Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Пояснювальна записка

до бакалаврської дипломної роботи

на тему: «Комп'ютеризована система для управління освітленням в оранжереї»

Виконав: студент 2 курсу, групи 1КІ-20мс
напряму підготовки (спеціальності)
123 – «Комп'ютерна інженерія»
Храпко Я.О.
Керівник к.т.н., доц. Колесник І.С.
Рецензент д.т.н., проф. Яремчук Ю.Є.

Допущено до захисту
д.т.н., проф. Азаров О.Д.

"17" 06 2022 р.

м. Вінниця – 2022 року

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень — бакалавр

Спеціальність — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри

обчислювальної техніки

проф., д.т.н. О.Д.Азаров

«09» 02 2022 р.

З А В Д А Н Н Я

НА БАКАЛАВСЬКУ ДИПЛОМНУ РОБОТУ

Храпко Яні Олександрівні

1. Тема роботи «Комп'ютеризована система для управління освітленням в оранжереї» , керівник роботи Колесник Ірина Сергіївна, к.т.н., доцент, затверджені наказом вищого навчального закладу від від 24.03.22 року №66

2 Строк подання студентом роботи: 13.06.2022р.

3. Вихідні дані до роботи: Технічні параметри систем управління, технічний опис HTML, технічна документація коди бібліотек та електронні компоненти.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ, вплив світла на рослини і мікроорганізми, мікропроцесорна система та програмне забезпечення комп'ютеризованої системи управління освітленням, проектування комп'ютеризованої системи управління освітленням, висновки,

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

7. Дата видачі завдання 10.02.2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задач роботи	14.02.22	вик.
2	Огляд інформаційних джерел	15.02-28.02.22	вик.
3	Огляд і аналіз сучасного стану галузі	01.03-14.03.22	вик.
4	Проектування системи управління будинком	15.03-28.03.22	вик.
5	програмне забезпечення системи управління будинком	29.03-11.04.22	вик.
6	Оформлення пояснювальної записки та ілюстративного матеріалу	12.04-25.04.22	вик.
7	Аналіз виконання роботи. Висновки. Додатки	26.04-09.05.22	вик.
8	Перевірка якості виконання бакалаврської роботи та усунення недоліків	17.05.22	вик.

Студент _____ Я.О. Храпко
(підпис) (прізвище та ініціали)

Керівник роботи _____ І.С. Колесник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота присвячена створенню комп’ютеризованої системи для управління освітленням в оранжереї.

Метою роботи є розробка комп’ютеризованої системи для управління освітленням в оранжереї.

Проаналізовано сучасні джерела випромінювання та доведено необхідність розробки комп’ютеризованої системи для управління освітленням в оранжереї.

Досліджено основні джерела випромінювання та способи управління світловим потоком.

Розроблено мікропроцесорну систему на основі мікроконтролера STM32F103C8T6.

Розроблено алгоритми для мікропроцесорної системи та програмного забезпечення комп’ютеризованої системи для управління освітленням.

Підібрано необхідний перелік електронних компонентів.

Розроблено та перевірено на працездатність програми на мікроконтролер серії STM32F1XX (IDE Eclipse Oxygen 2018.1) та на ПК (IDE Qt 5.9.1.).

Ключові слова: комп’ютеризована, система, управління, освітлення, мікропроцесор, мікроконтролер..

ANNOTATION

The bachelor's thesis is dedicated to the creation of a computerized system for lighting control in the greenhouse.

The aim of the work is to develop a computerized system for lighting control in the greenhouse.

Modern radiation sources are analyzed and the need to develop a computerized system for lighting control in the greenhouse is proved.

The main sources of radiation and methods of light flux control are studied.

A microprocessor system based on the STM32F103C8T6 microcontroller has been developed.

Algorithms for microprocessor system and computerized software for lighting control have been developed.

The necessary list of electronic components is selected.

Developed and tested for functionality of the STM32F1XX series microcontroller (IDE Eclipse Oxygen 2018.1) and PC (IDE Qt 5.9.1.).

Keywords: computerized, system, control, lighting, microprocessor, microcontroller

ЗМІСТ

ВСТУП

Як відомо рослини по-різному реагують на холод, спеку, вологість, склад грунту, параметри випромінювання (світло, радіація), механічні дії [1]. Кожен з цих параметрів важливий і впливає на розвиток рослини, а також на кількість і якість плодів, що цікаво в першу чергу постачальникам овочів, фруктів, квітів і т.д. Одним із цих параметрів є світло. Недолік або надлишок світла сильно впливає на життя і самопочуття рослини. Для кожної рослини ці пороги свої і для їх виявлення потрібні експерименти. Але найцікавішим для практичного застосування є знаходження оптимального режиму інтенсивності випромінювання світла різних довжин хвиль (складання спектру) для рослини, а також його зміна в залежності від стадії проростання рослини, де воно може дати максимальний урожай [2]. При цьому, для знаходження такого режиму необхідний пристрій, який міг би задавати спектральну щільність випромінювання.

Для мікроорганізмів все ще більш неоднозначно, всі мікроорганізми зовсім по-різному реагують на те, або інше випромінювання і інтенсивність цього випромінювання. Наприклад, бактерії, здатні до фототаксису при зміні інтенсивності випромінювання з певною довжиною хвилі, різко змінюють напрямок руху. Рухаються або до світла, або від нього, залежить від умов та виду самої бактерії. Звичайно це тільки один з прикладів, бувають і інші реакції.

На сьогоднішній день пропонується кілька видів джерел штучного випромінювання — освітлення фітосвітильниками на основі білих світлодіодів або на основі червоних і синіх, натрієві лампи високого тиску (наприклад дугові натрієві трубчасті (ДНаТ)), натрієві лампи низького тиску, люмінесцентні лампи, лампи розжарювання та ін. Також на виробництві з вирощування культур використовують регульовані інтенсивності синього, зеленого та червоного кольору, що звичайно добре, але все ж таки не достатньо. Система для досліджень повинна сама вміти змінювати щільність спектру, залежно від

часу, що налаштовується людиною, також повинна бути можливість модульності світлодіодів, а саме створення моделей в УФ, видимому та ІЧ діапазонах.

Виходячи з вищесказаного, було прийнято рішення необхідності розробки апаратно-програмного комплексу регулювання щільності спектра на модульній основі зі зміною інтенсивності випромінювання в часі.

Метою бакалаврської роботи є розробка комп’ютеризованої системи для управління освітленням в оранжереї.

Ефективність апаратно-програмного комплексу обумовлюється можливістю проведення дослідження інтенсивності світла на врожай рослини, середня кількість, розмір і якість плодів, що надалі, при використанні розрахованих параметрів, може збільшити якісну та кількісну характеристику врожаю. З таких міркувань ефективно досліджувати вплив випромінювання на мікроорганізми, які згодом можуть бути використані для отримання стимулюючої сироватки для тварин або інших корисних продуктів.

Необхідно вирішити такі **задачі:**

— аналіз впливу світла на рослини та міркоогранізми;

— огляд та вибір мікропроцесорни засобів та програмних середовищ розробки програмного забезпечення;

— проектування апаратно-програмної частин системи;

— проектування інтерфейсу користувача.

Об'єкт дослідження — комп’ютеризовані системи управління освітленням.

Предмет дослідження — методи та засоби проектування комп’ютеризованих систем управління освітленням.

1 ВПЛИВ СВІТЛА НА РОСЛИНИ І МІКРООРГАНІЗМИ

1.1 Вплив світла на рослини

Світло грає величезну роль життя рослин. При розгляді світлокультури огірків і помідорів можна відзначити деякі особливості: співвідношення синього (400-500 нм), зеленого, жовтого (500-600 нм) і червоного (600-700 нм) випромінювання 40:40:20% є прийнятним для огірків, хоча для помідорів ці значення будуть наступними 65:15:20%. Якщо на огірки світити довго світити червоним світлом у відношенні з іншими світловими складовими понад 40%, останні почнуть в'янути [3].

Висновок такий, що огіркам необхідно налаштовувати випромінювання червоної складової спектру. Помідори, навпаки, будуть почуватися краще під великою кількістю червоного світла.

Також цікавий той факт, що за допомогою інтенсивності певного спектра світла можна керувати «станами» рослини, а саме фотоморфогенез. Фотоморфогенез [4] — це процеси, що відбуваються в рослині під впливом світла різного спектрального складу та інтенсивності. Вони світло постає як сигнальний засіб, що регулює процеси зростання та розвитку рослини. Наприклад взаємодія червоного світла (660 нм) і далекого червоного світла (730 нм) для більшості рослин працює як перемикач, кажучи рослині, який зараз час доби і які процеси всередині рослини повинні бути запущені [5].

У природних умовах тінь від листя поряд рослин, що стоять, пропускає більше далекого червоного, ніж ближнього, і у більшості світлолюбних рослин запускається «синдром уникнення тіні» [6] — рослина тягнеться вгору. Помідорам під час зростання далекий червоний потрібен, щоб збільшити зростання і загальну займану площу, що природно збільшить урожай надалі.

У результаті можна зробити висновок, що світло може використовуватися як важіль для керування рослиною, але тільки проблема полягає в тому, що для кожної рослини потрібний індивідуальний підхід.

Фундаментальний закон екології «бочка Лібіха» [6] свідчить: розвиток обмежує фактор, сильніший за інших, що відхиляється від норми. Якщо за всіх нормальних оптимальних зовнішніх умов рослини, інтенсивність освітлення становить 30% від оптимального значення — рослина дасть трохи більше 30% максимально можливого врожаю.

Реакція рослини на світло та інші фактори — визначається лабораторним шляхом. Відгуки характеризують як фотосинтез, а й процеси зростання, цвітіння, синтезу необхідних смаку і аромату речовин.

Також впливає на щільність спектру поглинання рослиною та зовнішнє середовище (температура, вологість та ін.). Наприклад правильне забезпечення рослин мінеральними речовинами на холодних ґрунтах Півночі та Заполяр'я підвищує використання рослинами променистої енергії [1].

1.2. Вплив світла на мікроорганізми

Видимий світло впливає поведінка фототрофних бактерій. У них спостерігається явище фототаксису: бактерії здатні реагувати на зміну спектрального складу світла або освітленості. У еубактерій фоторецепторами служать бактеріохлорофіл і каротиноїди, тобто ті ж пігменти, які поглинають світло у процесі фотосинтезу. У архебактерій виявлено спеціальні сенсорні пігменти. Бактеріальні клітини, здатні до фототаксису, різко змінюють напрямок руху, якщо потрапляють з більш освітленої ділянки менш освітлений або зовсім неосвітлений. У результаті бактерії рухаються до світла — це позитивний фототаксис. Якщо бактерії рухаються у бік зменшення освітленості, то це — негативний фототаксис. Зміна сили світла може призвести до зміни швидкості руху бактерій, це називають фотокінез. У реакціях фототаксису у еубактерій беруть участь реакційні центри фотосинтезу. Мутанти з порушеними реакційними центрами втрачають здатність до фототаксису. У цьому відношенні фототрофні бактерії відрізняються від рухомих фототрофних еукаріотів, фототаксис яких не пов'язаний з фотосинтезом [7].

Таким чином, енергія світла може бути використана бактеріями. Шляхи її використання у прокаріотів значно різноманітніше, ніж у еукаріотів, де відомий тільки один вид оксигенного фотосинтезу. Для деяких бактерій, не здатних використовувати енергію світла, він служить як регулятор певних процесів обміну.

Так, у бактерії Pseudomonas putiola спостерігали активацію світлом деяких ферментів метаболізму, що розглядали як адаптацію, оскільки саме при освітленні починається фотосинтез фітопланктону, продукти якого використовуються гетеротрофними водними бактеріями.

Можна спостерігати різний вплив на біологічні ефекти бактерій [8] (рис. 1.1).

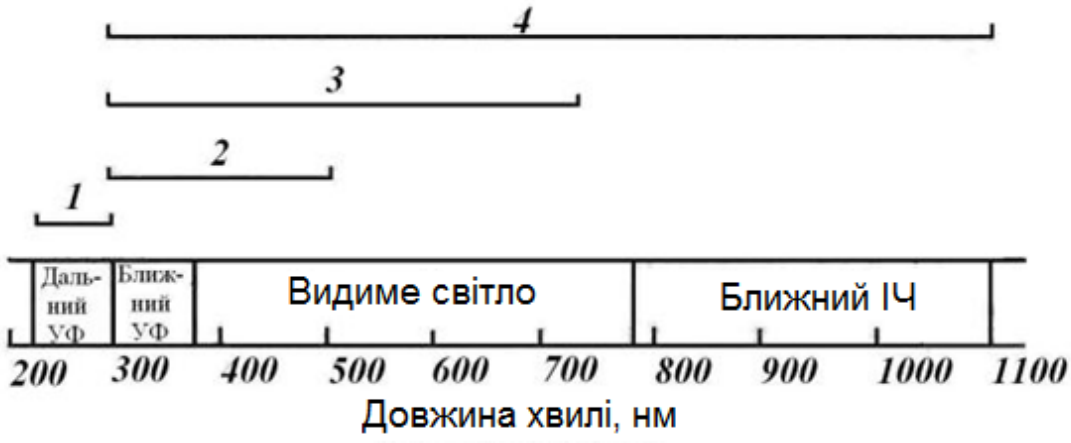


Рисунок 1.1 — Біологічні ефекти, що викликаються випромінюванням різної довжини хвилі: 1 - пошкодження ДНК і білків; 2 – фотореактивація ДНК; 3 – фототаксис та фотосинтез еукаріотів; 4 – фототаксис та фотосинтез прокаріотів.

1.3. Штучне освітлення

При штучному освітленні рослин використовуються, переважно, електричні джерела світла, розроблені для стимулювання зростання рослин з допомогою підбору спектра, сприятливого для фотосинтезу більшості ростен. Джерела освітлення використовуються при повній відсутності природного світла або при його нестачі. Взимку, коли інтенсивності та тривалості світла недостатньо для

росту рослин, штучне освітлення дозволяє збільшити тривалість та інтенсивність їх світлового опромінення.

Вперше застосував 1868 року гасові лампи для вирощування рослин російський ботанік Андрій Фамінцин.

Штучне світло має забезпечувати той спектр електромагнітного випромінювання, який рослини в природі отримують від сонця, або хоча б такий спектр, який задовольняв би потреби рослин, що вирощуються. Вуличні умови імітуються не лише шляхом підбору спектральних характеристик, але й зміною інтенсивності світіння ламп. Залежно від виду вирощуваної рослини, його стадії розвитку (проростання, зростання, цвітіння або дозрівання плодів), а також поточного фотоперіоду потрібен особливий спектр, світлова віддача та ін.

Застосовуються лампи різних типів, включаючи металогалогенні, люмінесцентні, розжарювання, натрієві високого тиску та світлодіодні.

Кожен із типів ламп освітлення має свою характерну спектральну щільність (рис. 1.2).

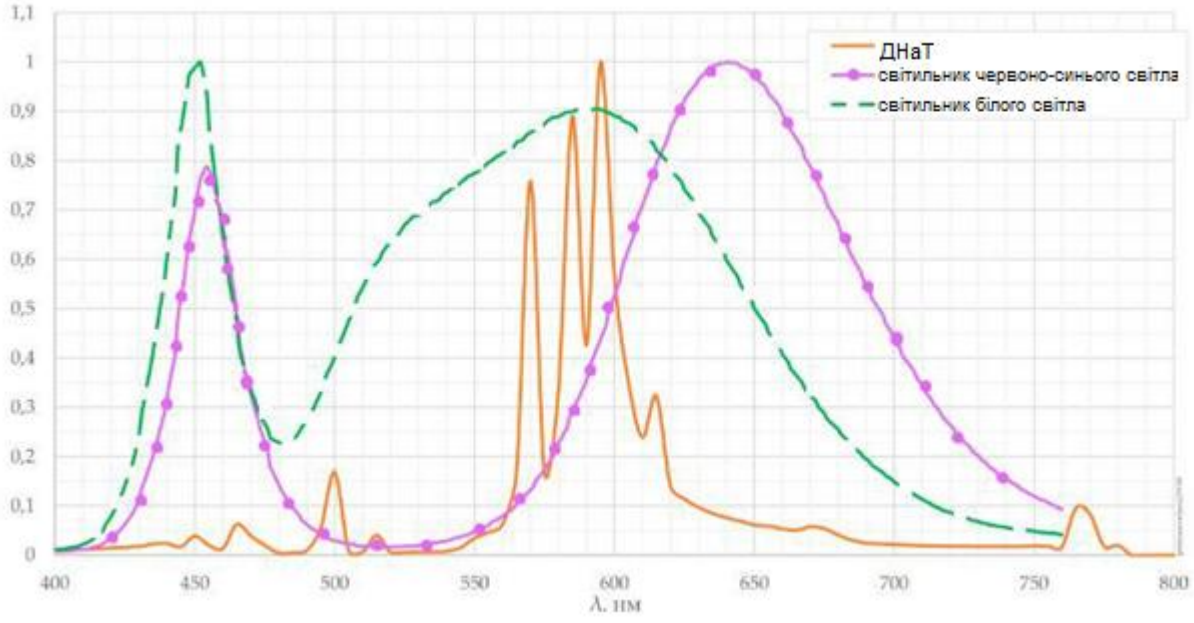


Рисунок 1.2 — Порівняльні параметри типового натрієвого світильника 600Вт для теплиць, спеціалізованого світлодіодного фітосвітильника та світильника для загального освітлення приміщень

Для вирощування рослин часто змінюють інтенсивність світіння джерела освітлення для отримання будь-яких результатів, але при цьому змінюється рівень по всьому спектру, що для адекватної постановки лабораторного експерименту недостатньо, навіть якщо використовувати в дослідах різні сучасні світильники.

Необхідність вивчення впливу світла на рослини не викликає змін, т.к. продукти харчування, лікувальні трави та інші інші подібні ресурси людство вже давно знайшло у рослинах. Адже для поліпшення кількості та якості цих продуктів можна використовувати спектральний режим, знайдений внаслідок експериментів.

Дослідження впливу світла на мікроорганізми обумовлено виготовленням з мікроорганізмів медичних лікувальних препаратів, їх контролем якості, також у діагностичних цілях, наприклад для виявлення окремих видів культур бактерій та ін До найбільш відомих промислових продуктів мікробного синтезу відносяться: ацетон , спирти (етанол, бута-нол, ізопропанол, гліцерин), органічні кислоти (лимонна, оцтова, молочна, глюконова, ітаконова, пропіонова), ароматизатори та речовини, що посилюють запахи (глутамат натрію) [9]. Так що мікроорганізми як вбивають, так і годують людство і вивчення впливу мікроорганізмів цілком виправдане, тим більше поле тут взагалі погано орано.

Для експериментів над мікроорганізмами природно, як можна здогадатися з вищесказаного необхідні ще УФ та ІЧ установки.

2 МІКРОПРОЦЕСОРНА СИСТЕМА ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ОСВІТЛЕННЯМ

2.1 Функції комп'ютеризованої системи для управління освітленням

Для того щоб провести експеримент над рослинами або мікроорганізмами за впливом спектра щільності інтенсивностей залежно від довжини хвилі на них необхідно дати можливість лаборанту налаштовувати інтенсивність випромінювання світла різних довжин хвиль і налаштовувати цей спектр у часі. Так, як біологічні реакції рослин вимірюються годинами, а мікроорганізми дадуть відгук у чашці Петрі [9] лише через якийсь час, то звіту часу в хвилинах буде цілком достатньо для вирішення цього завдання.

Через різні умови експериментів може знадобитися необхідність автономної роботи комплексу від акумулятора. Тим самим пояснюється те, що пристрій повинен мати якусь енергонезалежну пам'ять, щоб зберігати значення щільності спектра випромінювання, а також звіти часу, тобто. через який проміжок часу потрібно змінити значення інтенсивності світла у будь-якого модуля, який відповідає за конкретну довжину хвилі.

Пристрої необхідно запам'ятовувати свій стан в енергонезалежній пам'яті і мати незалежний годинник реального часу, для запобігання перебоїв режимів при короткочасному, або тривалому перебої джерела живлення (акумулятора, блоку живлення).

Також, для налаштування роботи, пристрій повинен вміти взаємодіяти з персональним комп'ютером (мати якийсь інтерфейс і протокол передачі даних). Пристрій повинен мати можливість для розширення каналів модуляції інтенсивності світла і числа станів, що задаються в системі, а також модернізації всього пристрою в цілому (наприклад, для управління сервоприводом або двигуном).

2.2 Модульність

Для різних експериментів над різними рослинами та мікроорганізмами не обов'язково використовувати весь спектр доступного видимого, ІЧ та УФ світла.

Набір довжин хвиль залежить від специфіки експерименту, від того, що хочуть побачити спостерігачі, підтвердити теорію або спростувати. Тому апаратно-програмний комплекс повинен бути побудований на модульній основі, щоб можна було поміняти модуль світіння на інший. Так можливо взаємозамінність [10] старих модулів новими, які можуть відрізнятися наявністю інших діапазонів довжин хвиль, енергоефективністю, потужністю випромінювання і т.д.

Також має бути можлива підтримка багатомодульності, коли нашим пристроєм можуть керуватися інтенсивність світіння декількох довжин хвиль. Один модуль може мати одну (точніше діапазон світіння, інтенсивність якого буде зосереджена на одній довжині хвилі (рис. 2.1)).

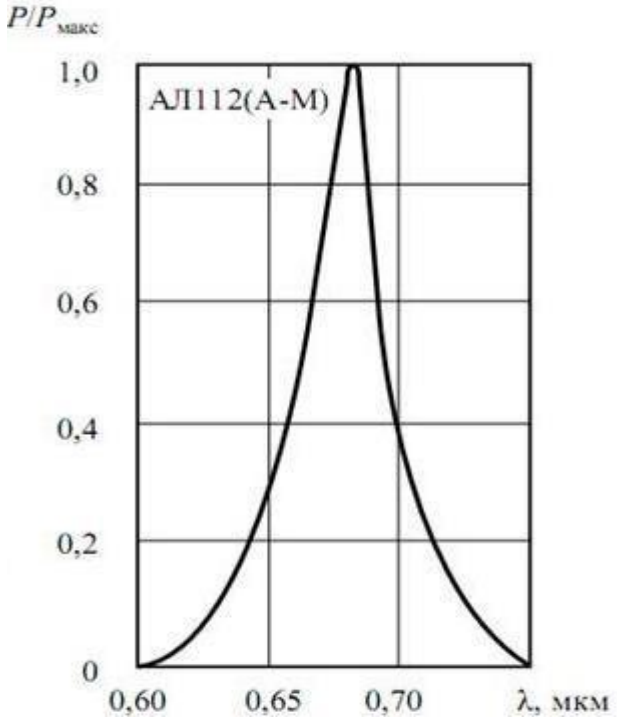


Рисунок 2.1 — Спектр випромінювання червоного світлодіода АЛ112

На рисунку 2.2 для знаходження загального спектра потрібно скласти всі інтенсивності всіх джерел світла.

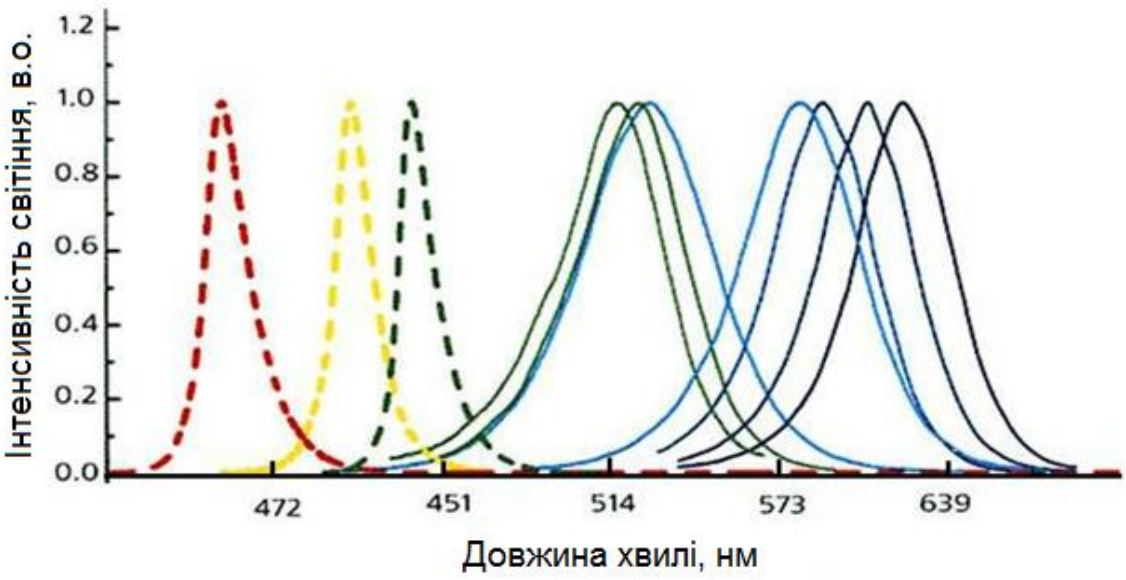


Рисунок 2.2 — Спектри випромінювання різних джерел

Модулі повинні мати окреме джерело живлення, з'єднаний з основним логічним модулем за загальним дротом (нулем), або один джерело, але сумісний за рівнями з логічним модулем.

