МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра Інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Зав.кафедрою _____
д.е.н.,доцент Левицький С.І.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ШТУЧНОЇ ЕКОСИСТЕМИ

Виконав
ст. гр. КІ-228 _____ Д.В. Хондожко
(підпис)

Керівник
к.т.н. _____ О.А. Хараджян
(підпис)

Запоріжжя
2023

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ

ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра Інформаційних технологій

ЗАТВЕРДЖУЮ
Зав. кафедрою

_____
д.е.н.,доцент Левицький С.І.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

Студенту гр. *КІ – 228*, спеціальності *«Комп'ютерна інженерія»*

Хондожко Денис Вікторович

1.Тема: *Розробка програмного забезпечення для штучної екосистеми.*

затверджена наказом по інституту № 02-10 від 27 01. 2023 р.

2. Термін здачі студентом закінченої роботи: 12.06. 2023 р.

3. Перелік питань, що підлягають розробці:

1. Аналіз природних та штучних екосистем
2. Аналіз систем управління штучними екосистемаи
3. Розробка алгоритму роботи системи забезпечення штучної екосистеми
4. Розробка складових системи керування
5. Розробка алгоритма роботи системи

6. Розробка система дистанційного контролю та керування екоситемою
7. Розробка програми для контролю та керування штучною екосистемою
8. Розробка основних функціональних модулів
9. Розробка модулів взаємодії з апаратним забезпеченням
10. Тестування системи

Дата видачі завдання: 16.01.2023 р.

Студент _____ Д.В. Хондожко
(підпис) (прізвище та ініціали)

Керівник роботи _____ О.А. Хараджян
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна бакалаврська робота містить 45 стор., 9 рис., 1 таблицю, 1 додаток, 7 використаних джерел.

Об’єкт роботи: системи керування та контроль штучних екосистем.

Предмет роботи: дистанційні системи керування та контроль штучних екосистем.

Мета роботи: розробка дистанційної системи керування та контроль штучної екосистеми.

Задачі роботи: аналіз природних та штучних екосистем; аналіз систем управління штучними екосистемаи; розробка алгоритму роботи системи забезпечення штучної екосистеми; розробка складових системи керування; розробка алгоритма роботи системи; розробка система дистанційного контролю та керування екоситемою; розробка програми для контролю та керування штучною екосистемою; розробка основних функціональних модулів; розробка модулів взаємодії з апаратним забезпеченням; тестування системи.

Система контролю та управління штучною екосистемою побудована на основі контролера ESP8266 та забезпечує управління фітолампою, світлодіодною лампою білого кольору, насосом для води, насосом для повітря та підігрівачем води.

Для віддаленого керування та моніторингу системи використовується система Blynk.

ДИСТАНЦІЙНЕ КЕРУВАННЯ, СИСТЕМА УПРАВЛІННЯ, ШТУЧНА ЕКОСИСТЕМА, ARDUINO, BLYNK

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Повна назва	Пояснення/переклад
ЗШЕ	Закрита штучна екосистема	
ПІД	Пропорційно-інтегрально-диференційний регулятор	
LCD	Liquid crystal display	
NTP	Network Time Protocol	
RTC	Real-time clock	

ВСТУП

Штучні екосистеми — це системи створені людиною, де біотичні та абіотичні компоненти взаємодіють один з одним. Люди створюють штучні екосистеми для власного використання. Без технологічних комплексів та втручання людини штучні екосистеми не можуть вижити. Штучні екосистеми не можуть підтримувати себе. Такі системи потребують постійного втручання людини для нормального функціонування.

У різних наукових та практичних застосуваннях використовуються закриті штучні екологічні системи, які не залежать від обміну речовиною з будь-якою частиною поза системою. Закриті штучні екосистеми зазвичай невеликі. Такі системи цікаві з наукової точки зору і потенційно можуть служити системою життєзабезпечення під час космічних польотів або в космічних середовищах існування. У закритій екосистемі будь-які відходи, вироблені одним видом, повинні використовуватися принаймні одним іншим видом.

Розробка ефективної системи контролювання та регулювання штучних факторів навколишнього середовища, щоб ЗШЕ надійно працювала у разі виникнення екологічних порушень, досить актуально. В даний час широко використовуються керування з відкритим і лінійним замкнутим контуром в ЗШЕ.

Після аналізу існуючих рішень для створення автономних систем, було прийнято рішення створити штучну екосистему на основі програмно-апаратного комплексу Arduino.

Об’єкт роботи: системи керування та контроль штучних екосистем.

Предмет роботи: дистанційні системи керування та контроль штучних екосистем.

Мета роботи: розробка дистанційної системи керування та контроль штучної екосистеми.

Задачі роботи:

- аналіз природних та штучних екосистем;
- аналіз систем управління штучними екосистемаи;
- розробка алгоритму роботи системи забезпечення штучної екосистеми;
- розробка складових системи керування;
- розробка алгоритма роботи системи;
- розробка система дистанційного контролю та керування екоситемою;
- розробка програми для контролю та керування штучною екосистемою;
- розробка основних функціональних модулів;
- розробка модулів взаємодії з апаратним забезпеченням;
- тестування системи.

РОЗДІЛ 1
ШТУЧНІ ЕКОСИСТЕМИ ТА СИСТЕМИ КЕРУВАННЯ

1.1. Природні та штучні екосистеми

Екосистеми - це функціональні одиниці, які складаються з біотичних і абіотичних компонентів, і вони взаємодіють одна з одною для існування. Екосистеми можна розділити на два типи:

- природна екосистема;
- штучна екосистема.

Природна екосистема — це система, в якій відбувається взаємодія між навколишнім середовищем і живими організмами. Такі системи утворюються природним шляхом і не потребують людської діяльності для свого функціонування. Прикладами природних екосистем є ставки, річки, ліси тощо.

Штучні екосистеми — це системи створені людиною, де біотичні та абіотичні компоненти взаємодіють один з одним. Однак такі системи не самопідтримується і можуть загинути без допомоги людини. Прикладами штучних екосистем є акваріуми, сільськогосподарські поля, зоопарки тощо. У табл. 1.1 наведені основні особливості екосисте.

Таблиця 1.1 - Різниця між природною та штучною екосистемою

Параметр	Природна екосистема	Штучна екосистема
Визначення	Це природна екосистема, де організми та навколишнє середовище взаємодіють один з одним, щоб співіснувати	Це створена людиною екосистема з біотичними та абіотичними компонентами, а також включає деякі технології
Приклади	Тропічні ліси, моря, озера, луки	Зоопарки, птахофабрики,

	тощо	акваріуми тощо
Виживання	Він може вижити сам по собі	Для процвітання йому потрібна допомога людини.
Генетичне різноманіття	Має велике генетичне різноманіття	Оскільки він створений людиною, він має дуже обмежену генетичну різноманітність.
Еволюція	Живі організми в цьому типі екосистеми мають високі шанси на еволюцію, оскільки вони навчаться пристосовуватися до свого оточення	Він має дуже низькі шанси на еволюцію.
Харчовий ланцюг	Харчовий ланцюг довгий і складний.	Харчові ланцюги короткі і зазвичай неповні.
Цикли поживних речовин	Цикли поживних речовин повні та проходять природним шляхом.	Кругообіги поживних речовин завжди неповні.
Важливість	Ці екосистеми забезпечують збереження природних ресурсів, дарують красу навколишньому середовищу та дозволяють біологічну спадщину наступним поколінням.	Ці екосистеми створені людиною для власного використання, наприклад, для високої продуктивності та розваг.

Для розуміння структури системи управління та моніторингу для штучної екосистеми необхідно розглянути більш детально особливості цих систем.

Люди створюють штучні екосистеми для власного використання.

Структура штучної екосистеми складається з 3 частин — біотичних факторів, абіотичних факторів і технологічних комплексів, які створені

людиною. Біотичні фактори включають три основних компоненти - продуценти, консументи і розкладачі. Вони також визначають напрямок потоку енергії в екосистемі. До абіотичних факторів належать все неживе: ґрунт, сонячне світло, поживні речовини, гази та інше. Абіотичні фактори утворюють середовище існування екосистеми, що підтримує організми, що живуть у ній. У цих екосистемах розвиток організмів та потоки енергії регулюються людьми.

Без технологічних комплексів та втручання людини штучні екосистеми не можуть вижити. Штучні екосистеми не можуть підтримувати себе. Такі системи потребують постійного втручання людини для нормального функціонування (под ци необхідно розуміти, як підтримання технологічних систем, так і контроль за функіонуванням системи вцілому).

До них відносяться акваріуми, поля з посівами, тераріуми та багато іншого.

Основні особливості штучних екосистем належать:

- штучні екосистеми створені людьми для виробництва чи розваг;
- оскільки люди створюють ці екосистеми, кількість і різноманітність організмів регулюється, отже, генетичне різноманіття цих екосистем надзвичайно низьке;
- мале генетичне різноманіття означає відсутність еволюції через відсутність природних стимулюючих факторів;
- у більшості випадків через відсутність відповідних розкладачів і бактерій цикли поживних речовин і харчові ланцюги перериваються. Тому їх потрібно виконувати вручну.

Штучні екосистеми цілеспрямовані. Такі системи плануються і будуються для конкретних цілей.

Закриті штучні екологічні системи— це екосистеми, які не залежать від обміну речовиною з будь-якою частиною поза системою. Цей термін найчастіше використовується для опису невеликих екосистем, створених людиною. Такі системи цікаві з наукової точки зору і потенційно можуть служити системою життєзабезпечення під час космічних польотів, на

космічних станціях або в космічних середовищах існування. У закритій екологічній системі будь-які відходи, вироблені одним видом, повинні використовуватися принаймні одним іншим видом. Якщо мета полягає в підтримці форми життя, такої як миша чи людина, продукти життєдіяльності, такі як вуглекислий газ, фекалії та сеча, повинні зрештою перетворюватися на кисень, їжу та воду. Замкнена екологічна система повинна містити принаймні один автотрофний організм. Хоча і хемотрофні, і фототрофні організми ймовірні, майже всі закриті екологічні системи на сьогоднішній день засновані на автотрофі, такому як зелені водорості.

