

Міністерство освіти і науки України  
Державний заклад  
«Луганський національний університет імені Тараса Шевченка»

Навчально-науковий інститут математики та інформаційних технологій

Кафедра інформаційних технологій та систем

**Бородін Микита Сергійович**

**АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ПАРАМЕТРАМИ  
АКВАРІУМА З ВИКОРИСТАННЯМ ІОТ-ТЕХНОЛОГІЙ**

**кваліфікаційна робота**

**здобувача вищої освіти другого (магістерського) рівня**

**освітньої програми «Мультимедійні системи»**

**за спеціальністю 121 Інженерія програмного забезпечення**

Особистий підпис \_\_\_\_\_ Микита БОРОДІН

Науковий керівник \_\_\_\_\_ Володимир ДОНЧЕНКО,  
старший викладач кафедри  
інформаційних технологій та систем

Завідувач кафедри \_\_\_\_\_ Микола СЕМЕНОВ,  
кандидат педагогічних наук, доцент  
кафедри інформаційних технологій  
та систем

Полтава – 2025

## АНОТАЦІЯ

**Бородін М. С.**

**Тема:** Автоматизована система управління параметрами акваріума з використанням IoT-технологій.

**Спеціальність:** 121 «Інженерія програмного забезпечення».

**Установа:** ЛНУ імені Тараса Шевченка, 2025р.

**Магістерська робота містить:** 92 с., 49 рис., 2 табл., 1 додаток, 30 джерел.

**Об'єктом дослідження** - процес управління водною мікроекосистемою на прикладі домашніх акваріумів.

**Предмет дослідження** - управління "розумним" акваріумом з автономним середовищем на основі платформи Arduino.

**Мета роботи** - розробка автоматизованої система управління параметрами акваріума з використанням IoT-технологій за допомогою мікроконтролерної системи на основі платформи Arduino.

**Результати роботи** – в роботі виконано огляд та проаналізовано існуючі рішення, розроблено структурну та функціональну схеми системи, електричну принципову схему та алгоритм функціонування автоматизованої системи керування розумним акваріумом.

Обрано окремі компоненти та технології для розробки автоматизованої системи, розроблено макет системи, мобільний застосунок для дистанційного керування та сервер для комунікації між ними. Отримані результати можуть бути корисними для спрощення догляду за домашніми акваріумами та автоматизації різноманітних процесів.

**Ключові слова:** МІКРОКОНТРОЛЕР, РІВЕНЬ ВОДИ, АВТОМАТИЗОВАНА СИСТЕМА, РОЗУМНИЙ АКВАРІУМ, ТЕМПЕРАТУРНИЙ КОНТРОЛЬ, ДИСТАНЦІЙНЕ КЕРУВАННЯ, МАКЕТ ПРИСТРОЮ, ДАТЧИКИ, RASPBERRY PI, ARDUINO UNO, IoT.

## ANNOTATION

**Borodin Mykyta**

**Theme:** Automated aquarium parameter management system using IoT technologies.

**Speciality:** 121 "Software Engineering".

**Institution:** Luhansk Taras Shevchenko National University (LTSNU), 2025 year.

**Bachelor work of:** 92 p., 49 im, 2 table, 1 ap., 30 sources.

**A research object of:** - the process of managing an aquatic microecosystem using the example of home aquariums.

**The article of research-** control of a "smart" aquarium with an autonomous environment based on the Arduino platform.

**An aim of research is** - development of an automated aquarium parameter management system using IoT technologies using a microcontroller system based on the Arduino platform.

**Job performanes.-** the work reviewed and analyzed existing solutions, developed a structural and functional diagram of the system, an electrical schematic diagram and an algorithm for the functioning of an automated smart aquarium control system.

Selected individual components and technologies for the development of an automated system, developed a system layout, a mobile application for remote control and a server for communication between them. The results obtained can be useful for simplifying the care of home aquariums and automating various processes.

**Keywords:** MICROCONTROLLER, WATER LEVEL, AUTOMATED SYSTEM, SMART AQUARIUM, TEMPERATURE CONTROL, REMOTE CONTROL, DEVICE LAYOUT, SENSORS, RASPBERRY PI, ARDUINO UNO, IoT.

## ЗМІСТ

<b>ВСТУП.....</b>	<b>6</b>
<b>РОЗДІЛ 1. НАУКОВО-ДОСЛІДНА ЧАСТИНА .....</b>	<b>9</b>
1.1. Призначення та сфера застосування.....	9
1.2. Основні функції «розумного» акваріуму .....	10
1.3. Огляд існуючих рішень .....	18
1.4. Управління «розумним» кліматом .....	31
Висновки до розділу.....	38
<b>РОЗДІЛ 2. РОЗРОБКА ПРИНЦИПОВОЇ СХЕМИ ПРИСТРОЮ.....</b>	<b>39</b>
2.1. Розробка та обґрунтування алгоритму функціонування та структурної схеми пристрою .....	39
2.2. Розроблення та розрахунок принципової електричної схеми .....	53
2.3. Розробка друкованої плати.....	56
2.4. Розробка програмного забезпечення проєктованої системи .....	57
2.4.1. Теоретичні відомості.....	57
2.4.2. Програма для мікроконтролера.....	58
Висновки до розділу.....	68
<b>РОЗДІЛ 3. РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ПАРАМЕТРАМИ АКВАРІУМА З ВИКОРИСТАННЯМ ІoT-ТЕХНОЛОГІЙ .....</b>	<b>69</b>
3.1. Обґрунтування вибору технологічного стеку для розробки мобільних застосунків .....	69
3.1.1. Обґрунтування вибору технології React Native для розробки мобільного застосунку.....	70
3.1.2 Менеджер пакетів Node Package Manager.....	72
3.1.3. Мова програмування TypeScript .....	74
3.1.4. Бібліотеки Redux та Redux Toolkit.....	75
3.1.5. Платформа Expo .....	77
3.2. Обґрунтування вибору технологій для розробки серверного програмного забезпечення.....	78
3.2.1. Фреймворк Express.js .....	80

3.2.2 Платформа Node.js.....	80
3.3. Розробка мобільного застосунку .....	81
3.4. Розробка серверного програмного забезпечення.....	85
Висновки до розділу.....	87
<b>ВИСНОВКИ .....</b>	<b>88</b>
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....</b>	<b>90</b>
<b>ДОДАТОК.....</b>	<b>93</b>

## ВСТУП

Під час управління різними процесами в різних галузях господарства та виконання великого обсягу повторюваних завдань ефективність роботи людини зменшується, що призводить до втоми, уповільнення реакції та інших негативних наслідків. Вплив зовнішніх факторів може підвищувати схильність людини до прийняття помилкових рішень, що пов'язано з людським фактором. Тому розвиток високих технологій, робототехніки та автоматизації в різних сферах діяльності спрямований на зменшення обсягу важкої монотонної роботи та уникнення помилок, спричинених нею, що, у свою чергу, зменшує вплив людського фактора на автоматизовані системи.

У сфері управління мікроекосистемами на господарських підприємствах (лісових і тваринних господарствах, фермах тощо) або в побуті (догляд за домашніми улюбленцями) людський фактор також має значний вплив. Адже саме людина несе відповідальність за живі організми. І рівень цієї відповідальності не залежить від того, чи вирощує вона тварин для подальшого використання як їжу (як на фермах), чи доглядає домашніх тваринок для задоволення естетичних потреб (наприклад, у випадку з домашнім акваріумом).

Об'єктом даного дослідження стала акваріумістика, яка полягає у створенні моделі екосистеми в замкнутому штучному водоймищі [2]. Більшість водних мешканців дуже вимогливі до параметрів гідросфери, в якій вони живуть. Навіть незначні зміни в стані окремих компонентів водної системи можуть призвести до незворотних змін у штучній гідросфері. Тому уважне спостереження за водоймою та своєчасна реакція на зміни різних факторів є важливими проблемами в цій сфері. На сьогоднішній день, з швидким розвитком технологій автоматизації, використання інноваційних мікропроцесорних систем знаходить широке застосування в управлінні водними мікроекосистемами [3].

Для підтримки оптимальних умов життя тварин у закритих штучних середовищах необхідно ретельно спостерігати за різними компонентами середовища та своєчасно реагувати на найменші зміни їх показників. При автоматизації таких дій важливо забезпечити бездоганну роботу всіх частин

системи і своєчасне інформування користувача про несправності окремих модулів. Тому під час проектування та розробки автоматизованих систем необхідно чітко дотримуватися встановлених методів і практик. Дотримання цих методик при створенні автономних мікроконтролерних систем управління дозволяє отримати надійні та стійкі до збоїв пристрої, здатні довго і безперебійно функціонувати.

Мікропроцесорні та мікроконтролерні системи стали широко використовуваними в промисловості. Проте в галузі акваріумістики вони залишаються відносно рідкісними через складність налаштування для конкретного акваріума.

Власники звичайних домашніх акваріумів часто вимушені використовувати саморобні або замовлені автоматизовані системи, спеціально призначені для їхньої конкретної гідросфери. Такі системи часто потребують спеціального налаштування та підтримки, що створює проблеми для звичайних користувачів.

У зв'язку з цим існує актуальна потреба у розробці простого, але ефективного пристрою, який можна легко адаптувати для різних акваріумів.

Таким чином, робота над створенням "розумного" акваріума з автономним середовищем на базі Arduino була спрямована на аналіз існуючих систем автоматизації для підтримки гідросфери акваріума. Це включало порівняння різних моделей та розробку уніфікованої мікроконтролерної системи для широкого спектру акваріумних середовищ.

**Об'єктом дослідження** - процес управління водною мікроекосистемою на прикладі домашніх акваріумів.

**Предмет дослідження** - управління "розумним" акваріумом з автономним середовищем на основі платформи Arduino.

**Мета роботи** - розробка автоматизованої система управління параметрами акваріума з використанням IoT-технологій за допомогою мікроконтролерної системи на основі платформи Arduino.

Ця система повинна виконувати наступні функції:

- фіксувати стан компонентів середовища життєдіяльності живих організмів;
- запускати відповідні пристрої системи у відповідь на сигнали датчиків, які вимірюють параметри водної мікроекосистеми (водоймища, акваріума тощо).

**Досягнення зазначеної мети передбачає вирішення таких основних завдань:**

- дослідити актуальність проєктування та розробки пристрою шляхом аналізу переваг і недоліків існуючих рішень;
- вивчити методи проєктування та розробки автоматизованих систем на основі мікроконтролерів;
- обрати необхідні компоненти для розробки макету IoT пристрою;
- на основі результатів досліджень і експериментів спроєктувати та розробити ефективну автономну мікроконтролерну систему на основі платформи Arduino для управління водною мікроекосистемою на прикладі домашніх акваріумів;
- вибрати сучасні технології для створення програмного забезпечення;
- розробити програмне забезпечення для мобільного додатка;
- розробити серверне програмне забезпечення.



## РОЗДІЛ 1. НАУКОВО-ДОСЛІДНА ЧАСТИНА

### 1.1. Призначення та сфера застосування

Інтернет речей (IoT) – це одна з найшвидше зростаючих галузей технологій у сучасному світі. Щороку кількість IoT пристроїв у світі значно збільшується, і вони знаходять застосування у найрізноманітніших сферах людської діяльності.

Лідерами у впровадженні та розвитку технологій Інтернету речей є Японія, Китай, Сполучені Штати Америки та деякі країни Європейського Союзу. Важливий внесок у прогрес IoT технологій роблять такі промислові гіганти, як Toyota, Xiaomi, Huawei, Google, Microsoft, Amazon тощо.

IoT пристрої використовуються як у повсякденному житті звичайних людей, так і у професійній діяльності. Вони також можуть бути впроваджені на великих виробництвах та в медицині для автоматизації різних процесів. Крім того, поряд із системами домашньої безпеки, все більшої популярності набувають розумні домашні пристрої, які спрощують рутинні завдання та дозволяють людям забути про деякі домашні обов'язки.

Будь-яка будівля складається з різних підсистем, що відповідають за виконання різноманітних завдань. З часом ці технології еволюціонують, і автоматизовані системи починають інтегруватися у всі «екосистеми» дому. Завдяки цьому людині більше не доводиться виконувати всі роботи, які раніше вимагали ручної праці. «Розумний дім» - це комплексна система, в якій управління всіма пристроями та приладами здійснюється автоматично або за допомогою смартфона, ноутбука чи іншого пристрою. Саме поняття «Smart house» було сформульоване ще в 1970-х роках у Вашингтонському Університеті як «будівля, що забезпечує продуктивне та ефективне використання робочого простору» [20].

Принцип цієї системи передбачає абсолютно новий підхід до організації життєзабезпечення, який за допомогою комплексу програмно-апаратних засобів значно підвищує ефективність роботи та надійність управління всіма системами та виконавчими пристроями будівлі. На рисунку 1.1 нижче показано приклад такої системи. Управління підсистемами в цьому випадку здійснюється через мобільний додаток з використанням інтернет-з'єднання.



Рис. 1.1. Приклад системи «Розумний дім» [1]

Варто розуміти, що окремі системи у «розумному домі» можуть працювати як спільно, так і незалежно одна від одної, виконуючи свої завдання. Прикладом такої системи є домашній акваріум, управління яким здійснюється через мобільний додаток. Цей пристрій поєднує автоматизацію та інноваційні технології, що дозволяє власнику підтримувати оптимальні умови для флори і фауни акваріума, контролювати та керувати різними підсистемами акваріума з мінімальними зусиллями, оскільки IoT-пристрій призначений для полегшення життя власника.

## 1.2. Основні функції «розумного» акваріуму

У наш час люди швидко адаптуються до нових винаходів, і споживачі все більше вимагають розробки кращих, швидкодіючих, мобільних технологій, які роблять життя зручнішим і дають можливість не бути обмеженими домашніми клопотами або відстанню. В епоху прогресуючих технологій виробники все більше прагнуть уникнути проводів і перейти на бездротову передачу даних. Тому в моєму проєкті використовується бездротова передача даних за допомогою Wi-Fi роутера. Завдяки програмному забезпеченню на смартфоні власник може керувати функціоналом акваріума та змінювати його налаштування з будь-якої точки світу.

При розробці зональної системи ІАСУ (інтерактивної автоматичної системи «розумний будинок») для акваріумів важливо враховувати специфіку цього простору. Акваріум – це прозора ємність, призначена для постійного утримання водних організмів.



Рис. 1.2. Сучасний акваріум

Зазвичай під акваріумом мається на увазі домашній (кімнатний) акваріум для утримання рибок у домашніх умовах. Через технічні обмеження максимальний обсяг такого акваріума зазвичай не перевищує одного кубічного метра. Публічні акваріуми, які призначені для демонстрації водної флори та фауни глядачам і існують у складі зоопарків або як окремі видовищно-просвітницькі заклади, можуть досягати об'єму 7500 кубічних метрів. В акваріумі можна утримувати практично будь-яких водних мешканців: морських і прісноводних риб, рослини, ракоподібних, моллюсків, земноводних, рептилій та корали.

Для підтримки біологічної рівноваги в акваріумі використовуються різноманітні пристрої: аератори, механічні та біологічні фільтри, терморегулятори та інші. Розміри акваріумів можуть коливатися від одного до кількох тисяч літрів. Вимоги до розмірів акваріумів різняться залежно від виду мешканців. Хоча чітких мінімальних розмірів не існує, є певні рекомендації. Наприклад, для риб, що швидко плавають і в природі мешкають у річках зі

швидкою течією, потрібні великі акваріуми – приблизно в 10 разів більше їхньої довжини. Скати потребують акваріумів з великою площею дна, а для інших риб важливою може бути глибина акваріума. Деякі риби, як-от півники, дуже невибагливі і можуть жити в ємностях від 0,5 літра.

Акваріуми поділяються на:

- **Біотопні акваріуми:** Відтворюють природний біотоп, такий як стояче озеро, болото, повільна річка чи озеро. В них підбирають види рослин і риб, які природно співіснують, а також максимально відтворюють параметри навколишнього середовища: склад води, ґрунту, освітлення та температура.



Рис. 1.3. Акваріум біотоп

- **Псевдоморські акваріуми:** Зазвичай не містять рослин і оформлені камінням, схожим на корали. Вода має високу жорсткість – до 20°. Для освітлення використовують лампи з холодним синім відтінком. У таких акваріумах часто мешкають цихліди африканських озер.



Рис. 1.4. Акваріум у стилі Псевдоморе



- **Голландські акваріуми:** Основна увага приділяється рослинам. В них або зовсім немає риб, або їх дуже мало. Для освітлення використовуються спеціальні лампи з підвищеною світловіддачею та особливими спектральними характеристиками. Через густу посадку рослин використовуються комплексні добрива та системи збагачення води CO<sub>2</sub>.



Рис. 1.5. Голландський акваріум

- **Морські акваріуми:** Складніші в утриманні порівняно з прісноводними. Заміну води здійснюють не так часто через високі витрати на морську сіль. Для регенерації води використовують складне обладнання. Вони мають сильну течію, білкові відділювачі (флотатори, скіммери) та сильні помпи для імітації морської течії. Морські акваріуми бувають рифовими та рибними, причому рифові вимагають складнішого обладнання для утримання морських безхребетних.



Рис. 1.6. Морський акваріум

Інтеграція інженерних функцій у просторі. Основною особливістю системи "розумного будинку" є комфорт, неповторне задоволення від сценаріїв світла та звуку, а також зручність в управлінні. Сумарне управління всіма функціями "розумного будинку" (освітленням, кліматом, аудіо- та відеоналаштуваннями) в автоматичному та інтерактивному режимах робить його зручним і безпечним, створюючи додатковий емоційний комфорт.

Інтеграція простору з загальною системою будинку. Всі елементи системи "розумного будинку" непомітно вписуються в дизайн акваріума, покращуючи його, надаючи сучасний вигляд та роблячи простір живим і змінним відповідно до вашого настрою і потреб ваших вихованців.

Ядром керуючої системи "розумного будинку" є інформаційний кабель, який з'єднує керуючі та керовані пристрої.

Опції розумного будинку. Управління освітленням в акваріумі за допомогою будь-якого зручного пристрою (настінного вимикача, сенсорної панелі, комп'ютера чи пульта дистанційного керування) дозволяє регулювати яскравість і інтенсивність світла відповідно до потреб різних видів риб. Завдяки системі "розумного будинку" можливо підключати підводне та надводне підсвічування і регулювати яскравість освітлення. Система забезпечить ідеальний рівень освітлення для акваріума.

Зручне регулювання кліматичного режиму, необхідного для мешканців акваріума, можна здійснювати за допомогою комп'ютера або дистанційного керування (вмикання нагріву чи охолодження по телефону, СМС або через Інтернет). "Розумний дім" також дозволяє контролювати температуру в акваріумі та управляти кліматом з центральної сенсорної панелі.

Практично будь-який електричний привід може керуватися через настінні панелі, сенсорні екрани, бездротові пульти, а також дистанційно.

Функція SOS у "розумному домі" (контроль витоків води, пожежний контроль, моніторинг стану інженерного обладнання) дозволяє відстежувати стан датчиків з сенсорних панелей або комп'ютера. У разі необхідності автоматично викликається фахівець служби експлуатації та проводяться заходи для забезпечення безпеки.

Параметри води будуть автоматично регулюватися системою "розумного будинку". Це включає жорсткість води, підтримання необхідного рівня та інших параметрів.

Управління пристроями. "Розумний будинок" легко управляється за допомогою кольорового сенсорного екрану на стіні, вимикачів DLT, Neo, Saturn, переносного пульта, екрана телевізора або з пристроїв HTC.

Відсутність проводів, що особливо важливо для акваріумного обладнання, забезпечується завдяки бездротовій мережі.

Сценарії управління пристроями. Мобільна форма для створення сценаріїв автоматичного та інтерактивного управління системою "розумного будинку" з додатковими базовими варіантами. Сценарії легко встановлюються, змінюються і візуалізуються на екранах системи обладнання Clipsal C-Bus.

Це можливість поєднання світла і звуку для створення певної атмосфери та зміни дизайну в приміщенні з акваріумом.

Інше обладнання. Система "розумного будинку" може бути доповнена необхідними для акваріума функціями:

- Управління сенсорним сантехнічним обладнанням;
- Контроль рівня води в акваріумі;
- Відстеження терміну служби водяних фільтрів;

- Контроль рівня реагентів для самоочищення;
- Локалізація протікань.

Акваріум – це вікно в природу, яке відтворює в мініатюрі світ тиші і спокою. Завдяки зростаючій кількості любителів акваріумістики, він ефективно допомагає знімати стрес міського життя. Залежно від ваших бажань, смаків і фінансових можливостей, можна створити екосистеми різних типів у невеликому об'ємі: ділянку коралового рифу, густо зарослий зелений куточок Амазонії та багато іншого. Неймовірно привабливий, фантастичний світ, в якому на ваших очах живуть і розмножуються дивовижні істоти, принесе вам неперевершене задоволення. Створення складних сценаріїв освітлення для вашого акваріума, а також його автоматичне очищення – це непросте завдання. Система "розумний будинок" візьме на себе ці турботи, включаючи автоматичне годування ваших морських мешканців, щоб ви не хвилювалися про їх догляд.

Кожен акваріум унікальний як маленька екосистема, і проблема збереження та підтримки балансу стає особливо актуальною для людей, які часто подорожують, багато працюють або бувають у відрядженнях. Щоб забезпечити повноцінну життєдіяльність риб без постійного втручання, необхідні вісім основних функцій "розумного" акваріуму. Розглянемо ці функції детальніше.

Для розвитку, розмноження та підтримання яскравого забарвлення риб необхідне регулярне і повноцінне харчування. У нашому акваріумі встановлено пластикову годівницю з 20 контейнерами для їжі. Годівниця підключена до годинника реального часу, і завдяки таймеру кожен контейнер відкривається у визначений час, забезпечуючи риб їжею. Нижня кришка контейнера щільно прилягає до стінок, що дозволяє залити воду та покласти живий корм, запобігаючи його псуванню з часом.

Не менш важлива функція - забезпечення подачі кисню, оскільки низька концентрація кисню у воді може спричинити загибель її мешканців. Проте постійне постачання кисню не є необхідним. Для цього компресор підключений до годинника реального часу, який вмикає та вимикає прилад. Власник може контролювати та встановлювати час подачі кисню за допомогою смартфона. Водорості виробляють кисень вночі, а вдень його поглинають, тому система



автоматично вимикає компресор на ніч. Надмірне кисневе насичення у середовищі може призвести до різних захворювань у мешканців акваріуму. Перевагою регулювання подачі кисню є також економія електроенергії.

При належному освітленні, підводний світ стає загадковим та привабливим. Вчені з Британського наукового морського акваріуму в Плімуті дослідили фізичний та психологічний вплив спостереження за рибами. Вони виявили, що спостереження за рибами сприяє зниженню кров'яного тиску і серцевого ритму, покращенню настрою та зменшенню депресії. При цьому освітлення відіграє ключову роль. Дві лампи денного світла з'єднані з годинником реального часу, який регулює їх вмикання та вимикання за допомогою таймера.

Усі екзотичні акваріумні риби пристосувалися до постійної тропічної температури навколишнього середовища, тому зміна температури води в акваріумі може призвести до захворювань і навіть загибелі. Для забезпечення необхідної температури для цих видів риб використовується датчик температури з підігрівачем. Власник встановлює потрібні параметри, а датчик вимірює температуру води, активуючи підігрівач у разі необхідності.

У акваріумі часто зустрічаються різні види риб, деякі з яких плавають на поверхні, а інші приховуються в скелях або зариваються в пісок. Кожен вид звик до свого часу харчування, тому нам необхідно створити умовний рефлекс або звичку, щоб всі риби приймали їжу в один і той же час. Зазвичай господарі акваріумів використовують постукування по склу, щоб викликати риб до харчування. Ми можемо відтворити цей умовний рефлекс за допомогою механізму, який імітує постукування по склу і залучає глибоководних риб або тих, що живуть у скелях, до прийому їжі.