2.3 Світлодіоди. Широтно-імпульсна модуляція

Для отримання стабільної різноманітності та регулювання довжин хвиль дуже зручно та енергоефективно використовувати світлодіоди, так, як ККД енергопередачі у них найвищий на сьогоднішній день з усіх інших доступних джерел освітлення. Зараз світлодіоди, що випускаються, покривають весь видимий спектр, а також деякі УФ та ІЧ діапозони. Отже, для складання спектральної щільності інтенсивності світіння дуже зручно використовувати саме світлодіоди.

Сумарна потужність світлового випромінювання світлодіода виявляється у люменах (лм) чи ватах (Вт). Важливо розуміти також, що яскравість світлодіода залежить від середньої величини прямого струму.

Існують два поширені способи управління яскравістю світлодіодів: широтно-імпульсна модуляція (ШІМ) і аналогове регулювання [11]. Обидва

способи зводяться, зрештою, до підтримки певного рівня середнього струму через світлодіод, або ланцюжок світлодіодів.

Для аналогового регулювання характерні зміщення спектра свічення при різному струмі, що проходить через світлодіод, тобто від струму залежить основна довжина хвилі світлодіода. Можна зробити висновок, що для нашого завдання такий спосіб регулювання не підходить.

Регулювання за допомогою ШІМ полягає в управлінні моментами включення і вимикання струму через світлодіод, повторюваними з досить високою частотою, яка, з урахуванням фізіології людського ока, не повинна бути менше 200 Гц. В іншому випадку, може проявлятися ефект мерехтіння, хоча основну функцію передачі енергії з певною довжиною хвилі все одно буде виконувати.

Разом для моделювання в нашому випадку найдоцільніше використовувати світлодіоди і регулювати інтенсивність випромінювання за допомогою ШІМ.

Також ШІМ підходить для керування сервоприводами та двигунами, що дає можливість модернізації пристрою.

Для світлодіодних модулів можна використовувати світлодіоди компанії OSRAM, розроблені спеціально для рослинного освітлення (рис. 2.3).

	Блакитний	Синій	Зелений	Жовтий	Янтарний	Червоний	Гіпер-червоний	Дальний червоний
	LB CPDP	LD CQxP	LT CPxP	LY CP7P	LA CPxP	LR CPxP	LH CPxP	GF CSxPM1
Довжина хвилі, нм	464...476	449...461	513...537	583...595	612...624	620...632	646...666	730
Кут огляду	150°	80°/150°	80°/150°	80°/150°	80°/150°	80°/150°	80°/150°	80°/150°
Rth(тип)	7 К/Вт	7 К/Вт	7 К/Вт	7 К/Вт	7 К/Вт	7 К/Вт	7 К/Вт	7 К/Вт
Макс. струм	1 А	1 А	1 А	1 А	1 А	1 А	1 А	0,7 А
Vf (тип)	3,1 В (350 мА)	3,1 В (350 мА)	3,2 В (350 мА)	2,25 В (350 мА)	2,20 В (350 мА)	2,15 В (350 мА)	2,10 В (350 мА)	1,85 В (350 мА)

Рисунок 2.3 — Лінійка світлодіодів OSLON SSL (OSRAM)

2.4 Умови експлуатації

Для нормального росту рослин необхідно підтримувати температуру +15...+24℃ [6] в залежності від типу рослини. Але для експериментів над деякими мікроорганізмами потрібний діапазон 0...+60℃ [1, 8]. З іншого боку, деяким рослинам потрібна вологість до 99%.

Отже, пристрій має працювати в діапазоні від 0 до +60℃ і відносною вологістю до 99%.

2.5 Функції програмного забезпечення

Програмне забезпечення має забезпечувати зв'язок з апаратною частиною комплексу. Повинно надавати можливість вибору кількості модулів, що підключаються, і їх тип (надалі модернізація для можливості підрахунку випромінюваної енергії у ватах для кожного модуля), кількості станів системи (спектри інтенсивності та час їх дії разом утворюють стан пристрою), графічного налаштування інтенсивності спектру та завдання часу дії стану, передачі даних пристрою в зрозумілому вигляді, тобто, дані, що передаються від комп'ютера, до пристрою повинні бути узгоджені з програмою пристрою.

Для візуальної зручності програмне забезпечення має мати шкалу прогресу передачі на пристрій.

2.6 Калібрування значень енергії, що передається, для світлодіодів

Кожна серія світлодіодів характеризується своїми характеристиками енергії, що передається. Вивчаючи різні типи світлодіодів можна помітити, що в документації на світлодіоди передану світлову енергію фіксують у різних одиницях. В основному це одиниці люмен (lm) або миліват (mW).

З енергією, що передається представленої у ватах, все досить просто, в цих одиницях якраз і зручніше розраховувати передану енергію для рослин і мікроорганізмів.

Як це не дивно, але люмен — дуже незручна одиниця для вимірювання переданої енергії. Це обумовлюється тим, що люмен прив'язаний до чутливості людського ока, тим самим має якесь значення лише у видимій області спектра. А як же ІЧ та УФ діапазони? Для переведення з люмен у ват можна скористатися функцією видності V(λ) [12], (рис 2.4), яка вказує на чутливість бездефектного людського ока.

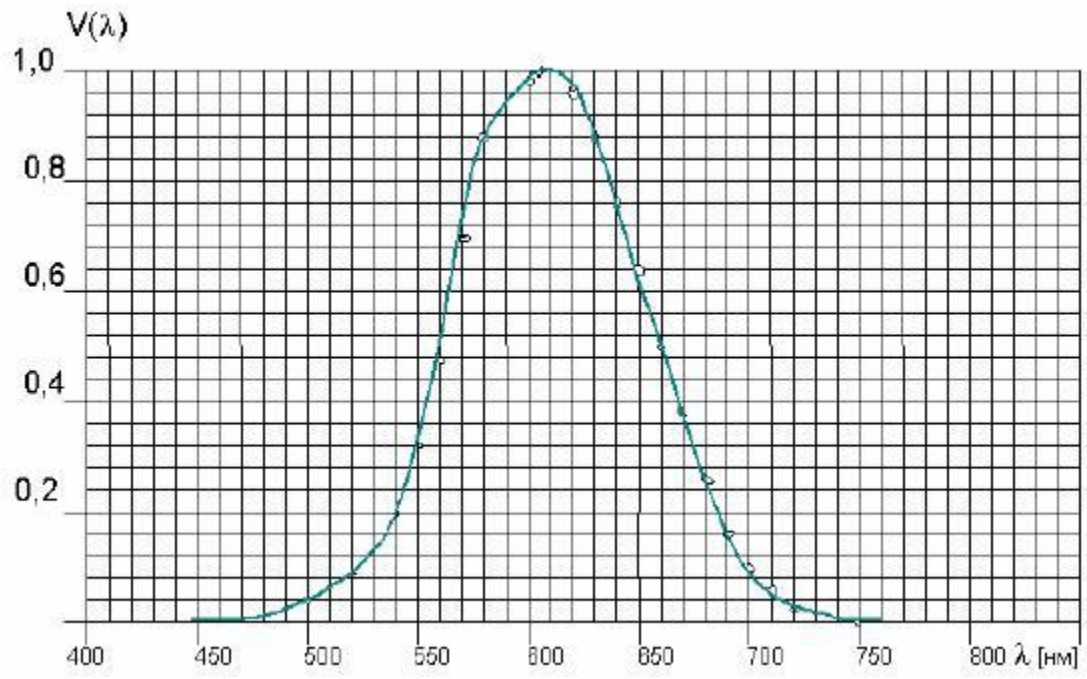


Рисунок 2.4 — Функція видимості [14]

Функція видимості показує взаємозв'язок між енергетичними та фотометричними параметрами оптичного випромінювання. Для обчислення функції для будь-якої довжини хвилі можна скористатися таблицею значень функції V(λ) (таблиця 2.1) та інтерполяційним багаточленом Лагранжа [13] (формула (1)):

$$L_n(x) = \sum_{i=0}^{n} p_{ni}(x) f_i, \tag{2.1}$$

де $L_n(x)$ —значення інтерполяційного багаточлена Лагранжа в точці x,

f_i — відоме значення функції в і-й точці,

$$p_{ni}(x) = \frac{(x - x_0)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}.$$

Таблиця 2.1 — Табличний вид спектральної світлової ефективності [14]

λ, нм	V(λ)	λ,нм	V(λ)	λ,нм	V(λ)	λ,нм	V(λ)
380	0,00004	490	0,208	590	0,757	700	0,0041
390	0,00012	500	0,323	600	0,631	710	0,0021
400	0,00040	510	0,503	610	0,503	720	0,00105
410	0,0012	520	0,710	620	0,381	730	0,00052
420	0,0040	530	0,862	630	0,265	740	0,00025
430	0,0116	540	0,954	640	0,175	750	0,00012
440	0,023	550	0,995	650	0,107	760	0,00006
450	0,038	555	1,0000	660	0,061	770	0,00003
460	0,060	560	0,995	670	0,032		
470	0,091	570	0,952	680	0,017		
480	0,139	580	0,870	690	0,0082		

Відносна функція видимості дозволяє розрахувати необхідну потужність випромінювання Pλ, відповідну світловому потоку в 1 лм, для будь-якої довжини хвилі з діапазону 400...760 нм [15]:

$$P_\lambda = \frac{A_e}{V(\lambda)}, \tag{2.2}$$

де $V(\lambda)$ — функція видності;

P_λ — потужність передана одним люмен, Вт/лм;

A_e — механічний еквівалент світла, що дорівнює 0,0016 Вт/лм [15];

Наприклад, світловому потоку в 1 лм при довжині хвилі λ=633 відповідає (λ) = 0,25. Згідно з формулою (2.2), для забезпечення такого світлового потоку необхідна потужність:

$$P_{\lambda=633} = \frac{A_e}{V(\lambda)} = \frac{0,0016}{0,25} = 6.4 \cdot 10^{-3} Вт.$$

Знаючи P_λ можна визначити потужність передану світлодіодом помноживши P_λ на кількість люмен, зазначену в документації світлодіода (формула (3)):

$$P_{світлодіода} = P_\lambda k, \tag{2.3}$$

де P_λ — потужність передана одним люмен, Вт/лм, $P_{світлодіода}$ — потужність світла передана світлодіодом, Вт, k — кількість люмен, зазначена у документації світлодіода.

Дізнавшись потужність світла, що передається одним світлодіодом, можна дізнатися про потужність світла, що передається всім модулем за формулою (2.4):

$$P_{модуля} = P_{світлодіода} m, \tag{2.4}$$

де $P_{модуля}$ — потужність передана всім модулем, Вт, m — кількість світлодіодів в одному модулі.

Але треба враховувати, що це потужність передається модулем у всіх можливих напрямах, тобто напрями визначаються кутом світіння.

3 ПРОЕКТУВАННЯ КОМП’ЮТЕРИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ОСВІТЛЕННЯМ

3.1 Формування апаратної бази та вибір інструментів розробки

Для головного обчислювального елемента мікропроцесорної системи було прийнято рішення взяти за основу мікроконтролер STM32F103 [16] виконаний у корпусі LQFP48 через його відносну дешевизну, доступність, набором відповідних вбудованих інтерфейсів передачі даних (I2C, SPI, UART) і можливостей. Має відповідний температурний діапазон роботи: -40...+85проС. Напруга живлення від 2,0 до 3,6 вольт. Частота роботи CPU 72 МГц. Споживаний струм до 50 мА. Має вбудований годинник реального часу. Пам'ять програм 64 Кбайт, пам'ять RAM 20 Кбайт.

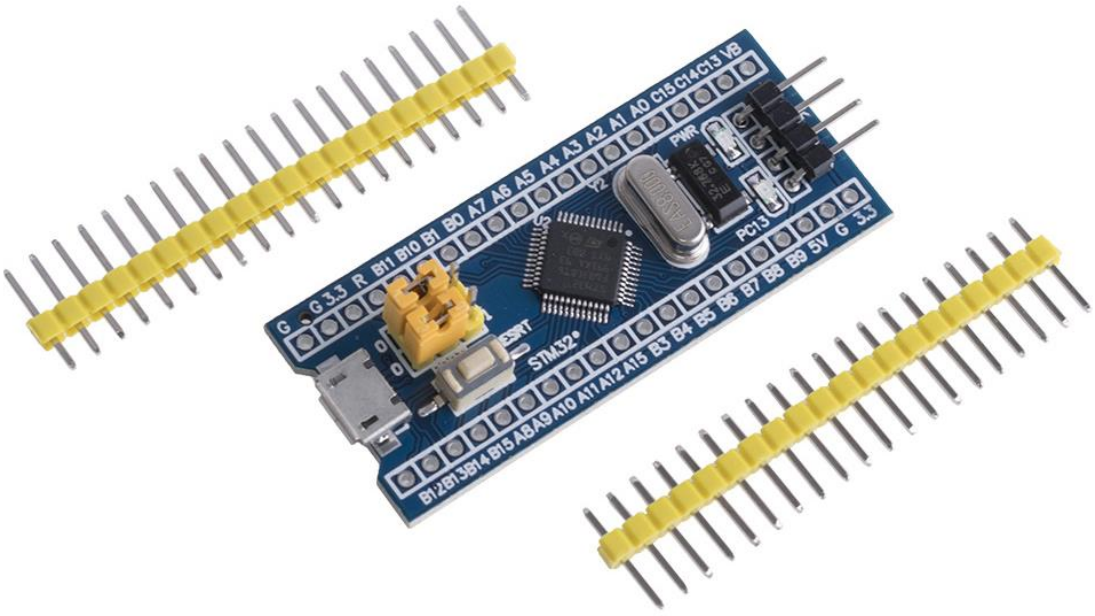


Рисунок 3.1 — Мікроконтролер STM32F103 на відлагодувальній платі

Призначення виводів відлагоджувальної плати показано на рисунку 3.2

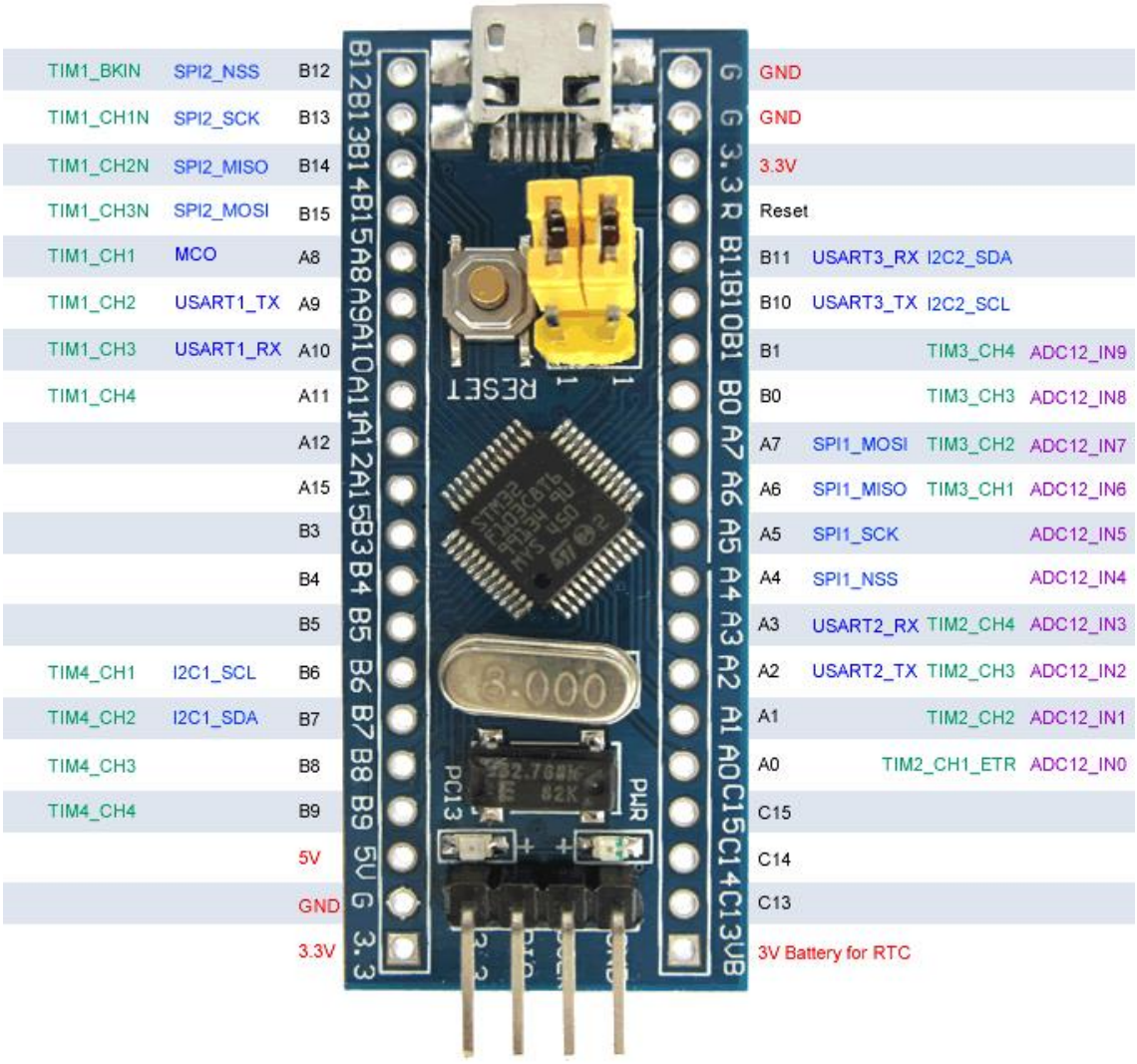


Рисунок 3.2 — Призначення виводів відлагодувальної плати

Щоб керувати інтенсивністю випромінювання світлодіодів за допомогою ШІМ, було прийнято використовувати окрему цифрову мікросхему PCA9685 [17] виконану в корпусі TSSOP28. Дана схема має шину передачі даних I2C, настроювану адресу до 64 адрес, 16 каналів ШІМ кожен з каналів налаштовується 12 біт, напругою живлення від 3 до 5 вольт, і частотою від 24 до 1526 Гц. Велика кількість адрес дозволяє модернізувати апаратно-програмний комплекс, тобто. збільшити чисельність ШИМ каналів до 1024. Має відповідний температурний діапазон роботи: -40...+85проС. Має налаштування типу зовнішнього драйвера P або N типу. Споживаний струм мікросхеми 10 мА.

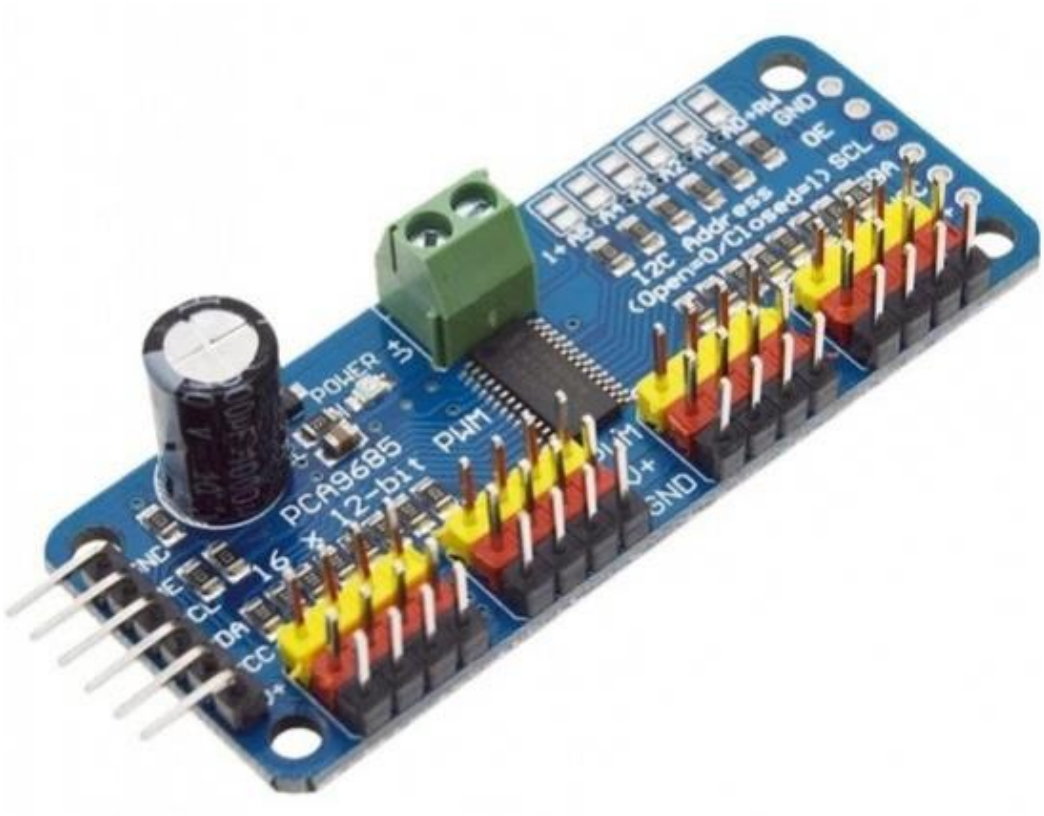


Рисунок 3.3 — Мікросхема PCA9685 на відлагоджувальній платі

Для збереження станів спектра було прийнято рішення використовувати мікросхему EEPROM M24C32WP [18] виконану в корпусі SO8. Об'єм пам'яті 4 Кбайт. Інтерфейс читання-запису даних I2C. Можливість адресації до 8 адрес. Напруга живлення від 2,5 до 5,5 вольт. До мільйона циклів записи. Має відповідний температурний діапазон роботи: -40...+85проС. Споживаний струм мікросхеми при записі до 5 мА.

Рисунок 3.4 — Мікросхема EEPROM M24C32WP

Для передачі даних через програмне забезпечення вирішено використовувати інтерфейс RS-232, так, як він простий і найпоширеніший у

виробництві. Багато ПК зараз не мають вбудованого COM-порту, а вже тим більше не мають перетворювач в рівні 3,3 вольта, що використовуються в UART STM32F103C8T6. Тому було прийнято рішення придбати готовий перетворювач USB-to-TTL PL-2303. Він дозволяє створювати віртуальний пристрій UART для логіки 3,3-5,0 вольт.

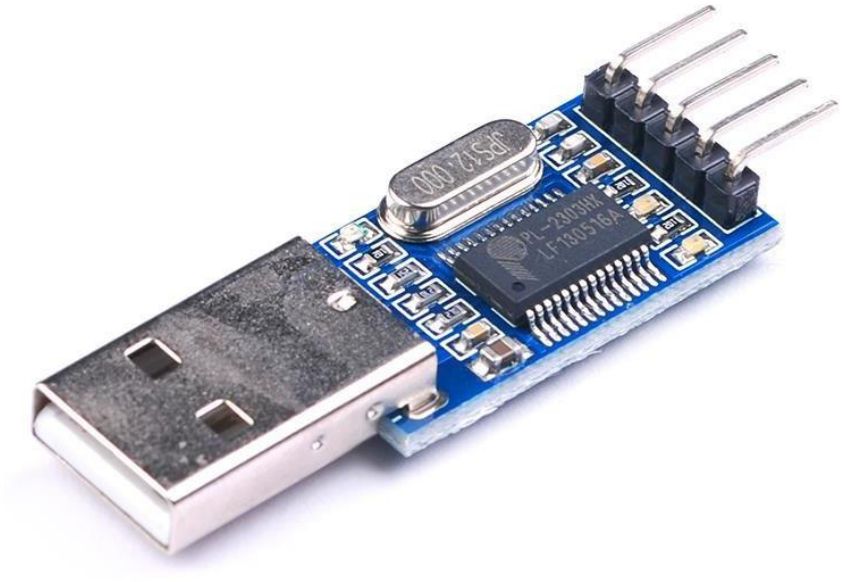


Рисунок 3.5 — Перетворювач USB-to-TTL PL-2303

Для модулів випромінювання було прийнято використовувати MOSFET транзистори IRLML2502 [19].

