Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Комп'ютерна система моніторингу стану повітря для холодильної установки

Виконав(ла): студент(ка) *IV* курсу, групи *CIc-43*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

	(підпис)	Купленний О.Д. (прізвище та ініціали)
Керівник	(підпис)	Тиш Є.В. (прізвище та ініціали)
Нормоконтроль	(підпис)	Луцик Н.С. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Осухівська Г.М. (прізвище та ініціали)
Рецензент	(підпис)	Загородна Н.В. (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« ____ » _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту Купленному Олександру Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Комп'ютерна система моніторингу стану повітря для холодильної установки

Керівник роботи Тиш Євгенія Володимирівна кандидат технічних наук, доцент кафедри
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*23*» *березня* *2022* року № *4/7-180*

2. Термін подання студентом завершеної роботи 23. 06. 2022 р.

3. Вихідні дані до роботи Технічне завдання

4. Зміст роботи (перелік питань, які потрібно розробити)

Вступ.

1. Аналіз технічного завдання.
2. Проектна частина.
3. Практична частина.
4. Безпека життєдіяльності, основи охорони праці.

Висновок.

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

1. Структурна схема.
2. Схема електрична принципова.
3. Налаштування мікроконтролера.
4. Блок – схема алгоритму роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	*Лазарюк В.В., к.т.н., доц. каф. МТ*		
основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Розробка технічного завдання	23.03-31.03.22	Виконано
2	Розділ 1 Аналіз технічного завдання	01.04-12.04.22	Виконано
3	Розділ 2 Проектна частина	13.04-28.04.22	Виконано
4	Розділ 3 Практична частина	29.04-15.05.22	Виконано
5	Розділ 4 Безпека життєдіяльності, основи охорони праці	16.05-27.05.22	Виконано
6	Оформлення пояснювальної записки	28.05-05.06.22	Виконано
7	Оформлення графічної документації	06.06-13.06.22	Виконано
8	Попередні захист кваліфікаційної роботи	14.06-21.06.22	Виконано
9	Захист кваліфікаційної роботи	22.06-23.06.22	Виконано

Студент _____ Купленний О. Д.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Тиш Є.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп’ютерна система моніторингу стану повітря для холодильної установки // Кваліфікаційна робота бакалавра // Купленний Олександр Дмитрович // Тернопільський національний технічний університет імені Івана Пулюя, спеціальність 123 «Комп’ютерна інженерія» // Тернопіль, 2022 // с. – 57, рис. – 36, табл. – 5, аркушів А1 – 4, бібліогр. – 18.

Ключові слова: комп’ютерна система, температура, вологість, тиск, датчики, STM32, I2C, SPI.

Кваліфікаційна робота бакалавра присвячена розробці комп’ютерної системи моніторингу стану повітря для холодильної установки. Система повинна вимірювати температуру, відносну вологість повітря і атмосферний тиск. Пояснювальна записка складається з 4 розділів.

У першому розділі здійснено аналіз технічного завдання, розглянуто існуючі рішення і проаналізовано можливості вирішення поставленого завдання.

У другому розділі описана проектна частина кваліфікаційної роботи. Розроблено структурну схему комп’ютерної системи, розроблено електричну принципову схему, обґрунтовано вибір апаратного і програмного забезпечення і обґрунтовано вибір шин для обміну інформацією.

Третій розділ містить практичну частину. В ньому описана реалізація програмного забезпечення, розробка алгоритму роботи програми і тестування вже готового програмного забезпечення.

В четвертому розділі описані питання безпеки життєдіяльності і основи охорони праці.

ABSTRACT

Computer – aided system for monitoring the condition of the air for the refrigeration unit // Bachelor's work // Kuplennyi Oleksandr Dmytrovych // Ivan Puluy Ternopil National Technical University, specialty 123 «Computer engineering» // Ternopil, 2022 // p. – 57, fig. – 36, tab. – 5, posters A1 – 4, ref. – 18.

Keywords: computer system, temperature, humidity, pressure, sensor, STM32, I2C, SPI.

The bachelor`s work is dedicated to the development of computer – aided system for monitoring the condition of the air for the refrigeration unit. The system must measure temperature, relative humidity and atmospheric pressure. The bachelor`s work consist of four sections.

In the first section the analysis of the technical task is carried out, the existing decisions are considered and possibilities of the decision of the set task are analyzed.

The second section describes the project part of the bachelor`s work. The structural scheme of the computer system is developed, the electric schematic scheme is developed, the choice of hardware and software is substantiated and the choice of buses for information exchange is substantiated.

The third section contains a practical part. It describes the implementation of software, development of the algorithm of the program and testing of ready-made software.

The fourth section describes the issues of life safety and the basics of labor protection.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

I2C – Inter – integrated Circuit;

SPI – Serial Peripheral Interface;

OLED – organic light – emitted diode;

RGB – red. green, blue;

АЦП – аналогово – цифровий перетворювач;

°C – градуси Цельсія;

ВСТУП

Люди давно навчились використовувати різні фізичні параметри, такі як температура, відносна вологість чи атмосферний тиск, для своїх потреб. З часом з’явилась необхідність вимірювати певні параметри, для того, щоб краще їх використовувати. Перші термометри і барометри винайшов Галілео Галілей зі своїми учнями на межі 16 – 17 століття. З часом вимірювальні прилади ставали досконалішими і точнішими, а також з’явились системи вимірювання тих чи інших величин.

В наш час з’явилось багато нових досконаліших вимірювальних приладів. На відміну від старовинних приладів, які засновані на зміні фізичних параметрів речовин, сучасні пристрої є електронними. Об’єднавши кілька електронних вимірювальних приладів в одну систему, можна досягнути великої точності вимірювання і спростити процес отримання даних великої території.

У цій кваліфікаційній роботі бакалавра розглянуто створення комп’ютерної системи моніторингу стану повітря для холодильної установки. Система буде вимірювати фізичні параметри повітря, зокрема його температуру, вологість і тиск в установці.

Для вимірювання необхідних параметрів будуть використовуватись цифрові датчики SHT30 і BMP280. Для передачі даних в середині системи будуть використовуватись протоколи I2C і SPI. Зв'язок з користувачем система буде підтримувати через OLED –дисплей, світлодіодну індикацію і кнопку керування.

Пристрій розробляється на основі мікроконтролера STM32F103C8. Налаштування і програмування мікроконтроллера здійснювалось в середовищі STM32CubeMX і CoIDE (Coocox).

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз вимог до комп’ютерної системи

Давно помічено що на тривалість зберігання різних речовин, будь то продукти, ліки чи щось інше, впливають фактори навколишнього середовища, а саме температура, відносна вологість і атмосферний тиск. Оптимальні параметри при яких певна речовина зберігається якнайдовше і при тому зберігає необхідні властивості є індивідуальною для кожного продукту. Наприклад дорогі сири після виготовлення протягом певного часу дозрівають у сирних льохах, де вони наповнюються смаком і ароматом. Температура повітря в таких підвалах не має виходити за межі 0 – 8°C, а вологість повітря повинна бути від 75% до 90%. Відхилення від цих параметрів призведе до того, що дорогий продукт втратить свою якість і ціну відповідно. В загальному ж тенденція вказує на те, що при нижчих температурах і сухішому повітрі речовини, зокрема продукти, зберігаються краще. Що ж до тиску, то він використовується у спеціальному науковому обладнанні, зокрема в барокамерах і кліматичних кімнатах. В них протягом певного часу зберігаються різні речі, наприклад радіодеталі, і перевіряється вплив на них тиску і інших факторів. В таких пристроях точність і швидкість вимірювання є запорукою того, що експеримент буде проведено правильно.

У кваліфікаційній роботі бакалавра описана розробка системи для моніторингу стану повітря для холодильної установки. Цей пристрій призначений для того щоб у режимі реального часу слідкувати за параметрами температури, вологості і тиску повітря у замкненому просторі. Ідеальним прикладом такого середовища виступають морозилки чи холодильники.

Комп’ютерна система підтримує підключення кількох пар однакових датчиків. Це дає можливість проводити одночасне вимірювання параметрів у кількох невеликих морозилках. Якщо потрібно встановити систему у великому приміщенні, наприклад холодильній кімнаті, то датчики системи можна встановити у кількох критичних місцях кімнати, наприклад біля дверей, вентиляції і т.д. Це дасть можливість оцінити вологість і температуру в різних кінцях приміщення, також система може виводити середнє значення виміряних параметрів. Такий підхід до вимірювання дає можливість збільшити точність і зменшити вплив випадкових чинників на результати вимірів.

Датчики, які використовуються у системі мають великий діапазон вимірюваних температур (можуть працювати, як з від’ємними, так і з додатними значеннями), вологості і тиску. Таким чином, внісши певні зміни в програмний код, пристрій можна використовувати для моніторингу параметрів при нормальних і високих температурах, наприклад у теплицях чи тераріумах. Інформація з датчиків буде виводитись у зрозумілій для користувача формі (градусах Цельсія, відсотках відносної вологості і міліметрах ртутного стовпчика) на OLED – дисплей, який може забезпечити хорошу контрастність зображення і великий кут огляду.

Головні вимоги до системи:

– Комп’ютерна система повинна забезпечувати коректне зчитування параметрів температури, вологості і атмосферного тиску з навколишнього середовища.