1.2. Системи управління штучними екосистемаи

Екологічні порушення часто негативно впливають на нормальну роботу закритих штучних екосистем. Конкретну закриту штучну екосистему можна розглядали як дисипативну систему, її кінетичну модель можна побудувати на основі динаміки систем та експериментальних даних. На основі екологічної термодинаміки і динаміки систем оптимальний закон зворотного зв’язку для контролю інтенсивності світла, температури та швидкості аерації можна отримати за допомогою функції накопиченої енергії та принципу максимальної потужності Одума. Результати цифрового моделювання показують, що замкнуту систему керування замкненою штучною екосистемою можна стабілізувати в заданій робочій точці з бажаними характеристиками динамічного відгуку, що супроводжується проведенням еко-роботи та розсіюванням накопиченої енергії, створеної впливами навколишнього середовища з різною силою.

Закриті штучні екосистеми в основному складаються з біологічного співтовариства, в якому переважають вищі рослини, і штучних факторів навколишнього середовища, таких як світло, температура, вода, повітря та добрива, які відповідають екологічним вимогам біологічного співтовариства та

здійснюють вирощування сільськогосподарських культур, овочів, квітів і лікарських трав високої якості і ефективності. При проектуванні та будівництві ЗШЕ найважливішим критерієм є експлуатаційна надійність і стабільність, оскільки робочий процес ЗШЕ сприйнятливий до впливів навколишнього середовища, що призведе до зниження врожайності рослин і погіршення екологічних функцій, і навіть до поломки системи. Отже, необхідно розробити, як ефективно контролювати та регулювати штучні фактори навколишнього середовища, щоб ЗШЕ надійно працював у разі виникнення екологічних порушень. В даний час керування з відкритим і лінійним замкнутим контуром широко використовуються в ЗШЕ.

Управління з відкритим контуром базується лише на суб'єктивному досвіді та робочих специфікаціях, а не на інформації зворотного зв'язку в режимі онлайн про роботу системи, отже, керування з відкритим контуром дуже чутливе до збурень із низькою точністю керування та стабільністю роботи. Просте керування замкнутим циклом, як правило, застосовується для лінійних систем, отриманих зазвичай шляхом лінеаризації нелінійної системи в конкретній робочій точці та в її околицях, а алгоритм лінійного зворотного зв'язку отримано з класичної кібернетики, як-от геометричне місце кореня, діаграма Боде, полюс розміщення, квадратичне оптимальне керування тощо. Незважаючи на це, ЗШЕ є надзвичайно нелінійною складною системою, робочий стан якої часто значно відхиляється від певної робочої точки під впливом збурень, тому процедура лінеаризації та прості замкнуті регулятори, такі як ПІД та квадратично-гауссове керування для механічних, електромагнітних і рідинних систем непридатні для ЗШЕ для досягнення бажаної динамічної продуктивності та навіть викликають серйозні збої в управлінні. Простий, але переконливий аргумент Лотки вже запропонував, що, як принцип, «природний відбір прагне зробити потік енергії через систему максимальним».

Принцип максимальної потужності Одума пояснює, що переважають екосистеми, які максимізують потік корисної енергії для підтримки та

зростання. З точки зору екологічної термодинаміки, штучну екосистему також можна розглядати як дисипативну структурну систему з природною тенденцією відходу від термодинамічної рівноваги в найбільшій мірі та поступового набуття здатності проводити максимальну еко-роботу.

Відповідно до теорії дисипативної структури, якщо можна знайти належну функцію накопиченої енергії (так звану функцію Ляпунова), можна отримати оптимальний алгоритм керування зворотним зв’язком для стійкої стабілізації на задовільному робочому рівні з гарною динамікою. реагування на керуючу дію.

РОЗДІЛ 2
РОЗРОБКА АЛГОРИТМУ РОБОТИ СИСТЕМИ ЗАБЕЗПЕЧЕННЯ ШТУЧНОЇ ЕКОСИСТЕМИ

2.1. Складові системи керування та алгоритм роботи системи

Система контролю та управління штучною екосистемою складається з наступних компонентів:

- контролер ESP8266-ESP-12E-WeMos-Board;
- модуль реле 8-Channel – 5V Relay Module;
- індикатор LCD 2004;
- модуль годинника RTC;
- датчик температури DS18B20;
- фітолампа;
- світлодіодна лампа (колір білий);
- насос для води мембранний;
- насос для повітря;
- підігрівач води.

Одна світлодіодна стрічка використовується для освітлення росли, а друга – для освітлення акваріуму.

Після ретельного аналізу існуючих рішень для створення автономних систем, було прийнято рішення створити штучну екосистему на основі програмно-апаратного комплексу Arduino

Arduino — апаратна обчислювальна платформа для аматорського конструювання, створена для швидкої і легкої розробки різноманітних електронних пристроїв. Вона може отримувати дані про навколишній світ завдяки датчикам і реагувати, керуючи світлом, моторами і іншими приводами.

Для програмування не потрібен програматор, програма завантажується через порт USB.

Насос забезпечує накачування мулу до коренів рослин та відкачування залишків води до акваріуму (рис. 2.2).

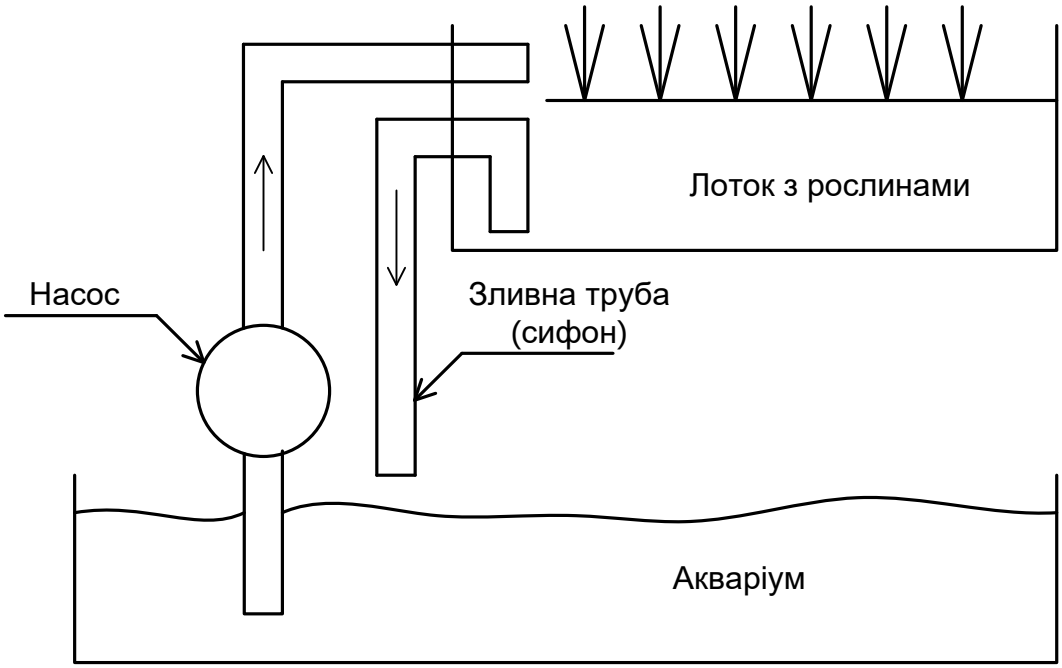

Рис. 2.2 - Схема установки штучної екосистеми

Накачування та відкачування мулу виконується за допомогою насосу та сифона.

На першому етапі виконується накачування води в лоток з рослинами нижче рівня коліна сифона (рис2.3).

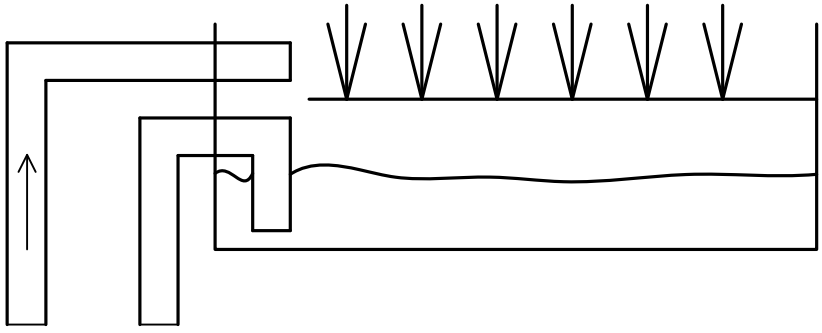

Рис. 2.3 - Перший рівень води

Для зливу води насос додатково накачує воду в лоток вище рівня коліна сифону (рис.2.4).

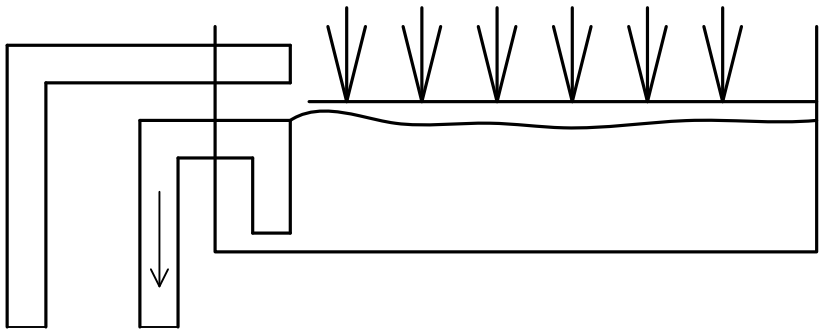

Рис. 2.4 - Другий рівень води

Після чого насос вимикається і вода по сифону зливається в акваріум (рис.2.5).