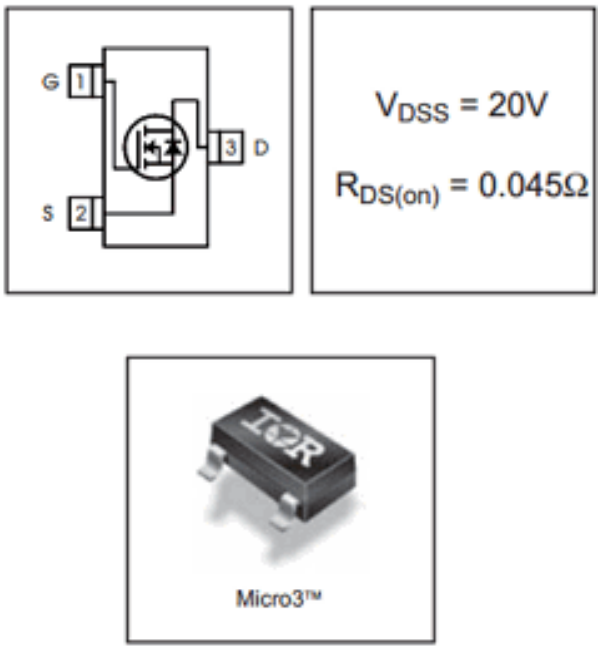


Рисунок 3.5 — MOSFET транзистор IRLML2502

Так, як вони підтримують логічну напругу на затворі, мають максимальний імпульсний струм до 4,2 А. Виконані в маленькому корпусі SOT-23, але максимально можлива напруга на витоку-стоку складає всього 20 В. Тим самим вони ставлять обмеження на використання акумуляторів та блоків живлення, напруга яких не повинна перевищувати 20 В. Мають відповідний температурний діапазон роботи: -55...+150проС. Необхідне перетворення напруги від акумуляторів та блоків живлення в 3,3 з метою забезпечення живлення мікросхем STM32F103C8T6, EEPROM M24C32WP, PCA9685. Для цього було прийнято рішення використовувати мікросхему AMS1117-3.3 [20], що перетворює вхідну напругу до 15,0 В на напругу 3,3 В, максимальний струм 1 А, температурний діапазон 0...+150проЗ, виконана в корпусі SOT-223. Тепер напруга для акумуляторів і блоків живлення не повинна перевищувати 15 В.

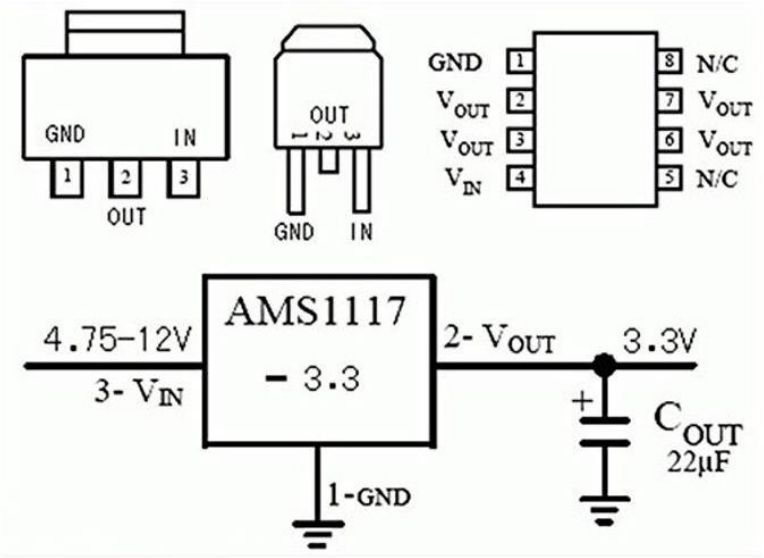


Рисунок 3.6 — Мікросхема AMS1117-3.3

Як середовище розробки (IDE) для програми на мікроконтроллер було вирішено використовувати Eclipse Oxygen.2 Release (4.7.2) [21].

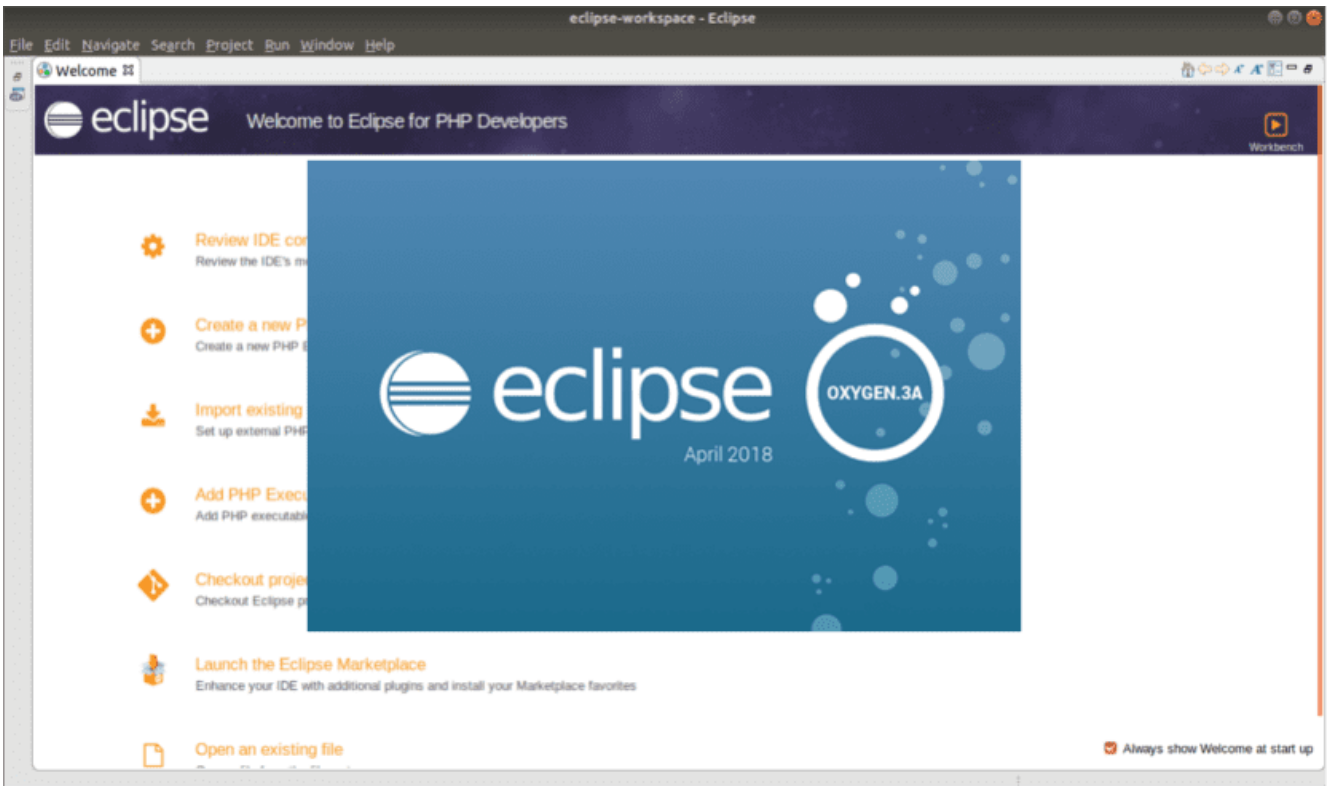


Рисунок 3.7 — Середовище розробки Eclipse Oxygen.2 Release

Як вирішальні фактори за вибір Eclipse виступними:

— підтримка компіляторів та лінковників для STM32F1XX серії;

— підтримка мови програмування C/C++ [22];

— безкоштовність IDE;

— відсутність обмежень на розмір програм у мікроконтролерах серії STM32.

Як IDE для розробки програми на ПК було вирішено використовувати Qt 5.9.1. Як вирішальні фактори за вибір Qt 5.9.1 виступили [23-25]:

— велика кількість зібраних, готових до використання бібліотек для управління периферією, в тому числі UART, потоками та ін.;

— можливість швидкого створення графічної оболонки для програм;

— просте управління подіями;

— мультиплатформенність;

— можливість швидкого розроблення програмного забезпечення;

— безкоштовність.

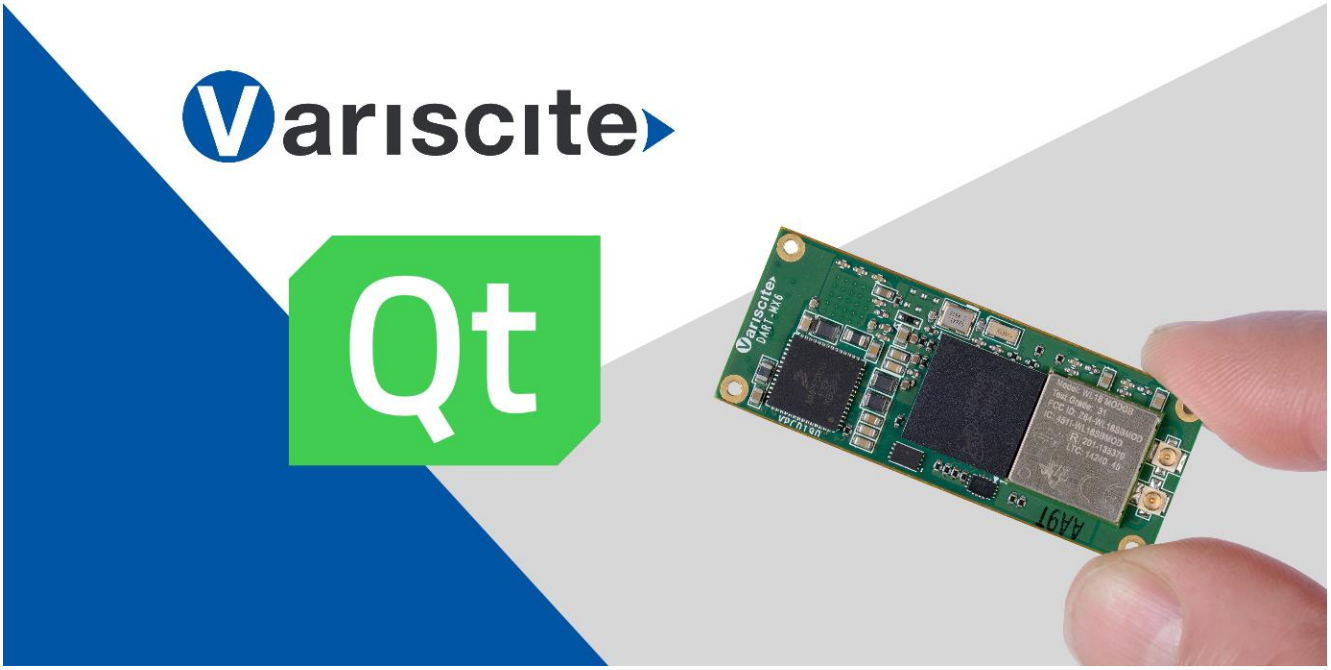


Рисунок 3.8 — Середовище розробки Qt 5.9.1

Для можливості подальшого розширення кількості модулів ШІМ та енергонезалежної пам'яті було прийнято використовувати дві різні незалежні шини I2C.

Структура даних для станів повинна містити початковий час стану, що визначається годинами реального часу, час дії режиму, значення для інтенсивності 16-ти каналів ШІМ, поточний стан і вказівник на наступний стан.

Таким чином наприклад для 10 станів необхідно енергонезалеж-ної пам'яті:1б + 10 · (1б + 4б + 4б + 64б) = 731 байт.

Під час включення мікропроцесорної системи мікроконтролер зчитує всі значення з EEPROM, визначає поточний стан і встановлює значення ШІМ. Перевіряє поточне значення реального часу із сумою часів початку стану та дії режиму, якщо поточне значення реального часу більше, то мікроконтролер змінює стан. Для цього спочатку записує значення реального часу в EEPROM для нового стану, потім встановлює нові значення ШІМ і далі перевіряє значення часів.

3.2 Розробка схеми стистеми

Для більш детального розуміння процесу було складено структурну схему пристрою (рис. 3.9).

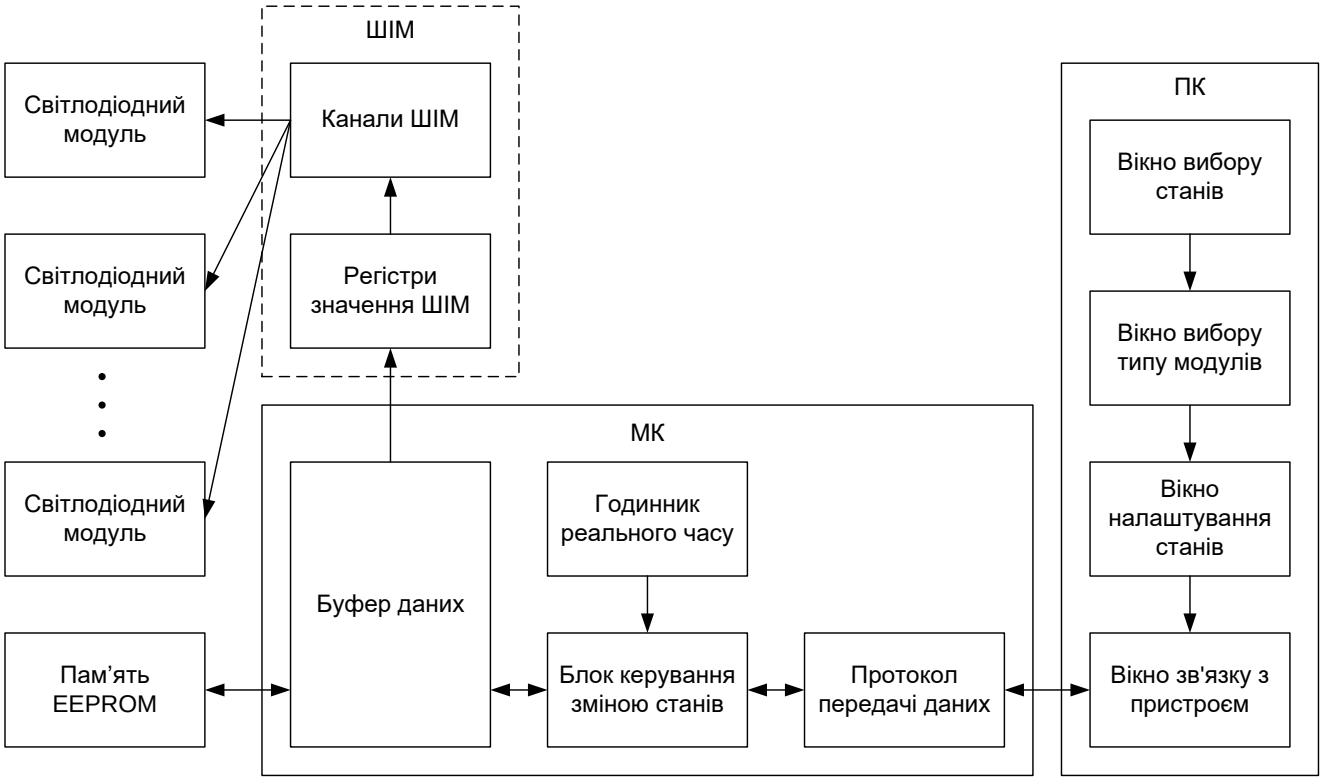


Рисунок 3.9 — Структурна схема системи

Використовуючи документацію за вибраними мікросхемами можна скласти перелік елементів і схему електричну принципову (див. додатки).

Світлодіодні модулі мають бути різні, так, як. всі світлодіоди можуть мати різні рівні напруги, потужність, прямий струм, розміри, потужність переданої енергії світлом, ТКЕ. Одним набором світлодіодів не обмежишся (не можна скласти список електронних компонентів). Зате можна скласти загальну схему електричну принципову (рис. 3.10).

На рисунку 3.10 номінали резисторів залежать від номінальної напруги світлодіодів та їх кількості на ділянці ланцюга (індекс k). Можна порахувати струм і напруги на ділянці Vcc-R1-E1.1-...-E1.k-T1 і розрахувати опір і потужність для R1, використовуючи формули (3.1-3.3) [27].

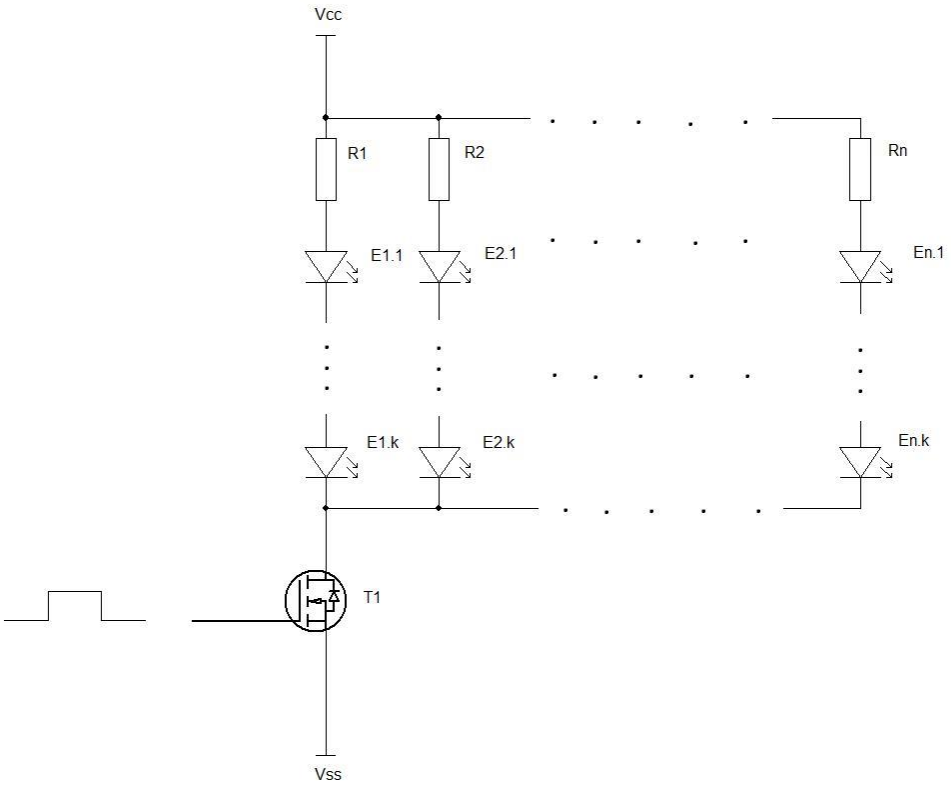


Рисунок 3.10 — Схема електрична принципова світлодіодного модуля

$$U_{R1} = V_{cc} - U_{E1.1} - U_{E1.2} - \cdots - -U_{E1.k} - -U_{T1}, \tag{3.1}$$

$$R1 = \frac{U_{R1}}{I_{Dном}}, \tag{3.2}$$

$$P_{R1} = \frac{(U_{R1})^2}{R1}, \tag{3.3}$$

де:

— $U_{E1.1} ... U_{E1.k}$ — напруги на світлодіодах, В;

— V_{cc} — напруга живлення, В;

— U_{T1} — напруга на відкритому транзисторі, В;

— $R1$ — опір на резисторі, Ом;

— U_{R1} — напруга на резисторі, В;

— $I_{Dном}$ — номінальний струм світлодіодів, А;

— P_{R1} — потужність, що розсіюється резистором R1, Вт.

Для однозначності світлодіоди $U_{E1.1}...U_{E1.k}$ повинні бути однакові, інакше вном.доведеться враховувати найменше значення характеристик всіх світлодіодів, що може спричинити низьку ефективність модуля.

3.3 Протоколи передачі даних та блок-схеми програми мікроконтролера

Виходячи з поставленого завдання, необхідно встановити протокол зв'язку між мікропроцесорною системою та ПК. Для цього було прийнято рішення розробити власний протокол передачі даних.

Протокол повинен встановлювати зв'язок з пристроєм і передавати дані за допомогою ключових слів, а також вміти підтверджувати прийняття даних.

Процес підключення та передачі даних можна представити таким чином (рис. 3.11).

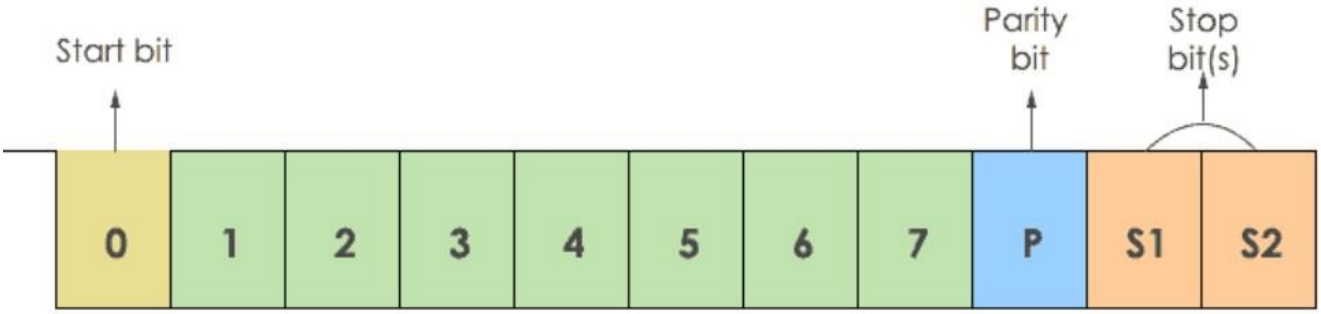


Рисунок 3.11 — Протокол передачі даних

Таким чином легко відслідковувати прогрес передачі даних, ініціювати початок і кінець передачі.

Розроблені алгоритми для мікропроцесорної системи можна і потрібно було розбити на дрібніші блоки так, як використовувалося багато апаратних переривань зі своїми пріоритетами.

Тому було вирішено розбити алгоритм на блоки:

— основна програма (передстановка);

— основна програма (основний цикл);

— ініціалізація USART1;

— обробник переривання щодо прийняття байта USART1;

— обробник переривання зі спрацьовування таймера 2;

— обробник переривання зі спрацьовування таймера 3;

— обробник переривання зі спрацьовування таймера 4;

— виведення PCA9685 із режиму сну;

— встановити режим PCA9685;

— рахувати стани з EEPROM;

— записати стани в EEPROM;

— затримки (мс).

Ініціалізовані змінні та буфери:

— Count_SOST — покажчик на номер стану прийому даних;

— Count_Next — покажчик на позицію у буфері buff_Pointer_next()()

— прийомі даних;

— Count_Second — покажчик на позицію в буфері buff_Second ()() при прийомі даних;

— Count_PCA — покажчик на позицію в буфері buff_PCA()() при прийомі даних;

— state — покажчик стан мікроконтролера;

— FlagRTC — прапор, що сигналізує перехід в інший стан ШІМ та запис значень RTC в EEPROM;

— F_state1 — прапор вчинення події у стані 1;

— F_state2 — прапор вчинення події у стані 2;

— F_state4 — прапор вчинення події у стані 4;

— IsData — змінна, що переводить стан мікроконтролера в стан 2 або 3;

— Delay_ms — змінна, що задає затримку мс;

— Complete() — буфер ключового слова “Complete”;

— Hello() — буфер ключового слова “Hello”;

— buff_PCA()() — буфер значень PCA;

— buff_RTC()() — буфер значень початку роботи режиму;

— buff_Second()() — буфер значень часу дії режиму;

— buff_Pointer_next() — буфер покажчиків на такий стан;

— Pointer_now — покажчик поточного стану;

— RTC_Counter — лічильник RTC;

— Hello_Counter — лічильник буфера Hello;

— Complete_Counter — лічильник буфера Complete.

Далі представлені блок-схеми програмних блоків: основний програми (предустановка) (рис. 3.12) та основна програма (основний цикл).

