СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

Пояснювальна записка

до магістерської дипломної роботи

магістр
(освітньо-кваліфікаційний рівень)

на тему: IoT-система контролю рівня рідини в системах опалення тепличного господарства з віддаленим доступом і дистанційним налаштуванням сенсорів

Виконав: студент 2 курсу, групи ЕЛ-24дм
171 «Електроніка»
(шифр і назва спеціальності)
Тесля М.С.
(прізвище та ініціали)

Керівник Захожай О.І.
(прізвище та ініціали)

Рецензент Меняйленко О.С.
(прізвище та ініціали)

Київ – 2025 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування
Освітньо-кваліфікаційний рівень магістр
Спеціальність 171 «Електроніка»
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ д.т.н., проф. Захожай О.І.
(підпис)
« ____ » _____ 2025 р.

ЗАВДАННЯ

на магістерську дипломну роботу студенту
Тесля Михайло Сергійович
(прізвище, ім'я, по батькові)

1.Тема роботи: IoT-система контролю рівня рідини в системах опалення тепличного господарства з віддаленим доступом і дистанційним налаштуванням сенсорів
керівник роботи проф., д.т.н. Захожай Олег Ігоревич
(вчене звання, науковий ступінь, прізвище, ім'я, по батькові)
затверджені наказом університету від: « 28 » 11 2025 року № 241/17.03
2. Строк подання студентом роботи: 20 грудня 2025 р.
3. Вихідні дані до роботи: матеріали науково-дослідної практики, науково-методична література, дані інтернет-мережі
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- 4.1. Вступ
- 4.2. Аналіз стану проблеми
- 4.3. Дослідження методів і пристроїв для вимірювання рівня рідини у ємності
- 4.4. Розробка функціональної моделі системи
- 4.5. Проектування архітектури приладу
- 4.6. Проектування архітектури серверної та клієнтської частин системи
- 4.7. Визначення переваг застосування системи і перспективи її розвитку
- 4.8. Висновки
- 4.9. Перелік використаних джерел
- 4.10. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Додаток А. Перший функціональний рівень нотації IDEF0. Декомпозиція нульового блоку, Додаток Б. Другий функціональний рівень нотації IDEF0. Декомпозиція блоку А4, Додаток В. Нотація IDEF3, Додаток Г. Повна схема пристрою, Додаток Д. Алгоритм роботи пристрою, Додаток Е. Текст програми «LIQUID_SENSOR_INFO»

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 10.11.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Одержання завдання на виконання роботи	10.11.2025	*виконано*
2.	Укладання і погодження з керівником плану і етапів виконання роботи	11.11.2025	*виконано*
3.	Узагальнення даних літературних джерел	14.11.2025	*виконано*
4.	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху виконання завдання	17.11.2025	*виконано*
5.	Аналіз технічних засобів та існуючих систем	19.11.2025	*виконано*
6.	Реалізація практичної частини завдання	26.11.2025	*виконано*
7.	Укладання, оформлення та погодження пояснювальної записки з керівником	09.12.2025	*виконано*
8.	Надання пояснювальної записки на кафедру	17.12.2025	*виконано*
9.	Підготовка доповіді та презентації	18.12.2025	*виконано*

Студент _____ Тесля М.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Захожай О.І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дипломна робота: 96 стор., 5 табл., 28 рис., 27 джерел, 6 додатків.

Об’єкт дослідження — системи контролю рівня рідини в опалювальних системах теплиць.

Мета роботи — скоротити залучення персоналу теплиці для регулярних особистих перевірок розширювального бака опалювальної системи за рахунок вдосконалення системи моніторингу рівня рідини в розширювальних баках систем опалення теплиць, з можливістю під’єднання декількох типів сенсорів, обробки даних в реальному часі та відображення інформації на веб-інтерфейсі й у мобільному додатку.

Проаналізовано стан проблеми, цінність і необхідність розробки системи контролю рівня рідини в опалювальних системах теплиць. Виконано огляд існуючих на поточний час технічних рішень даної проблеми. Проведено дослідження методів і пристроїв для вимірювання рівня рідини у ємності, на основі чого знайдені оптимальні для даної сфери методи та сенсори. Розроблена функціональна модель системи. Спроектована архітектура приладу та системи дистанційного моніторингу. Проведено оцінку фінансових витрат на здійснення ручного контролю рівня води персоналом, та розраховано період за який дана система окупить себе, і наскільки скоротяться витрати тепличного господарства.

Прогнозовані припущення про розвиток об’єкта дослідження — створення можливості бездротового під’єднання сенсорів до основного приладу.

Галузь застосування — опалювальна система відкритого типу малих тепличних господарств, резервуари для поливу у сільському господарстві.

ТЕПЛИЧНЕ ГОСПОДАРСТВО, СИСТЕМА ОПАЛЕННЯ, РІВЕНЬ РІДИНИ, IOT-СИСТЕМА, КОНДУКТИВНИЙ СЕНСОР, УЛЬТРАЗВУКОВИЙ СЕНСОР, ТЕМПЕРАТУРНИЙ СЕНСОР, ARDUINO, WEB-ДОДАТОК, ANDROID ДОДАТОК

ABSTRACT

Master’s thesis: 96 pages, 5 tables, 28 figures, 27 sources, 6 appendices.

The object of the research is liquid level control systems in greenhouse heating systems.

The purpose of the work is to reduce the involvement of greenhouse personnel in regular, manual inspections of the heating system’s expansion tank by improving the liquid level monitoring system in the expansion tanks of greenhouse heating systems, with the capability of connecting several types of sensors, processing data in real time, and displaying information in a web interface and a mobile application.

The state of the problem, as well as the value and necessity of developing a liquid level control system for greenhouse heating systems, has been analyzed. A review of the currently existing technical solutions to this problem has been carried out. A study of methods and devices for measuring liquid level in a tank has been conducted, on the basis of which the methods and sensors optimal for this field were identified. A functional model of the system has been developed. The architecture of the device and the remote monitoring system as a whole has been designed. An assessment of the financial costs of manual water level inspection performed by personnel has been conducted, and the payback period of the proposed system has been calculated, along with the extent to which it will reduce greenhouse operating expenses.

Forecasts regarding the future development of the research subject include the implementation of wireless sensor connectivity to the main monitoring device.

The proposed system is intended for application in open-type heating systems of small-scale greenhouse facilities and agricultural irrigation reservoirs.

GREENHOUSE FACILITY, HEATING SYSTEM, LIQUID LEVEL, IOT SYSTEM, CONDUCTIVE SENSOR, ULTRASONIC SENSOR, TEMPERATURE SENSOR, ARDUINO, WEB APPLICATION, ANDROID APPLICATION.

ЗМІСТ

ВСТУП

Актуальність. Визначення рівня рідини, це дуже важлива і розповсюджена операція у сучасному світі. Вона використовується як у крупних промислових підприємствах, гідроелектростанціях, наукових лабораторіях, так і у звичайних вуличних душах і резервуарах для води на дачі.

Існує велика кількість категорій і підкатегорій методів і сенсорів для визначення рівня рідини. Для кожної з цілей, виходячи з умов використання, необхідної точності, а іноді і вартості, використовуються різні методи та сенсори. Наприклад, у наукових лабораторіях, де дослідники працюють з кислотами та іншими речовинами, які активно, негативно впливають на більшість сенсорів і знаходяться в спеціальних умовах розміщення, більшість сенсорів використати неможливо, і за цією причиною потрібно використовувати сенсори набагато дорожчі, точніші, але з високою резистентністю до даного впливу, або які використовують непрямі засоби вимірювання.

Так само і для непрофесійних цілей, на кшталт резервуара для води на дачі, немає сенсу використовувати складні методи і дорогі сенсори, та можна обійтися звичайним поплавковим або кондуктивним сенсором [21, с. 65].

Тепличні господарства є вузькою і відносно малою, але важливою сферою сільського господарства. В даних господарствах є потреба постійно підтримувати певний температурний режим для покращення умов розміщення рослин та інших культур, що прямо впливає на ефективність роботи теплиці.

Таким чином, тепличні господарства повинні майже постійно використовувати опалювальні системи. Для підтримання рівня рідини в опалювальних системах відкритого типу, використовується розширювальний бак, у який надходять залишки води, та з якого поглинаються необхідна частина води.

Більшість господарств повинна власноруч контролювати даний рівень рідини у баку, що впливає на ефективність робітників, для яких це є додаткова задача і зайвий час на її виконання, який можна було б потратити на інші операції господарства.

Однак, більшість варіантів що пропонується для рішення даної проблеми, є неоптимальними, дорогими або без дистанційного контролю.

Об’єкт дослідження: системи контролю рівня рідини в опалювальних системах теплиць.

Предмет дослідження: методи та технічні засоби вимірювання рівня рідин у розширювальному баку для системи опалення приміщення теплиці.

Мета роботи — скоротити залучення персоналу теплиці для регулярних особистих перевірок розширювального бака опалювальної системи за рахунок вдосконалення системи моніторингу рівня рідини в розширювальних баках систем опалення теплиць, з можливістю під’єднання декількох типів сенсорів, обробки даних в реальному часі та відображення інформації на веб-інтерфейсі й у мобільному додатку.

Задачі дослідження:

- розглянути важливість розробки системи моніторингу рівня рідини для опалювальної системи теплиці;

- дослідити існуючі рішення по контролю рівня рідини у розширювальних баках систем опалення відкритого типу у теплицях;

- дослідити методи та технічні засоби вимірювання рівня рідин та визначити найбільш оптимальні;

- розробити архітектуру системи дистанційного моніторингу рівня рідини розширювального баку системи опалення, що включає в себе сенсори, прилад, серверну частину, web-інтерфейс та додаток для системи Android;

- визначити переваги застосування даної системи у тепличному господарстві та перспективи майбутнього розвитку даної IoT-системи.

РОЗДІЛ 1. АНАЛІЗ СТАНУ ПРОБЛЕМИ

1.1. Аналіз цінності і необхідності розробки системи

Класична система опалення відкритого типу, якщо розглядати її спрощено для розуміння базового механізму дії, складається з: центральне підключення води, зациклений контур труб опалення, котел та розширювальний бак. Дану схему можна побачити на рисунку 1.1:

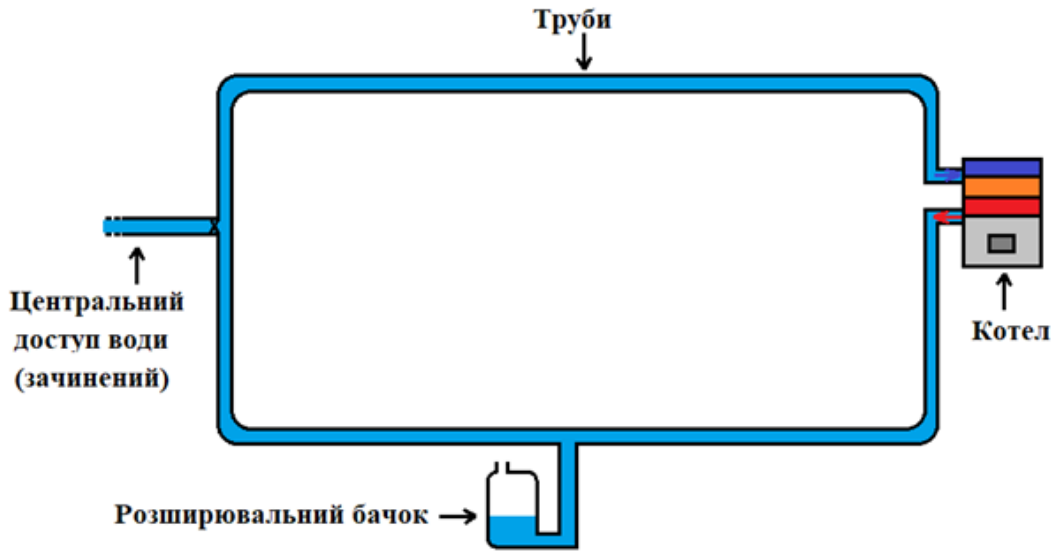


Рисунок 1.1 – Спрощена схема системи опалення відкритого типу

Вода при першому запуску надходить з центрального доступу води, наповнює труби, після чого даний доступ зачиняється. Коли котел вмикається, вода починає циркулювати по трубах і більш холодна входить до котлу, нагрівається і виходить вже тепла. Розширювальний бак з'єднаний із системою труб, тому надлишок води потрапляє саме у цей резервуар. Також, якщо вода з часом випарюється, за допомогою даного баку можна додавати воду, щоб компенсувати її витрати.

Велика кількість відказів котлів для опалювальної системи даного типу, може бути пов'язана з низьким або занадто високим рівнем води. Для безпечного функціонування котла, його необхідно використовувати із дотримуванням необхідних експлуатаційних вимог. В даному випадку – до рівня води. Низький рівень води може бути спричинений тим, що рівень води в барабані падає нижче

норми. При низькому рівні води може виникнути серйозне пошкодження котла, якщо котел продовжує працювати з низьким рівнем води. В умовах низького рівня води, топка котла та сталеві труби можуть розплавитись – це називається руйнуванням котла.

Високий рівень води збільшує швидкість виходу пару, розширювання води і як наслідок надмірне збільшення тиску. У результаті, опалювальна система не справляється, що може привести к вибуху котла, небезпечному гідравлічному удару та пошкодженню труб опалювальної системи або обладнання.

Система дистанційного моніторингу рівня рідини у розширювальному баку системи опалення надає можливість спостерігати віддалено за даним рівнем з будь якого місця, де є інтернет.

Виділимо основні проблеми і потреби з якими стикаються тепличні господарства, відповідно до даного дослідження:

- необхідність ручних перевірок – це операції затратні за часом та які спираються на суб’єктивну оцінку – що не є надійним інструментом визначення рівня рідини. Робітники тепличних господарств вимушені витрачати час на відвідування розширювальних баків та їх візуальну перевірку. Людський фактор може приводити до помилкових оцінок і аварій;

- пізнє виявлення несправності – частота додавання води, залежить від температури нагріву яка встановлена у котлу, і частіше усього необхідність у компенсуванні води йде через певний час. Проте, різке втрачання води, сильне випаровування тощо – це індикатор про аварію, порушення цілісності або інші несправності, які повинні бути вирішені якомога швидше. В іншому випадку це може привести до псування обладнання, приміщення, та втраті врожаю;

- неефективність роботи опалення – невірний рівень води, повна відсутність води, може привести до потрапляння повітря у опалювальну систему, що знижує енергоефективність, збільшує витрати на ремонт та нормалізацію роботи;

- низький бюджет – зазвичай, малі тепличні господарства мають обмежений бюджет і потребують дешевих, але надійних рішень автоматизації господарства – в даному випадку моніторингу рівня рідини.

Розроблювана система віддаленого контролю за рівнем рідини, допомагає уникнути деяких проблемних моментів наведених вище. Детально пояснимо що саме:

- усуває необхідність частих фізичних обходів, через те що один працівник може спостерігати за декількома об’єктами через веб-сайт або мобільний додаток. Це дозволяє виконувати відвідування резервуару лише за реальною необхідності, що приводе к ефективному перерозподілу робочої сили у більш продуктивну роботу;

- автоматичне попередження про великий або маленький рівень рідини у резервуарі запобігає описаним вище аваріям. Швидке сповіщення про незадовільний рівень рідини або про різку зміну рівня дозволяє вчасно вжити заходів, що дозволять мінімізувати витрати. Також це вносить свій вклад у запобігання затопленню обладнання та приміщення;

- підвищення надійності – надається можливість підключення двох типів сенсорів: ультразвуковий та кондуктивний, що дозволяє підвищити гнучкість використання системи моніторингу та проводити паралельну перевірку;

- забезпечення низької вартості та простоти реалізації – використовується найдешевший варіант Arduino з вбудованим Wi-Fi модулем, безкоштовних (до певного ліміту) / дешевих серверів Google Firebase, створених саме для малого бізнесу, оптимальні сенсори за ціна/використання для даних цілей;

- емуляція даної системи у віртуальному просторі прискорює розробку і знижує ризики при побудові реального пристрою.

Таким чином, тепличні господарства які використовують дану систему для автоматизації моніторингу отримують миттєві сповіщення на додаток у смартфону про критичні рівні рідини, зниження трудових затрат у працівників, зниження витрат самої теплиці, попередження крупних поломок, можливість масштабувати моніторинг серед декількох теплиць (один центр з розповсюдженням сенсорів на декілька теплиць).

1.2. Огляд та класифікація наявних систем для моніторингу рівня рідини

На даний час існує декілька готових технічних рішень даної проблеми. Загалом, їх можна поділити на такі категорії:

- індустріальні;
- системи без дистанційного контролю;
- вбудовані у систему котлу;
- байпасні системи;
- власні рішення.

Розберемо кожен з цих прикладів. Індустріальні системи, це серійні, сертифіковані системи, які включають у себе сенсори не тільки рівня, а й додаткові, на кшталт тиску, щільності, тощо. Також в них є комунікаційний шлюз, окремий сервер. У пакет послуг часто входять консультація, професійна інсталяція та калібрування. Переваги даної категорії:

- висока точність, яка підтверджена професійними сертифікатами систем;
- широкий спектр вимірювання показників;
- додаткове ПЗ, яке допомагає у створенні звітів, розрахунків тощо;
- розроблена система бездротової передачі даних.

Недоліками даної категорії є:

- висока вартість системи;
- часто є підписка для розблокування додаткового функціоналу;
- надлишковість функціональності відносно малих господарств.

Прикладами таких систем є радарний: Micropilot FMR30B від «Endress+Hauser» з вартістю від 1200$ [14], VEGAPULS від «Vega» з вартістю від 650$ [20].

Якщо порівнювати дані системи для рішення проблемою що досліджується, то можна сказати, що дане рішення економічно надлишково. воно надає дуже високий рівень точності і надійності, проте для малих господарств і для сфери що досліджується, даний рівень точності не потрібен.

Системи без дистанційного контролю розраховані на локальне використання та мають більш низьку вартість, за рахунок того, що не мають хмари та додаткового програмного забезпечення.

Перевагами такого рішення є:

- висока надійність – сонячні панелі для підзарядки, захист сенсорів, можливість генерування аварійних сигналів);

- більш низька вартість порівняно з індустріальними рішеннями;

- є сертифікація з електробезпеки. Виходячи з цього, є можливість використовувати у закритих приміщеннях.

Недоліками такого рішення є:

- відсутність хмари і можливості дистанційного моніторингу за рівнем рідини;

- нема можливості налаштувати гнучкість повідомлень;

- часто дані рішення використовуються для вузькопрофільних ланок, наприклад, для занурювальних двигунів і розроблюється комплексно.

Прикладом таких систем є: «FranklinWater» [7]. Проте дане рішення як було описано в мінусах вище, йде у комплекті із насосами, двигунами і іншими рішеннями. Використання насосної технології у даному випадку надлишково.

Наступна категорія, це вбудовані у котел сенсори та вивід на дисплей інформації. Дані сенсори вбудовані у котел або у барабан котла система, що відслідковує рівень води у барабані, тиск та паро-формування. Поточні дані відправляються на дисплей. Проте дана система реалізована не у всіх котлах, а переважно у дорогих та сучасних моделях. Також зауважимо, що дистанційне отримання результату на мобільний додаток або на веб-сайт, також неможливо через те, що дану можливість переважна більшість котлів не реалізує.

Переваги такого рішення:

- надійність;

- тривалий період життя;

Недоліки такого рішення:

- висока вартість;

- відсутність системи дистанційного контролю.

Прикладом даної категорії є котел TGB HiFin від «Kiturami» [25]. В котел вбудовані різноманітні сенсори та реалізований захист від можливих аварійних ситуацій: контроль тиску, температури, рівня рідини, тощо.

Окремо виділимо байпасні рішення, як доволі популярне і поширене. Під байпасним рішенням ми маємо на увазі повністю автономний байпасний індикатор. Байпасний індикатор рівня являє собою виносну байпасну колонку, яка кріпиться до резервуара або у нашому випадку бака опалювальної системи на бокову стінку наскрізь, двома приєднаннями: зверху і знизу. Таким чином, пристрій утворює з резервуаром сполучену конструкцію, тому рівні рідини в колонці і баку рівні. Це дозволяю візуально контролювати ступінь наповнення резервуару.

Переваги такого рішення:

- швидкий монтаж;

- можливість модернізацій;

- немає потреби у електроенергії.

Недоліки такого рішення:

- порушення цілісності резервуара;

- немає готового рішення для віддаленого контролю за рівнем рідини;

- чутливі до засмічення патрубків.

Остання категорія, це власні рішення. Для реалізації самостійних систем, часто використовують такі сенсори:

- гідростатичний сенсор. Сенсор занурюється на дно та видає товщину води над собою у міліметрах [21, с.59].

- ультразвуковий сенсор. Він закріплюється зверху, посилає сигнал до низу, і коли сигнал досягає води, він відбивається та прямує до сенсору. На основі цього, вимірюється рівень води.

У цій категорії кожне рішення відрізняється одне від одного, наявність системи дистанційного моніторингу також варіюються від бажання та знань розробника.

Переваги такого рішення:

- можливість повного контролю над системою та комплектуючими – архітектура системи, як і її покращення, цілком залежить від розробника;

- низька вартість;

- велика гнучкість у вибору засобів для реалізації.

Недоліки такого рішення:

- покращення системи та її працездатність цілком залежить від розробника;

- не доказана надійність – відсутність сертифікації;

- не доказана точність вимірювання і коректність даних що надходять.

Таким чином, існує велика кількість існуючих технічних рішень даної проблеми, проте більшість з них направлені на крупну промисловість або господарства, інші на вузьку спеціалізацію застосування, треті направлені на локальне застосування і не мають хмари, в четвертих присутня наявність надлишкового функціоналу який збільшує вартість систему, і як наслідок, більшість малих господарств прибігають до п’ятої частини – самостійно конструюють системи, що складаються лише з функціоналу що їм потрібен.

1.3. Дослідження існуючих схемотехнічних рішень

Розглянемо більш детально деякі існуючі рішення наведені вище. Існує широкий спектр схемотехнічних рішень даної проблеми, що розрізняються за принципом дії та складністю реалізації.

Аналіз існуючих у вільному доступі існуючих схемотехнічних рішень, що застосовуються на практиці, дозволяє краще зрозуміти принципи роботи різних підходів та створити основу для початку дослідження безпосередньо методів вимірювання рівня рідини. Проте, важливо також зазначити, що більшість комерційних рішень не поставляє розвернуті схеми своїх приладів і тому в даному підрозділі будемо орієнтуватися на приклади, що знаходяться у вільному доступі.

Розберемо схему простого оптичного сенсору рівня рідини (рис. 1.2):

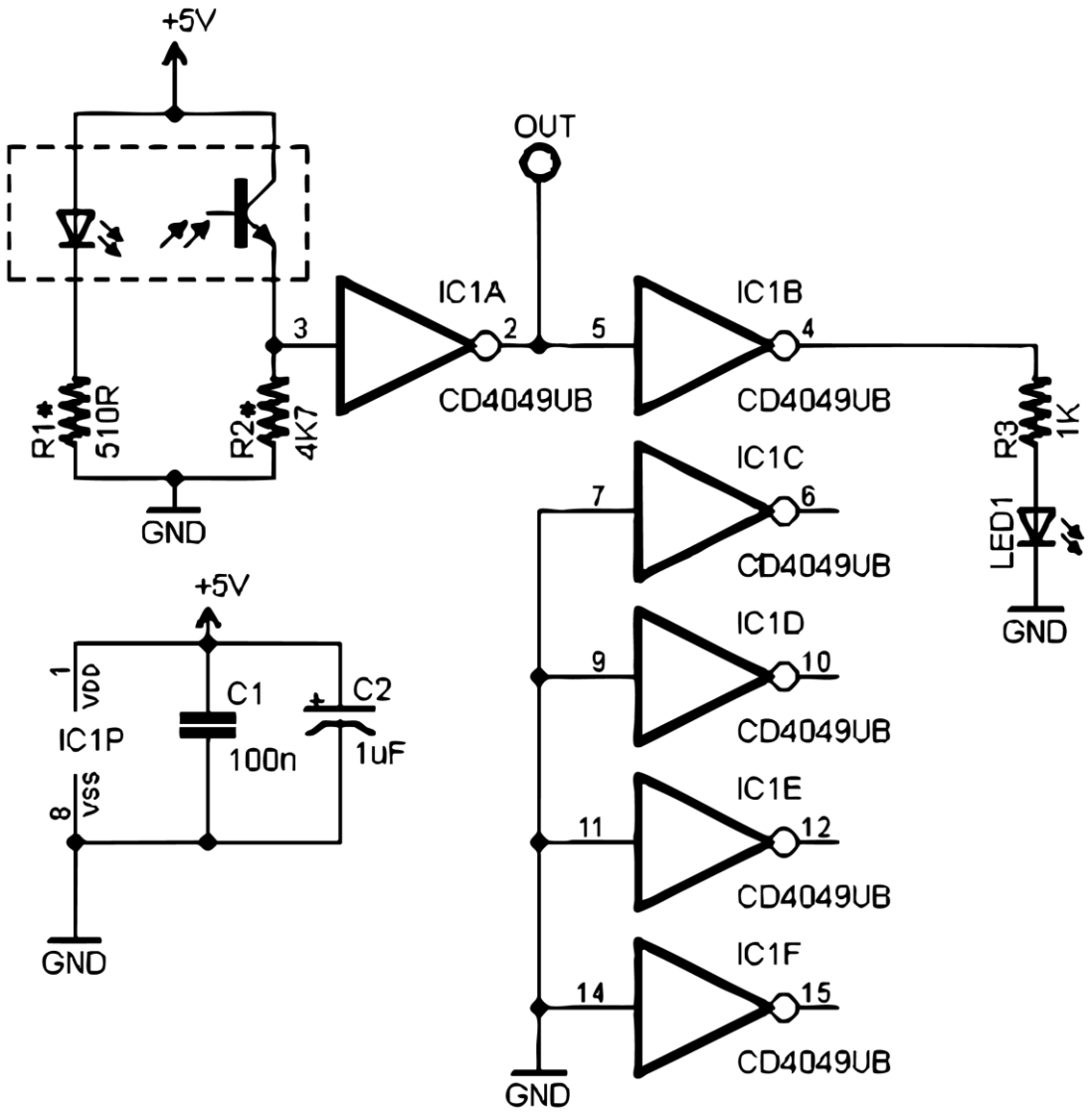


Рисунок 1.2 – Схема оптичного сенсору рівня рідини [15]

Потрібно одразу зазначити, що даний пристрій розроблено як двоканальний і зможе лише реагувати на те, чи є рідина, чи ні, тобто за моделлю «dry/wet». Також, даний пристрій побудований як недороге рішення без можливості дистанційного моніторингу, що прямо впливає на переваги, що отримує тепличне господарство. При досягненні певного рівня рідини («wet»), LED індикатор починає світити. Тобто, формується логічний рівень (0 або 1), який надходить на індикатор, в залежності від того, з’являється або зникає рідина біля сенсору.