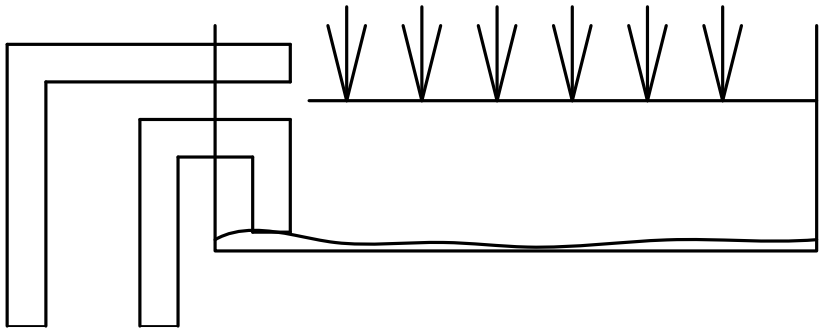

Рис. 2.5 - Третій рівень води

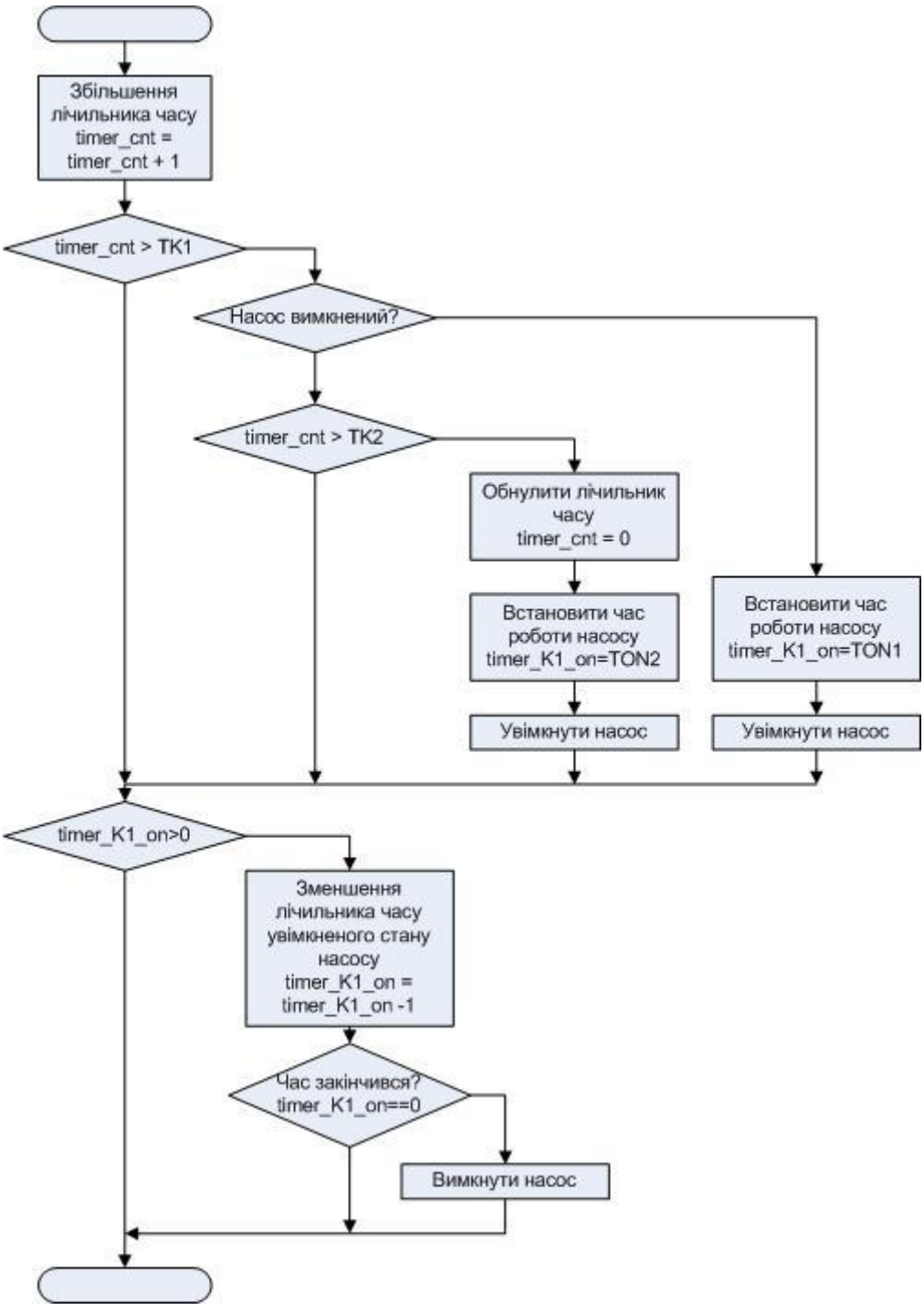

Рис. 2.6 - Блок-схема алгоритму роботи насосу

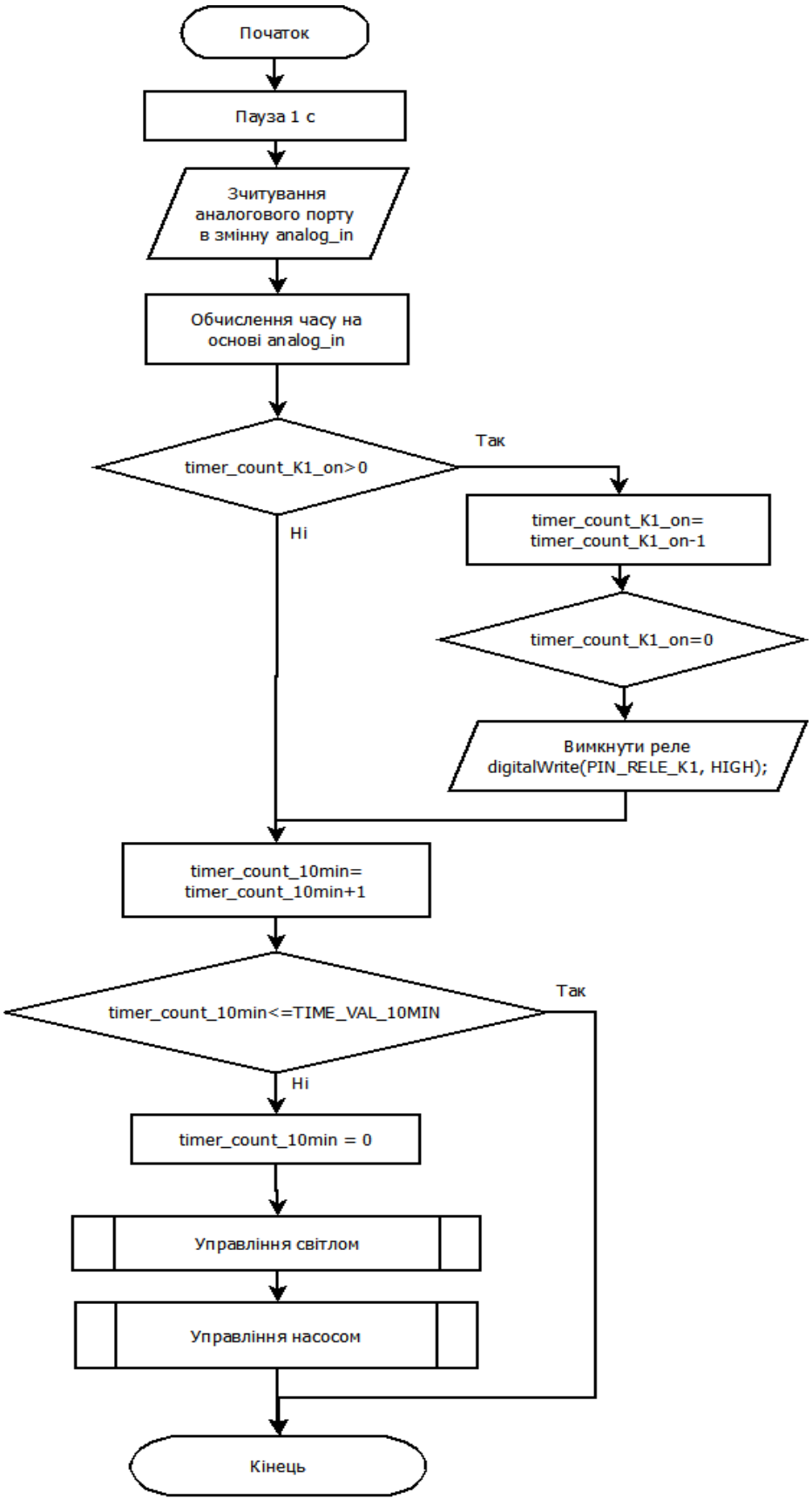

Рисунок 2.7 - Блок-схема алгоритму функції кроку виконання

Процедура управління насосом по діаграмі, представлена на рис. 2.7.

Схема з’єднання модулів системи управління замкненою екосистемою наведена на рис. 2.8.