Всяка система потребує візуального контролю, тому одним з ключових заходів безпеки є встановлення камери. Коли власник відсутній вдома, він може постійно насолоджуватися спостереженням за життям своїх підопічних, налаштувати свій фізичний та емоційний стан, насолоджуючись естетикою. Наша камера здатна охопити весь об'єм акваріуму; за допомогою регулювання джойстика на смартфоні власник може рухати камеру у потрібний йому кут.

Крім звичайного спостереження за мешканцями акваріуму, важливо здійснювати огляд риб для виявлення перших ознак захворювання.

Будь-яка замкнута екосистема не може залишатися стабільною, оскільки постійно накопичуються продукти життєдіяльності, що призводить до збільшення кислотності. Рівень кислотності визначається кількістю катіонів водню і аніонів гідроксиду у воді. Щоб знизити кислотність, можна здійснювати постійну аерацію цілодобово, а для її підвищення — внести кальцій. Для контролю рівня кислотності використовується рН-метр. Підвищена кислотність може призвести до захворювань, наприклад, відшарування луски.

У разі виявлення підвищення кислотності за допомогою відеоспостереження або датчика, створено спеціальний контейнер з двома відсіками. У перший відсік додається розчин метиленової зелені, а у другий — концентрат кальцію. При виявленні захворювань через камеру, власник за допомогою смартфона відкриває дозатор у першому контейнері, і потрібна кількість крапель потрапляє у акваріум. Якщо зафіксовано підвищення рН, то виконується та ж сама операція, але з другим контейнером.

### **1.3. Огляд існуючих рішень**

Догляд за акваріумами вимагає значних зусиль. Кожного разу, коли необхідно почистити акваріум або нагодувати рибку, потрібно виконати безліч дій. Спочатку треба вимкнути насос, погодувати рибку вручну, а через годину знову увімкнути насос. У традиційних системах всі пристрої, такі як світильник, обігрівач, фільтр тощо, керуються вручну за допомогою електричних перемикачів. Для цього людині доводиться підходити до акваріума і самостійно вмикати або вимикати пристрої.

Рибку необхідно годувати щодня, що вимагає від власника регулярно підходити до акваріума для їх годування, ускладнюючи догляд за акваріумом. Коли власник не вдома, він не може доглядати за системою і годувати рибку. У таких ситуаціях на допомогу приходить «Розумний акваріум». Цей проєкт є більш ефективним та корисним, ніж традиційні системи, поширені в побуті. Він орієнтований на людей, які хочуть тримати риб у домашніх умовах або в офісі, але не мають на це часу або не хочуть просити знайомих або сусідів доглядати

за рибками під час їхньої відсутності. Такі проекти є автоматизованими системами догляду за рибами, які замінюють ручний догляд автоматизованими функціями.

Різні великі компанії, що спеціалізуються на виробництві гаджетів, також створюють розумні акваріуми і досягають успіхів у їх продажах. В Україні розумні системи для дому лише починають набирати популярність і поки що не є широко розповсюдженими. Однак у Сполучених Штатах, Японії, Китаї та інших великих країнах люди вже не уявляють життя без автоматизованих систем у своїх будинках. Вони звикли до того, що більшість процесів відбуваються автоматично. Тому навіть такі пристрої, як розумні домашні акваріуми, користуються великим попитом.

Більшість розумних акваріумів керуються і моніторяться за допомогою мобільного додатка.

### **Розумний акваріум EcoQube C**

EcoQube C є інноваційним розумним пристроєм, який надає унікальну можливість насолоджуватися природою прямо у власному домі. Представлений на платформі Kickstarter, цей пристрій отримав підтримку спільноти. Основна ідея розумного акваріуму від цього виробника полягає в створенні маленької екосистеми у вигляді прозорого акваріума, який можна розмістити на вікні. Прозорість дозволяє користувачам спостерігати за життям акваріуму, його флорою та фауною, і насолоджуватися красою природи з мінімальними зусиллями на догляд.

Однією з головних переваг EcoQube C є його автономність, яка забезпечує подачу світла та поживних речовин до акваріуму без необхідності підключення до електромережі. Це робить його дуже зручним у використанні і дозволяє розмістити навіть у віддалених місцях без доступу до електрики. Крім того, EcoQube C оснащений інтегрованою фільтраційною системою, яка підтримує чистоту води, а також системою контролю параметрів води, таких як температура і рівень кисню. Це забезпечує здорове середовище для флори і фауни акваріуму.

EcoQube C має мобільний застосунок, який дозволяє користувачам віддалено керувати пристроєм та відстежувати його параметри. З його допомогою можна налаштовувати режим освітлення, отримувати сповіщення про стан води та керувати іншими параметрами, забезпечуючи оптимальні умови для цієї міні-екосистеми.

Загалом, EcoQube C є цікавим пристроєм, який об'єднує природу, технології та зручність в одному продукті. Він дозволяє насолоджуватися природою у будь-який час і в будь-якому місці, не виходячи з дому. Процес фільтрації води в акваріумі та зовнішній вигляд пристрою зображені на рисунках 1.5 та 1.6 відповідно.

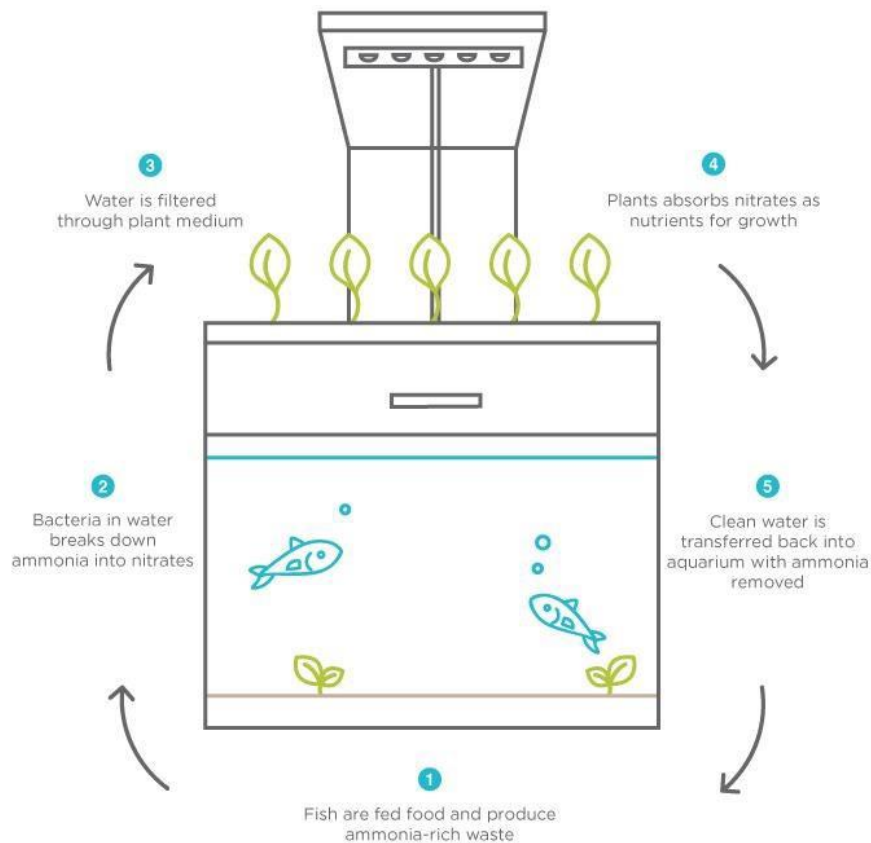


Рис. 1.6. Процес фільтрування води у EcoQube C [4]



Рис. 1.7. Зовнішній вигляд розумного акваріуму EcoQube C [4]

### **Розумний акваріум Fluval Flex**

Акваріумний комплект Fluval Flex надає все необхідне для створення привабливого акваріуму вдома чи в офісі. Виготовлений компанією Fluval Aquatics, відомою своїми високоякісними акваріумними продуктами, цей комплект має стильний та сучасний дизайн. Прозорі стінки та технологія LED-підсвічування забезпечують яскраве і рівномірне освітлення акваріумного середовища, підкреслюючи його кольори і деталі. Зовнішній вигляд акваріума та підсистема освітлення зображені на рисунках 1.8 і 1.9.

Комплект включає не тільки сам акваріум, але й додаткові компоненти, що полегшують його утримання та догляд. До них належить фільтраційна система з механічним та хімічним фільтром, яка допомагає підтримувати чистоту води, забезпечуючи належну циркуляцію та аерацію.

Flex Aquarium Kit також оснащений інтегрованою системою освітлення з LED-лампами. Ця система дозволяє створювати різноманітні світлові ефекти, включаючи налаштування яскравості та кольорової температури, що підкреслює красу рослин і рибок. Крім того, вона має вбудований контроль параметрів води,

що дає змогу користувачеві відстежувати температуру та інші важливі показники, забезпечуючи комфортні умови для життя рослин і риб.

Комплект також включає насадки для подачі води та створення потоку, що додає динамічності та живості акваріумному середовищу. Завдяки інтегрованому дистанційному керуванню, можна зручно управляти освітленням і налаштуваннями фільтрації на відстані.

Flex Aquarium Kit відрізняється компактністю та ефективністю використання простору, що робить його ідеальним вибором як для початківців, так і для досвідчених акваріумістів з обмеженим простором. Він пропонує зручне та стильне рішення для створення привабливого акваріуму з усіма необхідними компонентами.

Загалом, Fluval Flex Aquarium Kit є надійним і функціональним рішенням для створення привабливого акваріумного середовища. Він забезпечує зручність в утриманні і створює прекрасну атмосферу для спостереження за рослинами та рибками, додаючи нотку природи до життя користувача.

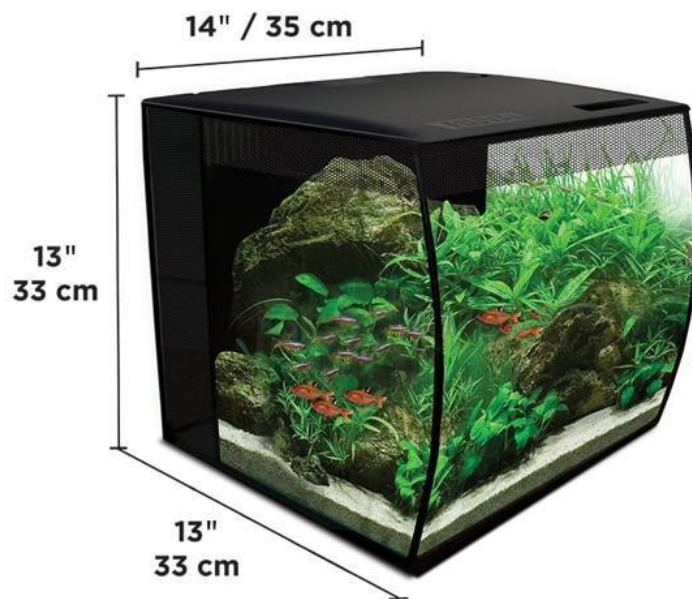


Рис. 1.8. Зовнішній вигляд розумного акваріуму Fluval Flex [5]



Рис. 1.9. Підсистема LED підсвітки разом з пультом керування [5]

### **Розумний акваріум Innovative Marine Nuvo Fusion Peninsula PRO Smart Aquarium**

Серія Nuvo Fusion Pro складається з повноцінних розумних акваріумів, розроблених компанією Innovative Marine. Ці акваріуми поєднують елегантний дизайн, функціональність та інноваційні технології, забезпечуючи комфортне та зручне утримання риб та рослин.

Однією з головних особливостей Nuvo Fusion Pro є їх модульна конструкція, що дозволяє власникам налаштовувати акваріум відповідно до своїх потреб. Вони доступні у різних розмірах та формах, включаючи класичні прямокутні та панорамні акваріуми з фронтальним панорамним склом. На рисунках 1.10 та 1.11 показано комплектацію та зовнішній вигляд продукту [6].

Кожен акваріум Nuvo Fusion Pro оснащений інтегрованою системою керування, яка дозволяє контролювати різні аспекти акваріума за допомогою мобільного додатка. Цей додаток дає можливість налаштовувати освітлення, температуру, фільтрацію, циркуляцію води та інші параметри в зручний спосіб. Крім того, він дозволяє створювати розклади освітлення, налаштовувати режими симуляції сонячного та місячного світла, а також отримувати сповіщення про важливі зміни в акваріумі.

Акваріуми Nuvo Fusion Pro також мають вбудовану систему бездротового з'єднання, яка дозволяє підключати різні аксесуари та додаткові пристрої для покращення функціональності акваріума. Серед таких пристроїв можуть бути



автоматичні дозатори, системи моніторингу параметрів води, системи автоматичного додавання добрив тощо.

Загалом, цей продукт є розумним акваріумом, який дозволяє користувачам легко керувати та налаштовувати середовище за допомогою мобільного додатка, забезпечуючи оптимальні умови для екосистеми акваріума.



Рис. 1.10. Комплектація Nuvo Fusion Pro [6]



Рис. 1.11. Зовнішній вигляд Nuvo Fusion Pro [6]



## **Система керування акваріумом Fishbit**

Fishbit є розумною системою керування акваріумом, яка пропонує зручний спосіб управління та моніторингу акваріума через мобільний додаток. Ця система розроблена для спрощення догляду за рибами та розширення можливостей акваріумної галузі.

Однією з функцій Fishbit є моніторинг параметрів акваріума. Завдяки вбудованим сенсорам, Fishbit може вимірювати температуру води, рівень рН, рівень аміаку та інші важливі параметри води. Ці дані відображаються в режимі реального часу у мобільному додатку, дозволяючи користувачу здійснювати контроль і моніторинг акваріума з будь-якого місця.

Fishbit дозволяє автоматизувати деякі процеси в акваріумі. Користувач може налаштовувати графіки освітлення, режими живлення та циркуляції води, що створює комфортні умови для акваріуму та його мешканців.

Крім того, пристрій надсилає сповіщення на мобільний телефон про важливі зміни в параметрах акваріума, наприклад, високу або низьку температуру, недостатній рівень води чи незвичайний рівень рН. Користувач також може отримати поради та рекомендації щодо догляду за рибами та підтримання комфортного середовища в акваріумі.

Fishbit дозволяє контролювати освітлення в акваріумі, налаштовуючи його відтінок та інтенсивність. Є також можливість налаштувати звукові ефекти, щоб створити різноманітну атмосферу в акваріумі.

Пристрій може бути інтегрований з іншими розумними пристроями, такими як розумний домашній автоматизаційний хаб або інші IoT пристрої, для забезпечення зручного керування всіма аспектами домашнього акваріуму.

Загалом, Fishbit є компактним, інноваційним розумним пристроєм для керування акваріумом, що пропонує широкі можливості управління та моніторингу акваріумного середовища. Він дозволяє власникам акваріумів легко доглядати за своїми рибками та створювати комфортні умови для їх здоров'я і добробуту. На рисунку 1.12 показано зовнішній вигляд системи та мобільного додатка для керування акваріумом.



Рис. 1.12 - Зовнішній вигляд пристрою та мобільного застосунку для керування акваріумом [7]

На сьогоднішній день ідеальним прикладом розумного акваріума є AquaDigitalLife. Цей акваріум може функціонувати дуже тривалий час без втручання людини. Вбудоване джерело безперебійного живлення забезпечує роботу всієї системи навіть при перебоях з електрикою, що дозволяє значно заощадити на оплаті рахунків. Контроль витоку струму допомагає уникнути ураження електричним струмом, а також вчасно виявити та усунути несправність.

Вбудований генератор забезпечує точний, швидкий і оперативний контроль електроживлення (Рис. 1.13). Автоматичний запуск генератора електричного струму та анулювання неприоритетного навантаження продовжує роботу генератора в кілька разів. Генератор можна віддалено контролювати і керувати ним через повідомлення.

Акваріум оснащений стандартними функціями, які будуть описані нижче. У нього вмонтовані два двигуни для зміни води: один працює постійно, а другий використовується як резервний. Крім того, встановлені датчики для вимірювання солоності води, значення рН та інші додаткові функції.

Монтаж цього акваріума є досить складним, і для його функціонування потрібно мінімум 5 м<sup>2</sup>. Найбільшим недоліком є висока ціна, яка становить від

10 до 15 тисяч доларів. Такі акваріуми найчастіше використовуються для вирощування декоративних морських риб та коралів.

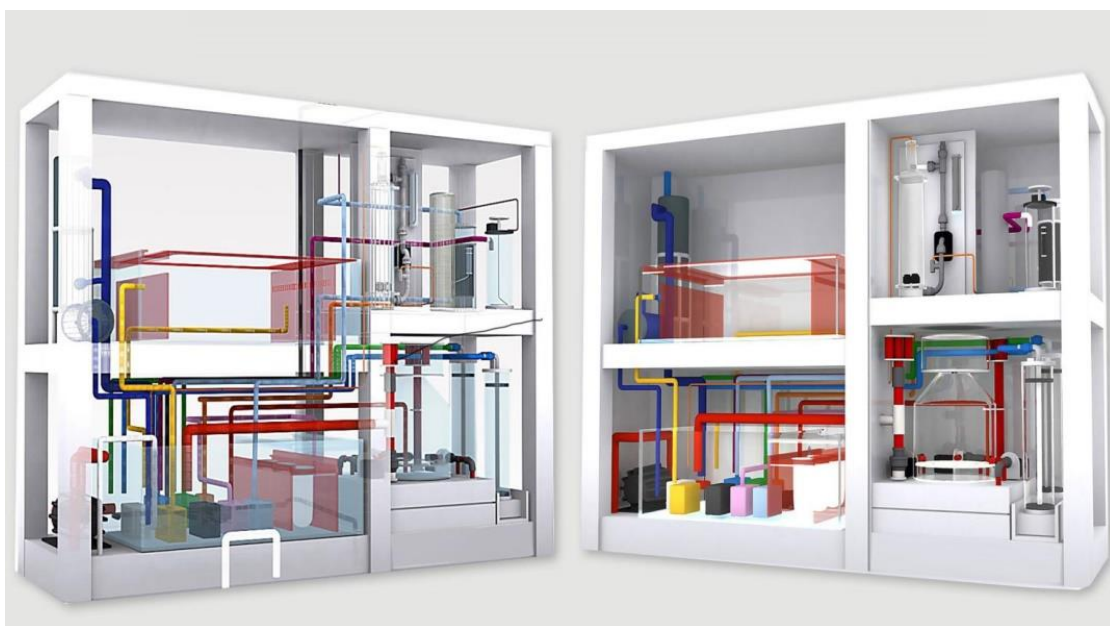


Рис. 1.13. Розумна система для акваріумів фірми AquaDigitalLife



Рис. 1.14. Система контролю живлення

Для розробки більш надійної, практичної та економічної моделі ми беремо за основу відомий розумний акваріум з прямокутним скляним резервуаром, який включає пластмасову або дерев'яну кришку. Акваріум оснащений центральним приладом для обробки даних, комплектом датчиків, регуляторів та модулем дистанційного зв'язку. Центральний процесор пов'язаний з набором датчиків і контролерів, отримуючи інформацію про стан акваріума від датчиків і передаючи керуючі сигнали на контролери, встановлені для моніторингу стану акваріума.

Модуль дистанційного зв'язку з'єднаний з центральним блоком обробки даних, а бортовий прилад споживача передає дані щодо управління до центрального блоку обробки даних через модуль дистанційного зв'язку. Комплект датчиків включає датчик температури води, датчик якості води, датчики рівня води і камеру. Рибовод може використовувати свій мобільний телефон, комп'ютер або інші мобільні пристрої для дистанційного керування акваріумом через модуль Wi-Fi, мережу Ethernet RJ45 і мережу 3G/4G, що дозволяє йому керувати акваріумом з будь-якого місця та організувати свій час більш вільно.

Недоліком такого розумного акваріуму є те, що власник не може залишати його без нагляду більше ніж на три дні. Господар пов'язаний з акваріумом, оскільки повинен кожні два дні додавати їжу в спеціальні контейнери, а в разі захворювання риб потрібно капати ліки кожні 5-8 годин.

Давайте порівняємо звичайний акваріум, існуючий прототип розумного акваріуму та експериментальну модель розумного акваріуму.

Для кращого порівняння візьмемо акваріум об'ємом 180 літрів. Звичайний акваріум може працювати без втручання людини до одного дня. Це пов'язано з необхідністю контролювати процес подачі кисню та годування риб.

Порівнюючи ціни, звичайний акваріум з фільтром, лампами, підігрівом та іншими додатками буде коштувати приблизно 100 доларів.

Розумні акваріуми стандартно комплектуються мікроконтролерами ATmega. Недоліком таких акваріумів є складність їх збирання порівняно з акваріумами на мікроконтролерах Arduino. Усі розумні акваріуми зазвичай виготовляються та збираються за кордоном, що підвищує їх вартість через додаткові витрати на транспортування. Давайте побудуємо графік приблизної вартості розумних акваріумів.

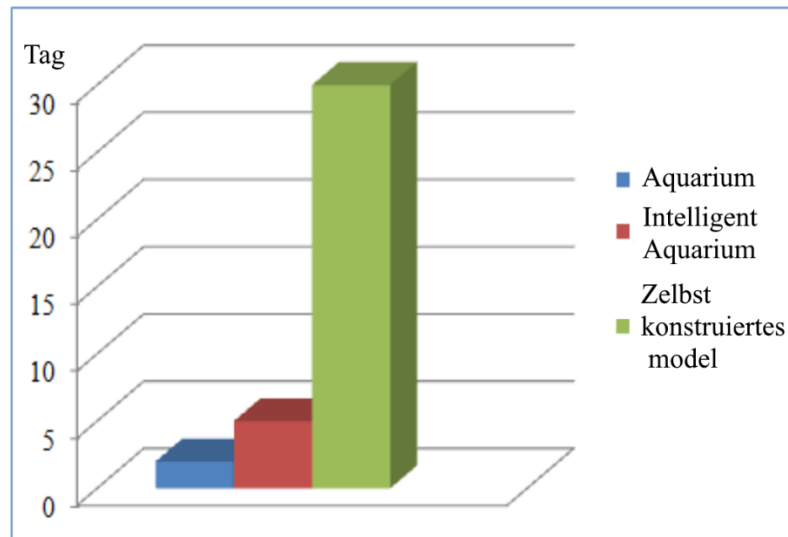


Рис. 1.15. Порівняння прототипу з існуючими акваріумами

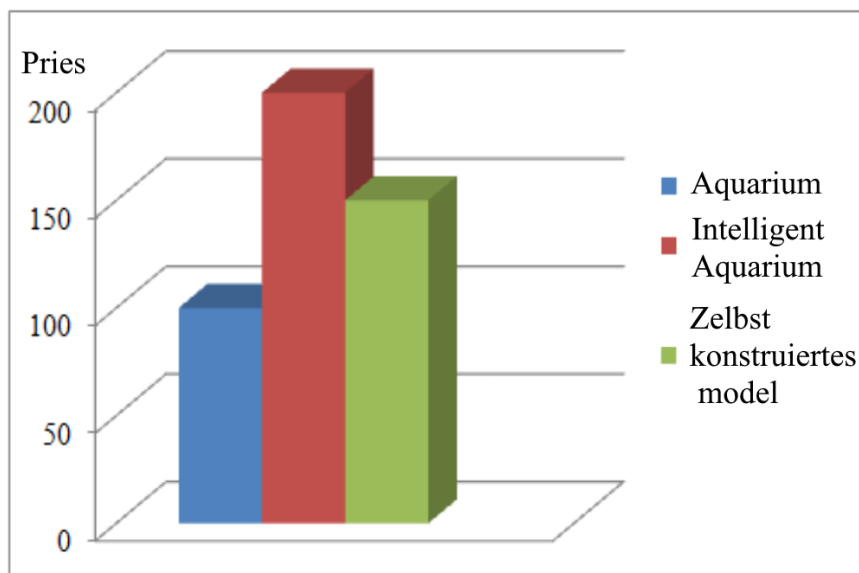


Рис. 1.16. Порівняння вартості акваріумів

Давайте проаналізуємо енергоспоживання звичайного та розумного акваріумів. Для цього ми розглянемо найпоширеніші електроприлади для акваріумів. Наприклад, у акваріумі є дві люмінесцентні лампи по 25 Вт кожна, а також блок живлення, який споживає 20% від потужності ламп. Отже, загальна потужність ламп становить 60 Вт. Враховуючи, що власник працює протягом усього дня і не може постійно вмикати та вимикати освітлення, припустимо, що лампи будуть увімкнені 14 годин на добу. За формулою (1.1) ми можемо розрахувати кількість спожитої електроенергії на освітлення акваріуму протягом доби.

$$0,060 \text{ кВт*год} * 14 \text{ год} = 0,84 \text{ кВт*год} \quad (1.1)$$

Потім ми розрахуємо витрати на фільтр. Зазвичай фільтр повинен працювати до 10 годин на добу безперервно, але на практиці ніхто не вмикає та не вимикає його кожну півгодини. Тому ми можемо припустити, що фільтр буде працювати протягом всієї доби. Зазвичай фільтри мають потужність близько 12 Вт.

$$0,012 \text{ кВт*год} * 24 \text{ год} = 0,288 \text{ кВт*год} \quad (1.2)$$

Давайте проведемо розрахунок для обігрівача акваріума. Так само, як і з іншими пристроями, власник не може постійно контролювати температуру води та вмикати чи вимикати обігрівач. Обігрівач має потужність 50 Вт, а приблизний час його роботи протягом доби становить 8 годин.

$$0,050 \text{ кВт*год} * 8 \text{ год} = 0,4 \text{ кВт*год} \quad (1.3)$$

Отже, звичайний акваріум споживає 1,752 кВт електроенергії за добу. Давайте розглянемо витрати для розумного акваріуму. Розумна система акваріуму може автоматично вмикати та вимикати прилади за необхідності, що дозволяє економити електроенергію. Для того, щоб не працювати впустую, лампи денного світла розумного акваріуму повинні працювати максимум 8 годин на добу. Крім того, в нашому акваріумі регулювання яскравості світла буде автоматично залежно від рівня сонячного світла. Ця система може зменшити витрати електроенергії на світло ще на 20-30%.

$$0,060 \text{ кВт*год} * 8 \text{ год} = 0,48 \text{ кВт*год} \quad (1.4)$$

Для забезпечення умов для життєдіяльності риб достатньо, щоб фільтр працював протягом 10 годин.

$$0,12 \text{ кВт*год} * 10 \text{ год} = 0,12 \text{ кВт*год} \quad (1.5)$$

Для забезпечення потрібної температури в акваріумі, обігрівач повинен бути включений приблизно на три години щодня.

$$0,050 \text{ кВт*год} * 3 \text{ год} = 0,15 \text{ кВт*год} \quad (1.6)$$

Також для керування та роботи приладів використовується Arduino Uno, яка, в свою чергу, споживає приблизно 0,25 кВт за добу. Таким чином, сумарна витрата електроенергії становить 1 кВт на добу. На рисунку (1.17) видно залежність витрат електроенергії для розумного акваріуму та звичайного

акваріуму протягом місяця. За допомогою розумного акваріуму власник може зекономити до 45% витрат на електроенергію.

