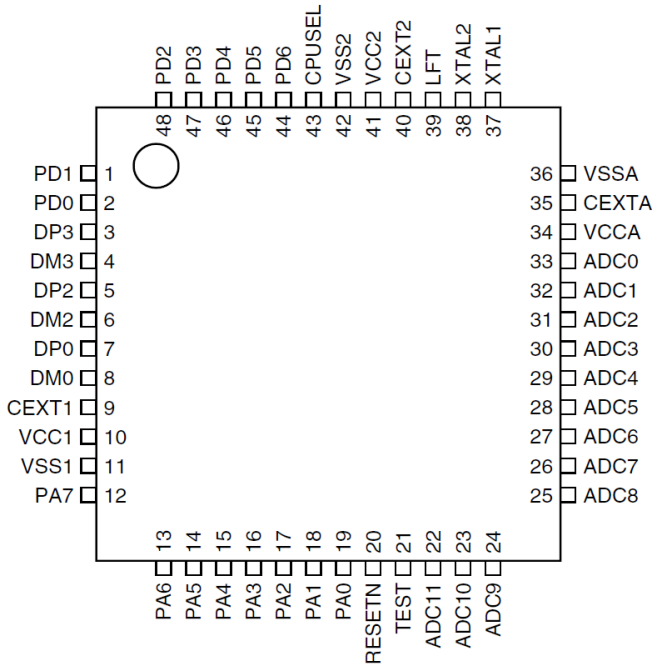


Рисунок 5.1 – 48 виводів мікроконтролера AT43USB353M у корпусі LQFP

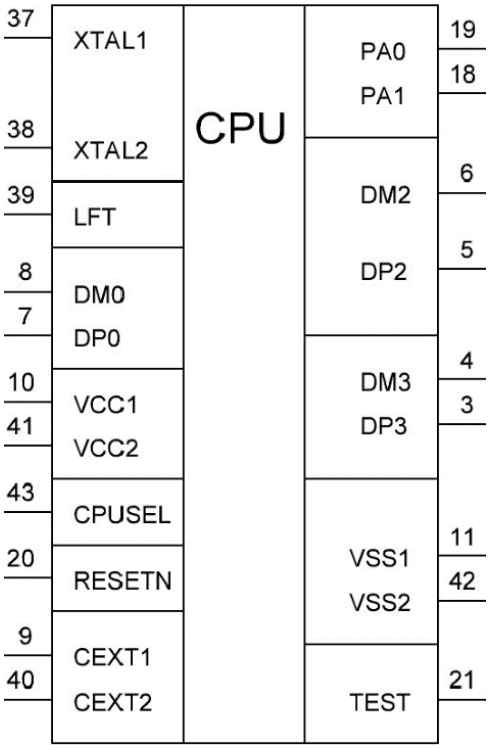


Рисунок 5.2 – Умовне позначення на принциповій схемі мікроконтролера AT43USB353M з портами, які використовуються

Таблиця 5.1 – Призначення портів

№ порту	Назва	Тип	№ порту	Назва	Тип
1	PD1	Двонаправлений	25	ADC8	Вхідний
2	PD0	Двонаправлений	26	ADC7	Вхідний
3	DP3	Двонаправлений	27	ADC6	Вхідний
4	DM3	Двонаправлений	28	ADC5	Вхідний
5	DP2	Двонаправлений	29	ADC4	Вхідний
6	DM2	Двонаправлений	30	ADC3	Вхідний
7	DP0	Двонаправлений	31	ADC2	Вхідний
8	DM0	Двонаправлений	32	ADC1	Вхідний
9	CEXT1	Живлення/Земля	33	ADC0	Вхідний
10	VCC1	Живлення/Земля	34	VCCA	Живлення/Земля
11	VSS1	Живлення/Земля	35	VSSA	Живлення/Земля
12	PA7	Двонаправлений	36	XTAL1	Вхідний
13	PA6	Двонаправлений	37	XTAL2	Вихідний
14	PA5	Двонаправлений	38	LFT	Вхідний
15	PA4	Двонаправлений	39	CEXT2	Живлення/Земля
16	PA3	Двонаправлений	40	VCC2	Живлення/Земля
17	PA2	Двонаправлений	41	VSS2	Живлення/Земля
18	PA1	Двонаправлений	42	CPUSEL	Вхідний
19	PA0	Двонаправлений	43	PD6	Двонаправлений
20	RESETN	Вхідний	44	PD5	Двонаправлений
21	TEST	Вхідний	45	PD4	Двонаправлений
22	ADC11	Вхідний	46	PD3	Двонаправлений
23	ADC10	Вхідний	47	PD2	Двонаправлений
24	ADC9	Вхідний	48	PD2	Двонаправлений

Таблиця 5.2 – Опис призначення виводів мікроконтролера

Вивід	Тип	Опис
PA[0:7]	Двонаправлений	Виводи A[0:7] — двонаправлені 8-бітні порти введення-виведення зі струмом споживання 2 мА та програмованим підтягуючим резистором.
PD[0:6]	Двонаправлений	Виводи D[0:7] — двонаправлені 8-бітні порти введення-виведення зі струмом споживання 2 мА та програмованим підтягуючим резистором. PD[2,3,4,5,6] виконує подвійну функцію, як показано нижче: Альтернативні функції виводів порту PD2 — INT0, зовнішнє переривання 0 PD3 — INT1, зовнішнє переривання 1 PD4 — ICP, таймер/лічильник PD5 — OC1A таймер/лічильник 1, вихід порівняння A PD6 — OC1B таймер/лічильник 1, вихід порівняння B
ADC[0:11]	Вхідний	АЦП [0:11] — 12-бітні вхідні контакти для АЦП
CEXT1, 2	Живлення/Земля	Зовнішні конденсатори для блока живлення — високоякісні конденсатори ємністю 2,2 мкФ повинні бути підключені до CEXT1 та 2 для правильної роботи мікросхеми.
CEXTA	Живлення/Земля	Зовнішні конденсатори для аналогового блоку живлення.
DM0	Двонаправлений	Висхідний мінус USB I/O
DP0	Двонаправлений	Висхідний плюс USB I/O — цей контакт повинен бути підключений до CEXT1 через зовнішній резистор 1,5 кОм.
DM[2,3]	Двонаправлений	Низхідний мінус USB I/O — кожен з цих контактів повинен бути підключений до VSS через зовнішній резистор 15 кОм.
DP[2,3]	Двонаправлений	Низхідний плюс USB I/O — кожен з цих контактів повинен бути підключений до VSS через зовнішній резистор 15 кОм. DP[2,3] и DM[2,3] — це пара контактів диференційного сигналу для підключення USB-пристроїв низхідного потоку.

Продовження таблиці 5.2.

XTAL1	Вхідний	Вхід генератора — вхід інвертуючого підсилювального генератора.
XTAL2	Вихідний	Вихід генератора — вихід інвертуючого підсилювального генератора.
CPUSEL	Вхідний	Вибір швидкості процесора — цей контакт обирає тактову частоту процесора. При високому значенні процесор робить на частоті 12 МГц, при низькому — на частоті 24 МГц.
LFT	Вхідний	Фільтр PLL — для правильної роботи PLL цей вивід повинен бути підключений через конденсатор 0,01 мкФ паралельно до резистора 100 Ом з послідовно підключеним конденсатором 0,1 мкФ на землю (VSS). Обидва конденсатора повинні бути керамічними високої якості.
TEST	Вхідний	Контакт Тесту — цей контакт повинен бути заземлений.
RESETN	Вхідний	Скидання — Активний рівень - низький.
$V_{CC1, 2}$	Живлення/Земля	Цифрове джерело живлення 5 В
V_{CCA}	Живлення/Земля	Джерело живлення 5 В для АЦП
$V_{SS1, 2}$	Живлення/Земля	Цифрова земля
V_{SSA}	Живлення/Земля	Земля для АЦП

5.2.1 Внутрішня структура мікроконтролера

Мікроконтролер з сімейства megaAVR використовує велику частину своїх регістрів управління та стану для роботи з операціями USB, які відображаються в області пам'яті SRAM.

У нього є 32 робочих 8-бітних регістри загального призначення, які можуть бути швидко доступні за один такт процесора. Ці регістри забезпечують операції в рамках арифметико-логічного устрою (АЛУ), що означає виконання однієї операції за один тактовий цикл.

Шість із 32 регістрів можуть використовуватися як три 16-бітових непрямих покажчика регістрів адреси для адресації простору даних, що забезпечує ефективне обчислення адрес. Один із трьох адресних покажчиків також використовується як адресний покажчик для довідкових таблиць у пам'яті програм.

Взаємодія з адресним простором розміром 24 КБ в AVR здійснюється через використання інструкцій відносного переходу та виклику, що дозволяє прямий

доступ до цієї області пам'яті.

Більшість інструкцій в AVR мають формат одного 16-бітного слова. Кожна адреса програмної пам'яті містить 16- або 32-розрядну інструкцію.

Отримання доступу до SRAM даних об'ємом 1 Кбайт у архітектурі AVR може бути здійснено через п'ять різних режимів адресації, які підтримуються. Це різні методи взаємодії з цим обсягом пам'яті для забезпечення зручності та ефективності роботи з даними.