– У системі має бути коректне відображення інформації на дисплеї у зрозумілій для користувача формі.

– Система повинна надавати користувачу можливість встановлювати контрольні межі температури, вологості і тиску.

– Система повинна забезпечити постійний нагляд за параметрами на всіх датчиках у режимі реального часу, для цього буде використана світлодіодна індикація.

– Система повинна обраховувати середнє значення параметрів для кількох датчиків.

– У випадку виходу параметрів за межі задані користувачем світлодіоди повинні міняти колір, вказуючи таким чином у яку сторону виміряне значення відрізняється від заданого і на якому датчику це сталося. .

– Система повинна забезпечити по можливості низьке енергоспоживання.

– У системі має залишатись можливість адаптації для роботи не лише при низьких, а й при високих температурах.

– Система повинна бути автономною і працювати без підключення до ПК.

1.2 Огляд існуючих рішень

1.2.1 Логер температури AZ-88394

Прикладом пристрою, який виконує функцію схожу до розроблюваної системи є Логер температури AZ-88394. Це комбінований двоканальний температурний реєстратор. Він містить контактний датчик і використовується для контролю і запису температури в режимі реального часу (див. рис. 1.1).

Рисунок 1.1 - Логер температури AZ-88394

Логер температури двоканальний AZ-88394 використовують для збору інформації про температуру в різних приміщеннях, таких, як склади, лабораторії, підвали, винні льохи, кабіни літаків, автомобілі з холодильними установками, контейнери для зберігання та перевезення харчових продуктів і медичних засобів, в інкубаторах, художніх галереях, музеях тощо.

Для зчитування показань вимірів використовується РК – дисплей, також в логері передбачена можливість запису показників. Логер можливо підключити до ПК або ноутбука за допомогою USB – роз'єму. Вимірювання здійснюється автоматично, згідно із заданим інтервалом. Інтервал може становити від 1 секунди до 12 годин [13].

Даний пристрій має кілька недоліків, а саме він не може вимірювати відносну вологість повітря і атмосферний тиск. Також логер використовує лише один датчик температури. В наслідок цього пристрій не можна використовувати для одночасного контролю температури в кількох приміщеннях.

1.2.2 Система моніторингу температури в холодильних камерах «ОВЕН»

Ця система призначена для вимірювання температури в холодильних камерах, з її допомогою користувач може віддалено здійснювати контроль за температурою і оперативно відреагувати на будь – яку аварійну ситуацію.

Для моніторингу за холодильними камерам використовується хмарний сервіс. Система розрахована на підключення восьми холодильних камер (або менше). Вона організовує безперервне вимірювання температур,а також архівує дані і надсилає аварійні повідомлення користувачу на ПК або смартфон.

Вимірювання здійснюється за допомогою датчиків температури ОВЕН ДТС125Л, від них сигнали передаються до модуля аналогового вводу ОВЕН МВ110-8А. Результати вимірювань температури передаються на зберігання у хмарний сервіс (для цього використовується GPRS або Ethernet). На рис. 1.2 зображено основні функціональні вузли системи.

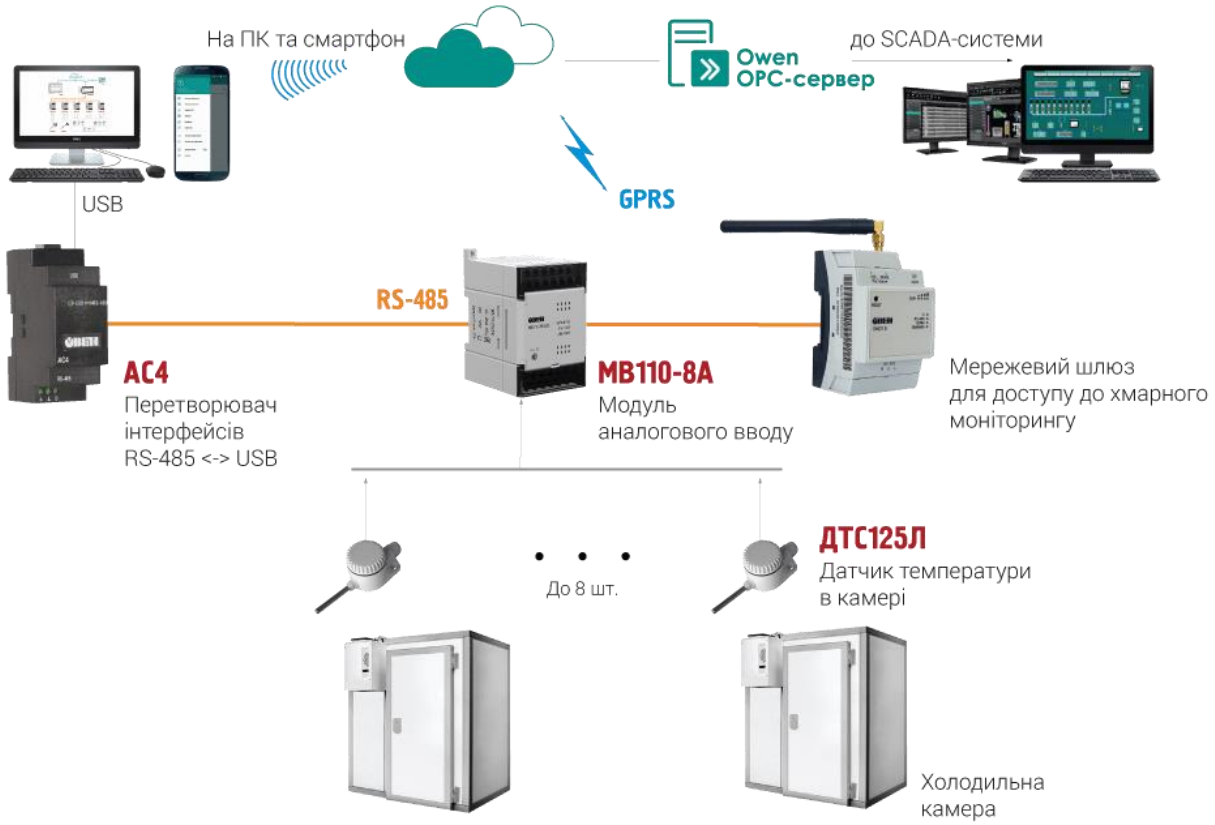

Рисунок 1.2 – Основні функціональні вузли системи «ОВЕН»

Ця система має ряд переваг, а саме:

– Можливість постійного моніторингу температури в холодильних камерах і контроль за температурним режимом.

– Простота встановлення системи. Для впровадження не потрібно змінювати поточну автоматику, таким чином це може зробити штатний співробітник без допомоги спеціаліста.

– Можливість тривалого збереження даних. Для архівації виміряної інформації використовується хмарний сервіс, також дані можуть інтегруватися в SCADA-систему.

– Наявність системи сповіщень і віддаленого контролю. Система надсилає на ПК або смартфон інформацію про вимірювання або у разі непередбачуваної ситуації.

Особливості системи:

– Збереження подій у вигляді графіків і таблиць.

– Створення різних груп користувачів, з різними правами доступу.

– Розсилка повідомлень про аварії окремо для кожної камери (за допомогою Telegram, e-mail, Push-повідомлення).

– Інтеграція в SCADA-систему з підтримкою технології OPC.

– Архів подій і вимірювань може безкоштовно зберігатись на серверах ОВЕН протягом 90 днів з подальшою заміною старих даних новими.

– За необхідності систему можна розширити. Для цього потрібно підключити модулі МВ110-8А (до 40 холодильних камер).

Стандартна комплектація передбачає підключення 8 холодильних камер одночасно. В неї входить вісім датчиків температури повітря, один модуль аналогового входу з універсальними входами і автоматичний перетворювач інтерфейсів USB/RS-485. В разі необхідності систему можна розширити до 40 холодильних камер [14].

Розглянута система є схожою до розроблюваної, проте має певні відмінності. Розроблювана система повинна вимірювати не лише температуру повітря, а й відносну вологість і атмосферний тиск, оскільки ці параметри також надзвичайно важливі для тривалого зберігання продукції. В розглянутій системі дані з датчиків виводяться через хмарне середовище на ПК або смартфон, в розроблюваній системі інформація буде виводитись на екран, в якості допоміжного інструмента буде використовуватись світлодіодна індикація.

1.3 Аналіз можливих рішень поставленого завдання

Комп’ютерна система моніторингу стану повітря для холодильної установки буде працювати на основі мікроконтролера STM32F103C8T6. Цей мікроконтролер є ідеальним вибором за співвідношенням ціна – якість. При відносно невисокій ціні він надає можливість роботи з різними шинами. При тому, що мікроконтролер залишається простим у використанні і програмуванні,

його потужність набагато вища порівняно з аналогами, наприклад платами Arduino.

Мікроконтролери серії STM32 є надзвичайно популярними, завдяки цьому існує велика кількість програмних середовищ призначених для роботи з цими мікроконтролерами. Їх всіх можна умовно поділити на дві частини: ті що були розроблені спеціально для STM32 і адаптовані середовища.