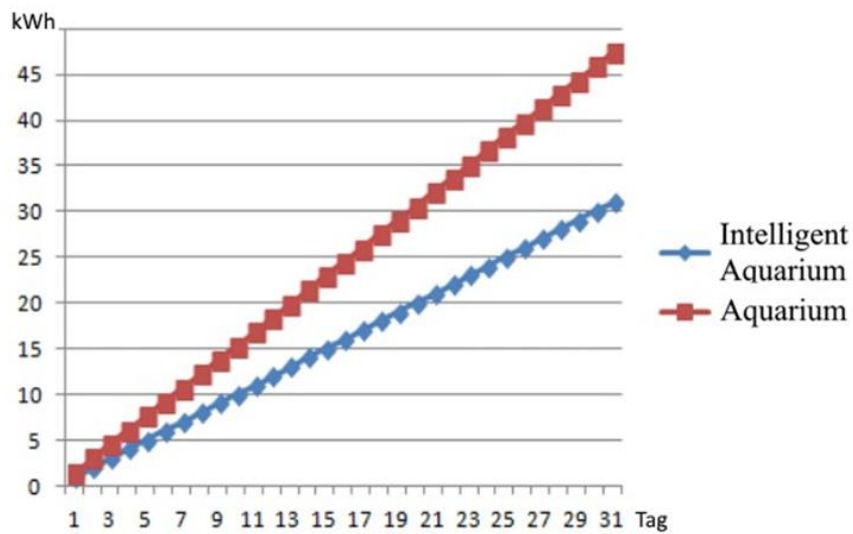


Рис. 1.17. Залежність витрат електроенергії

#### 1.4. Управління «розумним» кліматом

Постійне забезпечення оптимального клімату в розумних акваріумах є найважливішим аспектом. Це можна порівняти з автомобілями, де якість контролю за кліматом може впливати на загальний комфорт під час поїздок. Автомобілі з високоякісними системами клімат-контролю здатні врахувати різноманіття умов і забезпечити комфорт пасажирів навіть під час зміни кліматичних умов. Інтелектуальні системи також дбають про здоров'я пасажирів, намагаючись запобігти захворюванням. Багато автовиробників, таких як Lexus, BMW, Mercedes, Audi, Mazda, Porsche, Mitsubishi, Honda, Land Rover, Subaru, Toyota, Volvo, Nissan, Volkswagen, обладнували свої автомобілі індивідуальним керуванням кліматом для водія та пасажирів. Аналогічно, вимоги до якості клімату є важливими для нашого домашнього середовища та офісів, де ми проводимо більшу частину часу. Якість клімату безпосередньо впливає на наше самопочуття, настрій і здоров'я. Хоча люди можуть звикнути до різних кліматичних умов, наші домашні улюбленці менше адаптовані до таких змін.

- Індивідуальне налаштування параметрів.
- Спрямована робота фільтра, компресора та освітлення.
- Просте налаштування температурних графіків.
- Економія, навіть без усвідомлення.

Зниження температури на 1 градус може призвести до економії до 6% енергії. Контроль клімату передбачає вимірювання та підтримку параметрів води в акваріумі: температури, рівня кисню, хімічного складу. Для цього використовуються різні системи: обігрівачі, охолоджувачі, вентиляційні системи, кондиціонери і т.д.

Отже, для інтегрованої системи клімат-контролю важливо забезпечити гармонійне управління всіма цими пристроями. Оркестровим диригентом цього процесу є "розумний акваріум", чия одна з ключових функцій полягає в клімат-контролі. При такому співробітництві різні системи працюють узгоджено, не заважаючи одна одній. Наприклад, система підігріву води не активується, коли відбувається вентиляція. При розумному управлінні пристрої реагують по-різному на різні ситуації: при відкритих або закритих вікнах, при присутності або відсутності людей у приміщенні та інших обставинах.

Додамо сюди можливість програмного управління кліматом за графіком, який може бути гнучко налаштований для різних днів тижня або з урахуванням вподобань власника. Також врахуємо можливість дистанційного керування кліматом через Інтернет або мобільний телефон. Необхідно також пам'ятати про автоматичне використання для опалення різних джерел енергії залежно від зміни тарифікації протягом доби. На завершення варто згадати про оптимізацію роботи обладнання з метою збереження енергії. Очевидно, що енергозбереження залежить як від клімату країни, так і від інших факторів і може сягати навіть 50%. Усе це надає розумний акваріум, віднімаючи від людини всі труднощі та надаючи можливість насолоджуватися природою у власному будинку.

Найбільш розумна серед них працює за принципом моніторингу інформації, і якщо будь-який параметр відхиляється від норми, то приладами автоматично коригується до досягнення бажаних значень. Давайте розглянемо більш детально принцип функціонування цього розумного середовища, зокрема процес зчитування інформації за допомогою датчиків.

Для швидкого визначення рівня рН (тобто кислотності) різних середовищ застосовуються рН-метри. Вони можуть використовуватися для вимірювання рівня кислотності в технічній або питній воді, розчинах кислот, солях або лугах,



а також в різних рідинах організму, таких як кров чи сеча. Також вони застосовуються для вимірювання кислотності у фруктах, овочах, харчових продуктах, медичних препаратах і багато іншого. Фактично будь-який рідкий чи розчинний зразок може бути об'єктом оперативного вимірювання за значенням рН.

Вимірювання рН, по суті, є оцінкою активності іонів водню у середовищі. Навіть сам термін "рН" в перекладі з латинської мови означає "pondus Hydrogenii", що буквально означає "вага водню".

На сьогоднішній день рН-метри широко застосовуються в різних галузях, таких як мікробіологія, медицина, водопідготовка, агрохімія, ґрунтознавство, гідропоніка, лабораторні та польові дослідження, хімічна та харчова промисловість, акваріумістика і багато інших.

Сучасні рН-метри дозволяють швидко та точно вимірювати значення рН. У нейтральному середовищі, наприклад, дистильована вода, рН рівний 7, де позитивні іони водню  $H^+$  та негативні гідроксид-іони  $OH^-$  у рівновазі. Якщо значення рН перевищує 7, це вказує на лужність середовища, а якщо менше 7 - на його кислотність.

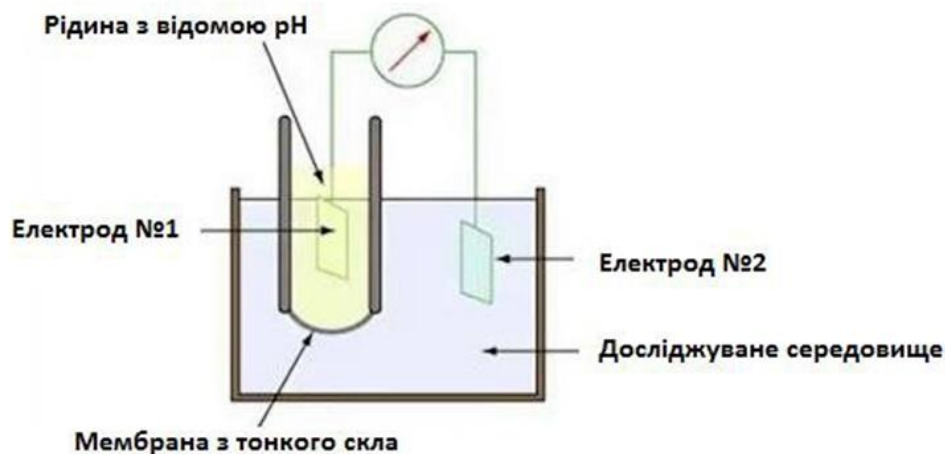


Рис. 1.18. Принцип дії рН-метрів

рН-метр насправді є електронним мілівольтметром, оскільки він вимірює різницю потенціалів в електрохімічній системі з парою електродів та досліджуваного середовища, в яке вони занурені. Однак шкала приладу не градується в мілівольтах, а в одиницях рН, оскільки виміряний ЕРС відноситься

до рівня рН. У цій системі два електрода: скляний індикаторний (який володіє великим опором у десятки мегаом) і хлорсрібний, що виступає як додатковий електрод порівняння. Для градування рН-метра використовують буферні розчини з відомим рівнем рН. Так як температура впливає на величину ЕРС, багато рН-метрів мають термокомпенсацію для вимірювань при різних температурах. Однак для досягнення високої точності вимірювань рекомендується проводити їх при температурі  $+25\text{ }^{\circ}\text{C}$ , тому багато рН-метрів оснащуються вбудованим термометром для відстеження температури досліджуваного середовища.

Індикаторний скляний електрод має форму трубки з тонкостінними стінками і кульковим закінченням. Виготовлений з особливого електропровідного боросилікатного скла, він включається в електричний ланцюг. Внутрішні переміщення позитивних іонів  $\text{H}^{+}$  всередині такого скла (катіони в скляній матерії пересуваються в напрямку поліаніонного кремнієвого скла). У трубку наливається суспензія хлориду срібла у розчині соляної кислоти, після чого до неї опускається срібний дріт - це формує хлорсрібний електрод. Скляний електрод опускають у досліджувану середу, замикаючи електричний ланцюг і вводячи (через електролітичний ключ або безпосередньо) додатковий електрод. Зазвичай цей додатковий електрод розташовують у скляному корпусі, який не пропускає воду для іонів  $\text{H}^{+}$ .

Контакт між розчином хлориду калію у електроді порівняння та досліджуваним розчином виникає за допомогою тонкої нитки або капіляри в скляному корпусі. Це призводить до утворення гальванічного елемента між електродом порівняння та хлорсрібним електродом. Електролітична частина такого елемента включає провідну скляну плівку і досліджувану середу. Різницю потенціалів електродної системи (ЕРС) вимірюється мілівольтметром, чий шкала градуйована в рН. Електрони з хлорсрібного електрода переносяться до електрода порівняння під впливом вимірюваної ЕРС, що супроводжується перенесенням рівної кількості протонів з внутрішньої сторони скляного електрода в середу. Якщо прийняти концентрацію позитивних іонів водню  $\text{H}^{+}$  всередині скляного електрода постійною, то ЕРС стане функцією активності  $\text{H}^{+}$

+, тобто рН досліджуваного середовища. Сучасні моделі рН-метрів працюють за допомогою мікропроцесорів, які виконують термокомпенсацію. Чим складніший пристрій, тим більше завдань він може вирішувати. Клас точності приладів змінюється від моделі до моделі, і для різних сфер застосування можна підібрати відповідний рН-метр. Є кишенькові побутові рН-метри, лабораторні, портативні та промислові стаціонарні. Деякі з них вимірюють концентрацію іонів у середовищі, вміст нітратів тощо, мають вбудовану пам'ять для збереження результатів, можливість зв'язку з комп'ютером та функцію корекції параметрів за допомогою ланцюга зворотного зв'язку. Також вони можуть бути оснащені датчиком температури.

Терморезистори - це пристрої, чий опір змінюється зі зміною температури. Проте не всі пристрої, що реагують на температуру зміною опору, отримали назву "терморезистори". Наприклад, резистивні термометри, виготовлені з крученого дроту або металевих плівок, хоча залежать від температури, але працюють іншим чином, ніж терморезистори. Термін "терморезистор" зазвичай використовується для напівпровідникових пристроїв, які чутливі до температури. Терморезистори з негативним температурним коефіцієнтом (ТКО) виготовляються з напівпровідникових матеріалів, таких як кераміка, створена з суміші металевих оксидів. Терморезистори застосовуються у багатьох галузях: в системах протипожежної безпеки, для вимірювання та регулювання температури, контролю за теплом, компенсації температури, вимірювання потужності ВЧ. Вони також широко використовуються в промисловій електроніці, побутових пристроях, медицині, метеорології, хімічній промисловості та інших галузях. Деякі терморезистори, такі як ММТ-4 і КМТ-4, мають металеві капсули і герметичне ущільнення, що дозволяє їм працювати в умовах вологості і навіть у рідинах, які не агресивні до корпусу терморезистора.



Рис. 1.19. Герметизований датчик температури ММТ-4

Температурна залежність опору є ключовою характеристикою терморезисторів і в значній мірі визначає інші параметри цих пристроїв. Ця залежність подібна до температурної залежності питомого опору напівпровідника, з якого виготовлений терморезистор. Обігрівач води для акваріума, який також називається терморегулятором, це пристрій, призначений для нагрівання води, обладнаний вбудованим регулятором. Він складається зі скляної трубки з нагрівальним елементом. Терморегулятори автоматично вимикаються при досягненні заданого рівня тепла і включаються, коли температура знижується нижче необхідного значення.

Діапазон роботи обігрівачів для акваріума зазвичай становить 18-32 градуси за Цельсієм. При виборі обігрівача значну роль відіграє його потужність. Раніше було поширене прийняття, що оптимальна потужність обігрівача - це 1 ват на 1 літр води, але ця концепція не завжди відповідає реальності, так само як і рекомендація про 1 см риби на 1 літр води. На жаль, деякі виробники не завжди надають достатньо точної інформації про рекомендовані об'єми акваріумів для своїх обігрівачів. Таким чином, під час вибору обігрівача важливо враховувати реальні умови та потреби вашого акваріума.

Зазвичай власники акваріумів купують обігрівачі з вищою потужністю, ніж потрібно для їхнього акваріума, щоб забезпечити достатнє обігрівання. Однак це може призвести до перегріву води, особливо якщо температура навколишнього середовища вже висока. Для запобігання перегріву можна скористатися вбудованим регулятором температури у більш потужних обігрівачах, але найефективнішим рішенням може бути самостійне створення обігрівача, де можна управляти температурою води за допомогою водяного датчика і мікропроцесора. Така система автоматично вимикає обігрівач, коли досягається потрібний рівень тепла, щоб уникнути перегріву води в акваріумі.

Для створення такого пристрою знадобляться наступні матеріали та інструменти:

- Силіконова трубка або капельник;
- Трансформаторна обмотка або дроти різного типу;
- Джерело живлення з напругою від 12 до 24 вольт;

- Дві пластикові заглушки для трубки;
- Силікон для герметизації;
- Рідкий гліцерин;
- Паяльник і кусачки.

Спочатку потрібно розрахувати потужність обігрівача з розрахунку стандарту: 0,5-1 Вт на літр води. Чим тепліше потрібна вода, тим потужнішим має бути нагрівач. Розрахунок здійснюється за формулою:

$$W = w \cdot V, \quad (1.7)$$

Де  $W$  - це бажана потужність нагрівача,  $w$  - обрана потужність на один літр води, а  $V$  - об'єм акваріуму.

Тобто, якщо обсяг вашого акваріуму становить 20 літрів, а ви плануєте нагрівати його в середньому, то можна взяти середню потужність (0,75 Вт/л). Отже, отримаємо:  $0,75 \cdot 20 = 15$  Вт.

Тепер потрібно розрахувати довжину провідника від трансформаторного намотування. Для цього визначимо потрібний опір за формулою:

$$R = U \cdot \frac{U}{W}, \quad (1.8)$$

Де  $R$  - це шуканий опір,  $U$  - напруга джерела струму (12 В або 24 В),  $W$  - потрібна потужність.

Для прикладу, якщо вам потрібен обігрівач потужністю 15 Вт при напрузі джерела 12 В, то опір буде дорівнювати  $12 \cdot 12 / 15 = 9,6$  Ом.

Тепер нам знадобиться формула для розрахунку довжини самого проводу:

$$L = S \cdot \frac{R}{\rho}, \quad (1.9)$$

Де  $L$  - це шукана довжина,  $S$  - поперечний переріз проводу,  $\rho$  - питомий опір матеріалу, з якого він виготовлений.

У трансформаторній намотці товщина 0,3 мм. Отже, поперечний переріз буде рівний площі кола такого ж діаметру, тобто 0,07 квадратних міліметрів. Обмотка виготовлена з міді, питомий опір якої відомий і становить 0,018 Ом \* кв. мм.

Якщо нам потрібний опір у 9,6 Ом, підставляючи відомі значення в формулу, отримаємо:  $0,07 * 9,6 / 0,018 = 37,3$  метри.

Потім переходимо до збирання обігрівача власноруч. Вставляємо дріт в середину трубки і припаюємо його кінці до проводу, що йде до джерела струму. Місце спайки розташовуємо в заглибленні з пластика і заливаємо силіконом, щоб забезпечити герметичність трубки з одного кінця. Через інший кінець трубки наливаємо рідкий гліцерин. У разі його відсутності можна використовувати воду, але вона менш ефективно проводить тепло. Щоб забезпечити герметичність іншого кінця трубки, використовуємо другу заглибку і трохи силікону.

Тепер обігрівач можна використовувати відповідно до призначення: опускаємо його на дно і підключаємо до джерела живлення.

### **Висновки до розділу**

Проаналізувавши область та ринок інтелектуальних пристроїв для тварин, можна зауважити, що цей сектор останнім часом стрімко розвивається і стає дедалі популярнішим серед власників домашніх улюбленців.

Інтелектуальні гаджети для тварин стають необхідною складовою частиною їхнього життя, допомагаючи власникам стежити за здоров'ям та добробутом своїх улюбленців, забезпечуючи розваги та автоматизуючи деякі аспекти догляду. Головною метою дипломного проекту є розробка автоматизованої системи управління інтелектуальним акваріумом, для чого були розглянуті наявні рішення та обрані оптимальні функції для створення власного пристрою.

Отже, інтелектуальні акваріуми є зручним засобом догляду за декоративними рибками, що суттєво зменшує потребу в активному втручанні людини у процес догляду та підтримки акваріуму. Вони дозволяють власникам відслідковувати та керувати якістю води, автоматизувати процес годування, підсвічування тощо.

## РОЗДІЛ 2. РОЗРОБКА ПРИНЦИПОВОЇ СХЕМИ ПРИСТРОЮ

### 2.1. Розробка та обґрунтування алгоритму функціонування та структурної схеми пристрою

Будь-яке сучасне інтелектуальне середовище неможливе без мікроконтролерів. У нашому випадку цю роль виконує Arduino Uno R3.

Arduino Uno Rev3 - це плата на основі мікроконтролера ATmega328P. Вона має 14 цифрових пінів для входу/виходу, 6 з яких можуть працювати як ШІМ-виходи, 6 аналогових входів, 16 МГц кварцовий генератор, USB-роз'єм, силовий роз'єм, ICSP-роз'єм і кнопку перезавантаження. Для роботи плату потрібно підключити до комп'ютера за допомогою USB-кабелю або жити від адаптера AC/DC чи батареї.

На відміну від інших плат Arduino, в Uno для перетворення інтерфейсів USB-UART використовується мікроконтролер ATmega16U2 (ATmega8U2 до версії R2) замість мікросхеми FTDI. У китайських версіях використовується перетворювач інтерфейсів USB-UART CH340G.

На платі Arduino Uno версії R2 для спрощення процесу оновлення прошивки додано резистор, що підтягує до землі лінію HWB мікроконтролера 8U2.

Зміни в платі версії R3 включають:

- Додано контакти SDA і SCL (біля виведення AREF) та два нових виведення біля виведення RESET.
- Введено контакт IOREF, який дозволяє платам розширення підлаштовуватися під робочу напругу Arduino. Цей контакт забезпечує сумісність плат розширення як з 5В Arduino на базі мікроконтролерів AVR, так і з 3.3В платами Arduino Due. Другий новий вивід наразі не використовується і зарезервований для майбутніх потреб.
- Покращено стійкість ланцюга скидання.
- Мікроконтролер ATmega8U2 замінено на ATmega16U2.

На сьогоднішній день на ринку представлено безліч різновидів плат Arduino. Найпопулярнішими конкурентами моделі Uno є плати Nano і Mega.

Nano ідеально підходить для проектів, де важливий компактний розмір, тоді як Mega підходить для складних схем, що потребують великої кількості виходів.