В архітектурі AVR простори пам'яті можуть бути лінійними або звичайними картами пам'яті. Гнучкий модуль переривань має регістри управління, які розташовані у просторі введення-виведення, з додатковим глобальним бітом дозволу переривання у регістрі стану. Усі переривання мають свій окремий вектор, який відображений у таблиці векторів переривань на початку пам'яті програм. Переривання мають пріоритет відповідно до їхньої адреси вектора переривання: чим менше адреса вектора переривання, тим вищий пріоритет у системі. [6]

На рисунку 5.3 зображено внутрішню структуру мікроконтролеру

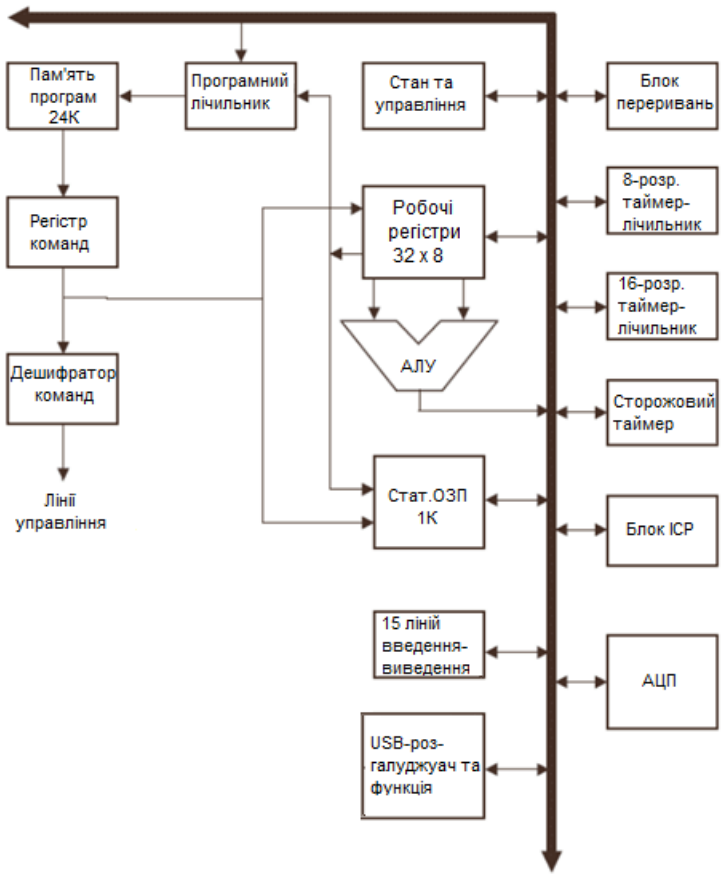


Рисунок 5.3 – Внутрішня структура мікроконтролера

5.2.2 Можливості мікросхеми AT43USB353M як USB-хабу та USB-функції

USB-концентратор AT43USB353M має 3 нижні порти. Порт 1 є постійно підключеним вбудованим портом, а порти 2 і 3 є зовнішніми. Кількість використовуваних портів контролером AT43USB353M визначається прошивкою і може змінюватися від 0 до 2. [6]

Ця конфігурація дозволяє програмній частині (прошивці) визначати функціональність портів 2 і 3 як постійно підключені, навіть у випадку, коли дескриптор концентратора ідентифікує їх як такі..

Вбудована функція USB в мікросхемі має свою власну адресу пристрою та кінцеву точку за замовчуванням, а також ще три програмовані кінцеві точки. Дві з цих кінцевих точок мають власний 64-байтовий FIFO, а третя - 8-байтовий FIFO. Кінцеві точки 1-3 можуть бути налаштовані як кінцеві точки IN або OUT для передачі або отримання даних, можуть бути налаштовані як точки переривання, або як групові кінцеві точки IN або OUT.

Блок-схема USB-обладнання AT43USB353M показана на рисунку 5.4.

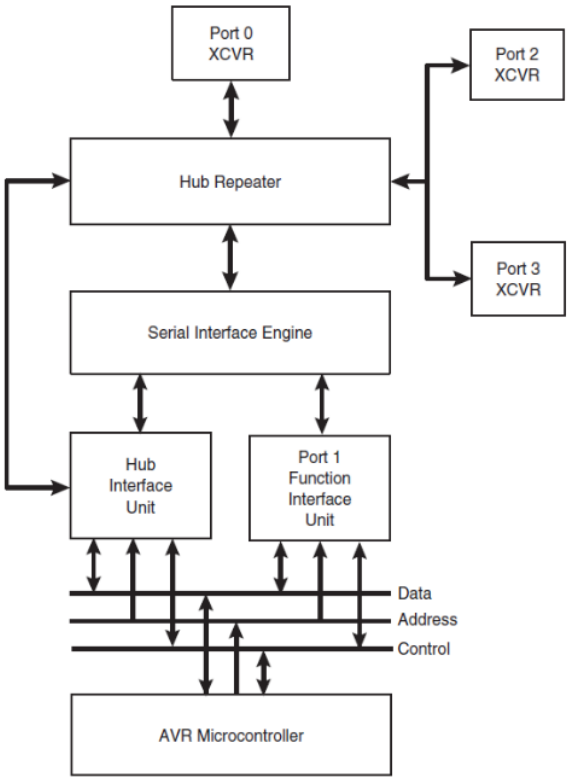


Рисунок 5.4 – Блок схема USB-обладнання мікросхеми AT43USB353M

5.2.3 Генератор та PLL-фільтр мікросхеми AT43USB353M

У мікросхемі AT43USB353M всі необхідні тактові сигнали генеруються внутрішнім генератором. Цей генератор розрахований на роботу з кварцом 6 МГц. Вбудований PLL-фільтр генерує високу частоту розділювача тактових імпульсів/даних модуля послідовного інтерфейсу. У зупиненому стані схема генератора вимкнена.

Схема підключення зображена на рисунку 5.5.

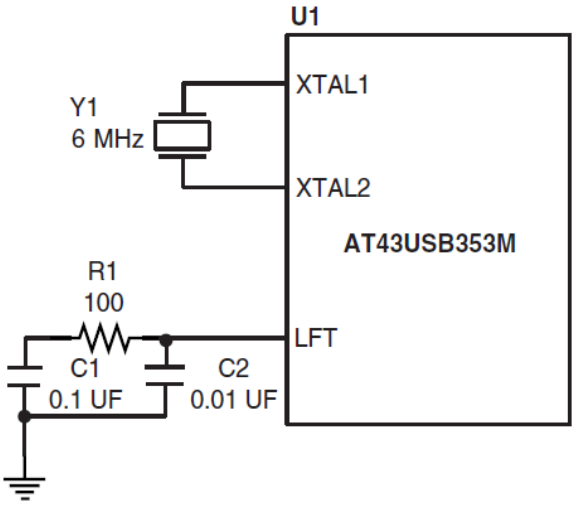


Рисунок 5.5 – Схема підключення портів генератору та фільтру

5.3 Зовнішня пам’ять – 24AA512-Microchip Technology

Мікросхема зовнішньої пам'яті 24AA512 - це ППЗУ з розміром 64К x 8 (512 Кбіт), яке може працювати при різних рівнях напруги. Ця мікросхема підтримує як довільне, так і послідовне зчитування обсягом до 512 Кбіт. Функціональні адресні лінії дозволяють підключити до однієї шини до восьми пристроїв з адресним простором до 4-х Мбіт [7].

24AA512 використовує протокол передачі даних, де пристрій, що відправляє дані, вважається передавачем, а пристрій, що приймає дані, - приймачем. Провідний пристрій, який генерує сигнал SCL, керує шиною, контролює доступ до неї та встановлює умови початку та завершення передачі, тоді як 24AA512 діє як приймальний пристрій. Обидва - і провідний, і приймальний пристрої – можуть

виступати як передавачі чи приймачі, проте режим роботи визначає провідний пристрій.

Мікросхема має 8 виводів, які зображені у корпусі PDIP на рисунку 5.6.

На рисунку 5.7 зображену умовне позначення на принциповій схемі зовнішньої пам’яті.

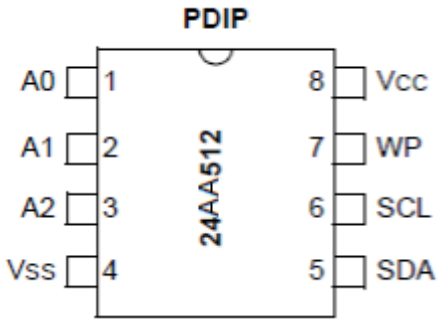


Рисунок 5.6 – Схема зовнішньої пам'яті 24AA512 у корпусі PDIP

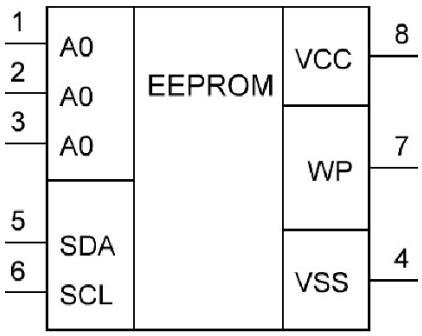


Рисунок 5.7 – Умовне позначення на принциповій схемі зовнішньої пам’яті

Таблиця 5.3 – Призначення виводів мікросхеми 24AA512

Назва контакту	Опис
A[0:2]	Адресний вибір чипу
SDA	Введення послідовних даних
SCL	Послідовний годинник
WP	Вхід з захистом від запису
GND	Земля
VCC	+1,7 В до 5,5 В

Мікросхема 24AA512 використовує входи A0, A1 і A2 для взаємодії з декількома пристроями. Ці входи визначають логічні рівні, які порівнюються з відповідними бітами у адресі пристрою. Чіп вибирається, якщо це порівняння виявиться істинним.

Одну шину можна використовувати для підключення до восьми пристроїв, використовуючи різні комбінації бітів. Зазвичай в програмах вхідні адресні піни мікросхеми A0, A1 і A2 фіксуються на логічний "0" або "1". Якщо цей пристрій керується мікроконтролером або іншим програмованим логічним пристроєм, контакти адреси мікросхеми мають бути установлені в логічне 0 або 1, відповідно до вимог програмного забезпечення.

Контакт SDA є двонапрямленим і використовується для передачі адрес та даних у пристрій або з нього. Важливо, що для зміни даних на SDA необхідний низький рівень на SCL. Зміни на SDA, які відбуваються при високому рівні SCL, використовуються для визначення умов запуску та зупинки.

Контакт SCL використовується для синхронізації передачі даних між пристроями.

WP-вхід може бути з'єднаний або з землею (VSS), або з живленням (VCC). Підключення до VSS дозволяє запис, а підключення до VCC блокує запис, хоча читання залишаються доступними [7].

Блок діаграма зовнішньої пам’яті наведена на рисунку 5.8.