Прикладом останніх може виступати Arduino IDE. Це середовище розроблене для роботи з платами Arduino і завантаживши додаткові бібліотеки його можна використовувати для програмування STM32.

Перевагами такого підходу є швидкість освоєння мікроконтролера і простота написання програм. Мова програмування Arduino є дуже простою і швидко вивчається, а користувачі які збираються працювати з STM32 здебільшого вже працювали з Arduino в минулому.

Проте в такого підходу є і недоліки, перш за все це самі бібліотеки. Більшість з них розроблена окремими користувачами, а не організаціями, відповідно ніхто не може гарантувати правильність роботи цих бібліотек. Також оскільки бібліотеки не є офіційними, то по них важко знайти додаткову інформацію в мережі Інтернет. Оскільки Arduino IDE не розроблялось для роботи з STM32, то воно не може в повній мірі розкрити і використати всі особливості і переваги, які надають ці мікроконтролери.

Прикладом програм створених для роботи з STM32 є набір програмних засобів від STMicroelectronics:

ST MCU Finder – додаток для смартфонів, який дозволяє на першому етапі розробки вибрати оптимальний контролер.

STM32CubeMX – графічний редактор для конфігурації мікроконтролерів STM32. Він дозволяє автоматично генерувати конфігураційний С – код і основу програми за допомогою візуальних утиліт.

SW4STM32 – інтегроване середовище розробки вбудованого ПЗ для мікроконтролерів STM32. Дозволяє писати програми, завантажувати і відлагоджувати їх.

STM Studio – утиліта яка дозволяє відслідковувати значення користувацьких змінних при виконанні програми в реальному часі.

При розробці комп'ютерної системи використовувався графічний редактор STM32CubeMX.

Крім продукції компанії STMicroelectronics існують і інші IDE, створені спеціально для роботи з STM32 наприклад ARM Keil і CooCox(CoIDE).

ARM Keil – це найбільш повне середовище для розробки ПЗ для різних сімейств мікроконтролерів STM32. Це середовище має широкий функціонал і дає можливість створювати проекти на основі великої кількості програмних модулів (операційні системи, файлові системи, стеки популярних протоколів). Основним недоліком цього середовища розробки є його ціна. Також Keil є досить складним і потребує часу, щоб в ньому розібратись.

CooCox(CoIDE) – це високо інтегроване середовище розробки програмного забезпечення, призначене для розробки коду мікроконтролерів архітектури ARM. На відміну від Keil CooCox є одним з найпростіших середовищ в плані встановлення, налаштування і освоєння. Програма заснована на базі Eclipse і має всі її переваги. Середовище пристосовано для роботи з мікроконтролерами серії ST, також його можна використовувати для роботи з контролерами сімейств: Atmel, Texas Instruments, Holtek, NXP, Freescale, Nuvoton, Energy Micro. Середовище має вбудований ST – Link відлагоджувач, який підтримує всі основні режими відлагодження.

Для написання і відлагодження програмного коду при створенні кваліфікаційної роботи бакалавра буде використовуватись середовище CooCox(CoIDE).

Таким чином, для створення програмного забезпечення комп'ютерної системи будуть використовуватись програми створені спеціально для роботи з мікроконтрлерами STM. За рахунок цього можливість виникнення програмних помилок і збоїв є дуже низькою.

РОЗДІЛ 2 ПРОЕКТНА ЧАСТИНА

2.1 Розробка узагальненої структури комп’ютерної системи

На рис. 2.1 зображена структурна схема системи моніторингу стану повітря для холодильної установки.

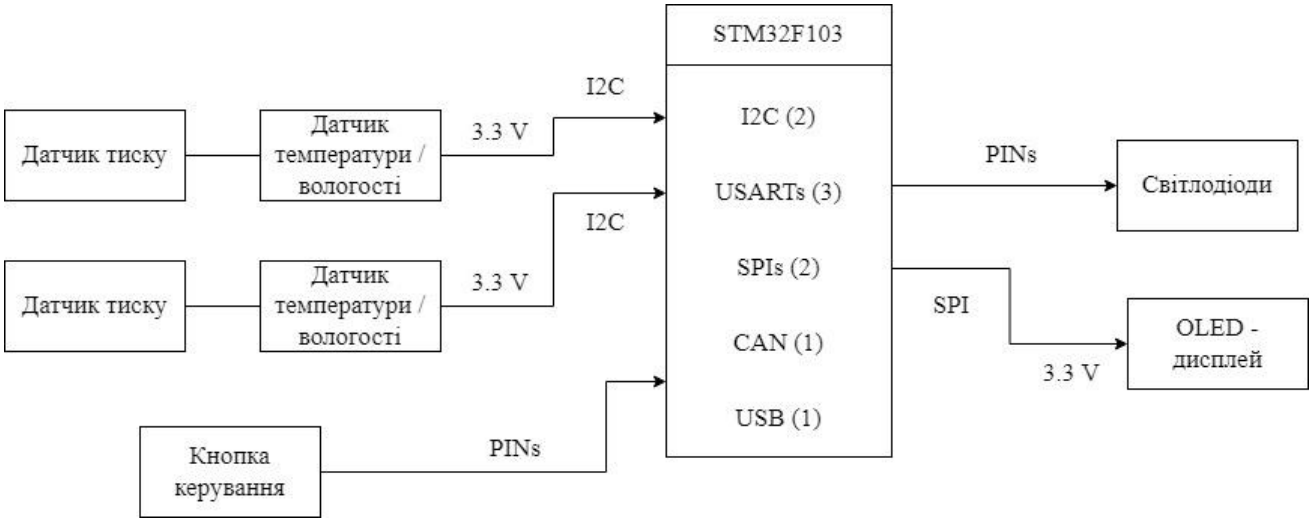

Рисунок 2.1 – Структурна схема комп’ютерної системи моніторингу стану повітря для холодильної установки

У даній системі можна логічно виділити три функціональні частини, а саме: блок збирання даних, блок керування системою і блок відображення інформації.

Блок збирання даних – ця частина пристрою призначена для зчитування фізичних параметрів з навколишнього середовища. Вона складається з цифрових датчиків температури/вологості і датчиків атмосферного тиску. В цьому проекті використовується по два комплекти датчиків (датчик температури/вологості + датчик тиску), проте в разі необхідності їхню кількість можна збільшити.

Блок відображення інформації – ця частина призначена для відображення інформації, отриманої від датчиків, користувачу. У цей блок входить екран і світлодіоди. На екрані дисплею будуть виводитись значення параметрів у зрозумілій користувачу формі (температура в градусах Цельсія, відносна вологість у відсотках, тиск в міліметрах ртутного стовпчика), проте за один раз можуть відображатись дані лише з одного комплекту датчиків, для перемикання між комплектами використовується кнопка. Використання лише самого екрану має певні недоліки, оскільки поки екран показує значення першого комплекту, користувач не знає що відбувається на інших датчиках. Для вирішення цієї проблеми використовуються світлодіоди. Кожен комплект датчиків під'єднаний до відповідного RGB - світлодіода, якщо значення фізичних параметрів на датчиках відповідає тим, які задав в програмному коді користувач, то діод світиться одним кольором. Якщо параметри відмінні в більшу або меншу сторону, то колір світлодіода змінюється.

Блок керування системою – ця частина здійснює керування всією системою і виступає зв'язною ланкою між блоком збирання даних і блоком відображення інформації. Ця частина складається з мікроконтролера STM32 F103 C8 і підключеної до нього кнопки. Мікроконтролер зберігає в собі код який керує роботою системи, координує роботу інших блоків системи і здійснює обробку інформації. Кнопка призначена для керування екраном. З її допомогою користувач може вибирати датчик з якого буде відображатись інформація на екран.

Електрична принципова схема пристрою зображена на рис. 2.2.