Даний оптичний сенсор рівня рідини побудовано на інфрачервоному світлодіоді, фототранзисторі, логічних інверторів CD4049UB та LED-індикації [15].

Пристрій використовує принцип повного внутрішнього віддзеркалювання у пластині або корпусі сенсора. Тобто, вода зменшує світло, що потрапляє у

фототранзистор, в результаті чого, якщо сенсор не занурений у воду – світло відбивається назад до приймача і навпаки, якщо віддзеркалення немає, світло уходить у воду і фототранзистор отримує менше світла – сенсор занурений в воду. Це відбувається за рахунок того, що коли світла багато, то фототранзистор насичується і напруга на R2 падає. Коли світла мало, то фототранзистор закритий і напруга на R2 зростає [15].

Звернемо увагу на інвертор CD4049UB, зокрема на IC1A. Даний інвертор посилює і формує сигнал. Тобто, фототранзистор надає аналоговий сигнал, а інвертор трансформує з нього чіткий цифровий логічний нуль або одиницю.

Також використовується додатковий каскад інверторів. Це потрібно через те, що кожен інвертор CD4049 може віддавати дуже маленький струм. Каскад паралельно підключених інверторів збільшує вихідний струм, стабілізує логічний рівень та надає можливість безпечно включити світлодіод.

Даний прилад є лише оптичним сенсором з підключенням до LED-індикатора, проте за рахунок отримання кінцевого результату у вигляді логічного нуля, або одиниці, дуже легко доєднати його до повноцінної IoT-системи.

Розглянемо інший приклад – двоканальний сигналізатор вологості оснований на кондуктивному принципі (рис. 1.3.) або кондуктивний релейний датчик рівня води, який керує навантаженням 220В через оптосимістор та симістор. Даний прилад використовується у зв’язці з насосом для відкачки води, проте, даний елемент можливо замінити на інший виконавчий пристрій, підключити мікроконтролер та переписати логіку. Таким чином, замість активації насосу для відкачки води, можливо навпаки налаштувати автоматичне додавання води до резервуару.

Принцип роботи схеми полягає у наступному: база транзистора VT1 підключена через резистор R1, що обмежує струм, до першого електроду сенсора. Другий електрод, що розташований поряд з ним, приєднаний до шини живлення зі знаком плюс. Коли вода доходить до електродів сенсору, виникає електричний струм і відкриває транзистор VT1. Світлодіод HL1 підключений до ланцюга колектору починає світити. Струм колектору транзистора також проходить через

світлодіод оптрона мікросхеми DA1, та одночасно з цим вмикає водяний насос. Конденсатора C1 підключеній між базою та колектором транзистора в ланцюзі негативного зворотного зв'язку дозволяє запобігти помилкових активацій, що можуть статися від сторонніх змінних наведень [18].

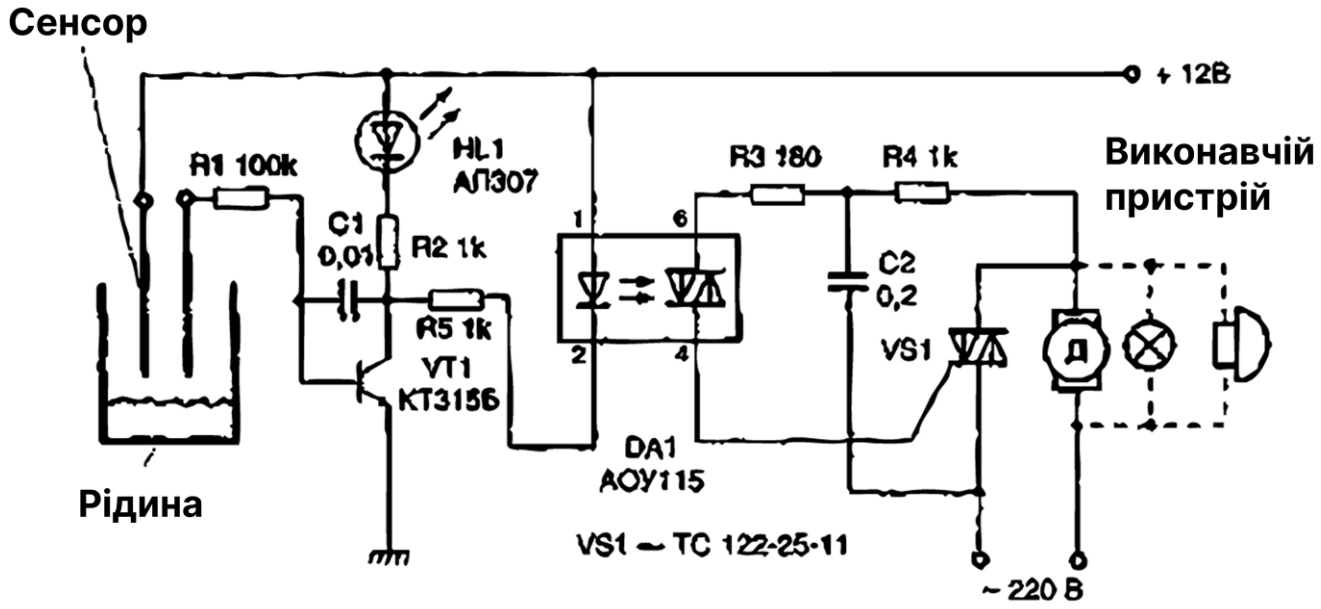


Рисунок 1.3 – Двоканальний кондуктивний релейний датчик рівня рідини [18]

Окремо звернемо увагу на роботу фотосимістору. Всередені нього знаходиться світлодіод, а напроти керуючий симістор. Коли транзистор VT1 відкривається, через його колектор і резистори струм протікає у світлодіод DA1, в результаті чого світлодіод починає сяяти і фотосимістор що знаходиться всередені замикається. На виході, DA1 запускає ланцюг керування силовим симістором VS1.

У свою чергу, симістор VS1 (TC122-25-11), це силовий симістор, що управляє навантаженням 220В. Оптосимістор DA1 послідовно підключений з керуючим електродом VS1. Щоб VS1 правильно відкривався, застосовується ланцюг, що включає в себе: R3 (180 Ω) – його задача обмежувати струму в керуючому електродів, R4 (1 кΩ) – його задача формувати ланцюг запуску, і C2 (0,2 μF) – його задача, це фільтрація перешкод та запобігання ненавмисному включенню [18].

Також необхідно зазначити, що через прямий контакт із водою, що необхідно для роботи сенсору, необхідно виготовляти електроди сенсору з нержавіючого та

неокислювального у воді металу. Це необхідно, щоб при тривалому використання уникнути збільшення опору. Проте, для коректної роботи, все одно необхідно час від часу очищати електроди від нальоту.

Для використання у тепличному господарстві, даний механізм керування насосом є надлишковим. Проте, для можливого майбутнього покращення системи (додавання до неї керуючих елементів), це можна використовувати як відправну точку.

Наступним прикладом є повноцінна IoT-система із застосуванням мікроконтролера Arduino Nano (або як заявляє розробник – Arduino Pro mini), модуля Wi-Fi ESP8266-01 на базі мікроконтролера ESP826, та ультразвукового сенсору HC-SR04 (рис. 1.4).

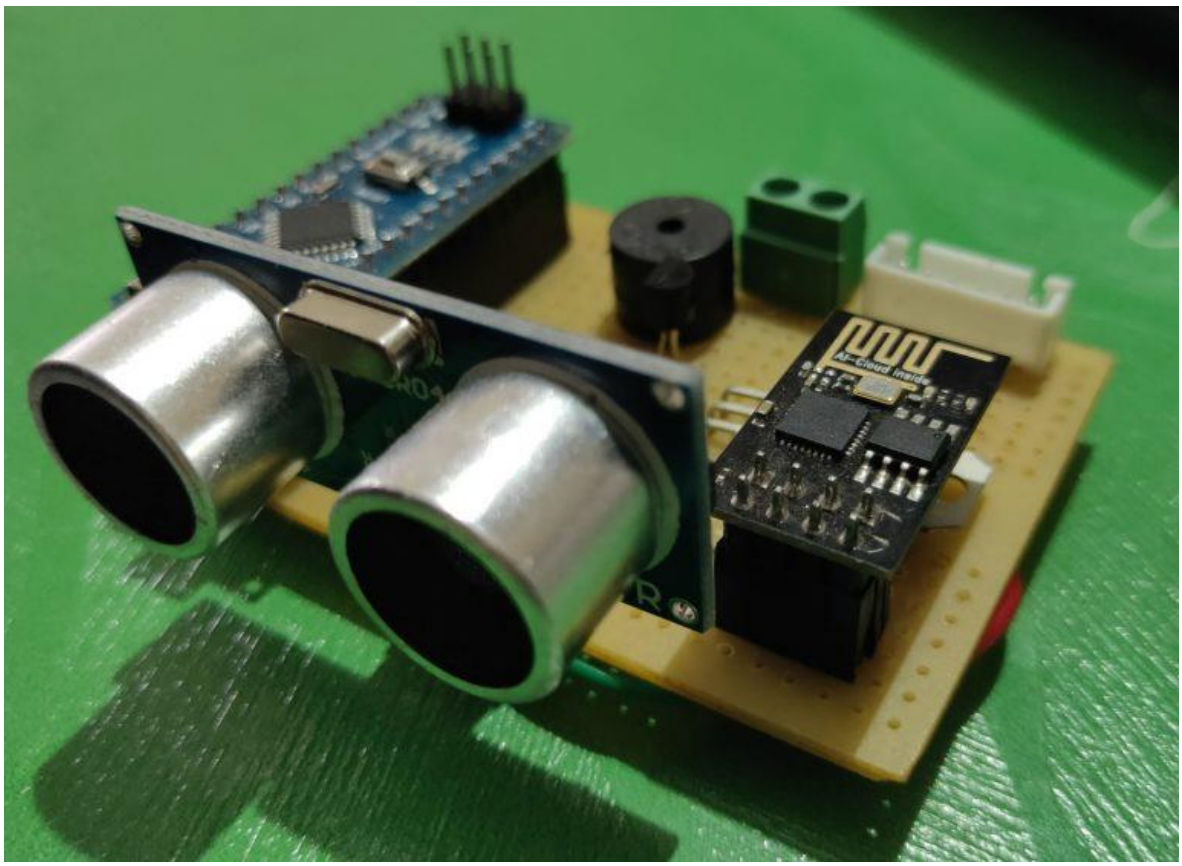


Рисунок 1.4 – Спрощена IoT-система для дистанційного моніторингу за рівнем рідини [9].

Дана IoT-система дозволяє вимірювати відстань від верхівки бака до поверхні рідини, локально відображати за допомогою світлодіодів рівень наповненості бака, та передавати дані до хмари Blynk через Wi-Fi модуль ESP8266-01 [9].

Blynk – це «low code» IoT-платформа, що дозволяє створювати мобільні та веб-додатки з можливістю підключення пристрою до однойменної хмари і керувати ними. Дана платформа пропонує рішення для малих команд і підприємств, для прискорення запуску IoT-продуктів. Для тестових запусків та налаштування пропонується безкоштовний режим користування платформою.

Принцип роботи: HC-SR04 отримує сигнал від Arduino, виконує відправку і отримання імпульсу, та надсилає до Arduino результат (тривалість часу що необхідно для імпульсу щоб дістатися до поверхні рідини і повернутися назад). Arduino відповідає за обробку інформації, розрахунок відсотку наповненості баку, контроль світлодіодів та взаємодію з сенсором та Wi-Fi модулем, через який дані відправляються до Blynk [9].

Наведемо принципову схему даного рішення (рис 1.5.) та зазначимо, що при з'єднанні TX Arduino з RX ESP необхідно зменшити дільник напруги, тому що Arduino TX = 5 V, а RX ESP = 3.3 V.

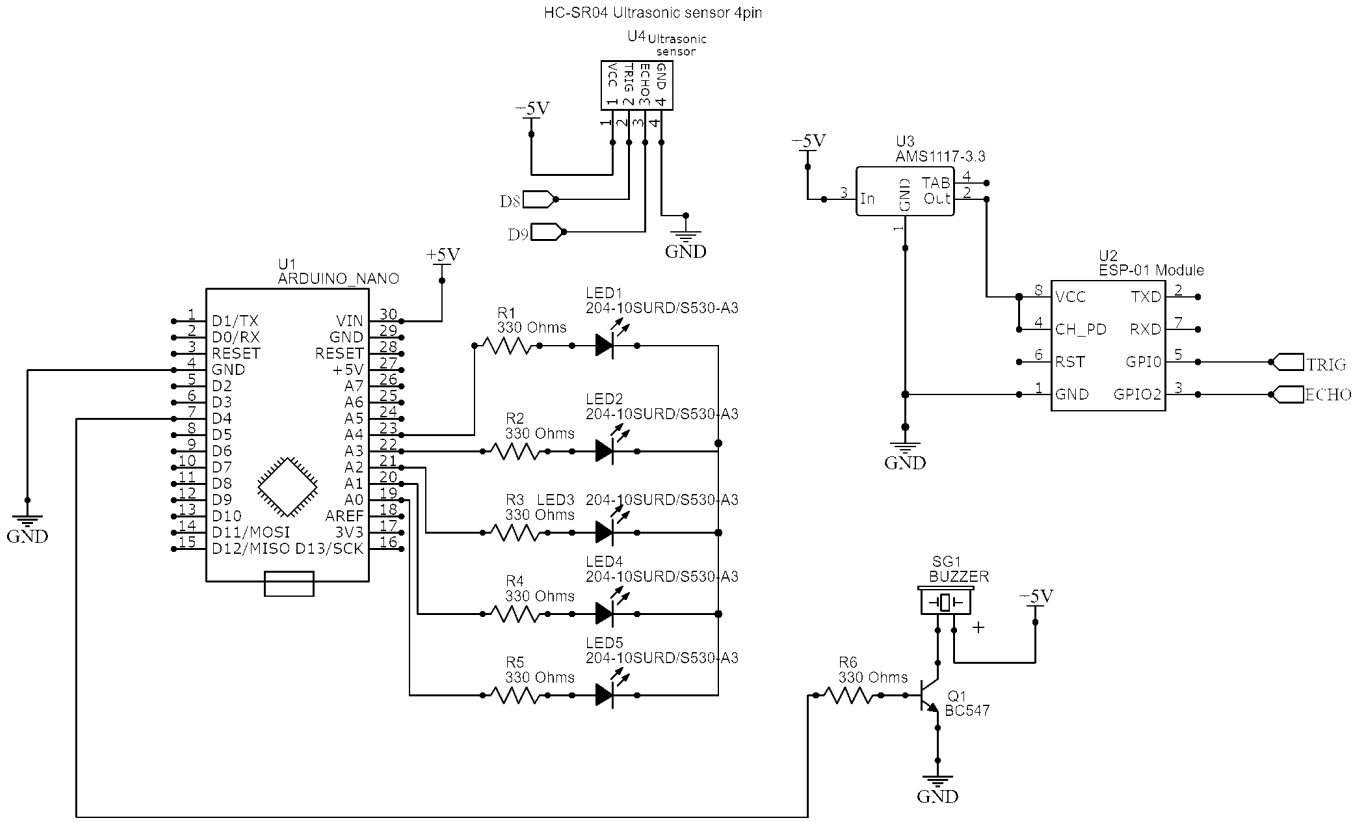


Рисунок 1.5 – Принципова схема спрощеної IoT-системи для дистанційного моніторингу за рівнем рідини [9]

Також зазначимо, що використання окремого ESP8266-01 Wi-Fi модулю, не є оптимальним рішенням через незручність пайці і подальших прошивок (оновлень системи) – у даному випадку, щоб оновити Arduino, необхідно відключати ESP, інакше завантажувач Arduino не спрацює. Інше рішення даної проблеми, це підключати ESP до Serial, а для Arduino використовувати SoftwareSerial для зв'язку з Wi-Fi модулем.

Рекомендується використовувати окреме, резервне джерело живлення та фільтрувати його. Це необхідно через те, що при передачі пакетів інформації, модуль на короткий час починає споживати більше живлення. Якщо не вистачає запасу живлення, то модуль може генерувати помилку, вимикатися, або втрачати з'єднання з сервером.

Останнім прикладом є IoT-система з сенсором тиску для вимірювання рівня рідини з підключеним насосом (рис. 1.6.).

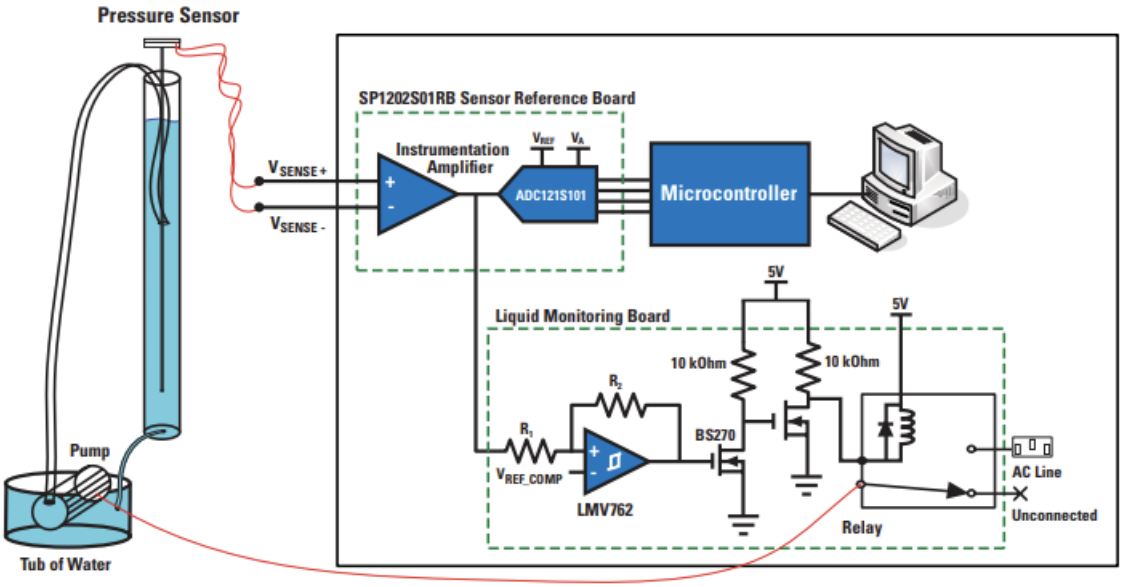


Рисунок 1.6 – Система моніторингу за рівнем рідини з сенсором тиску та електричним насосом [12, с.4]

Тиск у резервуарі визначається шляхом використання у роботі сенсору тиску NPC1210. Базовий вихідний сигнал сенсору становить 50 мВ для 25.4 см водяного стовпа. Таким чином, з цього виходить, що 25.4 см узгоджується з типовим диференційним вихідним рівнем напруги сенсора 50 мВ. Дана залежність дуже важлива при виконанні обчислення висоти рівня рідини у резервуарі та вибору

оптимального АЦП та підсилювача для системи моніторингу за рівнем рідини [12, с.2].

Компонент SP1202S01RB, у даній системі використовуватися як друкована плата для диференціального датчика тиску. Вона включає в себе диференційно-несиметричне налаштування та інструментальний підсилювач, що під'єднаний до несиметричного (1 канал) перетворювача АЦП121S101 (має 12 розрядів) [12, с.2]. Даний АЦП, має власний інструментальний підсилювач, але в роботі застосовує диференціальний АЦП161S626 (1 канал, 16 розрядів) в несиметричному режимі.

Обидві конфігурації містять каскад підсилення для збільшення вихідного сигналу сенсору до задовільного робочого діапазону АЦП від 0 В до 4.1 В. Вихідний код АЦП отримується мікроконтролером через SPI і прямує в ПК для аналізу.

Принцип дії: рідина постійно надходить з даного резервуара (трубки) у зовнішній бак з рідиною. Зовнішній бак включає в себе електричний насос. При низькому рівні рідини у резервуарі, даний насос запускається та перекачує рідину з зовнішнього баку у зворотному напрямку, тобто до резервуара. При даному процесі, коли рівень рідини досягає заданої межі (приблизно біля верхньої точки резервуара), електричний насос вимикається і переходить у режим очікування, коли рідина опуститься до нижньої межі, що являє собою тригер для вмикання насосу та повторення алгоритму. Даний цикл буде повторюватися до вимкнення приладу із мережі живлення.

Для реалізації даних переливань рідини, додається компаратор LMV762 з гістерезисом до уже вказаної конструкції. Вхід АЦП порівнюється з опорною напругою компаратора VREF_COMP. Якщо VIN_ADC більше VREF_COMP, то вихід компаратора має високий рівень, в іншому випадку низький. До компаратора додається гістерезис (рис. 1.7), що дозволяє створити дві межі перемикання на VIN1 та VIN2. Ці межі перемикання позначають положення коли насос вмикається та вимикається [12, с.4].

На виході, компаратор під’єднаний до двох польових транзисторів із живленням, що виконують роль буферу. У даному випадку інвертор не є необхідним через те, що головне призначення польових транзисторів – постачання достатнього струму для включення реле. Таким чином, у випадку коли один вивід підключений до джерела змінного струму, а інший ні, вказане вище реле буде перемикатися між зв’язком «насос-живлення» та зв’язком «насос-земля».

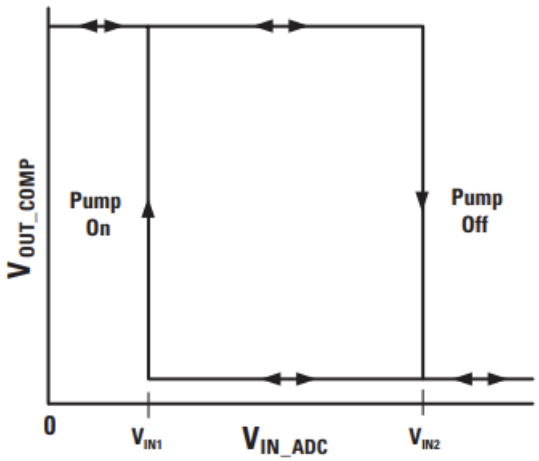


Рисунок 1.7 – Гістерезис [12, с.4]

Підводячи висновки до даній системі, можна сказати, що у даній системі виконується автоматичний контроль рідини паралельно з моніторингом поточного рівня рідини. Для перетворення аналогової напруги у цифровий код, у якому програмне забезпечення комп'ютера може математично обчислити висоту рідини, необхідний АЦП. Система не підключена до хмари, проте має прямий зв’язок к ПК, на програмному забезпеченні якого здійснюється уся обробка інформації.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ МЕТОДІВ І ПРИСТРОЇВ ДЛЯ ВИМІРЮВАННЯ РІВНЯ РІДИНИ У ЄМНОСТІ

2.1. Аналіз сфери та умов експлуатації пристрою

IoT-система контролю рівня рідини розроблюється для застосування у опалювальних системах тепличних господарств малого розміру, теплицях дачного типу тощо.

Основні вимоги до системи:

- робота у температурному діапазоні від 1 до 75 °C;
- сенсори повинні мати захист від вологи мінімум IP65 [10];
- віддалений та надійний доступ до даних відносно рівня рідини у розширювальному баку опалювальної системи;
- низка вартість системи;
- можливість під’єднання до 8 сенсорів;
- можливість встановлення баку під кутом;
- можливість працювати з рідинами типу: вода;
- можливість працювати в умовах без наявності піни, забруднення, рухливих поверхень рідини.

Розглянемо більш детально сферу та умови експлуатації. Основними стейкхолдерами даної системи є оператори або працівники теплиці – їх задача контроль за рівнем рідини та виконання своїх поточних задач.

Заданий діапазон температури будемо був обраний по таким причинах: згідно стандартам, у періоди опалення, у централізованому опаленні мінімальна температура становить 70 °C, а максимальна 95 °C. Для автономного опалення, мінімальна температура 40 °C, а максимальна 80 °C. У приватних будинках, оптимальна температура становить 50-60 °C.