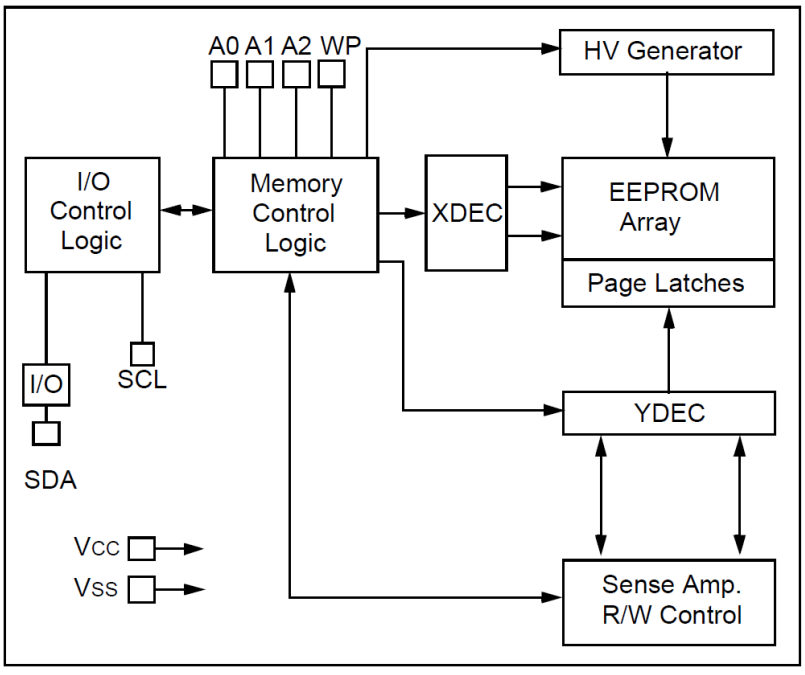


Рисунок 5.8 – Блок схема зовнішньої пам’яті 24AA512

5.4 Зовнішня пам’ять – 47C16-Mircochip Technology

Мікросхема пам’яті 47C16 представляє собою послідовну пам’ять EERAM 16 Кбіт та використовує послідовний інтерфейс I^2C. Данна пам’ять складається з SRAM 2K x 8 біт та EEPROM 2K x 8 біт для резервного копіювання.

Мікросхема 47C16 забезпечує нескінченні цикли читання та запису в SRAM, а комірки EEPROM забезпечують довговічне енергонезалежне зберігання даних. За допомогою зовнішнього конденсатора дані SRAM автоматично передаються в EEPROM після втрати живлення. Після ввімкнення живлення дані EEPROM автоматично повертаються до SRAM [8].

Мікросхема має 8 виводів, які зображені у корпусі PDIP на рисунку 5.9.

На рисунку 5.10 зображено умовне позначення на принциповій схемі пам’яті 47C16.

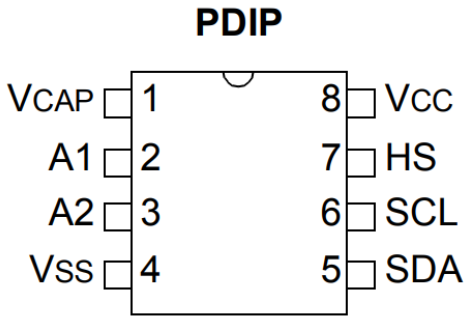


Рисунок 5.9 – Схема зовнішньої пам’яті 47C16 у корпусі PDIP

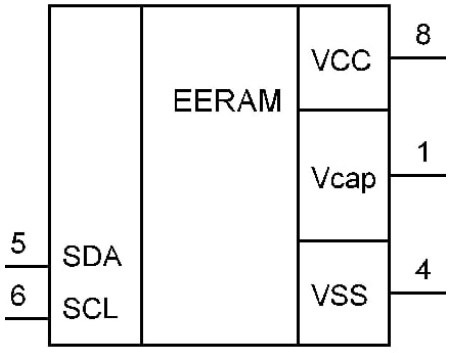


Рисунок 5.10 – Умовне позначення на принциповій схемі зовнішньої пам’яті 47C16

Таблиця 5.4 – Призначення виводів мікросхеми 47C16

Назва контакту	Опис
V_{CAP}	Вхід для конденсатору
A_1	Вхід вибору мікросхеми
A_2	Вхід вибору мікросхеми
V_{SS}	Земля
SDA	Послідовний вхід даних
SCL	Послідовний годинник
HS	Вхід виявлення події
V_{CC}	Живлення

Для використання функції Auto-Save вхід V_{CAP} необхідно підключити через конденсатор до землі. Конденсатор буде автоматично заряджатися через внутрішню шину живлення.

Контакти A_1 та A_2 використовуються мікросхемою 47C16 для операцій з декількома пристроями. Чіп обрається при порівнянні логічних рівнів на вході контактів та бітів адресу пристрою.

Контакт SDA – це двонаправлений контакт, який використовується для передачі адрес та даних до пристрою та з нього. Оскільки це контакт з відкритим стоком, тому для шини SDA потрібен підтягуючий резистор до VCC. Під час нормальної передачі даних SDA може бути змінена лише в той момент, коли SCL знаходиться в низькому стані. Зміни SDA під час високого стану SCL використовуються для вказівки умов запуску та зупинки передачі даних.

Контакт SCL використовується для синхронізації передачі даних із пристрою та до пристрою.

Контакт HS використовується для ініціювання роботи Hardware Store шляхом переведення на високий рівень на час T_{HSPW}. Якщо залишити не підключеним, цей вхід буде підключено внутрішньо до VSS [8].

Блок діаграма зовнішньої пам’яті наведена на рисунку 5.11.

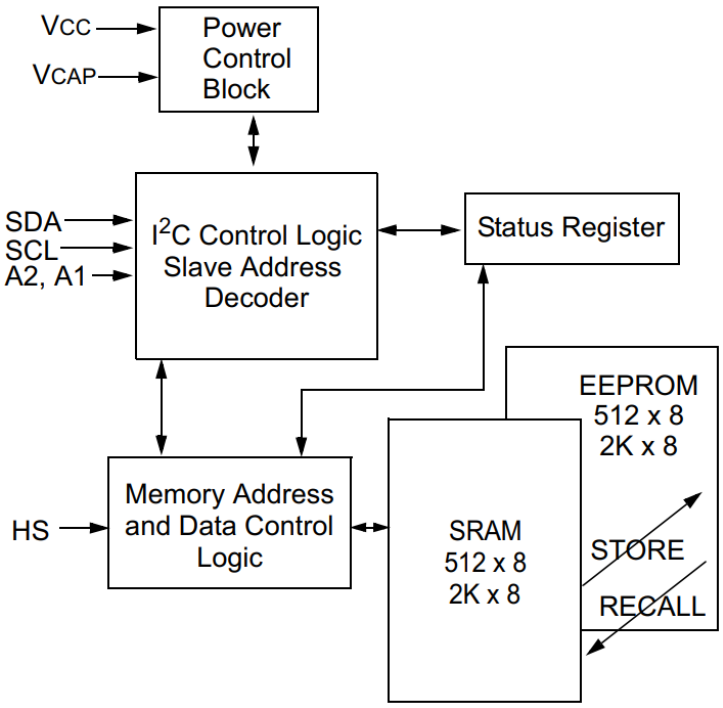


Рисунок 5.11 – Блок діаграма зовнішньої пам’яті 46C16

5.5 Демультиплексор – AT43312A-Atmel

Atmel AT43312A – це контролер USB-концентратора, що використовується як автономний концентратор або як додатковий концентратор для наявних периферійних пристроїв. [9].

Контролер AT43312A має один висхідний та чотири низхідних порти, які з'єднуються з висхіднім хабом або провідним/кореневим хабом через Port0. Решта портів приєднуються до зовнішніх USB-пристроїв.

Ця мікросхема передає диференційні USB-сигнали між портом Port0 та портами Port[1:4] у двох напрямках. Будучи доступним пристроєм, AT43312A вміє контролювати та відслідковувати живлення всіх портів одночасно або окремо для кожного.

Прилад має підтримку низької (1.5 Mb/s) та повної (12 Mb/s) швидкостей передачі даних. Його тактовий генератор працює на частоті 6 МГц, але внутрішні схеми функціонують на 48 МГц, що досягається за допомогою вбудованого PLL (фазової замкненості циклу).

У цьому пристрої є послідовний інтерфейс, повторювач хаба та контролер хаба. [9].

Завданнями механізму послідовного інтерфейсу є:

- управління протоколом USB комунікації;
- виявлення/генерація USB сигналу;
- поділ тактового сигналу/даних, шифрування/дешифрування даних, контроль/генерація CRC;
- послідовне/паралельне перетворення даних.

Повторювач хаба відповідає за:

- організацію висхідного з'єднання між вибраним пристроєм та ведучим;
- управління встановленням з'єднання та його припиненням;
- управління виявленням збоїв на шині та організацію відновлення правильних даних;
- виявлення з'єднань/роз'єднань по кожному порту.

Контролер хаба відповідає за:

- перерахунок з'єднань хаба;
- забезпечення провідного інформацією про конфігурацію;
- надання провідному інформації про стан кожного порту;
- управління кожним портом, відповідно до команд ведучого;
- управління живленням портів.

Блок діаграма мікросхеми демультиплексора/контролера хаба зображена на рисунку 5.12

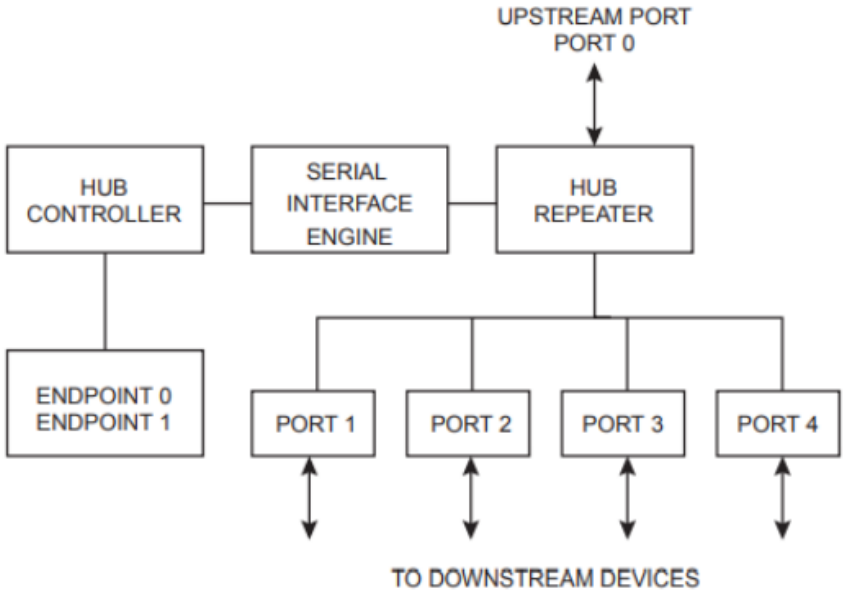


Рисунок 5.12 – Блок діаграма мікросхеми AT43312A

На рисунку 5.13 надано зображення контактів мікросхеми у корпусі SOIC та LQFP.