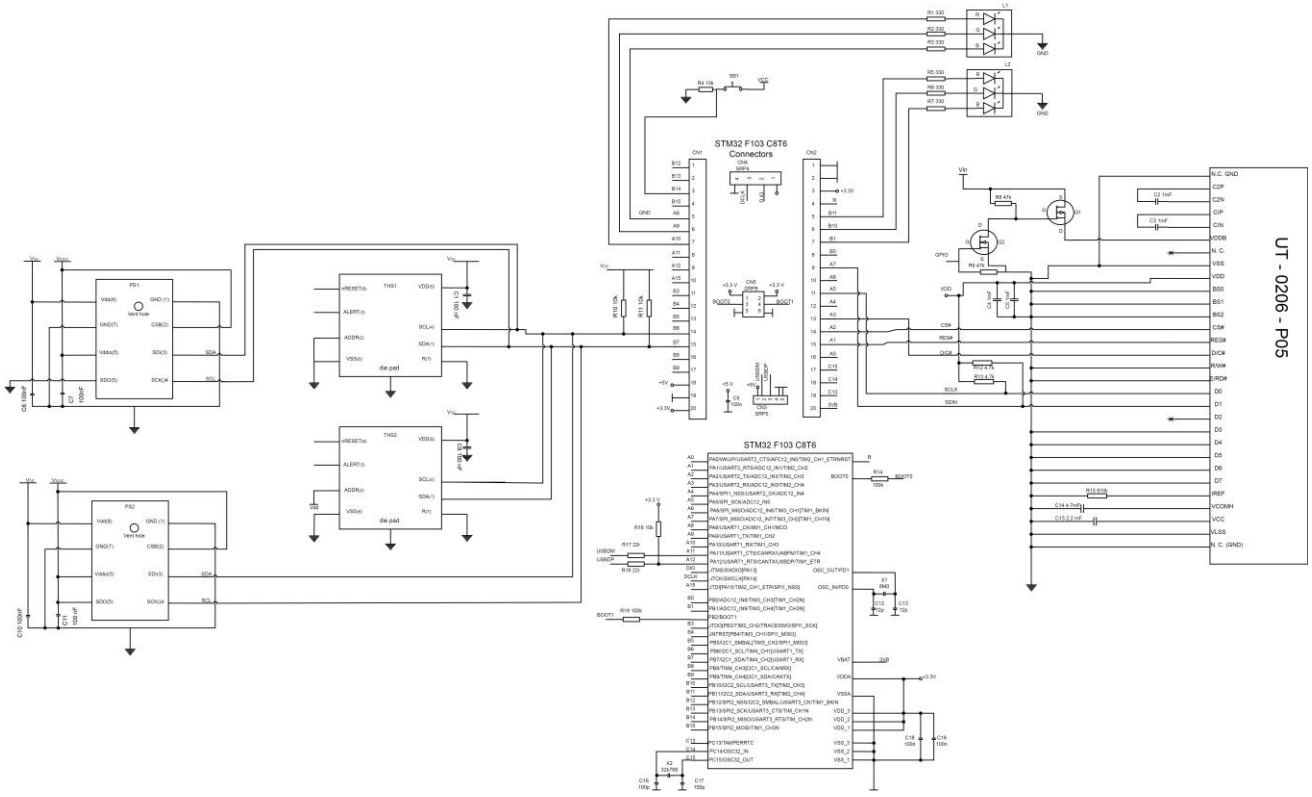

Рисунок 2.2 – Електрична принципова схема комп’ютерної системи моніторингу стану повітря для холодильної установки

2.2 Обґрунтування вибору апаратного забезпечення проектованого комп’ютерного засобу

2.2.1 STM32 F103 C8

Комп’ютерна система моніторингу стану повітря розробляється на базі мікроконтролера STM32 F103 C8. STM32 F103 – це 32 – розрядний мікроконтролер, по вартості він співставний з мікроконтролерами Arduino, проте він значно потужніший, що дозволяє виконувати на ньому набагато складніші завдання.

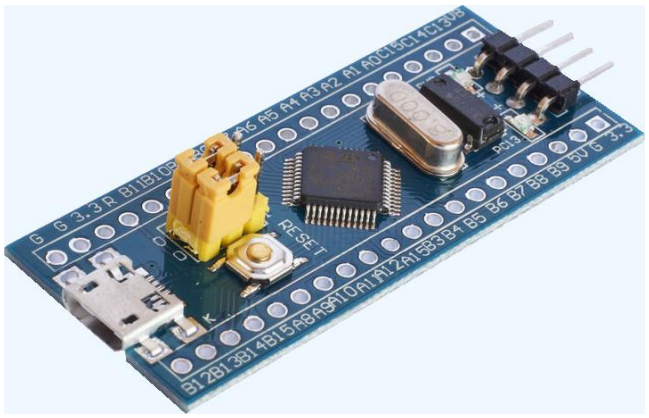

Рисунок 2.3 – Зовнішній вигляд мікроконтролера STM32 F103

Ще однієї особливістю STM32F103 порівняно з мікроконтролерами Arduino є те, що кожен вивід може виконувати кілька функцій. Виводи налаштовуються користувачем, який визначає яке завдання буде виконувати той чи інший вивід. На рис. 2.4 зображено виводи STM32F103 і функції які вони виконують.

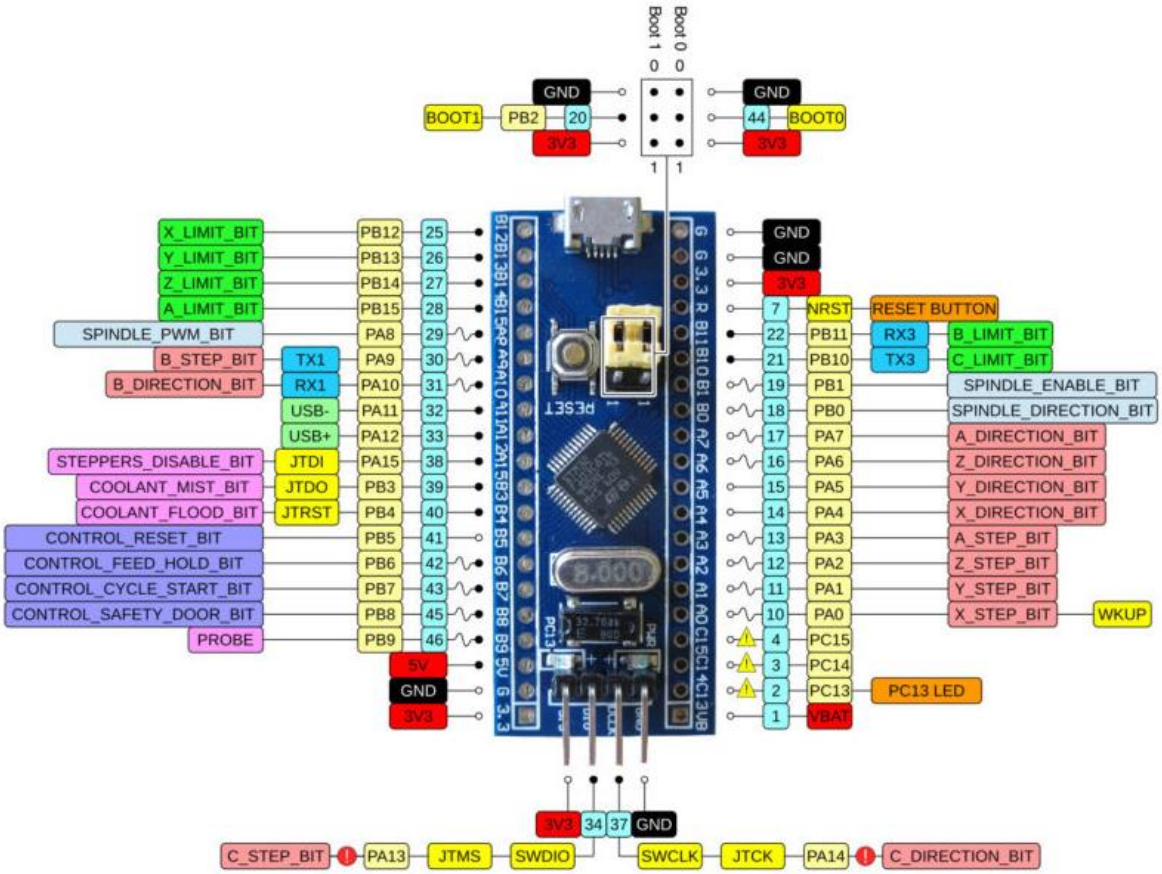

Рисунок 2.4 – Схема виводів мікроконтролера STM32 F103

Характеристики STM32 F103 [7]:

1) Ядро: ARM 32-bit Cortex™-M3 CPU

– 72 MHz максимальна частота, 1.25 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state memory access

– Одноциклове множення і ділення

2) Пам’ять:

– 64 або 128 Кб флеш – пам’яті

– 20 Кб SRA

3) Clock, скидання і керування підтримкою:

– Від 2.0 до 3.6 V живлення і I/Os

– POR, PDR, і програмований детектор напруги (PVD)

– Від 4 до 16 МГц кварцовий генератор

– Вбудований 8 МГц (заводське налаштування) RC

– Вбудований 40 КГц RC

– PLL для CPU clock

– 32 kHz oscillator for RTC with calibration

4) Низька потужність:

– Режими сну, зупинки і очікування

– VBAT підтримка для RTC і резервні регістри

5) 2 x 12-bit, 1 μs АЦП (підтримує до 16 каналів):

– Діапазон перетворення: від 0 до 3.6 V

– Можливість подвійного відбору та утримання

– Датчик температури

6) DMA:

– 7-канальний DMA контролер

– Підтримка периферії: таймери, АЦП, SPIs, I2Cs і USARTs

7) Підтримка до 80 швидких портів Вводу/виводу:

– 26/37/51/80 Вводу/виводу, всі відображаються на 16 зовнішніх векторах переривань і на більшості 5 V-толерантних

8) Режим відлагодження:

– Послідовна шина відлагодження (SWD) & JTAG інтерфейси

9) 7 таймерів:

– Три 16-бітних таймери, кожен підтримує до 4 IC/OC/PWM або лічильник імпульсів і квадратного (інкрементного) кодера

– 16-розрядний, ШІМ – контролер для керування двигуном з генерацією аварійної зупинки

– 2 сторожові таймери (Незалежний і Віконний)

– SysTick таймер: а 24-бітний лічильник

10) Підтримка до 9 комунікаційних інтерфейсів:

– Підтримка до 2 х I2C інтерфейсів (SMBus/PMBus)

– Підтримка до 3 USART (ISO 7816 інтерфейс, LIN, IrDA, керування модемом)

– Підтримка до 2 SPIs (18 Mbit/s)

– CAN інтерфейс (2.0B Active)

– USB 2.0 повно швидкісний інтерфейс

11) Блок обчислення CRC, 96-бітний унікальний ID

12) Пакет ECOPACK®

2.2.2 Датчик температури і вологості SHT30

Температури і відносна вологість повітря є двома з трьох параметрів, які має визначати система. Для їх вимірювання буде використовуватись модуль цифрового датчика SHT30. Він відзначається довговічністю і стабільністю результатів вимірювань, що особливо важливо при роботі в низьких температурах. В нього хороше співвідношення ціни і якості. Розглянутий датчик містить такий функціонал: схему обробки і посилення сигналу, блок пам'яті калібрування, Аналогово – цифровий перетворювач і енергозберігаючу схему живлення [10].