Температура може коливатися в залежності від типу опалювальної системи, зовнішньої температури, матеріалу труб, радіатора, котлу [23, с.131]. Також

зазначимо, що якщо температура буде занадто висока, то збільшується ризик перегріву котлу.

Будемо орієнтуватися на значення систем автономного опалення і опалення приватного будинку. Значення температури централізованого опалення не будемо враховувати через специфіку його роботи. Централізоване опалення використовується для опалення дуже великої території з переважним зростанням будівлі у гору, що робить затрати на підтримку температури у самому дальньому квадраті будівлі дуже важким, тому, це одна з причин підтримки такої великої температури. Інші фактори, це особливості системи підключення, проте їх ми розглядати у рамках даної роботи не будемо.

Температура у розширювальному баку завжди нижче ніж у основній системі опалення. Даний бак, це елемент компенсації теплового розширення води, а не частина активного теплообміну. Бак ставлять у місцях зверху, та на зворотному шляху води. Таким чином, бак контактує з більш холодною водою її температура менша. Виходячи з цього сенсори та електроніка повинна мати робочій діапазон від 1 до 70-75 °C.

Пристрій буде працювати в умовах підвищеної вологості, тому повинен мати захист від вологи, можливих бризок води тощо. Таким чином, рекомендується рівень захисту починаючі з IP65 [10]. Перша цифра «6» – повний захист від пилу, тобто частинки будь-якого розміру не можуть проникнути всередину корпусу. Друга цифра «5» – захист від водяних струменів. Другу цифру можна також обрати як «7», якщо пристрій, або його компоненти планується занурювати у воду. На даному етапі, вирішено залишити рівень захисту від вологи «5».

Через те, що дана система планується для використання у малих тепличних господарствах, нам потрібно орієнтуватися на обмежений бюджет. Таким чином, необхідно віддавати пріоритет недорогим за вартістю компонентам та сенсорам, застосуванню простої архітектури без використання додаткових компонентів. Також необхідно розглянути можливість під’єднання декількох сенсорів до одного пристрою, для можливості контролювати декілька баків одночасно.

Необхідно зазначити, що стандартне розташування баку, це рівна поверхня. Проте для деяких випадків, краще розташувати їх під кутом. Не всі сенсори, можуть вірно визначати рівень води у подібних положеннях, і таким чином, потрібно визначити оптимальні методи та сенсори для використання у подібних умовах.

2.2. Проведення дослідження методів і сенсорів для вимірювання рівня рідини

2.2.1. Загальний огляд методів та принципів вимірювання рівня рідини

Існує велика кількість задач, де необхідно вимірювати рівень рідини у резервуарах різних розмірів і форм. Дані задачі дуже поширені у різноманітних галузях науки та техніки. Наприклад, це поширена задача у системах екологічного моніторингу рік, озер, водосховищ, у дослідженні технічних процесів автомобільних систем, хімічній сфери, контролі витрат рідин при їх зберіганні або транспортуванні тощо. Все частіше з’являється необхідність у вимірюванні рівня рідин з розвитком автоматизації виробничих процесів та технологій.

Вище ми навели приклади застосування подібних вимірів для різноманітних галузей наук та техніки. Проте, для кожного з цих випадків необхідно використовувати окремий підхід до вимірювання та визначення вірного обладнання. Даний підхід визначається загальними і вузькопрофільними вимогами. Останні висуваються для виконання спеціальних вимірів і робіт, та визначаються для кожного випадку і кожної галузі науки окремо.

До основних універсальних вимог, що висуваються до вимірювачів рівня, відносяться:

- відповідність вимогам галузевих нормативів;
- захист від навмисного спотворення результатів вимірів;
- висока експлуатаційна надійність, в умовах впливу факторів реального виробничого середовища, для якого розроблюється пристрій;

- забезпеченість ресурсу функціонування не менше 5–15 років за мінімальної трудомісткості проведення регламентних заходів таких як: очищення, перевірка, технічне обслуговування, відновлювальний ремонт;

- забезпечення вказаної точності вимірюванні у середовищі (звичайні, багатокомпонентні, нестабільні, кислотні, їдкі середовища, тощо), для якого розроблюється пристрій;

- інваріантність вимірювальних характеристик та передбачуване (градієнтне) зниження точності при наростанні дестабілізуючих впливів (температурні коливання, зміна щільності середовища, хвилювання поверхні, відхилення геометрії резервуара та ін.);

- економічна обґрунтованість та зручність використання даного методу та інструменту вимірювання, з урахуванням співвідношення «функціональність-витрати».

Наведемо приклади деяких спеціальних умов до вимірювачів рівня:

- у транспортній сфері – робота в умовах вібрацій та/або нахилу резервуара;

- у житлово-комунальному господарстві – робота в рухомих, сильно забруднених рідинах (вода, каналізаційні стоки);

- у сільському господарстві – робота від автономних джерел живлення, можливість передачі даних по бездротових/дротових мережах та мінімальна вартість;

- у харчовій, хімічній та нафтохімічній промисловості — вимірювання рівня з малою похибкою, робота за наявності пилу, конденсату, піни, тощо.

Вимоги до точності вимірювань рівня рідини визначається для кожної задачі окремо. На них можуть впливати як державні нормативні документи, стандарти, вимоги проектних організацій, так і характеристики безпосередньо рідини (тип рідини, піна, повітря, забруднення) та резервуара у якому вони знаходяться.

Точність вимірювання, що визначається методом вимірювання і точністю пристрою, необхідно узгодити між собою, щоб уникнути зайвих фінансових витрат та використовувати обладнання з максимальною ефективністю. Наприклад, існує обладнання, яке забезпечує надзвичайно високий рівень точності вимірювання.

Проте, його вартість дуже висока, що потребує використання даного обладнання суто за ціллю. Також, невірно обраний метод вимірювання може спотворити дані отримані від пристрою, що зводить на ні, усю потужність пристрою.

2.2.2. Класифікація і аналіз існуючих методів

Приведемо найбільш розповсюджені методи дослідження рівня рідини зі звуженням вибірки у користь обраної проблематики дослідження. Виділимо три великі категорії або напрями методів: хвильові, не хвильові, комбіновані. Кожна категорія включає у себе велику кількість методів:

а) хвильові методи:

1) ультразвуковий локаційний;

2) радіолокаційний;

3) лазерний локаційний;

4) оптичний.

б) не хвильові методи:

1) кондуктиний;

2) гідростатичний;

3) буйковий;

4) поплавковий механічний.

в) комбіновані методи:

1) поплавковий магнітострикційний;

2) поплавковий радіолокаційний.

Загальні особливості кожної з категорій:

а) хвильові методи вимірювання ґрунтуються на фізичних явищах, що супроводжують поширення електромагнітних або акустичних хвиль у середовищах, таких як рідини, парогазові суміші, або в конструктивних елементах систем що знаходяться в контакті з вимірюваними середовищами.

б) нехвильові методи реалізують інші принципи виміру, які, як виходить з назви, не пов'язані з хвильовими процесами. Прикладом таких принципів може

бути: реєстрація зміни ємності конденсаторного вимірювального осередку, вимірювання гідростатичного тиску обумовленого висотою стовпа рідини, визначення величини сили, що виштовхує або що діє на занурене тіло [21, с.59]. Найбільш поширеним, простим і раннім рішенням, є використання звичайних механічних поплавкових та буйкових систем.

в) комбіновані методи, як виходить з назви, використовують підхід об'єднання методів інших категорій, для того щоб покрити слабкі сторони один одного. Тобто, комбіновані методи інтегрують у собі елементи як хвильових, так и і нехвильових підходів. Пояснимо на простому прикладі: поплавковий магнітострикційний метод – зонд занурюється у технологічне середовище, а за рахунок того, що поплавок вільно рухається вздовж зонда, здійснюється контроль рівня рідини. Позиціонування поплавця визначається за рахунок реєстрації механічних коливань, індукованих в магнітострикційному хвилеводі [21, с.63].

Кожна з цих категорій має свої позитивні та негативні сторони, які дозволяють точно підібрати необхідний метод для кожного випадку, з яким стикається дослідник. Для кожного з методів також існує певні експлуатаційні норми, які визначають ефективність використання даного методу. Прикладом цього може бути діапазон температур рідин, характеристика рідини: липкі, в'язкі, сильно забруднені рідин, тощо.

Наявність сильного хвилювання чи піни що лежить на поверхні рідини, різко зменшує коефіцієнт відбиття акустичного і електромагнітного випромінювання, роблячи більш придатними для даного кола завдань нехвильові і комбіновані методи. Якщо до цього додати сильні бічні навантаження на конструкцію та небезпеку її поломки при накопиченні сміття (гілок, листя, клаптиків паперу та ін.), то і більшість нехвильових і комбінованих методів стають непридатними для використання.

У рухомих резервуарах (паливних баках, цистернах суден), які можуть бути сильно, більш ніж на 10-15°, нахилені до горизонту, не раціонально використовувати поплавкові методи через небезпеку блокування рухомих елементів.

Таким чином, усі методи вимірювання мають свої переваги, недоліки і похибки. У сукупності, все це визначає конкретні напрями і сфери використання даних методів. Також важливо зазначити, що похибки та вимоги до експлуатації можуть бути частково компенсовані за рахунок використання додаткових технічних засобів. Візьмемо приклад вище, де рідина має сильне хвилювання, піну, та забруднення гілками, листями, папером, тощо, а також бічні навантаження на конструкцію, ми сказали, що хвильові, нехвильові, комбіновані методи, будуть застосовані з малою ефективністю. Проте, якщо ввести корегування за рахунок додаткових технічних засобів, наприклад: самоочищення, фізична амортизація резервуара, хімічна ліквідація піни, або інше, то це одразу робить більшість методів придатними для вимірювання.

Проте, повна компенсація як правило неможлива через фізичні, економічні та експлуатаційні обмеження, тільки часткова. Також важливо розуміти, що майже будь-яка модернізація і ускладнення пристрою або системи, веде до того, що система стає дорожчою, що не завжди є бажаним результатом.

2.2.3. Характеристика методів та обґрунтування вибору оптимального варіанта

У даному дослідженні, в нас є певна задача, сфера використання, та бюджет. Таким чином, для вибору оптимального методу по-перше будемо оцінювати вартість (сенсору), діапазон вимірювання, та нижню і верхню зони.

Спочатку дамо пояснення що таке нижня і верхня зони. Це зони, де знаходиться перехідний рівень для виміру рівня рідини. Іншими словами, це ділянки резервуару, де проведення вимірів або неможлива, або точність дуже мала. Верхня зона, це мінімальна відстань від верхньої опорної точки до поверхні рідини. Нижня зона, це відстань від кінця зонда до нижньої частини резервуара. Логіка появи даних зон різноманітна, наприклад, нижня зона може з’являтися через паразитні відображення від дна, що мають більшу потужність порівняно з корисним сигналом, або наприклад, деякі хвильові методи потребують мінімальної товщини

рідини, щоб вірно відбитися від неї. Також, запас зверху (верхня зона), може бути необхідний з міркувань безпеки. Наприклад, це потрібно, щоб сигнал не підходив дуже близько до сенсору, щоб він не був занурений у рідину, або просто щоб резервуар не переповнився.

Загальні характеристики методів вимірювання рівні рідини, з акцентом на тему дослідження і сферу використання, наведемо у таблиці 2.1.

Таблиця 2.1 – Загальні характеристики методів вимірювання рівня

Методи	**Діапазон, м.**	**Верхня зона, м.**	**Нижня зона, м.**	**Вартість**
Хвильові				
Ультразвуковий локаційний	до 5…15	≈ 0	≈ від 0,05 до 0,15	Середня
Радіолокаційний	до 25…30	від 0,5 до 1,0	≈ 0	Висока
Лазерний локаційний	до 15…35	від 0,4 до 0,6	≈ 0	Висока
Оптичний	до 2…3	менше 0.1	менше 0,1	Середня
Не хвильові				
Кондуктивний	до 2	≈ 0	≈ 0	Низька
Гідростатичний	до 100	≈ 0	≈ 0	Середня
Буйковий	до 4	від 0,3 до 0,5	≈ 0	Середня
Поплавковий механічний з сервоприводом	до 20..40	від 0,05 до 0,3	від 0,05 до 0,3	Висока
Комбіновані				
Поплавковий магнітострикційний	6...30	від 0,05 до 0,3	від 0,05 до 0,3	Середня
Поплавковий радіолокаційний	до 25-30	від 0,5 до 1,0	від 0,05 до 0,1	Висока

Звернемо увагу, що у хвильових та комбінованих локаційних методах, позначені дуже високі значення верхньої зони. Це пояснюється тим, що при роботі у близькій від сенсора зоні (менш ніж заявлено), виникає велика кількість

віддзеркалень від поверхні рідини і кришки резервуару, перевантаженням приймача сенсору і унеможливленням коректного визначення рівня рідини.

Дані зони у інших випадках залежать більше від конструктивних особливостей резервуару та/або системи вимірювання (висота, форма, додаткові елементи). Якщо сенсор заходить у вказані зони, то збільшується спотворення як даних що надходять до сенсорів, так, відповідно і даних що дослідник отримує після обробки. В деяких випадках можливий повний відказ пристрою.

Також необхідно вказати на можливі фактори, що зменшують діапазон вимірювання. Особливо критичне це для хвильових локаційних методів. Наприклад, це можуть буть предмети або поверхні, що здатні віддзеркалювати сигнал: стіни, сходи, тощо [24, с.5].

Проте, щоб надати більш коректний висновок для даного дослідження, необхідно також дослідити основні експлуатаційні обмеження (таблиця 2.2). За рахунок впливу даних факторів (фактори зовнішнього середовища, у якому планується використання даних сенсорів, пристроїв або систем), коректна робота пристроїв може бути ускладнена або зовсім неможлива. Також є диференціація між тим, що пристрій може вийти з ладу та збиранням некоректних даних. Кожен метод та сенсор має свій набор обмежень, не дотримання до яких, може спотворити результат дослідження.

Хвильові методи мають більш вузькі межі використання за температурою, порівняно з іншими. Це пов’язано в першу чергу з тим, що в них використовуються електроні модулі, для яких температурний вплив має критичне значення.

Окрім цього, ми бачимо, що хвильові методи дуже чутливі до осадження парогазових фракцій на компонентах приладу. Хвильові методи погано працюють у таких умовах через те, що осад спотворює або навіть може поглинати сигнал, виникають помилкові відображення, як правило змінюються властивості середовища, і як наслідок усього цього, знижується надійність вимірів.

Таблиця 2.2 – Основні експлуатаційні обмеження

Методи	**Фактори впливу**				
	Діапазон температур рідини, °С	Осадження фракцій парогазової суміші (води, парогазової суміші) на компоненти приладу	Липкі, в'язкі, сильно забруднені рідини	Наявність піни та сильне хвилювання поверхні рідини	Рухомі резервуари з можливістю їх сильного коливання
Хвильові					
Ультразвуковий локаційний	від -40 до +80	–	+	–	+
Радіолокаційний	від -40 до +80	У	+	У	+
Лазерний локаційний	від -40 до +60	–	+	–	+
Оптичний	від -40 до +60	–	–	–	+
Не хвильові					
Кондуктивний	від 0 до +120	+	–	У	+
Гідростатичний	від -40 до +100	+	–	+	+
Буйковий	від -60 до +400	+	–	+	У
Поплавковий механічний з сервоприводом	від -50 до +150	+	–	У	-
Комбіновані					
Поплавковий магнітострикційний	від -50 до +150	+	–	+	-
Поплавковий радіолокаційний	від -40 до +80	У	–	+	+

У даній таблиці, позначення: «У» – ускладнене використання, «–» – використання неможливе, «+» – використання можливе.

Для вимірів у в’язких або забруднених рідинах, краще справляються саме хвильові методи. Це відбувається тому, що хвильові методи – безконтактні методи,

які напряму не контактують із рідиною. Звісно, більшість з цих методів занадто коштовні, проте якщо не має можливості уникнути забруднення рідини (щоб використати більш дешеві методи), то обирають хвильові методи [26, с.32].

Наявність піни та/або сильне хвилювання поверхні рідин, також робить неможливим коректну роботу хвильових методів. Це пояснюється тим, що за рахунок сильного хвилювання поверхні рідини, формується динамічна, нерівна поверхня і за рахунок цього отримаємо коливання сигналу та нестабільні дані. Піна, у свою чергу, послаблює віддзеркалювання і посилює розсіювання сигналу, результатом чого є невірний рівень або повна відсутність сигналу.

Обидва ці фактори також формують багатократні віддзеркалювання (бульбашки, змішане середовище) та нахил поверхні рідини, за рахунок чого, сигнал уходить у бік і не відбивається до сенсору. Таким чином, вимірювання рівня даними методами буде нестійкім через малу амплітуду віддзеркаленого сигналу.

Стосовно рухомих конструкцій, обмеження виникають головним чином у поплавкових методів. Рухомі конструкції можуть мати різний рівень нахилу (динамічний рівень ще більше ускладнює вимірювання), в результаті чого, механічні, рухливі з’єднання можуть бути заблоковані, що веде до спотворювання результатів вимірювання, а у гіршому випадку, до поломки пристрою.

Хвильові методи, незважаючи на те що дослідження можливе, також мають сильне обмеження. Через можливий рух резервуару, рідина буде рухатись, і ми отримуємо проблему, яка описана вище. При статичному нахилі резервуара, потрібно враховувати позицію пристрою, тому що рівень рідини буде відрізнятися.

На основі вимог описаних вище, щоб знайти найбільш оптимальний метод та сенсор за думкою дослідника, для рішення проблеми з визначенням рівня рідині у опалювальній системі даної сфери сільського господарства, будемо йти від зворотного, та опишемо які сенсори не слід використовувати для рішення даної проблеми:

- методи та пристрої з високою вартістю (радіолокаційний, лазерний локаційний, поплавковий механічний з сервприводом, комбінований поплавковий радіолокаційний);

- буйковий має велику верхню зону, що буде встановлювати певні труднощі при інсталяції у резервуари невеликих розмірів;

- комбінований поплавковий магнітострикційний має коливання верхньої і нижньої зони у великому діапазоні. Це може бути пов’язано з конструкцією системи (електроніка зверху, різноманітний тип і розмір поплавків, тощо). Даний діапазон значень показує, що для отримання найбільш придатної зони, потребується шукати спеціальні сенсори, тобто падає адаптивність та інколи зростає вартість;

- оптичний метод вимірювання має дуже звужену за температурою сферу застосування. Як ми вказували вище, на прикладі приватних будинків, оптимальна температура становить 50-60 °С, що становить максимум для цього методу. Проте температура може бути більша, і в даному випадку даний метод не підходить;

Таким чином, найбільш підходящими методами є не хвильовий кондуктивний, не хвильовий гідростатичний, і хвильовий ультразвуковий.

Розглянемо більш детально не хвильовий кондуктивний і не хвильовий гідростатичний методи:

а) Гідростатичний метод вимірювання завжди залежить від щільності середовища. Тобто, якщо до рідини (в нашому випадку води) додати антифриз, сіль, або інші хімічні речовини, змінити температуру, то дані отримані даним методом спотворюються.

Наприклад, при підвищенні температури молекули води рухаються швидше і відстань між ними збільшується, що знижує щільність. Напроти, при охолодженні молекули починають утворювати вільніші структури, що призводить до зменшення щільності.

Вимірювання рівня даним методом проходить за формулою гідростатичного тиску (формула 2.1, 2.2).

$$P = \rho * g * h \tag{2.1}$$

$$h = \frac{P}{\rho * g} \tag{2.2}$$

де

- P – тиск (Па);

- ρ – щільність рідини (кг/м^3);

- g – прискорення вільного падіння (9,81 м/с^2);

- h – висота стовпа рідини (м).

б) Кондуктивний метод, у свою чергу не залежить від щільності. Він працює на основі зміни електричного опору, який залежить від наявності рідини на певному рівні сенсора і електропровідності рідини.

Тобто, вода (або інші провідні рідини), має певну електропровідність. У дану рідину вертикально занурюється паралельні струмопровідні електроди. При наявності між ними рідини формується струмопровідний шлях і загальний опір зменшується. Чим вищий рівень рідини, тим більша площа контакту, відповідно нижчий опір. Таким чином, якщо рідина відсутня (електроди не занурені до рідини), то опір прагне позитивної нескінченності – ланцюг розімкнений. І навпаки, коли рідина присутня, опір падає – ланцюг замикається.

Важливо відмітити роль температури рідини у вимірюванні рівня. Температура важлива як для гідростатичного, так і для кондуктивного методу. Зі збільшенням температури в'язкість води зменшується, рухливість іонів зростає і іонні реакції відбуваються активніше. Тобто, при підвищенні температури електропровідність зростає, а опір зменшується.

Потрібно зазначити, що кондуктивний метод часто використовують щоб оцінити наявність рідини у цілому (модель «сухо/мокро»), або приблизні значення. У цих випадках, температурні коригування надлишкові. Проте, для отримання необхідної точності рівня, врахування температура носить обов'язковий характер. Щоб розрахувати залежність електропровідність від температури, скористаємося формулою:

$$k_T = k_{25} * [1 + \alpha * (T - 25)] \quad (2.3)$$

де

- k_T – електропровідність при температурі T (°C);

- k_{25} – електропровідність при 25 °C ~ 0.5 mS/cm;

- α – температурний коефіцієнт електропровідності. Для водопровідної води ~ 0.02 [3, с. 858];

- T – поточна температура води.

У свою чергу, опір обернено пропорційний до провідності (формула 2.4). Тобто, чим вище провідність, тим менший опір.

$$R_T = \frac{1}{k_T} \tag{2.4}$$

Якщо вода буде має більшу температуру, то провідність більша, а опір менше. І навпаки, якщо вода холодніше, то провідність менша, а опір вище. Наприклад, при температурі води 60 °C, отримана через analogRead напруга буде більшою, аніж фактичний рівень рідини. Таким чином, при отримані даних дуже важливо робити температурну поправку, щоб вірно розрахувати рівень занурення сенсору.

Серед цих двох методів, для рішення даної проблеми більш підходить кондуктивний метод, тому що він має меншу вартість і для нього потрібно додатково отримати лише температурні дані, на відміну від гідростатичного, де потребується щільність і температура. Остання, у свою чергу, впливає на щільність рідини. Обмеження кондуктивного методу у порівнянні з іншими методами даної категорії не є критичними для сфери використання сенсору. Також важливо зазначити, що сенсор не працює з дистильованою водою та маслами (вони э частковими або повними діелектриками. Оскільки, середовище чітко визначене, то дані обмеження будуть відігравати роль лише у використанні сенсору із рідиною відмінної від води, та у місцях відмінних від опалювальної системи відкритого типу.

Ультразвуковий, у свою чергу, має свої недоліки, але також покриває слабкі сторони попередніх методів. Однією з головних переваг даного методу є відсутність контакту з рідиною, що дозволяє значно розширити діапазон можливих рідин для дослідження, наприклад, дистильована вода або масло, де попередній метод не може працювати. Ультразвуковий метод робить систему більш функціональною та допомагає уніфікувати використання пристрою та поширити його застосування понад системою опалення теплиці.

РОЗДІЛ 3. РОЗРОБКА ФУНКЦІОНАЛЬНОЇ МОДЕЛІ СИСТЕМИ

Для створення функціональної моделі системи вимірювання рівня рідини у розширювальних баках опалювальних систем теплиць, будемо використовувати нотації IDEF0 та IDEF3.

3.1. Розробка функціональної моделі системи у нотації IDEF0

IDEF0 (Integration Definition for Function Modeling) – це методологія функціонального моделювання, що призначена для опису функцій та їх взаємозв’язок в складних системах.

Ціль IDEF0 – формування функціональної моделі, що демонструє:

- що робить система, які функції вона виконує;
- що необхідно для виконання функцій системи;
- входи, виходи, механізми взаємодії і управління;
- результат роботи функцій;
- хто і що забезпечує виконання.

Для даного проекту, дана методологія додатково важлива ще й по наступним причинам: вона дозволяє виявити усі необхідні функції заздалегідь і допомагає обрати вірний напрям для побудови архітектури системи.

Також, і IDEF0 і IDEF3 можна називати нотаціями через те, що вони описують бізнес-процеси за допомогою графічних об’єктів, а також побудова схем виконується за певними правилами. Тобто, нотація – це графічна мова або система умовних позначень, що дозволяє описати роботу системи, продемонструвати взаємодію між різними компонентами та описати бізнес процеси.

Завершена модель IDEF0 являє собою функціональну діаграму, яка поділяється на різні рівні глибини або деталізації. Для даної системи, вона виглядає наступним чином:

- 0 рівень – на даному рівні знаходиться лише один функціональний блок. Даний рівень встановлює область моделювання і її межу;

- 1 рівень – будується на основі декомпозиції рівня 0. На даному рівні повинні бути відображені функції системи, що реалізовані в рамках основної функції. Для даної роботи буде проведено декомпозицію одного функціонального блоку;

- 2 рівень – будується на основі декомпозиції рівня 1. На даному рівні відображаються конкретні процеси. Буде проведено декомпозицію двох функціональних блоків;

- 3 рівень – будується на основі декомпозиції рівня 2. На даному рівні процес розглядається більш детально. Буде проведено декомпозицію одного функціонального блока.

Розглянемо нульовий рівень моделі (рис. 3.1). Він являє собою загальну структуру додатка та позначає головну мету, навіщо він створюється та які ресурси потребує.