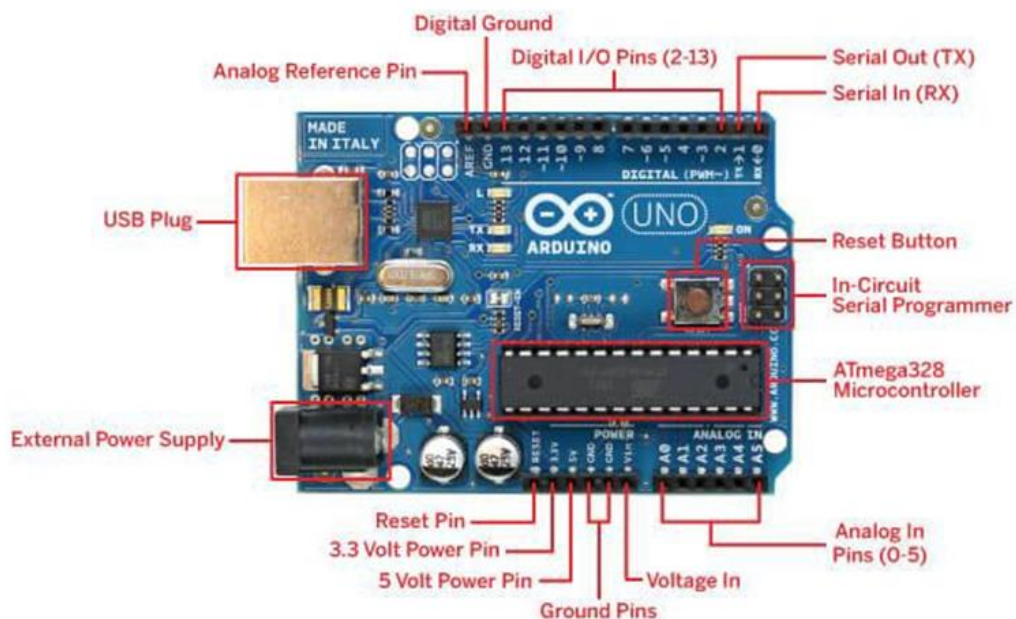


Рис. 2.1. Опис елементів плати Arduino Uno Rev3

- **USB Plug** – роз'єм для підключення USB-пристроїв;
- **Analog Reference Pin** – для встановлення опорної напруги АЦП;
- **Digital Ground** – земляний контакт;
- **Digital I/O Pins (2-13)** – цифрові входи/виходи;
- **Serial OUT (TX)** – пін для передачі даних через UART;
- **Serial IN (RX)** – пін для прийому даних через UART;
- **Reset Button** – кнопка перезавантаження мікроконтролера;
- **In-Circuit Serial Programmer (ISP)** – контакти для перепрограмування плати;
- **ATmega328P Microcontroller** – сам мікроконтролер, що виконує функції процесора;
- **Analog In Pins (0-5)** – аналогові входи;
- **Voltage In** – вхід для живлення від зовнішнього джерела;
- **Ground Pins** – земляні контакти;
- **5 Volt Power Pin** – живлення 5 В;
- **3 Volt Power Pin** – живлення 3.3 В;
- **Reset Pin** – вхід для перезавантаження;
- **External Power Supply** – роз'єм для підключення зовнішнього джерела живлення.

Піни Arduino використовуються для підключення зовнішніх пристроїв і можуть працювати як у режимі входу, так і у режимі виходу. Кожен пін має



навантажувальний резистор (за замовчуванням відключений) на 20-50 кОм і може пропускати до 40 мА струму.

Деякі піни мають спеціальні функції:

- Піни 0 і 1 – UART контакти (RX і TX).
- Піни 10-13 – SPI контакти (SS, MOSI, MISO і SCK).
- Піни A4 і A5 – I2C контакти (SDA і SCL).

Аналогові піни Arduino Uno Аналогові піни Arduino Uno Rev3 призначені для підключення аналогових пристроїв і служать входами для вбудованого аналогово-цифрового перетворювача (АЦП), який у Arduino є десятибітним.

Таблиця 2.1

### Аналогові піни

Пін	Адресація	Спеціальне призначення
0	A0 або 14	
1	A1 або 15	
2	A2 або 16	
3	A3 або 17	
4	A4 або 18	I2C (SDA)
5	A5 або 19	I2C (SCL)

### Цифрові піни плати Uno

Піни з номерами від 0 до 13 є цифровими. Це означає, що на них можна зчитувати і подавати тільки два види сигналів: HIGH і LOW. Завдяки ШІМ цифрові порти також можна використовувати для керування потужністю підключених пристроїв.

## Цифрові піни

Пін	Адресація	Спеціальне призначення	ІІМ
0	0	RX	
1	1	TX	
2	2	Вхід для переривань	
3	3	Вхід для переривань	ІІМ
4	4		
5	5		ІІМ
6	6		ІІМ
7	7		
8	8		
9	9		ІІМ
10	10	SPI(SS)	ІІМ
11	11	SPI(MOSI)	ІІМ
12	12	SPI(MISO)	
13	13	SPI(SCK) до виходу також приєднаний вбудований світлодіод (є в більшості плат Arduino)	

## Додаткові піни на платі

AREF – видає опорну напругу для вбудованого АЦП і може керуватися функцією `analogReference()`. RESET – подача низького рівня сигналу на цей пін перезавантажує мікроконтролер. Зазвичай використовується для підключення кнопки перезавантаження на платі розширення, яка може закривати доступ до кнопки на самій платі Arduino.

## Живлення Arduino Uno Rev3

Плата Arduino Uno R3 може отримувати живлення через USB або від зовнішнього джерела живлення, і джерело вибирається автоматично.

Способи живлення плати:

- **Від зовнішнього адаптера** - рекомендована напруга від 7 до 12 В. При напрузі вище 12 В регулятор напруги може перегрітися і пошкодити плату. При напрузі нижче 7 В вивід 5V може видавати менше 5 В, що спричинить нестабільну роботу плати.

- **Від USB-порту комп'ютера.**
- **Подача 5 В безпосередньо на пін 5V** - у цьому випадку обхідний вхідний стабілізатор не використовується, і навіть найменше перевищення напруги може призвести до проблем з пристроєм.

Виводи живлення:

- **5V** – подає 5 В, можна використовувати для живлення зовнішніх пристроїв.
- **3.3V** – подає 3.3 В від внутрішнього стабілізатора.
- **GND** – вивід землі.
- **VIN** – пін для подачі зовнішньої напруги.
- **IREF** – пін для інформування зовнішніх пристроїв про робочу напругу плати.

### **Підключення пристроїв**

Підключення будь-яких пристроїв до плати здійснюється шляхом приєднання до контактів, розташованих на платі контролера: до одного з цифрових або аналогових пінів, або до пінів живлення. Простий світлодіод можна підключити, використовуючи два контакти: землю (GND) і сигнальний (або пін живлення).

Найпростіший датчик вимагає мінімум трьох контактів: два для живлення і один для сигналу.

При будь-якому варіанті підключення зовнішнього пристрою важливо пам'ятати, що використання плати як джерела живлення можливо тільки в тому випадку, якщо пристрій не споживає більше дозволеного граничного струму контролера.

### **Пам'ять Arduino Uno R3**

Плата Uno за замовчуванням підтримує три типи пам'яті:

- **Flash** – пам'ять об'ємом 32 кБ. Це основне сховище для команд. Під час прошивки контролера своїм скетчем, він записується саме сюди. З цієї пам'яті 2 кБ відводяться на bootloader-програму, яка відповідає за ініціалізацію системи, завантаження через USB і запуск скетчу.

- Оперативна SRAM пам'ять об'ємом 2 кБ. Тут за замовчуванням зберігаються змінні і об'єкти, створені в ході роботи програми. Ця пам'ять є енергозалежною, тому всі дані зітруться при вимкненні живлення.
- Незалежна пам'ять (EEPROM) обсягом 1 кБ. Тут можна зберігати дані, які залишаться після вимкнення контролера. Проте процедура запису і зчитування EEPROM вимагає використання додаткової бібліотеки, доступної в Arduino IDE за замовчуванням. Також слід пам'ятати про обмеження циклів перезапису, характерних для технології EEPROM.

Деякі модифікації стандартної плати Uno можуть мати розширену пам'ять, що дозволяє зберігати більші обсяги даних, ніж в стандартному варіанті. Проте використання цих модифікованих плат вимагає встановлення додаткових бібліотек.

Екран WH1602, що є невід'ємною частиною багатьох електронних пристроїв, відіграє важливу роль у відображенні та графічному виведенні даних. Для користувача завжди зручніше і приємніше, коли результат роботи "розумної коробочки" можна візуально спостерігати. У даній дипломній роботі нашим експериментальним об'єктом стане досить популярний дисплей WH1602 від Winstar.

### **Характеристика екрану WH1602**

- Розміри: 80 x 36 мм
- Робоча температура: 0-50°C
- Підсвічування: блакитна
- Колір символів: білий
- Розмір символу: 4,35 x 2,95 мм
- Формат: 16 x 2
- Розмір точки: 0,5 x 0,5 мм
- Інтерфейс: HD44780
- Видима область: 64,5 x 13,8 мм

- Напруга живлення: 5В

Підключення дисплея WH1602 починається з його прямого з'єднання з контролером. Для цього ми користуємося документацією та шукаємо в ній розпіновку для WH1602. Дисплей WH1602 має 16 виводів, які ми розглянемо докладніше.

Піни Vss, Vdd і K мають бути підключені до землі і джерела живлення відповідно, відповідно до вказівок у таблиці.

Вивід під номером 3 використовується для регулювання контрастності. Підвищення напруги до +5 В призведе до того, що екран буде абсолютно темним, тоді як коротке з'єднання землею призведе до відображення двох рядків чорних квадратів. Щоб досягти оптимальної видимості символів, на цей вивід дисплея повинно бути підведено напругу у межах 0.5-0.7 В за допомогою потенціометра (регульованого резистора) для контрастності.

Пін RS призначений для управління дисплеєм з боку мікроконтролера. Низький рівень напруги (0) на цьому виводі вказує, що буде передана команда, тоді як високий рівень (1) позначає передачу даних для запису в пам'ять дисплея.

Пін R / W використовується для читання або записування даних на дисплей. Коли на цьому виводі встановлено значення 1, ми зчитуємо дані, а коли встановлено значення 0, ми записуємо команду або дані на дисплей.

DB7 – DB0 представляють шину даних.

Пін E відомий як сигнал Enable. Для того щоб взаємодіяти з дисплеєм - надсилати дані або команди - потрібно надіслати позитивний імпульс на цей вивід.

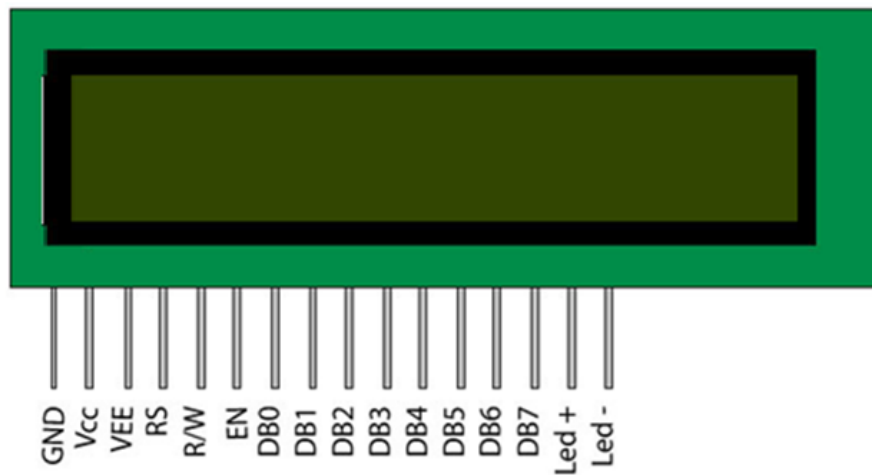


Рис. 2.2. Розташування пінів для екрану WH1602

Отже, процедура виглядатиме наступним чином:

1. На пінах RS, R / W, DB7 - DB0 - подаємо сигнали, відповідні нашій команді.
2. Подаємо логічну одиницю на вивід E.
3. Очікуємо (згідно з даташитом - не менше 150 наносекунд).

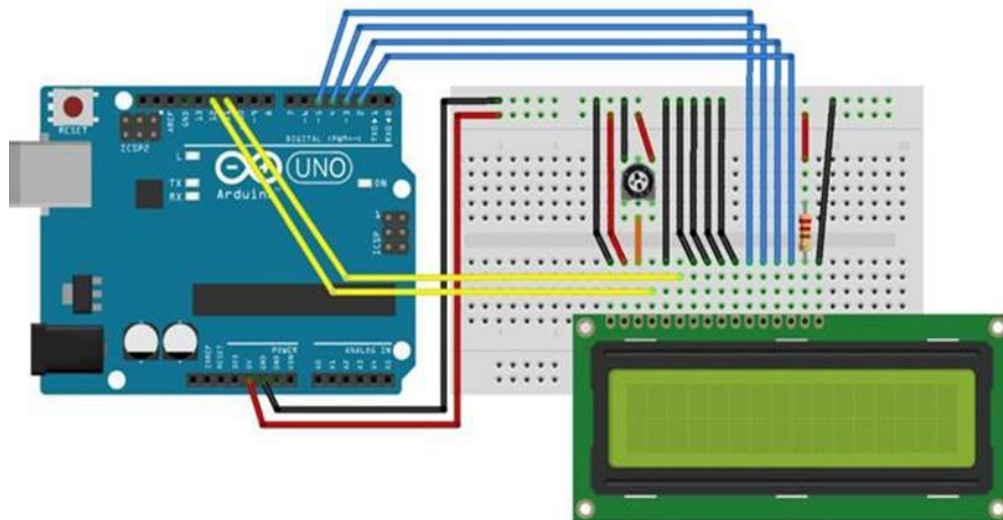


Рис. 2.3. Підключення LCD-екрану до Arduino

4. Подаємо на вивід E низький рівень (0).
5. На ніжку A / Vee треба подати 4.2 В для живлення підсвічування дисплея.

За допомогою потенціометра, показаного на рисунку 2.3, можна налаштовувати контрастність зображення на екрані.

Управління освітленням. У цьому проєкті було вирішено обладнати акваріум світлодіодами, щоб вони світилися, коли кімната стає темною. Для цього спочатку потрібно придбати світловий датчик. Принцип його роботи досить простий: чим більше світла, тим менший опір має датчик. У темряві опір значно зростає. Потім датчик повинен бути підключений до аналогового порту Arduino. Таким чином, аналоговий пін перетворює отриману напругу від 0 до 5 В в значення від 0 до 1024. За допомогою програмування ці значення перетворюються в відсоток яскравості від 0 до 100%.

Для проведення проводки ми використовували резистор як дільник напруги. Для даного типу датчика ми обрали опір 10 кОм. У програмуванні ми встановили значення (наприклад, 20%), при досягненні якого реле переходить із стану "нормально відкритий" у "нормально закритий", щоб жити світлодіоди. Проте варто відзначити, що цю систему можна покращити, оскільки користувач може не завжди бажати, щоб світлодіоди запаливалися, коли кімната темна. Для досягнення цього можна встановити перемикач, який дозволить користувачеві вільно керувати включенням або виключенням світлодіодів.

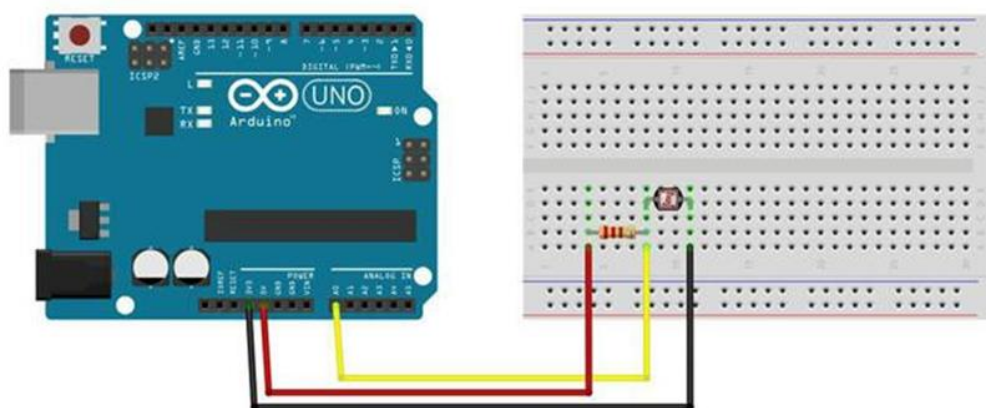


Рис. 2.4. Підключення датчику світла до Arduino

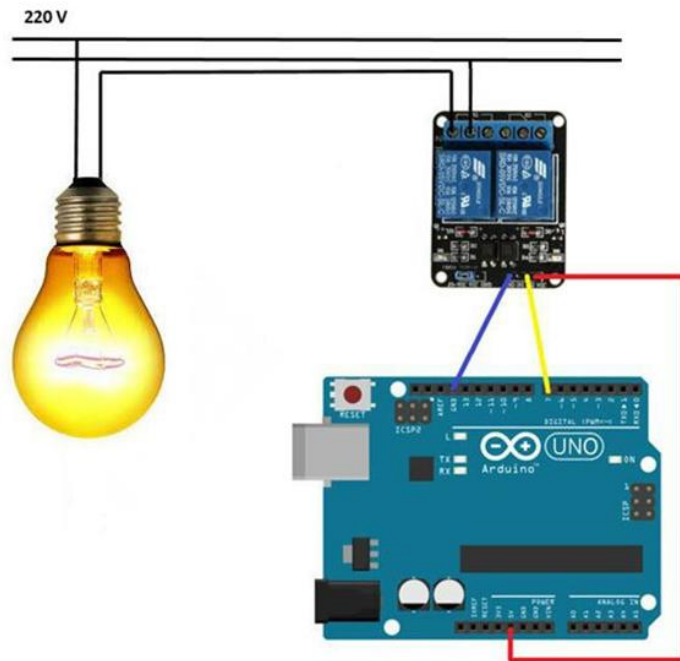


Рис. 2.5. Підключення світла до Arduino за допомогою реле

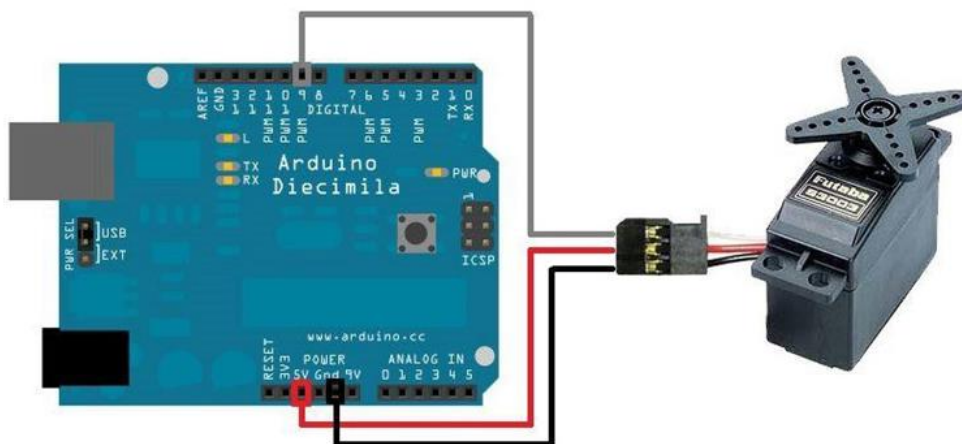


Рис. 2.6. Підключення серводвигуна

Серводвигун - це механізм, який забезпечує точне керування кутами повороту. У нашому проєкті я вирішив використати цей компонент для автоматичного подавання їжі рибам.

Ідея полягає в тому, що ми закріплюємо пластикову тару на валу серводвигуна. Таким чином, з можливістю повороту на  $180^\circ$ , тара буде обертатися, і їжа буде випадати через отвори в акваріум. Після певного часу привід повертається назад на  $180^\circ$ . Залежно від необхідної кількості їжі, яку ми хочемо подати рибам, можна варіювати кількість обертів двигуна.



Серводвигун має трьохконтактне підключення. Перший контакт призначений для заземлення. На другий контакт подається напруга 5 В. А на третій контакт подається сигнал управління. Коли на третій контакт подається логічна одиниця, серводвигун починає роботу. Коли сигнал зникає, серводвигун припиняє працювати.

Годинник реального часу базується на мікросхемі DS3231N. Резисторна збірка RP1 (4.7 кОм) використовується для підтягування ліній 32K, SQW, SCL і SDA. Друга збірка резисторів необхідна для підтягування ліній A0, A1 і A2, які використовуються для зміни адресації мікросхеми пам'яті AT24C32N. Резистор R5 і діод D1 використовуються для підзарядки батареї, проте їх можна випаяти, оскільки звичайної батарейки SR2032 вистачає на довгий час. Також на платі встановлена мікросхема пам'яті AT24C32N, яка може бути використана як бонус для роботи годинника RTC DS3231N. Резистор R1 і світлодіод Power використовуються для сигналізації про включення модуля. Так як модуль працює по шині I2C, для зручності шини виведені на два роз'єми J1 і J2.

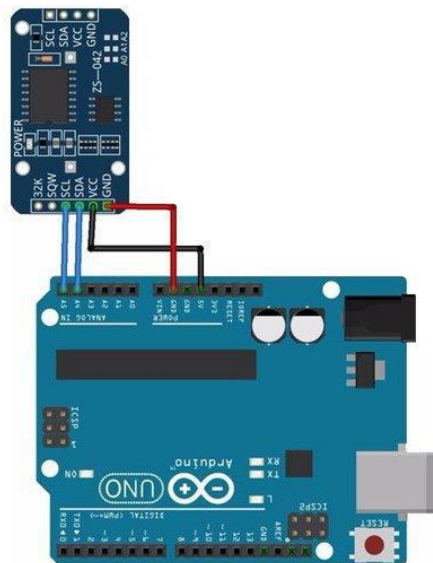


Рис. 2.7. Підключення RTC до Arduino Uno

Годинники підключаються за допомогою двохпроводної шини ІІС (Inter-Integrated Circuit) через виводи SDA і SCL до відповідних виводів SDA і SCL Arduino. Крім того, необхідно забезпечити живлення +5В і підключити землю (GND). Решта виводів, специфічних для DS3231, не підтримуються бібліотеками для DS1307.

Датчик температури підключається наступним чином. Оскільки датчик буде вимірювати температуру води, було обрано водонепроникний датчик DS18B20 від компанії Dallas, який відповідає потребам нашого застосування з діапазоном вимірювання від  $-55\text{ }^{\circ}\text{C}$  до  $125\text{ }^{\circ}\text{C}$ .

Однією з особливостей цього компонента є те, що він є цифровим датчиком, тому нам не потрібно використовувати аналоговий вивід Arduino як аналого-цифровий перетворювач.

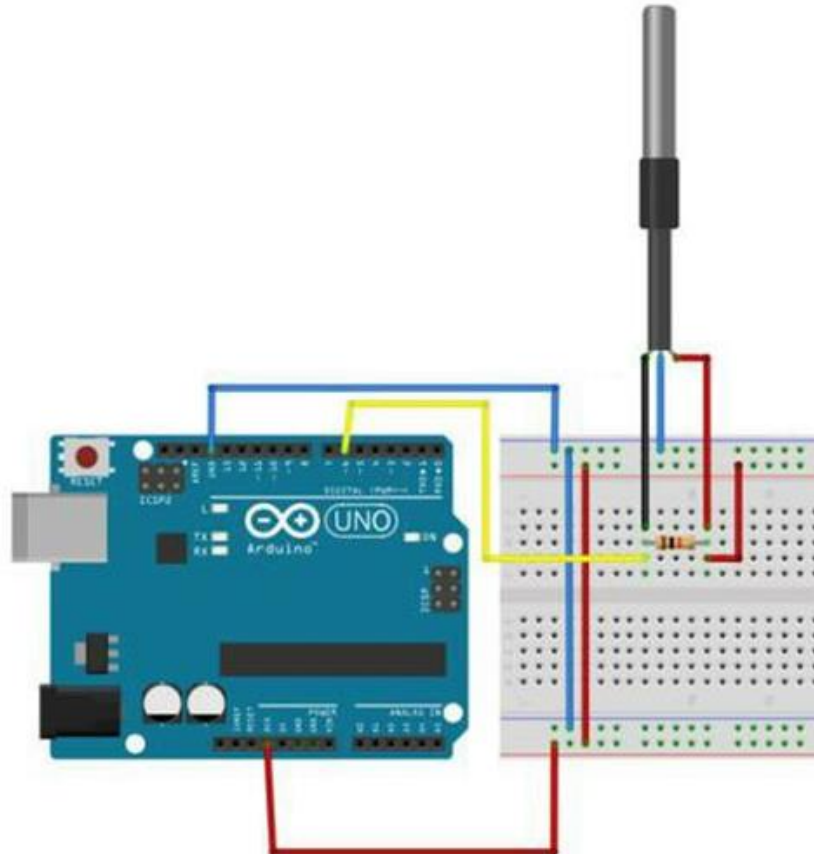


Рис. 2.8. Підключення датчика температури до Arduino

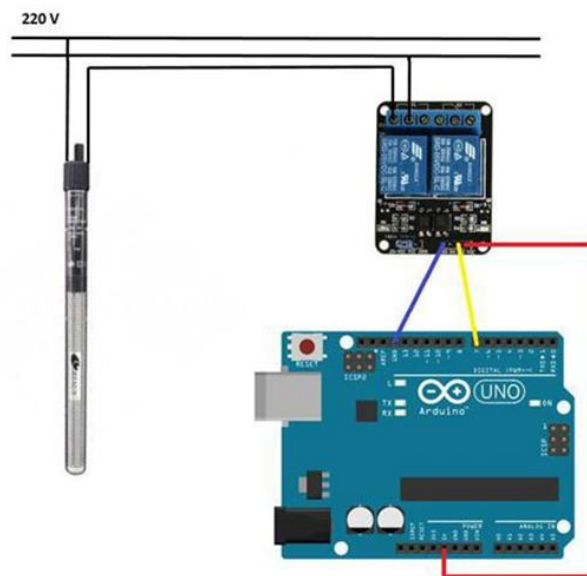


Рис. 2.9. Підключення підігрівача до Arduino

Отже, ми можемо підключити провід даних датчика до будь-яких контактів Arduino. Крім того, цей датчик використовує технологію однопровідного зв'язку, що дозволяє підключати кілька датчиків до одного контакту. Навіть якщо ми не збираємося використовувати цю функціональність, все одно нам потрібно включити бібліотеку "OneWire". Крім того, в специфікації DS18B20 зазначено, що для успішної інтеграції в схему необхіден резистор для підтягування 4,7 кОм.