Рисунок 2.5 – Зовнішній вигляд датчика температури SHT30 з встановленими виводами

Датчик має 4 виводи: живлення, земля, контакт для передачі інформації про температуру і контакт для передачі інформації про вологість. Детальні параметри датчика SHT30 наведено у табл. 2.1 [4, 10].

Таблиця 2.1 – Параметри датчика SHT30

Параметри	Значення
Діапазон вимірювання температури	-40 до +120 °C
Точність діапазону температури (-40 – +90 ° C)	+/- 0.3 °C
Діапазон вимірювання відносної вологості	0 - 100 %
Точність діапазону відносної вологості (20-80 RH)	3%
Напруга живлення	2.15 - 5.5 В
Тип інтерфейсу	I2C (Підтримує частоту до 1 МГц і використання 2 адрес)
Час відгуку вимірювання вологості	8 с
Ел. потужність споживання	мінімум 5 uW (при напрузі 2.4 В)

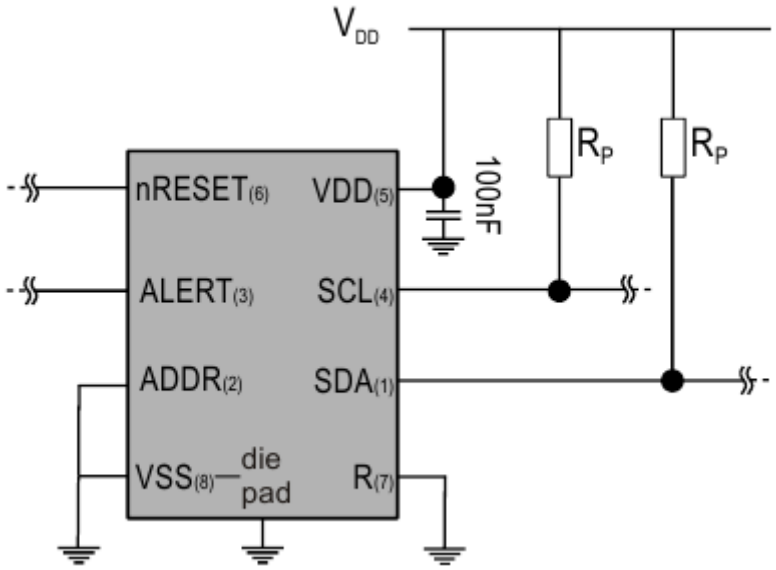

Рисунок 2.6 – Електрична принципова схема типового підключення датчика SHT30

2.2.3 Датчик тиску BMP280

Для вимірювання атмосферного тиску в холодильній установці використовується барометр BMP-280 від компанії BOSCH. Обраний датчик розроблений на основі датчика BMP180 і є його покращеною версією. Від свого попередника він відрізняється меншими розмірами, нижчим енергоспоживанням і високою точністю роботи. Датчик має точне заводського калібрування. Також датчик може користуватися двома послідовними інтерфейсами: I2C і SPI для зв’язку з мікроконтроллером.

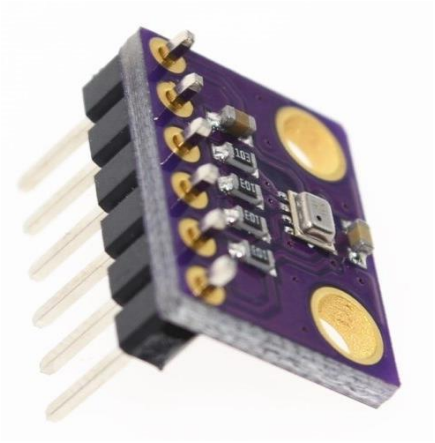

Рисунок 2.7 – Датчик BMP280 готовий до роботи

На відміну від старіших моделей (BMP085 і BMP180) датчик BMP280 підтримує три режими роботи:

– SLEEP – сплячий режим, в якому датчик споживає мінімум енергії.

– FORCED – в цьому режимі датчик прокидається коли приходить команда з мікроконтролера, дальше він проводить вимірювання, видає результати вимірювання контролера і після цього знову переходить в сплячий режим.

– NORMAL – режим самостійної роботи. Датчик виходить з режиму сну відповідно до налаштувань, проводить вимірювання параметрів і повертається в енергозберігаючий режим. Програмування всіх тимчасових параметри цього режиму здійснюється незалежно один від одного. Зчитування дані в цьому режимі можливе в будь-який час [1, 8].

У табл. 2.2 наведено детальні параметри датчика BMP280.

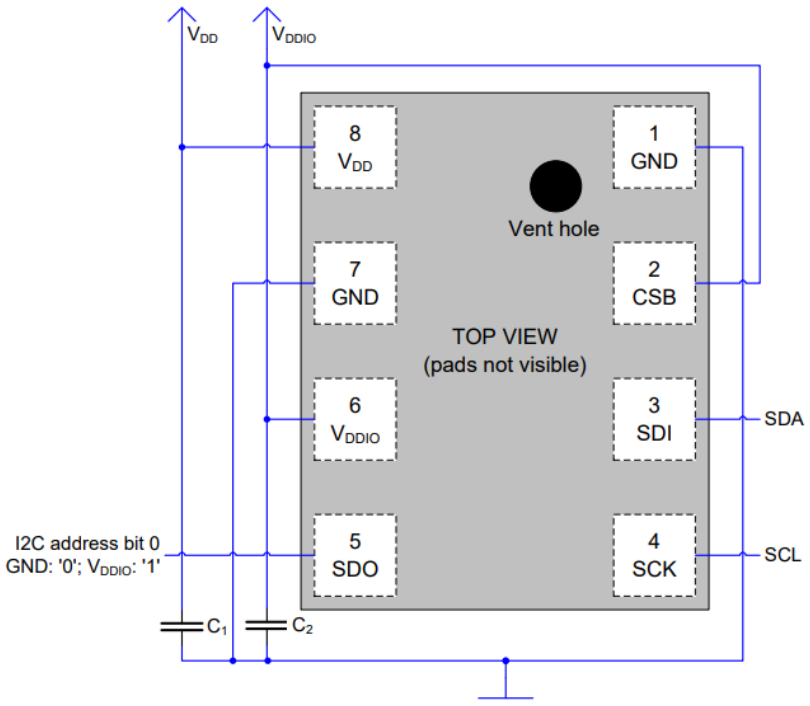

Рисунок 2.8 – Схема підключення датчика тиску по інтерфейсу I2C

Таблиця 2.2 – Детальні параметри датчика BMP280

Параметри	Значення
Напруга живлення	від 1.71 В до 3.6 В
Макс швидкість I2C інтерфейсу	3.4 МГц
Струм	2.7мкА при частоті відліків в 1 Гц
Інтерфейс	I2C, SPI (4 Провід), SPI (3 Провід)
Калібрування	заводське
Рівень шуму	до 0.2 Па (1.7 см) і 0.01 температури
Діапазон вимірюваного тиску	від 300hPa до 1100hPa
Розмір	21 мм х 18 мм

2.2.4 Екран

Екран є ще одним ключовим елементом всієї системи. В цьому проекті використовується OLED дисплей 0,96’’ від Waveshare. Перевагами цього дисплея є яскравість, економічність і висока контрастність. Це дає можливість використовувати його як при яскравому, так і при дуже поганому освітленні [2].

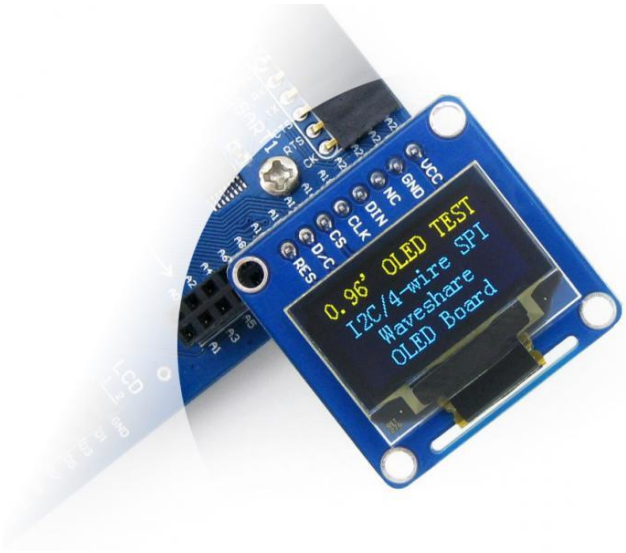

Рисунок 2.9 – Зовнішній вигляд OLED дисплею

На рис. 2.10 зображено розташування контактів дисплею.