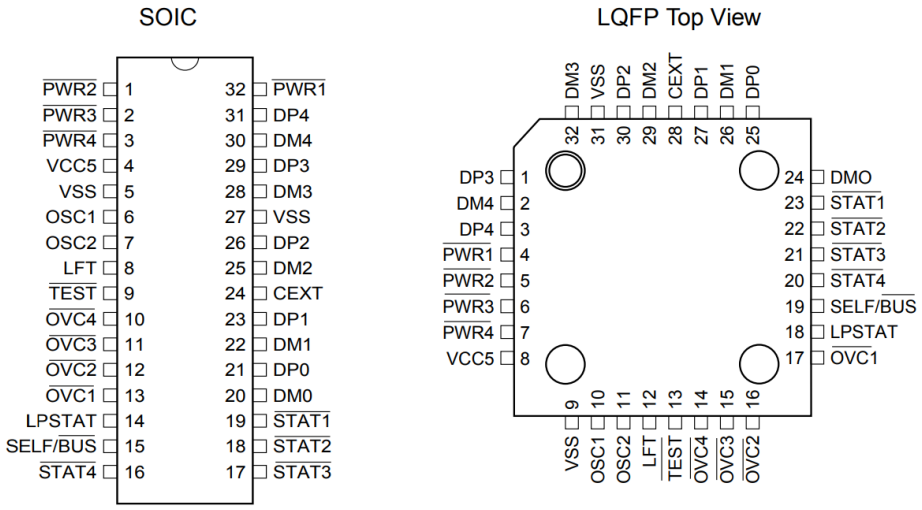


Рисунок 5.13 – Контакти мікросхема AT43312A у корпусі SOIC та LQFP

На рисунку 5.14 зображено умовне позначення на принциповій схемі контролер хабу AT43312A з портами, які використовуються.

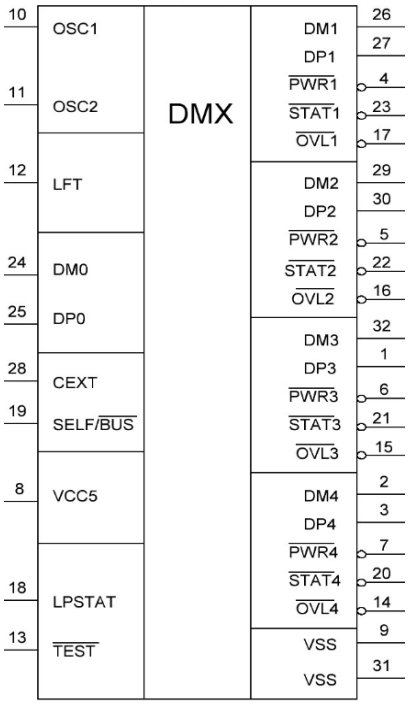


Рисунок 5.14 – Умовне позначення на принциповій схемі контролера хабу AT43312A

Таблиця 5.5 – Призначення контактів мікросхеми

Назва контакту	Тип контакту	Опис контакту
$\overline{\mathrm{PWR}}[1:4]$	Вихідний	Вимикач. Це вихідний сигнал, використовуваний для включення або виключення зовнішнього регулятора напруги, який подає живлення на порт. PWRx скидається при виявленні проблеми з живленням на OVCx.
$\overline{\mathrm{OVC}}[1:4]$	Вхідний	Перевантаження по струму. Це вхідний сигнал, що використовується для вказівки AT43312A, що на порту виявлено перевантаження по струму. Якщо встановлено значення OVCx, AT43312A активує виведення PWRx та повідомить про стан USB-хосту.
STAT[1:4]	Вихідний	Статус підключення. Це вихідний контакт, що вказує на те, що порт правильно підключений. STATx встановлюється, коли порт увімкнений.
OSC1	Вхідний	Вхід генератора. Вхід інвертуючого підсилювача генератора 6 МГц.
OSC2	Вихідний	Вихід генератора. Вихід інвертуючого підсилювача генератора.
LFT	Вхідний	PLL-фільтр. Для правильної роботи PLL цей вивід повинен бути підключений через конденсатор 2,2 нФ паралельно до резистора 100 Ом, який послідовно підключений з конденсатором 10 нФ на землю (VSS).
$\mathrm{SELF}/\overline{\mathrm{BUS}}$	Вхідний	Режим живлення концентратора. Вхідний сигнал, встановлюючий режим роботи шини або автономного живлення. Високий рівень на цьому контакті включає режим автономного живлення, низький рівень включає режим живлення від шини.
LPSTAT	Вхідний	Стан автономного живлення. У режимі автономного живлення цей вхідний контакт, який має бути підключений до локального джерела живлення через резистор 47 кОм. Напруга на цьому контакті використовується чіпом для повідомлення про стан автономного джерела живлення. У режимі живлення від шини цей контакт не використовується.

Продовження таблиці 5.5.

DP0	Двонаправлений	Висхідний плюс USB введення/виведення. Цей контакт повинен бути підключений до CEXT через зовнішній резистор 1,5 кОм. DP0 і DM0 утворюють пари контактів диференціального сигналу, підключені до хост-контролера або висхідного концентратора.
DM0	Двонаправлений	Висхідний мінус USB введення/виведення.
CEXT	Вихідний	Зовнішній конденсатор. Для правильної роботи вбудованого стабілізатора, до цього контакту необхідно підключити конденсатор ємністю 0,27 мкФ.
TEST	Вхідний	Тест. Для нормальної роботи, цей контакт має бути підключений до високого логічного рівня.
VCC	Живлення/Земля	Живлення 5В
VSS	Живлення/Земля	Земля

5.5.1 Функція управління енергоспоживанням

Концентратор є потужним пристроєм, який може споживати до 500 мА струму від хота або вищого концентратора. Однак сама мікросхема AT43312A та її зовнішній концентратор використовують значно менше струму, менше ніж 100 мА. Логіка керування живленням AT43312A співпрацює з зовнішніми пристроями для виявлення перевантаження по струму та управління живленням портів..

За допомогою контактів OVCx визначається перевантаження струмом для кожного порту окремо. Коли напруга на контакті OVCx перевищує норму, AT43312A ідентифікує це як стан перевантаження. Це може виникнути через перевантаження або коротке замикання. Відповідно, AT43312A активує біт стану порту PORT_OVER_CURRENT та біт зміни стану C_PORT_OVER_CURRENT. В той же час живлення на проблемному порту відключається, і його контакт STATx генерує прямокутний сигнал із частотою близько 1 секунди. [9].

Для контролю перевантаження струмом та виконання реального перемикання живлення портів, керованих AT43312A, потрібен зовнішній пристрій. Відповідати вимогам може будь-який відповідний перемикач або пристрій, проте важливо мати низьке падіння напруги, навіть при максимальному навантаженні. В основному випадку цей перемикач може бути реалізований за допомогою P-канального полевого МОП-транзистора. Використання такого MOSFET має перевагу в низькому падінні напруги та вигідності в ціні.

Кожен з портів у пристрої AT43312A має свій власний контроль живлення, який активується лише після отримання запиту SetPortFeature[PORT-POWER] від хосту. Порт PWRx скидається за таких умов:

1. Увімкнення
2. Скидання та ініціалізація
3. Стан перевантаження струмом
4. Запит від хосту через ClearPortFeature [PORT_POWER] для BCIX портів.

5.5.2 Режими живлення

У режимі автономного живлення нижні порти отримують живлення від зовнішнього джерела живлення. Це джерело має забезпечувати 500 мА для кожного окремого порту, а загалом — 2 А.

Специфікації USB стверджують, що падіння напруги на вимикачах живлення та трасах плати не повинно перевищувати 100 мВ. Максимальний допустимий рівень падіння напруги на самому вимикачі живлення має складати не більше 75 мВ. Щоб відповідати цим специфікаціям, необхідний ретельний проектування вибору вимикача живлення та структури плати. [9].

Приклад схеми автономного живлення портів з використанням мікросхеми AT43312A приведено на рисунку 5.15.

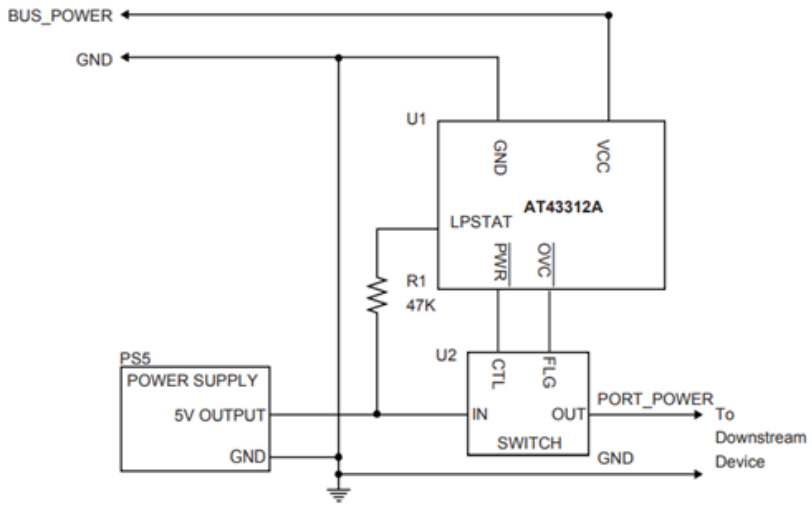


Рисунок 5.15 – Схема автономного живлення портів

У режимі живлення через шину, всю потужність для самого концентратора та його нижніх портів забезпечує кореневий або вищестоячий концентратор через USB. Кожному пристрою, який підключений до цього концентратора, доступно лише 100 мА, тому підтримуються лише пристрої з низьким енергоспоживанням. [9].

Приклад схеми живлення портів від шини з використанням мікросхеми AT43312A приведено на рисунку 5.16

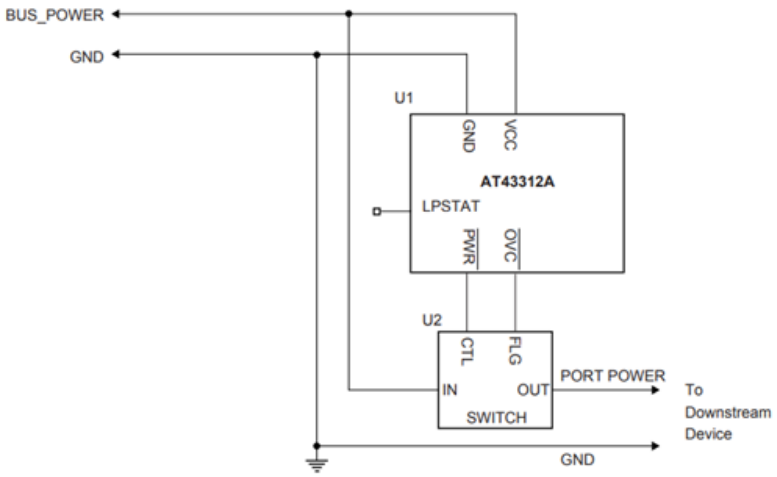


Рисунок 5.16 – Схема живлення портів від шини

5.5.3 AT43312A як HUB-контролер

AT43312A надає інформацію про свою конфігурацію хосту та може контролювати та керувати низхідними портами. Подача живлення на кожний порт контролером-концентратором відбувається після отримання команди SetPortFeature [PORT_POWER] від хосту. Проте перед цим контролер-концентратор повинен бути налаштований хостом, щоб мати можливість живити зовнішні пристрої [9].