Рисунок 3.1 – Рівень *A0* функціональної діаграми *IDEF0*

Дамо пояснення як читати дану діаграму. Для цього розберемо кожен з основних елементів діаграми:

- верхній елемент – це елемент керування. В даному випадку, керівником виступає користувач, так як саме він вирішує які сенсори підключати та які налаштування робити;

- нижній елемент – це елемент механізму. Механізм – це той компонент, який відповідальний за виконання усіх дій, що потребуються для досягнення цілі – IoT-система.

- лівий елемент – це дані що надходять зовні та які будуть використані для отримання даних відносно рівню рідини. Наприклад, користувач не зможе отримати дані, якщо не створить обліковий запис (для цього потрібна адреса електронної пошти та пароль), не відправить налаштування які сенсори він підключає і до яких портів, та додаткові налаштування окремих сенсорів (критичні рівня рідини, висота резервуара);

- правий елемент – це елемент результату. Коли всі елемент виконали свою роботу, всі дані отримані та перетворені, і вони надходять на вихід. Тобто в результаті, користувач отримає дані, що були виміряні сенсорами та представлені у графічному вигляді.

При наступному рівні декомпозиції, первісна діаграма поділяється на шість функціональних блоків (додаток А). Зазначимо, що вхідні дані залишилися тими ж. Дамо пояснення, що таке «Дія користувача». Більшість вхідних даних – це дії від користувача, тому у даному випадку, це більш абстрактне поняття для позначення цього. Це може бути клік, або звичайна взаємодія з елементом. Те, що дані повторюються, пояснюється тим, що даний рівень функціональної діаграми, це більш детально розібраний на блоки попередній рівень. У ньому не може з’явитися щось принципово нове, тому ще це йшло б врозріз з цілями IDEF.

Розглянемо варіант сценарію, коли користувач, хоче отримувати дані с сенсорів рівня рідини. Також будемо вважати, що фізично, користувач вже підключив сенсори до приладу згідно документації. Далі, користувач повинен або зареєструватися на сайті або авторизуватися, якщо він вже має обліковий запис. Будемо вважати, що в нього немає запису і йому потрібна реєстрація. Після реєстрації, користувачу необхідно додати унікальний ID пристрою до свого

облікового запису. Це дозволить налагодити зв’язок між пристроєм, обліковим записом і сервером. Наступний крок, це налаштування підключення сенсорів. У розділі «Settings», необхідно на графічному макеті пристрою, налаштувати входи до яких підключені сенсори. Загальна схема це підключення кондуктивних сенсорів до лівої половини пристрою (аналогові входи), а ультразвукові до лівої і дзеркально до правої половини.

Після загального налаштування підключення сенсорів, необхідно налаштувати кожен сенсор окремо. Це не обов’язковий крок для кондуктивного сенсора, проте все ж бажаний. Серед додаткових налаштувань є мінімальний і максимальний рівень рідини, при досягненні якого буде виконане звукове і візуальне попередження. Окрім цього можна встановити висоту резервуару, що є необхідним параметром для ультразвукового сенсора. Це необхідно для коректного обрахування проценту зайнятого простору у резервуарі.

Далі, необхідно підтвердити свій вибір, натиснувши кнопку «Enter» і зберегти налаштування. Налаштування прямують до серверу, а потім до пристрою. Після цього, необхідно перейти до розділу з відображенням інформації з сенсорів і отримати результат – актуальний рівень рідини у розширювальному баку опалювальної системи.

Виконаємо декомпозицію наступних двох блоків: A1 та A4, а також розглянемо отримані результати.

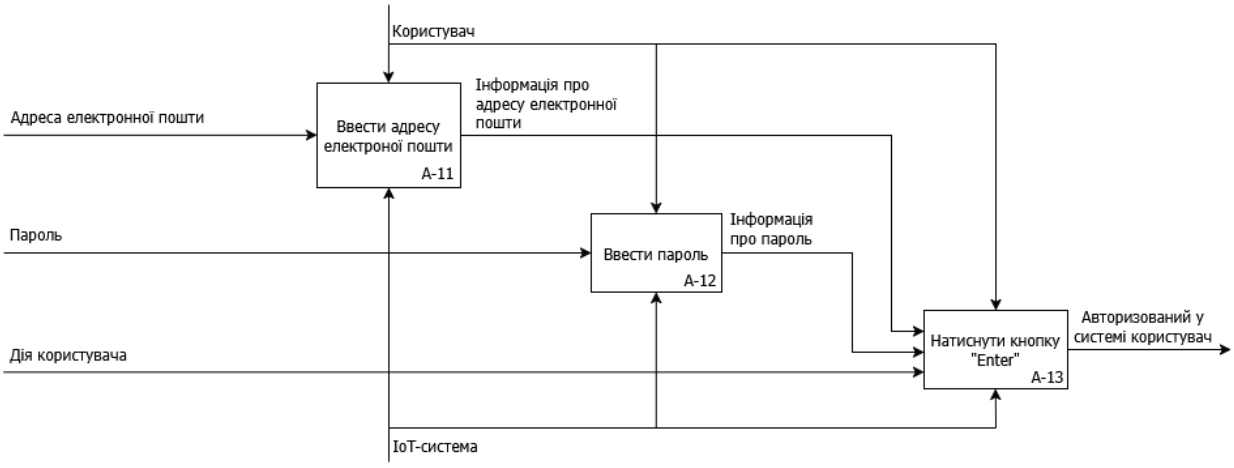


Рисунок 3.2 – Декомпозиція блоку *A1* функціональної діаграми *IDEF0*

Провівши декомпозицію блоку «А1» (рис. 3.2.), отримуємо діаграму «Авторизація», що складається з трьох блоків. Потрібно зазначити, для реєстрації потрібно додати додатковий блок: «Ввести код підтвердження». Даний код надсилається на електрону адресу вказану на попередніх кроках. Таким чином, для виконання авторизації, або реєстрації, необхідно послідовно виконати кроки наведені вище. Кожен крок пов'язаний один з одним і якщо користувач не завершить навіть один з них, авторизація або реєстрація буде неможлива.

У випадку з авторизацією, блок А-11 та А-12 це незалежні блоки, котрим неважливо що користувач ввів на попередньому кроці. Їм достатньо лише щоб дані були заповнені, після чого вони направляються на сервер, проходять перевірку і якщо все гаразд, то пропускають користувача до наступного кроку (або блоку).

Розглянемо блок А-4, на тому ж рівні декомпозиції (додаток Б). Даний блок присвячений налаштуванню окремого сенсору, та необхідний для коректної роботи ультразвукового сенсору. Усі правила описані вище, також застосовуються і до даного блоку. Важливо заповнити усі поля даними для коректного продовження роботи. Тобто, для того щоб продовжити роботу, необхідно в будь-якому порядку виконати блоки «А-41 – А-43», і тільки після цього переходити до блоку «А-44». Це дозволить надати максимальну інформацію щодо налаштувань і забезпечить коректну роботу сенсорів і системи загалом.

Найбільший рівень глибини декомпозиції досягається на третьому рівні (рис. 3.3). Для прикладу розберемо блок «А-11» – процес введення електронної пошти, який розпадається на три блока. Таким чином, ми продовжуємо ланцюжок перетворень з поступово зростаючою деталізацію. А даному випадку, користувачу необхідно активувати поле для вводу, ввести адресу електронної пошти і натиснути «Enter» або на інше поле.

Подальша декомпозиція відносно даної системи є надлишкової. При створенні діаграми необхідно сконцентруватися на побудові і загальному відображенні бізнес-процесів і не торкатися програмної реалізації.

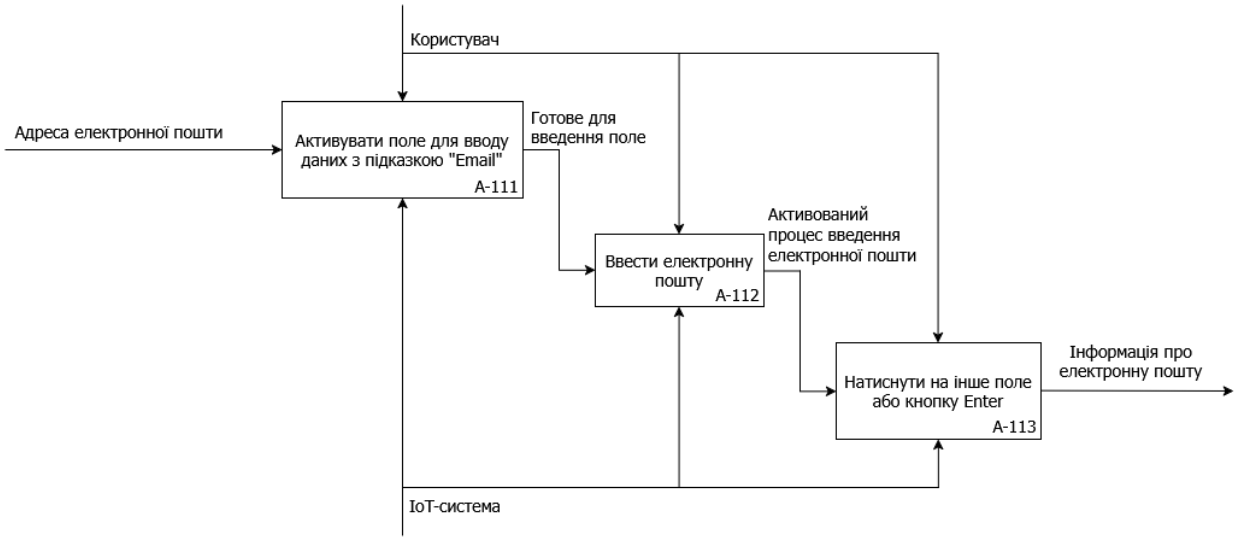


Рисунок 3.3 – Декомпозиція блоку *A-11* функціональної діаграми *IDEF0*

3.2. Розробка моделі процесів у нотації IDEF3

Нотація IDEF3 – це методологія моделювання та стандарт документування процесів (у т. ч. технологічних процесів), що відбуваються в системі, а також механізм збирання інформації про процеси. Це найважливіша нотація після IDEF0 і вона призначена для опису потоків робіт бізнес-процесу (WFD). Як правило, використовується спільно з нотацією IDEF0, але може і окремо.

IDEF3 показує причинно-наслідкові зв'язки між ситуаціями та подіями у зрозумілій експерту (аналітику) формі, використовуючи структурний метод вираження знань про те, як функціонує система, процес тощо.

Таким чином, IDEF3 описує як система працює протягом певного часу. IDEF3 виконує наступні функції:

- демонструє які процеси виконуються;

- визначає порядок виконання процесів;

- представляє які сценарії роботи можливі для даної системи.

Розглянемо нотацію IDEF3 створену для даної системи (додаток В). Дана нотація буде описувати механізм отримання актуальної інформації щодо рівня рідини у розширювальному баку опалювальної системи теплиці.

Перш за все користувачу потрібно увімкнути пристрій та підключити сенсори. Далі необхідно відкрити мобільний додаток або веб-сайт і авторизуватися або

зареєструватися, якщо в користувача немає облікового запису (саме цей сценарій ми будемо описувати).

Даний крок позначений знаком «&» (логічне «І»), тобто він зобов’язує послідовно виконати усі дії що в ньому знаходяться. Так, користувач не зможе продовжити процес реєстрація, якщо не введе електрону пошту і пароль, а потім і код підтвердження. Далі користувачу необхідно перейти до розділу або екрану «Profile», де вій має прив’язати ID пристрою. Наступним кроком він має виконати налаштування підключення сенсорів і окремі налаштування для кожного з них. Звернемо увагу, що знак «О» (логічне «АБО»), дозволяє користувачу виконати або щось одне, або декілька варіантів. Наприкінці необхідно підтвердити вибір, зберегти налаштування і перейти до розділу «Sensors». На цьому екрані буде здійснюватися демонстрація актуального рівня рідини за усіма сенсорами, що налаштував користувач. Логічним завершенням буде закриття додатку або веб-сайту, при відсутності необхідності у моніторингу рівня рідини на даний момент.

РОЗДІЛ 4. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПРИЛАДУ

Загальна структура приладу буде складатися з плати мікроконтролеру, COMPIM, кондуктивного, ультразвукового та температурних сенсорів. Компоненти мають відповідати загальним цілям, умовам використання та вимогам до пристрою у цілому.

4.1. Вибір мікроконтролерної платформи

Для даного проекту, основними вимогами до плати мікроконтролера і самого мікроконтролера є:

- низька вартість;
- 8 аналогових і 8 цифрових виходів;
- наявність вбудованого Wi-Fi модулю;
- здатність оброблювати JSON.

Основні варіанти що були розглянуті для даного проекту, це:

- Arduino Nano RP2040;
- Arduino Nano ESP32;
- Arduino UNO R4 Minima;
- ESP32-S3 DevKit;
- ESP32-C3 DevKit.

Дані за даними елементами відобразимо у порівняльній таблиці 4.1.

Таблиця 4.1. Порівняння мікроконтролерів

	Arduino			**ESP32**	
	Nano RP2040	**Nano ESP32**	**UNO R4 Minima**	**S3 DevKit**	**C3 DevKit**
Архітектура	Raspberry Pi, Cortex-M0+	ESP32-S3, Xtensa	Arm Cortex-M4	Xtensa LX7	RISC-V single-core
Частота ядра	133 MHz, dual-core	до 240 MHz, dual-core	8 MHz, single-core	до 240 MHz**,** dual-core	160 MHz (single core)

	Arduino			ESP32	
	Nano RP2040	**Nano ESP32**	**UNO R4 Minima**	**S3 DevKit**	**C3 DevKit**
Тип/об'єм пам'яті	16 MB Flash / 264 KB SRAM	16 MB Flash / 512 KB SRAM	256 KB Flash / 32 KB SRAM	8-16 MB Flash / 512 KB SRAM*	4 MB Flash / ~400 KB SRAM*
Wi-Fi	+	+	-	+	+
Digital I/O pins	20	14	14	25	11
Analog input pins	8	8	6	20	4
Розмір (W/L mm)	18 / 45	18 / 45	68.85 / 53.34	70 / 50	25 / 63
Вартість	21.30$	19.30$	20.00$	15.00$	8.00-9.00$

* – може змінюватися в залежності від модифікації

Для даного проекту нам потрібен вбудований Wi-Fi модуль. Це важливо через те, що плати без вбудованого модулю потребують додаткового модема або модуля. Це у свою чергу збільшує складність і вартість пристрою.

ESP32 C3 не підходить для проекту за кількістю виводів. Arduino UNO R4 має збільшену ціну, великий форм фактор та в ньому відсутній вбудований Wi-Fi.

Варіант, що найбільш підходить для проекту, це або Arduino Nano ESP32, або класична версія ESP32 S3 DevKit. Проте, вибір у користь першого ми робимо через наступні фактори:

- офіційна плата з підтримкою від Arduino;
- спрощена розробка і використання у екосистемі Arduino;
- повна сумісність із бібліотеками Arduino;
- зменшений розмір плати;
- додаткова сертифікація і контроль якості.

Для даного проекту, створимо загальну схему підключення. Підключення сенсорів буде виконано за певною логікою, а саме: кондуктивний сенсор повинен підключатися виключно до аналогових виводів. Для ультразвукового сенсора, вивід

ECHO повинен підключатися до аналогового виводу, наприклад, A0, а TRIG (TR) до цифрового виводу що знаходиться дзеркально: до D10 (рис. 4.1.). Інтерфейс для підключення температурних сенсорів буде завжди підключено до цифрового виводу D3, до якого будуть підключитися дані сенсори. Логіка підключення температурних сенсорів полягає у наступному: порядок даних сенсорів, повністю повторює порядок підключення сенсорів рівня рідини. Тобто, якщо кондуктивний сенсор підключений до A2 (це третій порт), то, відповідно і температурний сенсор повинен бути підключений до третього порту на своєму інтерфейсі.

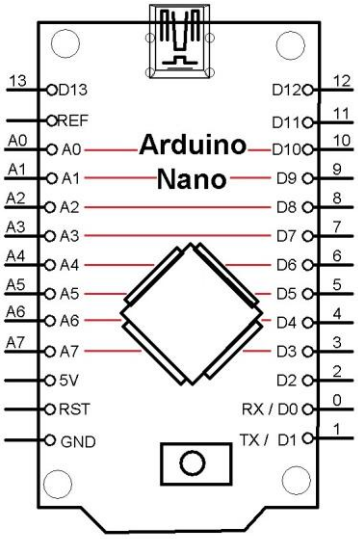


Рисунок 4.1 – Мікроконтролер *Arduino Nano ESP32*

Аргументи для даного спрощеного підходу:

- спрощення конструкції та зменшення додаткових елементів у схемі;

- полегшення розуміння під’єднання;

- перенесення логіки підключення до хмари, що збільшує гнучкість налаштувань приладу та свободу користувачів.

Також важливо зазначити, що незважаючи на те, що у даному Arduino є вбудований Wi-Fi, при розробці проекту і при тестуванні його у віртуальному середовищі, необхідно використовувати додатковий елемент – COMPIM (рис.4.2). Це пов’язано з тим, що у програмі неможливо створити реальне інтернет-з’єднання з сервером через те, що немає TCP/IP стеку, доступу до фізичної мережі та можливості відкривати Sockets на реальні IP-адреси. Таким чином, ми можемо

імітувати обмін даними через UART, а саме створити послідовний порт та налагодити відправку даних від сенсорів на сервер, і прийом налаштувань сенсорів звідти. Даний компонент не потрібен при створенні реального проекту систему, він лише дозволяє імітувати обмін даними, налаштувати логіку і протестувати працездатність. Логіка створена для COMPIM більшою частиною повторює логіку, що розроблювалася б для Wi-Fi модуля.

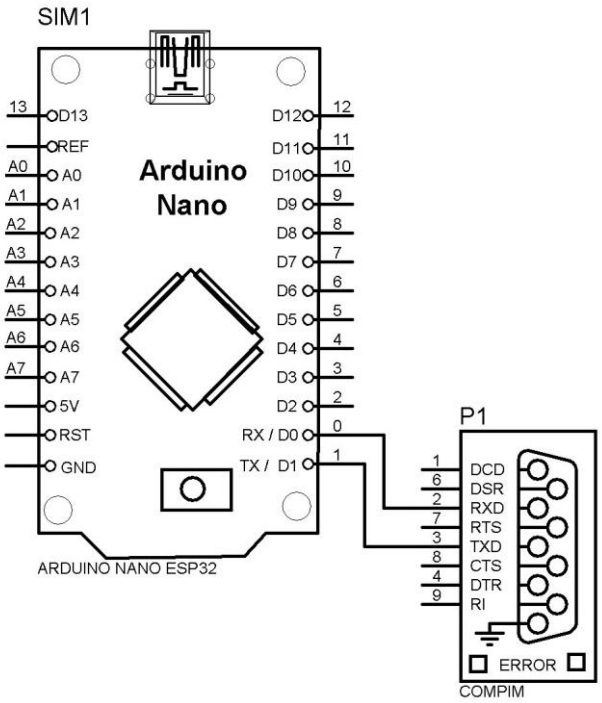


Рисунок 4.2 – Підключення *COMPIM* до *Arduino Nano ESP32*

Загальна структура взаємодія між Arduino і COMPIM виглядає наступним чином:

- Arduino відправляє AT-команди через UART;

- COMPIM перехоплює ці команди та пересилає у віртуальний COM-порт комп’ютера;

- додатково розроблена програма або скрипт зчитує інформацію з цього COM-порта і діє згідно запрограмованої логіки, наприклад, відсилає дані на сервер. Важливо зазначити, що дану програму потрібно писати власноруч. Для даної системи, ми будемо використовувати далі мову JavaScript. Дана проміжна ланка потрібна лише щоб обійти пряме використання Wi-Fi і імітувати рух даних до серверу.

4.2. Ультразвуковий сенсор відстані

Для даного проекту буде обрано ультразвуковий сенсор HC-SR04. Даний модуль простий для використання, має низьке енергоспоживання, малу вартість та широку доступність.

Зазначимо, що HC-SR04 є безконтактним сенсором (рис. 4.3), що перешкоджає виникненню корозії на поверхні сенсору.

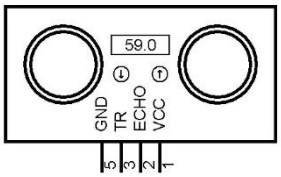


Рисунок 4.3 – Ультразвуковий сенсор *HC-SR04*

За загальною структурою ультразвукового сенсору, позначимо виводи що мають значення для побудови архітектури даного приладу:

- контакт типу «+5В»;
- Trig (TR) – сигналу входу;
- Echo – сигнал виходу;
- GND – «Земля».

Сенсор не придатний для визначення дистанції до звукопоглинаючих об'єктів у силу природи ультразвуку. Проте, для використання у тепличному господарстві, де резервуарами є пластикові баки, це не є критичним фактором.

Як вказувалося вище, логіка підключення виводів для даної системи буде полягати у тому, що ECHO буде підключатися до аналогового виводу, а TRIG (TR) до цифрового (рис. 4.4).

HC-SR04 це сенсор:

- безконтактного типу;
- діапазон вимірювання від 2 до 400см;
- потребує напругу 5В;
- звукова частота: 40 кГц;

- оглядовий кут: 15°;
- кут вимірювання: 30°.

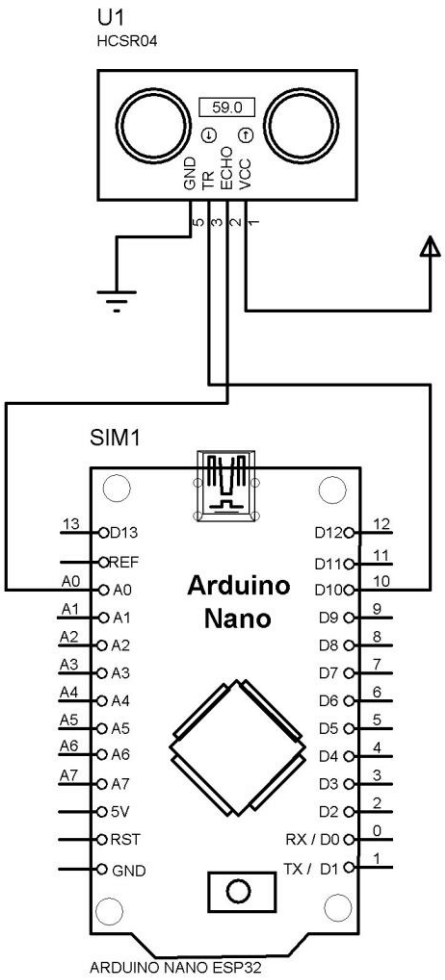


Рисунок 4.4 – Підключення ультразвукового сенсору *HC-SR04* до *Arduino Nano ESP32*

Потрібно окремо позначити, що таке оглядовий кут і кут вимірювання.

Оглядовий кут – це оптимальний кут огляду. У цих межах забезпечується максимальна ефективність, точність і надійність роботи сенсору. Також у даних межах буде вимірювання з найменшою похибкою.

Кут вимірювання – це максимальний кут, у межах якого сенсор може визначити об'єкт (поверхню рідини у нашому випадку). У даному куті, точність знижується по краях, але сенсор все ще виконує свої функції задовільно. Чим менша відстань до об’єкту, тим більша точність для обох цих кутів.

Також важливо робити певну похибку на діапазон вимірювання. Розглянемо полярну діаграму вимірювання ефективності не у ідеальних умовах (рис. 4.5).

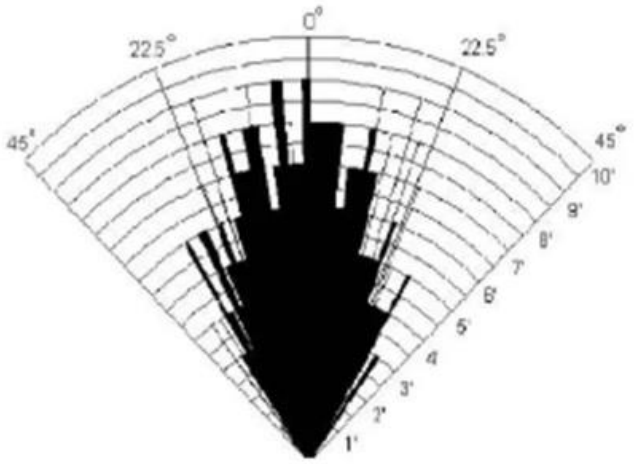


Рисунок 4.5 – Практичне випробування продуктивності сенсору *HC-SR04* [2]

На рисунку 4.5 ми бачимо, градуси що йдуть горизонтально, і фути що йдуть вертикально. 1 фут дорівнює 30.48 см. Сенсор знаходиться знизу діаграми, у місці, звідки розходяться чорні сигнали. Незважаючи на те, що у офіційній документації зазначений діапазон вимірювання від 2 см до 400 см, у практичних тестах, із можливими поміхами, ефективний, гарантований діапазон зменшується до 180-200 см (6-7 футів). Після даної відстані, чим точніше розташований об’єкт, відносно центру сенсору, тим менша можливість отримати помилку (некоректний результат).

Розходження між заявленими даними і тестовими, можуть залежати від додаткових факторів на кшталт: температура середовища, додаткові шуми тощо.