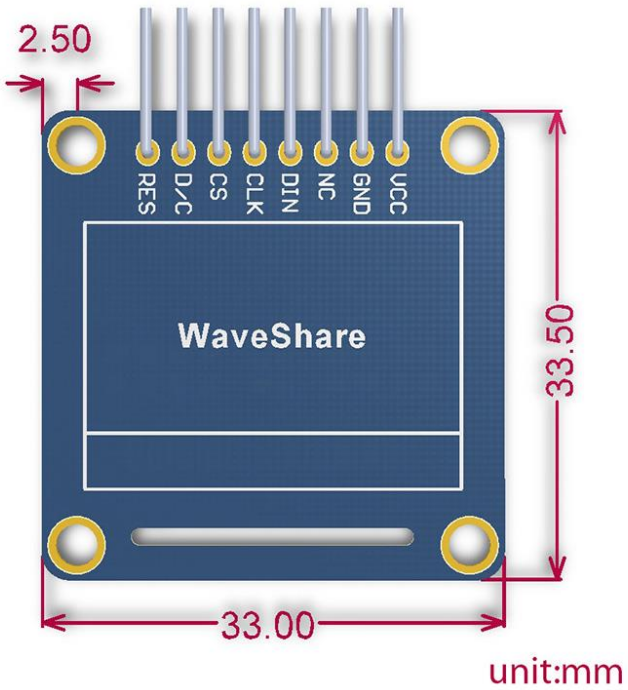

Рисунок 2.10 – Схема виводів дисплею

Детальна інформація про призначення виводів дисплею міститься у табл. 2.3.

Таблиця 2.3 – Призначення виводів дисплею

№	Назва виводу	Призначення
1	VCC	Напруга живлення (3,3 В-5В)
2	GND	Загальний
3	NC	НЕ використовується
4	DIN	Вхід даних
5	CLK	Вхід тактових імпульсів
6	CS	Вибір кристала, активний низький рівень
7	D / C	Вибір Команди / Дані. низький рівень для команд, високий для даних
8	RES	Сигнал скидання, активний низький рівень

Інформація про параметри дисплею наведена у табл. 2.4 [3].

Таблиця 2.4 – Параметри OLED дисплея

Параметри	Значення
Драйвер	SSD1306
Інтерфейс	3-дротовий SPI, 4-дротовий SPI, I2C
Робоча напруга	3,3 В
Напруга інтерфейсів	3,3 В
Роздільна здатність	128 * 64
Розмір дисплея	0.96-дюйма
Кольори	Жовтий і Синій
Кут огляду	> 160 °
Розміри	33мм * 33.50мм
Робоча температура	від 20°C до 70°C
Температура зберігання	від -30°C до 80°C

На рис. 2.11 зображено електричну принципову схему підключення OLED – дисплею. Для підключення екрану до мікроконтроллера використовується 4 – проводовий SPI. У табл. 2.5 наведено конфігурацію інтерфейсу дисплея.

Таблиця 2.5 – Конфігурація типу інтерфейсу

Інтерфейс	BS0 / BS1	DIN	SCK
3 – проводовий SPI	1 / 0	MOSI	SCLK
4 – проводовий SPI	0 / 0	MOSI	SCLK
I2C	0 / 1	SDA	SCL

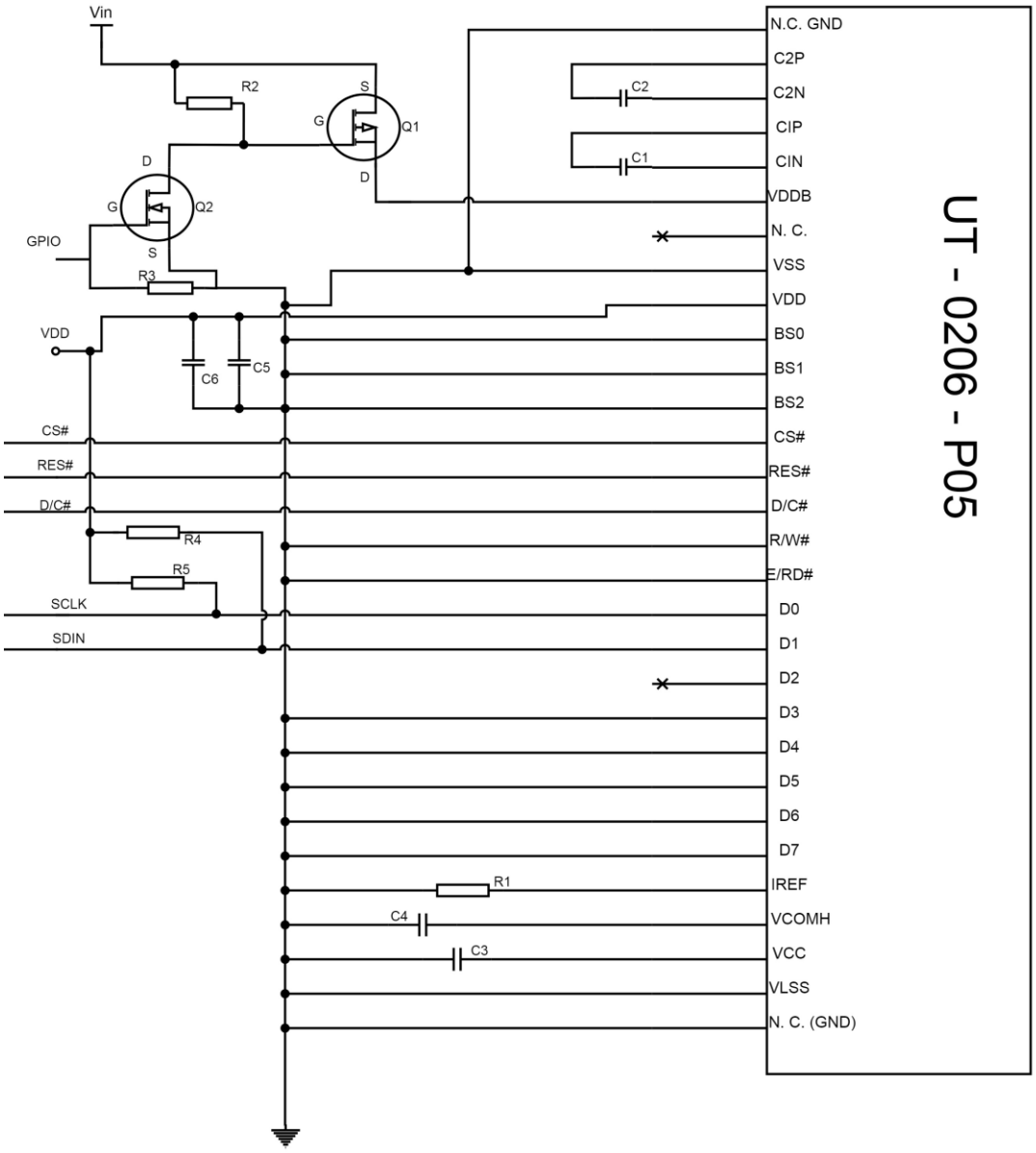

Рисунок 2.11 – Електрична принципова схема підключення OLED дисплею для використання 4 – проводового SPI

2.3 Опис шин які використовуються в комп’ютерній системі

2.3.1 I2C

I2C – послідовна шина для зв’язку інтегральних схем. Використовується для з’єднання низько швидкісних периферійних пристроїв з мікроконтролерами і процесорами. В даному випадку вона використовується для зв’язку датчиків з мікроконтроллером.

У шині I2C дані передаються по двох провідниках – провідник даних

SDA і провідник тактів SCL. В обміні інформації беруть участь мінімум 2 пристрої, один з них є Master, а другий Slave. Master встановлює тактову частоту з якою буде відбуватись обмін інформацією. На одній шині можливо встановити до 127 пристроїв. Приймання і передавання інформації здійснюється скиданням лінії в 0. Логічна 1 встановлюється завдяки підтягуючим резисторам. Вони є необхідними для роботи шини, оптимальний опір 10кОм. На рис. 2.12 зображене підключення кількох пристроїв по шині I2C.

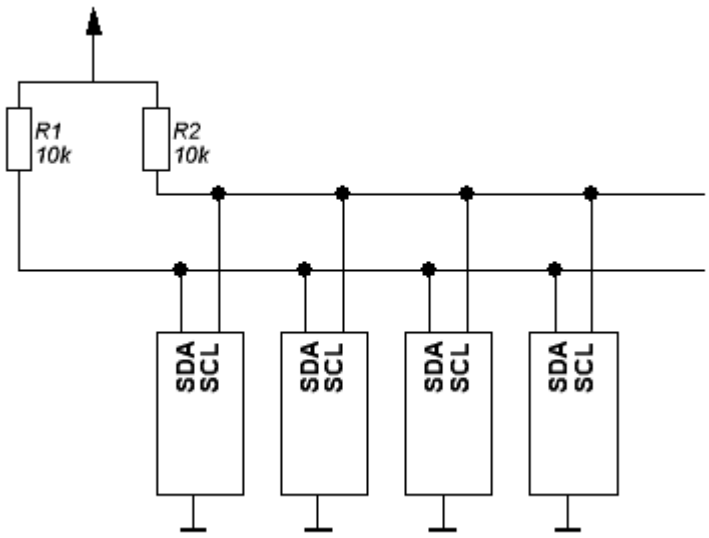

Рисунок 2.12 – Схема підключення по шині I2C

По шині інформація передається пакетами, розмір пакету 8 біт. Спочатку передається стартовий біт, який означає початок передачі даних. Дальше йде службовий пакет. В ньому вказано фізичний адрес Slave – пристрою, з яким буде проводитись обмін інформацією. Також в цьому пакеті передається біт в якому вказано напрямок передачі інформації до Master чи до Slave. Після першого пакету йдуть пакети з інформацією. В кінці обміну інформацією приходить стоповий біт. На рис. 2.13 зображено схему обміну інформацією по шині I2C [12].