Контролер-концентратор AT43312A включає в себе дві кінцеві точки, Endpoint0 та Endpoint1. Також присутній регістр стану контролера, який слугує відображенням поточних налаштувань пристрою. При початковому включенні живлення всі біти цього регістру автоматично встановлюються в нульовий стан.

5.5.4 Генератор та фільтр PLL контролера AT43312A

Система AT43312A отримує всі необхідні тактові сигнали від внутрішнього генератора. Щоб знизити електромагнітні перешкоди та мінімізувати розсіювання потужності в системі, цей генератор розрахований на операцію з кварцем частотою 6 МГц. Вбудований PLL (ФАПЧ) створює високочастотний сигнал для поділу тактових імпульсів/даних у послідовному інтерфейсі. У режимі пасивного очікування схема генератора вимикається. Щоб забезпечити швидкий старт, рекомендується використовувати кристал з високою добротністю.

На рисунку 5.17 приведено схему підключення портів генератора та фільтру.

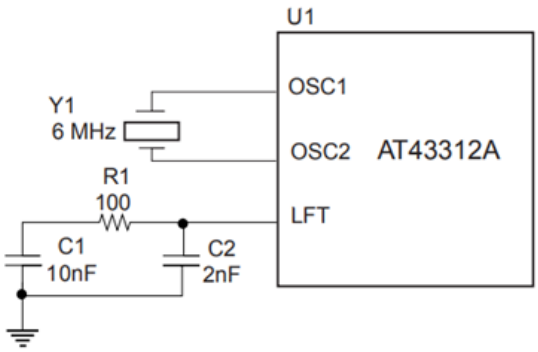


Рисунок 5.17 – Схема підключення портів генератора та фільтру

5.6 Ключ – MIC2526-2BM-Micrel

MIC2526 – це подвійний вбудований перемикач живлення верхнього плеча, який має дві незалежні функції включення та прапори. Він спеціально оптимізований для використання в універсальній послідовній шині (USB) як для автономного живлення, так і для живлення через шину. Для відповідності вимогам USB, його функціонування вимагає використання кількох зовнішніх компонентів. [10].

MIC2526 відповідає наступним вимогам USB:

- кожен канал перемикача забезпечує струм до 500 мА відповідно до вимог USB-пристрої;
- низький опір перемикача у включеному стані відповідає вимогам USB щодо падіння напруги;

- струм короткого замикання зазвичай обмежений 750 мА, що значно нижче за вимоги безпеки UL 25VA;
- вихід прапора доступний для визначення умов несправності на локальному USB-контролері.

Додаткові функції включають відключення при перегріві для запобігання можливій критичній відмові перемикача у випадку надмірного теплового навантаження, а також функцію блокування при зниженні напруги (UVLO), яка гарантує, що пристрій залишиться вимкненим до того часу, поки не буде подано відповідну вхідну напругу. Крім того, входи включення сумісні з логікою 3,3 В і 5 В, що забезпечує сумісність з різними рівнями логічної напруги.

На рисунку 5.18 зображено схему портів перемикача MIC2526-2BM у корпусі SOIC.

На рисунку 5.19 зображено умовне позначення на принциповій схемі мікросхеми ключа MIC2526-2BM

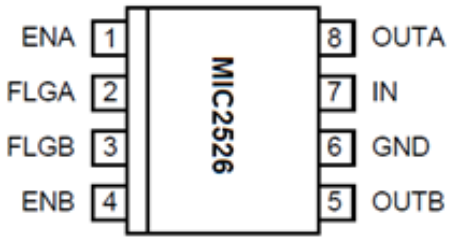


Рисунок 5.18 – Схема портів перемикача MIC2526-2BM у корпусі SOIC

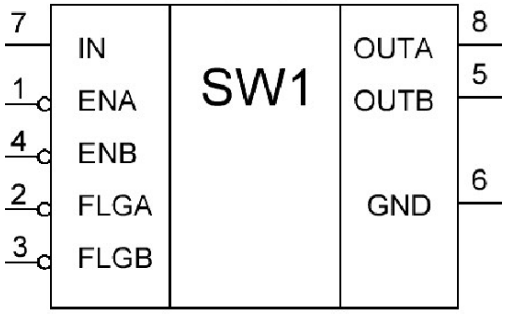


Рисунок 5.19 – Умовне позначення на принциповій схемі мікросхеми ключа MIC2526-2BM

Таблиця 5.6 – Конфігурація портів MIC2526-2BM

Номер порту	Назва порту	Опис портів
1, 4	EN(A/B)	Дозвіл (вхід): логічно-сумісний вхід дозволу. Низький вхідний сигнал.
2, 3	FLG(A/B)	Прапор несправності (вихід): вихід із активним низьким рівнем, відкритий стік. Вказує перевантаження по струму, блокування при зниженій напрузі та теплове відключення.
6	GND	Земля
7	IN	Вхід живлення: стік вихідного МОП-транзистора. Також живить внутрішню схему IC. Підключається до джерела живлення.
8, 5	OUT(A/B)	Вихід перемикача: вихідне джерело MOSFET. Зазвичай підключається до сторони навантаження, що комутується.

Мікросхема MIC2526-2BM це подвійний перемикач верхнього рівня зі входом активації "активний низький". У випадку виявлення несправності, вона відключає або фіксує увімкнення одного або кількох вихідних транзисторів залежно від характеру несправності. Такі несправності активують прапори несправностей транзисторів з відкритим стоком, що вимушує їх виводити струм до землі [10].

Блок діаграма перемикача MIC2527-2BM наведено на рисунку 5.20

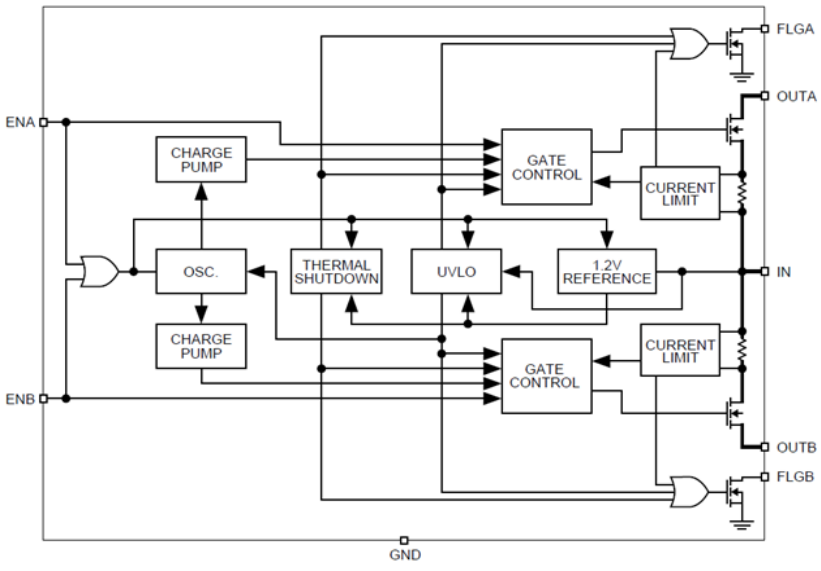


Рисунок 5.21 – Блок схема перемикача MIC2526-2BM

5.7 Вибір типу резисторів

При відборі резисторів важливо враховувати такі критерії:

- номінальний опір;
- номінальний допуск;
- розсіювана потужність;
- тип резистора;
- робоча температура.

Зважаючи на вимоги, обираємо безкорпусні SMD резистори типу 0805 з розсіюваною потужністю 0,125 Вт, номінальним допуском ±5%, та робочим температурним діапазоном від -55 °C до +125 °C. Діапазон номінальних значень для цих резисторів складає від 0 Ом до 30 МОм, що дозволяє підібрати опір за різних умов використання.

5.8 Вибір типу конденсаторів

Обираючи конденсатори, потрібно звертати увагу на кілька параметрів:

– Номінальний ємність конденсатора: Визначаємо потрібний обсяг зберігання заряду.

– Робоча напруга конденсатора: Має бути достатньою для безпечної роботи в вашому пристрої.

– Тип конденсатора: Керамічні та електролітичні конденсатори є популярними. Вибір залежить від вимог до ємності, робочої напруги та температурних умов.

– Номінальний допуск: Похибка в ємності, яку допускається прийняти.

– Робоча температура: Важливий параметр, оскільки пристрої можуть працювати в різних умовах температури.

Керамічні конденсатори типу SMD 0805, 1206 напругою від 10 В та з температурним коефіцієнтом X7R (-55 °C… +125 °C, похибка ±10%) – гарний вибір для надійної роботи в широкому температурному діапазоні.

Для електролітичних SMD конденсаторів потрібно вибирати напругу у два-три рази вище, ніж напруга живлення пристрою (в даному випадку 5 В), і забезпечувати робочий температурний діапазон від -55 °C до +125 °C. Обидва типи конденсаторів мають свої властивості, тому важливо зрозуміти вимоги до вашого пристрою та вибрати конденсатори, які краще відповідають цим вимогам.

5.9 Вибір типу реле

Для даного пристрою обираємо LEG-5, 5VDC, SPDT, 10A/240VAC. Це реле можна віднести до категорії "мініатюрних" пристроїв, що відрізняються компактним дизайном та широким функціоналом. Його максимальний струм становить 10 А, а напруга котушки - 5 В. Такий тип реле забезпечує надійне виконання завдань і залишається досить ефективним завдяки своїм компактним розмірам і високій функціональності.

5.10 Вибір типу світлодіодів

Світлодіоди використовуються як індикатори для нижчих портів. Якщо один або кілька портів вийшли з ладу, наприклад, через коротке замикання, то світлодіоди будуть загорятися с частотою близькою до 1 секунди. Для цього можна використати стандартні вивідні світлодіоди розміром 5 мм. Робоча напруга від 3,2 до 3,8 В досягається за допомогою послідовного підключення резистора. Робочий струм світлодіоду становить 25 мА, а робоча температура знаходиться в межах від -40°C до +60°C.