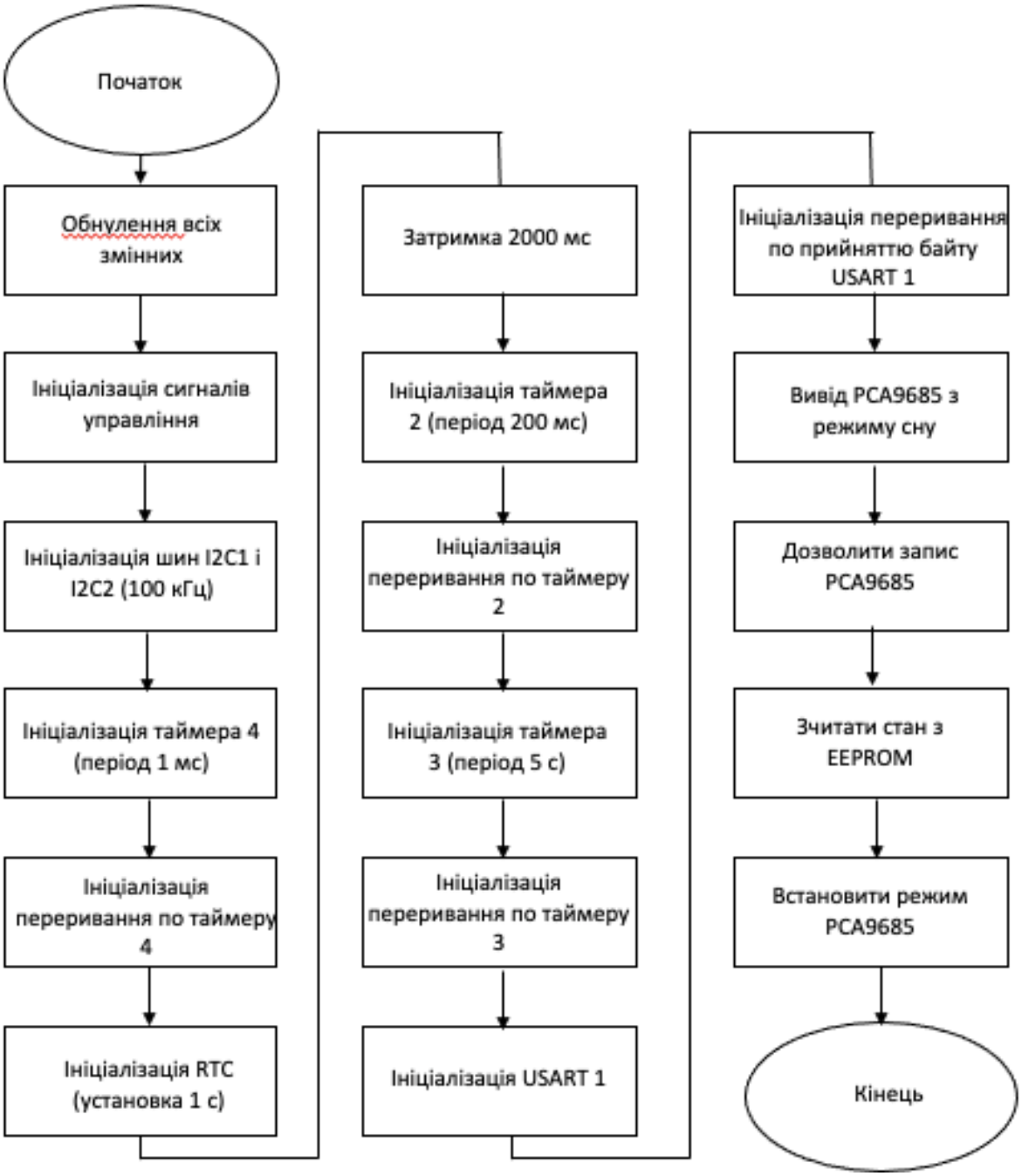


Рисунок 3.12 — Основна програма (передстановка)

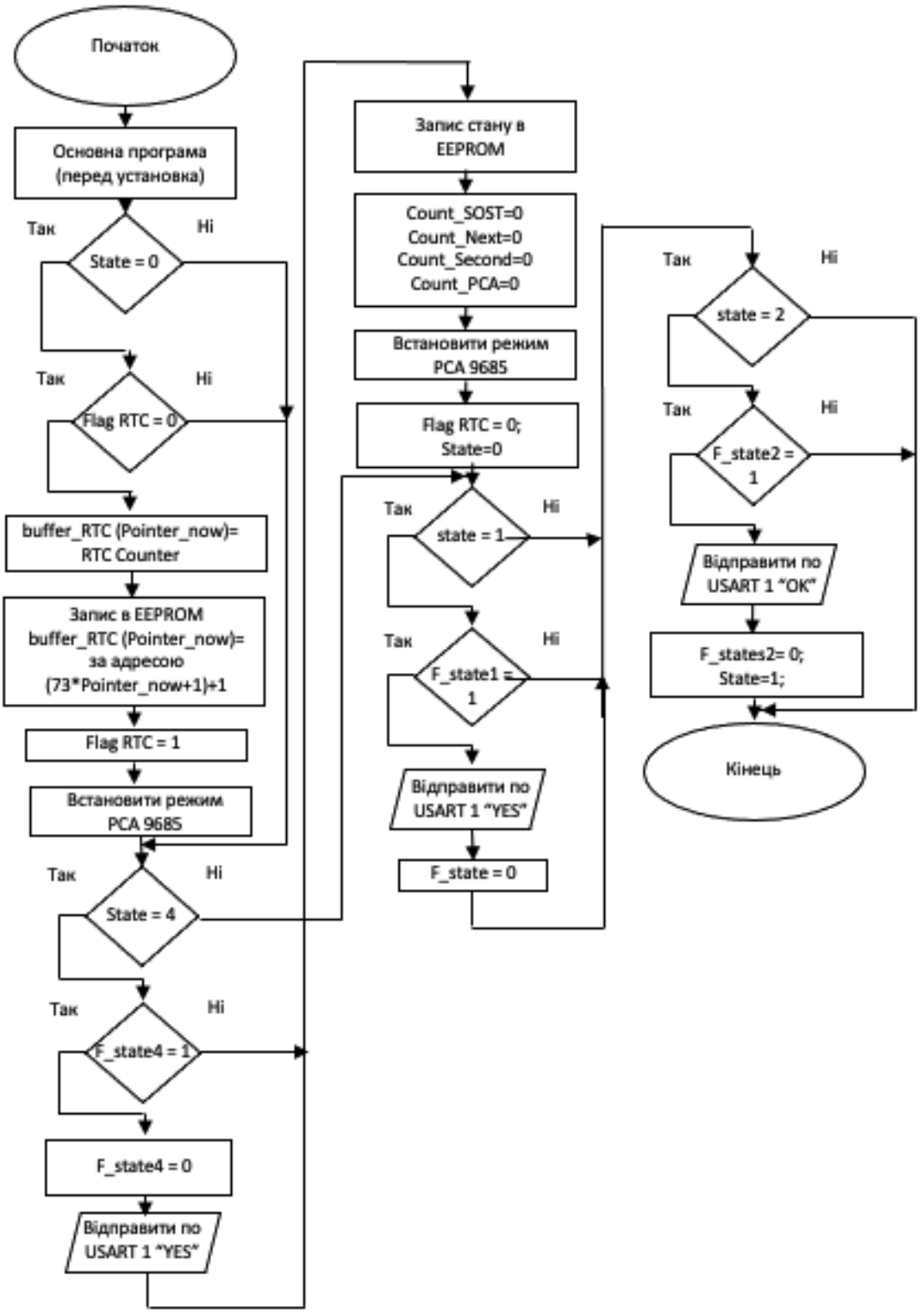


Рисунок 3.13 — Основна програма (основний цикл)

Текст програми для мікроконтролера представлений у додатку.

3.4 Програмне забезпечення система

Бібліотеки і саме середовище розробки Qt надає можливість поєднання подій різних класів проекту за допомогою політики сигнально-слотових з'єднань.

Структура сигнально-слотових з'єднань програми (рис. 3.14).

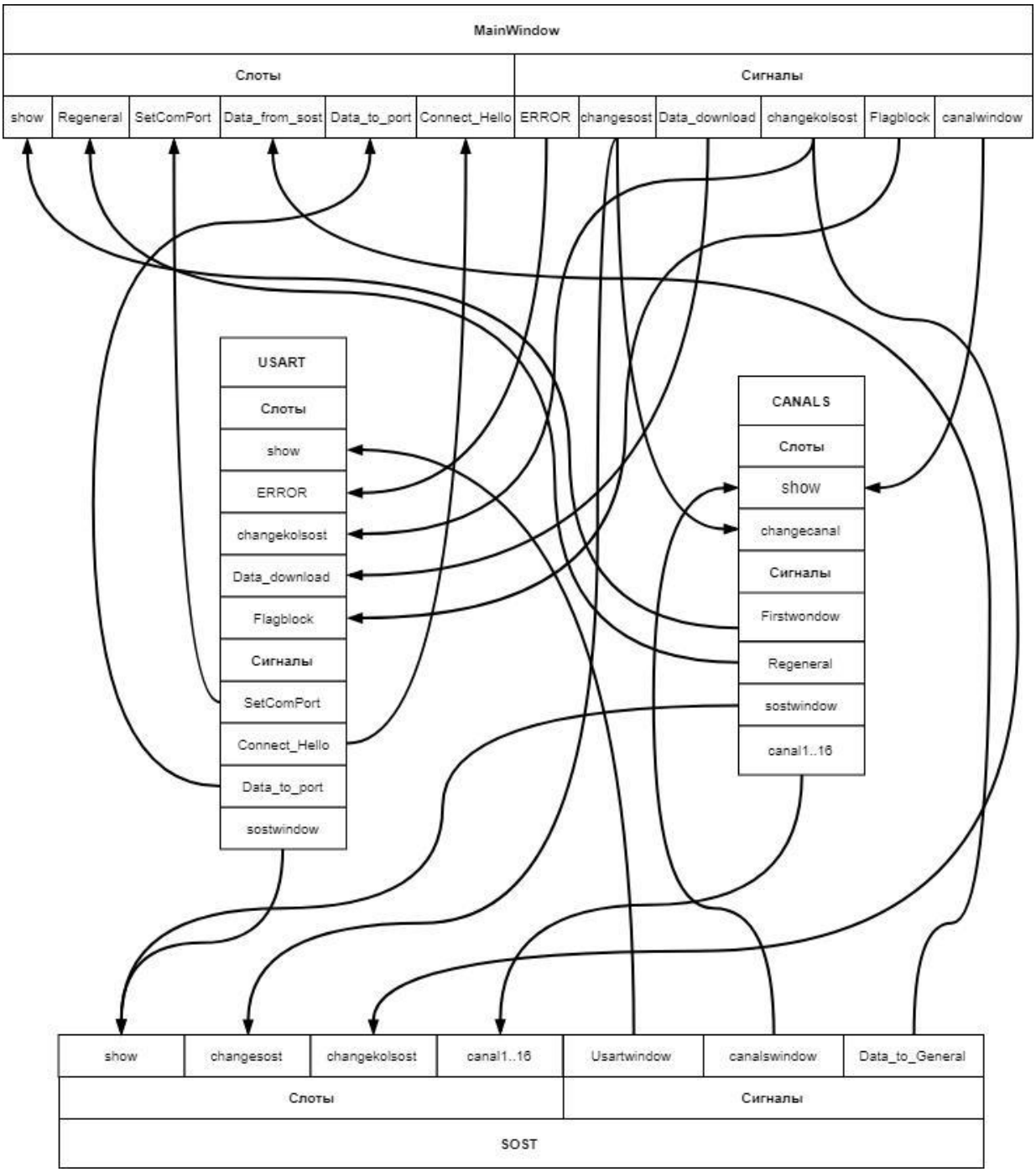


Рисунок 3.14 — Структура сигнально-слотових з'єднань програми Qt

Неважко помітити, що структура тільки сигнально-слотових з'єднань програми вийшло дуже складною, і це незважаючи на внутрішні функції та структуру програми. Стрілками показані напрямки сигналів, що активізують виконання слотів класів.

Установка станів системи складається з 4 етапів:

— початкова установка;

— вибір модуля;

— встановлення режиму;

— з'єднання з пристроєм.

Розглянемо деякі алгоритми та сигнали на прикладі використання програми, запустимо програму і побачимо наступне вікно (рис. 3.15).

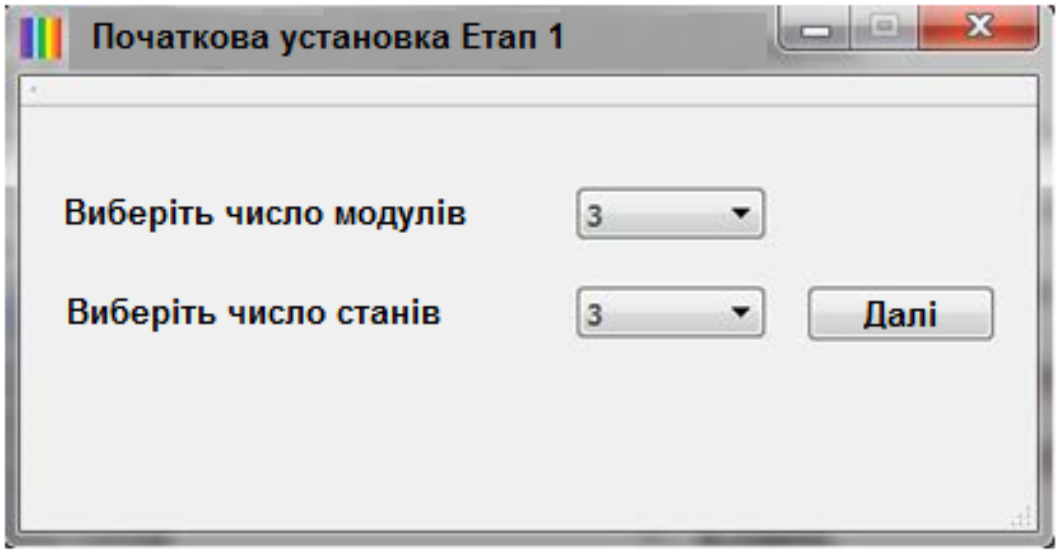


Рисунок 3.15 — Початкова установка Етап 1

На першому етапі необхідно вибрати число модулів, що беруть участь, і число станів системи. При натисканні кнопки "Далі" виконується наступний алгоритм:

1. Початок.
2. Виділити пам'ять під структуру передачі.
3. Показати вікно 2 етапу.
4. Закрити вікно.

5. Сигнал changesost(кількість вибраних модулів).
6. Сигнал changekolsost(кількість вибраних станів).
7. Кінець.

Після виконання алгоритму відкриється вікно 2-го етапу вибір модуля (рис. 3.16) і відправиться інформація про кількість обраних модулів класам CANALS та SOST та кількість станів класам SOST та USART.

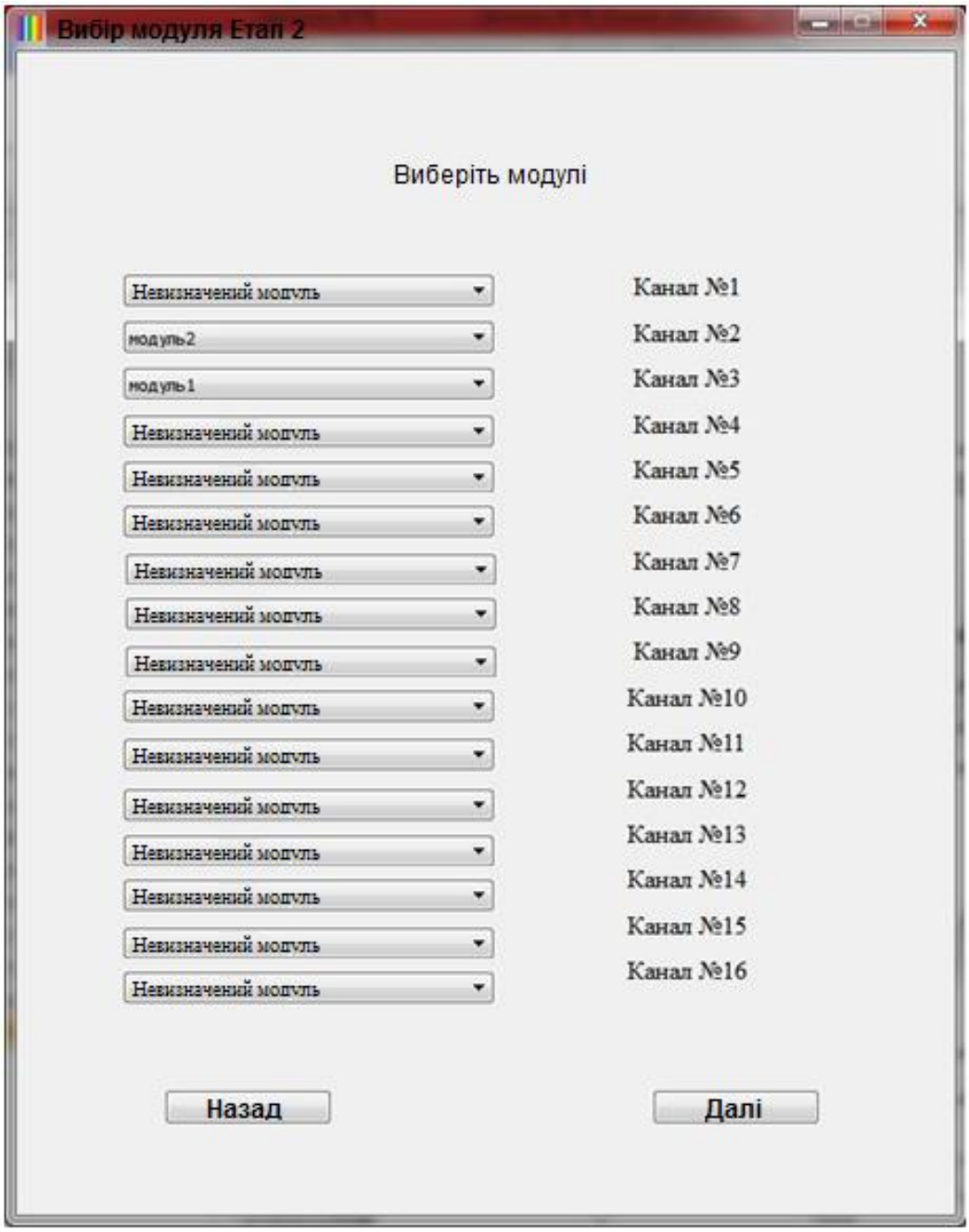


Рисунок 3.16 — Вибір модуля Етап 2

В вікні можна вибрати відкалібровані типи модулів на кожен із використовуваних каналів. При натисканні кнопки “Назад” виконується наступний алгоритм:

1. Початок.
2. Закрити вікно.
3. Сигнал Regeneral (очищає пам'ять створеної структури передачі даних, створених у першому вікні).
4. Сигнал першого вікна (показати вікно 1-го етапу).
5. Кінець.

Після виконання алгоритму відкриється перше вікно і налаштування почнеться по-новому.

При натисканні кнопки "Далі" у другому етапі виконується наступний алгоритм:

1. Початок.
2. Сигнали canal1..16 (назва модуля).
3. Закрити вікно.
4. Сигнал сигналізує (показати вікно 3-го етапу).
5. Кінець.

Після виконання алгоритму відкриється вікно 3-го етапу встановлення режиму (рис. 3.17), в якому є можливість налаштування значень даних, що завантажуються кожного з перемиканих режимів.

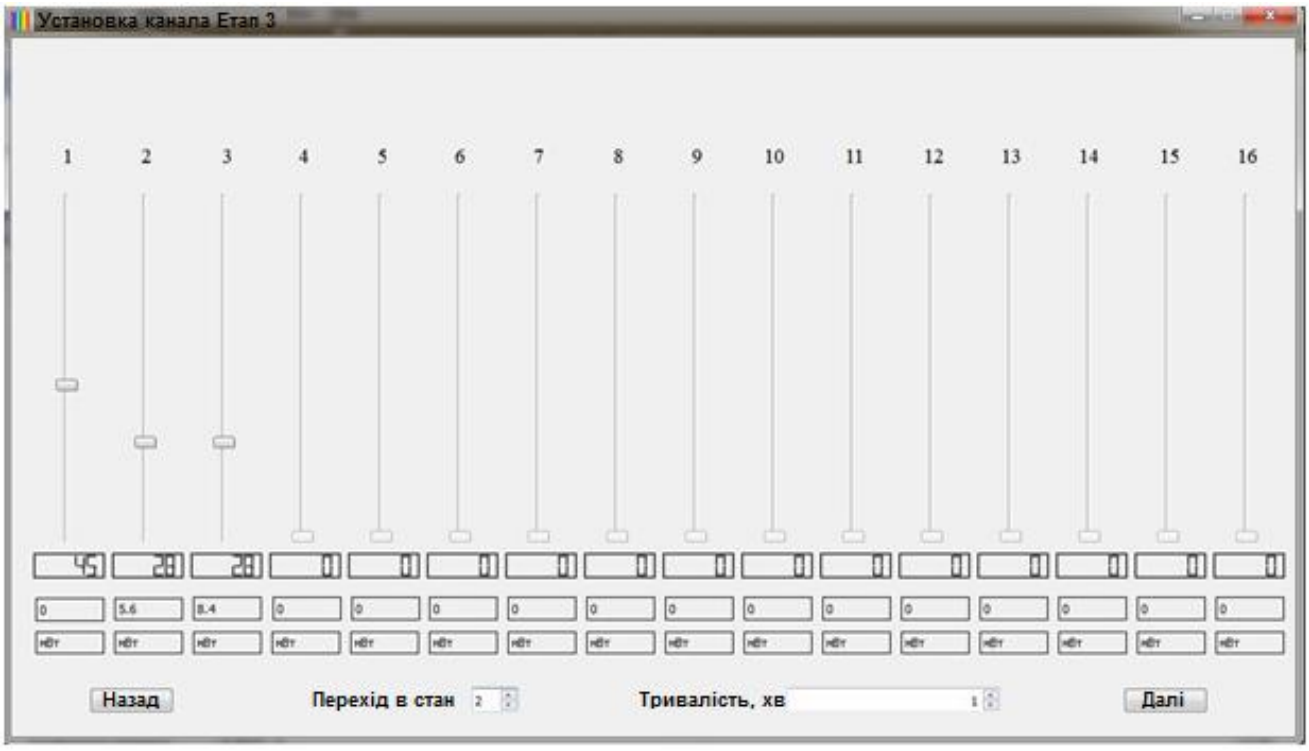


Рисунок 3.17 — Установка канала Етап 3

Після отримання сигналу про кількість каналів, кількість станів, отримання типів модулів по кожному з каналів, вікно установки режимів блокує повзунки каналів, що не використовуються, запам'ятовує кількість станів, і встановлює калібрувальний коефіцієнт передачі потужності світловим потоком кожному з каналів (калібрування не здійснено, коефіцієнти обрані випадково, щоб показати наявність можливості калібрування модулів).

На 3-му етапі є можливість складання структури передачі даних для кожного зі станів, а саме — налаштування за 100 відсотковою шкалою потужності, що передається світловим потоком діючим модулем на каналі (повзунки), встановлення покажчика на наступний стан (перехід у стан), встановлення тривалості режиму (тривалість, хв.).

При натисканні кнопки “Назад”, якщо стан, що настроюється, дорівнює 1, то закриває поточне вікно і посилає сигнал canalswindow (відкриває вікно другого етапу), інакше декрементується номер стану, що змінюється.

При натисканні кнопки “Далі”, якщо стан, що налаштовується, є останнім, то перетворює дані поточного стану на значення створеної структури переданих даних на першому етапі і записує в неї ці значення, закриває поточне вікно і посилає сигнал usartwindow (відкриває вікно четвертого етапу), інакше перетворює дані поточного стану в значення створеної структури переданих даних на першому етапі і записує в неї ці значення і інкрементує настроюваний стан. Після сигналу з використаннямwindow відкривається вікно з'єднання з пристроєм 4 етапу (рис. 3.18).

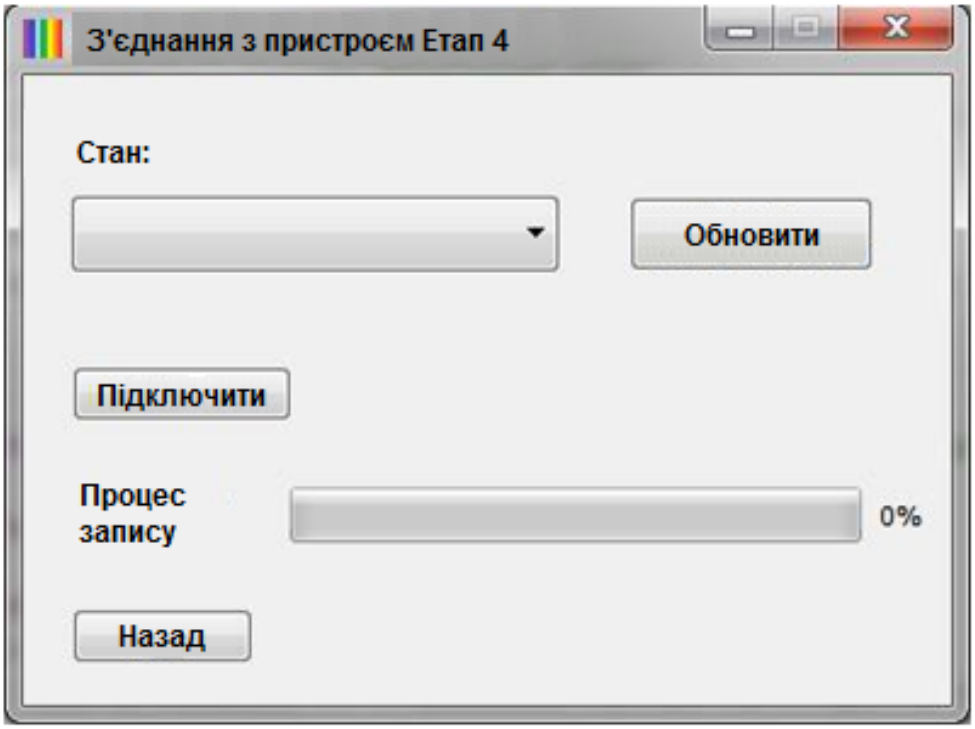


Рисунок 3.18 — З’єднання з пристроєм Етап 4

На 4 етапі організується відкриття COM-порту, підключення пристрою ключовим словом і передача даних по вище представленому протоколу. Для зручності стеження прогресом передачі було додано шкала процесу запису.