Даний сенсор, як і будь який інший ультразвуковий сенсор, визначає дистанцію до об’єкту за принципом сонара. Загальна модель роботи даного сенсору полягає у наступному: випромінювач посилає пучок ультразвуку, з частотою 40 кГц, який розповсюджується скрізь повітря. Потім, якщо даний пучок натикається на перешкоду, то сигнал відбивається у зворотному напрямку і сенсор отримує дане відображення із певною затримкою. На основі даної затримки визначається наявність об’єкту у цілому та відстань до нього. Проще кажучи, сигнали що генерує сенсор, прямують до об’єкту, відбиваються від нього і повертаються до сенсору. Часовий інтервал цього циклу, допомагає визначити відстань до об’єкту.

Математично, вимірювання дистанції розраховується за такою формулою:

$$s = \frac{t * v_{sos}}{2} \qquad (4.1.)$$

де

- s – відстань;

- t – час за який пучок виходить з сенсору, відбивається і повертається назад;

- v_{sos} – швидкість звуку. Дана швидкість не постійна, а залежить від температури середовища. Якщо брати температуру середовища за 20°C та відсутність пару, то дана величина дорівнює 343.42 м/с або 0.034 см/мкс;

- у кінці необхідно поділити на 2, через те що «t» це час повного шляху, а нас цікавить лише та частина, що йде назад.

Формула розрахунку швидкості звуку із урахуванням температури (T):

$$v_{sos} = 331.3 + 0.606 * T \quad (4.2.)$$

Наприклад, є резервуар висотою 150 см, наповнений на 100 см. Час за який сенсор генерує та приймає пучок ультразвуку складає 2912 мкс (рис. 4.6).

Для цього резервуару дистанція буде розраховуватися наступним чином:

$$v_{sos} = 343.42\ {}^{\text{м}}/_{\text{с}} = 0.034\ {}^{\text{см}}/_{\text{мкс}} \quad (4.3.)$$

$$s = \frac{t * v_{sos}}{2} = \frac{2912\ \text{мкс} * 0.034\ {}^{\text{см}}/_{\text{мкс}}}{2} = 49.504\ \text{см} \sim 50\ \text{см} \quad (4.4.)$$

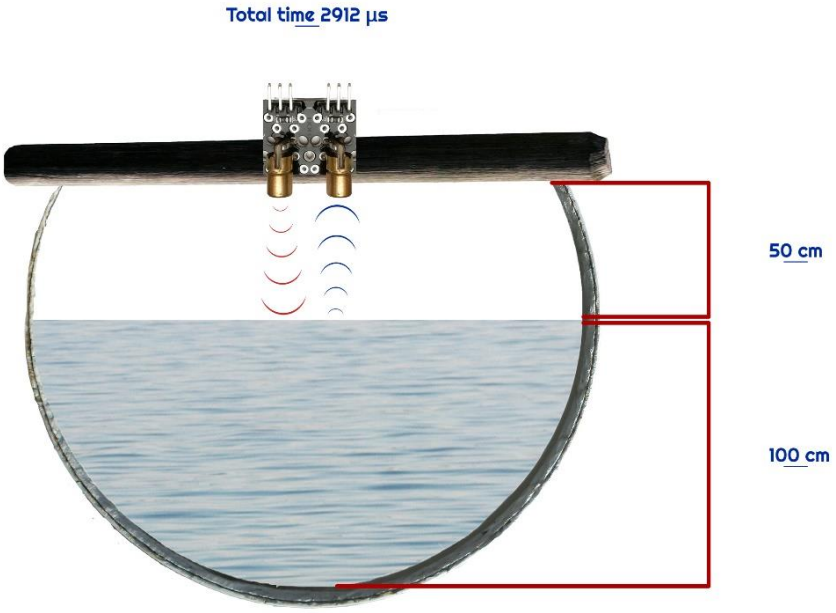


Рисунок 4.6 – Приклад визначення рівня рідини за допомогою ультразвукового сенсору

4.3. Кондуктивний сенсор рівня рідини

У якості кондуктивного сенсора, візьмемо неофіційну розробку [22], проте яка точно відображає принципи роботи реальних кондуктивних сенсорів.

Загальний принцип роботи даного кондуктивного сенсору полягає у вимірюванні електричної провідності між двома або більш електродами, наприклад, мідними доріжками на платі. Чим більша площа контакту електродів з водою, тим вища провідність, або нижчий опір між ними. Даний сенсор необхідно занурити у воду, де він вимірює електропровідність.

Якщо розглядати основні виводи даного сенсору, то він має:

- «S» – аналоговий вихідний контакт, що використовується для з’єднання із мікроконтролером – Arduino Nano ESP32;

- «-» – підключення до землі;

- «+» – контакт живлення, що використовується для роботи сенсора. Рекомендується під’єднати даний контакт до джерела живлення від 3.3В до 5В;

- «TestPin» – тестовий контакт, якого нема у реальному пристрої. Він допомагає симулювати роботу сенсору, його занурення у віртуальну «воду».

Сенсор має також десять відкритих мідних доріжок, одна половина з яких це сенсорні доріжки, інша половина це доріжки живлення.

Більш детально, робота даного сенсору полягає у наступному: виведені назовні паралельні доріжки працюють як змінний резистор, опір якого залежить від рівня води. Зазначимо, що опір сенсора обернено пропорційний рівню води. Коли сенсор повністю занурений, він вказує на низький опір, що вказує на велику висоту води. А коли сенсор частково занурений, він показує більший опір і меншу провідність, тим самим вказуючи меншу висоту відповідно до опору. Цей змінний опір безпосередньо пов’язаний з напругою на сенсорі. Вимірюючи цю напругу, ми можемо визначити рівень води.

Для того, щоб імітувати динамічний рівень води, до тестового входу (відповідальний за моделювання рівня води і існує лише у віртуальному

середовищі) під’єднаємо змінний резистор. Це дозволить нам маніпулювати електричним опором, як би сенсор занурювався у воду.

Якщо опір близький до нуля, тобто сенсор повністю занурений у воду, і має великий рівень висоти води над собою, то відповідно буде максимальна напруга, яку можна отримати на вольтметрі або через Arduino. Відповідно, вірно і те, що якщо збільшити опір, тобто імітувати що сенсор частково занурений у воду, то напруга буде зменшена. Таким чином, можна аналізувати рівень води у резервуарі.

Для підключення даного сенсору до Arduino, у віртуальній симуляції необхідно підключити сенсор до LC-ланцюгу (рис. 4.7). Ідея у тому, що програмне забезпечення що ми використовуємо, надає значення повного розмаху сигналу сенсора, а нам потрібно перетворити його на середньоквадратичну напругу (Vrms). Проте, якщо перенести проект у реальний світ, даний ланцюг необхідно прибрати.

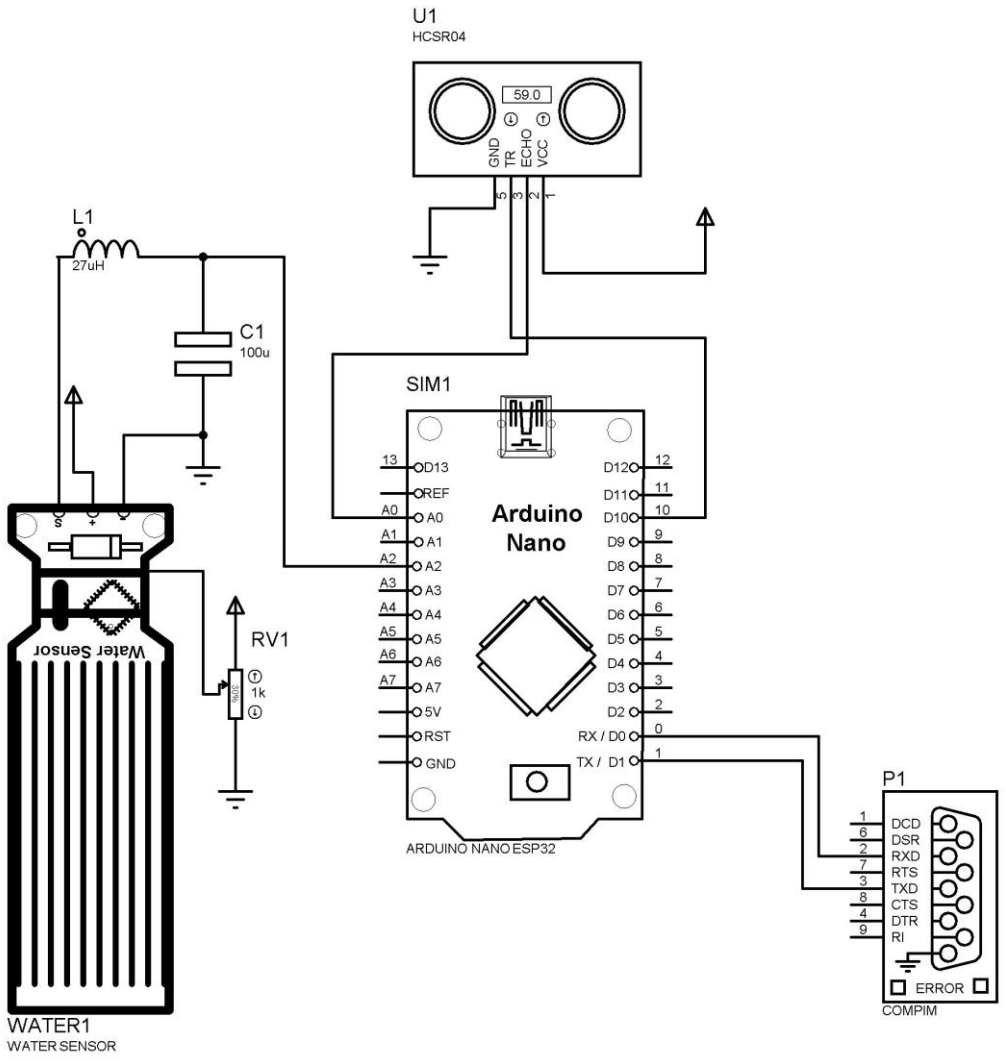


Рисунок 4.7 – Підключення кондуктивного сенсора до *Arduino Nano ESP32*

Повний варіант схеми наведено у додатку В.

4.4. Сенсор температури рідини

Для даного проекту, візьмемо температурний сенсор DS18B20 (водостійкий) (рис. 4.8). Даний цифровий температурний сенсор являє собою металеву капсулу довжиною 40-50мм, з внутрішньою заливкою епоксидною смолою та з'єднану з дротом.

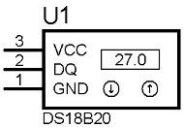


Рисунок 4.8 – Температурний сенсор *DS18B20*

Сенсор використовує інтерфейс «One-Wire» (таблиця 4.2), у якому дані та живлення можуть йти по одному провіднику. Даний сенсор може працювати у великому діапазоні температур, що повністю покриває вимоги теплиць.

Таблиця 4.2 – Загальна характеристика сенсору температури *DS18B20*

Параметр	**Значення**
Діапазон температур	-55 C° – +125 °C
Точність	±0.5 C°
Напруга живлення	3.0 – 5.0 В
Інтерфейс	One-Wire
Захист від води	IP68 [10]

Також, дуже важливо зазначити, що використовуючи інтерфейс «One-Wire, це надає можливість взаємодіяти з єдиною загальною шиною, до якої може підключитися безліч сенсорів. У кожного температурного сенсору є власний ROM-код, що можна використовувати як унікальний Id або адресу підключення сенсору. Всі сенсори підключені за рахунок даного інтерфейсу до одного виводу, у нашому випадку до D3, що дозволяє економити значну кількість виводів пристрою.

Для роботи с даним сенсором э готові бібліотеки для Arudino: OneWire та DallasTemperature. Вони дозволяють створювати об’єкти з необхідним функціоналом, для пошуку сенсорів, запиту до їх даних тощо.

Даний сенсор, для коректної роботи потребує додаткового підтягуючого резистора, що дозволяє використовувати дроти до 25 метрів і отримувати дані без спотворення. Сенсор має середній рівень вартості, проте це найдешевший варіант серед прямих конкурентів: NTC-термістори (складний, погана водостійкість), TMP36 (не водостійкий), PT100 (висока вартість). Проте, заявлена життєздатність (близько 10 років) повністю виправдовує і окупає дану вартість.

4.5. Програмна реалізація функціональних модулів пристрою

Прилад повинен реалізовувати функції налаштування підключення і сенсорів, збирання даних, та відправку їх до серверу (в нашому випадку через COM-порт до програми, яка буде відправляти їх до серверу тощо).

Загалом, роботу самого пристрою можна подивитись на додатку Д.

На початку, необхідно увімкнути пристрій. Після цього почнеться процес первинного налаштування, а саме встановлення швидкості передачі даних через послідовний порт (baud rate) та пошук сенсорів температури. Зазначимо, що baud rate повинен бути встановлений однаково на обох пристроях - відправнику (Arduino) та приймачі (програма для COMPIM, сервер тощо). В іншому випадку, дані будуть спотворені і коректний аналіз и демонстрація на графічному інтерфейсі буде неможливі. Далі почнеться робочий цикл програми.

У даному циклі, кожен раз перевіряється чи є нові дані з налаштуваннями від серверу. Для цього викликається функція «checkNewSettings()». Логіка даної функції полягає у наступному: необхідно перевірити, чи є якась інформація у буфері обміну. Буфер обмежений 64 байтами і дані можуть надходити уривками. Щоб уникнути цього, було вирішено використати ітераційний, ручний підхід із збереженням кожного байту інформації, з послідовним об’єднанням їх у тип даних String.

Таким чином, якщо якісь дані очікують у буфері, то їх необхідно опрацювати і зберегти у масив спеціальних структур, розроблених спеціально для даного приладу, і які дозволяють відмежувати налаштування одного сенсору від іншого.

Після цього, або якщо буфер пустий (ніяких налаштувань немає), починається процес зчитування даних з сенсорів – викликається функція «getAndSendMeasurements()». Дана функція працює наступним чином, вона проходиться по масиву сенсорів, і для тих сенсорів підключення яких налаштовано, в залежності від типу сенсору викликається відповідна функція.

Якщо це ультразвуковий сенсор, то викликається функція «getSonicData()», яка оброблює дані згідно формулам наведеними вище. Якщо поточний сенсор кондуктивного типу, то викликається функція «getConductiveData()», яка розраховує ступінь занурення в залежності від напруги, що зчитується з аналогового виводу, та враховуючі вплив температури рідини на отриману напругу. Функція по вимірянню температури має велике значення для точності вимірювання рівня рідини. Вона викликається кожного разу, безпосередньо перед розрахунком рівня рідини для кожного сенсору. Незважаючи на те, що дані про температуру не йдуть до сервера, вони застосовуються у формулах розрахунків, при опитуванні сенсорів.

Після цього дані відправляються на сервер і логіка програми для пристрою також досягає кінця. Якщо користувач вимкне пристрій, то його робота припиниться. Якщо пристрій буде продовжувати працювати, то він знов перейде до початку, на етап де виконується перевірка на наявність нових даних, спробує зчитати дані з серверу тощо. Таким чином, робота буде продовжена до того моменту, коли пристрій не буде вимкнено

РОЗДІЛ 5. ПРОЕКТУВАННЯ АРХІТЕКТУРИ СЕРВЕРНОЇ ТА КЛІЄНТСЬКОЇ ЧАСТИН СИСТЕМИ

IoT-система контролю рівня рідини в системах опалення тепличного господарства з віддаленим доступом і дистанційним налаштуванням сенсорів, буде складатися з компонентів наведених на рисунку 5.1.

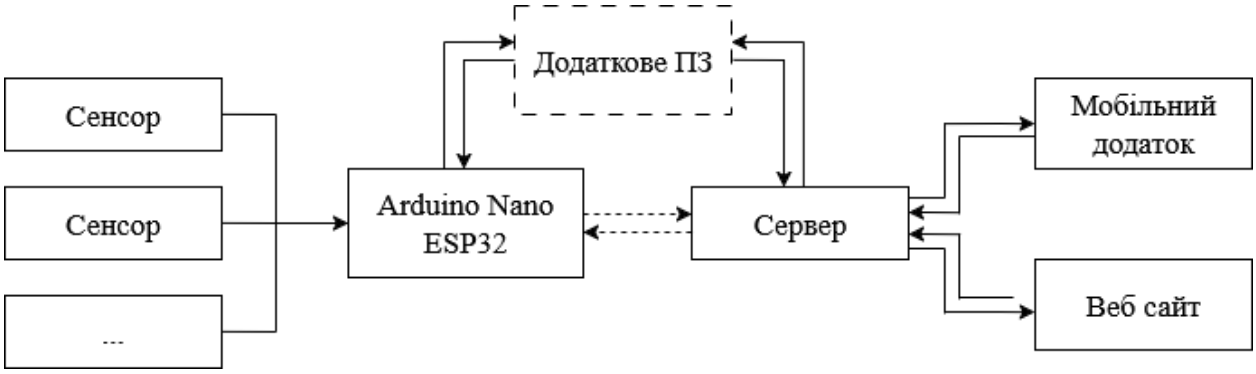


Рисунок 5.1 – Загальна схема системи у вигляді логічних компонентів

Сенсори отримують інформацію із середовища та передають її до Arduino Nano ESP32. Arduino налаштовує параметри збори даних з сенсорів, збирає дані від них пакує їх у JSON та відправляє їх до серверу. На сервері вони зберігаються у базу даних. Далі, дані відправляються за запитом до мобільного додатку та/або веб-сайту. Налаштування підключення сенсорів надходить з сервера, на який потрапляє з веб-сайту або мобільного додатку. Таким чином, забезпечується ефективна взаємодія між компонентами пристрою та сервером, мобільним додатком і веб-сайтом.

Окремо потрібно вказати, що для роботи системи у програмному середовищі, потрібно додаткове ПЗ, щоб забезпечити взаємодію пристрою із сервером: зчитування даних з COM-порту і відправку на сервер. При переносі проекту у реальний світ, це ПЗ не потрібно, і взаємодія йде напряму через Wi-Fi модуль.

5.1. Проектування серверної частини системи

У якості серверної частини, будемо використовувати Firebase. Firebase – це BaaS (Backend-as-a-Service), що поставляється і розроблюється Google. BaaS

дозволяє отримати готову серверну інфраструктуру, багато API, а також налаштованими функціями авторизації, реєстрації та вбудованими різноманітними базами даних.

Принцип роботи Firebase полягає у декомпозиції звичайного backend на незалежні мікро-сервіси, кожен з яких вирішує певне завдання за допомогою спеціалізованого API. Наприклад, сервіс автентифікації, виконаний як окремий сервіс ідентифікації, використовує протоколи OAuth 2.0 та OpenID Connect. Або, наприклад, система миттєвих обновлень: вона працює за моделлю моніторингу через постійні підключення. Будь які зміни передаються через хмарну систему розсилання повідомлень. А у свою чергу хостингова інфраструктура побудована на глобальній CDN-мережі з автоматичним шифруванням з'єднань і кешуванням до найближчого до користувача серверу. Це забезпечує мінімальні затримки з'єднання.

Таким чином, Firebase приховує від розробника складність інфраструктури і інфраструктурних задач (автоматичне збільшення потужності, оптимальний розподіл навантаження, оновлення системи захисту тощо), що в свою чергу дозволяє займатися лише клієнтською частиною, а серверну частину перекласти на сервіс. Також необхідно зазначити, що майже усі необхідні аспекти можуть гнучко налаштовуватися через конфігураційну панель Firebase.

Використання даного рішення замість повноцінної розробки сервера полягає у наступному:

- швидкий розвиток прототипу – виключення необхідності писати backend на перших етапах розвитку проекту;

- готова, вбудована Realtime Database – NoSQL база даних з синхронізацією у реальному часі між пристроями та користувачами;

- облегшена інтеграція з Web та мобільними додатками – є готові SDK для Web (JavaScript) та Android (Kotlin). Також є підтримка автентифікації, а саме за допомогою email/password, Google accounts тощо;

- Firebase Hosting – готові інструменти для хостингу веб-сайту;

- багато REST API для взаємодії із сервером;

- низька вартість – Firebase надає можливість безкоштовно використовувати більшість сервісів, до певного ліміту. Наприклад, для веб-сайту надається до 1 ГБ місця для зберігання даних, підтримка 100 одночасних підключень, 10 ГБ вихідного трафіку у місяць, до 50 тис. записів у день тощо. При виході за цей ліміт, можна або купити повноцінний пакет хостингу, який значно розширює ліміти та можливості, або виконати розрахунок тих компонентів, що вийшли за ліміт і оплачувати лише ці частини, в залежності від того, наскільки ліміт було перевищено.

Серед недоліків Google Firebase відносно даної системи, можна виділити наступне:

- обробка інформації* – вона буде проходити на веб-сайті, а не на серверній частині. Наприклад, за бізнес-логіку відносно відправки повідомлення про небезпечний рівень рідини буде відповідати код на веб-сайті, а не сервер;

- відсутність вбудованих інструментів обробки даних* – неможливо автоматично оброблювати дані при їх надходженні до Firebase за замовчуванням. Ми зберігаємо їх і потім надсилаємо за запитом на веб-сайт або додаток;

- залежність від Google – при закритті платформи, зміни умов безплатного користування або критичній зміні плану, відключенні важливих для нашої системи API – буде неможливо перейти на інший сервер без повного переписування, краще сказати створення наново, коду для backend.

* потрібно зауважити, що писати власний backend для Firebase можливо за допомогою Firebase Cloud Functions. Код можливо писати на різних мовах програмування, наприклад, JavaScript та Node.js і потім інтегрувати його з Firebase. Важливий сам факт відсутності за замовчуванням вказаних вище інструментів обробки.

Для даної системи будуть використовуватися наступні компоненти Firebase:

- Realtime Database;

- Cloud Firestore;

- Firebase Authentication;

- Firebase Security Rules;

Потрібно одразу надати коментар стосовно використанню різних видів баз даних у проекту. Справа у тому, що у Realtime Database ми будемо зберігати основні дані та дані що надходять до сенсорів. У Cloud Firestore виконується запис історії вимірювання.

Логіка обрання Realtime DB для поточних даних і статичних налаштувань у тому, що дана БД, це JSON-дерево. В ньому дуже дешеві записи, висока швидкість взаємодії, простота використання. Проте з недоліків можна виділити те, що в ній відсутні складні запити (наприклад, неможливо робити агрегацію даних) і при великій кількості вузлів і даних падає продуктивність [6]. Для поточних даних це повністю підходить, проте для зберігання великих масивів даних краще обрати інший варіант – Cloud Firestore [5].

Окрім цього, Realtime Database цікава тим, що підтримує оновлення даних в режимі реального часу. Будь-які зміни в даних одразу відображаються на всіх підключених клієнтах (Android додаток та веб-сайт), і таким чином, при зміні даних на сенсорах, інформація збережеться у базі даних і одразу ж сформує тригер до оновлення даних у додатку та веб-сайті.

Cloud Firestore, з іншого боку, дозволяє зберігати середні об’єми даних, в неї є певна структура, але без запитів і фільтрації, що підходить для нашої задачі – зберігання даних.

Для автентифікації користувачів у даній системі використовується Firebase Authentication – хмарний сервіс автентифікації від Google. Він дозволяє користувачам реєструватися по електронній пошті і паролю (із листом підтвердженням і без), а також через облікові записи Google, Facebook, Apple, GitHub тощо. Кожному користувачу автоматично генерується унікальний UID, який буде використовуватися для структурування інформації у базі даних [4].

Для всієї системи можна налаштувати правила, за якими надати користувачам певні права, наприклад, тільки зчитувати дані, або і вносити зміни.