Щодо компресора та магнітного клапана, в цьому проєкті компресор використовується для подачі кисню риbam, а магнітний клапан відповідає за подачу або перекриття потоку CO<sub>2</sub> для стимулювання росту водоростей у акваріумі.

Підключення аналогічне прикладам підключення підігрівача та світла. Спочатку ми підключаємо вивід VCC і GND до виводів Arduino 5V і GND. Виводи IN1 і IN2 реле можна підключити до будь-якого виводу Arduino; у моєму випадку компресор підключено до цифрового виводу 8, а магнітний клапан - до цифрового виводу 1.

На кінчику рН-зонда розміщена скляна мембрана, яка дозволяє водню проникати з вимірюваної рідини, розряджаючись на зовнішній шар скла. Більші іони залишаються в розчині. Різниця в концентрації іонів водню поза зондом і

всередині зонда створює дуже малий струм. Цей струм пропорційний концентрації іонів водню в вимірюваній рідині.

Вигляд проєктуваного та реалізованого пристрою наведено на рисунку 2.10.

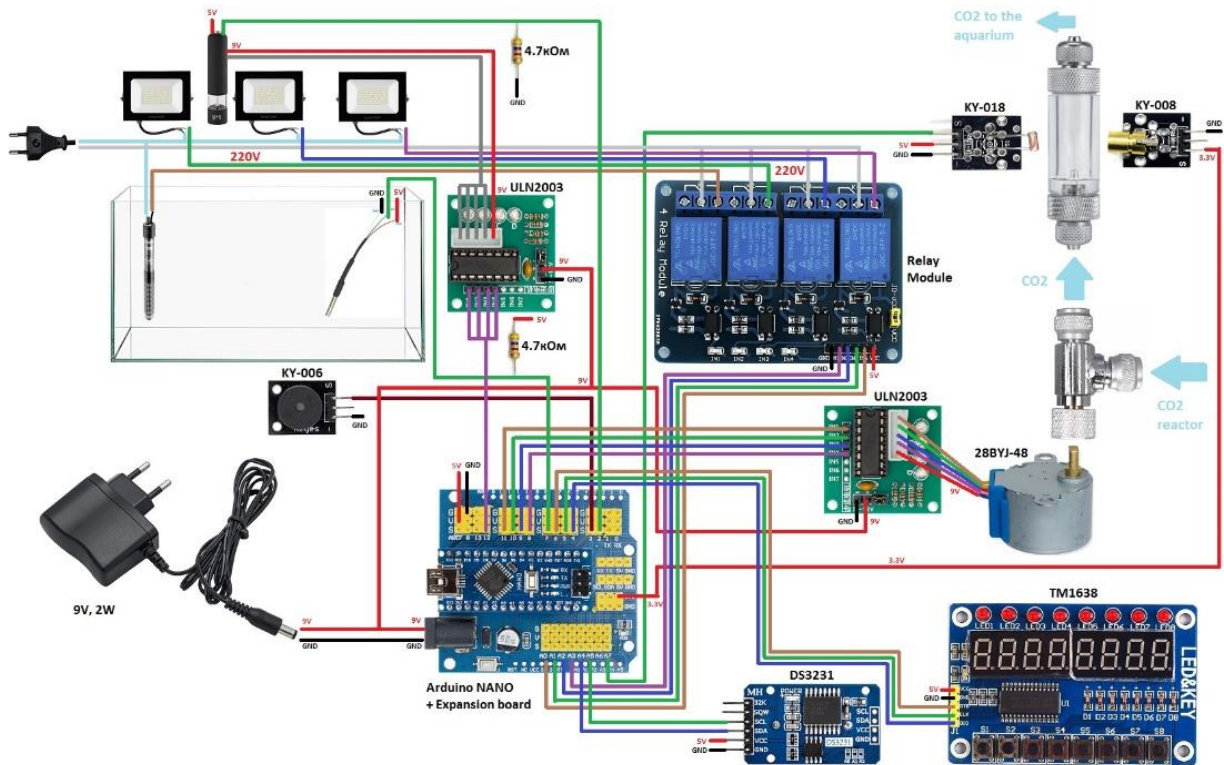


Рис. 2.10. Схема пристрою автоматизації керування акваріумом



Рис. 2.11. Карманный комп'ютер Raspberry Pi

Raspberry Pi - це одноплатний комп'ютер, який об'єднує різні компоненти комп'ютера, які зазвичай розташовуються на окремих платах, на одній платі. Крім того, ця плата має відносно компактні розміри - близько 8,5 \* 5,5 см.

У моєму проєкті плата Raspberry Pi виконує функцію передачі інформації через Інтернет. Ця плата має роз'єм для підключення до Інтернету за допомогою кабелю Ethernet, а деякі моделі мають вбудовані Wi-Fi модулі. Таким чином, завдяки Raspberry Pi ми можемо контролювати та налаштовувати параметри розумного середовища з будь-якої точки світу, використовуючи телефон та Інтернет.

Плати Raspberry Pi, так само як і плати Arduino, можуть бути програмовані, але на різних мовах програмування.

## 2.2. Розроблення та розрахунок принципової електричної схеми

Під час розробки принципової схеми виникло питання, як краще підключити пристрої для підтримки життєдіяльності акваріуму. В результаті розглянуті два варіанти - використання реле та оптосимісторів. Вибір був зроблений на користь оптосимісторів, і ось чому.

Симістор, складаючись з двох тиристорів, має три електрода. Один з них - керуючий, позначений буквою G (від англійського "gate" - "затвор"). Два інших електрода - силові (T1 і T2), можуть позначатися також буквою A (A1 і A2). На відміну від тиристора, у якого є конкретні анод і катод, у симістора кожен електрод є і анодом, і катодом одночасно. Тому симістор може проводити струм у двох напрямках, що робить його ідеальним для використання у мережах змінного струму.

Для захисту від помилкового спрацювання між силовими висновками симістора підключається RC-ланцюг. Величина резистора R1 зазвичай становить від 50 до 470 ом, а величина конденсатора C1 - від 0,01 до 0,1 мкФ. У деяких випадках ці значення можуть бути підібрані експериментально.

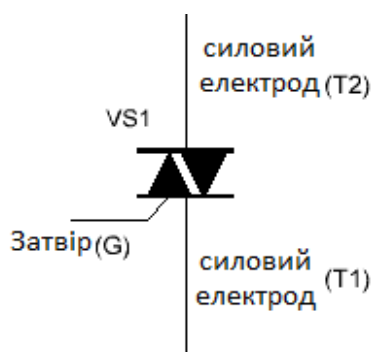


Рис. 2.12. Зображення симістор на принциповій схемі

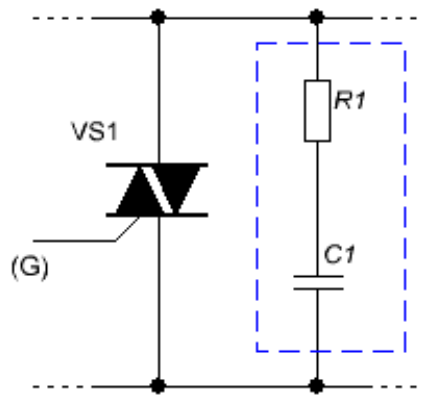


Рис. 2.13. Захист від помилкових спрацювань

Максимальна обернена напруга становить 400 В, що означає, що він без проблем може керувати навантаженням в мережі 220 В, навіть з запасом.

У імпульсному режимі напруга така ж. Максимальний струм у відкритому стані складає 5 А, а у імпульсному режимі - 10 А.

Мінімальний постійний струм, необхідний для відкриття симістора, становить 300 мА, а мінімальний імпульсний струм - 160 мА.

Він відкривається при напрузі 2,5 В при струмі 300 мА і при напрузі 5 В при струмі 160 мА. Час включення складає 10 мкс, а час виключення - 150 мкс.

Розглянемо тепер оптосимістори.

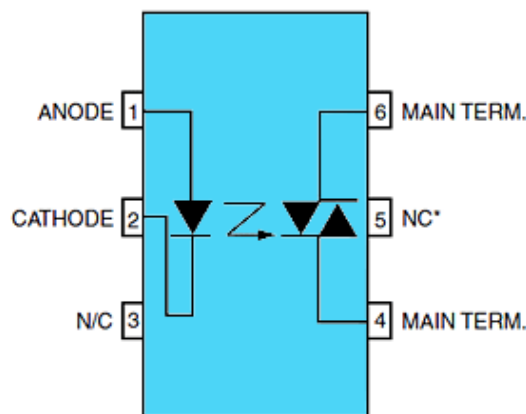


Рис. 2.14. Будова оптосимістора

Одним із сучасних та перспективних типів симісторів є оптосимістори. Назва відображає їх сутність: замість керуючого електрода у корпусі симістора є світлодіод, і керування відбувається зміною напруги на цьому світлодіоді. На зображенні показано зовнішній вигляд оптосимістора MOC3023 і його внутрішня структура. Всередині корпусу розташовані світлодіод і симістор, який



керується за допомогою випромінювання світлодіода. Виводи, позначені як N/C і NC, не використовуються і не з'єднуються з жодними елементами схеми. NC означає "Not Connect", що перекладається як "не підключено". Давайте розглянемо більш детально підключення симісторів з оптосимістором.

Це базове підключення, яке відповідає даташиту, проте існує безліч варіантів підключення для різних проєктів. Розробник повинен самостійно вирішити, як краще підключити симістори в кожній конкретній ситуації. Давайте розрахуємо оптимальний резистор  $R_{in}$  для підключення світлодіода. Ніколи не слід підключати світлодіод безпосередньо до джерела живлення або батареї, оскільки це може призвести до його миттєвого перегріву і знищення. Світлодіоди мають бути підключені через обмежувальний резистор. Зазвичай для швидкого тестування підходить резистор 1 кОм для більшості світлодіодів, якщо напруга не перевищує 12 В. Необхідно також дотримуватися правильної полярності підключення світлодіодів. Резистор повинен бути з'єднаний послідовно зі світлодіодом, щоб обмежити струм, який проходить через нього, інакше він може вигоріти миттєво.

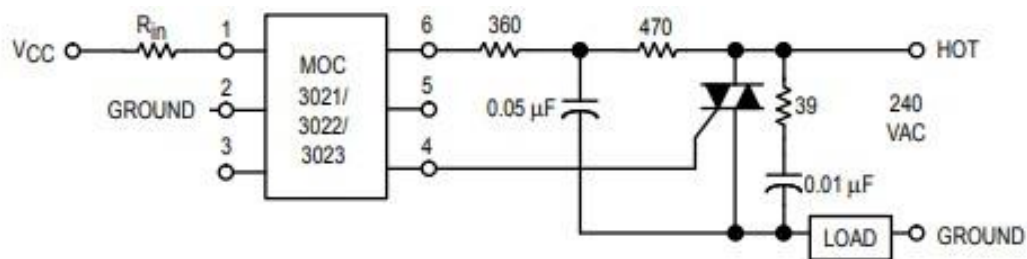


Рис. 2.15. Принципова схема підключення симістора та оптосимістора

Резистор  $R_{in}$  визначається за формулою:

$$R = \frac{(V_s - V_L)}{I} \quad (2.1)$$

$V_s$  = напруга живлення

$V_L$  = пряма напруга, розрахункова для кожного типу діодів (як правило від 2 до 4 вольт)

$I$  = ток світлодіода (наприклад 20 мА), це повинно бути менше максимально допустимого для вашого діода.

Якщо точно підібрати розмір опору виявляється складно, то рекомендується вибрати резистор більшого номіналу. Фактично, це призведе лише до невеликого зниження яскравості світіння. Наприклад, при напрузі живлення  $V_S = 5\text{ В}$  і червоному світлодіоді ( $V = 2\text{ В}$ ), який вимагає  $I = 20\text{ мА}$  ( $0.020\text{ А}$ ), опір буде  $R = (5 - 2)\text{ В} / 0.020\text{ А} = 150\text{ Ом}$ . У цьому випадку можна вибрати резистор  $390\text{ Ом}$ , що є найближчим стандартним значенням.

### 2.3. Розробка друкованої плати

Таким чином, ми дійшли висновку, що необхідно виготовити друковану плату. Це рішення було прийняте після того, як на стенді зібралося стільки проводів, що при складанні готового пристрою частина з них почала випадково відключатися через натискання інших проводів. Це відбувалося непомітно і могло спричинити незрозумілі результати, а зовнішній вигляд такого пристрою був далекий від ідеалу.

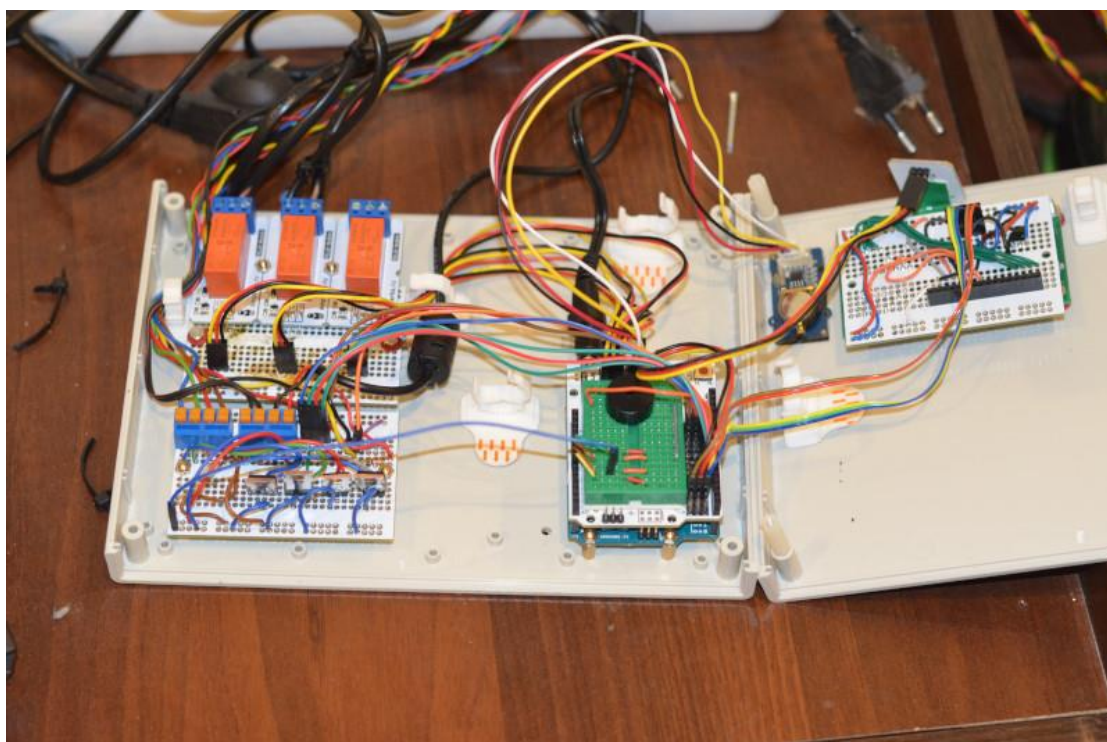


Рис. 2.16. Складання на монтажних платах (використовується Arduino Uno)  
Друкована плата розроблялася за допомогою спеціального програмного забезпечення Fritzing.

На Рисунку 2.17. показаний вид проєктованої друкованої плати.



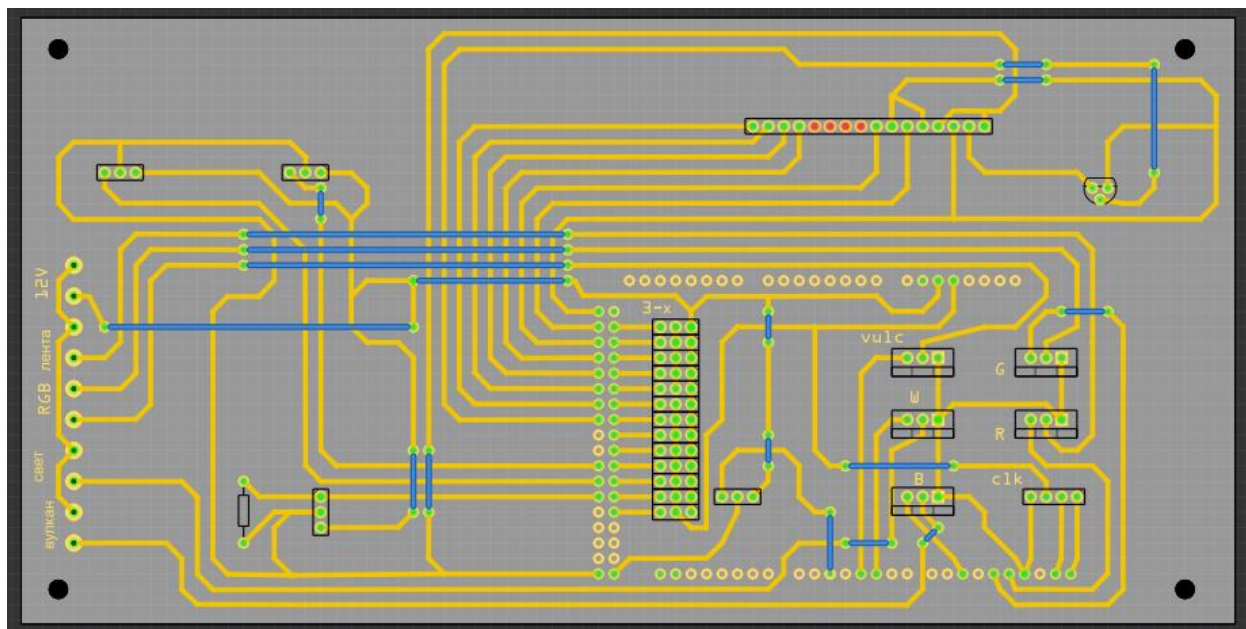


Рис. 2.17. Вид друкованої плати

## 2.4. Розробка програмного забезпечення проєктованої системи

### 2.4.1. Теоретичні відомості

#### Програмне забезпечення

Інтегроване середовище розробки Arduino - це багатоплатформовий додаток, заснований на Java, який включає в себе редактор коду, компілятор і модуль передачі прошивки на плату. Це середовище призначене для новачків у програмуванні, які не мають багатого досвіду у цій сфері. Мова програмування, використовувана в Arduino, аналогічна мові Wiring, яка базується на мові C++ з додатковими бібліотеками. Програми Arduino пишуться на мові програмування C або C++. Середовище розробки постачається з бібліотекою програм, відомою як "Wiring", яка спрощує багато стандартних операцій вводу/виводу. Користувачам потрібно визначити лише дві функції для створення програми, яка буде виконуватися у циклі.



Рис. 2.18. Arduino з прикладом простої програми

- **setup**: Ця функція виконується один раз при старті програми і призначена для встановлення початкових параметрів.
- **loop**: Ця функція виконується періодично, доки плата не буде вимкнена, і відповідає за циклічне виконання програми.

## 2.4.2. Програма для мікроконтролера

### Програмування для плати Uno

Для створення програм (скетчів) для контролера Arduino необхідно встановити середовище програмування. Простим варіантом є встановлення безкоштовної Arduino IDE, яку можна завантажити з офіційного сайту.

Після встановлення IDE слід переконатися, що вибрано потрібну плату. Для цього в Arduino IDE у меню "Інструменти" і підпункті "Плата" потрібно вибрати нашу плату (Arduino / Genuino Uno). Після вибору плати автоматично змінюються параметри збірки проєкту, і підсумковий скетч буде скомпільовано у формат, який підтримує плата. Підключивши контролер до комп'ютера через USB, ми можемо одним дотиком завантажити на нього вашу програму, використовуючи команду "Завантажити".

Зазвичай сам скетч представляє собою нескінченний цикл, в якому регулярно опитуються піни з приєднаними датчиками, і за допомогою спеціальних команд формується керуючий вплив на зовнішні пристрої (вони включаються або вимикаються). У програміста Arduino є можливість

підключити готові бібліотеки, як вбудовані в IDE, так і доступні на різноманітних сайтах і форумах.

Написана і скомпільована програма завантажується через USB-з'єднання (UART- Serial) за допомогою bootloader, який відповідає за цей процес з боку контролера.

Для програмування плат Arduino не потрібні великі навички в програмуванні. Для всіх пристроїв, датчиків, інших плат тощо, існує безліч кодів в Інтернеті, які можна завантажувати і використовувати у своїх проєктах.

В першу чергу потрібно записати всі необхідні бібліотеки. Без наявності бібліотек Arduino не зможе взаємодіяти з іншими пристроями, які є у схемі.

```
#include <OneWire.h>
```

```
#include <DallasTemperature.h>
```

```
#include "Wire.h"
```

```
#include <Servo.h>
```

```
#include <LiquidCrystal.h>
```

```
#include <DS3231.h>
```

```
Вводимо всі використовувані змінні int in1 = 8;
```

```
int in2 = 1;
```

```
int separator = 44;
```

```
float TEMPSOUHAITEE = 21 ; int LUMSOUHAITEE = 90;
```

```
int j = 0;
```

```
int EtatLampe; int HH;
```

```
int Second; int Minute; int Hour;
```

```
int DayOfWeek; int DayOfMonth; int Month;
```

```
int Year;
```

```
int PROGH1; int PROGM1; int PROGH2; int PROGM2;
```

```
String RecTEMPSOUHAITE; long nombreReception=0;
```

```
int recieved_integer = 0; String chaine;
```

```
int pinServo= 9;
```

```
int TEMPSOUHAITEE1; int LUMSOUHAITEE1;
```

```
String cul; int PROGH3;
```

```

int PROGM3; int PROGH4; int PROGM4; int PROGH1b; int PROGM1b; int
PROGH2b; int PROGM2b;

String RecupHoraire2; String RecupHoraireT; String RecupHoraireT2;

DallasTemperature sensors(&oneWire); int resistChauffante = 7;

int photoResist = A0;

```

Ми використовуємо стандартний скетч для годинника реального часу DS3231. Після завантаження скетчу в плату, ми відкриваємо монітор послідовного порту (Ctrl + Shift + M). Ймовірно, ми побачимо неправильний час або його взагалі не побачимо, оскільки годинник ще не налаштований.