Якщо натиснути кнопку “Оновити ”, програма додає до списку (ліворуч від кнопки) імена пристроїв, що належать до COM-портів. При виборі відповідного імені пристрою підключається до нього, настроює режим передачі даних COM-порту як мікроконтролера в пункті 3.5 цієї глави. При успішному відкритті порту праворуч від слова “Стан:” буде написано “Порт підключено”.

При натисканні на кнопку "Підключити" програма надішле у відкритий COM-порт ключове слово, і якщо дочекається відповіді від мікроконтролера, то заблокує кнопку "Підключити" сигналом Flagblock, і розблокує кнопку "Встановити режим" ”.

При натисканні на кнопку “Встановити режим” програма почне передачу даних із створеної структури на першому етапі згідно з протоколом, описаним у пункті 3.4 цього розділу, заблокує кнопку

“Встановити режим” та розблокує кнопку “Підключити”. У разі успішної передачі всіх даних можна спостерігати наступне (рис. 3.19):

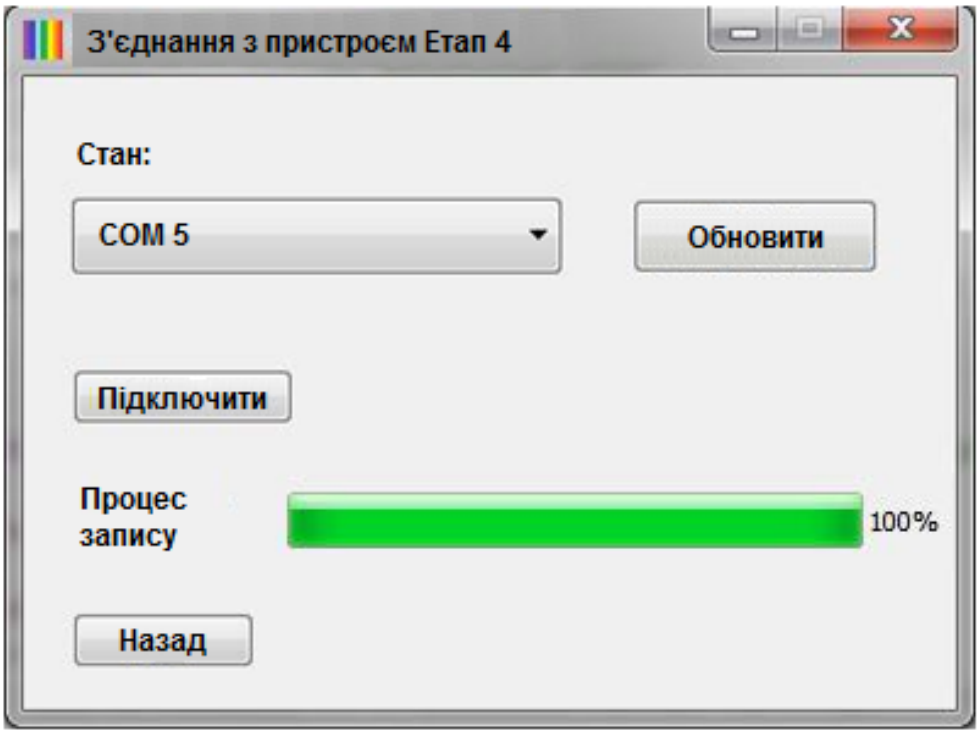


Рисунок 2.19 — Успішна передача даних на 4 етапі

Після успішної передачі даних мікроконтролер витримує паузу, це пов'язано із записом даних з буфера в EEPROM [31], а потім переключається в робочий режим.

Текст програми для ПК представлений у додатках.

ВИСНОВКИ

У процесі роботи було досліджено основні джерела випромінювання та способи управління світловим потоком. За результатами аналізу сучасних джерел випромінювання було доведено необхідність розробки комп’ютеризованої системи для управління освітленням в оранжереї.

Розроблено мікропроцесорну систему на основі мікроконтролера STM32F103C8T6, цифрових мікросхем M24C32WP, PCA968, аналогової мікросхеми AMS1117-3.3, та готового віртуального пристрою COM-port USB-to-TTL PL-2303. Максимальний струм керуючого модуля становив 65 мА, напруга живлення модуля керуючого від 6 до 15 В.

Розроблено алгоритми для мікропроцесорної системи та програмного забезпечення комп’ютеризованосистема для управління освітленням. Підібрано необхідний перелік електронних компонентів.

Розроблено та перевірено на працездатність програми на мікроконтролер серії STM32F1XX (IDE Eclipse Oxygen 2018.1) та на ПК (IDE Qt 5.9.1.).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. С.А. Верстато. Питання фотоенергетики рослин на півночі // Автореферат дисертації на здобуття наукового ступеня кандидата біологічних наук. - Москва, 1963.

2. Успіхи сучасної біології: Навчальний посібник /М. І. Сисоєва, Є. Ф. Марковська – 2008, том 128, № 6, с. 580-591

7. Пухова Н.Ю. Екологічна фізіологія мікроорганізмів. Ч. 2. Аутекологія мікроорганізмів: навчальний посібник / Н.Ю. Пухова; Яросл. держ. ун-т. - Ярославль: ЯрГУ, 2006. - 128 с.

8. Філіна Н.Ю., Верховцева Н.В. Екологічна фізіологія мікроорганізмів. Ч. 1. Фізіологія мікроорганізмів: Навч. посібник/Яросл. держ. ун-т.- Ярославль.: 2001. - 92 с.

3. Видимі спектри, ультрафіолетове випромінювання та їх вплив на рослини. - [Електронний ресурс]. - Електрон. текст. - Режим доступу:

http://pryamie-ruki.su/vidimye-spektry-ultrafioletovoe-izluchenie-i-ix-vozdejstvie-na-rasteniya/, вільний. - Яз. рус.

9. Мікробіологія вдома. - [Електронний ресурс]. - Електрон. текст. - Режим доступу: https://probakterii.ru/prokaryotes/raznoe/vyrashhivanie-bakterij.html.

10. Дімов Ю. В. Метрологія, стандартизація та сертифікація: підручник –2-ге вид. - СПб.: Пітер, 2005. - 432 с.

11.Способи управління яскравістю світіння світлодіодів за допомогою імпульсних драйверів // Журнал Радіолоцман.- 2011.

13. Волков Є. А. Численні методи: Навч. Посібник для вузів. - 2-ге вид.,іспр. - М: Наука. Гол. ред. фіз.-мат. літ., 1987. - 248 с.

27. Угрюмов Є. П. Цифрова схемотехніка: Навч. посібник для вузів. - 2-ге вид., Перероб. та дод. - СПб.: БХВ-Петербург, 2005. - 800 с.: іл.

28. Преснухін Л.М., Шахнов В. А. Конструювання електронних вирахувань машин і систем. Навч. Для втузів за спец. «ЕОМ» та «Констру-ювання та виробництво ЕВА». - М.: Вищ. шк., 1986 512 с.: іл.

2. Сила світла. Одиниці сили світла та світлового потоку. - [Електронний ресурс]. - Електрон. текст.- Режим доступу: http://lib.sernam.ru/book_t_phis.php?id=307, вільний. - Яз. рус.

4. Розсада рослин: світло та спектр. - [Електронний ресурс]. - Електрон. текст. - Режим доступу: http://my-3sotki.ru/?p=2042, вільний. - Яз. рус.

5. Спектри світла для зростання рослин. - [Електронний ресурс]. - Електрон. текст. – Режим доступу: http://led-com.ru/info/articles/osveshchenie-rasteniy-fitosvet/spektry-sveta-dlya-rosta-rasteniy/, вільний. - Яз. рус.

6. Освітлення рослин білими світлодіодами. - [Електронний ресурс]. - Електрон. текст. – Режим доступу: https://geektimes.com/post/293045/ , свободний. - Яз. рус.

14. Люмени, кандели, вати та фотони. - [Електронний ресурс]. - Електрон. текст. – Режим доступу: http://videoscan.ru/page/714 2 , вільний. - Загл. з екрану. - Яз. рус.

15. Функція видимості та її залежність від довжини електромагнітної волні. - [Електронний ресурс]. - Електрон. - Режим доступу: https://studfiles.net/preview/5787762/page:7/, вільний. - Яз. рус.

16. Medium-density performance line ARM®-based 32-bit MCU з 64 або 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. Interfaces. - [Електронний ресурс]. -Електрон. текст.- Режим доступу: http://www.st.com/content/ccc/resource/technical/document/datasheet/33/d4/6f/1d/ df/0b/4c/6d/CD00161566 /files/

17. PCA9685 Datasheet - [Електронний ресурс]. - Електрон. текст. – Режим доступу: https://www.nxp.com/docs/en/data-sheet/PCA9685, вільний. -Яз.анг.

18. M24C32WBN3G Datasheet. - [Електронний ресурс]. - Електрон. текст.- Режим доступу: http://www.alldatasheet.net/datasheet-pdf/pdf/246410/STMICROELECTRONICS/M24C32WBN3G.html, вільний. - Яз. англ.

19. IRLML2502 HEXFET Power MOSFET.- [Електронний ресурс]. - Електрон. текст. – Режим доступу: https://www.infineon.com

20. AMS1117 800mA LOW DROPOUT VOLTAGE REGULATOR. - [Електронний ресурс]. - Електрон. текст. - Режим доступу: https://lib.chipdip.ru

21.Eclipse швидкий старт. - [Електронний ресурс]. - Електрон. текст. -

Режим доступу: http://we.easyelectronics.ru/STM32/stm32cubemx-start-code-eclipse-bystryy-start-otladka-v-eclipse-cherez-st-link-discovery.html , вільний.

22. Програмування STM32F103. - [Електронний ресурс]. - Електрон.текст. – Режим доступу: http://www.avislab.com/blog/stm32_st_link_ru/ , вільний. - Яз. рус.

23. Qt Documentation [Електронний ресурс] - Режим доступу:

http://doc.qt.io/qt-5/index, вільний. - Загл. з екрану. - Яз. русявий, англ.

24. QThread+QSerialPort. - [Електронний ресурс]. - Електрон. текст. - Режим доступу: http://we.easyelectronics.ru/electro-and-pc/qthread-qserialport-krutim-v-otdelnom-potoke-rabotu-s-som-portom.html, вільний. - Яз. рус.

25. Запуск Qt додатків .exe поза Qt Creator.- [Електронний ресурс]. - Електрон. текст. – Режим доступу: http://blog.harrix.org/article/1015

26. Професійна робота у системі DesignSpark PCB.- [Електронний ресурс]. - Електрон. текст. - Режим доступу: http://www.yeint.ru

29. STM32 з нуля. таймери. - [Електронний ресурс]. - Електрон. текст. - Режим доступу: http://microtechnics.ru/stm32-uchebnyj-kurs-tajmery/, вільний.

30. Управління текстовими командами (USART на STM32). - [Електронний ресурс].- Електрон. текст.- Режим доступу:http://cxem.net/mc/mc401.php, вільний.

31. STM32 I2C EEPROM 24CXX. - [Електронний ресурс]. - Електрон.текст. – Режим доступу: http://we.easyelectronics.ru/blog/STM32/998.html

ДОДАТОК А

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ ВНТУ

д.т.н., проф.

_____О. Д. Азаров

“__” _____2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

«Комп’ютеризована система для управління освітленням в оранжереї»
08-23.БДР.011.00.000 ТЗ

Науковий керівник к.т.н., доц. каф. ОТ

_____ Колесник І.С.

Студент групи 1КІ-18б

_____ Храпко Я.О.

Вінниця 2022

1. Найменування та область застосування

Комп’ютеризована система для управління освітленням в оранжереї . Системи із застосуванням сучасних досягнень в мікропроцесорній техніці для автоматизації та для управління освітленням в оранжереї.

2. Основи для розробки

Потреба в управлінні, контролі, автоматизації освітленням в оранжереї і.

3. Мета та призначення розробки

Комп’ютеризована система для управління освітленням в оранжереї, , яка надасть користувачеві нові можливості регулювання росту рослин з урахуванням явища фотоморфогенезу.

4. Етапи БДР та очікувані результати

Робота виконується в п’ять етапів, що наведені в таблиці 4.1.

Таблиця 4.1 – Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	14.02.20	14.02.20	Вступ
2	Ввлив світла на рослини і мікроорганізми	15.02.20	28.02.20	Розділ 1
3	Мікропроцесорна система та програмне забезпечення комп'ютеризованої системи управління освітленням	01.03.20	14.03.20	Розділ 2
4	Проектування компютеризованої системи для управління освітленням	15.03.20	28.4.20	Розділ 3
5	Оформлення пояснювальної записки	29.04.20	17.05.20	ПЗ,презентація

5. Матеріали, що подаються до захисту БДР

Пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відзив наукового керівника, рецензія опонента, протоколи складання державних екзаменів, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

6. Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

7. Вимоги до оформлення БДР

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

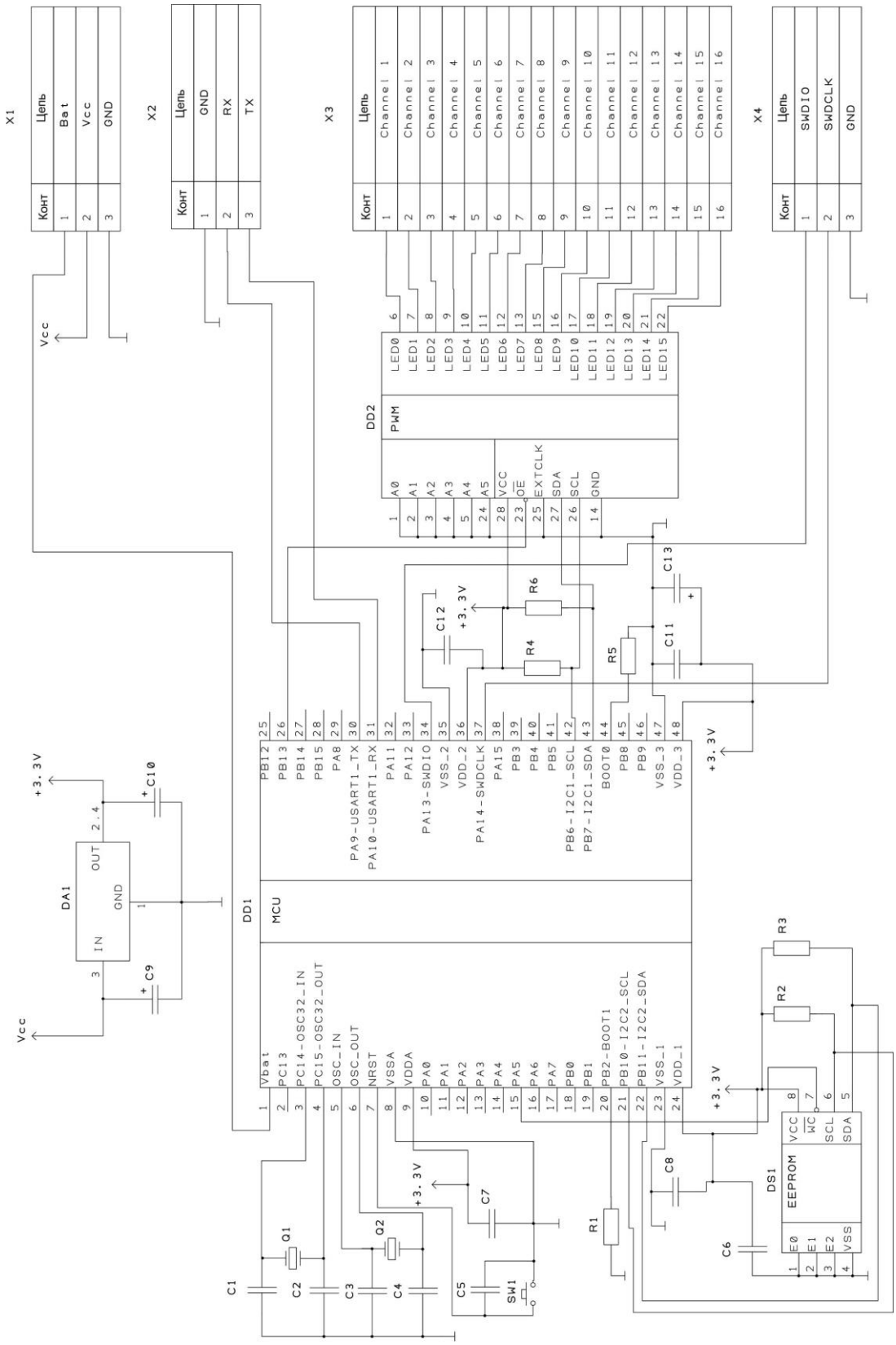
— ГОСТ 2.104-2006 «Единa система конструкторської документації. Основні написи»;

— Документами на які посилаються у вище вказаних.

Технічне завдання до виконання отримав _____ Храпко Я.О.

ДОДАТОК Б

Схема електрична принципова


X1
Цепь
Bat
Vcc
GND
Конт
X2
Цепь
GND
RX
TX
Конт
X3
Цепь
Channel 1
Channel 2
Channel 3
Channel 4
Channel 5
Channel 6
Channel 7
Channel 8
Channel 9
Channel 10
Channel 11
Channel 12
Channel 13
Channel 14
Channel 15
Channel 16
Конт
X4
Цепь
SWDIO
SWDCLK
GND
Конт
DD2
PWM
LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
LED9
LED10
LED11
LED12
LED13
LED14
LED15
A0
A1
A2
A3
A4
A5
VCC
OE
EXTCLK
SDA
SCL
GND
DD1
MCU
PB12
PB13
PB14
PB15
PA8
PA9-USART1_TX
PA10-USART1_RX
PA11
PA12
PA13-SWDIO
VSS_2
VDD_2
PA14-SWDCLK
PA15
PB3
PB4
PB5
PB6-I2C1_SCL
PB7-I2C1_SDA
BOOT0
PB8
PB9
VSS_3
VDD_3
Vbat
PC13
PC14-OSC32_IN
PC15-OSC32_OUT
OSC_IN
OSC_OUT
NRST
VSSA
VDDA
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2-BOOT1
PB10-I2C2_SCL
PB11-I2C2_SDA
VSS_1
VDD_1
DA1
IN
GND
OUT
2.4
DS1
EEPROM
E0
E1
E2
VSS
SDA
SCL
WC
VCC
Vcc
+3.3V
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
R1
R2
R3
R4
R5
R6
Q1
Q2
SW1