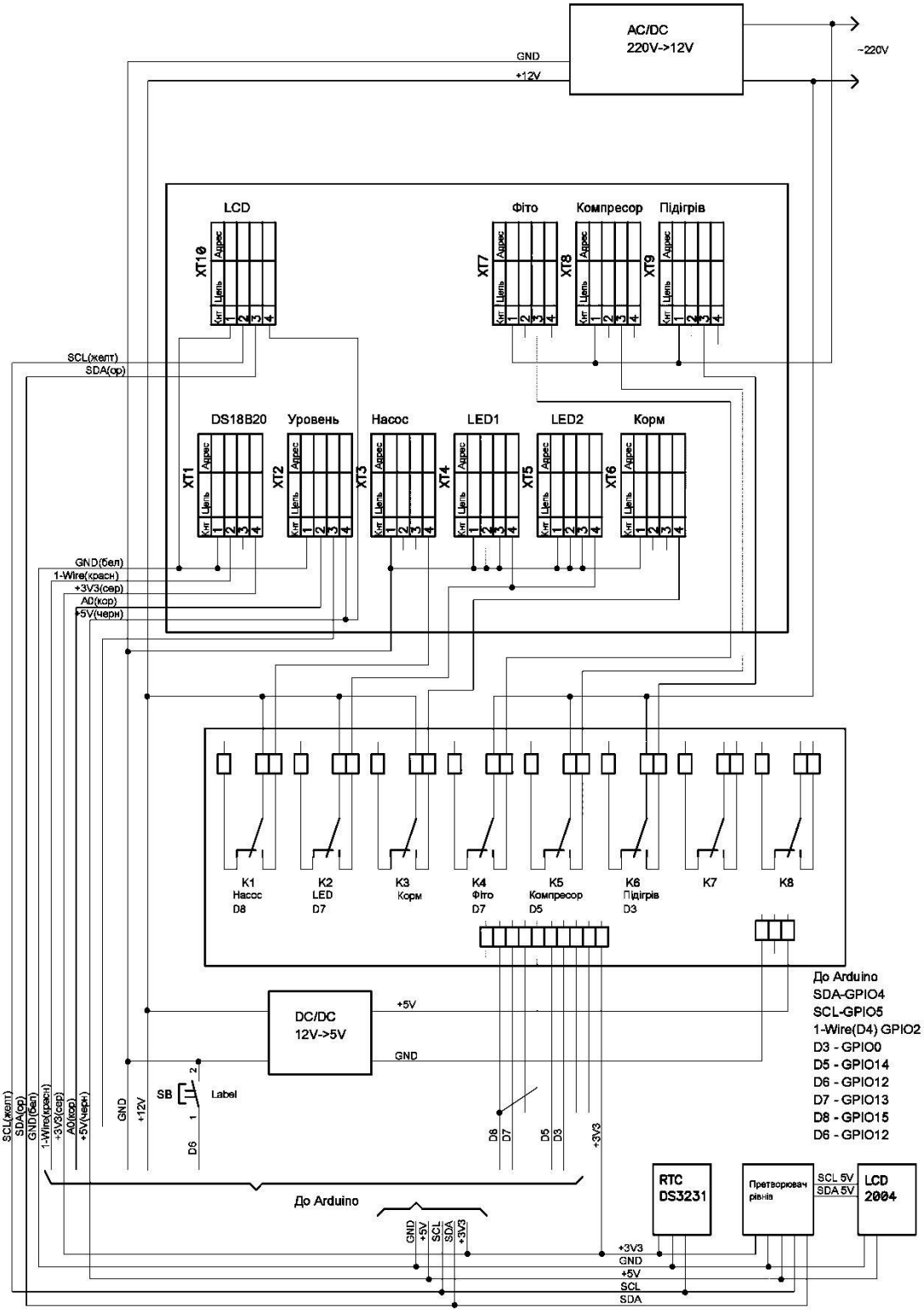

Рис. 2.8 – Схема системи управління замкненою екосистемою

В системі використано наступні компоненти: контролер ESP8266-ESP-12E-WeMos-Board, модуль реле 8-Channel – 5V Relay Module, рідкокристалічний індикатор LCD 2004 з модулем інтерфейсу I2C, модуль годинника реального часу RTC DC3231, датчик температури DS18B20, перетворювач інтерфейсів 5V-3V3 та джерело живлення 220В-12В, 5В.

2.2. Система дистанційного контролю та керування екоситемою

Для віддаленого керування та моніторингу системи використовується система Blynk.

Система Blynk — це набір програмного забезпечення, необхідного для прототипування, розгортання та віддаленого керування підключеними електронними пристроями будь-якого масштабу: від персональних IoT-проектів до комерційних продуктів.

За допомогою Blynk можна підключити своє обладнання до хмари та створити безкодові програми для iOS, Android і Web-додатки для аналізу даних у режимі реального часу та історії даних, що надходять із пристроїв, керувати ними віддалено з будь-якої точки світу, отримувати важливі сповіщення та багато іншого.

Blynk є рішенням яке розрахований на багато користувачів. Його можливо налаштувати на окремі ролі користувачів.

Програми, створені за допомогою Blynk, готові для кінцевих користувачів.

Розглянемо процес створення додатку. Спочатку створюється шаблон швидкого запуску та пристрій швидкого запуску з програмою scketch для середовища виконання.

Scketch необхідно завантажити до плати мікроконтролеру. Він використовує параметри для підключення до Blynk.Cloud і стає видимим у програмах.

У Blynk використовуються шаблони пристроїв, щоб полегшити роботу з кількома пристроями, які виконують однакові функції.

Елементи шаблону пристрою. Кожен шаблон пристрою містить набір параметрів. Розглянемо лише найважливіші з них. Найважливішим елементом шаблону є TemplateID. Це унікальний ідентифікатор кожного шаблону, який потрібно вказати в коді на пристрої

```
#define BLYNK_TEMPLATE_ID "SomeTemplateID"
#define BLYNK_TEMPLATE_NAME "Quickstart Device"
```

Визначення ідентифікатора шаблону завжди має бути першим рядком у програмі

Знайти TemplateID можна за посиланням де наведено фрагмент коду

→ Templates → YourTemplate

Потоки даних Datastream це канал для передачі даних між пристроєм і Blynk.Cloud. Щоб правильно їх обробити, Blynk.Cloud має знати, які дані передаються.

Під час процесу швидкого запуску створено та налаштовано чотири потоки даних. Зразки шаблонів наведені за посиланням

→ Templates → Quickstart Template → Datastreams

Після визначення потоки даних використовуються в мобільних програмах і веб-панельках під час візуалізації даних у віджетах.

Віджети є частиною шаблону пристрою.

Пристрій швидкого запуску. Розглянемо код, який він використовував для підключення та зв’язку з Blynk. Код залежить від обладнання та способу підключення, які ви використовуєте. Ви завжди можете знайти правильний приклад коду для свого обладнання

Розглянемо частини коду. Визначення основних параметрів:

```
#define BLYNK_TEMPLATE_ID "MyTemplateID"
#define BLYNK_TEMPLATE_NAME "MyTemplateName"
#define BLYNK_AUTH_TOKEN "MyAuthToken"
```

Назва пристрою BLYNK_TEMPLATE_NAME може бути будь-якою.

AuthToken — це унікальний ідентифікатор, створений Blynk.Cloud. Він повинен бути в кожному пристрої. Для кожного пристрою генерують новий AuthToken.

AuthToken можна отримати вручну або автоматично для пристроїв WiFi.

Для кожної апаратної платформи необхідно вказувати свій файл з визначеннями

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
```

Облікові дані WiFi

```
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";
```

Таймери. При роботі з пристроями IoT зазвичай потрібно надсилати дані через певні проміжки часу. При роботі з хмарою потрібно визначити, як часто надсилати дані в хмару. Blynk.Library пропонує власний таймер.

```
BlynkTimer timer; // Створення об'єкта таймера
void setup(){
timer.setInterval(1000L, myTimerEvent); // Запуск таймера
}
void myTimerEvent() // Цей цикл визначає, що відбувається, коли запускається таймер
{}
```

```
void loop(){
Blynk.run();
timer.run(); // запускає таймер у циклі
}
```

Надсилання/отримання даних. Мобільні програми та Web-панелі мають віджети, які можуть дистанційно контролювати пристрій.

У шаблоні використано чотири віджети, розміщені на інформаційних панелях. Кожен віджет призначається певному потоку даних.

Віджет кнопки записує 1 і 0 у потік даних V0 Datastream.

Два віджети міток, які зчитують значення з потоку даних V1 Datastream і потоку даних V2 Datastream.

Зображення Web-кнопки - це кнопка, до якої ви можете додавати власні зображення для ON та OFF стану. Після натискання можна відкрити веб-сторінку в Blynk.App

Кнопка. Наведений нижче код прослуховує дії від Blynk до V0 Datastream, а потім записує значення до змінної. Для цього використовується `BLYNK_WRITE(V0)`.

```
BLYNK_WRITE(V0)
{
   // Встановити вхідне значення з виводу V0 у змінну
int value = param.asInt();
   // Стан оновлення
Blynk.virtualWrite(V1, value);
}
```

Після запису значення Button також надсилаємо його до віджету Label за допомогою `Blynk.virtualWrite(V1, value)`. Ця функція щосекунди надсилає дані про роботу Arduino на віртуальний контакт Virtual Pin 2.

```
void myTimerEvent()
```

```
{
   // Ви можете надіслати будь-яке значення в будь-який час.
   // Будь ласка, не надсилайте більше 10 значень за секунду.
Blynk.virtualWrite(V2, millis() / 1000);
}
```

За допомогою Blynk можна змінювати різні параметри віджетів безпосередньо з обладнання.

```
BLYNK_CONNECTED()
{
   // Змінити повідомлення кнопки веб-посилання на "Вітаємо!"
Blynk.setProperty(V3, "offImageUrl", "https://static-image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");
Blynk.setProperty(V3, "onImageUrl", "https://static-image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_pressed.png");
Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-started/template-quick-setup");
}
```

У коді вище відстежується, коли пристрій підключається до Blynk.Cloud за допомогою `BLYNK_CONNECTED`. Коли пристрій успішно підключається до Blynk Cloud, змінюється URL-адреса зображення для відображення у віджеті, а також призначається URL-адреса для натискання

РОЗДІЛ 3
ОПИС ПРОГРАМИ ДЛЯ КОНТРОЛЮ ТА КЕРУВАННЯ ШТУЧНОЮ ЕКОСИСТЕМОЮ

3.1. Структура програмних основних функціональних модулів

При розробці програми керування штучною екосистемою використано наступні бібліотеки та модулі:

- Blynk – віддалене керування;
- ds3231FS – модуль годинника реального часу;
- IotWebConf – бібліотека для локального конфгурування;
- LiquidCrystal_I2C-master – модуль для роботи з LCD-дисплеєм;
- NTPClient – модуль NTP-клієнта для сінхронізації часу;
- OneWire – модуль шини One Wire;
- Time – бібліотека роботи з часом;
- Timer-master – логічні таймери.

Функція ініціалізації програми передбачає ініціалізацію усіх апаратних та програмних ресурсів контролера

```
void setup()
{
  String str;
  Serial.begin(115200);
```

Ініціалізація дискретних портів контролера