Рисунок 2.13 – Обмін інформацією по шині I2C

Переваги:

– Для управління цілим набором пристроїв достатньо лише одного мікроконтролера.

– Для підключення кількох пристроїв можна використати всього два провідники.

– Можлива одночасна робота декількох ведучих пристроїв підключених до однієї шини.

– Передбачено гаряче підключення пристроїв.

Недоліки:

– Програмування контролера I2C ускладнене через велику кількість можливих позаштатних ситуацій на шині. Тому більшість систем використовують I2C з одним ведучим.

– Важкість виявлення несправності, якщо один з підключених пристроїв помилково встановлює на шині низький стан.

2.3.2 SPI

SPI – це синхронна чотирьохпроводова шина, яка може працювати в повнодуплексному або напівдуплексному режимі. Шина призначена для забезпечення простого високошвидкісного сполучення між мікроконтролером і периферією, в даному випадку між мікроконтроллером і дисплеєм. Для з’єднання пристроїв використовують конфігурацію Master / Slave. Інформація в шині передається по синхроімпульсах, які генерує Master. В схемі може бути

лише один Master. Для передачі інформації вихід Master підключається до входу Slave, а вихід Slave до входу Master. Шина SPI завжди працює в повно дуплексному режимі. На рис. 2.14 показана схема підключення SPI [15].

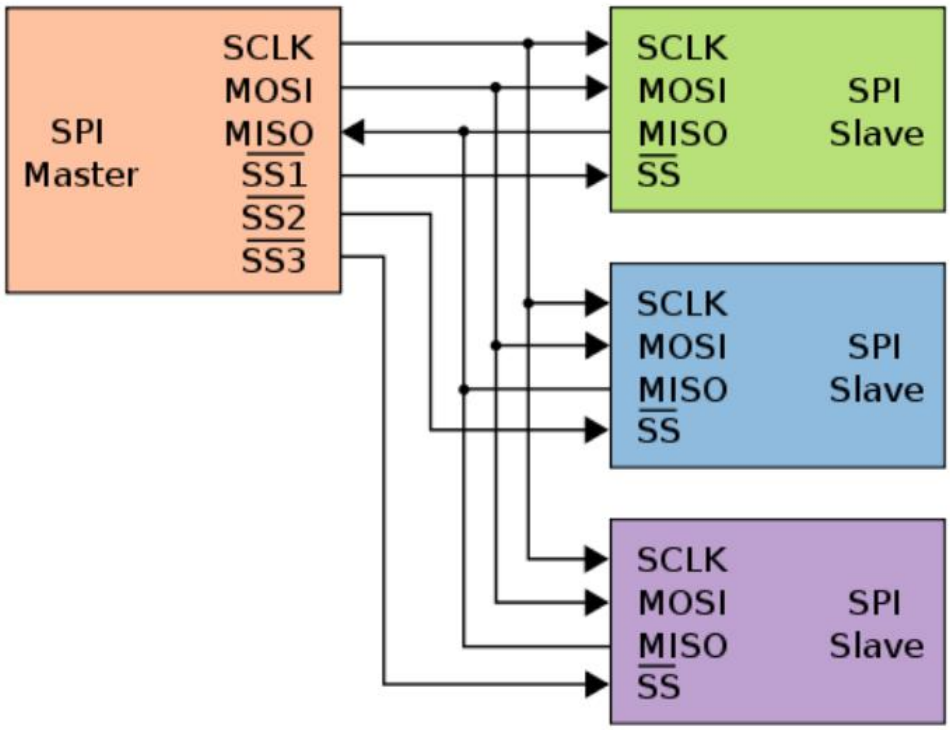

Рисунок 2.14 – Схема підключення SPI

Для обміну інформацією використовуються такі сигнали:

– MOSI — Master Output, Slave Input. Передача даних від Master до Slave.

– MISO — Master Input, Slave Output . Передача даних від Slave до Master.

– SCK — Serial Clock. Цей сигнал використовується для синхронізації пристроїв при передачі даних.

– ~CS — Chip Select. За допомогою цього сигналу здійснюється активація Slave перистрою

Переваги:

– Повнодуплексна передача даних.

– Вища пропускна здатність порівняно з I2C або SMBus.

– Довжина пакету не обмежена 8 бітами.

– Простота апаратної реалізації:

– Використання лише 4 виводів.

– Не має обмеження на максимальну тактову частоту, вона залежить лише від швидкості пристроїв, які обмінюються інформацією.

Недоліки:

– Порівняно з I2C SPI потребує більше виводів для встановлення зв’язку.

– Не існує визначеного стандартом протоколу для знаходження помилок.

– Менша дальність передачі даних порівняно з інтерфейсами CAN і UART.

– Не підтримує гаряче підключення пристроїв.

2.4 Обґрунтування вибору програмного забезпечення проектованого комп’ютерного засобу

Для налаштування і програмування мікроконтролера використовуються середовища STM32CubeMX і CoIDE (CooCox).

STM32CubeMX – це безкоштовний графічний інструмент розроблений компанією STMicroelectronics. З його допомогою можна легко конфігурувати мікроконтролери STM32 і генерувати відповідний код ініціалізації [5].

Переваги:

– Інтуїтивно зрозумілий вибір мікроконтролера чи мікропроцесора STM32.

– Графічний інтерфейс, який є простим у використанні, він дозволяє встановлювати такі налаштування :

1) Виводи контактів з автоматичним вирішенням конфліктів.

2) Режими роботи периферії і проміжного програмного забезпечення з динамічною перевіркою обмежень параметрів для ядра Arm ® Cortex ® -M.

3) Налаштування параметрів годинника з динамічною перевіркою конфігурації.

4) Налаштування параметрів живлення і розрахунок приблизних результатів споживання.

– Створення С - кодів ініціалізації , сумісних з різними середовищами розробки для ядра Arm ® Cortex ® -M

– Створення нових покращених пакетів розширення STM32Cube використовуючи STM32PackCreator

– Наявність в якості автономного програмного забезпечення , що працює на ОС Windows ® , Linux ® і MacOS ®.

CoIDE (CooCox) – це безкоштовне середовище для розробки програмного забезпечення створене на основі ланцюга інструментів Eclipse і GCC. Середовище було налаштоване і спрощене для того, щоб надати користувачеві легкий доступ до мікроконтролерів ARM® Cortex®-M.

Переваги:

– Повна підтримка мікроконтролерів STM32, плат STM32 Nucleo, а також програмних бібліотек STM32Cube.

– Компілятор GCC C/C++.

– Налагоджувач на основі GDB.

– Спрощена IDE Eclipse.

– Підтримка ST-Link.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Моделювання проектних рішень комп’ютерної системи моніторингу стану повітря для холодильної установки

Після запуску пристрою відбувається ініціалізація бібліотек і змінних необхідних для роботи програми. Дальше пристрій зчитує дані з датчиків температури, вологості і тиску і зберігає інформацію у відповідні змінні. Після цього відбувається порівняння зчитаних даних із мінімальним і максимальним значенням параметрів для кожного датчика, ці значення задаються користувачем в коді програми. Якщо якесь з виміряних значень менше мінімального заданого, то на відповідному RGB – світлодіоді засвічується синій колір. Якщо якесь із значень більше за максимальне задане, то світлодіод стає червоним, а якщо всі три параметри в межах норми – світлодіод зелений. Керування дисплеєм здійснюється за допомогою кнопки. Кожного разу коли натискають на кнопку, значення змінної Nomer_dat, яке спочатку рівне 1, збільшується на 1. Програма перевіряє чи Nomer_dat більше 3. Якщо так, то Nomer_dat присвоюється 1. Дальше якщо Nomer_dat дорівнює 1, то на дисплеї відображається інформація з першого комплекту датчиків, якщо 2 – то з другого комплекту, якщо 3 – відображаються середні арифметичні значення по всіх датчиках . Після цього виконання програми закінчується. На рис. 3.1 зображено алгоритм роботи програми.