ДОДАТОК В

Лістинг файлу mainwindow.cpp

```
#include"mainwindow.h"
#include"ui_mainwindow.h"
#include<QStatusBar>
#include<QUrl>
#include<QByteArray>
#include<QFile>

QThread*mythread;
enumMyStates{
NoConnection,
PortConnection,
Hello,
DataOutPort,
Complete,
WRStatus,
};
MyStatesstates=MyStates::NoConnection;
MainWindow::MainWindow(QWidget*parent) :
QMainWindow(parent),
ui(new Ui::MainWindow)
{
ui->setupUi(this);

Canal=new canals();// виділяємо пам'ять під вікна
SST=new sost();
US=new usat1();
PointerSost=0;
connect(Canal, &canals::firstWindow,this, &MainWindow::show);// перехід від вікна вибору модулів у головне перше вікно
connect(Canal, &canals::ReGeneral,this, &MainWindow::ReGeneral);// кон-нект Звільнення пам'яті під General щоб ще раз не виділяти пам'ять почнемо все заново
connect(Canal, &canals::sostWindow,SST, &sost::show);// перехід від вікна вибору модулів у вікно станів ДАЛІ
connect(SST, &sost::canalsWindow,Canal, &canals::show);// перехід від вікна станів у вікно вибору модулів НАЗАД
connect(SST, &sost::usartWindow,US, &usat1::show);// перехід від вікна станів у вікно запису ДАЛІ
connect(US, &usat1::sostWindow,SST, &sost::show);// перехід від вікна записи у вікно станів НАЗАД
connect(US,SIGNAL(SetComPort (QString)),this,
SLOT(SetComPort (QString)));// передача даних про кому порт головного вікна
connect(this,SIGNAL(ERROR(QString)),US,SLOT(ERROR(QString)));// передача рядка стану прошивки
connect(this,SIGNAL(changesost(int)),Canal,
SLOT(changecanals(int)));// блокуємо керування для вибірки модулів
connect(this,SIGNAL(changesost(int)),SST,
SLOT(changecanals(int)));//передаємо вікну станів кількість модулів
connect(this,SIGNAL(changekolsost(int)),SST,
SLOT(changekolvosost(int)));// передаємо вікну станів скільки буде ітера-цій станів
connect(this,SIGNAL(changekolsost(int)),US,
```

```
SLOT(changekolvosost(int)));// передаємо вікну завантаження скільки буде
ітерацій
станів
connect(this,SIGNAL(Data_download(int)),US,
SLOT(Data_download(int)));//передаємо дані про прогрес пристрою для за-
навантаження даних
connect(this,SIGNAL(Flag_block(bool)),US,SLOT(Flag_block(bool)));
```

```
connect(this,SIGNAL(Data_download_EEPROM(int)),US,
SLOT(Data_download_EEPROM(int)));//передаємо дані про прогрес пристрою
для запису даних

connect(SST,SIG-
NAL(DataToGeneral(int,QByteArray,QByteArray,QByteArray)),this,
SLOT(DataFromSost(int,QByteArray,QByteArray,QByteArray)));// заповнення
структури General з вікна станів у перше вікно


connect(US,&usat1::ConnectHello,this,&MainWindow::ConnectHello);//кнопочка
підключити до підключення встановлює з'єднання за протоколом, переводить СТМ у
стан прослуховування передачі даних
connect(US,&usat1::Data_to_port,this,&MainWindow:: Data_To_port);// кнопка
встановити режим запускає передачу даних

connect(Canal,SIGNAL(canal1(QString)),SST,SLOT(canal1(QString)));
connect(Canal,SIGNAL(canal2(QString)),SST,SLOT(canal2(QString)));
connect(Canal,SIGNAL(canal3(QString)),SST,SLOT(canal3(QString)));
connect(Canal,SIGNAL(canal4(QString)),SST,SLOT(canal4(QString)));
connect(Canal,SIGNAL(canal5(QString)),SST,SLOT(canal5(QString)));
connect(Canal,SIGNAL(canal6(QString)),SST,SLOT(canal6(QString)));
connect(Canal,SIGNAL(canal7(QString)),SST,SLOT(canal7(QString)));
connect(Canal,SIGNAL(canal8(QString)),SST,SLOT(canal8(QString)));
connect(Canal,SIGNAL(canal9(QString)),SST,SLOT(canal9(QString)));
connect(Canal,SIGNAL(canal10(QString)),SST,SLOT(canal10(QString)));
connect(Canal,SIGNAL(canal11(QString)),SST,SLOT(canal11(QString)));
connect(Canal,SIGNAL(canal12(QString)),SST,SLOT(canal12(QString)));
connect(Canal,SIGNAL(canal13(QString)),SST,SLOT(canal13(QString)));
connect(Canal,SIGNAL(canal14(QString)),SST,SLOT(canal14(QString)));
connect(Canal,SIGNAL(canal15(QString)),SST,SLOT(canal15(QString)));
connect(Canal,SIGNAL(canal16(QString)),SST,SLOT(canal16(QString)));
//////////Бокс вибору кількості модулівui-
>comboBox->addItem(QString("1"));
ui->comboBox->addItem(QString("2"));
ui->comboBox->addItem(QString("3"));
ui->comboBox->addItem(QString("4"));
ui->comboBox->addItem(QString("5"));
ui->comboBox->addItem(QString("6"));
ui->comboBox->addItem(QString("7"));
ui->comboBox->addItem(QString("8"));
ui->comboBox->addItem(QString("9"));
ui->comboBox->addItem(QString("10"));
ui->comboBox->addItem(QString("11"));
ui->comboBox->addItem(QString("12"));
ui->comboBox->addItem(QString("13"));
ui->comboBox->addItem(QString("14"));
ui->comboBox->addItem(QString("15"));
ui->comboBox->addItem(QString("16"));
//////////Бокс вибору кількості станівui-
>comboBox_2->addItem(QString("1"));
ui->comboBox_2->addItem(QString("2"));
ui->comboBox_2->addItem(QString("3"));
ui->comboBox_2->addItem(QString("4"));
ui->comboBox_2->addItem(QString("5"));
ui->comboBox_2->addItem(QString("6"));
ui->comboBox_2->addItem(QString("7"));
ui->comboBox_2->addItem(QString("8"));
ui->comboBox_2->addItem(QString("9"));
ui->comboBox_2->addItem(QString("10"));


// mylport->setPortName();
```

```
}
MainWindow::~MainWindow()
{
deleteui;
}


voidMainWindow::ReGeneral()
{

deleteGeneral;

}



voidMainWindow::on_pushButton_clicked()//Далі
{
kolvoMod=ui->comboBox-
>currentIndex()+1;//скількиkolvoSost=ui->comboBox_2-
>currentIndex()+1;General=new myinf[kolvoSost];
Canal->show(); // Показуємо друге вікно this-
>close(); // Закриваємо основне вікно

emitchangesost(kolvoMod);// передаємо кількість модулів
emitchangekolsost(kolvoSost);// передаємо кількість станів
}
voidMainWindow::DataFromSost(intpoint,QByteArray Rezim,QByteArray
TimeToDeath,QByteArrayNextSost)// на передачу даних від 3 вікна станів{
PointerSost=point-1;// щоб записувати з 0-го стану забираємо 1
General[PointerSost].Rezim=Rezim;
General[PointerSost].TimeToDeath= TimeToDeath;
General[PointerSost].NextSost=NextSost;
}

voidMainWindow::SetComPort(QString name)//ініціалізуємо та налаштовуємо
usat
{
if(states==MyStates::PortConnection)
{

          // mylport-
           >close();mylport-
           >deleteLater();
           QThread::msleep(100);
           mythread->exit(0);
mythread->msleep(100);
mythread->deleteLater();


QThread::msleep(100);/ / Чекаємо на закриття
mythread =new QThread();//Виділяємо пам'ять mylport=
new QSerialPort();mylport->setPortName(name);
}
if(states==MyStates::NoConnection)
{
mythread =new QThread();
mylport= new QSerialPort();
mylport->setPortName(name);
mylport->setReadBufferSize(0);
}
```

```
if( mylport->open(QIODevice::ReadWrite))
{
if(!mylport->setBaudRate(QSerialPort::Baud9600))
{
emitERROR(QString("Помилка відкриття порту"));return;
}

if(!mylport->setDataBits(QSerialPort::DataBits::Data8))
{
emitERROR(QString("Помилка відкриття порту"));return;
}
if(!mylport->setParity(QSerialPort::Parity::NoParity))
{
emitERROR(QString("Помилка відкриття порту"));return;
}

if(!mylport->setStopBits(QSerialPort::StopBits::OneStop))
{
emitERROR(QString("Помилка відкриття порту"));return;
}
mylport->moveToThread(mythread);
//mythread->
mythread->start(QThread::TimeCriticalPriority);//
mythread->
states=MyStates::PortConnection;
emitERROR(QString("Порт підключений"));
}else
{
emitERROR(QString("Помилка відкриття порту"));return;
}
}

QByteArray MainWindow::Send_To_Serial_Port(QSerialPort*q,QByteArrayarray,
intReturn_size)
  // функція передачі в ком порт з наступним фіксованим відповіддю від
  ведомого
{

QByteArraytemp, temp_return;// temp - передаємо, temp_return - отримуємо
temp.append(array);
q-> write (temp);
q->waitForBytesWritten(30);
     // q->flush();
     // QThread::msleep(90);//
      //this->thread()->msleep(90);
for(inti=0;i<Return_size;i++)
{
if(q->waitForReadyRead(1050))/ / Чекаємо пів секунди, якщо немає відповіді, то
вихід інакше збираємо
            //  займаємося збиранням із ком-порту
{
temp_return.append(q->read(q->bytesAvailable())); }else
{
emitERROR(QString("Помилка зв'язку з пристроєм"));
states=MyStates::PortConnection;
returntemp_return;
```

```
}
}
QThread::msleep(10);
returntemp_return;
}
voidMainWindow::ConnectHello()//установка з'єднання з STM32
{
if(states==MyStates::PortConnection)
{
mylport->clear();
QByteArraytemp;
temp.append('H');
temp.append('e');
temp.append('l');
temp.append('l');
temp.append('o');
QByteArrayrequest;
request=Send_To_Serial_Port(mylport,temp,5);

if(request.at (0)=='H'&&request.at(1)=='e'&&request.at(2)=='l'&&request.at(3)
=='l'&&request.at(4)=='o'&&(request.size()>=4))
{
states=MyStates::Hello;
emitERROR(QString("З'єднання встановлено"));
emitFlag_block(true);
return;
}else
{
emitERROR(QString("Помилка з'єднання с
пристроєм"));emitFlag_block(false);
return;
}
}else
{
emitERROR(QString("З'єднання встановлено, або не вибраний
порт"));emitFlag_block(false);
}
}

voidMainWindow::Data_To_port()
{
intpoint_to_data=0;
intpoint_to_EEPROM=0;
intpaketmy=0;
if(states==MyStates::Hello)
{
states=MyStates::DataOutPort;
emitERROR(QString("Передача даних"));
QByteArraymybyte;// Передані байт
QByteArrayanswer;//відповідь від STM
emitFlag_block(false);
for(inti=0;i<kolvoSost;i++)//передаємо дані з порту в STM всю структуру
{
            // mylport->clear();mybyte.append('D');
           answer=Send_To_Serial_Port(mylport,mybyte,0);
           mybyte.clear();
answer.clear();//mylport-
>clear();
          // QThread::msleep(100);
mybyte.append(General[i].NextSost.at(0));//передаємо покажчик на слід стан
```

```
answer=Send_To_Serial_Port(mylport,mybyte,2);

if((answer.at(1)=='O'&&answer.at(2)=='K')||(answer.at(0)=='O'&&answer.at(1)=
='K'))
{
paketmy++;
mybyte.clear();
answer.clear();
point_to_data++;
//mylport->clear();

emitData_download(point_to_data);
             // QThread::msleep(100);
}else
{
emitERROR(QString("Помилка передачі
даних")+QString::number(paketmy));
states=MyStates::PortConnection;
return;
}
for(intj=0;j<General[i].TimeToDeath.size();j++)//передаємо час, яке має
опрацювати режим
{
mybyte.append('D');
answer=Send_To_Serial_Port(mylport,mybyte,0);
mybyte.clear();
answer.clear();
             // mylport->clear();
           //   QThread::msleep(100);
mybyte.append(General[i].TimeToDeath.at(j));
answer=Send_To_Serial_Port(mylport,mybyte,2);

if((answer.at(1)=='O'&&answer.at(2)=='K')||(answer.at(0)=='O'&&answer.at(1)=
='K'))
{
paketmy++;
mybyte.clear();
answer.clear();
point_to_data++;
// mylport->clear();

emitData_download(point_to_data);
              //    QThread::msleep(100);
}else
{
emitERROR(QString("Помилка передачі
даних")+QString::number(paketmy));
states=MyStates::PortConnection;
return;
}
}
for(intj=0;j<General[i].Rezim.size();j++)// передаємо значення самого режиму
{
mybyte.append('D');
answer=Send_To_Serial_Port(mylport,mybyte,0);
mybyte.clear();
answer.clear();
             // mylport->clear();
            //  QThread::msleep(100);
mybyte.append(General[i].Rezim.at(j));
answer=Send_To_Serial_Port(mylport,mybyte,2);
```

```
if((answer.at(1)=='O'&&answer.at(2)=='K')||(answer.at(0)=='O'&&answer.at(1)=
='K'))
{
paketmy++;
mybyte.clear();
answer.clear();
point_to_data++;
emitData_download(point_to_data);
                //   QThread::msleep(100);
}else
{
emitERROR(QString("Помилка передачі
даних")+QString::number(paketmy));
states=MyStates::PortConnection;
return;
}
}
}
mybyte.append('C');
mybyte.append('o');
mybyte.append('m');
mybyte.append('p');
mybyte.append('l');
mybyte.append('e');
mybyte.append('t');
mybyte.append('e');
answer=Send_To_Serial_Port(mylport,mybyte,3);
if((answer.at(1)=='Y'&&answer.at(2)=='E'&&answer.at(3)=='S')||(answer.at(0)=
='Y'&&answer.at(1)=='E'&&answer.at(2)=='S'))
{
mybyte.clear();
answer.clear();
                    //mylport-
                      >clear();states=MyStates::C
                      omplete;
emitERROR(QString("Передача даних
завершено"));//+QString::number(paketmy));}els
e
{
emitERROR(QString("Помилка передачі даних
Yes"));//+QString::number(paketmy));
states=MyStates::PortConnection;
return;
}
if(paketmy==kolvoSost*69)
{
emitERROR(QString("Передача
завершено"));//+QString::number(paketmy));
mylport->clear();
emitFlag_block(false);
}else
{emitERROR(QString("помилка передачі
даних"));//+QString::number(paketmy));
mylport->clear();
emitFlag_block(false);
}
states=MyStates::PortConnection;

}else{
emitERROR(QString("Передача даних неможлива, підключіть пристрій-
ство"));
}
}
```

ДОДАТОК Г

Лістинг файлу mainwindow.h

```
#ifndefMAINWINDOW_H
#defineMAINWINDOW_H

#include<QMainWindow>
#include<canals.h>
#include<sost.h>
#include<usart1.h>
//#include <serialthread.h>
#include<myinf.h>
#include<QByteArray>// масиви можна 1
байт#include<QThread>// потоки
окремі#include<QtSerialPort/QSerialPort>
namespaceUi {
classMainWindow;// клас головного вікна

}

classmyinf;


classMainWindow :publicQMainWindow
{
Q_OBJECT

public:
explicitMainWindow(QWidget *parent = 0);
~MainWindow();
signals:
voidchangesost(int);
voidchangekolsost(int);
voidERROR(QString);
voidData_download(int);
voidData_download_EEPROM(int);
voidFlag_block(bool);
private slots:
      // Слоти від кнопок головного вікна
voidon_pushButton_clicked();//кнопка переходу до наступного
вікнаvoidReGeneral();
voidSetComPort(QString name);
voidConnectHello();
voidData_To_port();//Передача даних за своїм протоколом (ППСІС) QByteArray
Send_To_Serial_Port(QSerialPort *q , QByteArray array,intRe-
turn_size);
voidDataFromSost(intpoint,QByteArray Rezim,QByteArray Time-ToDeath,QByteArray
NextSost);//приймаємо дані від сост і записуємо в gen-eral
private:
Ui::MainWindow *ui;
canals *Canal;//Вікно каналів
sost *SST;// вікно стану
usat1 *US;// вікно програмування
public:
myinf *General;
intkolvoMod;// кількість станів у першому
вікніintkolvoSost;
intPointerSost;// вказівник на поточний стан із вікна sost
QSerialPort *mylport;

};
```

```
#endif// MAINWINDOW_H
#include"canals.h"
#include"ui_canals.h"
canals::canals(QWidget*parent) :
QWidget(parent),
ui(new Ui::canals)
{
ui->setupUi(this);

}
canals::~canals()
{
deleteui;
}

voidcanals::on_pushButton_5_clicked()
{
this->close();             // Закриваємо вікно
emitReGeneral();//очищаємо структури General
emitfirstWindow(); // І викликаємо сигнал на відкриття головного вікна
}

voidcanals::on_pushButton_6_clicked()
{
emitcanal1(ui->comboBox_33-
>currentText());emitcanal2(ui->comboBox_34-
>currentText());emitcanal3(ui->comboBox_35-
>currentText());emitcanal4(ui->comboBox_36-
>currentText());emitcanal5(ui->comboBox_37-
>currentText());emitcanal6(ui->comboBox_38-
>currentText());emitcanal7(ui->comboBox_39-
>currentText());emitcanal8(ui->comboBox_40-
>currentText());emitcanal9(ui->comboBox_41-
>currentText());emitcanal10(ui->comboBox_42-
>currentText());emitcanal11(ui->comboBox_43-
>currentText());emitcanal12(ui->comboBox_44-
>currentText());emitcanal13(ui->comboBox_45-
>currentText());emitcanal14(ui->comboBox_46-
>currentText());emitcanal15(ui->comboBox_47-
>currentText());emitcanal16(ui->comboBox_48-
>currentText());
this->close();             // Закриваємо вікно
emitsostWindow(); // І викликаємо сигнал на відкриття головного вікна

}
voidcanals::changecanals(intkolvo)// блокуємо канали всі
{
ui->comboBox_33->clear();
ui->comboBox_34->clear();
ui->comboBox_35->clear();
ui->comboBox_36->clear();
ui->comboBox_37->clear();
ui->comboBox_38->clear();
ui->comboBox_39->clear();
ui->comboBox_40->clear();
ui->comboBox_41->clear();
ui->comboBox_42->clear();
ui->comboBox_43->clear();
ui->comboBox_44->clear();
ui->comboBox_45->clear();
ui->comboBox_46->clear();
```

```
ui->comboBox_47->clear();
ui->comboBox_48->clear();
//////////
ui->comboBox_33->setEnabled(true);
ui->comboBox_34->setEnabled(true);
ui->comboBox_35->setEnabled(true);
ui->comboBox_36->setEnabled(true);
ui->comboBox_37->setEnabled(true);
ui->comboBox_38->setEnabled(true);
ui->comboBox_39->setEnabled(true);
ui->comboBox_40->setEnabled(true);
ui->comboBox_41->setEnabled(true);
ui->comboBox_42->setEnabled(true);
ui->comboBox_43->setEnabled(true);
ui->comboBox_44->setEnabled(true);
ui->comboBox_45->setEnabled(true);
ui->comboBox_46->setEnabled(true);
ui->comboBox_47->setEnabled(true);
ui->comboBox_48->setEnabled(true);
//////////////
QStringListk;
k.append(QString("Невизначений модуль"));
k.append(QString("модуль1"));
k.append(QString("модуль2"));
k.append(QString("модуль3"));
//////////////
ui->comboBox_33->addItems(k);
ui->comboBox_34->addItems(k);
ui->comboBox_35->addItems(k);
ui->comboBox_36->addItems(k);
ui->comboBox_37->addItems(k);
ui->comboBox_38->addItems(k);
ui->comboBox_39->addItems(k);
ui->comboBox_40->addItems(k);
ui->comboBox_41->addItems(k);
ui->comboBox_42->addItems(k);
ui->comboBox_43->addItems(k);
ui->comboBox_44->addItems(k);
ui->comboBox_45->addItems(k);
ui->comboBox_46->addItems(k);
ui->comboBox_47->addItems(k);
ui->comboBox_48->addItems(k);

if(kolvo==1)
{
ui->comboBox_34->setDisabled(true);
ui->comboBox_35->setDisabled(true);
ui->comboBox_36->setDisabled(true);
ui->comboBox_37->setDisabled(true);
ui->comboBox_38->setDisabled(true);
ui->comboBox_39->setDisabled(true);
ui->comboBox_40->setDisabled(true);
ui->comboBox_41->setDisabled(true);
ui->comboBox_42->setDisabled(true);
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==2)
{
```

```
ui->comboBox_35->setDisabled(true);
ui->comboBox_36->setDisabled(true);
ui->comboBox_37->setDisabled(true);
ui->comboBox_38->setDisabled(true);
ui->comboBox_39->setDisabled(true);
ui->comboBox_40->setDisabled(true);
ui->comboBox_41->setDisabled(true);
ui->comboBox_42->setDisabled(true);
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==3)
{
ui->comboBox_36->setDisabled(true);
ui->comboBox_37->setDisabled(true);
ui->comboBox_38->setDisabled(true);
ui->comboBox_39->setDisabled(true);
ui->comboBox_40->setDisabled(true);
ui->comboBox_41->setDisabled(true);
ui->comboBox_42->setDisabled(true);
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==4)
{
ui->comboBox_37->setDisabled(true);
ui->comboBox_38->setDisabled(true);
ui->comboBox_39->setDisabled(true);
ui->comboBox_40->setDisabled(true);
ui->comboBox_41->setDisabled(true);
ui->comboBox_42->setDisabled(true);
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==5)
{
ui->comboBox_38->setDisabled(true);
ui->comboBox_39->setDisabled(true);
ui->comboBox_40->setDisabled(true);
ui->comboBox_41->setDisabled(true);
ui->comboBox_42->setDisabled(true);
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==6)
{
ui->comboBox_39->setDisabled(true);
```

```
ui->comboBox_40->setDisabled(true);
ui->comboBox_41->setDisabled(true);
ui->comboBox_42->setDisabled(true);
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==7)
{
ui->comboBox_40->setDisabled(true);
ui->comboBox_41->setDisabled(true);
ui->comboBox_42->setDisabled(true);
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==8)
{
ui->comboBox_41->setDisabled(true);
ui->comboBox_42->setDisabled(true);
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==9)
{
ui->comboBox_42->setDisabled(true);
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==10)
{
ui->comboBox_43->setDisabled(true);
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==11)
{
ui->comboBox_44->setDisabled(true);
ui->comboBox_45->setDisabled(true);
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==12)
{
ui->comboBox_45->setDisabled(true);
```

```
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==13)
{
ui->comboBox_46->setDisabled(true);
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==14)
{
ui->comboBox_47->setDisabled(true);
ui->comboBox_48->setDisabled(true);
}
if(kolvo==15)
{
ui->comboBox_48->setDisabled(true);
}
}
```

ДОДАТОК Д
Лістинг файлу Canals.h

```
#ifndefCANALS_H
#defineCANALS_H

#include<QWidget>
QT_BEGIN_NAMESPACE
namespaceUi {
classcanals;
}
QT_END_NAMESPACE
classcanals :publicQWidget
{
Q_OBJECT

public:
explicitcanals(QWidget *parent = 0);
~canals();

signals:
voidfirstWindow(); // Сигнал для першого вікнаvoidsostWindow();//
сигнал відкриття вікна стану
voidReGeneral();
voidcanal1(QString);
voidcanal2(QString);
voidcanal3(QString);
voidcanal4(QString);
voidcanal5(QString);
voidcanal6(QString);
voidcanal7(QString);
voidcanal8(QString);
voidcanal9(QString);
voidcanal10(QString);
voidcanal11(QString);
voidcanal12(QString);
voidcanal13(QString);
voidcanal14(QString);
voidcanal15(QString);
voidcanal16(QString);
private slots:
      // Слот-обробник натискання кнопки
voidon_pushButton_5_clicked();// кнопка тому
voidchangecanals(intkolvo);// змінює кількість
каналівvoidon_pushButton_6_clicked();// кнопка Далі

public:
Ui::canals *ui;
};

#endif// CANALS_H
```