```
  pinMode(D8, OUTPUT);
  pinMode(D7, OUTPUT);
  pinMode(D5, OUTPUT);
```

```
pinMode(D3, OUTPUT);
digitalWrite(D8, HIGH);
digitalWrite(D7, HIGH);
digitalWrite(D5, HIGH);
digitalWrite(D3, HIGH);
pinMode(D6, INPUT_PULLUP);
delay(200);
```

Інінцалізація системи локального Web-конфігкратора

```
iotWebConf.addParameter(&intParam_Demo);
iotWebConf.addParameter(&separator1);
iotWebConf.addParameter(&intParam_Time_LampOn);
iotWebConf.addParameter(&intParam_Time_PumpOn);
iotWebConf.addParameter(&separator2);
iotWebConf.addParameter(&intParam_Time_PumpTime1_On);
iotWebConf.addParameter(&intParam_Time_PumpTime2_Off);
iotWebConf.addParameter(&intParam_Time_PumpTime3_On);
iotWebConf.setConfigSavedCallback(&configSaved);
iotWebConf.getApTimeoutParameter()->visible = true;
iotWebConf.init();
```

Якщо система знаходиться у режимі конфігурування запускається локальний Web-сервер

```
if(CFG_Mode )
{
  server.on("/", handleRoot);
  server.on("/config", []{ iotWebConf.handleConfig(); });
  server.onNotFound([](){ iotWebConf.handleNotFound(); });
}
```

Ініціалізація початковими значеннями параметрів налаштування

```
        Test_Mode=atoi(intParam_Demo.valueBuffer);
        Set_Time_LampOn = atoi(intParam_Time_LampOn.valueBuffer);
        Set_Time_PumpOn = atoi(intParam_Time_PumpOn.valueBuffer);
        Set_Time_PumpTime1_On   =   atoi(intParam_Time_PumpTime1_On.
valueBuffer);
        Set_Time_PumpTime2_Off  =  atoi(  intParam_Time_PumpTime2_Off.
valueBuffer);
        Set_Time_PumpTime3_On   =   atoi(intParam_Time_PumpTime3_On.
valueBuffer);
```

Створення WiFi – з’єднання

```
        WiFi.softAPdisconnect(true);
        WiFi.mode(WIFI_STA);
```

Ініціалізація інтерфейсу I2C для обміну з LCD-диспреєм та ініціалізація дісплею.

```
        Wire.begin();
        lcd.init();
        lcd.backlight();
        lcd.setCursor(0,0);
        lcd.print("Eco System");
```

Ініціалізація годинника реального часу на шині I2C

```
        DS3231_init(DS3231_INTCN);
```

Ініціалізація WiFi-з’єднання та сервісу NTP.

```
        WiFi.begin(ssid, password);
        while ( WiFi.status() != WL_CONNECTED )
        {    delay ( 500 );  }
```

```
  timeClient.begin();
  timeClient.setTimeOffset(2*3600L);
  time_begin = millis();
```

Ініціалізація датчика темеператури та логічних таймерів

```
  Init_DS18B20();
  logTimer.every(1000, TimeGetFunc);
  logTimer.every(2000, TermoGetFunc);
  logTimer.every(1000, MainEngine);
  logTimer.every(3000, TimerBlynkSend);
```

Виконання з’єднання Blynk з сервером

```
  Blynk.begin(auth, ssid, password);
  lcd.clear();
}
```

Головний цикл програми має наступний вигляд

```
void loop()
{
```

Якщо програма знаходиться у режимі локального конфігурування, то запускається цикл обробки локального Web-серверу

```
  if(CFG_Mode )
  {
    iotWebConf.doLoop();
    return;
  }
```

Для зберігання даних виконується їх перетворення у рядковий вигляд

```
      if( Need_SaveParam )
      {
        sprintf(intParam_Demo.valueBuffer, "%d",Test_Mode);
        sprintf(intParam_Time_LampOn.valueBuffer,          "%d",
Set_Time_LampOn);
        sprintf(intParam_Time_PumpOn.valueBuffer,          "%d",
Set_Time_PumpOn);
        sprintf(intParam_Time_PumpTime1_On.valueBuffer,    "%d",
Set_Time_PumpTime1_On);
        sprintf(intParam_Time_PumpTime2_Off.valueBuffer,   "%d",
Set_Time_PumpTime2_Off);
        sprintf(intParam_Time_PumpTime3_On.valueBuffer,    "%d",
Set_Time_PumpTime3_On);
        iotWebConf.configSave();
        Need_SaveParam=0;
      }
```

При першому вході у головний цикл виконується синхронізація чодинника з сервером

```
      if( first_step ) {
        while(!timeClient.update())
        {
          if(millis() - time_begin > 5000)
            break;
          timeClient.forceUpdate();
        }
        timeClient.end();
        first_step = 0;
        setTimeStruct(timeClient.getEpochTime(), &time_str);
        DS3231_set(time_str);
      }
```

Оновлення логічних таймерів

```
  logTimer.update();
```

Виконання обробника Blynk

```
  Blynk.run();
}
```

Таким чином в основному циклі програми виконується оновлення даних та взаємодія з сервером Blynk.

3.2. Модулі взаємодії з апаратним забезпеченням

Підключення файлів бібліотек та визначення основних елементів

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Створення об’єкту для відображення на рідкокристалічному дісплеї

```
LiquidCrystal_I2C lcd(0x27,20,4);  // встановіть адресу LCD-дисплея на 0x27 для відображення 16 символів і 2 рядки
```

Для виводу даних на дісплей використовується глобальна змінна для тексту, в яку розміщується текст в різних частинах програми, та функція очищення дисплею та виводу рядків.

```
String display[4];
uint16_t display_cnt=0;
void DisplayFunc()
{
  if(display_cnt++>30)
    lcd.clear();
  for(int i=0; i<4; i++)
  {
    lcd.setCursor(0,i);
    lcd.print(display[i]);
  }}
```

Мікроконтролер підключається до мережі за допомогою WiFi-з’єднання

```
char ssid[128]     = "ssid";
char password[64] = "password";
```

Для забезпечення синхронізації часу використовується NTP клієнт, який створюється наступним чином

```
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <ds3231.h>
ts time_str; // структура ts знаходиться в файлі ds3231.h
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);
```

Перетворення часу з мережевого формату до формату відображення виконується за допомогою функції

```
void setTimeStruct(unsigned long secs, struct ts *time_str)
{
  unsigned long rawTime = secs  / 86400L;  // in days
  unsigned long days = 0, year = 1970;
```

```
      uint8_t month;
      static                    const                    uint8_t
monthDays[]={31,28,31,30,31,30,31,31,30,31,30,31};

      while((days += (LEAP_YEAR(year) ? 366 : 365)) <= rawTime)
        year++;
      rawTime -= days - (LEAP_YEAR(year) ? 366 : 365); // now it
is days in this year, starting at 0
      days=0;
      for (month=0; month<12; month++)
      {
        uint8_t monthLength;
        if (month==1)
        { // лютий
          monthLength = LEAP_YEAR(year) ? 29 : 28;
        } else
        {
          monthLength = monthDays[month];
        }
        if (rawTime < monthLength) break;
        rawTime -= monthLength;
      }

      time_str->mday = rawTime+1;        /* day of the month */
      time_str->mon = month+1;         /* month */
      time_str->year = year;        /* year */
      time_str->sec = secs % 60;         /* seconds */
      time_str->min = (secs % 3600) / 60;         /* minutes */
      time_str->hour = (secs % 86400L) / 3600;        /* hours */
    }
```

Для вимірювання температури використовується датчик DS18B20. Для роботи з ним використовується бібліотека OneWire.h.

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

```
OneWire  ds(D4);
byte ow_addr[8];
byte ow_type_s=0;
```

Ініціалізація шини OneWire та пошук підключених пристроїв

```
void Init_DS18B20()
{
  byte i;
  if ( !ds.search(ow_addr)) {
    Serial.println("No more addresses.");
    ds.reset_search();
    delay(250);
    return;
  }}
  if (OneWire::crc8(ow_addr, 7) != ow_addr[7]) {
      Serial.println("CRC is not valid!");
      return;
  }
```

Визначення типу підключенного пристрою

```
  switch (ow_addr[0]) {
    case 0x10: // old DS1820
      ow_type_s = 1;
      break;
    case 0x28: // DS18B20
      ow_type_s = 0;
      break;
    case 0x22: // DS1822
      ow_type_s = 0;
      break;
    default:
```

```
            Serial.println("Device is not a DS18x20 family
device.");
            return;
        }
      }
```

Запуск перетворення для датчику DS18B20 виконується функцією

```
void StartConversion_DS18B20()
{
  ds.reset();
  ds.select(ow_addr);
  ds.write(0x44, 1);
}
```

Отримання значення температури та перетворення з внутрішнього представлення до десятичного формату

```
float termo=0.0;
void GetData_DS18B20()
{
  byte present = 0;
  byte data[12];
  byte i;
  present = ds.reset();
  ds.select(ow_addr);
  ds.write(0xBE);         // Read Scratchpad
  for ( i = 0; i < 9; i++)
  {           // зчитування 9 байт
    data[i] = ds.read();
  }
  int16_t dd = (data[1] << 8) | data[0];
  if (ow_type_s) {
    dd = dd << 3; // 9 біт даних
```

```
    if (data[7] == 0x10) { // 12 біт даних
      dd = (dd & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    if (cfg == 0x00) dd = dd & ~7;  // 9 біт даних
    else if (cfg == 0x20) dd = dd & ~3; // 10 біт даних
    else if (cfg == 0x40) dd = dd & ~1; // 11 біт даних
  }
  termo =  (float) dd / 16.0;
}
```

Метод зчитування стану аналогового входу та запуск перетворення температури і її відображення на дісплеї.