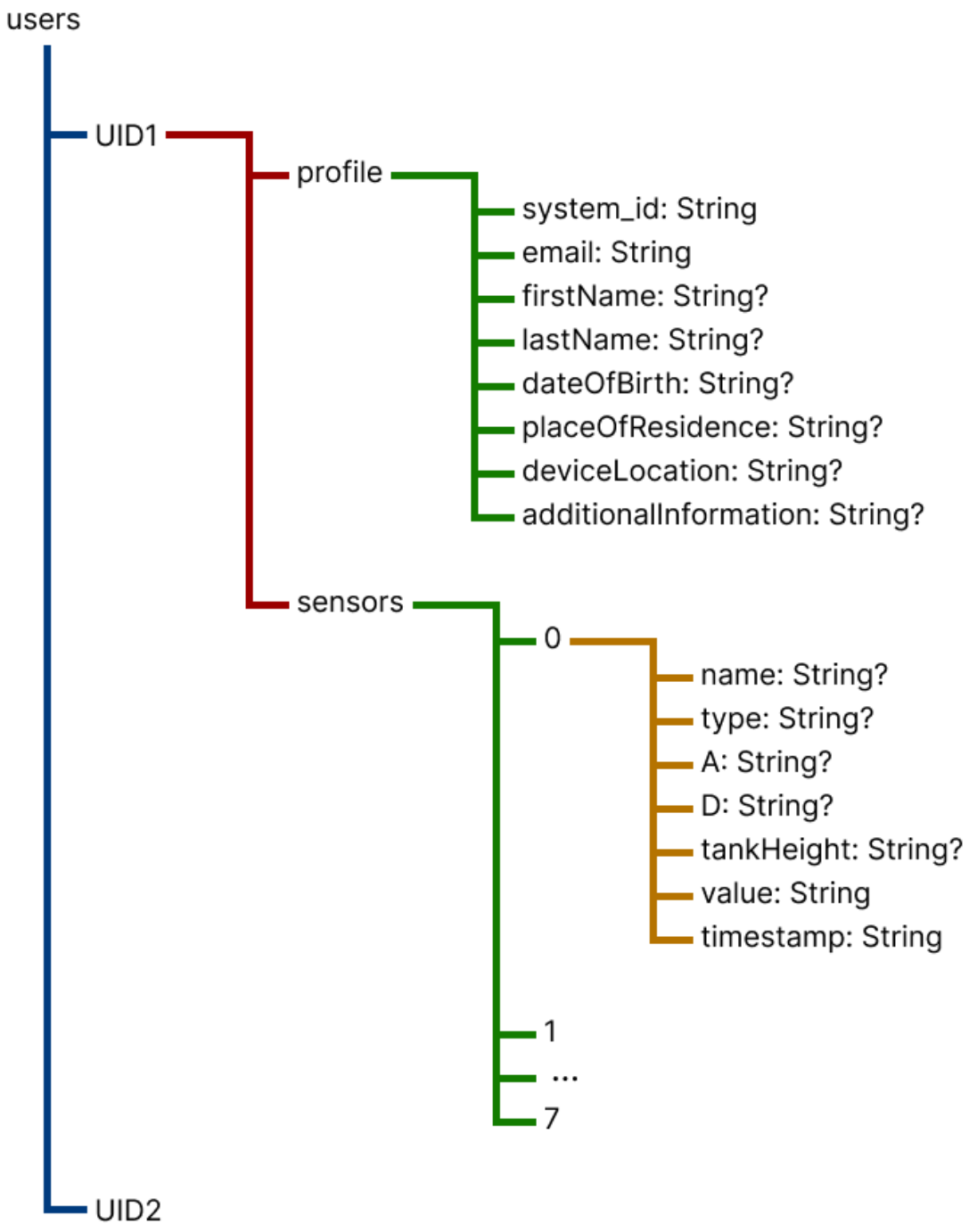


Рисунок 5.2 – Макет бази даних *Firebase Realtime Database*

Розглянемо загальну структуру бази даних Realtime Database (рис. 5.2). Пояснення щодо синтаксису: даний макет є спрощеним графічним зображенням і використовує загальні правила написання змінних у мові програмування Kotlin. У даному випадку:

- fieldName: String – поле обов’язкове для заповнення, додається у вигляді строкового типу даних;

- fieldName: String? – поле необов’язкове для заповнення, додається у вигляді строкового типу даних або null;

За даним макетом, є загальна категорія «users», куди будуть включатися усі користувачі. При реєстрації, кожному користувачу, буде присвоєно власний UID, за яким він і буде додаватися у базу даних. Потім, база даних для кожного користувача поділяється на два розділу: «profile» та «sensors». У «profile» буде знаходитися інформація відносно унікального номеру пристрою, email користувача, та необов’язкові додаткові дані, які користувач може заповнювати за власним бажанням. Ці дані можуть знадобитися з часом, при розвитку проекту, для надання більш якісної допомоги, особливих пропозицій і персоналізованої реклами.

У розділі sensors знаходиться інформація по кожному з доступних сенсорів від «0» до «7». Для кожного з сенсорів, необхідно надати додаткові параметри, як обов'язкові, так і необов'язкові. Наприклад, поле «name» не є обов’язковим, користувач може дати ім’я сенсору, щоб йому було легше орієнтуватися. Поля «type», «A» є обов’язковими для обох типів сенсорів рівня («A» – це номер аналогового порту). Поля «D» та «tankHeight» необхідні для ультразвукового сенсора («D» – цифровий порт, «tankHeight» – висота резервуару). В свою чергу, поля «value» – значення від сенсора, і «timestamp» – час вимірювання, не залежать від користувача і надходить від пристрою.

Також наведемо правила безпеки що застовповані до бази даних (рис. 5.3.). У даних правилах, ми дозволяємо користувачу змінювати і читати дані, що відносяться до нього, тобто до його UID. Користувач може взаємодіяти з базою даних лише через інтерфейс веб-сайту або мобільний додаток, тому можливо надати досить вільні правила. Взаємодія з «value» та «timestamp» налаштована таким чином, що запис до цих полів виконується лише пристроєм, і в одному випадку через веб-сайт або додаток – якщо сенсор видаляється, тобто значення встановлюються на нуль.

При подальшій модернізації системи, є практична необхідність орендувати віддалений сервер і розроблювати власну інфраструктуру.

```
{
  "rules": {
    "users": {
      "$uid": {
        ".read": "auth != null && auth.uid == $uid",
        ".write": "auth != null && auth.uid == $uid",
        "profile": {
          ".read": "auth != null && auth.uid == $uid",
          ".write": "auth != null && auth.uid == $uid"
        },
        "sensors": {
          "$pin": {
            ".read": "auth != null && auth.uid == $uid",
            ".write": "auth != null && auth.uid == $uid"
          }
        }
      }
    }
  }
}
```

Рисунок 5.3 – Правила безпеки для *Firebase Realtime Database*

5.2. Проектування web-частини системи

Web частина представляє собою клієнтський додаток, розробка якого виконується за допомогою класичного стеку HTML+CSS+JavaScript і призначена для взаємодії користувача з пристроєм на базі Arduino Nano ESP32, через хмарний сервер Firebase і базою даних (Firebase Realtime Database, Firebase Cloud Firestore). Архітектура веб-інтерфейсу має бути побудована за асинхронною подіє-орієнтованою моделлю, забезпечуючи реактивність і можливість масштабування без необхідності розробки або використання складних серверних компонентів [19, с.36].

Для реалізації буде використовуватися стек з HTML, CSS, JavaScript. Даний класичний стек є базою для більшості веб-сайтів. Його використання забезпечує мінімальну вагу кодової бази і проекту у цілому, високу продуктивність, відсутність зайвих залежностей, та повний контроль над кодовою базою.

С урахуванням мети проекту та його масштабу, використання важких бібліотек на кшталт React, фреймворків типу Angular, Vue, або утилітарних CSS-систем, наприклад, Tailwind, не дають значних переваг, але збільшують складність розробки.

Таким чином, даний класичний підхід, дозволяє швидше досягнути цілі та підтримувати чистоту коду без значної втрати в якості кінцевого продукту [13, с.34].

Визначимо основні цілі проекту, принципи та функції що буде виконувати веб-сайт:

- відображення поточного рівня рідини;

- відображення рівня рідини за певний період часу;

- налаштування сенсорів, а саме вибір типу сенсора, визначення виводів у які будуть підключатися сенсори;

- повідомлення – звукові і візуальні сигналі при досягнені критичного значення рівня рідини;

- користувацька ізоляція – користувач бачить лише дані за своїми сенсорами;

- реактивність – дані оновлюється у реальному часі;

- адаптивність на різних розмірах екрану – сайт повинен правильно відображатися у мобільних браузерах, планшетах та класичних моніторах.

Звернемо увагу на алгоритм дій при отриманні з серверу даних і проаналізованих як небезпечний рівень рідини (рис. 5.4, рис. 5.5). Після успішної авторизації користувача, при увімкненому пристрої починається запис даних до серверу і відстеження зміни даних у базі даних на веб-сайті [1, с. 328]. Якщо користувач не авторизувався, то до основної програми він не дійде і перенесений до початку – авторизації або реєстрації. Веб-сайт отримує дані, оброблює їх по заданій нами логікою і виконує додаткову перевірку, наприклад, чи отримані значення більше 20% та менше 80%? Дані значення обираються користувачем, в залежності від потреб. Якщо умова виконується, то за нашою логікою все гаразд і можна переходити до наступного кроку – умови виходу. Під умовою виходу, розуміється вихід користувача з облікового запису або закриття веб-сайту. Якщо умова не виконується, то необхідно попередити, що для того сенсора, для якого не виконалася умова, досягнений критичний рівень рідини, і потрібно вжити певних заходів. Буде змінений на червоний загальний background для сайту, і елементи відповідальні за демонстрацію роботи сенсора також будуть змінені на червоні

кольори [16, с. 76]. Також почнеться програвання аудіо-сигналів, щоб звернути увагу користувача на поточну проблему.

Після виконання даних дій, якщо користувач виходить з сайту, виконання програми завершується, у іншому випадку все починається знов з моменту отримання нових даних з серверу.

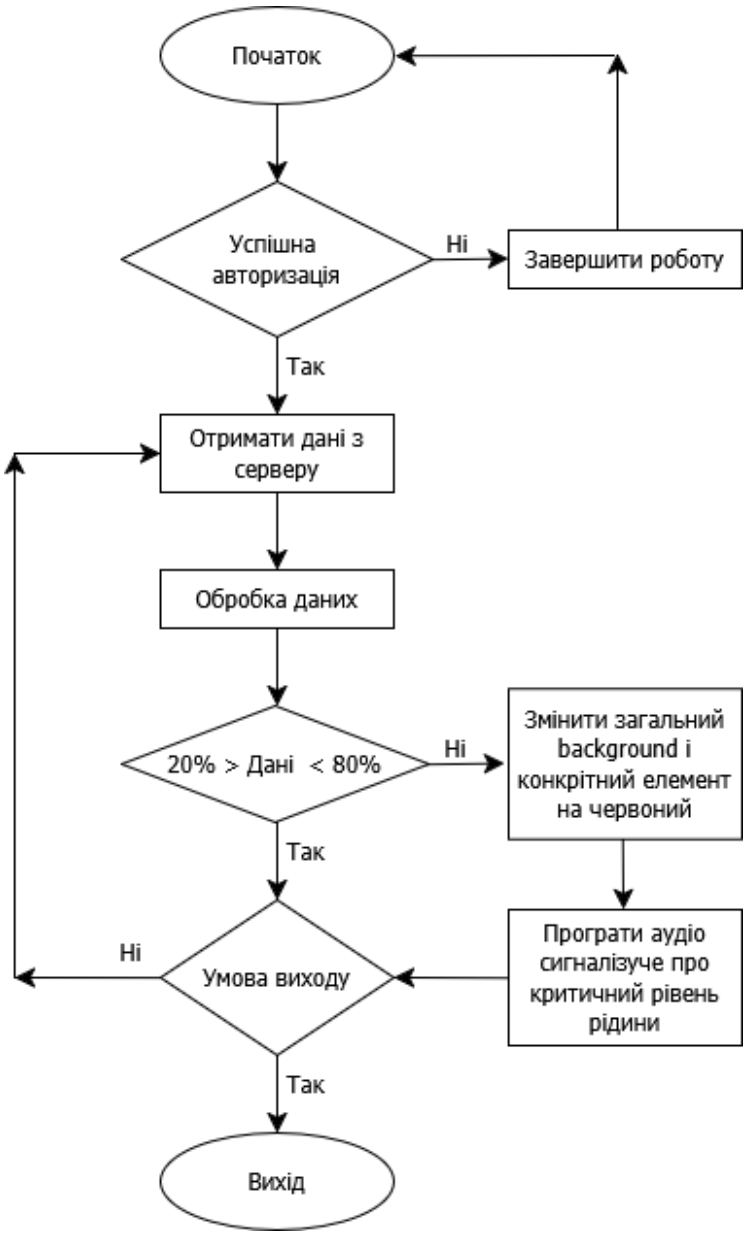


Рисунок 5.4 – Блок-схема поведінки *IoT*-системи при визначенні небезпечного рівня рідини

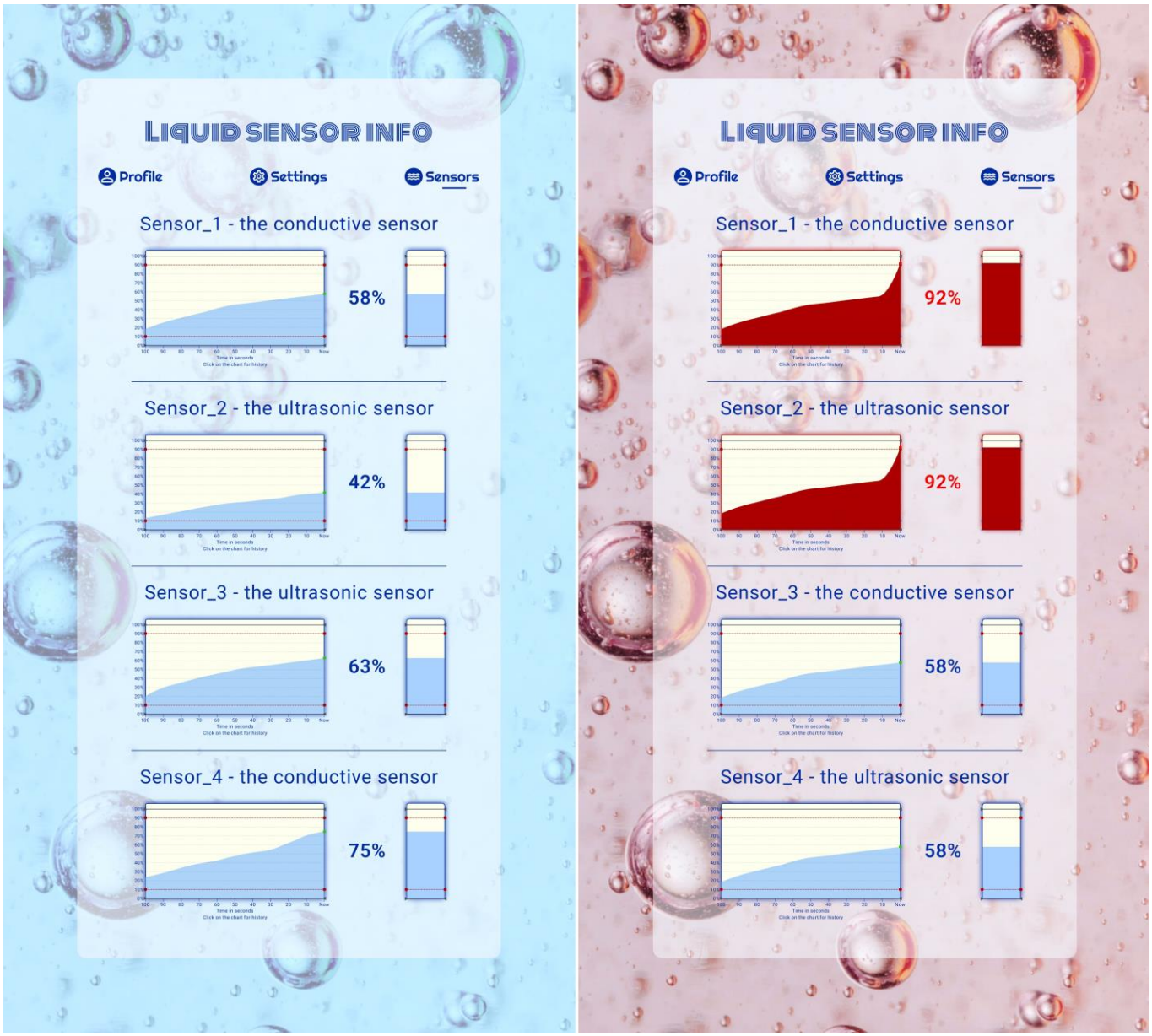


Рисунок 5.5 – Графічне зображення нормального та небезпечного рівня рідини

Дана система підтримує два типу сенсорів вимірювання рівня рідини, а також необхідний для роботи температурний сенсор. Користувачу необхідно налаштувати схему підключення даних сенсорів. Дане налаштування виконується на сторінці «Settings» на веб-сайті або мобільному додатку (рис.5.6).

Логіка вибору у тому, що якщо обрати сенсор зліва, то це буде кондуктивний сенсор, якщо з обох боків, то це ультразвуковий. Температурний сенсор повинен повторювати розташування сенсора рівня рідини, з яким він працює, проте на своєму інтерфейсі. Загалом можна підключати до 8 сенсорів. Після підключення, нижче на сторінці буде відображатися який тип сенсора був підключено та його порядковий номер. Якщо певний номер вже підключено, то він буде виділений

сірим кольором. Для кожного сенсора є можливість додатки опис: де він знаходиться, у якому резервуарі тощо.

Рисунок 5.6 – Налаштування підключення сенсорів

У даному проекті також необхідно застосовувати модульну архітектуру. Кожен модуль, це окремий JS файл, який відповідає за певний функціонал. Це дозволяє розмежувати відповідальність блоків коду, наприклад, замість багатьох сотень стрічок коду, де складно орієнтуватися, виділимо певний функціонал і перенесемо його у окремий файл. Наступним кроком, імпортуємо даний файл у основний скрипт.

Для збірки проекту будемо використовувати WebPack. У загальному вигляді логіка WebPack полягає у наступному: він бере «основний модуль» який планується підключати до сторінки, аналізує усі залежності (імпорти, імпорти імпортів тощо) і збирає їх у один файл з усіма модулями.

У процесі трансформації також виконуються додаткова оптимізація коду:

- код що неможливо досягти – сміття – буде видалений;

- видалення експортів що не використовуються;

- специфічні оператори, що використовуються при розробці, наприклад: console, debugger, також видаляються.

Оскільки у даному сайті буде декілька сторінок (авторизація/реєстрація та основна частина), необхідно налаштувати WebPack на створення і збірку декількох файлів. Файл з конфігураціями наведений на рисунку 5.7. Дамо пояснення основним моментам, на які потрібно звернути увагу:

- зміна path – це необхідна технічна зміна;

- module.exports – це об’єкт налаштувань зборки;

- mode – режим у якому буде працювати Webpack. Якщо проект знаходиться на стадії розробки, бажано ставити режим «development» – це дозволяє проекту працювати швидше, проте не виконує автоматичну оптимізацію коду;

- entry (entry point) – це той файл, у випадку багатосторінкового сайту – ті файли, по яким буде створений граф залежностей. Тобто, є центральний файл (entry point), у якому знаходиться багато imports на внутрішні та зовнішні «.js» файли. За цими залежностями буде побудований новий файл;

- output – файл виходу, куди буде згенеровано увесь необхідний код з файлу входу. Для багатосторінкового сайту, ми вказуємо що є параметр «[name]», який динамічно змінюється в залежності від імен файлів входу;

- watch – встановлення прапору, що дозволяє Webpack збирати проект автоматично при кожній зміні (при кожному збереженні файлу);

- plugins – підключення plugins, у нашому випадку HtmlWebpackPlugin.

Даний plugin дозволяє автоматично підключати створені «.js» скрипти у html файли. Це особливо важливо, коли даний проект перейду у production. У даному режимі, Webpack буде створювати хешовані імена файлів, таким чином, власноруч додавати дані скрити до сторінки буде проблематично. Проте, завдяки даному плагіну, це робиться автоматично.

```
JS webpack.config.js > ...
'use strict';

let path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {
  mode: 'development',
  entry: {
   auth: './js/auth.js',
   main: './js/main.js'
  },
  output: {
    filename: '[name].bundle.js',
    path: __dirname + '/js'
  },
  watch: true,

  devtool: "source-map",

  plugins: [
    new HtmlWebpackPlugin({
      template: './src/index.html',
      filename: 'index.html',
      chunks: ['index']
    }),
    new HtmlWebpackPlugin({
      template: './src/main.html',
      filename: 'main.html',
      chunks: ['main']
    })
  ],
};
```

Рисунок 5.7 – Файл с конфігураціями проекту для *Webpack*

5.3. Проектування мобільної частини системи

Мобільний додаток розроблюється для пристроїв (smartphones, tablets, тощо) з операційною системою Android. Для розробки додатка буде використовуватися мова програмування Kotlin.

Вибір розробки IoT-системи виключно під ОС Android, було зроблено з урахуванням ряду технічних, економічних і експлуатаційних факторів, які роблять дану операційну систему найбільш раціональним вибором для досягнення мети роботи.

Кратко дамо опис факторів, через які була обрана дана ОС.

Був розглянутий профіль цільової аудиторії, користувацьке обладнання і бюджет організації. Використання в Україні даної ОС є переважним, окрім цього, використання iOS у сільському господарстві по всьому світі мінімальне і являє собою виключення з правил. Напроти, смартфони на ОС Android, враховуючі їх низьку вартість і широку доступність – це популярне рішення для даної сфери.

Також, були розглянуті технічні особливості ОС у сукупності із можливими модернізаціями IoT системи в майбутньому. Класична ОС Android надає більш гнучкий доступ до різних функцій смартфона (менеджер сповіщень, Bluetooth, USB-OTG, Serial тощо), ніж кросплатформне рішення.

Даний мобільний додаток повинен повторювати усі необхідні для коректної роботи елементи, що й описаний вище веб-сайт. Також, і веб-сайт, і мобільний додаток повинен мати спільний дизайн. У даному розділі ми сконцентруємося на технічних аспектах розробки моделі додатку.

Firebase підтримує Android, iOS, Web, і надає усі необхідні API для взаємодії. Таким чином, Android додаток буде працювати на мобільному пристрої і зв’язуватися через інтернет з Firebase для отримання даних.

Мобільний додаток повинен працювати на операційній системі Android не нижче версії 7.0 (Nougat, API 24). Загальна структура додатку повинна складатися з однієї вхідної точки («main») і логічно пов’язаних екранів, що виходять з цієї точки, на яких буде створено повноцінну взаємодії з користувачем і надання йому детальної інформації про рівень рідини.

Для даного додатку була обрана модель архітектури «Single activity». Тобто, як було сказано вище, буде створена одна Activity, що грає роль «entry point», і багато Fragments [8, с.76].

Activity – це основний блок, на якому буде строїтися увесь додаток. Саме ця Activity буде відповідальна за підключення та відображення графічного дизайну, налаштування загальної логіки і надання основного функціоналу.

Fragment – це модульна частина, яка може жити незалежно від Activity, але розміщуватися і відображатися може лише на Activity. Для даного типу архітектури кількість Fragments не обмежена.

Використання багатьох Activity – це затратно за ресурсами, ефективністю та швидкодією. Напроти, використання багатьох легких Fragment, дозволяє створити гнучку, модульну структуру, з можливістю багаторазового використання однакових, але незалежних (різні екземпляри) фрагментів у різних частинах додатку, та використовувати різні макети для різних розмірів екрану.

Таким чином, розглянуту архітектуру відносно даного проекту можна представити наступним чином (рис. 5.8): є загальна Activity – MainActivity, яка є контейнером для фрагментів. Кожен фрагмент виконує свою роль і відображається за необхідністю. Якщо фрагмент необхідно змінити, то він або видаляється зі стеку, або переходить у неактивний стан, дозволяє іншому фрагменті заповнити контейнер.

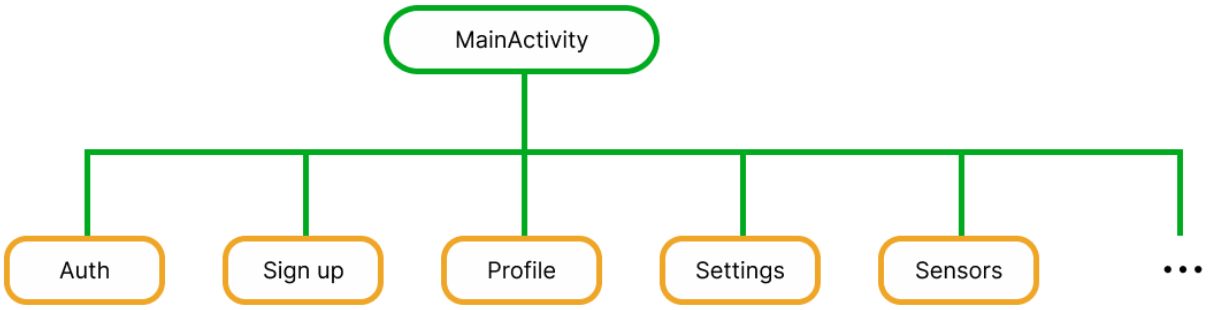


Рисунок 5.8 – Приклад використання архітектури *Single Activity* (*Activity + Fragments*) у проекті

Щоб реалізувати можливість забезпечення доступу до даних між фрагментами і у додатку у цілому, раціонально використовувати підхід із застосуванням ViewModel.

ViewModel – це архітектурний компонент для Android з бібліотек Jetpack, призначений для зберігання і керування даними UI з урахуванням життєвого циклу. Головна задача ViewModel, надати Activity або Fragment можливість володіти даними, які не знищується при повороті екрану, перестворенню фрагменту, або зміні конфігурацій [11, с.134]. Даний компонент дозволяє:

- зберігати дані UI і інші дані, що необхідно захистити від знищення, але немає сенсу заносити у локальну базу даних;

- відділяти бізнес-логіку від інтерфейсу;

- ділитися даними між фрагментами або Activities.

Також важливо зазначити, що можна використовувати підхід з ViewModel для кожного Fragment. Це ускладнює процес передачі даних між фрагментами та збільшує навантаження на систему, проте дозволяє зробити код чистіше за рахунок відділення різноманітних функцій, що відносяться до певних фрагментів, від іншого

коду. Проте, існує також інший підхід, який і буде використовуватися у даному додатку – одна ViewModel створена для Activity у єдиному екземплярі, і до якої доступ будуть отримувати фрагменти на підставі батьківських залежностей (кожен fragment, це child для activity). Таким чином, ми маємо спільну ViewModel, яка працює для усіх компонентів (рис. 5.9).