Для налаштування годинника ми запускаємо приклад SetTime (Файл > Приклади > DS3231RTC > SetTime) і завантажуюмо його в плату. Після завантаження час буде налаштований за системним часом ПК на момент компіляції. Проте, якщо ми перезавантажимо плату будь-яким чином, час знову скинеться до часу компіляції. Крім того, у коді вже прописано, в яких комірках пам'яті годинника будуть зберігатися дані.

```

void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte
dayOfMonth, byte month, byte year)
{
    //      встановлює дані часу та дати до DS3231
Wire.beginTransmission(DS3231_I2C_ADDRESS); Wire.write(0);
Wire.write(decToBcd(second)); Wire.write(decToBcd(minute));
Wire.write(decToBcd(hour)); Wire.write(decToBcd(dayOfWeek));
Wire.write(decToBcd(dayOfMonth)); Wire.write(decToBcd(month));
Wire.write(decToBcd(year)); Wire.endTransmission();
}

void readDS3231time(byte *second, byte *minute,
byte *hour,
byte *dayOfWeek, byte *dayOfMonth, byte *month,
byte *year)
{

```

```

Wire.beginTransmission(DS3231_I2C_ADDRESS);    Wire.write(0);    //
встановити DS3231 вказівник на 00h Wire.endTransmission();
Wire.requestFrom(DS3231_I2C_ADDRESS, 7);

// Запитувати сім байтів даних з DS3231, починаючи з реєстрації 00h
*second = bcdToDec(Wire.read() & 0x7f);
*minute = bcdToDec(Wire.read());
*hour = bcdToDec(Wire.read() & 0x3f);
*dayOfWeek = bcdToDec(Wire.read());
*dayOfMonth = bcdToDec(Wire.read());
*month = bcdToDec(Wire.read());
*year = bcdToDec(Wire.read());
}
void displayTime()
{
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
// завантажувати дані з DS3231
readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth,
&month, &year);
// надішліть його на послідовний монітор Serial.print(hour, DEC);
Serial.print(":"); if (minute<10)
{
Serial.print("0");
}
Serial.print(minute, DEC); Serial.print(":");
if (second<10)
{
Serial.print("0");
}
Serial.print(second, DEC); Serial.print(" "); Serial.print(dayOfMonth, DEC);
Serial.print("/"); Serial.print(month, DEC); Serial.print("/"); Serial.print(year, DEC);
Serial.print(" Day of week: "); switch(dayOfWeek){

```

```

case 1: Serial.println("Sunday"); break;
case 2: Serial.println("Monday"); break;
case 3: Serial.println("Tuesday"); break;
case 4: Serial.println("Wednesday"); break;
case 5: Serial.println("Thursday"); break;
case 6: Serial.println("Friday"); break;
case 7: Serial.println("Saturday"); break;
}
}

```

Для того щоб власник розумів який зараз час в годиннику, візьмемо скетч який переобразує шістнадцятирічний код в десятковий і навпаки.

```

#define DS3231_I2C_ADDRESS 0x68

// Convert normal decimal numbers to binary coded decimal byte decToBcd(byte
val)
{
return( (val/10*16) + (val% 10) );
}

// Convert binary coded decimal to normal decimal numbers byte
bcdToDec(byte val)
{
return( (val/16*10) + (val% 16) );
}

```

Далі пишемо код для серводвигуну. Їжа буде подаватися 2 рази на добу.

```

void setup(void)
{
Wire.begin();

leServo.attach (pinServo);    // підключити об'єкт до контрольного виводу
leServo.write(12);

// Основний графік їжі PROGH1=07; PROGM1=00; PROGH2=19;
PROGM2=00;

Void setup ()

```

```

{
pinMode(pinServo , OUTPUT);
}

Void loop ()
{
DigitalWrite(pinServo, HIGH); //Вмикання реле Delay(5000);
DigitalWrite(pinServo, LOW); //Вимикання реле
}
}

```

Скетч для автоматичної подачі їжі матиме багато спільного зі скетчем для подачі кисню, оскільки принцип їх роботи практично ідентичний.

```

void setup(void)
{
Wire.begin();
}

Void setup ()
{
pinMode(in1 , OUTPUT);
}

Void loop ()
{
PROGH3=08; PROGM3=00; PROGH4=18; PROGM4=00;
DigitalWrite(pinServo, HIGH); Delay(10800000);
DigitalWrite(pinServo, LOW);
}
}

```

Напишемо скетч для датчика температури void loop(void)

```

{
float tempRecup = 0; sensors.requestTemperatures();
tempRecup = sensors.getTempCByIndex(0);
if(tempRecup<TEMPSOUHAITEE) // Перевірка температури

```

```

{
digitalWrite(resistChauffante,0);
}
else
{
digitalWrite(resistChauffante,1);
}
}

```

Наступним кроком нам треба взяти вже існуючий скетч для фото резистора.

```

capteurlum=map(CapteurlumAnalog,0,1024,0,100);   CapteurlumAnalog   =
analogRead(photoResist); etatBouton = digitalRead(bouton);
if (etatBouton == HIGH)
{
if(capteurlum < LUMSOUHAITEE) // Test luminosité
{
digitalWrite(Capteurlum,0); EtatLampe = 1;
}
else
{
digitalWrite(Capteurlum,1); EtatLampe = 0;
}
}
else
{
// Текщее состояние подачи света buttonState = digitalRead(buttonPin);
// порівняти поточний стан кнопки до попереднього збереженого стану if
(buttonState != lastButtonState) {
// якщо стан кнопки змінився і є високим, тоді збільшуємо змінну if
(buttonState == HIGH) {
// якщо поточний стан кнопки HIGH

```



```

// вона пішла від LOW до HIGH buttonPushCounter++;
}
else {
}

// запам'ятовує поточний стан кнопки
// для наступних проходів у циклі циклу lastButtonState = buttonState;
int pair;
pair= (buttonPushCounter/2); if ( pair!= buttonPushCounter)
{
digitalWrite(CapteurLum,1); digitalWrite(ledPin, HIGH);
}
else
{
digitalWrite(CapteurLum,0); digitalWrite(ledPin, LOW);
}
}

```

Наступним кроком є створення скетчу для забезпечення комунікації між Arduino Uno та Raspberry Pi. Цей скетч дозволить нам віддалено керувати параметрами акваріуму і, у разі потреби, вносити зміни в їх налаштування.

```

if (Serial.available()) { demandtemp = Serial.read()-'0';
//Переглянути всі дані на інтерфейсі // if (demandtemp == 1){
String chainetemp; String chainecapteur; String chainePROGH1; String
chainePROGM1; String chainePROGM2; String chainePROGH2; String
chaineEtatLampe;

chainetemp = String(tempRecup); chainecapteur = String(capteurLum);
chainePROGH1 = String(PROGH1); chainePROGM1 = String(PROGM1);
chainePROGH2 = String(PROGH2); chainePROGM2 = String(PROGM2);
chaineEtatLampe = String(EtatLampe);

chaine= chainetemp + ";" +chainecapteur+ ";" + chainePROGH1 + ";" +
chainePROGM1 + ";" + chainePROGH2 + ";" + chainePROGM2 + ";" +
chaineEtatLampe + ";";

```

```

Serial.println(chaine);
}
// заміна годин для подачі їжі ранок if (demandtemp == 2){
PROGH1b = PROGH1;
while ( PROGH1b == PROGH1)
if (Serial.available()>0)
{
PROGH1 = recevoirNombre();
}
}
// заміна хвилин для подачі їжі ранок if (demandtemp == 3){
PROGM1b = PROGM1;
while ( PROGM1b == PROGM1)
if (Serial.available()>0)
{
PROGM1 = recevoirNombre();
}
}
if (demandtemp == 4){
PROGH2b = PROGH2;
while ( PROGH2b == PROGH2)
if (Serial.available()>0)
{
PROGH2 = recevoirNombre();
}
}
if (demandtemp == 5){ PROGM2b = PROGM2;
while ( PROGM2b == PROGM2)
if (Serial.available()>0)
{
PROGM2 = recevoirNombre();
}
}

```

```

    }
    }
    if (demandtemp == 6)
    {
        TEMPSOUHAITEE1 = TEMPSOUHAITEE;
        while ( TEMPSOUHAITEE == TEMPSOUHAITEE1)
        if (Serial.available()>0)
        {
            TEMPSOUHAITEE=recevoirNombre();
        }
    }
    if (demandtemp == 7){
        LUMSOUHAITEE1 = LUMSOUHAITEE;
        while ( LUMSOUHAITEE == LUMSOUHAITEE1)
        if (Serial.available()>0)
        {
            LUMSOUHAITEE=recevoirNombre();
        }
    }
    }
    }

```

Оскільки температура води є ключовим параметром акваріуму, ми відображаємо всі дані про температуру на LCD-екрані.

```

AfficheurTemp.begin(16,2);
AfficheurTemp.clear();
AfficheurTemp.print("Actuelle:");
AfficheurTemp.print(tempRecup);
AfficheurTemp.print(" C");
AfficheurTemp.setCursor(0,1);
AfficheurTemp.print("Consigne:");
AfficheurTemp.print(TEMPSOUHAITEE);

```

```
AfficheurTemp.print(" C");
```

### **Висновки до розділу**

У цьому розділі Проведено повний аналіз датчиків, необхідних для створення розумного середовища. Обрані датчики температури мають герметизований корпус для занурення у воду. Розроблено підігрівач, який запобігає можливим ризикам для життя акваріумних мешканців у разі відмови датчика температури. Параметри підігрівача розраховані з урахуванням об'єму акваріуму.

Розроблено чотири алгоритми для функціонування акваріуму, які не мають взаємних залежностей. Наведені схеми підключення пристроїв до платформи Arduino Uno Rev 3, а також структурна та функціональна електричні схеми. Для контролю параметрів використано портативний комп'ютер Raspberry Pi, який взаємодіє з Wi-Fi та передає дані на мобільні пристрої.

Розроблено принципову електричну схему для "розумного" акваріуму на базі Arduino з використанням контролера ATmega328. Пристрої, що живляться від мережі 220В, підключені до Arduino за допомогою оптосимісторів та симісторів. Приведено розрахунок резистора для оптосимістора з метою забезпечення тривалого функціонування світлодіода.

## **РОЗДІЛ 3. РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ПАРАМЕТРАМИ АКВАРІУМА З ВИКОРИСТАННЯМ ІОТ-ТЕХНОЛОГІЙ**

### **3.1. Обґрунтування вибору технологічного стеку для розробки мобільних застосунків**

Сучасний світ характеризується високою мобільністю користувачів, що обумовлює постійний ріст популярності мобільних пристроїв та відповідно, мобільних додатків. Вибір технологій для розробки таких додатків є критичним рішенням, яке впливає на ефективність, вартість та час виведення продукту на ринок.

Дві домінуючі мобільні операційні системи – Android та iOS – вимагають використання різних наборів інструментів і мов програмування. Традиційно, для розробки додатків під Android використовується мова Java, а для iOS – Swift. Такий підхід дозволяє створити високопродуктивні додатки, які максимально використовують можливості кожної платформи.

Однак, з метою скорочення витрат на розробку та часу виведення продукту на ринок, все більшої популярності набувають кросплатформні рішення. Ці рішення дозволяють створювати єдину кодову базу, яка може бути використана для розробки додатків як для Android, так і для iOS. Серед популярних кросплатформених технологій можна виділити Flutter, React Native та Xamarin.

Переваги кросплатформної розробки:

- Підвищення продуктивності розробників. Завдяки використанню єдиної кодової бази, розробники можуть зосередитися на розробці бізнес-логіки додатку, а не на вирішенні специфічних проблем, пов'язаних з різними платформами. Це дозволяє скоротити час розробки та виведення продукту на ринок.
- Збільшення охоплення аудиторії. Кросплатформні додатки можуть бути розгорнуті на різних мобільних пристроях, що дозволяє охопити більшу аудиторію користувачів.

- Економічна ефективність. Створення єдиної кодової бази знижує витрати на розробку та підтримку додатків.

Виклики кроссплатформної розробки:

- Кроссплатформні додатки можуть мати нижчу продуктивність порівняно з нативними додатками, особливо в ресурсомістких операціях.
- Не завжди можливо реалізувати всі функції, доступні на кожній платформі, використовуючи кроссплатформні інструменти.
- Розробка кроссплатформних додатків вимагає глибоких знань як мов програмування, так і особливостей різних мобільних платформ.

Вибір між нативною та кроссплатформною розробкою залежить від конкретних вимог проєкту. При прийнятті рішення необхідно враховувати такі фактори як:

1. Бюджет. Кроссплатформні рішення можуть бути більш економічно ефективними, але можуть вимагати додаткових інвестицій у майбутньому.
2. Час виведення на ринок. Якщо важливо швидко випустити додаток на обидві платформи, кроссплатформна розробка може бути кращим вибором.
3. Вимоги до продуктивності. Для додатків з високими вимогами до продуктивності може бути більш доцільним використовувати нативну розробку.
4. Складність додатку. Для складних додатків з багатьма інтеграціями може знадобитися більш глибоке знання платформи.

### **3.1.1. Обґрунтування вибору технології React Native для розробки мобільного застосунку**

Для розробки проєкту було прийнято рішення про використання кроссплатформного підходу з метою оптимізації процесу розробки та забезпечення максимальної сумісності з різними мобільними платформами.

Серед існуючих кросплатформних фреймворків, таких як React Native та Flutter, був обраний React Native.

Вибір React Native обумовлений низкою факторів. React Native є одним з найстаріших і найпоширеніших кросплатформних фреймворків, що забезпечує велику спільноту розробників, багату документацію та велику кількість готових рішень. Незважаючи на те, що Flutter демонструє високу продуктивність, особливо на сучасних пристроях, React Native, за даними досліджень, забезпечує кращу продуктивність на старих пристроях [13].

Використання JavaScript як основної мови програмування робить React Native доступним для широкого кола розробників, які вже мають досвід роботи з веб-розробкою. Тісна інтеграція з React, одним з найпопулярніших фреймворків для розробки веб-додатків, забезпечує доступ до великої кількості бібліотек і інструментів.

Таким чином, вибір React Native дозволив забезпечити швидку розробку кросплатформного мобільного додатку з високою продуктивністю та гарною підтримкою спільноти.

Фреймворк React Native, заснований на JavaScript, за останні роки набув значної популярності в розробці мобільних додатків. Його широке використання в реальних проєктах свідчить про ефективність та універсальність цього інструменту.

Серед компаній, які використовують React Native для розробки своїх мобільних додатків, можна назвати такі гіганти, як Facebook, Instagram, Skype, Airbnb та Walmart. Ці компанії обрали React Native завдяки його здатності забезпечити швидку розробку, високу продуктивність та єдиний користувацький досвід на різних платформах.

Наприклад, Facebook та Instagram, будучи спочатку розробленими як нативні додатки для iOS та Android, згодом перейшли на React Native. Це дозволило їм значно прискорити процес розробки та забезпечити одночасний реліз нових функцій на обох платформах.

Переваги використання React Native

- React Native дозволяє розробникам створювати мобільні додатки для iOS та Android на основі єдиної кодової бази, що значно скорочує час і вартість розробки.
- Незважаючи на те, що React Native використовує JavaScript, він забезпечує високу продуктивність додатків завдяки використанню нативних компонентів.
- Активна спільнота React Native забезпечує велику кількість готових рішень, бібліотек та інструментів, що прискорює процес розробки.
- Завдяки гарячій перезавантаженню та іншим інструментам розробки, React Native дозволяє швидко вносити зміни в код і бачити результат.

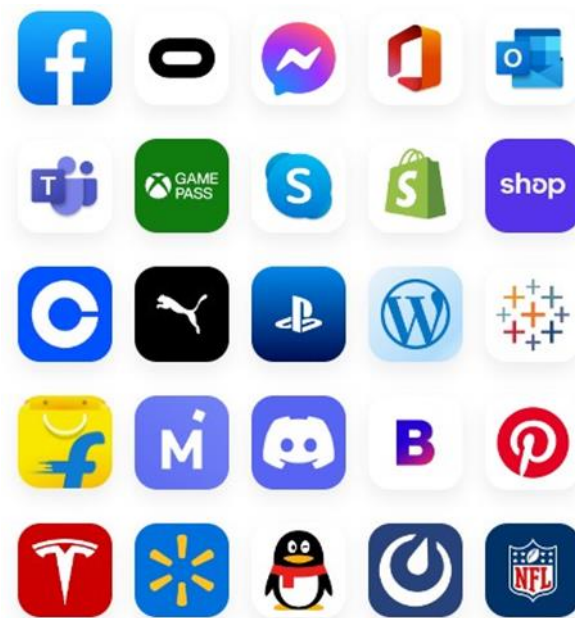


Рис. 3.1. Приклад застосунків, розроблених на React Native [13]

Однією з ключових переваг фреймворку є функція "гарячої перезавантаження", яка дозволяє розробникам миттєво бачити результати своїх змін. Цей механізм суттєво скорочує цикл розробки, оскільки відсутня необхідність повторної збірки та запуску додатку після кожної незначної модифікації коду.

### 3.1.2 Менеджер пакетів Node Package Manager

Node Package Manager (npm) є де-факто стандартом для керування пакетами в екосистемі Node.js. Він забезпечує ефективний механізм



встановлення, оновлення та видалення модулів, що значно спрощує процес розробки програмного забезпечення.

Основні функціональні можливості npm:

1. Реєстр пакетів: npm підтримує великий реєстр публічних пакетів, що дозволяє розробникам легко знаходити та встановлювати необхідні модулі для своїх проєктів.
2. Керування залежностями: npm автоматично керує залежностями проєкту, створюючи файл `package.json`, в якому зберігається інформація про всі встановлені пакети та їх версії. Це дозволяє легко відтворити середовище розробки на іншому комп'ютері або в іншій команді.
3. Скрипти: npm дозволяє автоматизувати різноманітні завдання за допомогою скриптів, які описуються в файлі `package.json`. Це можуть бути скрипти для запуску тестів, збирання проєкту, форматування коду тощо.
4. Інструменти для розробки: npm надає доступ до великої кількості інструментів для розробки, таких як лінери, тестери та будівельники.

Node Package Manager відіграє ключову роль у сучасній розробці програмного забезпечення на JavaScript. Він дозволяє:

- прискорити розробку використовуючи готові пакети розробниками, зосередитися на вирішенні бізнес-завдань, а не на реалізації базових функцій.
- покращити якість коду завдяки використанню пакетів, написаних досвідченими розробниками, можна підвищити якість коду і зменшити кількість помилок.
- спростити співпрацю в команді завдяки файлу `package.json`, всі члени команди можуть легко встановити необхідні залежності і працювати над одним проєктом.

NPM є незамінним інструментом для будь-якого JavaScript-розробника. Він забезпечує ефективне керування залежностями, спрощує процес розробки і підвищує якість кінцевого продукту.

### **3.1.3. Мова програмування TypeScript**

Комбінація React Native та TypeScript є потужним інструментом для створення сучасних мобільних додатків. Використання TypeScript, як надмножини JavaScript, надає розробникам ряд значних переваг, особливо в контексті великих і складних проєктів.

Однією з ключових особливостей TypeScript є статична типізація. На відміну від динамічно типізованого JavaScript, TypeScript вимагає від розробника явно вказувати типи даних для змінних, функцій та інших елементів програми. Це дозволяє:

1. Раннє виявлення помилок. Компілятор TypeScript може виявити багато помилок на етапі розробки, ще до запуску програми, що значно скорочує час налагодження.
2. Покращення читабельності коду. Явна типізація робить код більш зрозумілим як для самого розробника, так і для інших членів команди, що полегшує підтримку та модифікацію коду в майбутньому.
3. Підвищення надійності. Статична типізація допомагає уникнути багатьох поширених помилок, пов'язаних з невірним використанням типів даних.
4. Краща інтеграція з інструментами розробки. Сучасні IDE забезпечують розширену підтримку TypeScript, включаючи автодоповнення коду, перевірку типів та навігацію по коду.

Таким чином, використання TypeScript в поєднанні з React Native дозволяє створювати більш надійні, масштабовані та підтримувані мобільні додатки.

TypeScript, розроблена компанією Microsoft, є строго типізованою мовою програмування, яка надає розширені можливості порівняно з JavaScript. Її широке застосування в сучасній розробці програмного забезпечення обумовлено рядом переваг.

### Переваги використання TypeScript:

- Наявність статичної типізації дозволяє виявляти потенційні помилки на етапі розробки, що значно спрощує процес налагодження та підвищує загальну надійність програмного продукту.
- Модульна система TypeScript дозволяє організувати код в логічно відокремлені блоки, що покращує читабельність, підтримуваність та повторне використання коду.
- TypeScript підтримує концепції інтерфейсів та класів, що дозволяє створювати більш структурований і об'єктно-орієнтований код.
- TypeScript є надмножиною JavaScript, що означає, що будь-який код на JavaScript є також валідним кодом на TypeScript. Це дозволяє поступово переходити на TypeScript в існуючих проєктах.
- TypeScript має велику та активну спільноту розробників, що забезпечує доступ до широкого спектру інструментів, бібліотек та навчальних матеріалів.

TypeScript широко використовується в розробці мобільних додатків, зокрема в поєднанні з фреймворками React Native та Angular. Завдяки своїм можливостям, TypeScript дозволяє створювати масштабовані, надійні та легко підтримувані мобільні додатки.

TypeScript є потужним інструментом для сучасних розробників, який дозволяє підвищити продуктивність, надійність та масштабованість програмного забезпечення. Завдяки своїм можливостям, TypeScript стає все більш популярним вибором для розробки веб-додатків, серверних застосунків та мобільних додатків.

#### **3.1.4. Бібліотеки Redux та Redux Toolkit**

Redux є потужним інструментом керування станом у JavaScript-додатках, особливо в контексті фреймворків React та React Native. Він забезпечує передбачуваний і централізований спосіб управління даними в додатку, що полегшує розробку складних інтерфейсів користувача та сприяє кращій структурованості коду [16].

## Основні концепції Redux:

**Store:** єдиний об'єкт, що містить весь стан додатку. Він є джерелом істини для всіх компонентів.

**Actions:** об'єкти, що описують зміни, які потрібно внести до стану. Кожна дія має тип (string) та, опціонально, payload з додатковими даними.

**Reducers:** чисті функції, які приймають поточний стан і дію, а повертають новий стан. Reducers відповідальні за оновлення стану в відповідь на дії.

**Dispatch:** функція для відправки дій в Store. Коли дія відправляється, всі reducer-и викликаються послідовно, і стан оновлюється.

Redux Toolkit – це набір інструментів, розроблений для спрощення роботи з Redux. Він надає готові рішення для типових задач, таких як створення store, написання reducer-ів та обробки асинхронних дій. Це дозволяє розробникам швидше створювати ефективні Redux-додатки.

## Переваги використання Redux:

- Зміни стану відбуваються виключно в результаті відправки дій, що робить додаток більш передбачуваним і легким для налагодження.
- Redux дозволяє розділити стан на менші частини, що спрощує управління великими додатками.
- Існує багато розширень для браузерів та інтеграція з IDE, які допомагають візуалізувати стан додатку, відслідковувати зміни та налагоджувати код.
- Велика спільнота розробників, що використовують Redux, забезпечує багату документацію, приклади та готові рішення для різних задач.

Redux Toolkit є потужним інструментом, який значно спрощує розробку додатків на базі Redux. Він надає набір готових функцій і хуків, що дозволяють автоматизувати рутинні завдання та скоротити кількість Boilerplate коду. Завдяки стандартизованому підходу до створення Redux-додатків, Redux Toolkit підвищує продуктивність розробки, покращує читабельність коду та полегшує співпрацю в команді. Крім того, Redux Toolkit забезпечує кращу підтримку

асинхронних операцій та допомагає уникнути поширених помилок, що виникають при роботі з Redux.

### **3.1.5. Платформа Ехро**

Процес розробки мобільних додатків за допомогою React Native може бути досить трудомістким, особливо на етапі налаштування середовища розробки. Для спрощення цього процесу було створено платформу Ехро.

Ехро – це набір інструментів з відкритим кодом, який значно спрощує розробку, тестування та публікацію мобільних додатків на базі React Native. На відміну від традиційного підходу, який вимагає встановлення Xcode або Android Studio, Ехро забезпечує все необхідне для початку роботи в одному інструменті.