```
uint8_t flag_DS18B20=0;
uint16_t WaterHeight=0;
void TermoGetFunc()
{  char buf[24]="";
  WaterHeight = analogRead(A0);
  if(flag_DS18B20)
  {
    flag_DS18B20 = 0;
    GetData_DS18B20();
    sprintf(buf, "T=%.1f  H=%d    ",termo, WaterHeight);
    display[1]=buf;
  }  else  {
    flag_DS18B20 = 1;
    StartConversion_DS18B20();
  }}
```

Для локального визначення часу використовується модуль на основі мікросхеми DS3231 – годинника реального часу. Метод зчитування часу з мікросхеми та його відображення на діспеї представлено нижче

```
#include "Timer.h"
Timer logTimer;
void TimeGetFunc()
{
  char buf[24]="";
  DS3231_get(&time_str);
  sprintf(buf, "%02d/%02d/%04d %02d:%02d:%02d", time_str.mday,
time_str.mon,    time_str.year,    time_str.hour,    time_str.min,
time_str.sec);
  display[0] = buf;
}
```

Для управління насосом та освітленням випористовується автомат станів. Для роботи автомата станів необхідні наступні змінні для його ініціалізації та роботи:

```
uint32_t Lamp_time_cnt=0; // лічильник часу для освітлення
uint32_t Pump_time_cnt=0; // лічильник часу для насосу
uint16_t PumpCycle_time_cnt=0; // лічильник часу для циклів
насосу

uint32_t Day_Time_Max=24*60*60L;
uint16_t Set_Time_LampOn = 8*60; //[min] час, протягом якого
лампи включені
uint16_t Set_Time_PumpOn = 3*60;  //[min] період часу між
включеннями насоса
uint16_t Set_Time_PumpTime1_On = 15; //[s] час включеного
стану насоса 1 (накачування мулу)
uint16_t Set_Time_PumpTime2_Off = 60; //[s] час вимкненого
стану насоса (вода з мулом знаходиться в лотку)
uint16_t Set_Time_PumpTime3_On = 30; //[s] час увімкненого
стану насоса 2 (злив води)
uint8_t Lamp_On, Pump_On, Air_On, Heat_On;
```

```
uint8_t Air_flag = 0;
float termo_hist=-0.5;
```

Основна функція зміни стану автомата виконується кожну секунду

```
void MainEngine()
{
```

Керування освітленням

```
  if( Lamp_time_cnt < Set_Time_LampOn*60L )
    Lamp_On = 1;
  else
    Lamp_On = 0;
```

Керування насосом

```
  if( Pump_time_cnt > Set_Time_PumpOn*60L )
  {
```

Перший цикл роботи насосу

```
    if(PumpCycle_time_cnt < Set_Time_PumpTime1_On)
    {
      if( Air_flag ==0 )
      {
        Air_flag =1;
        if(Air_On) Air_On=0; else Air_On=1;
      }
      Pump_On = 1;
    }
```

Другий цикл роботи насосу

```
        else  if(  PumpCycle_time_cnt  <  (Set_Time_PumpTime2_Off
+Set_Time_PumpTime1_On)  )
        {
          Pump_On = 0;
        }
```

Третій цикл роботи насосу

```
        else  if(  PumpCycle_time_cnt  <  (Set_Time_PumpTime3_On
+Set_Time_PumpTime2_Off +Set_Time_PumpTime1_On)  )
        {
          Pump_On = 1;
        }
        else
        {
          Pump_On = 0;
          Pump_time_cnt = 0;
        }
        PumpCycle_time_cnt++;
      }
```

Завершення циклів роботи насосу

```
      else
      {
        Air_flag =0;
        PumpCycle_time_cnt = 0;
        Pump_time_cnt++;
      }
```

Лічильник часу для керування освітленням

```
      Lamp_time_cnt++;
```

```
  if(Lamp_time_cnt >= Day_Time_Max)
    Lamp_time_cnt=0;
```

Блок керування підіргрівом з гістерезисом

```
  if ( termo  > (23.0-termo_hist)  )
  {
    termo_hist=0.5;
    Heat_On = 0;
  }
  else
  {
    termo_hist=-0.5;
    Heat_On = 1;
  }
```

Формування вихідних сигналів на вихідні порти мікроконтролера

```
  if(Pump_On)  digitalWrite(D8,  LOW);  else  digitalWrite(D8,
HIGH);
  if(Lamp_On)  digitalWrite(D7,  LOW);  else  digitalWrite(D7,
HIGH);
  if(Air_On)  digitalWrite(D5,  LOW);  else  digitalWrite(D5,
HIGH);
  if(Heat_On)  digitalWrite(D3,  LOW);  else  digitalWrite(D3,
HIGH);
```

Оновлення стану дісплея

```
  DisplayFunc();
}
```

Для віддаленого моніторингу та керування системою використовується система Blynk. Для роботи системи необхідно отримати токен ідентифікації пристрою і додати його до програми.

```
char auth[128] = "token";
```

Нижче наведені методи для створення віртуальних сигналів для обміну даними.

Метод для віртуального сигналу V0 (час, протягом якого лампи включені):

```
uint8_t Need_SaveParam=0;
BLYNK_WRITE(V0)
{
  int Value = param.asInt();
  Set_Time_LampOn = Value;
  Need_SaveParam=1;
}
```

Метод для віртуального сигналу V1 [min] період часу між включеннями насоса

```
BLYNK_WRITE(V1)
{
  int Value = param.asInt();
  Set_Time_PumpOn = Value;
  Need_SaveParam=1;
}
```

Метод для віртуального сигналу V2 [s] час включеного стану насоса 1 (накачування мулу)

```
BLYNK_WRITE(V2)
{
  int Value = param.asInt();
  Set_Time_PumpTime1_On = Value;
  Need_SaveParam=1;
}
```

Метод для віртуального сигналу V3 [s] час вимкненого стану насоса (вода з мулом знаходиться в лотку)

```
BLYNK_WRITE(V3)
{
  int Value = param.asInt();
  Set_Time_PumpTime2_Off = Value;
  Need_SaveParam=1;
}
```

Метод для віртуального сигналу V4 [s] час увімкненого стану насоса 2 (злив води)

```
BLYNK_WRITE(V4)
{
  int Value = param.asInt();
  Set_Time_PumpTime3_On = Value;
  Need_SaveParam=1;
}
```

Створення віджетів для відображення стану:

- V5 – рівня води;
- V6 – температури;
- V7 – роботи освітлення;
- V8 – роботи насосу.

```
WidgetLED led_Lamp(V7);
WidgetLED led_Pump(V8);
WidgetLED led_Air(V9);
WidgetLED led_Heat(V10);
```

Функція періодичного формування віртуальних сигналів