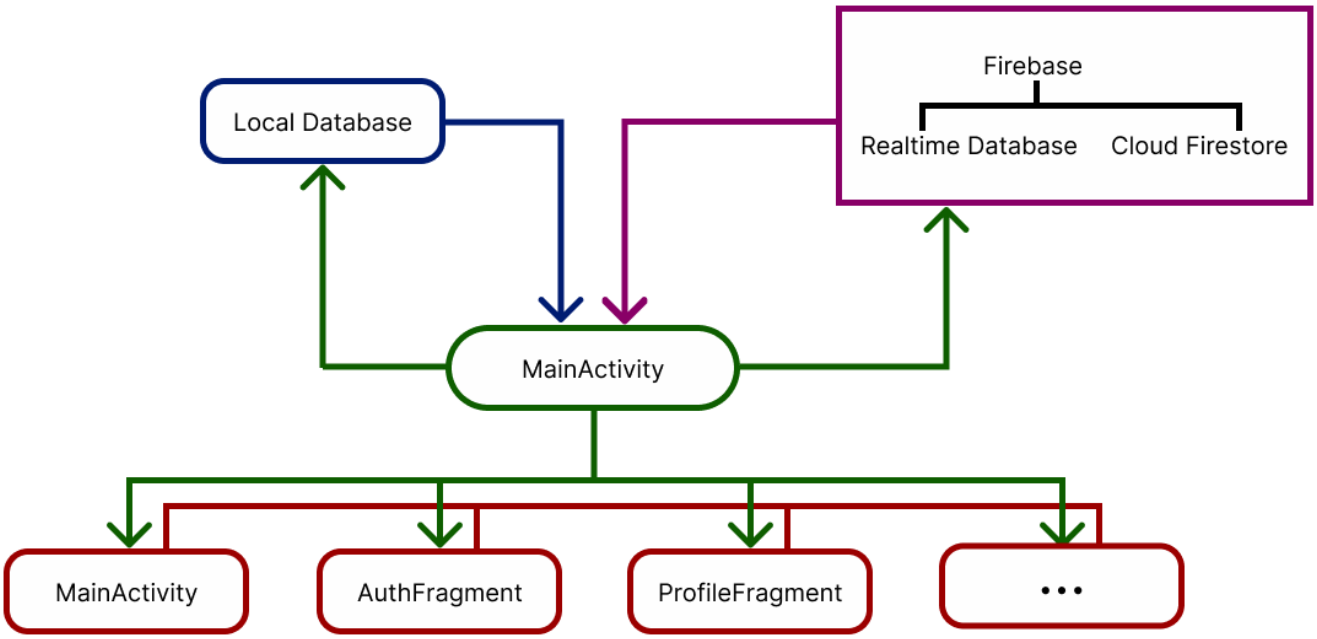


Рисунок 5.9 – Взаємодія основних елементів додатку з *ViewModel*

Також зазначимо, що у дану ViewModel дані надходять не тільки від фрагментів, але й від локальної бази даних (SQLite) і від Firebase.

В основному, взаємодія між користувачем і сервером буде односторонньою. Користувач, коли необхідно відправляє нові налаштування, проте переважну частину часу він отримує дані з пристрою і серверу.

Як вже було наведено вище, Firebase Realtime Database являє собою великий JSON файл. При запиті до серверу, наприклад, до Firebase Realtime Database, при успішній авторизації користувача ми отримаємо ділянку JSON файлу, що відноситься до користувача. Далі можна або виконувати парсінг даного коду, або у Kotlin можна створити data class, який повторює структуру ділянки JSON – буде виконаний процес запису даних у data class, проте структура повинна повністю повторювати структуру ділянки JSON, у іншому випадку отримуємо помилку.

Оскільки ми працюємо і з Web (JS), і з Android (Kotlin), при подальших модернізаціях системи, необхідно створити власний API за допомогою Cloud Functions, який би давав уніфіковану структуру одразу для обох варіантів.

Основні дані що надходять з бази даних Firebase, повинні зберігатися у локальній базі даних SQLite. Виходячи із специфіки додатку, база даних здебільше буде використовуватися для збереження основних даних (профіль, налаштування, історія), а дані що швидко змінюються будуть зберігатися у прості змінні, та перезаписуватися.

Використання SQLite у Android, це дуже легко та зручно через те, що у даній системи дана база даних встановлена за замовчуванням. Окрім цього, для взаємодії з базою даних є офіційна бібліотека Room від Google, яка дозволяє оминути використання складних конструкцій для формування запитів до бази даних SQLite, і за допомогою власних API облегшити даний процес. Можна сказати, що Room, це огортка над класичним підходом до взаємодії з базою даних. Саме вона бере на себе всі обов’язки по генерації коду, в свою чергу розробнику залишається лише зосередитися на формуванні логіки запитів.

Зазначимо також, що з технічного сторони, локальна база даних повинна ініціалізуватися раніше ніж Activity завантажаться, існувати до знищення додатку з пам’яті, і бути у вигляді singleton*. Справа у тому, що додаток бере багато даних саме з бази даних, таким чином, для коректної первинної ініціалізації додатку необхідно забезпечити постійний доступ до даної локальної бази даних. Загальна структура взаємодію додатку і бази даних можна подивитись на рисунку 5.10.

* база даних у вигляді singleton – це не обов'язково, проте при створенні об'єкту бази даних, витрачаються певні ресурси і збільшується навантаження на систему. Таким чином, замість багатьох створень і знищень нових екземплярів, краще створити один, і тримати його доступним на протязі усього життєвого циклу додатку.

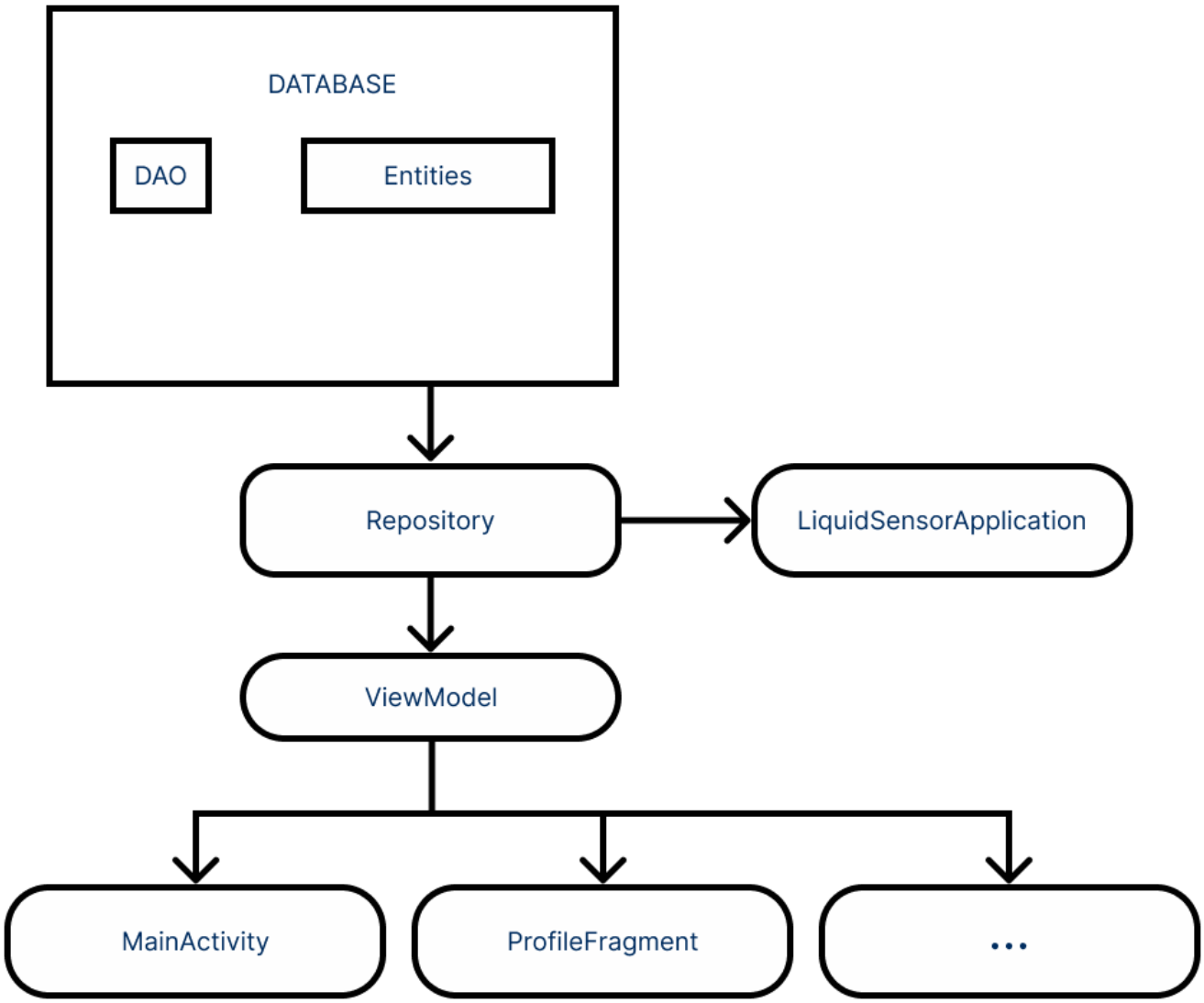


Рисунок 5.10 – Взаємодія додатку і локальної бази даних

На рисунку 5.10 зображено, що для роботи з базою даних необхідно створити сутності – data classes. Дані елементи будуть являти собою таблиці у базі даних. Для даного додатку нам необхідні таблиці для збереження даних про користувача, та загальні налаштування для усіх сенсорів.

DAO (Data Access Object) – це інтерфейс, що являє перелік дій які можна виконувати. Важливо, що в даному випадку це лише інтерфейс, детальну інструкцію про те, як потрібно виконувати запит до бази даних, ми створюємо в іншому місці, а тут лише вказуємо ім’я тієї функції. Завдяки анотаціям, препроцесор Room сам створює додатковий код, що потрібно виконати (але опираючись на наш SQLite скрипт).

Як було вказано вище, для роботи з даною бібліотекою буде використаний патерн Singleton. Він гарантує, що тільки один екземпляр класу існує для додатку і неможливо створити додаткові екземпляри. Виходячи з цього, в нас є єдина точка доступу до бази даних, яка оновлюється і являє собою завжди актуальну інформацію відносно стану локальної бази даних.

За цим патерном необхідно створити Repository у якому буде виконана первинна ініціалізація локальної бази даних. Важливо створити і налаштувати додатковий стартовий клас (LiquidSensorApplication), який завантажується раніше MainActivity. Він необхідний, щоб якомога раніше створити базу даних, а потім продовжити стандартний шлях запуску додатку.

Потім єдиний доступ до функцій взаємодії з базою даних ми можемо отримувати через зв'язку Repository – ViewModel.

Вище ми сказали, що Realtime Database має тригер, що повідомляє систему про появу нових даних у базі даних. Проте, для того щоб автоматично виконувати певні дії у відповідь на оновлення інформації, потрібно використовувати механізми відстеження подій.

Суть відстеження подій полягає у постійному моніторингу за станом інформації і якщо цей стан змінився, тобто змінилася інформація, то необхідно виконати певні дії: оновити дані на графічному інтерфейсі, провести розрахунки, створити повідомлення тощо.

У Kotlin можна використовувати поведінковий шаблон проектування – Observer. Суть даного шаблону, у реалізації залежності «One-to-Many», в результаті чого при зміні об'єкта, що виконує роль Observer, усі об'єкти, що залежать від нього, отримують повідомлення про зміну інформації у ньому.

Раніше стандартом було використання LiveData, проте на даний момент, рекомендується використовувати StateFlow [17, с.251]. Це гілка від основного компонента Flow, що також базується на концепції потоків (асинхронних потоків даних), але також додає спеціальні функції, наприклад, передачу нових значень своїм збирачам, при зміні стану інформації.

Переваги StateFlow:

- наявність центрального джерела. Дане джерело має лише одне значення за раз, і при зміні цього значення, тобто при будь-якому оновленні даних, всі збирачі що підписані на нього, автоматично отримують дане нове значення. Наприклад, коли Arduino отримує нову інформацію від сенсора, він надсилає його до серверу, з серверу дані йдуть, в даному випадку у додаток, і зберігаються у StateFlow. Коли дані зберіглися, тобто перезаписали попереднє значення, оновили його, усі UI елементи, можливо база даних, або інші підписані на даний StateFlow елементи, отримують сигнал про нові дані і оновлять свій стан згідно з новими даними;

- урахування життєвого циклу – при запуску у coroutine (наприклад, у viewLifecycleOwner.lifecycleScope), відповідно в межах життєвого циклу пов’язаного з даним структурним елементом (фрагментом або Activity), StateFlow автоматично припиняє видавати оновлення коли UI знищується, і відновлює, коли він знову стає активним. Це допомагає уникнути витоків пам’яті та несподіваної поведінки під час змін конфігурації;

- актуальність даних – StateFlow зберігає лише останнє значення. При отриманні нового значення, попереднє знищується. Така поведінка спрощує керування станом, порівняно з керуванням кількома змінними стану.

Таким чином, використання даного елементу у додатку, дозволяє автоматизувати отримання нових даних, а також підтримувати актуальність даних, відображених на UI, на протязі усього часу роботи додатку.

РОЗДІЛ 6. ПЕРЕВАГИ ЗАСТОСУВАННЯ І ПЕРСПЕКТИВИ РОЗВИТКУ ІОТ-СИСТЕМИ

6.1. Переваги практичного використання системи

Дана IoT-система проектувалась для використання у тепличному господарстві, з упором на основні вимоги даної сфери, а саме:

- дешевизна системи;

- гнучкість налаштування;

- цілодобовий збір даних відповідно рівня рідини та його історії для операторів теплиці;

- автоматичні оповіщення при досягненні критичних рівнів;

- запобігання аваріям (мало води/переповнення), зниження ручних обходів та витрат на обслуговування;

- економі часу працівників теплиці.

Розглянемо, як дана система впливає на економію часу працівників теплиці. Вхідні дані:

- кількість ручних обходів резервуарів на добу: 2;

- час що займає обхід: 10 хвилин;

- середня погодинна ставка робітника теплиці (США) [27]: 15$ / годину.

$$t_{wt} = \frac{2\ \text{обх} * 10\ \text{хв}}{60\ \text{хв}} = 0.333\ \text{години} \quad (6.1)$$

$$total\_salary = t_{wt} * 365 * x_{salary} = 0.3333 * 365 * 15 = 1824.82\$/\text{рік} \quad (6.2)$$

де

- t_{wt} – характеризує скільки часу у годинах витрачається на обхід;

- total_salary – заробітна плата за рік, з врахуванням тільки часу витраченого на перевірки резервуару;

- x_{salary} – заробітна плата у доларах;

Таким чином, теплиця витрачає близько 1825$ у рік, лише на оплату перевірки робітником резервуарів, двічі на день, протягом року.

Далі наведемо витрати на виготовлення пристрою. Дані витрати носять приблизний характер і повинні показувати загальну картину вартості пристрою:

Таблиця 6.1 – Витрати на виготовлення і підтримку

Назва	**Вартість**
Arduino Nano ESP32	19.30$
Temperature sensor	4$
Conductive sensor	2$
Firebase	1$/місяць[1]
Додаткові витрати	10$[2]
Усього	36.3$

де

- 1 – Firebase витрати, це витрати підписки для зберігання даних і інших взаємодій із сервером. При встановленому часі опитування сенсорів, плата у вигляді 1$ у місяць повністю покриває усі необхідні затрати для обслуговування облікового запису даного користувача;

- 2 – додаткові витрати, це витрати на акумулятори або батарейки, що йдуть у комплекті, пластиковий корпус для пристрою, тощо.

Розрахунок окупності пристрою для теплиці буде виглядати наступним чином:

$$\frac{total\_price}{total\_salary} * year = \frac{(36.3+11)}{1824.85} * 365 = 9.5 \quad (6.3)$$

де

- total_price – це загальна вартість пристрою з доданими 11 місяцями сплати підписки (додатковий місяць входить у вартість пристрою);

- total_salary – це загальна плата робітника за перевірку резервуару, за умовою двох обходів кожен день;

- year – кількість днів у році.

Таким чином, даний пристрій окупиться за 9.5 днів, що дозволяє зекономити вагому частину бюджету теплиці і часу працівників.

Вище був приведений спрощений, проте показовий приклад. В реальному тепличному господарстві, один працівник може перевіряти декілька резервуарів, тобто в один обхід включається декілька точок для перевірки. З часом, необхідно замінювати батарейки або заряджати акумулятор, проводити технічне обслуговування у вигляді очищення сенсорів від конденсату і нальоту, тощо. Все це робить окупність пристрою більш тривалою, проте усе одно вигідною.

Незважаючи на те, що первинно система проектувалась для тепличного господарства, вона також може використовуватись і у інших сферах зі схожими вимогами, наприклад:

- контроль рівня в акваріумах;
- системи рівня у резервуарах протипожежного водопостачання;
- використання на дачних ділянках та у приватних будинках;
- моніторинг рівня води у резервуарах для поливу у сільському господарстві.

6.2. Можливості подальшого розвитку системи

Дана IoT-система повністю виконує вказані вимоги, проте має певні сторони, які можна покращити при майбутніх модернізаціях системи. Виділимо основні напрямки, на яких необхідно зосередитися.

Віддаленість вимірювань. На даний момент, усі сенсори з’єднані з Arduino за допомогою дротів. Для роботи з двома і більшою кількістю резервуарів, це знижує зручність. Необхідно реалізувати можливість бездротового з’єднання. Це можна зробити за допомогою додаткових радіомодулів (наприклад: nRF24L01, LoRa, тощо). Тип радіомодуля залежить від віддаленості основного пристрою від сенсору та кількості перешкод між ними (стіни, конструкції, тощо).

Універсальність. На даний момент є можливість підключити лише кондуктивний та ультразвуковий сенсор, з можливістю налаштування конфігурації підключення. Даних сенсорів вистачає, щоб вірно, з певною похибкою виміряти

рівень води і задовольнити потенціальних клієнтів. Проте, для подальшого розвитку необхідно розширити набір різноманітних типів сенсорів, для можливості використанні даної системи у більш широкому діапазоні задач. Основний алгоритм роботи необхідно прописати у коді пристрою, а налаштування винести до хмари. Вибір для даної системи Arduino і залишок невикористаної пам'яті, роблять даний напрям покращення, одним із найбільш пріоритетних шляхів модернізації системи.

Власний сервер. На даний момент, у якості сервера використовується BaaS, проте, незважаючи на доступність, простоту використання і налаштування, дане рішення у майбутньому необхідно замінити на власний сервер. Дана необхідність викликана тим, що власний сервер має набагато більшу гнучкість створення, налаштування і розробки функціонала, що дозволяє, наприклад, оброблювати дані на стороні сервера і відправляти попередження (додаток, пошта, телефон) навіть при вимкнутому веб-сайті або додатку. Це в свою чергу збільшує можливості системи і рівень взаємодії між користувачем та системою. Проте, серед негативних сторін є складність реалізації даної задачі та додаткові витрати залучення додаткових розробників, підтримку сервера та його оренду (це в свою чергу може включати або плату провайдеру, або пряму оплату хостингу).

Енергоефективність. На даний момент, пристрій отримує живлення від розетки за допомогою USB адаптеру (5V) або батарейок. Проте дане рішення негативно впливає на автономність та віддаленість використання пристрою, та напряму залежить доступу до розеток.

Для реалізації більш тривалої роботи, можливим рішенням є додаткова інтеграція у пристрій сонячних панелей. Дане рішення має свої переваги і недоліки. Серед недоліків є необхідність у сонячному світлі та збільшення вартості. Проте, перевагами є збільшення терміну роботи без заміни батарейок та зменшення енергоспоживання.

ВИСНОВКИ

Система опалення відкритого типу, що застосовується у невеликих тепличних господарствах, не є сучасним рішенням, проте, воно бюджетне, просте для налаштування і дуже поширене серед невеликих господарств. Однак, з даною простотою з’являється і додаткові нюанси, зокрема, потреба у регулярних перевірках резервуара розширювального баку.

Персональні перевірки робітниками протягом доби, займають робочий час і приводять к додатковим витратам. Згідно розрахункам наведеним вище, витрати на перевірки складають близько 1825$ у рік.

Рішення, що існують на даний момент, є або занадто коштовними і не націленими на малий та бюджетний сектор, або з обов’язковим надмірним функціоналом, або без дистанційного контролю, або ненадійні (з непідтвердженою надійністю) рішення.

Були проаналізовані вимоги до сенсорів і системи у цілому, та умови у яких дана IoT-система повинна працювати. Основний упор був зроблений на пошук оптимальних методів і сенсорів для вимірювання рівня рідини, можливість дистанційного моніторингу, і економічність даних рішень загалом, проте з можливістю подальшого розвитку без критичної зміни складових системи.

У якості методів і типів сенсорів, для рішення поставлених задач були обрані наступні варіанти: кондуктивний і ультразвуковий сенсори. Вибір даних типів сенсорів обумовлено специфікою середовища і низькою вартістю даного рішення.

Можливість підключення декількох типів сенсорів до даної системи є саме можливістю, а не необхідністю. Жоден із типів сенсорів не може бути універсальним. Декілька типів сенсорів покривають різні умови використання, підвищують гнучкість використання системи, та забезпечують точність у різних сценаріях. Наприклад, кондуктивний сенсор краще використовувати у невеликих за висотою резервуарах та з відносно чистою водою. Навпаки, ультразвуковий сенсор має набагато більший за висотою діапазон вимірювання та може працювати із

забрудненою водою або іншою рідиною. Проте, вартість у нього вище і він має залежність від зовнішньої температури, вологості, тощо.

У якості плати мікроконтролеру був обраний Arduino Nano ESP 32, який має середню вартість, проте йде з вбудованим Wi-Fi модулем та великою кількістю пам’яті, що робить можливим подальші модифікації. Зазначимо, що надалі є сенс робити оновлення програмної частини пристрою, що робить можливим перевикористання пристрою.

Код алгоритму роботи пристрою і сенсорів знаходиться на пристрої, проте необхідні змінні (конфігурації підключення, налаштування сенсорів) розташовані на сервері, і цілком залежать від потреб користувача.

У якості сервера був обраний Firebase. Дане рішення зумовлено швидкістю і простотою налаштування даної BaaS та дешевизною послуг, що вони пропонують. Проте, з часом, рекомендується налаштувати власний сервер для більшої гнучкості налаштувань і взаємодії між користувачем та системою (обробка даних на стороні сервера і відправка попереджень (додаток, пошта, телефон), навіть при вимкнутому веб-сайті або додатку).

Для веб-сайту була обрана класична модель архітектури без додаткових фрейморків. Дане рішення обумовлено швидкістю розробки, простотою підтримки сайту та відсутністю ускладнення логіки на даному етапі реалізації системи. Цей підхід дозволяє швидше досягнути цілі без значної втрати якості продукту.

Мобільний додаток проектується лише для ОС Android, через те, що переважна кількість пристроїв, що використовують працівники даної сфери, є саме Android. Була обрана архітектура Single Activity, що дозволила будувати додаток у вигляді блоків, що можна перевикористовувати для різних розмірів екрану і потреб.

Серед напрямів можливого вдосконалення, можна виділити:, реалізацію віддаленості вимірювань, універсальність, наявність власного серверу, енергоефективність.

Таким чином, незважаючи на те, що система повністю виконує свої функції, завжди є напрямки для покращення існуючого рішення. Дане формулювання можна застосувати до будь якого існуючого приладу, системи або рішення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Adam D. Scott, Matthew MacDonald, Shelley Powers, JavaScript Cookbook – USA: O'Reilly, 2023 – 526 с.

2. Article by Robert Brown. Topic: Ultrasonic sensor HC-SR04: [Електронний ресурс] – Режим доступу до ресурсу: https://nerdytechy.com/guide-for-arduino-ultrasonic-sensor-hc-sr04/?utm_medium=organic&utm_source=yasmartcamera

3. David R. Lide, ed., CRC Handbook of Chemistry and Physics, 90th Edition (CD-ROM Version 2010), CRC Press/Taylor and Francis, Boca Raton, FL, 2010 – 2760 с.

4. Firebase Authentication [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/auth

5. Firebase Cloud Firestore [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/products/firestore

6. Firebase Realtime Database [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/database

7. Franklin Electric [Електронний ресурс] – Режим доступу до ресурсу: https://www.franklinwater.com/

8. Griffths Dawn, Griffiths David, Head First, Android Development, Third Edition – USA: O'Reilly, 2023. – 905 с.

9. IoT based water level Indicator using Arduino and ESP8266-01 Wi-Fi module [Електронний ресурс] – Режим доступу до ресурсу: https://iotstarters.com/iot-based-water-level-indicator-using-esp8266/

10. IP ratings [Електронний ресурс] – Режим доступу до ресурсу: https://www.iec.ch/ip-ratings

11. Laurence Pierre-Olivar, Hinchman-Dominguez Amanda, Programming Android with Kotlin, Achieving Structured Concurrency with Coroutines, First Edition – USA: O'Reilly, 2024. – 338 с.

12. Liquid-Level Monitoring Using a Pressure Sensor [Електронний ресурс] – Режим доступу до ресурсу: https://www.ti.com/lit/an/snaa127/snaa127.pdf

13. Matt Frisbie, JavaScript for Web Developers, Fifth Edition – Canada: John Wiley & Sons, 2024. – 1105 с.

14. Micropilot FMR30B – radar sensor for basic applications [Електронний ресурс] – Режим доступу до ресурсу: https://www.endress.com/en/field-instruments-overview/level-measurement/Level-Micropilot-FMR30B?t.tabId=product-overview

15. Optical liquid level sensor system [Електронний ресурс] – Режим доступу до ресурсу: https://www.nxp.com/docs/en/application-note/optical-liquid-level-sensor-system-tsu8.pdf

16. Pascal Thormeier, Mastering CSS Grid, First Edition – UK: Packt Publishing Ltd, 2023. – 330 с.

17. Peter Spath, Pro Android with Kotlin, Second Edition – Leipzig, Germany: Apress, 2022. – 881 с.

18. System with conductive relay water level sensor [Електронний ресурс] – Режим доступу до ресурсу: https://www.analog.com/en/resources/technical-articles/conductive-relay-water-level-sensor-pt-ii.html

19. Thomas Hunter II, Bryan English, Multithreaded JavaScript – USA: O`Reilly Media, 2022. – 270 с.

20. VEGAPULS 61 Radar sensor for continuous level measurement of liquids [Електронний ресурс] – Режим доступу до ресурсу: https://www.vega.com/en/products/product-catalog/level/radar/vegapuls-61

21. Глущенко О. Л., Конспект лекцій з дисципліни теплотехнічні вимірювання та прилади / О. Л. Глущенко – Кам'янське: ДДТУ, 2018. – 92 с.