ДОДАТОК Е

Лістинг файлу sost.cpp

```
#include"sost.h"
#include"ui_sost.h"
sost::sost(QWidget*parent) :
QWidget(parent),
ui(new Ui::sost)
{
ui->setupUi(this);
sliders= new QSlider*[16];//створюємо вказівники на слайдери і працюємо вже
  с вказівником для зручного циклічного
      переборуsliders[0]=ui-
      >verticalSlider;sliders[1]=ui-
      >verticalSlider_2;sliders[2]=ui-
      >verticalSlider_3;sliders[3]=ui-
      >verticalSlider_4;sliders[4]=ui-
      >verticalSlider_5;sliders[5]=ui-
      >verticalSlider_6;sliders[6]=ui-
      >verticalSlider_7;sliders[7]=ui-
      >verticalSlider_8;sliders[8]=ui-
      >verticalSlider_9;sliders[9]=ui-
      >verticalSlider_10;sliders[10]=ui-
      >verticalSlider_11;sliders[11]=ui-
      >verticalSlider_12;sliders[12]=ui-
      >verticalSlider_13;sliders[13]=ui-
      >verticalSlider_14;
sliders[14]=ui->verticalSlider_15;
sliders[15]=ui->verticalSlider_16;
}
sost::~sost()
{
deleteui;
}

voidsost::on_pushButton_clicked()
{
if(PointSost==1)
{
this->close();             // Закриваємо вікно
emitcanalsWindow(); // І викликаємо сигнал на відкриття головного вікна
}else
{
PointSost--;
ui->label_2->setText(QString::number(PointSost));
}
}

voidsost::on_pushButton_2_clicked()// формуємо масив для передачі
{
if(PointSost==kolvoSost+1)
{
PointSost--;
this->close();             // Закриваємо вікно
emitusatWindow(); // І викликаємо сигнал на відкриття головного вікна
}
else
{
sostnow.clear();//звільняємо пам'ять під нові значення
for(inti=0;i<16;i++)
```

```
{
chara1, a2;//розділимо наші значення на 2 частини для led щоб було по
  8 біт
a1=char((4096/100)*sliders[i]->value());
a2=char((((4096/100)*(sliders[i]->value()))>>8));
                  // sostnow.clear();//звільняємо пам'ять під нові
          значенняif(sliders[i]->value()>=100)
{
a1=char(0xFE); a2=char(0x0F);
}
sostnow.append(char(0x00));sostnow.appe
nd(char(0x00));sostnow.append(a1);
sostnow.append(a2);
}

TimeToDeath.clear();//звільняємо пам'ять під нові значення

TimeToDeath.append(char((ui->spinBox_2-
>value()*60)>>24));TimeToDeath.append(char((ui->spinBox_2-
>value()*60)>>16));TimeToDeath.append(char((ui->spinBox_2-
>value()*60)>>8));TimeToDeath.append(char((ui->spinBox_2->value()*60)));



          //  NextSostPointer.append(char(ui->spinBox->value()>>24));
          //  NextSostPointer.append(char(ui->spinBox->value()>>16));
          //NextSostPointer.append(char(ui->spinBox-
            >value()>>8));NextSostPointer.clear(); //звільняємо пам'ять під
            нові значення
\
NextSostPointer.append(char(ui->spinBox->value())-1);
emitDataToGeneral(PointSost,sostnow,TimeToDeath,NextSostPointer);
PointSost++;
ui->label_2->setText(QString::number(PointSost));
if(PointSost==kolvoSost+1)
{
PointSost--;
this->close();                  // Закриваємо вікно
emitusatWindow(); // І викликаємо сигнал на відкриття головного вікна
}
}
}
voidsost::changekolvosost(intkolvo)// при зміні числа sostoyaniy
{
kolvoSost= Kolvo;
PointSost=1;
ui->spinBox-> setMaximum (kolvo);
}

voidsost::changecanals(intkolvo)// при зміні числа каналів
{
ui->verticalSlider->setEnabled(true);//Все дозволяємоui-
>verticalSlider_2->setEnabled(true);ui->verticalSlider_3-
>setEnabled(true);ui->verticalSlider_4-
>setEnabled(true);ui->verticalSlider_5-
>setEnabled(true);ui->verticalSlider_6-
>setEnabled(true);ui->verticalSlider_7-
>setEnabled(true);ui->verticalSlider_8-
>setEnabled(true);ui->verticalSlider_9-
>setEnabled(true);ui->verticalSlider_10-
>setEnabled(true);ui->verticalSlider_11->setEnabled(true);
```

```
ui->verticalSlider_12->setEnabled(true);ui-
>verticalSlider_13->setEnabled(true);ui-
>verticalSlider_14->setEnabled(true);ui-
>verticalSlider_15->setEnabled(true);ui-
>verticalSlider_16->setEnabled(true);



ui->verticalSlider->setValue(0);//все скидаємоui-
>verticalSlider_2->setValue(0);ui->verticalSlider_3-
>setValue(0);ui->verticalSlider_4->setValue(0);ui-
>verticalSlider_5->setValue(0);ui->verticalSlider_6-
>setValue(0);ui->verticalSlider_7->setValue(0);ui-
>verticalSlider_8->setValue(0);ui->verticalSlider_9-
>setValue(0);ui->verticalSlider_10->setValue(0);ui-
>verticalSlider_11->setValue(0);ui->verticalSlider_12-
>setValue(0);ui->verticalSlider_13->setValue(0);ui-
>verticalSlider_14->setValue(0);ui->verticalSlider_15-
>setValue(0);ui->verticalSlider_16->setValue(0);
QPalettemycolor;

mycolor.setColor(ui->verticalSlider_2->backgroundRole(),Qt::red);//колір
блокованих слайдерів
if(kolvo==1)
{
ui->verticalSlider_2->setPalette(mycolor);ui-
>verticalSlider_3->setPalette(mycolor);ui-
>verticalSlider_4->setPalette(mycolor);ui-
>verticalSlider_5->setPalette(mycolor);ui-
>verticalSlider_6->setPalette(mycolor);ui-
>verticalSlider_7->setPalette(mycolor);ui-
>verticalSlider_8->setPalette(mycolor);ui-
>verticalSlider_9->setPalette(mycolor);ui-
>verticalSlider_10->setPalette(mycolor);ui-
>verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_2->setDisabled(true);ui-
>verticalSlider_3->setDisabled(true);ui-
>verticalSlider_4->setDisabled(true);ui-
>verticalSlider_5->setDisabled(true);ui-
>verticalSlider_6->setDisabled(true);ui-
>verticalSlider_7->setDisabled(true);ui-
>verticalSlider_8->setDisabled(true);ui-
>verticalSlider_9->setDisabled(true);ui-
>verticalSlider_10->setDisabled(true);ui-
>verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==2)
{
```

```
ui->verticalSlider_3->setPalette(mycolor);ui-
>verticalSlider_4->setPalette(mycolor);ui-
>verticalSlider_5->setPalette(mycolor);ui-
>verticalSlider_6->setPalette(mycolor);ui-
>verticalSlider_7->setPalette(mycolor);ui-
>verticalSlider_8->setPalette(mycolor);ui-
>verticalSlider_9->setPalette(mycolor);ui-
>verticalSlider_10->setPalette(mycolor);ui-
>verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_3->setDisabled(true);ui-
>verticalSlider_4->setDisabled(true);ui-
>verticalSlider_5->setDisabled(true);ui-
>verticalSlider_6->setDisabled(true);ui-
>verticalSlider_7->setDisabled(true);ui-
>verticalSlider_8->setDisabled(true);ui-
>verticalSlider_9->setDisabled(true);ui-
>verticalSlider_10->setDisabled(true);ui-
>verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==3)
{

ui->verticalSlider_4->setPalette(mycolor);ui-
>verticalSlider_5->setPalette(mycolor);ui-
>verticalSlider_6->setPalette(mycolor);ui-
>verticalSlider_7->setPalette(mycolor);ui-
>verticalSlider_8->setPalette(mycolor);ui-
>verticalSlider_9->setPalette(mycolor);ui-
>verticalSlider_10->setPalette(mycolor);ui-
>verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_4->setDisabled(true);ui-
>verticalSlider_5->setDisabled(true);ui-
>verticalSlider_6->setDisabled(true);ui-
>verticalSlider_7->setDisabled(true);ui-
>verticalSlider_8->setDisabled(true);ui-
>verticalSlider_9->setDisabled(true);ui-
>verticalSlider_10->setDisabled(true);ui-
>verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==4)
{
```

```
ui->verticalSlider_5->setPalette(mycolor);ui-
>verticalSlider_6->setPalette(mycolor);ui-
>verticalSlider_7->setPalette(mycolor);ui-
>verticalSlider_8->setPalette(mycolor);ui-
>verticalSlider_9->setPalette(mycolor);ui-
>verticalSlider_10->setPalette(mycolor);ui-
>verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_5->setDisabled(true);ui-
>verticalSlider_6->setDisabled(true);ui-
>verticalSlider_7->setDisabled(true);ui-
>verticalSlider_8->setDisabled(true);ui-
>verticalSlider_9->setDisabled(true);ui-
>verticalSlider_10->setDisabled(true);ui-
>verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==5)
{
ui->verticalSlider_6->setPalette(mycolor);ui-
>verticalSlider_7->setPalette(mycolor);ui-
>verticalSlider_8->setPalette(mycolor);ui-
>verticalSlider_9->setPalette(mycolor);ui-
>verticalSlider_10->setPalette(mycolor);ui-
>verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_6->setDisabled(true);ui-
>verticalSlider_7->setDisabled(true);ui-
>verticalSlider_8->setDisabled(true);ui-
>verticalSlider_9->setDisabled(true);ui-
>verticalSlider_10->setDisabled(true);ui-
>verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==6)
{
ui->verticalSlider_7->setPalette(mycolor);ui-
>verticalSlider_8->setPalette(mycolor);ui-
>verticalSlider_9->setPalette(mycolor);ui-
>verticalSlider_10->setPalette(mycolor);ui-
>verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_7->setDisabled(true);
```

```
ui->verticalSlider_8->setDisabled(true);ui-
>verticalSlider_9->setDisabled(true);ui-
>verticalSlider_10->setDisabled(true);ui-
>verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==7)
{

ui->verticalSlider_8->setPalette(mycolor);ui-
>verticalSlider_9->setPalette(mycolor);ui-
>verticalSlider_10->setPalette(mycolor);ui-
>verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_8->setDisabled(true);ui-
>verticalSlider_9->setDisabled(true);ui-
>verticalSlider_10->setDisabled(true);ui-
>verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==8)
{

ui->verticalSlider_9->setPalette(mycolor);ui-
>verticalSlider_10->setPalette(mycolor);ui-
>verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_9->setDisabled(true);ui-
>verticalSlider_10->setDisabled(true);ui-
>verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==9)
{

ui->verticalSlider_10->setPalette(mycolor);ui-
>verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_10->setDisabled(true);
```

```
ui->verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==10)
{

ui->verticalSlider_11->setPalette(mycolor);ui-
>verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_11->setDisabled(true);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==11)
{

ui->verticalSlider_12->setPalette(mycolor);ui-
>verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_12->setDisabled(true);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==12)
{

ui->verticalSlider_13->setPalette(mycolor);ui-
>verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_13->setDisabled(true);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==13)
{

ui->verticalSlider_14->setPalette(mycolor);ui-
>verticalSlider_15->setPalette(mycolor);ui-
>verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_14->setDisabled(true);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==14)
{

ui->verticalSlider_15->setPalette(mycolor);
```

```
ui->verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_15->setDisabled(true);ui-
>verticalSlider_16->setDisabled(true);
}
if(kolvo==15)
{

ui->verticalSlider_16->setPalette(mycolor);ui-
>verticalSlider_16->setDisabled(true);
}
}
/////блоки відображення відсотків на слайдерах
voidsost::on_verticalSlider_valueChanged(intvalue)
{

ui->lcdNumber->display(value);
ui->label_19->setText(QString::number((KCanalVT[0])*(ui->verticalSlider-
>value())));
}
voidsost::on_verticalSlider_2_valueChanged(intvalue)
{
ui->lcdNumber_2->display(value);ui->label_20-
>setText(QString::number((KCanalVT[1])*(ui-
>verticalSlider_2->value())));
}
voidsost::on_verticalSlider_3_valueChanged(intvalue)
{
ui->lcdNumber_4->display(value);ui->label_21-
>setText(QString::number((KCanalVT[2])*(ui-
>verticalSlider_3->value())));
}
voidsost::on_verticalSlider_4_valueChanged(intvalue)
{
ui->lcdNumber_3->display(value);ui->label_22-
>setText(QString::number((KCanalVT[3])*(ui-
>verticalSlider_4->value())));
}
voidsost::on_verticalSlider_5_valueChanged(intvalue)
{
ui->label_23->setText(QString::number((KCanalVT[4])*(ui-
>verticalSlider_5->value())));
ui->lcdNumber_6->display(value);
}
voidsost::on_verticalSlider_6_valueChanged(intvalue)
{
ui->label_24->setText(QString::number((KCanalVT[5])*(ui-
>verticalSlider_6->value())));
ui->lcdNumber_5->display(value);
}
voidsost::on_verticalSlider_7_valueChanged(intvalue)
{
ui->label_26->setText(QString::number((KCanalVT[6])*(ui-
>verticalSlider_7->value())));
ui->lcdNumber_8->display(value);
}
voidsost::on_verticalSlider_8_valueChanged(intvalue)
{
ui->label_30->setText(QString::number((KCanalVT[7])*(ui-
>verticalSlider_8->value())));
ui->lcdNumber_7->display(value);
}
voidsost::on_verticalSlider_9_valueChanged(intvalue)
{
```

```
ui->label_29->setText(QString::number((KCanalVT[8])*(ui-
>verticalSlider_9->value())));
ui->lcdNumber_10->display(value);
}
voidsost::on_verticalSlider_10_valueChanged(intvalue)
{
ui->label_27->setText(QString::number((KCanalVT[9])*(ui-
>verticalSlider_10->value())));
ui->lcdNumber_9->display(value);
}
voidsost::on_verticalSlider_11_valueChanged(intvalue)
{
ui->label_25->setText(QString::number((KCanalVT[10])*(ui-
>verticalSlider_11->value())));
ui->lcdNumber_12->display(value);
}
voidsost::on_verticalSlider_12_valueChanged(intvalue)
{
ui->label_28->setText(QString::number((KCanalVT[11])*(ui-
>verticalSlider_12->value())));
ui->lcdNumber_11->display(value);
}
voidsost::on_verticalSlider_13_valueChanged(intvalue)
{
ui->label_34->setText(QString::number((KCanalVT[12])*(ui-
>verticalSlider_13->value())));
ui->lcdNumber_14->display(value);
}
voidsost::on_verticalSlider_14_valueChanged(intvalue)
{
ui->label_32->setText(QString::number((KCanalVT[13])*(ui-
>verticalSlider_14->value())));
ui->lcdNumber_13->display(value);
}
voidsost::on_verticalSlider_15_valueChanged(intvalue)
{
ui->label_31->setText(QString::number((KCanalVT[14])*(ui-
>verticalSlider_15->value())));
ui->lcdNumber_16->display(value);
}
voidsost::on_verticalSlider_16_valueChanged(intvalue)
{
ui->label_33->setText(QString::number((KCanalVT[15])*(ui-
>verticalSlider_16->value())));
ui->lcdNumber_15->display(value);
}

voidsost::Kcanal(intpointer, QString canal)
{
if(canal==QString("Невизначений модуль"))
{
KCanalVT[pointer]=0;
}
if(canal==QString("модуль1"))
{
KCanalVT[pointer]=0.1;
}
if(canal==QString("модуль2"))
{
KCanalVT[pointer]=0.2;
}
if(canal==QString("модуль3"))
{
```

```
KCanalVT[pointer]=0.3;
}
}
voidsost::canal1(QString canal)
{
ui->label_59-> setText (canal);
emitKcanal(0, Canal);
ui->label_19->setText(QString::number((KCanalVT[0])*(ui->verticalSlider-
>value())));
}
voidsost::canal2(QString canal)
{
ui->label_66-> setText (canal);
emitKcanal(1, Canal);
ui->label_20->setText(QString::number((KCanalVT[1])*(ui-
>verticalSlider_2->value())));
}
voidsost::canal3(QString canal)
{
ui->label_65-> setText (canal);
emitKcanal(2, Canal);
ui->label_21->setText(QString::number((KCanalVT[2])*(ui-
>verticalSlider_3->value())));
}
voidsost::canal4(QString canal)
{
ui->label_58-> setText (canal);
emitKcanal(3, Canal);
ui->label_22->setText(QString::number((KCanalVT[3])*(ui-
>verticalSlider_4->value())));
}
voidsost::canal5(QString canal)
{
ui->label_56-> setText (canal);
emitKcanal(4, Canal);
ui->label_23->setText(QString::number((KCanalVT[4])*(ui-
>verticalSlider_5->value())));
}
voidsost::canal6(QString canal)
{
ui->label_61-> setText (canal);
emitKcanal(5, Canal);
ui->label_24->setText(QString::number((KCanalVT[5])*(ui-
>verticalSlider_6->value())));
}
voidsost::canal7(QString canal)
{
ui->label_54-> setText (canal);
emitKcanal(6, Canal);
ui->label_26->setText(QString::number((KCanalVT[6])*(ui-
>verticalSlider_7->value())));
}
voidsost::canal8(QString canal)
{
ui->label_52-> setText (canal);
emitKcanal(7, Canal);
ui->label_30->setText(QString::number((KCanalVT[7])*(ui-
>verticalSlider_8->value())));
}
voidsost::canal9(QString canal)
{
ui->label_63-> setText (canal);
emitKcanal(8, Canal);
```

```
ui->label_29->setText(QString::number((KCanalVT[8])*(ui-
>verticalSlider_9->value())));
}
voidsost::canal10(QString canal)
{
ui->label_55-> setText (canal);
emitKcanal(9, Canal);
ui->label_27->setText(QString::number((KCanalVT[9])*(ui-
>verticalSlider_10->value()))); }
voidsost::canal11(QString canal)
{
ui->label_51-> setText (canal);
emitKcanal(10, Canal);
ui->label_25->setText(QString::number((KCanalVT[10])*(ui-
>verticalSlider_11->value()))); }
voidsost::canal12(QString canal)
{
ui->label_62-> setText (canal);
emitKcanal(11, Canal);
ui->label_28->setText(QString::number((KCanalVT[11])*(ui-
>verticalSlider_12->value()))); }
voidsost::canal13(QString canal)
{
ui->label_60-> setText (canal);
emitKcanal(12, Canal);
ui->label_34->setText(QString::number((KCanalVT[12])*(ui-
>verticalSlider_13->value()))); }
voidsost::canal14(QString canal)
{
ui->label_57-> setText (canal);
emitKcanal(13, Canal);
ui->label_32->setText(QString::number((KCanalVT[13])*(ui-
>verticalSlider_14->value()))); }
voidsost::canal15(QString canal)
{
ui->label_64-> setText (canal);
emitKcanal(14, Canal);
ui->label_31->setText(QString::number((KCanalVT[14])*(ui-
>verticalSlider_15->value()))); }
voidsost::canal16(QString canal)
{
ui->label_53-> setText (canal);
emitKcanal(15, Canal);
ui->label_33->setText(QString::number((KCanalVT[15])*(ui-
>verticalSlider_16->value()))); }
```

ДОДАТОК Є

Лістинг файлу sost.h

```
#ifndefSOST_H
#define SOST_H

#include<QWidget>
//#include <QByteArray>
#include<QSlider>
QT_BEGIN_NAMESPACE
namespaceUi {
classsost;
}
QT_END_NAMESPACE
classsost :publicQWidget
{
Q_OBJECT

public:
explicitsost(QWidget *parent = 0);
~sost();

signals:
voidusatWindow(); // Сигнал
надалivoidcanalsWindow();// Сигнал для назад
voidDataToGeneral(int,QByteArray,QByteArray,QByteArray);// сигнал для
налаштувань головного вікна, перезапис у масив нових значень тощо.private
slots:
      // Слот-обробник натискання кнопки
voidon_pushButton_clicked();// Сигнал для назад

voidon_pushButton_2_clicked();// Сигнал
надалivoidchangecanals(intkolvo);voidchangekolvosost(int
kolvo);
voidon_verticalSlider_valueChanged(intvalue);// міняємо значення
lcdvoidon_verticalSlider_2_valueChanged(intvalue);
voidon_verticalSlider_3_valueChanged(intvalue);voidon_verti
calSlider_4_valueChanged(intvalue);voidon_verticalSlider_5_
valueChanged(intvalue);voidon_verticalSlider_6_valueChanged
(intvalue);voidon_verticalSlider_7_valueChanged(intvalue);v
oidon_verticalSlider_8_valueChanged(intvalue);voidon_vertic
alSlider_9_valueChanged(intvalue);voidon_verticalSlider_10_
valueChanged(intvalue);voidon_verticalSlider_11_valueChange
d(intvalue);voidon_verticalSlider_12_valueChanged(intvalue)
;voidon_verticalSlider_13_valueChanged(intvalue);voidon_ver
ticalSlider_14_valueChanged(intvalue);voidon_verticalSlider
_15_valueChanged(intvalue);voidon_verticalSlider_16_valueCh
anged(intvalue);

voidcanal1(QString canal);
voidcanal2(QString canal);
voidcanal3(QString canal);
voidcanal4(QString canal);
voidcanal5(QString canal);
voidcanal6(QString canal);
voidcanal7(QString canal);
voidcanal8(QString canal);
```

```
voidcanal9(QString canal);
voidcanal10(QString canal);
voidcanal11(QString canal);
voidcanal12(QString canal);
voidcanal13(QString canal);
voidcanal14(QString canal);
voidcanal15(QString canal);
voidcanal16(QString canal);
voidKcanal(intpointer,QString canal);

public:
Ui::sost*ui;
intkolvoSost;//кількість станів та покажчик на поточний станintPointSost;
QByteArraysostnow;// масив значень світлодіодів у поточному стані
QByteArrayTimeToDeath;// масив значень часу даного стану
QByteArrayNextSostPointer;// масив значень вказівника на наступне від-
          працюваний стан
        QSlider**sliders;
doubleKCanalVT[16]; // коефіцієнт для каналу за потужністю mW
};

#endif// SOST_H
```

ДОДАТОК Ж

Лістинг файлу usat1.cpp

```
#include"usart1.h"
#include"ui_usart1.h"

usat1::usart1(QWidget*parent) :
QWidget(parent),
ui(new Ui::usat1)
{
ui->setupUi(this);

}
usat1::~usart1()
{
deleteui;
}

voidusat1::on_pushButton_2_clicked()
{
this->close();             // Закриваємо вікно
emitsostWindow(); // І викликаємо сигнал на відкриття головного вікна

}
voidusat1::changekolvosost(intkolvo)
{
kolvosost_US= Kolvo;
}



voidusat1::Data_download(intвідсоток)
{
ui->progresBar->setValue((%*100)/(69*kolvosost_US));
}

/*void usart1::Data_download_EEPROM(int%)
{
ui->progressBar_2->setValue((%*100)/731);
}*/

voidusat1::on_pushButton_4_clicked()//кнопка оновлення пристроїв для кому
портів
{
ui->comboBox->clear();
QList<QSerialPortInfo> listcom =QSerialPortInfo::availablePorts();
foreach(infocom, listcom) {
ui->comboBox->addItem(infocom.portName());//додавання в комбобокс
}
}

voidusat1::on_comboBox_activated(constQString &arg1)
// надсилаємо запит на підключення вибраного порту з заданими параметрами
  в головному вікні
{
emitusat1::SetComPort(arg1); //ui-
>label->setText(arg1);
}
```

```
voidusat1::ERROR(QString err)// типу помилка, але використовується також для
виводу повідомлень на рядок стану
{
ui->label_4->setText(err);
}

voidusat1::on_Podkl_clicked()
{
emitConnectHello();
}

voidusat1::on_Rezim_use_clicked()
{
emitData_to_port();
}
voidusat1::Flag_block(boolflag)
{
if(flag==true)
{
ui->Podkl->setDisabled(true);
ui->Rezim_use->setEnabled(true);
}else
{
ui->Podkl->setEnabled(true);
ui->Rezim_use->setDisabled(true);
}
}
```

ДОДАТОК И

Лістинг файлу usat1.h

```
#ifndefUSART1_H
#define USART1_H

#include<QtSerialPort/QSerialPortInfo>
#include<QWidget>
#include<QList>
QT_BEGIN_NAMESPACE
namespaceUi {
classusat1;
}
QT_END_NAMESPACE
classusat1 :publicQWidget
{
Q_OBJECT

public:
explicitusat1(QWidget *parent = 0);
~usart1();

signals:
      // void usatWindow(); // Сигнал для першого вікна на
       відкриттяvoidsostWindow();// Сигнал для назад
voidSetComPort(QString);
voidConnectHello();// потрібно згенерувати слово
вітаннявoidData_to_port();// Потрібно згенерувати передачу даних
private slots:
voidFlag_block(boolflag);
// Слот-обробник кількості
каналівvoidchangekolvosost(intkolvo); //void
on_pushButton_clicked();voidData_download(int
відсоток);
      // void Data_download_EEPROM(int
       відсоток);voidERROR(QString err);
voidon_pushButton_2_clicked();// Сигнал для назад

voidon_pushButton_4_clicked();

voidon_comboBox_activated(constQString &arg1);

voidon_Podkl_clicked();

voidon_Rezim_use_clicked();

public:
Ui::usat1*ui;
QSerialPortInfoinfocom;
intkolvosost_US=0;
};


#endif//USART_H
```