```
void TimerBlynkSend(){
    String str="Blynk.virtualWrite ";
  switch(TimerBlynkSend_cnt)  {
  case 1:
    Blynk.virtualWrite(V5, WaterHeight);
    str += TimerBlynkSend_cnt + " = " + WaterHeight;
    break;
  case 2:
    Blynk.virtualWrite(V6, termo);
    str += TimerBlynkSend_cnt+ " = "+ termo;
    break;
  case 3:
    if(Lamp_On) led_Lamp.on(); else led_Lamp.off();
    str += TimerBlynkSend_cnt+" = "+ Lamp_On;
    break;
  case 4:
    if(Pump_On) led_Pump.on(); else led_Pump.off();
    str += TimerBlynkSend_cnt+ " = "+ Pump_On;
    break;
  default:
    TimerBlynkSend_cnt=0;
  }
  Serial.println(str);
  TimerBlynkSend_cnt++;
}
```

ВИСНОВКИ

У різних наукових та практичних застосуваннях використовуються закриті штучні екологічні системи, які не залежать від обміну речовиною з будь-якою частиною поза системою. Закриті штучні екосистеми зазвичай невеликі. Такі системи цікаві з наукової точки зору і потенційно можуть служити системою життєзабезпечення. У закритій екосистемі будь-які відходи, вироблені одним видом, повинні використовуватися принаймні одним іншим видом.

При проектуванні та будівництві закриті штучні екосистеми найважливішим критерієм є експлуатаційна надійність і стабільність, оскільки робочий процес закритої штучної екосистеми сприйнятливий до впливів навколишнього середовища, що призведе до зниження врожайності рослин і погіршення екологічних функцій, і навіть до поломки системи. Отже, необхідно розробити, як ефективно контролювати та регулювати штучні фактори навколишнього середовища, щоб закриті штучні екосистеми надійно працював у разі виникнення екологічних порушень. В даний час керування з відкритим і лінійним замкнутим контуром широко використовуються в закритих штучних екосистемах.

Система контролю та управління штучною екосистемою побудована на основі контролера ESP8266 та забезпечує управління фітолампою, світлодіодною лампою білого кольору, насосом для води, насосом для повітря та підігрівачем води.

Для віддаленого керування та моніторингу системи використовується система Blynk.

При розробці програми керування штучною екосистемою використано наступні бібліотеки та модулі: Blynk – віддалене керування, ds3231FS – модуль годинника реального часу, IotWebConf – бібліотека для локального конфгурування, LiquidCrystal_I2C-master – модуль для роботи з LCD-дисплеєм, NTPClient – модуль NTP-клієнта для сінхронізації часу, OneWire – модуль

шини One Wire, Time – бібліотека роботи з часом, Timer-master – логічні таймери.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Arduino Home page [Електронний ресурс]. – Режим доступу: www. URL: https://www.arduino.cc/ (дата звернення: 15.02.23)
2. Arduino Libraries [Електронний ресурс]. – Режим доступу: www. URL: https://www.arduino.cc/reference/en/libraries/ (дата звернення: 03.03.23)
3. Arduino core for the ESP32 [Електронний ресурс]. – Режим доступу: www. URL: https://github.com/espressif/arduino-esp32 (дата звернення:17.02.23)
4. A fully integrated suite of IoT software [Електронний ресурс]. – Режим доступу: www. URL: https://blynk.io/ (дата звернення: 01.03.23).
5. Blynk Documentation [Електронний ресурс]. – Режим доступу: www. URL: https://docs.blynk.io/ (дата звернення: 01.03.23).
6. Arduino Libraries Wifi [Електронний ресурс]. – Режим доступу: www. URL: https://www.arduino.cc/reference/en/libraries/wifi/ (дата звернення: 05.03.23)
7. Closed Ecological Systems [Електронний ресурс]. – Режим доступу: www. URL: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/closed-ecological-systems (дата звернення: 01.03.23)
8. Sample records for closed ecological system [Електронний ресурс]. – Режим доступу: www. URL: https://www.science.gov/ topicpages/c/ closed+ ecological+ system.html (дата звернення: 01.03.23)
9. Closed Ecosystem: Definition, Examples [Електронний ресурс]. – Режим доступу: www. URL: https://lambdageeks.com/closed-ecosystem/ (дата звернення: 01.03.23)

ДОДАТКИ

ДОДАТОК А

Вихідний код програми

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#include <Wire.h>
//============================
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to
0x27 for a 16 chars and 2 line display

String display[4];
uint16_t display_cnt=0;
void DisplayFunc()
{
  if(display_cnt++>30)
    lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(display[0]);
  lcd.setCursor(0,1);
  lcd.print(display[1]);
  lcd.setCursor(0,2);
  lcd.print(display[2]);
  lcd.setCursor(0,3);
  lcd.print(display[3]);
}

#include <ds3231.h>
ts time_str; //ts is a struct findable in ds3231.h

char ssid[128]     = "";
char password[64] = "";
```

```
#include <NTPClient.h>
#include <WiFiUdp.h>

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

void setTimeStruct(unsigned long secs, struct ts *time_str)
{
  unsigned long rawTime = secs  / 86400L;  // in days
  unsigned long days = 0, year = 1970;
  uint8_t month;
  static                    const                    uint8_t
monthDays[]={31,28,31,30,31,30,31,31,30,31,30,31};

  while((days += (LEAP_YEAR(year) ? 366 : 365)) <= rawTime)
    year++;
  rawTime -= days - (LEAP_YEAR(year) ? 366 : 365); // now it
is days in this year, starting at 0
  days=0;
  for (month=0; month<12; month++)
  {
    uint8_t monthLength;
    if (month==1)
    { // february
      monthLength = LEAP_YEAR(year) ? 29 : 28;
    } else
    {
      monthLength = monthDays[month];
    }
    if (rawTime < monthLength) break;
    rawTime -= monthLength;
  }

  time_str->mday = rawTime+1;        /* day of the month */
  time_str->mon = month+1;         /* month */
  time_str->year = year;        /* year */
```

```
  time_str->sec = secs % 60;         /* seconds */
  time_str->min = (secs % 3600) / 60;         /* minutes */
  time_str->hour = (secs % 86400L) / 3600;        /* hours */
}
//============================

int16_t time_begin;
int16_t first_step=1;
//============================

#include <OneWire.h>
OneWire  ds(D4);  // on pin 10 (a 4.7K resistor is necessary)

byte ow_addr[8];
byte ow_type_s=0;

void Init_DS18B20()
{

  byte i;
  if ( !ds.search(ow_addr)) {
    Serial.println("No more addresses.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }

  Serial.print("ROM =");
  for( i = 0; i < 8; i++) {
    Serial.write(' ');
    Serial.print(ow_addr[i], HEX);
  }

  if (OneWire::crc8(ow_addr, 7) != ow_addr[7]) {
      Serial.println("CRC is not valid!");
```

```
        return;
      }
      Serial.println();

      switch (ow_addr[0]) {
        case 0x10:
          Serial.println("  Chip = DS18S20");  // or old DS1820
          ow_type_s = 1;
          break;
        case 0x28:
          Serial.println("  Chip = DS18B20");
          ow_type_s = 0;
          break;
        case 0x22:
          Serial.println("  Chip = DS1822");
          ow_type_s = 0;
          break;
        default:
          Serial.println("Device   is   not   a   DS18x20   family
device.");
          return;
      }
    }

    void StartConversion_DS18B20()
    {
      ds.reset();
      ds.select(ow_addr);
      ds.write(0x44, 1);        // start conversion, with parasite
power on at the end
    }

    float termo=0.0;

    void GetData_DS18B20()
    {
```

```
  byte present = 0;
  byte data[12];
  byte i;
  present = ds.reset();
  ds.select(ow_addr);
  for ( i = 0; i < 9; i++)
  {           // we need 9 bytes
    data[i] = ds.read();
  }
  int16_t raw = (data[1] << 8) | data[0];
  if (ow_type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // "count remain" gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    // at lower res, the low bits are undefined, so let's zero
them
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution,
93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5
ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375
ms
    //// default is 12 bit resolution, 750 ms conversion time
  }
  termo =  (float)raw / 16.0;
}
uint8_t flag_DS18B20=0;
uint16_t WaterHeight=0;

void TermoGetFunc()
{
  char buf[24]="";
```

```
      WaterHeight = analogRead(A0);
      if(flag_DS18B20)
      {
        flag_DS18B20 = 0;
        GetData_DS18B20();
        sprintf(buf, "T=%.1f  H=%d    ",termo, WaterHeight);
        display[1]=buf;
      }
      else
      {
        flag_DS18B20 = 1;
        StartConversion_DS18B20();
      }
    }
    //============================
    #include "Timer.h"
    Timer logTimer;
    void TimeGetFunc()
    {
      char buf[24]="";

      DS3231_get(&time_str);
      sprintf(buf, "%02d/%02d/%04d %02d:%02d:%02d", time_str.mday,
time_str.mon,    time_str.year,    time_str.hour,    time_str.min,
time_str.sec);
      display[0] = buf;
    }
    //============================
    uint32_t Lamp_time_cnt=0;
    uint32_t Pump_time_cnt=0;
    uint16_t PumpCycle_time_cnt=0;

    uint8_t  Test_Mode=1;
    uint32_t Day_Time_Max=24*60*60L;
    uint16_t  Set_Time_LampOn  =  8*60;  //[min]  время,  в  течение
которого лампы включены
```

```
    uint16_t Set_Time_PumpOn = 3*60;  //[min] период времени между включениями насоса
    uint16_t Set_Time_PumpTime1_On = 15; //[s] время включенного состояния насоса 1 (накачивание ила)
    uint16_t Set_Time_PumpTime2_Off = 60; //[s] время выключенного состояния насоса (вода с илом находится в лотке)
    uint16_t Set_Time_PumpTime3_On = 30; //[s] время включенного состояния насоса 2 (слив воды)

    uint8_t Lamp_On, Pump_On, Air_On, Heat_On;
    uint8_t Air_flag = 0;
    void MainEngine() // исполняется каждую секунду
    {
      if( Lamp_time_cnt < Set_Time_LampOn*60L )
        Lamp_On = 1;
      else
        Lamp_On = 0;
      if( Pump_time_cnt > Set_Time_PumpOn*60L )
        if(PumpCycle_time_cnt < Set_Time_PumpTime1_On)
        {
          if( Air_flag ==0 )
          {
            Air_flag =1;
            if(Air_On) Air_On=0; else Air_On=1;
          Pump_On = 1;
        }
        else if( PumpCycle_time_cnt < (Set_Time_PumpTime2_Off +Set_Time_PumpTime1_On) )
          Pump_On = 0;
        }
        else if( PumpCycle_time_cnt < (Set_Time_PumpTime3_On +Set_Time_PumpTime2_Off + Set_Time_PumpTime1_On) )
        {
          Pump_On = 1;
        }
        else
```

```
          {
            Pump_On = 0;
            Pump_time_cnt = 0;
          }
          PumpCycle_time_cnt++;
        }