5.11 Вибір типу зовнішнього джерела живлення

Блок живлення моделі HN-538, адаптер живлення AD/DC, відповідає вимогам Smart хабу, який потребує 5 В та 3 А. Ось його характеристики:

- Вхідна напруга: Підтримує вхідну напругу від 100 до 240 В змінного струму.
- Мінімальний струм вхідної напруги: Потребує мінімум 0,3 А вхідного струму.
- Частота змінного струму: Підтримує частоту змінного струму 50/60 Гц.
- Вихідна напруга: Забезпечує вихідну напругу 5 В постійного струму.
- Вихідний струм: Має потужність до 3 А.
- Вихід адаптера живлення: Має роз'єм USB типу А.

6 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРИСТРОЮ

6.1 Доступ мікроконтролера до пам’яті EEPROM на запис/читання

Для програмування пам'яті EEPROM у мікроконтролерах AVR не потрібно використовувати зовнішній пристрій. Кожен блок пам'яті EEPROM може бути програмований безпосередньо під час виконання користувацької програми.

Для цього використовуються три регістри пам'яті EEPROM: регістр адреси EEAR, регістр даних EEDR та регістр управління EECR. Регістри EEDR та EECR мають по 8 розрядів. Регістр EEAR вимагає 9 розрядів: він є 16-розрядним і складається з двох частин: EEARH (старший байт) та EEARL (молодший байт) [12].

6.2 Регістри EEAR/EEDR/EECR пам’яті EEPROM

Регістр адреси EEAR у пам'яті EEPROM складається з двох байтів і розташований в області вводу/виводу за адресами $1E (RAM:$3E – молодший байт) та $1F (RAM:$3F – старший байт). Ці байти можна зчитувати та записувати, і після сигналу скидання вони автоматично ініціалізуються нулями.

Для програмування або зчитування байту даних з пам'яті EEPROM потрібно записати відповідну адресу в регістр адреси EEAR.

Регістр адреси EEDR пам'яті EEPROM розташований за адресою $1D (RAM:$3D) у області вводу/виводу. Після сигналу скидання його також очищають до значення нуля.

Під час запису байта у пам'ять EEPROM, який планується програмувати, його дані завантажуються в регістр EEDR. Під час зчитування з пам'яті EEPROM дані відповідного байту записуються у регістр EEDR. Адреса в пам'яті EEPROM у цих випадках визначається змістом регістра EEAR.

Регістр адреси EEDR у пам'яті EEPROM розташований в області вводу/виводу за адресою $1C (RAM:$3C). Після скидання сигналу він автоматично ініціалізується нулями. У мікроконтролерах базової серії сімейства AVR доступні для читання та запису лише розряди 0...1 або 0...2 регістра EECR. Інші розряди цього регістра зарезервовані компанією Atmel та доступні лише для читання, завжди містять значення логічного нуля [12].

Під час запису в пам'ять EEPROM, байт, який потрібно програмувати, завантажується в регістр EEDR. Під час читання з пам'яті EEPROM вміст відповідного рядка EEPROM записується у регістр EEDR. Адреса в пам'яті EEPROM в обох випадках визначається змістом регістра EEAR.

6.3 Читання та запис в пам’яті EEPROM

Розряд EERE використовується для управління процесом читання. Після запису правильної адреси в регістр EEAR, процес читання може бути активований, встановивши розряд EERE в регістрі управління.

Коли розряд EERE закінчує свою роботу, апаратне забезпечення зчитує потрібний байт в регістр EEDR. Немає потреби знову опитувати розряд EERE після цього, оскільки зчитування відбувається лише протягом одного тактового циклу системної синхронізації.

Перед початком операції читання програма користувача повинна постійно перевіряти розряд EEWE і чекати, поки він не зміниться на логічний 0. Якщо під час програмування в пам'ять EEPROM буде здійснено запис нових адрес або даних, то процес програмування буде перерваний, і результат може бути непередбачуваним.

Розряд EEWE контролює процес запису в пам'ять EEPROM. Для того щоб записати байт в EEPROM, потрібно встановити розряд EEWE у значення логічної одиниці, якщо в регістрі EEAR знаходиться правильна адреса для пам'яті EEPROM, а в регістрі EEDR – байт даних, який має бути записаний. Щоб уникнути випадкового запису в пам'ять EEPROM, розряд EEWE може бути встановлений тільки тоді, коли розряд EEMWE також встановлений [12].

Для програмування пам'яті необхідно виконати наступні дії:

– Зачекати завершення процесу програмування пам'яті EEPROM (якщо він активований), тобто, доки розряд EEWE не повернеться до логічного значення 0.

– Записати нову адресу в регістр EEAR пам'яті EEPROM.

– Записати потрібний байт даних в регістр EEDR пам'яті EEPROM.

– Встановити розряд EEMWE у логічну одиницю.

– Протягом наступних чотирьох періодів системного такту після встановлення EEMWE в розряд EEWE має бути записана логічна одиниця. Це запустить процес програмування.

Після завершення циклу програмування, розряд EEWE автоматично скидається в логічний нуль. Програма користувача повинна безперервно перевіряти цей розряд, очікуючи його перехід у логічний нуль, перед тим як продовжити програмування наступного байта.

Для запису одного байта в пам'ять EEPROM також необхідно встановити логічну одиницю в розряд EEMWE. Після встановлення розряду EEMWE, рівень логічної одиниці у ньому зберігається протягом чотирьох періодів такту системної синхронізації, після чого відбувається автоматичне скидання в логічний нуль. Програма користувача може розпочати програмування байта шляхом запису логічної 1 в розряд EEWE лише протягом цих чотирьох тактів системної синхронізації. Якщо розряд EEWE встановлено, але розряд EEMWE не активовано, процес програмування не розпочнеться.

6.4 Лістинг процесу програмування та читання пам’яті EEPROM

```
.include "353def.inc"
.equ AdrWr = $100          ; адреса, по якій буде виконане програмування
.equ AdrRd = $101          ; адреса, по якій повинно бути виконане читання
.def EEdrd = r0            ; байт, зчитаний із пам’яті EEPROM
.def EEdwr = r16           ; байт, який підлягає програмуванню в пам’ять
.def Temp = r17            ; допоміжний регістр

RESET:
rjmp Initial               ; перехід до частини ініціалізації

EEWrite:                   ; **** Підпрограма «Запис в EEPROM»
sbic EECR, EEWE            ; якщо EEWE не логічний 0, то
rjmp EEWrite               ; чекати далі
ldi Temp, High(AdrWr)      ; старший байт адреси запису в EEPROM –
out EEARH, Temp            ; в регістр адреси
ldi Temp, Low(AdrWr)       ; молодший байт адреси запису в EEPROM –
out EEARL, Temp            ; в регістр адреси

out EEDR, EEdwr            ; байт даних – в регістр даних
sbi EECR, EEMWE            ; розряд EEMWE дозволяє програмування
```

```
sbi EECR, EEWE          ; розряд EEWE встановлений: початок програм-ня
                        ; команда виконується протягом 4-х тактів,
                        ; оскільки затримка МК складає 2 такта
ret

EERead:                 ; **** Підпрограма «Читання EEPROM»
sbic EECR, EEWE         ; якщо EEWE не логічний нуль, то
rjmp EERead             ; чекати далі
ldi Temp, High(AdrRd)   ; старший байт адреси читання EEPROM –
out EEARH, Temp         ; в регістр адреси
ldi Temp, Low(AdrRd)    ; молодший байт адреси читання EEPROM –
out EEARL, Temp         ; в регістр адреси
out EEDR, EEdwr         ; байт даних – в регістр даних
sbi EECR, EERE          ; встановити розряд EERE – почати
                        ; процес читання. Команда виконується 4
                        ; такти, оскільки МК затримується на 2 такти.
in EEdrd, EEDR          ; читання байта даних
ret

Initial:
ldi Temp, Low(RAMEND)
out SPL, Temp
ldi Temp, High(RAMEND)
out SPH, Temp           ; встановити покажчик стека
ldi Temp, $ff           ; напрям передачі даних – вивід
out DDRA, Temp          ; в регістр напряму передачі даних
ldi Temp, $00           ; напрям передачі даних – введення
out DDRC, Temp          ; в регістр напряму передачі даних
out PortC, Temp         ; обрати вхід з трьома станами
in EEdwr, PinC          ; завантажити байт даних із порту С
rcall EEWrite           ; програмуємо байт за адресою #100
rcall EERead            ; читання байту даних за адресою $101
out PortA, EEdrd        ; зчитаний байт передати в порт А
```

Ця програма використовує підпрограми EEWrite та EERead для програмування та зчитування даних у пам'ять EEPROM. Основна програма ініціалізує потрібні порти та зчитує байт з одного з портів, який потім записується в пам'ять EEPROM за адресою $100. Після цього програма використовує підпрограму EERead для зчитування даних з пам'яті EEPROM за адресою $101 у робочий регістр EEdrd. Останній зчитаний байт потім виводиться через порт A.

7 ТЕХНІКО-ЕКОНОМІЧНА ЧАСТИНА

7.1 Розрахунок собівартості Smart хабу

Собівартість продукту означає грошові витрати, які підприємство понесло на виробництво та продаж продукту. Витрати, пов'язані з виробництвом, формують виробничу (заводську) собівартість, тоді як витрати на виробництво і збут утворюють повну собівартість. Розрахунок собівартості продукту за окремими статтями витрат називається калькуляцією.

Витрати, пов'язані з виробництвом та реалізацією продукту "Smart хаб", можуть бути розділені на наступні статті витрат:

- матеріали й комплектуючі;
- основна заробітна плата;
- додаткова заробітна плата;
- соціальні відрахування від заробітної плати;
- оренда машинного часу або видатки на утримання й експлуатацію встаткування;
- загальвиробничі видатки;
- адміністративні видатки;
- видатки на збут (реалізацію) продукту.

Видатки на матеріали й комплектуючі.

Розгляд матеріалів та комплектуючих виробів базується на інформації про складові матеріали, сировину, необхідну для збирання, а також на витратах на виробництво, що розподіляються на кожну окрему виготовлену одиницю (таблиця 7.1).