Основні переваги Ехро:

- автоматизує багато рутинних завдань, пов'язаних з налаштуванням середовища розробки, що дозволяє розробникам швидше приступити до написання коду.
- надає зручні інструменти для тестування додатків на емуляторах або реальних пристроях, що значно прискорює процес розробки.
- включає в себе велику бібліотеку готових компонентів, що дозволяє розробникам швидше створювати користувацькі інтерфейси.
- дозволяє розробляти додатки для iOS, Android, а також веб-платформи.
- велика спільнота розробників, що використовують Ехро, забезпечує постійний розвиток платформи та наявність великої кількості ресурсів.

Ехро створює ізольоване середовище виконання для додатків, що дозволяє запускати їх без необхідності збирати нативні пакети для кожної платформи. Це значно спрощує процес розробки та дозволяє швидше отримати готовий продукт.

Ехро - це потужна платформа для розробки мобільних додатків на React Native, яка значно спрощує процес створення і тестування додатків. За допомогою Ехро CLI розробники можуть легко ініціалізувати нові проєкти, встановлювати необхідні залежності та запускати додатки на своїх пристроях.

Функція гарячої перезавантаження забезпечує швидку ітерацію в процесі розробки, дозволяючи розробникам миттєво бачити результати змін у коді. Крім того, Ехро пропонує широкий спектр готових рішень для таких функцій, як push-сповіщення, аутентифікація та інші, що дозволяє розробникам зосередитися на створенні унікальних функціональних можливостей своїх додатків. Завдяки своїм можливостям, Ехро є одним з найпопулярніших інструментів для розробки кросплатформних мобільних додатків.

### **3.2. Обґрунтування вибору технологій для розробки серверного програмного забезпечення**

Веб-сервер є фундаментальним компонентом сучасних інформаційних систем. Він відіграє роль посередника між клієнтськими додатками (браузерами, мобільними додатками тощо) та серверними ресурсами. Принцип роботи веб-сервера полягає в прийомі HTTP-запитів від клієнтів, обробці цих запитів та формуванні відповіді у вигляді HTML-сторінок, JSON, XML або інших форматів даних.

Вибір технології для розробки веб-сервера залежить від багатьох факторів, таких як:

1. Здатність сервера обробляти великі обсяги трафіку та масштабуватися відповідно до зростання навантаження.
2. Швидкість обробки запитів та формування відповідей.
3. Здатність сервера працювати безперервно і стійко до збоїв.
4. Захищеність від різних типів атак, таких як SQL-ін'єкції, XSS та інші.
5. Наявність великої спільноти розробників, багата документація та готові рішення.

Для забезпечення взаємодії між клієнтом та сервером використовуються протоколи HTTP (HyperText Transfer Protocol) та HTTPS (HyperText Transfer Protocol Secure). HTTP є основним протоколом для передачі даних у Всесвітній павутині, але він не забезпечує шифрування даних, що робить його вразливим до перехоплення та модифікації. HTTPS, з іншого боку, використовує протокол SSL/TLS для шифрування даних, що забезпечує безпечну передачу інформації між клієнтом та сервером.

Вибір технології для розробки веб-сервера є критично важливим рішенням, яке впливає на ефективність, безпеку та масштабованість веб-додатку. При виборі технології необхідно враховувати вимоги до продуктивності, надійності, безпеки та масштабованості, а також наявність необхідних навичок у розробників.

Веб-сервери можна поділити на два основних типи: статичні та динамічні. Статичні сервери просто надають користувачам статичний контент (наприклад, HTML-файли, зображення), тоді як динамічні сервери здатні обробляти запити, взаємодіяти з базами даних та генерувати динамічний вміст. У цьому проєкті використовується динамічний веб-сервер, який слугує посередником між мобільним додатком та IoT-пристроєм. Він приймає запити від мобільного додатку, обробляє їх та передає відповіді на IoT-пристрій і навпаки. Таким чином, сервер забезпечує двосторонню комунікацію між користувачем та фізичними пристроями. На рисунку 3.2 зображено принцип роботи HTTP запитів.

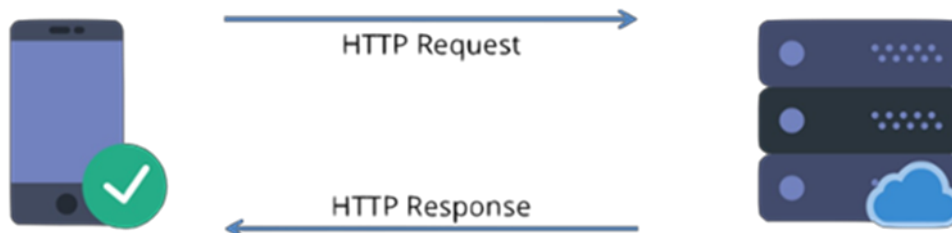


Рис. 3.2. Принцип роботи HTTP запитів [14]

Сьогодні існує безліч технологій для створення веб-серверів, кожна з яких має свої сильні сторони. Серед найбільш популярних можна виділити стеки Python Django, C# ASP.NET та Node.js Express. Вибір конкретного стеку залежить від таких факторів, як розмір проєкту, необхідна продуктивність, наявність досвіду розробки та вимоги до масштабованості. Наприклад, для створення невеликих веб-додатків добре підходить Python Django завдяки своїй простоті та великій кількості готових рішень. Для розробки масштабних корпоративних додатків часто використовують C# ASP.NET, який забезпечує

високу продуктивність і надійність. Node.js Express є відмінним вибором для створення реальних часових додатків та API.

### **3.2.1. Фреймворк Express.js**

Express.js - це мінімалістичний і гнучкий веб-фреймворк для Node.js, який дозволяє розробникам швидко і ефективно створювати різноманітні веб-додатки. Завдяки великій кількості готових модулів, доступних через npm, Express.js забезпечує високий рівень кастомізації та адаптації до конкретних потреб проєкту. Його модульна структура і простий API роблять його відмінним вибором як для новачків, так і для досвідчених розробників.

Завдяки механізму маршрутизації, можна легко налаштувати обробку різних типів HTTP-запитів для різних URL-адрес. Крім того, Express.js дозволяє легко працювати з різними форматами даних, такими як HTML, JSON та XML, що робить його універсальним інструментом для розробки веб-серверів. Проста у використанні і добре документована, ця платформа дозволяє розробникам швидко створювати прототипи і вводити нові функціональні можливості.

### **3.2.2 Платформа Node.js**

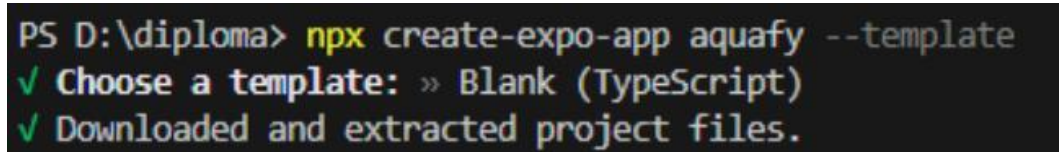
Node.js - це універсальна платформа, яка дозволяє розробникам використовувати JavaScript для створення як клієнтських, так і серверних додатків. Завдяки використанню неблокуючого вводу-виводу, Node.js забезпечує високу продуктивність і масштабованість, що робить його ідеальним вибором для створення сучасних веб-додатків. Велика спільнота розробників і багата екосистема модулів сприяють швидкому розвитку і широкому застосуванню Node.js.

Node.js, з його подієво-орієнтованою моделлю і неблокуючим вводом-виводом, забезпечує високу продуктивність і масштабованість, що робить його ідеальним вибором для створення сучасних веб-додатків. Велика і активна спільнота Node.js, а також багата екосистема бібліотек і фреймворків, таких як Express.js, надають розробникам все необхідне для швидкої і ефективної розробки.



### 3.3. Розробка мобільного застосунку

Мобільний застосунок виконує функцію дистанційного керування розумним акваріумом, а також відображає дані, що отримуються з датчиків, які знаходяться в акваріумі. Використовуючи обрані технології для розробки мобільного застосунку в інтегрованому середовищі розробки Visual Studio Code було створено новий Expo TypeScript проєкт. Для створення цього проєкту було використано команди npm CLI. На рисунку 3.3 зображено процес створення проєкту, а на рисунку 3.4 зображено структуру створеного проєкту.



```
PS D:\diploma> npx create-expo-app aquafy --template
✓ Choose a template: » Blank (TypeScript)
✓ Downloaded and extracted project files.
```

Рис. 3.3. Створення проєкту для розробки мобільного застосунку

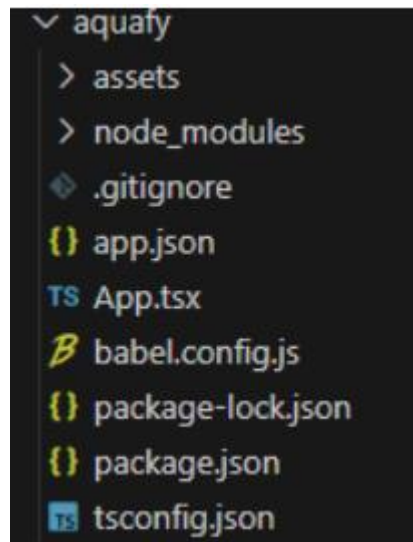


Рис. 3.4. Структура створеного проєкту

Після створення базового порожнього проєкту розпочалася розробка додатка. На головному екрані відображається інформація про температуру та рівень води в акваріумі, а також розташовані три кнопки для управління модулями розумного акваріума. Користувацький інтерфейс головного екрана додатка представлений на рисунку 3.5.

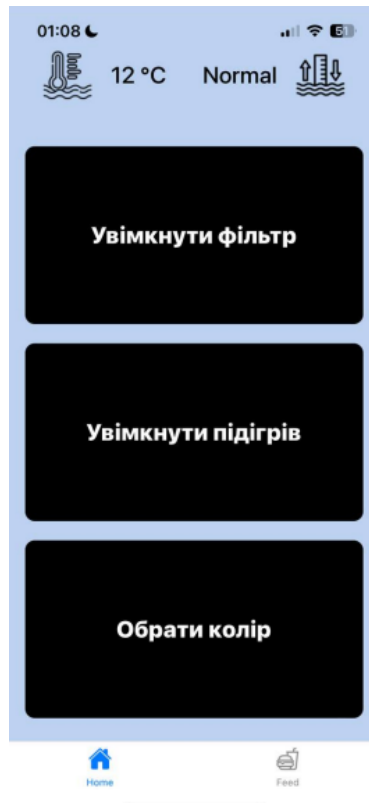


Рис. 3.5. Користувацький інтерфейс головного екрану застосунку

Після натискання кнопок «Увімкнути фільтр» та «Увімкнути підігрів» текст на кнопках змінюється на «Вимкнути фільтр» та «Вимкнути підігрів». При натисканні кнопки, яка відповідає за керування підігрівом, надсилається запит на сервер, звідки він передається на мікроконтролер. Мікроконтролер активує електронне реле, яке вмикає нагрівальний елемент. Коли температура води досягає максимально допустимого значення, підігрів автоматично вимикається, щоб запобігти перегріву та негативним наслідкам. Цей процес ілюструється на рисунку 3.6.

Натискання кнопки «Обрати колір» відкриває модальне вікно з інструментом вибору кольору підсвітки. Оскільки блок підсвічування не був включений у створений макет, вибір кольору не супроводжується запитом на сервер через відсутність необхідності. Вигляд модального вікна з інструментом вибору кольору підсвітки представлено на рисунку 3.7.

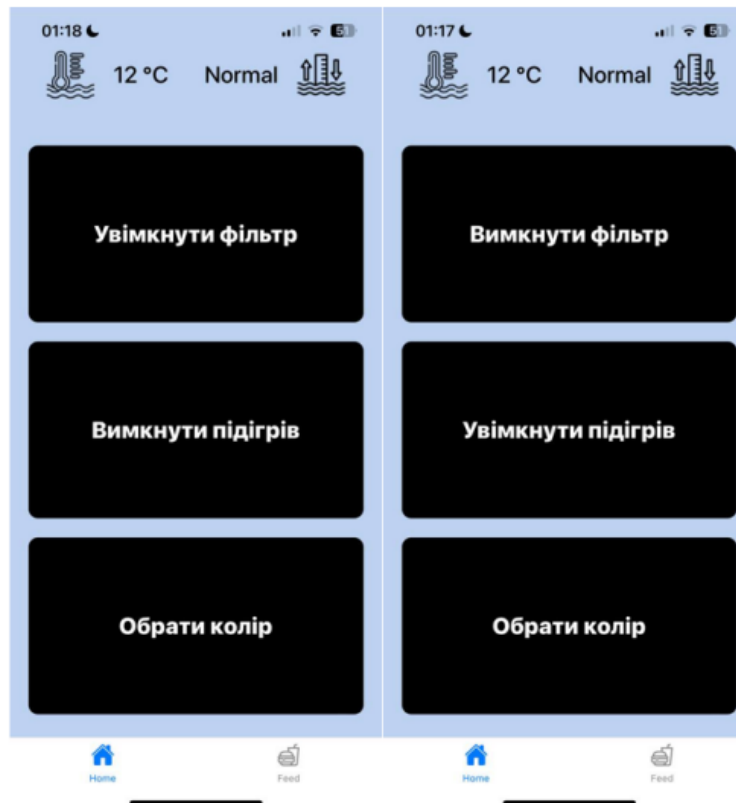


Рис. 3.6. Демонстрація зміни тексту на кнопках при натисканні

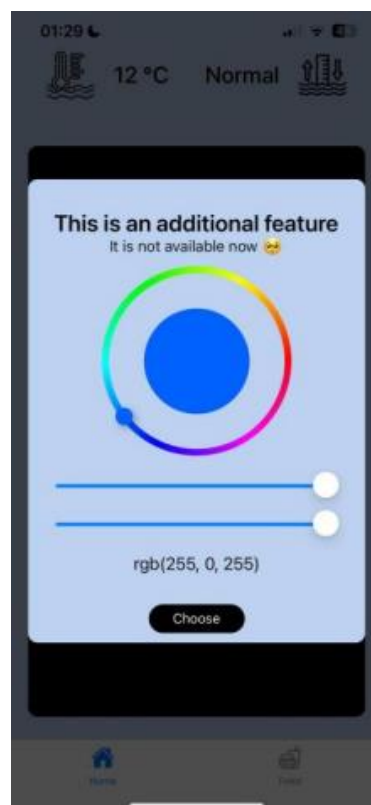


Рис. 3.7. Зовнішній вигляд модального вікна з інструментом вибору кольору підсвітки

Навігація між різними екранами додатка реалізована за допомогою Tab Navigation. Цей підхід дозволяє додати необхідну кількість вкладок у нижній

частині екрана смартфона. Натискання на вкладку змінює інтерфейс користувача, відкриваючи відповідний екран, а також підсвічує вибрану вкладку для зручності використання. Цей процес ілюструється на рисунку 3.8.

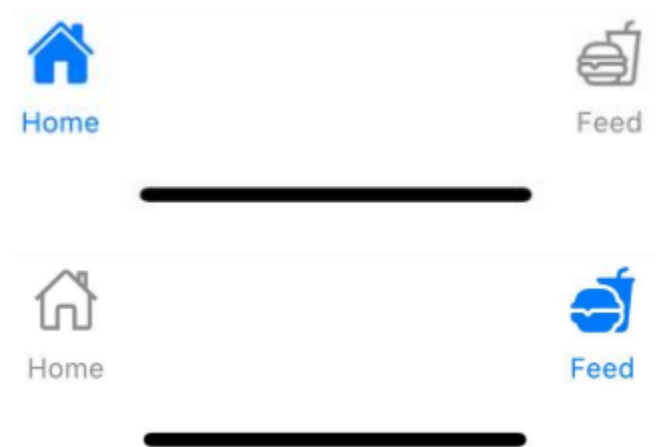


Рис. 3.8. Зовнішній вигляд Tab Navigation у застосунку

Екран годування риб містить дві основні опції: вибір кількості порцій корму та встановлення часу годування. Максимальна кількість порцій обмежена до 5, оскільки надмірна кількість корму може швидко забруднити воду в акваріумі та спричинити небажані наслідки. Користувач має можливість вибрати будь-який час доби для годування риб. Система передбачає одноразове годування на день, оскільки в більшості випадків цього достатньо для нормального харчування риб.

На рисунку 3.9 показано інтерфейс екрану налаштування годування риб. При натисканні кнопки «Обрати» відкривається модальне вікно для вибору часу годування, вигляд якого зображено на рисунку 3.10.

Застосувавши всі необхідні технології, було створено мобільний додаток, який дозволяє дистанційно керувати автоматизованою системою розумного акваріума та отримувати візуальну інформацію про параметри води.



Рис. 3.9. Зовнішній вигляд екрану годування риб

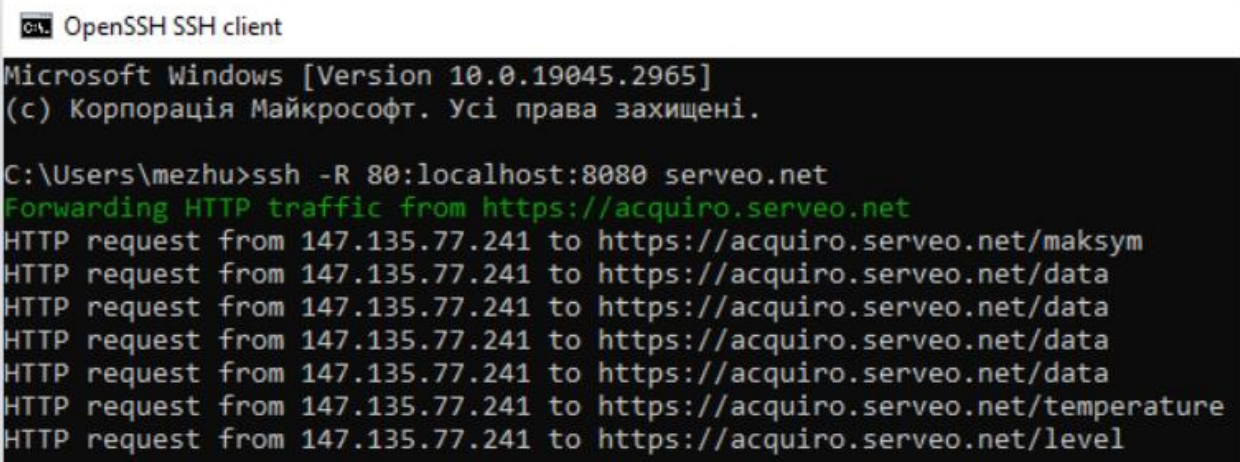


Рис. 3.10 Модальне вікно для вибору часу годування риб

### 3.4. Розробка серверного програмного забезпечення

Для інтеграції мобільного додатка з макетом системи необхідно створити серверне програмне забезпечення. Оскільки смартфон користувача має доступ до інтернету, а мікроконтролер обладнаний модулем WiFi, який забезпечує

підключення до мережі, веб-сервер повинен бути розміщений у глобальному доступі. Це дозволить керувати розумним акваріумом з будь-якого місця за наявності доступу до інтернету. Існує багато способів розміщення розробленого програмного забезпечення у відкритому доступі, але найпростішим є використання сервісу serveo для створення публічної адреси локального сервера. На рисунку 3.11 показано, як локальний сервер розміщений на публічній веб-адресі.

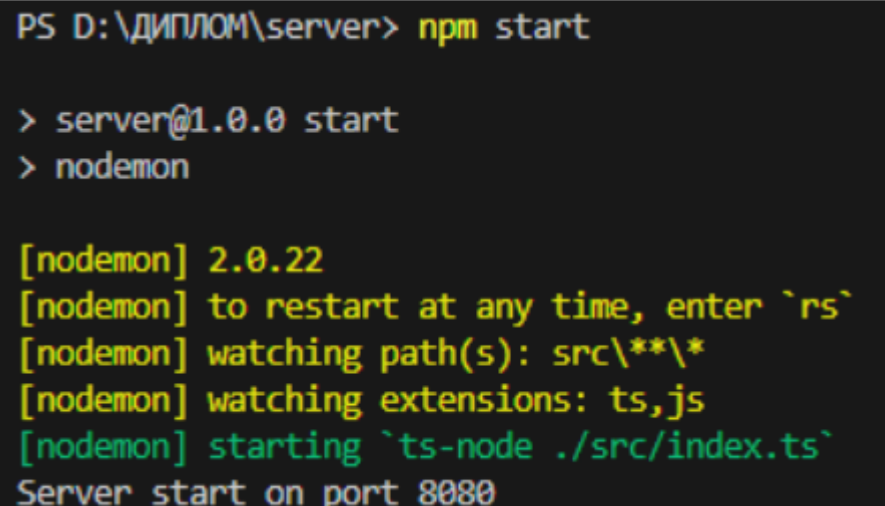


```
OpenSSH SSH client
Microsoft Windows [Version 10.0.19045.2965]
(c) Корпорація Майкрософт. Усі права захищені.

C:\Users\mezhu>ssh -R 80:localhost:8080 serveo.net
Forwarding HTTP traffic from https://acquire.serveo.net
HTTP request from 147.135.77.241 to https://acquire.serveo.net/maksym
HTTP request from 147.135.77.241 to https://acquire.serveo.net/data
HTTP request from 147.135.77.241 to https://acquire.serveo.net/data
HTTP request from 147.135.77.241 to https://acquire.serveo.net/data
HTTP request from 147.135.77.241 to https://acquire.serveo.net/data
HTTP request from 147.135.77.241 to https://acquire.serveo.net/temperature
HTTP request from 147.135.77.241 to https://acquire.serveo.net/level
```

Рис. 3.11. Розміщення серверу на публічній веб-адресі

Для забезпечення взаємодії між мобільним додатком і IoT-пристроєм на сервері необхідно розробити різноманітні запити, які реалізують потрібний функціонал та виконують задані завдання. Запуск сервера здійснюється за допомогою npm CLI за стандартним скриптом `npm start`, і сервер працює на порті, який задається вручну. Процес запуску сервера ілюструється на рисунку 3.12.



```
PS D:\ДИПЛОМ\server> npm start

> server@1.0.0 start
> nodemon

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): src\**\*
[nodemon] watching extensions: ts,js
[nodemon] starting `ts-node ./src/index.ts`
Server start on port 8080
```

Рис. 3.12. Локальний сервер запущений на порті 8080

## Висновки до розділу

Після вибору необхідних технологій для розробки мобільного додатка, серверного програмного забезпечення можна зробити кілька висновків. Обрані мови програмування, середовища розробки, фреймворки, бібліотеки та менеджери пакетів забезпечують створення проєкту, що складається з двох автономних частин, які взаємодіють між собою через сервер. Сервер виконує роль посередника, забезпечуючи з'єднання між IoT-пристроєм і мобільним додатком.

У цьому розділі розглянуто всі технології, необхідні для створення цих автономних компонентів проєкту. Кожен підрозділ містить огляд окремих фреймворків, середовищ розробки та мов програмування, які разом формують потужний набір інструментів для реалізації різних підсистем.

Створено автоматизовану систему управління розумним акваріумом, яка об'єднує мобільний додаток, IoT-пристрій і серверне програмне забезпечення для забезпечення зв'язку між мобільним додатком і мікроконтролером. У результаті розробки було створено зручний та інтуїтивно зрозумілий мобільний додаток, який надає користувачеві можливість керувати і стежити за станом акваріума за допомогою смартфона. Через додаток користувач може налаштовувати графік годування риб, регулювати кількість корму, контролювати температуру води та її рівень у акваріумі. Для забезпечення взаємодії між мобільним додатком і мікроконтролером було розроблено серверне програмне забезпечення, яке дозволяє ефективно передавати дані й команди, забезпечуючи надійну і швидку комунікацію між компонентами системи.

## ВИСНОВКИ

Оскільки розробка автоматизованих систем керування штучними середовищами існування живих організмів є досить вузькою темою в дослідженнях, готових рішень і серійного виробництва таких пристроїв поки що немає, навіть незважаючи на швидкий розвиток новітніх технологій у сфері робототехніки. Це дослідження також покликане підвищити інтерес суспільства та наукової спільноти до досліджень у цій галузі автоматизації. Наукова новизна отриманих результатів полягає в підвищенні ефективності управління мікроекосистемою домашніх акваріумів за допомогою мікроконтролерної системи. Практичне значення роботи полягає в розробці експериментального прототипу пристрою, який підвищує якість управління водними екосистемами на основі сформованого алгоритму проектування.