22. Документація кондуктивного сенсора: [Електронний ресурс] – Режим доступу до ресурсу: https://www.theengineeringprojects.com/2020/07/water-sensor-library-for-proteus.html

23. Константінов Ю. М., Гіжа О. О., Технічна механіка рідини і газу / Ю. М. Константінов, О. О. Гіжа – Київ: Вища шк., 2002. – 277 с.

24. Кореньков В. П., Ділай І. В., Вимірювання рівня рідини / В. П. Кореньков, І. В. Ділай – Львів: НУЛП, 2012. – 11 с.

25. Котли із вбудованими сенсорами вимірювання основних показників Kiturami [Електронний ресурс] – Режим доступу до ресурсу: https://krb.co.kr/

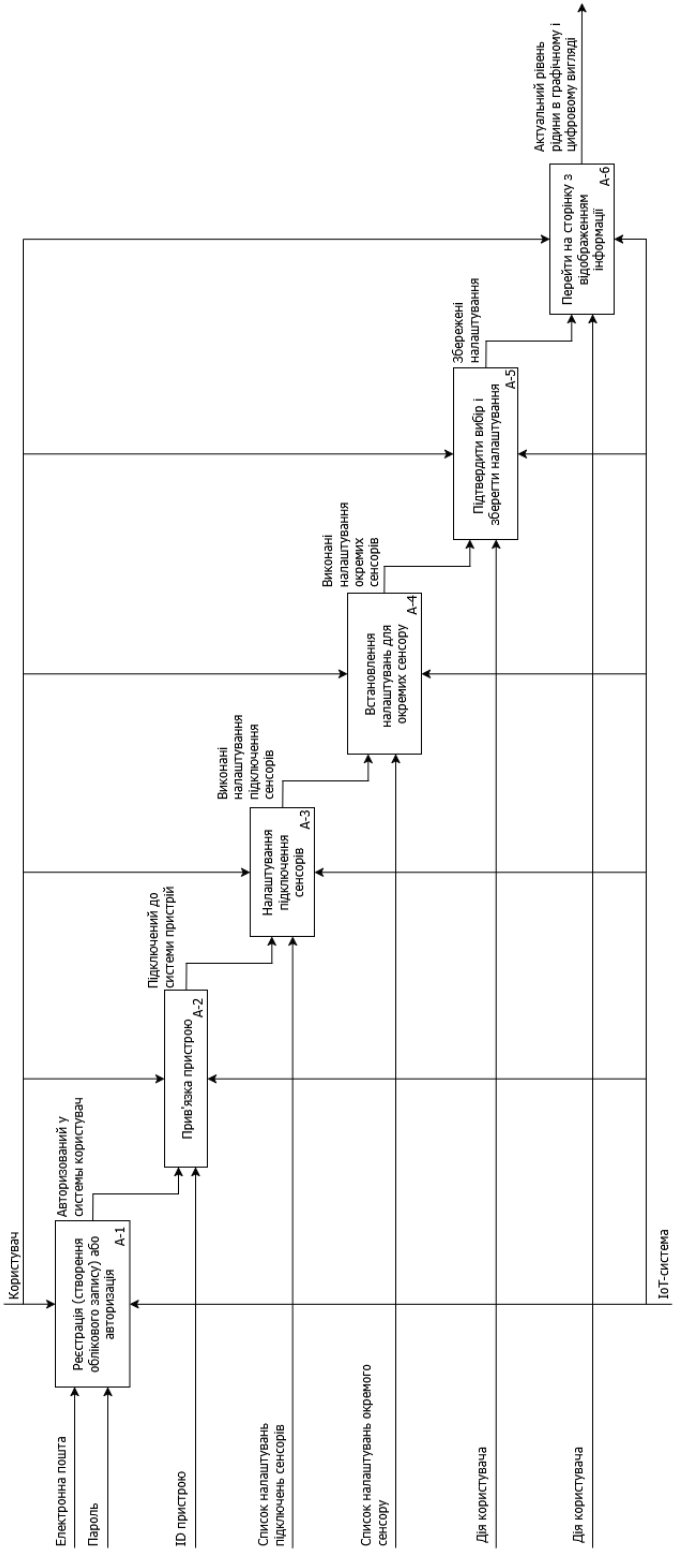
26. Патрєва Л. С., Каницька І. В., Метрологія / Л. С. Патрєва, І. В. Каницька / Миколаїв – 106 с. – [Електронний ресурс] – Режим доступу до ресурсу: https://dspace.mnau.edu.ua/jspui/bitstream/123456789/9323/1/metrologiya-konspekt.pdf

27. Середня погодинна ставка робітника теплиці: [Електронний ресурс] – Режим доступу до ресурсу: https://www.bls.gov/oes/2023/may/oes452092.htm

ДОДАТКИ

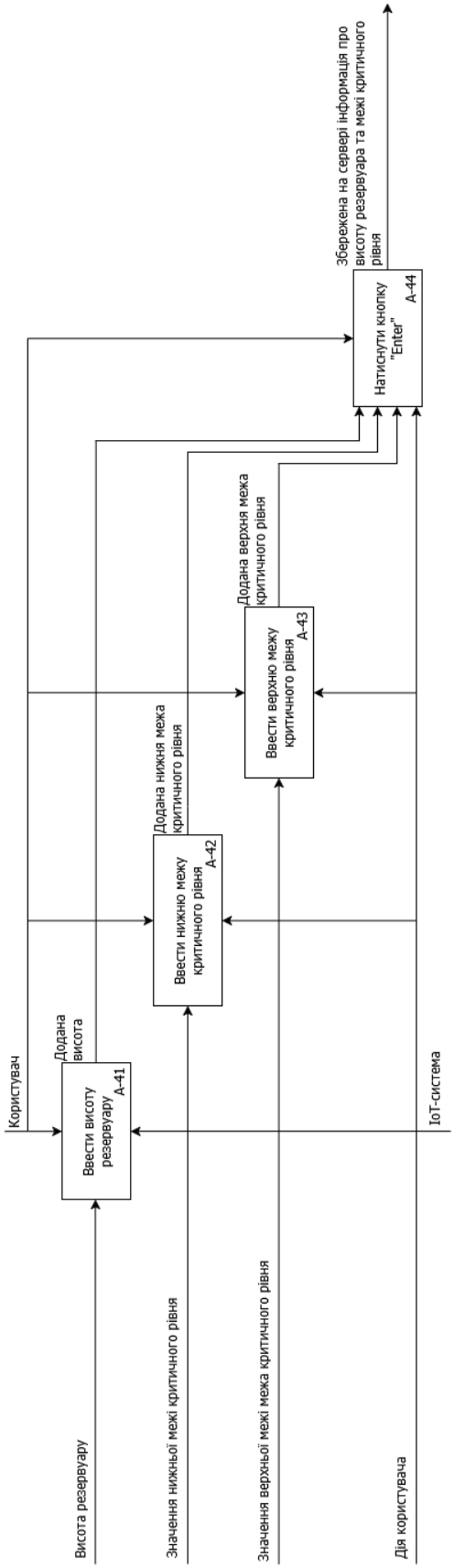
Додаток А

Перший функціональний рівень нотації *IDEF0*. Декомпозиція нульового блоку



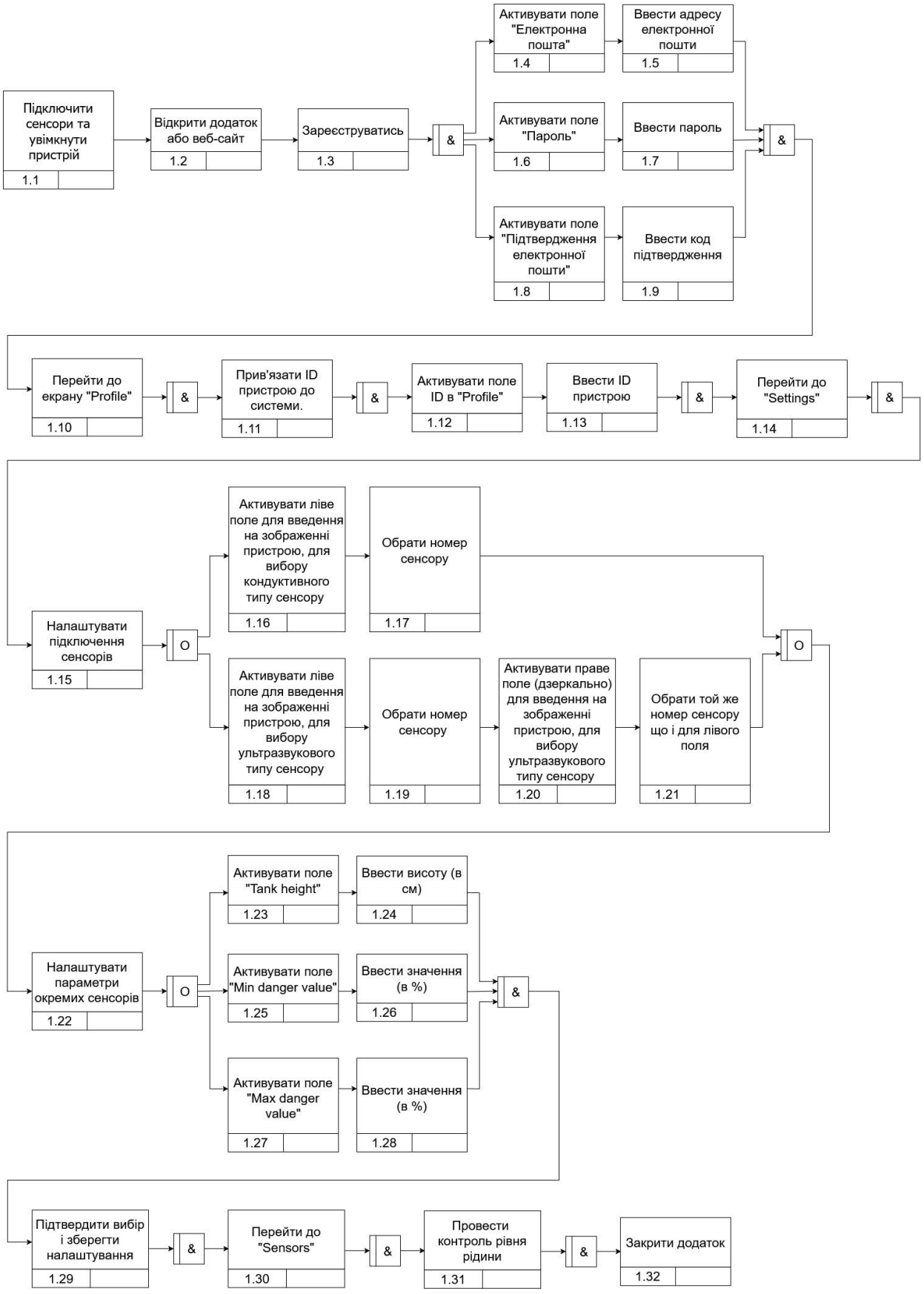
Додаток Б

Другий функціональний рівень нотації *IDEF0.* Декомпозиція блоку А4



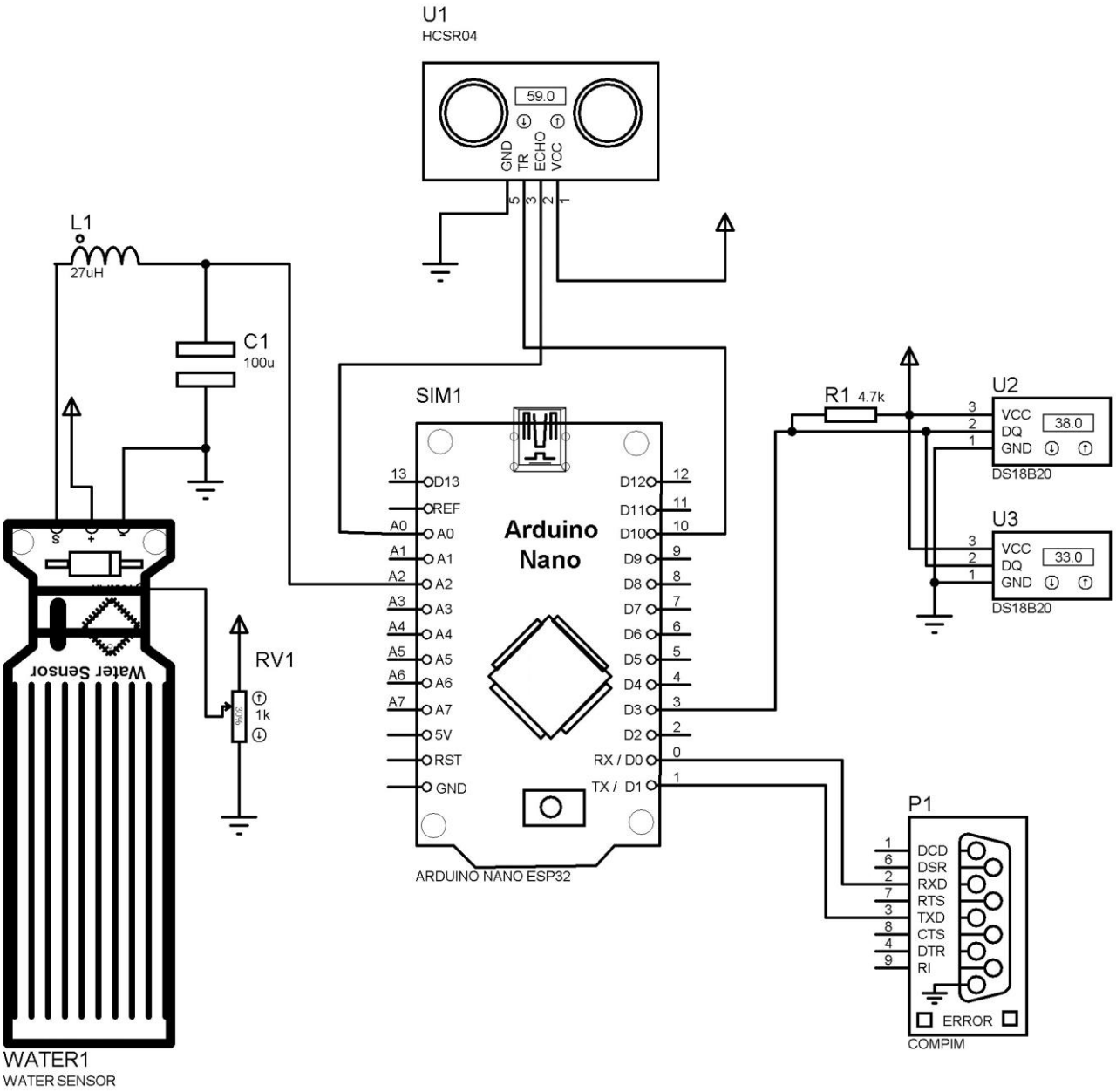
Додаток В

Нотація *IDEF3*



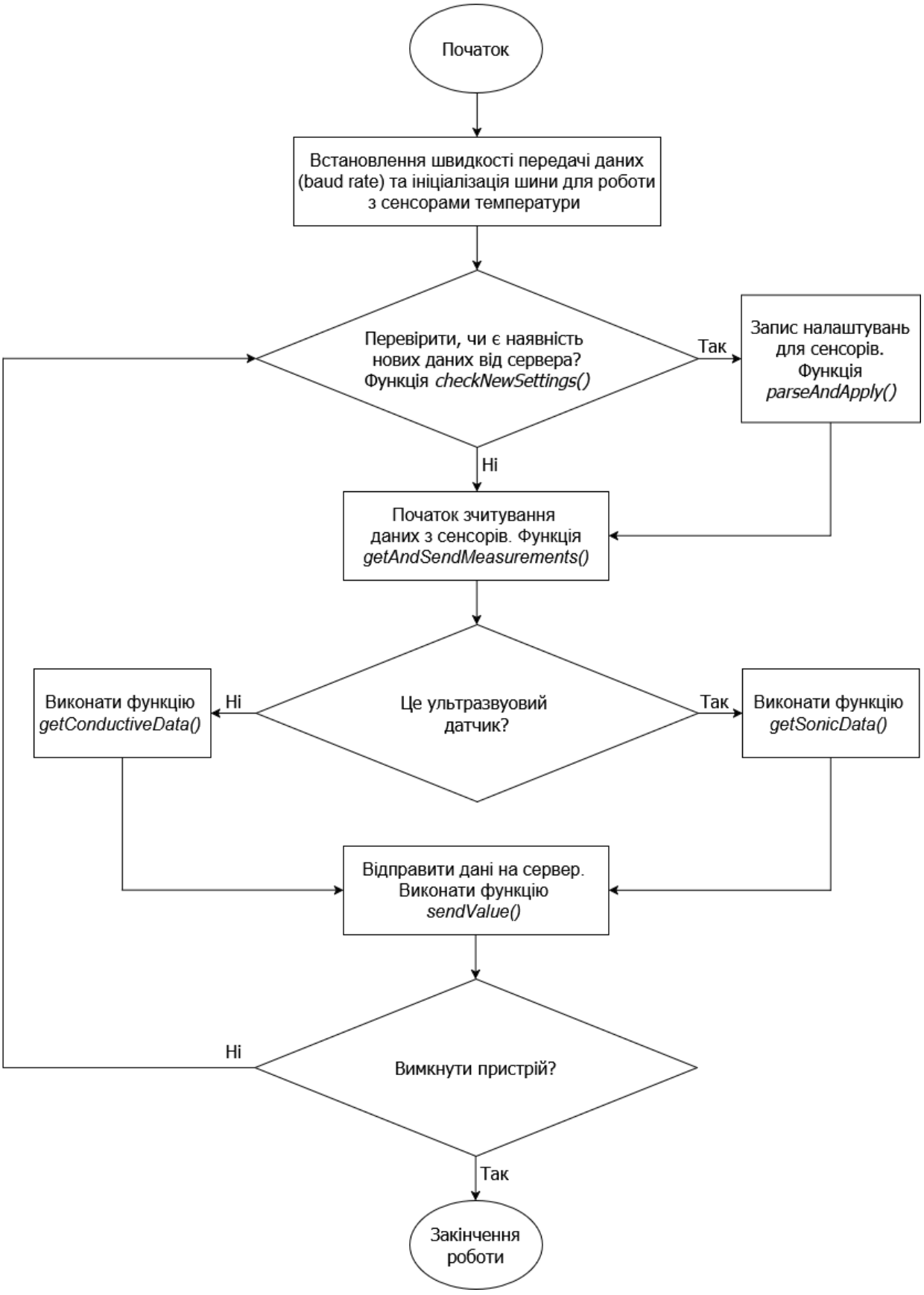
Додаток Г

Повна схема пристрою



Додаток Д

Алгоритм роботи пристрою



Додаток Е

Текст програми «LIQUID_SENSOR_INFO»

```
#include <Arduino.h>
#include <ArduinoJson.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 3

struct SensorData{
  String type;
  int a;
  int d;
  int tankHeight;
  int minLevel;
  int maxLevel;
  float value;
};

/* Config temperature sensors */
OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature tSensors(&oneWire);
float tCurrent = 0.0;

/* Maximum number of sensors */
const int maxSensors = 7;

/* An array with all sensors + their settings */
SensorData sensors[maxSensors];

/* Temporary variable for storing data from the Serial buffer */
String data = "";

/* A temporary variable for storing the measurement result in centimeters */
long cmInner = 0;

/* Delay time of the acoustic signal on the echo sounder */
long durationInner = 0;

/* Array with analog output addresses */
int aPins[] = {A0, A1, A2, A3, A4, A5, A6};

/* A temporary variable to store a character from the Serial buffer */
char charIn = 'a';

/* For conductive sensor */
const int R_FIXED = 10000;
const int VCC = 5;
const float ALPHA_WATER = 0.02;
float vOut = 0.0;
float rSensor = 0.0;
float r25 = 0.0;
float vCorr = 0.0;

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

```
void loop() {
  checkNewSettings();
  getAndSendMeasurements(sensors);
  delay(20000);
}

/* Checks the stream for new data and writes it if there is any */
void checkNewSettings(){
  // Check the stream, if anything is available
  if (Serial.available()) {
    /* We need to make sure that the whole part is ready */
    delay(10);
    /* Read the data until the character '#' (non-include)*/
    while(Serial.available()){
      charIn = Serial.read();
      if(charIn == '#'){
        break;
      } else {
        data += charIn;
      }
    }

    Serial.println(data);
    Serial.println("#");

    /* We can send to Arduino only sensors settings,
    so if length > 0, that is JSON with settings */
    if (data.length() > 0) {
      parseAndApply(data);
    } else {
      Serial.println("Empty input");
      Serial.println("#");
    }

    /* Clear the data variable */
    data = "";
  }
}

/* Receiving settings from the server */
void parseAndApply(const String &jsonStr){
  // allocate space for JSON
  StaticJsonDocument<512> doc;

  // if the JSON is correct, then save it to 'doc', otherwise initialize err
  DeserializationError err = deserializeJson(doc, jsonStr);

  // check the error
  if(err){
    Serial.print("JSON parse failed: ");
    Serial.println(err.c_str());
    return;
  }

  JsonObject root = doc.as<JsonObject>();

  for (JsonPair kv : root) {
    const char* key = kv.key().c_str();
    JsonObject obj = kv.value().as<JsonObject>();
    sensors[atoi(key)] = createSensorData(
      obj["type"].as<String>(),
```

```
      obj["a"].as<int>(),
      obj["d"].as<int>(),
      obj["tankH"].as<int>(),
      obj["min"].as<int>(),
      obj["max"].as<int>()
    );
  }
}

/* Creating and populating one sensor with settings */
SensorData createSensorData(String type, int a, int d, int tankSize,
                            int minLevel, int maxLevel){
  SensorData result;
  result.type = type;
  result.a = a;
  result.d = d;
  result.tankHeight = tankSize;
  result.minLevel = minLevel;
  result.maxLevel = maxLevel;
  return result;
}

/* Displaying the settings of one sensor */
void showSensorSettings(SensorData sensor){
  Serial.println((String) "Type: " + sensor.type +
  + "\na: " + sensor.a
  + "\nd: " + sensor.d
  + "\ntankSize: " + sensor.tankHeight
  + "\nminLevel: " + sensor.minLevel
  + "\nmaxLevel: " + sensor.maxLevel
  + "\nvalue: " + sensor.value);
  Serial.println("#");
}

/* Shows all parameters for all sensors that are connected */
void showAllEnabledSensors(SensorData sensors[]){
  for(int i = 0; i < maxSensors; i++){
    if(sensors[i].type != ""){
      showSensorSettings(sensors[i]);
    }
  }
}

/* Receive and send data to the server */
void getAndSendMeasurements(SensorData sensors[]){
  for(int i = 0; i < maxSensors; i++){
    tSensors.requestTemperatures();
    tCurrent = tSensors.getTempCByIndex(i); // search on one bus.
    if(sensors[i].type == "UlS"){ // ultrasonic
      sensors[i].value = getSonicData(aPins[i], sensors[i].d, sensors[i].tankHeight,
tCurrent);
      sendValue(i, sensors[i].value);
    } else if(sensors[i].type == "Con"){ // conductive
      sensors[i].value = getConductiveData(aPins[i], tCurrent);
      sendValue(i, sensors[i].value);
    }
  }
}

/* Collecting data from an ultrasonic sensor */
float getSonicData(int echoA, int trigD, int tankHeight, float temperature){
  pinMode(trigD, OUTPUT);
  pinMode(echoA, INPUT);
```

```
  digitalWrite(trigD, LOW);
  delayMicroseconds(5);
  digitalWrite(trigD, HIGH);

  /* After setting the signal level to high, wait about 10 microseconds.
  At this point, the sensor will send signals at a frequency of 40 kHz */
  delayMicroseconds(10);
  digitalWrite(trigD, LOW);

  /* Delay time of the acoustic signal on the echo sounder */
  durationInner = pulseIn(echoA, HIGH);

  /* The number of centimeters between the sensor and the surface */
  cmInner = (durationInner / 2) * ((331.3 + 0.606 * temperature) / 10000);

  /* Percentage of occupied space */
  return 100.0 * cmInner / tankHeight;
}

/* Collecting data from a conductive sensor */
float getConductiveData(int aPin, int temperature){
  pinMode(aPin, INPUT);
  vOut = analogRead(aPin) * VCC / 1023.0;
  rSensor = R_FIXED * (vOut / (VCC - vOut));
  r25 = rSensor / (1 + 0.02 * (temperature - 25));
  vCorr = VCC * (r25 / (r25 + R_FIXED));
  return vCorr / VCC * 100.0;
}

/* Sending data to the server (via COM port) */
void sendValue(int sensorId, float value) {
  Serial.println((String) "Id: " + sensorId + " || Value: " + value);
  Serial.println("#");
}
```