З урахуванням транспортно-заготівельних витрат ($k_{т-з}$=5÷15%) вартість комплектуючих та матеріалів складе:

$$КМ=(К+М)\cdot(100+k_{т-з})/100 \quad (7.1)$$

$$КМ=(609,32+319,94)\cdot(100+10)/100=1022,2 \text{ грн}$$

Таблиця 7.1 – Видатки на матеріали й комплектуючі вироби

Найменування комплектуючих	Ціна, грн.	Кільк., шт.	Сумарно, грн.
Мікросхема AT43USB353M	76,00	1	76,00
Мікросхема MIC2526-2BM	38,00	2	76,00
Мікросхема 47C16	11,40	1	11,40
Мікросхема 24AA512	15,20	1	15,20
Мікросхема AT43312A	38,00	1	38,00
Конденсатор SMD 0805 X7R-10В-10 мкФ	1,00	1	1,00
Конденсатор SMD 35В-4,7 мкФ (А)	4,00	2	8,00
Конденсатор SMD 0805 X7R-50В-0,01 мкФ	0,35	3	1,05
Конденсатор SMD 0805 X7R-25В-0,1 мкФ	2,03	2	4,06
Конденсатор SMD 0805 X5R-16В-2,2 мкФ	4,00	2	8,00
Конденсатор SMD 1206 X5R-16В-6,8 мкФ	4,00	2	8,00
Конденсатор SMD 16В-220 мкФ (Е)	3,60	4	14,40
Конденсатор SMD 0805 X7R-50В-100 нФ	0,59	1	0,59
Конденсатор SMD 0805 NPO-50В-47 пФ	0,50	10	5,00
Конденсатор SMD 0805 X7R-50В-2,2 нФ	4,00	1	4,00
Конденсатор SMD 0805 X7R-50В-330 нФ	3,05	1	3,05
Резистор SMD 0805 10 кОм 0,125 Вт	1,00	1	1,00
Резистор SMD 0805 100 кОм 0,125 Вт	1,00	4	4,00
Резистор SMD 0805 100 Ом 0,125 Вт	1,00	2	2,00
Резистор SMD 0805 1,5 кОм 0,125 Вт	1,00	2	2,00
Резистор SMD 0805 2 кОм 0,125 Вт	1,00	2	2,00
Резистор SMD 0805 15 кОм 0,125 Вт	1,00	12	12,00
Резистор SMD 0805 22 Ом 0,125 Вт	1,00	10	10,00
Резистор SMD 0805 47 кОм 0,125 Вт	1,00	1	1,00
Резистор SMD 0805 330 Ом 0,125 Вт	1,00	4	4,00
Реле LEG-5, 5VDC, SPDT, 10A/240VAC	34,00	1	34,00
Роз’єм USB-2AF10-5	25,00	5	125,00
Світлодіод FYL-5013 URC/U 8-10cd	6,40	4	25,60
Джерело живлення HN-538, 5 В 3 А	112,97	1	112,97
		Ціна комплектуючих:	609,32
Провід монтажний	20,00	0,3	6,00
Склотекстоліт	175,00	0,4	70,00
Флюс	1350,00	0,1	135,00
Припій	100,00	0,1	10
Лак	298,00	0,03	8,94
Сировину для корпусу	300,00	0,3	90
		Ціна матеріалів:	319,94

Видатки на основну заробітну плату ($З_о$). Формула для розрахунку основної заробітної плати виглядає наступним чином:

$$З_о = \sum_{i=2}^{n} Тс_i \cdot Нч_i \quad (7.2)$$

де *Тс* = 107 – тарифна ставка робітника, що задіяний у виробництві устрою, грн./година;

Нч = 35 – витрачене робітником час на виробництво й налагодження пристрою;

n = 2 – кількість працівників задіяних у виробництві.

Таким чином, використовуючи вираження (7.2),

$$З_о = \sum_{i=2}^{n} 107 \cdot 35 = 2 \cdot 107 \cdot 35 \approx 7500 \text{ грн}$$

Додаткова заробітна плата. Витрати на додаткову заробітну плату зазвичай складають від 10% до 30% від основної заробітної плати:

$$З_д = З_о \frac{К_д\%}{100} \quad (7.3)$$

де $К_д$ – відсоток додаткової заробітної плати.

Приймаючи $К_д$ = 20%, по формулі (7.2) знаходимо

$$З_д = 7500 \cdot 0{,}2 = 1500 \text{ грн}$$

Соціальні відрахування від заробітної плати. Дані про відрахування охоплюють відрахування від суми основної та додаткової заробітної плати, здійснені відповідно до установлених ставок:

- на обов'язкове державне пенсійне страхування;
- на державне страхування від нещасних випадків;
- на обов'язкове державне соціальне страхування на випадок безробіття;
- у зв'язку з тимчасовою втратою працездатності й витратами, обумовленими народженням дитини і похоронами:

$$В_{св} = (З_о + З_д) \cdot \frac{38,52\%}{100}, \quad (7.4)$$

Підставляючи у вираження (7.4) значення $З_о$ й $З_д$,

$$В_{св} = (7500 + 1500) \cdot \frac{38,52\%}{100} = 3466,8 \text{ грн}$$

Видатки на утримання і експлуатацію встаткування. Традиційно прийнято, що обладнання перебуває на балансі підприємства. Витрати на експлуатацію та утримання обладнання (ВУЕ) становлять відсоток від вартості обладнання та дорівнюють певному відсотку від основної заробітної плати. Цей відсоток ВУЕ визначається за допомогою аналізу загальної собівартості продукту і зазвичай становить приблизно 120-150%:

$$ВУЕ = З_о \cdot \frac{ВУЕ\%}{100}, \quad (7.5)$$

або з обліком (7.4), приймаючи *ВУЕ* = 130%, одержуємо

$$ВУЕ = 7500 \cdot 1,3 = 9750 \text{ грн}$$

Загальвиробничі видатки. Загальні виробничі витрати (Вз) включають витрати, пов'язані з управлінням підрозділом (цехом), витрати на службові відрядження працівників цього підрозділу, амортизаційні відрахування від вартості основних фондів загального призначення і так далі. Вз визначаються з аналізу повної собівартості продукту та зазвичай становлять у середньому від 130% до 250% виробничих витрат. Вони розраховуються як відсоток загальних виробничих витрат від основної заробітної плати:

$$В_з = З_о \cdot \frac{В_з\%}{100}, \quad (7.6)$$

Отже, з обліком $В_з\%$ = 190% з вираження (7.6) можна визначити

$$В_з = 7500 \cdot 1,9 = 14250 \text{ грн}$$

Виробнича собівартість $C_в$ включає видатки на виробництво пристрою (таблиця 7.2):

$$C_в = 1022{,}20 + 7500 + 1500 + 3466{,}8 + 9750 + 14250 \approx 37489 \text{ грн.}$$

Адміністративні видатки. Адміністративні витрати включають у себе різні витрати, пов'язані з управлінням виробництвом, відрядженнями, охороною підприємства, підготовкою кадрів, транспортом працівників та оплатою різних послуг і витрат. Адміністративні витрати Ва визначаються з відомостей по аналізу повної собівартості продукту та можуть становити у середньому від 140% до 200% адміністративних витрат від основної зарплати:

$$\text{В}_\text{а} = \text{З}_\text{о} \cdot \frac{\text{В}_\text{а}\%}{100}, \tag{7.7}$$

Таким чином, приймаючи B_a = 160%, з виразу (7.7) слідує

$$\text{В}_\text{а} = 7500 \cdot 1{,}6 = 12000 \text{ грн}$$

Позавиробничі (комерційні) видатки $B_п$ включають видатки на рекламу й передпродажну підготовку продукту. Орієнтовно ці видатки визначаються в розмірі 5-10 % від виробничої собівартості:

$$\text{В}_n = C_\text{в} \cdot \frac{B_n\%}{100}, \tag{7.8}$$

Отже, приймаючи $B_п$ = 7% , з вираження (7.8) можна визначити

$$\text{В}_n = 37489 \cdot 0{,}07 = 2624{,}23 \text{ грн}$$

Повна собівартість С виробленого продукту обчислюється як

$$C = C_в + B_a + B_п,$$

тобто з урахуванням знайдених значень $C_в$ = 37489 грн., B_a = 12000 грн. і $B_п$ = 2624,23 грн. одержуємо

C = 37489 + 12000 + 2624,23 = 52113,23 грн.

Калькуляція собівартості продукту зведена в таблицю 7.2.

Таблиця 7.2 – Калькуляція собівартості продукту

Найменування статей калькуляції	Значення, грн.
1. Основна заробітна плата	7500
2. Додаткова заробітна плата	1500
3. Відрахування від заробітної плати	3466,8
4. Матеріали й комплектуючі вироби	1022,2
5. Видатки на утримання й експлуатацію встаткування	9750
6. Загальвиробничі видатки	14250
Виробнича собівартість:	37489
7. Адміністративні видатки	12000
8. Позавиробничі (комерційні) видатки	2624,23
Повна собівартість:	52113,23

7.2 Розрахунок ціни Smart хабу

У ринковій економіці існують різні методи ціноутворення: собівартість плюс прибуток, забезпечення фіксованого обсягу прибутку, залежно від рівня попиту . Розрахунок оптової ціни продукту провадимо за схемою "собівартість плюс прибуток":

$$Ц_{опт} = С + П, \quad (7.9)$$

де C – собівартість продукту;

$П$ – величина прибутку.

Прибуток $П$ визначається виходячи з нормативу (показника) рентабельності R виробництва продукції, установлюваного підприємством:

$$R = \frac{П}{C} \cdot 100\%, \quad (7.10)$$

Рентабельність *R* продукту береться в розмірі до 35%.

Тоді з формул (7.9) і (7.10) оптова ціна продукту визначається як

$$Ц_{опт} = С + \frac{R \cdot C}{100}, \quad (7.11)$$

а роздрібна ціна з урахуванням ПДВ, що становить 20%,

$$Ц_{роз} = 1{,}2 \cdot Ц_{опт}. \quad (7.12)$$

Застосовуючи вираження (7.11) і (7.12) з обліком R = 35%, знаходимо значення оптової ціни

$$Ц_{опт} = 52113{,}23 + \frac{35 \cdot 52113{,}23}{100} \approx 70352{,}86 \text{ грн}$$

і значення роздрібної ціни

$$Ц_{роз} = 1{,}2 \cdot 70352{,}86 = 84423{,}43 \text{ грн}$$

7.3 Висновки з техніко-економічної частини

Отже, узагальнена оцінка витрат на розробку системи і визначення приблизної ціни за формулою "витрати плюс прибуток" дає змогу оцінити економічну доцільність використання продукту. Переваги цього підходу полягають у його простоті та зрозумілості, як відшкодуванні витрат на виробництво і забезпеченні прибутковості. Однак недолік полягає в тому, що цей метод слабко враховує ринкові фактори, зокрема попит. Проте в перехідній економіці існують обставини, коли цей підхід виправданий: у відсутності конкуренції (монополії), при обмеженні рентабельності продукції з боку держави, виконанні одноразових замовлень або при створенні оригінальних товарів.