Отже, синтезована за допомогою розглянутих методів мікроконтролерна система дозволяє автономно виконувати поставлені задачі, мінімізуючи людський фактор і зберігаючи час. Зокрема, система може:

- вимірювати температуру водного середовища і вмикати/вимикати обігрівач за потреби;
- вимірювати рівень води в акваріумі та подавати сигнал, коли він опуститься нижче визначеного рівня (через випаровування);
- вимірювати кількість світла від лампи і подавати сигнал при зниженні потужності до недостатнього рівня;
- керувати пристроями віддалено за допомогою пульта, посиляючи команди через модуль ІЧ-порту.

У процесі вирішення поставлених у роботі завдань було досягнуто наступних результатів:

- проведено дослідження існуючих рішень автоматизованих систем та обґрунтовано актуальність теми;
- розглянуто і досліджено методи синтезу автоматизованих систем;
- на основі проведених експериментів та досліджень створено мікроконтролерну систему управління водним середовищем для мікроекосистем.



Як результат було виявлено, що автоматизована система в змозі звести до мінімуму людський фактор помилок у процесі керування важливими компонентами штучного водного середовища існування живих організмів, і тим самим покращити загальний стан акваріума автономно, без участі в цьому процесі людини.

У процесі розробки було вирішено низку технічних завдань, зокрема створення зручного та інтуїтивно зрозумілого інтерфейсу для мобільного додатка, інтеграцію мікроконтролерів для управління акваріумом, а також реалізацію взаємодії між мобільним додатком і IoT-пристроєм через сервер. Успішне виконання всіх поставлених задач дозволило розробити багатофункціональну та надійну систему, яка стабільно виконує свої функції.

Розроблена автоматизована система забезпечує користувачам зручний контроль за станом акваріума через мобільний додаток. Додаток дозволяє налаштовувати різні параметри та організовувати автоматичне годування риб. Окрім цього, система повідомляє про стан акваріума, надсилаючи сповіщення, і дає можливість оперативно реагувати на критичні ситуації, наприклад, зниження рівня води або підвищення температури понад норму.

Таким чином, аналізуючи отримані результати на прикладі акваріуму, було доведено доцільність подальшого розвитку і популяризації розробленого методу та алгоритму проектування для впровадження в автоматичні засоби керування.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цифрова схемотехніка: підручник / О.А. Борисенко. – Суми: Сумський державний університет, 2016. -200 с.
2. AquaChef Automatic Fish Feeder. ©2005-2009 Current Inc. | Updated: March 2, 2016- 16:10: <http://www.current-usa.com/aquachef>.
3. Nutrafin ProFeed Automatic Feeder. All rights reserved. A part of New York Times Company: <http://saltaquarium.about.com/od/toppicks/tp/TPautofeeders>.
4. Automatic Aquarium. Hong Kong Limited and licensors. All rights reserved.: [http://www.alibaba.com/product-gs/238098574/automatic\\_aquarium](http://www.alibaba.com/product-gs/238098574/automatic_aquarium).
5. Aquarium size. All rights reserved: <http://www.buzzle.com/articles/aquarium-care-choosing-the-right-aquarium-size-for-your-fish>.
6. Aquarium and aquarium filters. Keith Seyffarth. All rights reserved.<http://www.aqua-fish.net/articles/what-does-my-filter-do>.
7. AquaChef Automatic Fish Feeder. Updated: March 2, 2014 - 16:10 <http://www.current-usa.com/aquachef>.
8. Nutrafin ProFeed Automatic Feeder. All rights reserved. A part of New York Times Company <http://saltaquarium.about.com/od/toppicks/tp/TPautofeeders>.
9. Мікроконтролери AVR сімейства Mega / Євстіфеев А.В. – Москва: Видавничий дім «Додека - XXI», 2007.-595с.
- 10.Оптосимістори. Відкритий електронний ресурс. Режим доступу: <http://www.joyta.ru/4692-optosimistory-parametry-i-sxemy-podklyucheniya/>.
- 11.РН0-14. Відкритий електронний ресурс. Режим доступу: <https://famecart.com/p/liquid-ph0-14-value-detect-sensor-module-ph-electrode-362065314743>.
12. DS18B20 DATASHEET. Відкритий електронний ресурс. Режим доступу:<https://datasheets.maximintegrated.com/en/ds/2812.pdf>.

13. Годинник реального часу DS3231. Відкритий електронний ресурс. Режим доступу: <http://www.avrki.ru/articles/content/ds3231/>.
14. Arduino Uno R3. Відкритий електронний ресурс. Режим доступу: [https://hcomp.ru/downloads/arduino/UNOr3/arduino\\_uno\\_r3\\_RUS.pdf](https://hcomp.ru/downloads/arduino/UNOr3/arduino_uno_r3_RUS.pdf).
15. Arduino Cookbook / Michael Margolis - New York O'Reilly Media - М., 2011. - 648 с.
16. The 12 Days of Smart Home Automation! Day 5: Water Leak Detectors. URL: <https://www.ez-integration.com/blog/the-12-days-of-smart-homeautomation-day-5-water-leak-detectors>
17. Акваріумістика, історія акваріумістики, Wikipedia. URL: <https://uk.wikipedia.org/wiki/Акваріумістика>
18. Акваріум, домашній акваріум. Wikipedia. URL: <https://uk.wikipedia.org/wiki/Акваріум> (дата звернення: 15.04.2023).
19. EcoQube C+ | Your window to NATURE. URL: <https://www.kickstarter.com/projects/kevinzl/ecoqube-c-your-window-to-nature0>
20. Сайт виробника Fluval Aquatics. URL: <https://fluvalaquatics.com/>
21. Сайт виробника Innovative Marine. URL: <https://www.innovativemarine.com/nuvo-fusion-pro>
22. Сайт виробника FishBit. URL: <https://getfishbit.com/>
23. React Native. Facebook Supported, Community Driven. URL: <https://reactnative.dev/>.
24. Top 3 Online Tools for Simulating HTTP Requests. URL: <https://ubidots.com/blog/top-3-online-tools-for-simulating-http-requests/>
25. TypeScript Documentation. URL: <https://www.typescriptlang.org/docs/>
26. Getting Started with Redux Toolkit. URL: <https://reduxtoolkit.js.org/introduction/getting-started> (дата звернення: 30.04.2023).
27. Node.js documentation. URL: <https://nodejs.org/en/docs> (дата звернення: 30.04.2023).

28. Arduino documentation. URL: <https://docs.arduino.cc/> (дата звернення: 30.04.2023).
29. Сайт виробника мікроконтролерів Espressif. URL: <https://www.espressif.com/en/products/socs/esp32>
30. Розумний дім. Основні відомості. Wikipedia. URL: [https://uk.wikipedia.org/wiki/Розумний\\_дім](https://uk.wikipedia.org/wiki/Розумний_дім)

## ДОДАТОК

### Лістинг коду головного меню

```
/*
Menu.h - Library for displaying menu in TM1638.
*/
#pragma once
#define BLINK_INTERVAL 500
#include "TM1638My.h"
#include "CurrSettings.h"
#include "MenuItem.h"
enum submenu
{
    timeMenu, timer, morning, evening, alarm, lampInterval,
    curTemp, logTemp, dayTemp, nightTemp, deltaTemp,
    bubblesInSecond, bubbleControlSettings, sensorValue, bubbleSettings, bubbleDaySpeed,
    bubbleNightSpeed, beforeMorningStart, bubbleControlSound,
    feedingMenu, morningFeeding, eveningFeeding, dayFeedingSettings, nightFeeding, durations,
    motorPosition, motorSpeed, bubbleDurations, bubbleCount, bubblesInMinute, sensorInSecond,
    errorsInSecond,
    anon
};
class Menu {
public:
    Menu(TM1638My* _module, ControlTemp* _controlTemp, BubbleCounter* _bubbleCounter,
        StepMotor* _stepMotor, BubbleControl* _bubbleControl, Feeding* _feeding, MicroDS3231* _rtc,
        CurrSettings* _currSettings);
    void display();
    bool loopNeedControl();
    submenu getSubmenu();
private:
    TM1638My* module;
    MenuItem* subMenu[6];
    ControlTemp* controlTemp;
    BubbleCounter* bubbleCounter;
    StepMotor* stepMotor;
    BubbleControl* bubbleControl;
    Feeding* feeding;
    MicroDS3231* rtc;
    CurrSettings* currSettings;
    byte gorInd, verInd;
    unsigned long nextKeyboardTime, lastBlinkTime;
    byte numEditItem;
    void initSubmenu(submenu _submenu);
    submenu submenuName(byte _gorInd, byte _verInd);
    byte getDots(submenu _submenu);
    bool readKeyboardNeedControl();
    void blinkDisplay();
};
Menu::Menu(TM1638My* _module, ControlTemp* _controlTemp, BubbleCounter*
    _bubbleCounter, StepMotor* _stepMotor, BubbleControl* _bubbleControl, Feeding* _feeding,
    MicroDS3231* _rtc, CurrSettings* _currSettings) {
    module = _module;
```

```

controlTemp = _controlTemp;
bubbleCounter = _bubbleCounter;
stepMotor = _stepMotor;
feeding = _feeding;
bubbleControl = _bubbleControl;
rtc = _rtc;
currSettings = _currSettings;
gorInd = 0;
verInd = 0;
numEditItem = 0;
initSubmenu(submenuName(gorInd, verInd));
nextKeyboardTime = millis() + KEYBOARD_INTERVAL;
lastBlinkTime = 0;
};
bool Menu::loopNeedControl() {
    blinkDisplay();
    return readKeyboardNeedControl();
}
void Menu::blinkDisplay() {
    if (numEditItem == 0 || (millis() - lastBlinkTime) <= BLINK_INTERVAL) return;
    lastBlinkTime = millis();
    subMenu[numEditItem - 1]->changeBlink();
    display();
}
bool Menu::readKeyboardNeedControl() {
    if (millis() <= nextKeyboardTime) return false;
    nextKeyboardTime = millis() + KEYBOARD_INTERVAL;
    bool ans = false;
    byte keys = module->keysPressed(B00111111, B00111100);
    if (module->keyPressed(0, keys) && currSettings->alarmMelody != nullptr) {
        // Esc - выход из мелодии
        delete currSettings->alarmMelody;
        currSettings->alarmMelody = nullptr;
    }
    if (numEditItem) {
        if (module->keyPressed(0, keys)) {
            // Esc - выход из редактирования
            subMenu[numEditItem - 1]->exitEditing();
            numEditItem = 0;
            display();
        }
        if (module->keyPressed(1, keys)) {
            // Enter - сохраняем и к следующему
            subMenu[numEditItem - 1]->saveEditing();
            ans = true;
            int i;
            int sizeSubMenu = sizeof(subMenu) / sizeof(subMenu[0]);
            for (i = numEditItem; i < sizeSubMenu; i++) {
                if (subMenu[i] != nullptr && subMenu[i]->editing()) break;
            }
            if (i < sizeSubMenu) {
                numEditItem = i + 1;
                subMenu[numEditItem - 1]->enterEditing();
            }
        }
    }
}

```

```

        display();
    }
    else {
        numEditItem = 0;
        display();
    }
}
if (module->keyPressed(2, keys)) {
    subMenu[numEditItem - 1]->downValue();
    lastBlinkTime = millis();
    display();
}
if (module->keyPressed(3, keys)) {
    subMenu[numEditItem - 1]->upValue();
    lastBlinkTime = millis();
    display();
}
}
else {
    if (module->keyPressed(0, keys)) {
        // Esc - возврат меню на адрес 0-0
        if (verInd) {
            verInd = 0;
            initSubmenu(submenuName(gorInd, verInd));
            display();
        }
        else if (gorInd) {
            gorInd = 0;
            initSubmenu(submenuName(gorInd, verInd));
            display();
        }
    }
    if (module->keyPressed(1, keys)) {
        // Enter - режим редактирования
        int i;
        int sizeSubMenu = sizeof(subMenu) / sizeof(subMenu[0]);
        for (i = numEditItem; i < sizeSubMenu; i++) {
            if (subMenu[i] != nullptr && subMenu[i]->editing()) break;
        }
        if (i < sizeSubMenu) {
            numEditItem = i + 1;
            subMenu[numEditItem - 1]->enterEditing();
            display();
        }
    }
}
int dVer = 0, dGor = 0;
if (verInd == 0 && module->keyPressed(2, keys)) dGor--; // left
if (verInd == 0 && module->keyPressed(3, keys)) dGor++; // right
if (module->keyPressed(4, keys)) dVer++; // down
if (module->keyPressed(5, keys)) dVer--; // up
if ((dGor || dVer) && (submenuName(gorInd + dGor, verInd + dVer) != anon)) {
    gorInd += dGor;
    verInd += dVer;
}

```

```

        initSubmenu(submenuName(gorInd, verInd));
        display();
    }
}
return ans;
}
submenu Menu::getSubmenu() {
    return submenuName(gorInd, verInd);
};
void Menu::display() {
    String sOut;
    currSettings->startEndDurations(0);
    for (auto& menuItem : subMenu) {
        if (menuItem == nullptr) break;
        sOut += menuItem->display();
    }
    int deltaLen = sOut.length() - 8;
    if (deltaLen > 0) sOut = sOut.substring(0, 8);
    while (deltaLen++ < 0) {
        sOut += ' ';
    }
    module->setDisplayToString(sOut, getDots(submenuName(gorInd, verInd)));
    currSettings->startEndDurations(1);
}
submenu Menu::submenuName(byte _gorInd, byte _verInd) {
    switch (_gorInd) {
        case 0: switch (_verInd) {
            case 0: return timeMenu; break;
            case 1: return timer; break;
            case 2: return morning; break;
            case 3: return evening; break;
            case 4: return alarm; break;
            case 5: return lampInterval; break;
            default: return anon; break;
        }
        case 1: switch (_verInd) {
            case 0: return curTemp; break;
            case 1: return logTemp; break;
            case 2: return dayTemp; break;
            case 3: return nightTemp; break;
            case 4: return deltaTemp; break;
            default: return anon; break;
        }
        case 2: switch (_verInd) {
            case 0: return bubblesInSecond; break;
            case 1: return bubbleControlSettings; break;
            case 2: return sensorValue; break;
            case 3: return bubbleSettings; break;
            case 4: return bubbleDaySpeed; break;
            case 5: return bubbleNightSpeed; break;
            case 6: return beforeMorningStart; break;
            case 7: return bubbleControlSound; break;
            default: return anon; break;
        }
    }
}

```



```

    }
    case 3: switch (_verInd) {
        case 0: return feedingMenu; break;
        case 1: return morningFeeding; break;
        case 2: return eveningFeeding; break;
        case 3: return dayFeedingSettings; break;
        case 4: return nightFeeding; break;
        case 5: return durations; break;
        default: return anon; break;
    }
    case 4: switch (_verInd) {
        case 0: return motorPosition; break;
        case 1: return motorSpeed; break;
        case 2: return bubbleDurations; break;
        case 3: return bubbleCount; break;
        case 4: return bubblesInMinute; break;
        case 5: return sensorInSecond; break;
        case 6: return errorsInSecond; break;
        default: return anon; break;
    }
    }
    return anon;
}
byte Menu::getDots(submenu _submenu) {
    switch (_submenu) {
        case timeMenu: return currSettings->secondLed ? B00010000 : 0; break;
        case bubbleDaySpeed:
        case bubbleNightSpeed:
        case timer:
        case morning:
        case evening:
        case morningFeeding:
        case eveningFeeding:
        case alarm: return B00010000; break;
        case curTemp: return controlTemp->getAquaTempConnected() ? B00100010 : B00100000; break;
        case bubbleControlSettings: return B01000100; break;
        case logTemp: return B01000010; break;
        case nightFeeding: return B01000100; break;
        case deltaTemp: return B00001000; break;
        case dayFeedingSettings:
        case bubblesInSecond: return B01000000; break;
    }
    return 0;
}
void Menu::initSubmenu(submenu _submenu) {
    for (auto& menuItem : subMenu) {
        if (menuItem != nullptr) {
            delete menuItem;
            menuItem = nullptr;
        }
    }
    switch (_submenu) {
        case timeMenu:

```

```

subMenu[0] = new SettingsValue(currSettings, dayNight);
subMenu[1] = new TextItem(" ");
subMenu[2] = new TimeValue(currSettings, 0, rtc);
subMenu[3] = new TimeValue(currSettings, 1, rtc);
subMenu[4] = new SettingsValue(currSettings, timerOn);
subMenu[5] = new AlarmFlag();
break;
case timer:
subMenu[0] = new TextItem("St");
subMenu[1] = new TimerValue(currSettings, 0);
subMenu[2] = new TimerValue(currSettings, 1);
subMenu[3] = new TimerStart(currSettings);
break;
case morning:
subMenu[0] = new TextItem("Sd");
subMenu[1] = new byteEEPROMvalue(EEPROM_MORNING_HOUR, 0, 23, 2);
subMenu[2] = new byteEEPROMvalue(EEPROM_MORNING_MINUTE, 0, 59, 2);
break;
case evening:
subMenu[0] = new TextItem("Sn");
subMenu[1] = new byteEEPROMvalue(EEPROM_EVENING_HOUR, 0, 23, 2);
subMenu[2] = new byteEEPROMvalue(EEPROM_EVENING_MINUTE, 0, 59, 2);
break;
case alarm:
subMenu[0] = new TextItem("Sb");
subMenu[1] = new byteEEPROMvalue(EEPROM_ALARM_HOUR, 0, 23, 2);
subMenu[2] = new byteEEPROMvalue(EEPROM_ALARM_MINUTE, 0, 59, 2);
subMenu[3] = new byteEEPROMvalue(EEPROM_ALARM, 0, 1, 2, 1);
break;
case lampInterval:
subMenu[0] = new TextItem("Sdn ");
subMenu[1] = new byteEEPROMvalue(EEPROM_LAMP_INTERVAL, 0, 30, 2);
break;
case curTemp:
subMenu[0] = new TextItem("i");
subMenu[1] = new RtsTemp(rtc);
subMenu[2] = new TextItem("o");
subMenu[3] = new AquaTemp(controlTemp);
break;
case logTemp:
subMenu[0] = new TempLog(currSettings, controlTemp);
break;
case dayTemp:
subMenu[0] = new TextItem("Td ");
subMenu[1] = new byteEEPROMvalue(EEPROM_DAY_TEMP, 14, 30, 2);
subMenu[2] = new byteEEPROMvalue(EEPROM_DAY_TEMP_ON, 0, 1, 2, 1);
break;
case nightTemp:
subMenu[0] = new TextItem("Tn ");
subMenu[1] = new byteEEPROMvalue(EEPROM_NIGHT_TEMP, 14, 30, 2);
subMenu[2] = new byteEEPROMvalue(EEPROM_NIGHT_TEMP_ON, 0, 1, 2, 1);
break;
case deltaTemp:

```

```

    subMenu[0] = new TextItem("dt ");
    subMenu[1] = new byteEEPROMvalue(EEPROM_DELTA_TEMP, 5, 10, 2);
    break;
case bubblesInSecond:
    subMenu[0] = new bubbleCounterValue(bubbleCounter, bubbleIn100Second);
    subMenu[1] = new bubbleControlValue(bubbleControl, controlCondition);
    break;
case bubbleControlSettings:
    subMenu[0] = new bubbleControlValue(bubbleControl, minBubblesIn100Second);
    subMenu[1] = new bubbleControlValue(bubbleControl, maxBubblesIn100Second);
    break;
case sensorValue:
    subMenu[0] = new bubbleCounterValue(bubbleCounter, minLevel);
    subMenu[1] = new bubbleCounterValue(bubbleCounter, maxLevel);
    break;
case bubbleSettings:
    subMenu[0] = new TextItem("d ");
    subMenu[1] = new byteEEPROMvalue(EEPROM_MAX_DURATION_BUBBLE, 0, 50, 2, 1);
    subMenu[2] = new TextItem("h ");
    subMenu[3] = new byteEEPROMvalue(EEPROM_MIN_LEVEL_BUBBLE, 0, 50, 2, 1);
    break;
case bubbleDaySpeed:
    subMenu[0] = new TextItem("Bd");
    subMenu[1] = new byteEEPROMvalue(EEPROM_DAY_BUBBLE_SPEED, 19, 250, 4, 1);
    subMenu[2] = new byteEEPROMvalue(EEPROM_DAY_BUBBLE_ON, 0, 1, 2, 1);
    break;
case bubbleNightSpeed:
    subMenu[0] = new TextItem("Bn");
    subMenu[1] = new byteEEPROMvalue(EEPROM_NIGHT_BUBBLE_SPEED, 19, 250, 4, 1);
    subMenu[2] = new byteEEPROMvalue(EEPROM_NIGHT_BUBBLE_ON, 0, 1, 2, 1);
    break;
case beforeMorningStart:
    subMenu[0] = new TextItem("bd");
    subMenu[1] = new byteEEPROMvalue(EEPROM_BEFORE_MORNING_BUBBLE_START,
0, 250, 4, 3);
    break;
case bubbleControlSound:
    subMenu[0] = new TextItem("Sound");
    subMenu[1] = new byteEEPROMvalue(EEPROM_CONTROL_BUBBLE_SOUND_ON, 0, 1, 3,
2);
    break;
case feedingMenu:
    subMenu[0] = new TextItem("Feed");
    subMenu[1] = new FeedingValue(feeding);
    break;
case morningFeeding:
    subMenu[0] = new TextItem("Fd");
    subMenu[1] = new byteEEPROMvalue(EEPROM_MORNING_FEEDING_HOUR, 0, 23, 2);
    subMenu[2] = new byteEEPROMvalue(EEPROM_MORNING_FEEDING_MINUTE, 0, 59, 2);
    subMenu[3] = new byteEEPROMvalue(EEPROM_MORNING_FEEDING_LOOP, 0, 20, 2, 1);
    break;
case eveningFeeding:
    subMenu[0] = new TextItem("Fd");

```

```

    subMenu[1] = new byteEEPROMvalue(EEPROM_EVENING_FEEDING_HOUR, 0, 23, 2);
    subMenu[2] = new byteEEPROMvalue(EEPROM_EVENING_FEEDING_MINUTE, 0, 59, 2);
    subMenu[3] = new byteEEPROMvalue(EEPROM_EVENING_FEEDING_LOOP, 0, 20, 2, 1);
    break;
case dayFeedingSettings:
    subMenu[0] = new TextItem("d");
    subMenu[1] = new byteEEPROMvalue(EEPROM_DAY_FEEDING_DURATION, 0, 250, 3, 0);
    subMenu[2] = new TextItem("p");
    subMenu[3] = new byteEEPROMvalue(EEPROM_DAY_FEEDING_PAUSE, 0, 99, 3, 2, 10);
    break;
case nightFeeding:
    subMenu[0] = new byteEEPROMvalue(EEPROM_NIGHT_FEEDING_HOUR, 0, 23, 2);
    subMenu[1] = new byteEEPROMvalue(EEPROM_NIGHT_FEEDING_MINUTE, 0, 23, 2);
    subMenu[2] = new TextItem("d");
    subMenu[3] = new byteEEPROMvalue(EEPROM_NIGHT_FEEDING_DURATION, 0, 250, 3,
0);
    break;
case durations:
    subMenu[0] = new SettingsValue(currSettings, dur);
    break;
case motorPosition:
    subMenu[0] = new TextItem("POS ");
    subMenu[1] = new MotorPosition(stepMotor);
    break;
case motorSpeed:
    subMenu[0] = new TextItem("51");
    break;
case bubbleDurations:
    subMenu[0] = new TextItem("52");
    break;
case bubbleCount:
    subMenu[0] = new TextItem("53");
    break;
case bubblesInMinute:
    subMenu[0] = new TextItem("54");
    break;
case sensorInSecond:
    subMenu[0] = new TextItem("55");
    break;
case errorsInSecond:
    subMenu[0] = new TextItem("56");
    break;
}
}

```