ДОДАТОК К

Лістинг програми для мікроконтролера

```
// ---------------------------------------------- ----------------------------

#include<stdio.h>
#include<stdlib.h>
#include"diag/Trace.h"

#pragmaGCC diagnostic push
#pragmaGCC diagnostic ignored "-Wunused-parameter"
#pragmaGCC diagnostic ignored "-Wmissing-declarations"
#pragmaGCC diagnostic ignored "-Wreturn-type"

#include"stm32f10x.h"
#include"stm32f10x_gpio.h"
#include"stm32f10x_rcc.h"
#include"stm32f10x_tim.h"
#include"stm32f10x_i2c.h"
#include"stm32f10x_rtc.h"
#include"stm32f10x_pwr.h"
#include"misc.h"
////////////////////////////////////////////// //================

uint8_tstateses=0;//стани для прошивки
I2C_InitTypeDef    I2C_InitStructure;
GPIO_InitTypeDef    GPIO_InitStructure3;
uint8_tbuffer_i2c_from_eeprom_to_PCA_VALUE[10][64];//значення    для    ШІМ    від
ЇЄПРОМuint8_tbuffer_i2c_from_eeprom_RTC_VALUE[10][4];//начення RTC зі стану, також
воно доступне лише для поточного стану та оновлюється під час переходу
uint8_tbuffer_i2c_from_eeprom_SECOND_TO_DEATH[10][4];//секунд остаточно стану,
порівнюємо їх із поточним RTC якщо сума більша, то робимо перехід
uint8_tbuffer_i2c_from_eeprom_POINTER_NEXT[10];//покажчик на наступний стан для
кожного стану окремо
uint8_tfrom_eeprom_POINTER_NOW;//Покажчик зараз, зчитується з першого байта ЕЕПРОМ,
а також динамічно листується
//////////////////////////////////////                  ================
uint32_tRTC_Counter = 0;
uint8_tHello_counter=0;
uint8_tComplete_counter=0;
uint8_tFlagRTC=0;//1 - якщо вже записано в її пром, 0 - якщо ще не записано
uint8_tF_states1 = 0;
uint8_tF_states2 = 0;
//uint8_t F_states3=0;
uint8_tF_states4 = 0;
uint16_tDelay_ms;
voidDelay(uint16_t ms)
{
Delay_ms=ms;
while(Delay_ms! = 0);

}
voidusart_init(void)
{
/* Enable USART1 and GPIOA clock */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
```

```
/* NVIC Configuration */
NVIC_InitTypeDefNVIC_InitStructure;
/* Enable the USARTx Interrupt */
NVIC_InitStructure.NVIC_IRQChannel=USART1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority= 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority= 0;
NVIC_InitStructure.NVIC_IRQChannelCmd=ENABLE;
NVIC_Init(&NVIC_InitStructure); NVIC_EnableIRQ(USART1_IRQn);
/* Configure the GPIOs */
GPIO_InitTypeDefGPIO_InitStructure;

/* Configure USART1 Tx (PA.09) as alternate function push-pull */
GPIO_InitStructure.GPIO_Pin= GPIO_Pin_9;
GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);

/* Configure USART1 Rx (PA.10) as input floating */
GPIO_InitStructure.GPIO_Pin= GPIO_Pin_10;
GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOA, &GPIO_InitStructure);

/* Configure the USART1 */
USART_InitTypeDefUSART_InitStructure;

/* USART1 configuration ------------------------------------------------ --------
*/
USART_InitStructure.USART_BaudRate= 9600;
USART_InitStructure.USART_WordLength= USART_WordLength_8b;
USART_InitStructure.USART_StopBits= USART_StopBits_1;
USART_InitStructure.USART_Parity= USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl= USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode= USART_Mode_Rx | USART_Mode_Tx;

USART_Init(USART1, &USART_InitStructure);

/* Enable USART1 */
USART_Cmd(USART1,ENABLE);

/* Enable the USART1 Receive interrupt: this interrupt is generated when the
USART1 receive data register is not empty */
USART_ITConfig(USART1, USART_IT_RXNE,ENABLE);
}

uint8_tHello [5];
uint8_tComplete[8];
uint8_tIsData=0;
uint8_tCount_Next=0;
uint8_tCount_Death=0;
uint8_tCount_PCA=0;
uint8_tCount_SOST = 0;
//uint8_t isUART_OK=0;
voidUSART1_IRQHandler(void)
{
if((USART1->SR & USART_FLAG_RXNE) != (u16)RESET)
{
if(stateses==2)
{
if(Count_PCA>=64)
{
```

```
Count_SOST++;
Count_Next=0;
Count_Death=0;
Count_PCA=0;
}
if(Count_Next==0)
{
buffer_i2c_from_eeprom_POINTER_NEXT[Count_SOST] =
USART_ReceiveData(USART1);
Count_Next++;
}else
{
if(Count_Death<4)
{
buffer_i2c_from_eeprom_SECOND_TO_DEATH[Count_SOST][3-
Count_Death]=USART_ReceiveData(USART1);
Count_Death++;
}else
{
if(Count_PCA<64)
{
buff-
er_i2c_from_eeprom_to_PCA_VALUE[Count_SOST][Count_PCA]=USART_ReceiveData(USART1);
Count_PCA++;
}
}

}
F_states2 = 1;
}else{
if(stateses==1)
{
IsData = USART_ReceiveData(USART1);
if(IsData=='D')
{stateses=2;

}
if(IsData=='C')
{
stateses=3;
Complete[0]='C';
Complete_counter++;
}
}else
{
if(stateses==3)
{
Complete[Complete_counter]=USART_ReceiveData(USART1); Complete_counter++;
}

if((Complete[0]=='C')&&(Complete[1]=='o')&&(Complete[2]=='m')&&(Complete[3]=='p')&&
(Com-
plete[4]=='l')&&(Complete[5]=='e')&&(Complete[6]=='t')&&(Complete[7]=='e')&&(Comple
te_counter==8))
{
stateses=4;
F_states4=1;
Complete_counter=0;
}
}
```

```
}



if(stateses==0)
{
Hello[Hello_counter] = USART_ReceiveData(USART1);
Hello_counter++;

if((Hello[0]=='H')&&(Hello[1]=='e')&&(Hello[2]=='l')&&(Hello[3]=='l')&&(Hello
[4]=='o')&&(Hello_counter==5))
{
F_states1 = 1;
stateses=1;
Hello_counter=0;
}
}

USART_ClearITPendingBit(USART1, USART_IT_RXNE);
}
}



voidSetSysClockToHSE(void)
{
ErrorStatusHSEStartUpStatus;

/* SYSCLK, HCLK, PCLK2 та PCLK1 configuration ----------------------------*/ /* RCC
system reset(for debug purpose ) */
RCC_DeInit();

/* Enable HSE */
RCC_HSEConfig(RCC_HSE_ON);

/* Wait till HSE is ready */
HSEStartUpStatus = RCC_WaitForHSEStartUp();

if(HSEStartUpStatus == SUCCESS)
{
/* Enable Prefetch Buffer */
//FLASH_PrefetchBufferCmd( FLASH_PrefetchBuffer_Enable);

/* Flash 0 wait state */
//FLASH_SetLatency( FLASH_Latency_0);

/* HCLK = SYSCLK */
RCC_HCLKConfig(RCC_SYSCLK_Div1);

/* PCLK2 = HCLK */
RCC_PCLK2Config(RCC_HCLK_Div1);

/* PCLK1 = HCLK */
RCC_PCLK1Config(RCC_HCLK_Div1);

/* Select HSE as system clock source */
RCC_SYSCLKConfig(RCC_SYSCLKSource_HSE);

/* Wait till PLL is used as system clock source
*/while(RCC_GetSYSCLKSource() != 0x04)
```

```
{
}
}
else
{/* Якщо HSE fails to start-up, application will have wrong clock configuration.
User can add here some code to deal with this error */

/* Go to infinite loop */
while(1)
{
}
}
}


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



unsigned charRTC_Init(void)
{
      // Включити тактування модулів керування живленням та керуванням резервною
  областю
RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR | RCC_APB1Periph_BKP,ENABLE);
      // Дозволити доступ до області резервних даних

      // Якщо RTC вимкнено – ініціалізувати
if((RCC->BDCR & RCC_BDCR_RTCEN) != RCC_BDCR_RTCEN)
{
          // Скидання даних у резервній області
RCC_BackupResetCmd(ENABLE);
RCC_BackupResetCmd(DISABLE);

          // Встановити джерело тактування кварц 32768
RCC_LSEConfig(RCC_LSE_ON);
while((RCC->BDCR & RCC_BDCR_LSERDY) != RCC_BDCR_LSERDY) {}
RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE);

RTC_SetPrescaler(0x7FFF);// Встановлюємо дільник, щоб годинник вважав се-
кунди

          // Включаємо RTC
RCC_RTCCLKCmd(ENABLE);

          // Чекаємо на синхронізацію
RTC_WaitForSynchro();

return1;
}
return0;
}

voidI2C1_init(void)//ініціалізація iic порту
{
RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1,ENABLE);
/* Configure I2C_EE pins: SCL and SDA */
GPIO_InitStructure3.GPIO_Pin=               GPIO_Pin_6 | GPIO_Pin_7;
GPIO_InitStructure3.GPIO_Speed=GPIO_Speed_50MHz;
```

```
GPIO_InitStructure3.GPIO_Mode=GPIO_Mode_AF_OD; GPIO_Init(GPIOB,

&GPIO_InitStructure3);

I2C_InitStructure.I2C_Mode= I2C_Mode_I2C;
I2C_InitStructure.I2C_DutyCycle= I2C_DutyCycle_2;
I2C_InitStructure.I2C_OwnAddress1= 0x0;
I2C_InitStructure.I2C_Ack= I2C_Ack_Enable;//з бітом у відповідь
I2C_InitStructure.I2C_AcknowledgedAddress=
I2C_AcknowledgedAddress_7bit;
I2C_InitStructure.I2C_ClockSpeed= 100 000;//100кГц
I2C_Init(I2C1, &I2C_InitStructure);
/* I2C configuration */
/* I2C Peripheral Enable */
/* Apply I2C configuration after enabling it
*/I2C_Cmd(I2C1,ENABLE);
}
//////////////////////////////////==================
I2C_InitTypeDefI2C_InitStructure2;
GPIO_InitTypeDefGPIO_InitStructure4;
//////////////////////////////////////////==========
========voidI2C2_init(void)//ініціалізація iic порту 2
{
RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C2,ENABLE);
/* Configure I2C_EE pins: SCL and SDA */
GPIO_InitStructure4.GPIO_Pin=                GPIO_Pin_10 | GPIO_Pin_11;
GPIO_InitStructure4.GPIO_Speed=GPIO_Speed_50MHz;
GPIO_InitStructure4.GPIO_Mode=GPIO_Mode_AF_OD;

GPIO_Init(GPIOB, &GPIO_InitStructure4);

I2C_InitStructure2.I2C_Mode= I2C_Mode_I2C;
I2C_InitStructure2.I2C_DutyCycle= I2C_DutyCycle_2;
I2C_InitStructure2.I2C_OwnAddress1= 0x0;
I2C_InitStructure2.I2C_Ack= I2C_Ack_Enable;//з бітом у відповідь
I2C_InitStructure2.I2C_AcknowledgedAddress=
I2C_AcknowledgedAddress_7bit;
I2C_InitStructure2.I2C_ClockSpeed= 100 000;//400кГц
I2C_Init(I2C2, &I2C_InitStructure2);


/* I2C configuration */


/* I2C Peripheral Enable */

/* Apply I2C configuration after enabling it */

I2C_Cmd(I2C2,ENABLE);
}
//////////////////////////========================
uint8_tI2C_ReadData(I2C_TypeDef* I2Cx)
{
uint8_tdata;
      // Тут картина схожа, як тільки дані прийшли швиденько зчитуємо їх і по-
обертаємо
while( !I2C_CheckEvent(I2Cx, I2C_EVENT_MASTER_BYTE_RECEIVED) );

data = I2C_ReceiveData(I2Cx);
```

```
returndata;
}
//////////////////////////////////////========== ===========
voidI2C_StartTransmission(I2C_TypeDef* I2Cx, uint8_t
transmissionDirection,uint8_tslaveAddress)
{
        // На всяк випадок чекаємо, доки шина
        зійдетьсяwhile(I2C_GetFlagStatus(I2Cx,
        I2C_FLAG_BUSY));

        // Генеруємо старт
I2C_GenerateSTART(I2Cx,ENABLE);

        // Чекаємо, доки злетить потрібний прапор EV_5
while(!I2C_CheckEvent(I2Cx, I2C_EVENT_MASTER_MODE_SELECT));
        // Посилаємо адресу підлеглому // можливо тут потрібне зрушення вліво
//http://microtechnics.ru/stm32-ispolzovanie-i2c/#comment-8109
//
I2C_Send7bitAddress(I2Cx, slaveAddress, transmissionDirection);
        // А тепер у нас два варіанти розвитку подій – залежно від
  //     обраного напрямку обміну даними

if(transmissionDirection== I2C_Direction_Transmitter)
{
while(!I2C_CheckEvent(I2Cx,
I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED));
}
if(transmissionDirection== I2C_Direction_Receiver)
{
while(!I2C_CheckEvent(I2Cx, I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED));
}
}
////////////////////////////////////////////=====  ======================
voidI2C_EndTransmission(I2C_TypeDef* I2Cx)
{
         // Просто викликаємо готову функцію із SPL для формування стоп сигналу
I2C_GenerateSTOP(I2Cx,ENABLE);
}
////////////////////////////////////////////== ========================
voidI2C_WriteData(I2C_TypeDef* I2Cx, uint8_t data)
{
        // Просто викликаємо готову функцію з SPL і чекаємо, поки дані відлетять
I2C_SendData(I2Cx, data);
while(!I2C_CheckEvent(I2Cx, I2C_EVENT_MASTER_BYTE_TRANSMITTED));
}

//////////////////////////=====================  ================================
voidsleepoffPCA9685()// виведення з режиму сну
{
I2C_StartTransmission (I2C1,
I2C_Direction_Transmitter,0x80);
I2C_WriteData(I2C1, 0x00);
I2C_WriteData(I2C1, 0x80);
I2C_EndTransmission(I2C1);//стоп
}
//////////////////////////////////////////=====  ================================
voidwritePCA9685(uint8_t reg,uint8_t data)// запис у регістри PCA діють після виходу
з режиму sleep
{
I2C_StartTransmission (I2C1, I2C_Direction_Transmitter,0x80);
I2C_WriteData(I2C1, reg);
```

```
I2C_WriteData(I2C1, data);
I2C_EndTransmission(I2C1);//стоп
}

////////////////////////////////////============== ================================
voidTO_PCA()// з буфера в PCA ним після прочитання з eeprom
{

uint8_ti=0x06;

while(i<=0x45)
{
writeP-
CA9685(i,buffer_i2c_from_eeprom_to_PCA_VALUE[from_eeprom_POINTER_NOW][i-6]);
i++;
Delay(2);
}
}
////////////////////////////////////==============
====================================
voidwriteEEPROM(uint16_t adress,uint8_t data)// запис у ЄEPROM необхідна затримка 5
мс
{
uint8_ta_up, a_dn;

I2C_StartTransmission (I2C2, I2C_Direction_Transmitter,0xA0);
a_up=(uint8_t)((adress & 0xFF00)>>8); a_dn=(uint8_t)(adress& 0x00FF);
I2C_WriteData(I2C2, a_up);
I2C_WriteData(I2C2, a_dn);
I2C_WriteData(I2C2, data);
I2C_EndTransmission(I2C2);//стоп
}

uint8_tReadEEPROM(uint16_tadress)// Читання з ЄEPROM
{
uint8_ta_up, a_dn;
uint8_tdata;
I2C_StartTransmission (I2C2,
I2C_Direction_Transmitter,0xA0);
a_up=(uint8_t)((adress & 0xFF00)>>8);//Розбито пекло-
ресу на 2 частини
a_dn=(uint8_t)(adress& 0x00FF);
I2C_WriteData(I2C2, a_up);
I2C_WriteData(I2C2, a_dn);


I2C_GenerateSTART(I2C2,ENABLE);

while(!I2C_CheckEvent(I2C2, I2C_EVENT_MASTER_MODE_SELECT));

I2C_Send7bitAddress(I2C2, 0xA0, I2C_Direction_Receiver);// установка читання за
адресою 0xA0


while(!I2C_CheckEvent(I2C2, I2C_EVENT_MASTER_BYTE_RECEIVED ));

data=I2C_ReadData(I2C2);
I2C_AcknowledgeConfig(I2C2,DISABLE);//ACK не
активний
I2C_EndTransmission(I2C2);//стоп
```

```
returndata;
}

//////////////////================================ ================================
//функція затримки працює разом із перериванням 4 таймери
////////////////////////////////==================== ==========================
voidTIM4_IRQHandler(void)
{

if(TIM_GetITStatus(TIM4, TIM_IT_Update) != RESET)
{
Delay_ms--;//декремент мс
                    // Обов'язково скидаємо прапор
TIM_ClearITPendingBit(TIM4, TIM_IT_Update);
}
}


//////////////////////============================ ========================
voidTIM2_IRQHandler(void)
{

if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
{

if(stateses==0)
{
uint32_ttemp_buf=0;// у цій змінній знаходиться час
закінчення режиму

temp_buf=buffer_i2c_from_eeprom_RTC_VALUE[from_eeprom_POINTER_NOW][0]+buffer_
i2c_from_eeprom_RTC_VALUE[from_eeprom_POINTER_NOW][1]*256+
                                        buff-
er_i2c_from_eeprom_RTC_VALUE[from_eeprom_POINTER_NOW][2]*65536+buffer_i2c_from_eepr
om_RTC_VALUE[from_eeprom_POINTER_NOW][3]*16777216;

temp_buf+=buffer_i2c_from_eeprom_SECOND_TO_DEATH[from_eeprom_POINTER_NOW][0]+
buffer_i2c_from_eeprom_SECOND_TO_DEATH[from_eeprom_POINTER_NOW][1]*256+
                                        buff-
er_i2c_from_eeprom_SECOND_TO_DEATH[from_eeprom_POINTER_NOW][2]*65536+buffer_i2c_fro
m_eeprom_SECOND_TO_DEATH[from_eeprom_POINTER_NOW][3]*1677
RTC_Counter=RTC_GetCounter();

if(temp_buf<RTC_Counter)//якщо час вийшов, то переходимо
в інший режим
{

from_eeprom_POINTER_NOW=buffer_i2c_from_eeprom_POINTER_NEXT[from_eeprom_POINT
ER_NOW];
FlagRTC=0;
writeEEPROM(0x00,from_eeprom_POINTER_NOW);
Delay (5);
}
}
                            // Обов'язково скидаємо прапор
TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
}
}

         voidTIM3_IRQHandler(void)
{
```

```
if(TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)
{
//1 раз на 5 секунд скидаємо покажчик ключового слова
HelloHello_counter=0;
TIM_ClearITPendingBit(TIM3, TIM_IT_Update);
}
}
////////////////////========================== ==========================
 TIM_TimeBaseInitTypeDefTIMER_InitStructure;NVIC_InitT
                            ypeDefNVIC_InitStructure;

GPIO_InitTypeDef                    GPIO_InitStructure2;
voidRead_sosts()//зчитування під час запуску, або при перезаписі в
їїпром
{
uint16_tread_adress=0x00;
uint16_ti=0;
uint16_tj=0;
from_eeprom_POINTER_NOW=ReadEEPROM(read_adress); Delay (5);
read_adress++;
while(i<10)
{
j=0;
buff-
er_i2c_from_eeprom_POINTER_NEXT[i]=ReadEEPROM(read_adress);
read_adress++;
Delay (5);
while(j<4)
{
buff-
er_i2c_from_eeprom_RTC_VALUE[i][j]=ReadEEPROM(read_adress);
read_adress++;
Delay (5);
j++;
}
j=0;
while(j<4)
{
buff-
er_i2c_from_eeprom_SECOND_TO_DEATH[i][j]=ReadEEPROM(read_adress);
read_adress++;
Delay (5);
j++;
}
j=0;
while(j<64)
{
buff-
er_i2c_from_eeprom_to_PCA_VALUE[i][j]=ReadEEPROM(read_adress);
read_adress++;
Delay (5);
j++;
}
j=0;
i++;
}
}
voidWrite_sosts()//запис у їїпром
{
```

```
uint16_twrite_adress=0x00;
uint16_ti=0;
uint16_tj=0;
from_eeprom_POINTER_NOW=0;
writeEEPROM(0x00,0x00);
Delay (5);
USART_SendData(USART1,'W');
Delay (5);
USART_SendData(USART1,'R');
Delay (5);
write_adress++;
while(i<10)
{
j=0;
writeEEPROM(write_adress,buffer_i2c_from_eeprom_POINTER_NEXT[i]);
Delay (5);
write_adress++;

USART_SendData(USART1,'W');
Delay (5);
USART_SendData(USART1,'R');
Delay (5);
while(j<4)
{
writeEEPROM(write_adress,buffer_i2c_from_eeprom_RTC_VALUE[i][j]); Delay
(5);
USART_SendData(USART1,'W');
Delay (5);

USART_SendData(USART1,'R');
write_adress++;
Delay (5);
j++;
}
j=0;
while(j<4)
{
writeEEPROM(write_adress,buffer_i2c_from_eeprom_SECOND_TO_DEATH[i][j]);

Delay(5);USART_SendData(USART1,'W'); Delay (5);

USART_SendData(USART1,'R');
_adress++;
Delay (5);
j++;
}
j=0;
while(j<64)
{
writeEEPROM(write_adress,buffer_i2c_from_eeprom_to_PCA_VALUE[i][j]);
Delay(5);USART_SendData(USART1,'W'); Delay (5);


USART_SendData(USART1,'R');
write_adress++;
Delay (15);
j++;
}
j=0;
i++;
}
```

```
Delay (10);
USART_SendData(USART1,'D');
Delay (5);
USART_SendData(USART1,'o');
Delay (5);
USART_SendData(USART1,'w');
Delay (5);
USART_SendData(USART1,'n');
Delay (5);
}
int
main(int argc,char* argv[])
{
//SetSysClockToHSE();
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
I2C1_init();
I2C2_init();
GPIO_InitTypeDefGPIO_InitStructureA;

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
/* Configure the GPIO_LED pin */
GPIO_InitStructureA.GPIO_Pin=
GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_P
in_7;// PC13 пін на GPIOC режим
GPIO_InitStructureA.GPIO_Mode=GPIO_Mode_Out_PP;
GPIO_InitStructureA.GPIO_Speed=GPIO_Speed_2MHz;
GPIO_Init(GPIOA, &GPIO_InitStructureA);

/* Initialize LED which connected to PC13 */
                                    // Enable PORTC Clock
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
/* Configure the GPIO_LED pin */
GPIO_InitStructure2.GPIO_Pin= GPIO_Pin_13;// PC13 пін на GPIOC
режимGPIO_InitStructure2.GPIO_Mode=GPIO_Mode_Out_PP;
GPIO_InitStructure2.GPIO_Speed=GPIO_Speed_50MHz; GPIO_Init(GPIOC,
&GPIO_InitStructure2);

GPIO_SetBits(GPIOC, GPIO_Pin_13);

GPIO_ResetBits(GPIOC, GPIO_Pin_13);
         // налаштовуємо лічильник на 1 мс, це буде для функції затримки

GPIO_InitTypeDef    GPIO_InitStructure;
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);/*
Configure the GPIO_LED pin */
GPIO_InitStructure.GPIO_Pin= GPIO_Pin_8|GPIO_Pin_5;// PB8 пін на GPIOB
режимGPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz; GPIO_Init(GPIOB,
&GPIO_InitStructure);
GPIO_ResetBits(GPIOB, GPIO_Pin_8);
////////////////////////////////////////////// ////////////////////////////////
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4,ENABLE);

NVIC_InitStructure.NVIC_IRQChannel=TIM4_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority= 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority= 0;
NVIC_InitStructure.NVIC_IRQChannelCmd=ENABLE;
NVIC_Init(&NVIC_InitStructure);

RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4,ENABLE);
```

```
TIM_TimeBaseStructInit(&TIMER_InitStructure);
TIMER_InitStructure.TIM_CounterMode= TIM_CounterMode_Up;
TIMER_InitStructure.TIM_Prescaler= 7200;
TIMER_InitStructure.TIM_Period= 10;
TIM_TimeBaseInit(TIM4, &TIMER_InitStructure);

TIM_ITConfig(TIM4, TIM_IT_Update,ENABLE);
TIM_Cmd(TIM4,ENABLE);

//////////////////////////
if(RTC_Init() == 1) {//ініціалізуємо RTC
Delay (2000);
}

/////////////////////////// ініціалізація 2-го таймера і переривання по
ньомуTIM_TimeBaseInitTypeDefTIMER_InitStructure_TIM2;NVIC_InitTypeDefNVIC_Ini
tStructure_TIM2;
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2,ENABLE);

NVIC_InitStructure_TIM2.NVIC_IRQChannel=TIM2_IRQn;

NVIC_InitStructure_TIM2.NVIC_IRQChannelPreemptionPriority= 1;
NVIC_InitStructure_TIM2.NVIC_IRQChannelSubPriority= 0;
NVIC_InitStructure_TIM2.NVIC_IRQChannelCmd=ENABLE;

NVIC_Init(&NVIC_InitStructure_TIM2);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2,ENABLE);
TIM_TimeBaseStructInit(&TIMER_InitStructure_TIM2);
TIMER_InitStructure_TIM2.TIM_CounterMode= TIM_CounterMode_Up;
TIMER_InitStructure_TIM2.TIM_Prescaler= 7200;
TIMER_InitStructure_TIM2.TIM_Period= 2000;
TIM_TimeBaseInit(TIM2, &TIMER_InitStructure_TIM2);
TIM_ITConfig(TIM2, TIM_IT_Update,ENABLE);
TIM_Cmd(TIM2,ENABLE);

////////////////////////////////////// ініціалізація 3-го таймера і переривання

TIM_TimeBaseInitTypeDefTIMER_InitStructure_TIM3;
NVIC_InitTypeDefNVIC_InitStructure_TIM3;
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3,ENABLE);
NVIC_InitStructure_TIM3.NVIC_IRQChannel=TIM3_IRQn;
NVIC_InitStructure_TIM3.NVIC_IRQChannelPreemptionPriority= 1;
NVIC_InitStructure_TIM3.NVIC_IRQChannelSubPriority= 1;
NVIC_InitStructure_TIM3.NVIC_IRQChannelCmd=ENABLE;
NVIC_Init(&NVIC_InitStructure_TIM3);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3,ENABLE);
TIM_TimeBaseStructInit(&TIMER_InitStructure_TIM3);
TIMER_InitStructure_TIM3.TIM_CounterMode= TIM_CounterMode_Up;
TIMER_InitStructure_TIM3.TIM_Prescaler= 7200;
TIMER_InitStructure_TIM3.TIM_Period= 50 000;
TIM_TimeBaseInit(TIM3, &TIMER_InitStructure_TIM3);
TIM_ITConfig(TIM3, TIM_IT_Update,ENABLE);
TIM_Cmd(TIM3,ENABLE);
//////////////////////////////
usart_init();//ініціалізація USART1
sleepoffPCA9685();//З режиму сну PCA
GPIO_SetBits(GPIOB, GPIO_Pin_8);//Дозволили запис у PCA
GPIO_ResetBits(GPIOB, GPIO_Pin_8);
GPIO_ResetBits(GPIOB, GPIO_Pin_5);//дозволили запис у ЄЕПРОМ
Read_sosts();// вважали всі з її пром
TO_PCA();//встановили значення ШІМ
```

```
GPIO_SetBits(GPIOC, GPIO_Pin_13);
while(1)
{
if(stateses==0)
{
if(FlagRTC==0)//якщо новий стан, то записуємо значення RTC за адресою
{
uint32_tbuff=RTC_GetCounter();
buffer_i2c_from_eeprom_RTC_VALUE[from_eeprom_POINTER_NOW][0]=(buff<<24)>>24;
buffer_i2c_from_eeprom_RTC_VALUE[from_eeprom_POINTER_NOW][1]=(buff<<16)>>24;
buffer_i2c_from_eeprom_RTC_VALUE[from_eeprom_POINTER_NOW][2]=(buff<<8)>>24;
buffer_i2c_from_eeprom_RTC_VALUE[from_eeprom_POINTER_NOW][3]=buff>>24;
writeEEPROM((73*from_eeprom_POINTER_NOW+1)+1,buffer_i2c_from_eeprom_POINTER_N
EXT[3]); Delay (5);
writeEEPROM((73*from_eeprom_POINTER_NOW+1)+2,buffer_i2c_from_eeprom_POINTER_N
EXT[2]); Delay (5);
writeEEPROM((73*from_eeprom_POINTER_NOW+1)+3,buffer_i2c_from_eeprom_POINTER_N
EXT[1]); Delay (5);
writeEEPROM((73*from_eeprom_POINTER_NOW+1)+4,buffer_i2c_from_eeprom_POINTER_N
EXT[0]); Delay (5);
FlagRTC=1;
TO_PCA();//Установка значень ШІМ
}

}
if(stateses==4)//Якщо передача даних закінчено
{
if(F_states4==1)
{
F_states4 = 0;
Delay (15);
USART_SendData(USART1,'Y');
Delay (15);
USART_SendData(USART1,'E');
Delay (15);
USART_SendData(USART1,'S');
Delay (15);
USART_SendData(USART1,'');
                }
         Write_sosts();
GPIO_ResetBits(GPIOB, GPIO_Pin_5);//дозволили запис у ЄЕПРОМDelay
(10);
GPIO_SetBits(GPIOB, GPIO_Pin_5);//заборонили запис у їїпром
Delay (10);
Read_sosts();
GPIO_ResetBits(GPIOB, GPIO_Pin_5);//дозволили запис у ЄЕПРОМ
Delay (10);
Count_SOST = 0;
Count_Next=0;
Count_Death=0;
Count_PCA=0;
TO_PCA();
FlagRTC=0;
stateses=0;
}
if(stateses==1)//якщо почалася передача даних
{
Delay (10);
if(F_states1==1)
{
Delay (3);
```

```
USART_SendData(USART1,'H');
Delay (3);
USART_SendData(USART1,'e');
Delay (3);
USART_SendData(USART1,'l');
Delay (3);
USART_SendData(USART1,'l');
Delay (3);
USART_SendData(USART1,'o');
Delay (3);
USART_SendData(USART1,'');
F_states1 = 0;
}
}
if(stateses==2)// якщо прийшло ключове слово D
{
if(F_states2==1)
{
Delay (10);
USART_SendData(USART1,'O');
Delay (10);
USART_SendData(USART1,'K');
Delay (10);
USART_SendData(USART1,'');
F_states2 = 0;
stateses=1;
}
}
}
}

#pragmaGCC diagnostic pop
```

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Комп'ютеризована система для управління освітленням в оранжереї.

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 98,9% Схожість 1,1%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Храпко Я. О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Колесник І. С.
(підпис) (прізвище, ініціали)