З урахуванням розрахованих параметрів, можна зробити висновок про те, що розробка і впровадження Smart хабу має техніко-економічних сенс. Саме застосування Smart хабу в автоматизованій електронній системі дає велику перевагу перед аналогами.

Також слід зазначити, що впровадження серійного виробництва може значно знизити вартість Smart хабу, оскільки більша частина загальної ціни пристрою припадає на заробітну плату та адміністративні видатки.

ВИСНОВКИ

Виконуючи кваліфікаційну роботу магістра було проведено аналіз літературних джерел, який визначив основний напрямок проектування.

Огляд літератури встановив завдання створення автоматизованої електронної системи збору даних із застосуванням Smart хабу та пристрою стиснення даних на базі адаптивного алгоритму Хаффмана.

Для вирішення завдання створення системи був проведений аналіз технологій бездротової передачі даних, досліджено їхню принципову сутність та основні функції. Це дало змогу прийняти рішення щодо апаратних та програмних рішень.

Розроблення пристрою стиснення даних передбачало вибір алгоритму стиснення, що потребувало дослідження характеристик різних алгоритмів та їх порівняння.

В результаті магістерської роботи були розроблені структурні схеми та алгоритми роботи для автоматизованої електронної системи збору даних, Smart хабу та пристрою стиснення. Було розроблено функціональну електричну схему для Smart хабу, де опис функцій кожного окремого блоку.

При створенні схеми електричної принципової було запропоновано технічне рішення по вибору елементної бази пристрою.

Також, у роботі була розглянута техніко-економічна частина, в якій розраховано виробничу та повну собівартості Smart хабу, а також встановлено оптову та роздрібну ціну пристрою.

СПИСОК ЛІТЕРАТУРИ

1. Писарець, А. В. Автоматизовані системи передачі показань від приладів обліку енергоносіїв. Частина 1 / Писарець А. В., Писарець Є. В. // Вісник КПІ. Серія Приладобудування : збірник наукових праць. – 2020. – Вип. 59(1). – С. 95–101. Доступно: https://ela.kpi.ua/handle/123456789/40751
2. Писарець А.В., Писарець Є.В. Автоматизовані системи передачі показань від приладів обліку енергоносіїв. Частина 2/дис./ Національний технічний університет України. Київський політехнічний інститут імені Ігоря Сікорського. Київ 2020 р. Доступно: http://visnykpb.kpi.ua/article/view/221452
3. Основи інформаційних технологій [Електронний ресурс] – Доступно: https://informatics6.webnode.com.ua/zanyattya-10/
4. Жураковський Б. Ю., Пархомей І. Р., Дружинін В. А. Обробка інформації в сенсорних мережах [Електронний ресурс] – Доступно: https://ela.kpi.ua/bitstream/123456789/34181/1/asau-2018-1_05.pdf
5. Електронні системи [Електронний ресурс] – Доступно: https://web.posibnyky.vntu.edu.ua/firen/6bilynskyj_elektronni_systemy/563.htm
6. Документація на AT43USB353M [Електронний ресурс] – Доступно: https://www.alldatasheet.com/datasheet-pdf/pdf/83715/ATMEL/AT43USB353M.html
7. Документація на 24AA512 [Електронний ресурс] – Доступно: https://www.alldatasheet.com/datasheet-pdf/pdf/101595/MICROCHIP/24AA512.html
8. Документація на AT43312A [Електронний ресурс] – Доступно: https://www.alldatasheet.com/datasheet-pdf/pdf/175063/ATMEL/AT43312A.html
9. Документація на 47C16 [Електронний ресурс] – Доступно: https://pdf1.alldatasheet.com/datasheet-pdf/view/1253513/MICROCHIP/47C16.html
10. Документація на MIC2526-2BM [Електронний ресурс] – Доступно: https://www.alldatasheet.com/datasheet-pdf/pdf/74549/MICREL/MIC2526-2BM.html
11. Introduction to Microcontroller Programming for Power Electronics Control Applications Coding with MATLAB® and Simulink® / Mattia Rossi, Nicola Toscani, Marco Mauri, Francesco Castelli Dezza // 2022. – 429 с.
12. The Introduction to Programmable Logic Controllers for Beginners / John Mulindi // 2020. – 81 с.

13. Нечипоренко О. В., Корпань Я. В. Системи збору даних та їх компактного представлення [Електронний ресурс] – Доступно: https://er.chdtu.edu.ua/bitstream/ChSTU/3515/1/Конспект%20лекций.pdf

14. Винар Я.Ю. Комплексний енергомоніторинг на ринку електричної енергії з використанням Smart Metering System/дис./Національний технічний університет України. Київський політехнічний інститут імені Ігоря Сікорського. Київ 2020 р.

15. Тисячний С.Г. Автоматизована система розширеного моніторингу та комерційного обліку електроенергії/дис./ Національний технічний університет України. Київський політехнічний інститут імені Ігоря Сікорського. Київ 2020 р

16. Кривоніс Ю.І. Застосування біноміальних чисел в адаптивних інформаційних системах / Бережна О.В., Борисенко О.А., Горішняк А.О., Кривоніс Ю.І., Юрченко В.І. // Фізика, електроніка, електротехніка (ФЕЕ-2022). Матеріали та програма науково-технічної конференції. – Суми: СумДу, 2022. – С.63

17. Кривоніс Ю.І. «Розумний» USB-хаб модемного пулу автоматизованої системи / Бережна О.В., Арбузов В.В., Кривоніс Ю.І., Кондратенко О.А., Васильєв В.Р // Фізика, електроніка, електротехніка (ФЕЕ-2023). Матеріали та програма науково-технічної конференції. – Суми: СумДу, 2023. – С.72

Додаток А

Застосування біноміальних чисел в адаптивних інформаційних системах

Борисенко О.А., *професор*; Бережна О.В., *доцент*;
Горішняк А.О., *аспірант*; Кривоніс Ю.І., *студент гр. ЕС-81*;
Юрченко В.І., *студент гр. ЕС-81*
Сумський державний університет, м. Суми, Україна

В даний час актуальним завданням є розвиток методів кодування повідомлень як засобу підвищення ефективності роботи інформаційних систем із нестаціонарними каналами зв'язку.

Забезпечення необхідної достовірності передачі повідомлень у таких системах сьогодні досягається засобами завадостійкого кодування при виявленні та виправленні помилок. Однак при цьому зменшується швидкість передачі інформації. Підвищити її можна зменшенням надлишковості завадостійких кодових комбінацій, які підлягають передачі по каналу зв'язку в той час, коли завади невеликі. Компроміс досягається при використанні завадостійких кодів, параметри яких мають можливість адаптуватись при зміні рівня завад. В якості таких кодів пропонується застосовувати біноміальні коди, які за своєю природою

дозволяють змінювати в залежності від потужності завад власну завадостійкість і відповідно власну довжину. В результаті застосування таких кодів з’являється можливість забезпечити більш високу швидкість передачі інформації при достатньому рівні її завадостійкості. Наприклад, якщо вночі рівень завад буде незначним, то довжина кодів зменшиться, а вдень, якщо рівень завад підвищиться, то довжина кодових комбінацій збільшиться.

Перевагами таких кодових комбінацій, сформованих на базі біноміальних чисел, є невелика складність процедур виявлення помилок відносно до інших адаптивних завадостійких кодів. Параметрами цих чисел є довжина n і контрольне число k. Крім того, на основі біноміальних чисел можливе створення спеціалізованих пристроїв кодування та декодування з функцією самоконтролю, що дозволить підвищити надійність приймально-передавальної апаратури.

Таким чином, запропоновані методи та алгоритми адаптивного біноміального кодування та декодування є достатньо ефективними при впровадженні інформаційних систем зі змінним рівнем завад у каналах зв’язку.

ДОДАТОК Б

«Розумний» USB-хаб модемного пулу автоматизованої системи

Бережна О.В. , доц.; Арбузов В.В. , директор;
Кривоніс Ю.І. , студент гр. ЕС.м-21; Кондратенко О.А. , студент
гр. ЕС-91; Васильєв В.Р. , студент гр. ЕС.м-11
Сумський державний університет, м. Суми, Україна
ТОВ «Енергосервісне підприємство «Преобразователь»», м. Суми, Україна

Актуальним завданням в питаннях розвитку автоматизованих систем комерційного обліку електроенергії (АСКОЕ) є забезпечення своєчасного зчитування даних з лічильників учасниками ринку електроенергії, які лишаються можливості закупівлі електроенергії за більш дешевими тарифами при несвоєчасності надання даних.

В сучасних АСКОЕ дані з лічильників, які підключені до підстанційних GSM/GPRS модемів, зчитуються сервером АСКОЕ за допомогою модемного пулу, до складу якого входять GSM модеми та USB-хаб, до якого вони приєднані. Нестабільність живлення модемного пулу, наявність збоїв в роботі модемів, автоматичне оновлення операційних систем та віртуальних машин на сервері призводить до відмов у читанні лічильників. Відновлення роботи здійснюється вручну оператором АСКОЕ з ризиками несвоєчасного зчитування даних.

Дослідження показали, що для автоматичного відновлення роботи модемного пулу без участі людини необхідно до стандартної функціональності USB-хабу додати такі функції, які виконує оператор АСКОЕ при усуненні збоїв, що робить роботу хабу «розумною». Наприклад, якщо в процесі самоконтролю працездатності пулу USB-хаб зареєстрував невдалі спроби з’єднання з підстанційними модемами, хаб здійснює програмний рестарт відповідного модему шляхом відправки відповідної AT-команди через USB-порт або, при відсутності результату, хаб здійснює «рестарт» модему шляхом тимчасового вимкнення живлення GSM модему. При відсутності результату USB-хаб повідомляє оператора АСКОЕ про відмову шляхом відправки СМС-повідомлення через працездатний GSM модем, а потік з’єднань автоматично переводиться до вільного модему у модемному пулі.

Автоматизація функцій Оператору АСКОЕ за рахунок «розумного» USB-хабу при виникненні різноманітних збоїв підвищує рівень своєчасності зчитування даних з лічильників в АСКОЕ