        else
        {
          Air_flag =0;
          PumpCycle_time_cnt = 0;
          Pump_time_cnt++;
        }

        Lamp_time_cnt++;
        if(Lamp_time_cnt >= Day_Time_Max)
          Lamp_time_cnt=0;
        if ( termo  > (23.0-termo_hist)  )
        {
          termo_hist=0.5;
          Heat_On = 0;
        }
        else
        {
          termo_hist=-0.5;
          Heat_On = 1;
        }
        if(Pump_On)  digitalWrite(D8,  LOW);  else  digitalWrite(D8,
HIGH);
        if(Lamp_On)  digitalWrite(D7,  LOW);  else  digitalWrite(D7,
HIGH);
        if(Air_On)  digitalWrite(D5,  LOW);  else  digitalWrite(D5,
HIGH);
        if(Heat_On)  digitalWrite(D3,  LOW);  else  digitalWrite(D3,
HIGH);
        DisplayFunc();
      }
```

```
uint8_t Need_SaveParam=0;
BLYNK_WRITE(V0)
{
  int Value = param.asInt(); // assigning incoming value from
pin V1 to a variable
  Set_Time_LampOn = Value;
  Need_SaveParam=1;
}
BLYNK_WRITE(V1)
{
  int Value = param.asInt(); // assigning incoming value from
pin V1 to a variable
  Set_Time_PumpOn = Value;
  Need_SaveParam=1;
}
BLYNK_WRITE(V2)
{
  int Value = param.asInt(); // assigning incoming value from
pin V1 to a variable
  Set_Time_PumpTime1_On = Value;
  Need_SaveParam=1;
}
BLYNK_WRITE(V3)
{
  int Value = param.asInt(); // assigning incoming value from
pin V1 to a variable
  Set_Time_PumpTime2_Off = Value;
  Need_SaveParam=1;
}
BLYNK_WRITE(V4)
{
  int Value = param.asInt(); // assigning incoming value from
pin V1 to a variable
  Set_Time_PumpTime3_On = Value;
  Need_SaveParam=1;
}
```

```
BLYNK_WRITE(V11)
{
  int Value = param.asInt(); // assigning incoming value from
pin V1 to a variable
  Test_Mode = Value;
  Need_SaveParam=1;
}

WidgetLED led_Lamp(V7);
WidgetLED led_Pump(V8);
WidgetLED led_Air(V9);
WidgetLED led_Heat(V10);

uint8_t TimerBlynkSend_cnt=0;
void TimerBlynkSend()
{
    String str="Blynk.virtualWrite ";
  switch(TimerBlynkSend_cnt)
  {
  case 1:
    Blynk.virtualWrite(V5, WaterHeight);
    str += TimerBlynkSend_cnt;
    str += " = ";
    str += WaterHeight;
    break;
  case 2:
    Blynk.virtualWrite(V6, termo);
    str += TimerBlynkSend_cnt;
    str += " = ";
    str += termo;
    break;
  case 3:
    //Blynk.virtualWrite(V7, Lamp_On);
    if(Lamp_On) led_Lamp.on(); else led_Lamp.off();
    str += TimerBlynkSend_cnt;
```

```
        str += " = ";
        str += Lamp_On;
        break;
      case 4:
        if(Pump_On) led_Pump.on(); else led_Pump.off();
        str += TimerBlynkSend_cnt;
        str += " = ";
        str += Pump_On;
        break;
      case 5:
        if(Air_On) led_Air.on(); else led_Air.off();
        break;
      case 6:
        if(Heat_On) led_Heat.on(); else led_Heat.off();
        break;
      default:
        TimerBlynkSend_cnt=0;
      }
      Serial.println(str);

      TimerBlynkSend_cnt++;
    }

    #include <IotWebConf.h>

    const char thingName[] = "EcoSystem";
    const char wifiInitialApPassword[] = "12345678";

    DNSServer dnsServer;
    WebServer server(80);

    #define STRING_LEN 128
    #define NUMBER_LEN 32
    // -- Configuration specific key. The value should be modified
if config structure was changed.
    #define CONFIG_VERSION "es_v1"
```

```
IotWebConf iotWebConf(thingName, &dnsServer, &server, wifiInitialApPassword, CONFIG_VERSION);

char stringParamValue[STRING_LEN];
char intParamValue[NUMBER_LEN];
char floatParamValue[NUMBER_LEN];

#define PARAM_STRING_LEN 16
char strTest_Mode[PARAM_STRING_LEN]="1";
char strSet_Time_LampOn[PARAM_STRING_LEN]="480"; //[min] время, в течение которого лампы включены
char strSet_Time_PumpOn[PARAM_STRING_LEN]="180"; //[min] период времени между включениями насоса
char strSet_Time_PumpTime1_On[PARAM_STRING_LEN]="15"; //[s] время включенного состояния насоса 1 (накачивание ила)
char strSet_Time_PumpTime2_Off[PARAM_STRING_LEN]="30"; //[s] время выключенного состояния насоса (вода с илом находится в лотке)
char strSet_Time_PumpTime3_On[PARAM_STRING_LEN]="25"; //[s] время включенного состояния насоса 2 (слив воды)

IotWebConfParameter intParam_Demo = IotWebConfParameter("Демо режим", "Test_Mode", strTest_Mode, PARAM_STRING_LEN, "number", "0..1", NULL, "min='0' max='1' step='1'");

IotWebConfSeparator separator1 = IotWebConfSeparator("Основные настройки");

IotWebConfSeparator separator2 = IotWebConfSeparator("Настройки насоса");
IotWebConfParameter intParam_Time_PumpTime1_On = IotWebConfParameter("Время набора воды [сек]", "PumpTime1_On", strSet_Time_PumpTime1_On, PARAM_STRING_LEN, "number", "0..1000", NULL, "min='0' max='1000' step='1'");
IotWebConfParameter intParam_Time_PumpTime2_Off = IotWebConfParameter("Время паузы [сек]", "PumpTime2_Off",
```

```
strSet_Time_PumpTime2_Off, PARAM_STRING_LEN, "number", "0..1000",
NULL, "min='0' max='1000' step='1'");

     int16_t CFG_Mode=0;

     void handleRoot()
     {

     // -- Let IotWebConf test and handle captive portal requests.
       if (iotWebConf.handleCaptivePortal())
       {
         // -- Captive portal request were already served.
         return;
       }
       String s = "<!DOCTYPE html><html lang=\"en\"><head><meta
name=\"viewport\" content=\"width=device-width, initial-scale=1,
user-scalable=no\"/>";
      s += "</body></html>\n";
       server.send(200, "text/html", s);
     }

     void configSaved()
     {
       Serial.println("Configuration was updated.");
     }
     //==========================
     void setup()
     {
       String str;
       Serial.begin(115200);
       Serial.println();
       pinMode(D8, OUTPUT);
       pinMode(D7, OUTPUT);
       pinMode(D5, OUTPUT);
       pinMode(D3, OUTPUT);
```

```
digitalWrite(D8, HIGH);
digitalWrite(D7, HIGH);
digitalWrite(D5, HIGH);
digitalWrite(D3, HIGH);
Serial.println("\nEcoSystem");
pinMode(D6, INPUT_PULLUP);
delay(200);

iotWebConf.addParameter(&intParam_Demo);
iotWebConf.addParameter(&separator1);
iotWebConf.addParameter(&intParam_Time_LampOn);
iotWebConf.addParameter(&intParam_Time_PumpOn);
iotWebConf.addParameter(&intParam_Time_PumpTime1_On);
iotWebConf.setConfigSavedCallback(&configSaved);
iotWebConf.getApTimeoutParameter()->visible = true;

iotWebConf.init();

if(CFG_Mode )
{
  server.on("/", handleRoot);
  server.on("/config", []{ iotWebConf.handleConfig(); });
  server.onNotFound([](){ iotWebConf.handleNotFound(); });
}
Serial.println("Ready.");
if(CFG_Mode )
{
  return;
}

IotWebConfParameter *param;
param = iotWebConf.getWifiSsidParameter();
Serial.print("getWifiSsidParameter = ");
Serial.println(param->valueBuffer);
if(iotWebConf.getWifiSsidParameter()->valueBuffer)
```

```
        strcpy(ssid,            iotWebConf.getWifiSsidParameter()-
>valueBuffer);
      str = "ssid = ";
      str+=ssid;
      Serial.println(str);
      if(iotWebConf.getWifiPasswordParameter()->valueBuffer)
        strcpy(password,    iotWebConf.getWifiPasswordParameter()-
>valueBuffer);
      str = "password = ";
      str+=password;
      Serial.println(str);

      Test_Mode=atoi(intParam_Demo.valueBuffer);
      Set_Time_LampOn = atoi(intParam_Time_LampOn.valueBuffer);
      Set_Time_PumpTime1_On   =   atoi(intParam_Time_PumpTime1_On.
valueBuffer);
      Set_Time_PumpTime2_Off  =  atoi(intParam_Time_PumpTime2_Off.
valueBuffer);
      Set_Time_PumpTime3_On   =   atoi(intParam_Time_PumpTime3_On.
valueBuffer);

      Serial.println("Parametrs:");
      str = "Test_Mode=";
      str+= Test_Mode;
      Serial.println(str);

      if(Test_Mode)
      {
        Serial.println("Select Demo Mode");
        Day_Time_Max=4*60L;
        Set_Time_LampOn = 2; //[min] время, в течение которого
        Set_Time_PumpTime1_On  =  15;  //[s]  время  включенного
состояния насоса 1 (накачивание ила)
        Set_Time_PumpTime2_Off  =  20;  //[s]  время  выключенного
состояния насоса (вода с илом находится в лотке)
```

```
        Set_Time_PumpTime3_On = 30; //[s] время включенного
состояния насоса 2 (слив воды)
      }

      Serial.println("Parametrs:");
      str = "Test_Mode=";
      str+= Test_Mode;
      str+="\nSet_Time_LampOn=";
      str+=Set_Time_LampOn;
      str+="\nSet_Time_PumpTime2_Off=";
      str+=Set_Time_PumpTime2_Off;
      str+="\nSet_Time_PumpTime3_On=";
      str+=Set_Time_PumpTime3_On;
      Serial.println(str);
       //======================
      WiFi.softAPdisconnect(true);
      WiFi.mode(WIFI_STA);
      Serial.println("ConfigMode Close.");
      Wire.begin(); //start i2c (required for connection)
      lcd.init();
      lcd.backlight();
      lcd.setCursor(0,0);
      lcd.print("Eco System");
      //----------------
      DS3231_init(DS3231_INTCN);     //register     the     ds3231
(DS3231_INTCN is the default address of ds3231, this is set by
macro for no performance loss)

      //----------------
      lcd.setCursor(0,1);
      str = "Connect to ";
      str += ssid;
      lcd.print(str);

      WiFi.begin(ssid, password);
```

```
  while ( WiFi.status() != WL_CONNECTED )
  {
    delay ( 500 );
    Serial.print ( "." );
  }
  timeClient.begin();
  timeClient.setTimeOffset(2*3600L);
  time_begin = millis();
  Init_DS18B20();
  logTimer.every(1000, TimeGetFunc);
  logTimer.every(2000, TermoGetFunc);
  logTimer.every(1000, MainEngine);
  logTimer.every(3000, TimerBlynkSend);

  lcd.setCursor(0,2);
  lcd.print("Connect to Blynk");
  Blynk.begin(auth, ssid, password);
  lcd.clear();
}
void loop()
{

  if(CFG_Mode )
  {
    iotWebConf.doLoop();
    return;
  }
  if( Need_SaveParam )
  {
    sprintf(intParam_Demo.valueBuffer, "%d",Test_Mode);
    sprintf(intParam_Time_LampOn.valueBuffer,
"%d",Set_Time_LampOn);
    sprintf(intParam_Time_PumpOn.valueBuffer,
"%d",Set_Time_PumpOn);
```

```
          sprintf(intParam_Time_PumpTime1_On.valueBuffer,
"%d",Set_Time_PumpTime1_On);
          iotWebConf.configSave();
          Need_SaveParam=0;
        }

        if( first_step )
        {
          while(!timeClient.update())
          {
            if(millis() - time_begin > 5000)
              break;
            timeClient.forceUpdate();
          }
          timeClient.end();

          first_step = 0;
          setTimeStruct(timeClient.getEpochTime(), &time_str);
          DS3231_set(time_str);
        }
        logTimer.update();
        Blynk.run();
      }
```