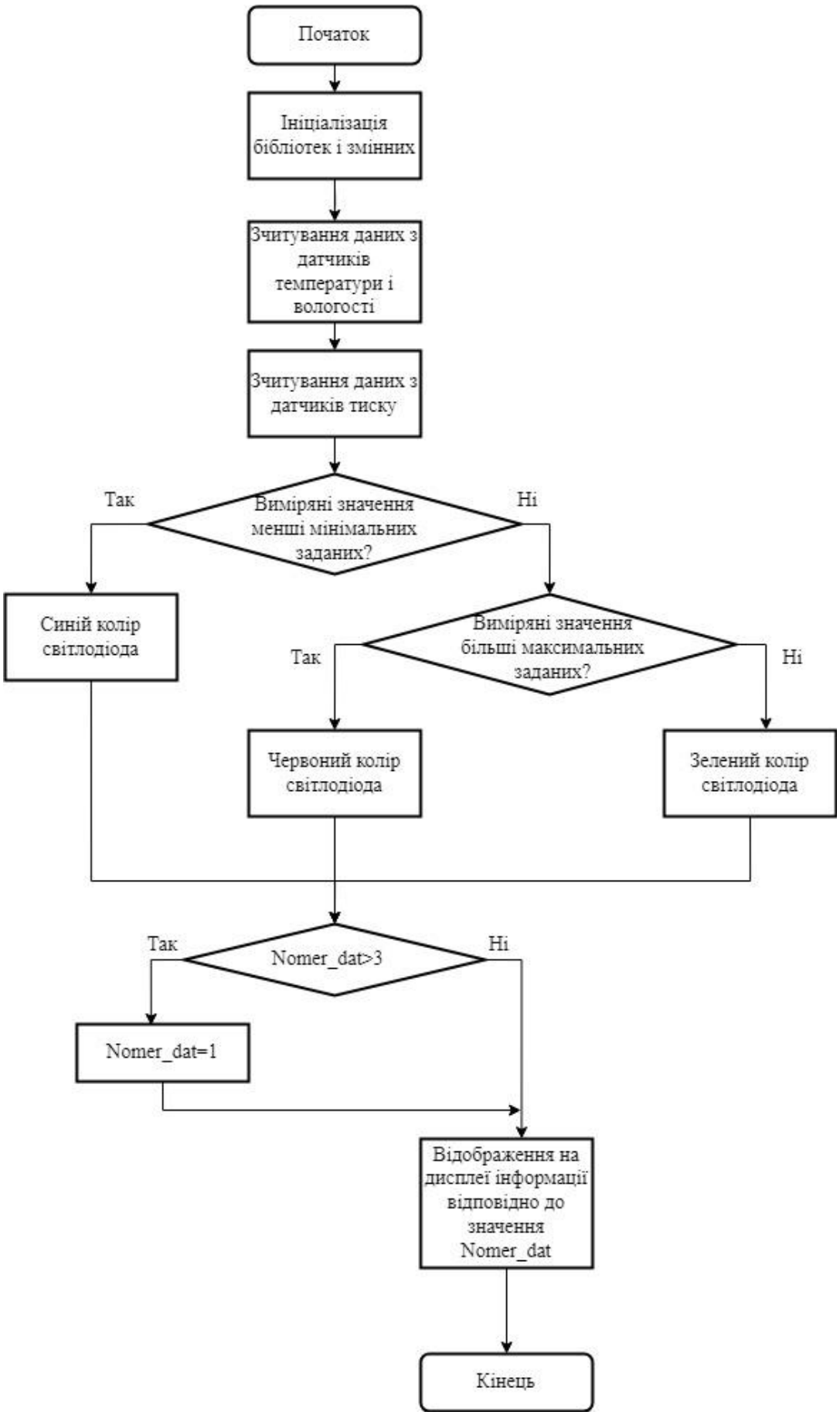

Рисунок 3.1 – Блок – схема алгоритму роботи програми

Для налаштування мікроконтроллера використовується графічне середовище STM32 CubeMX. Воно надає можливість легко і швидко робити базові налаштування мікроконтроллера. На рис. 3.2 зображено налаштування виводів мікроконтролера.

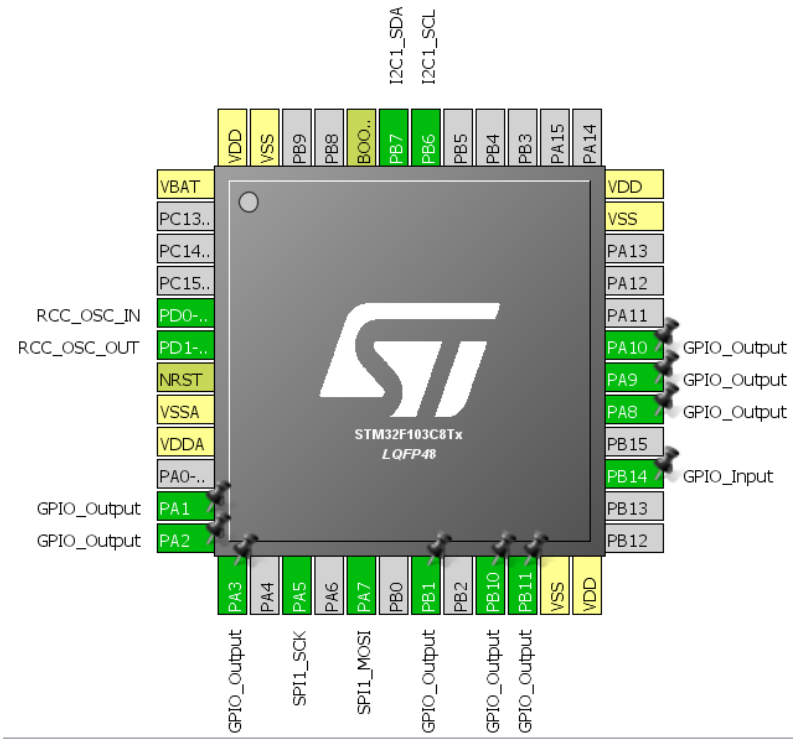

Рисунок 3.2 – Налаштування виводів мікроконтроллера

Для передачі даних від датчиків до контролера використовується шина I2C. Налаштування цієї шини зображені на рис. 3.3.

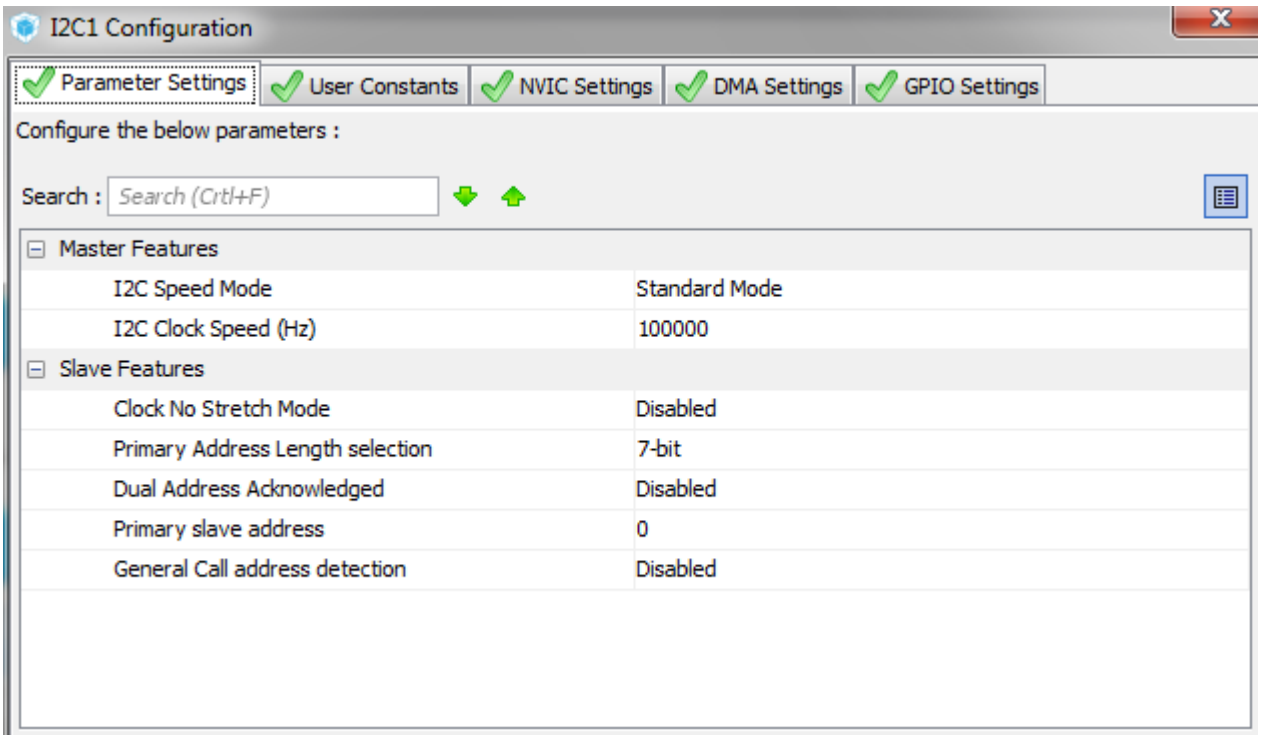

Рисунок 3.3 – Налаштування шини I2C

Для передачі даних від мікроконтролера до OLED дисплею використовується шина SPI. В розроблюваній системі буде використовуватись напівдуплексний режим роботи, оскільки дисплей лише приймає інформацію і нічого не передає. На рис. 3.4 зображено налаштування SPI.

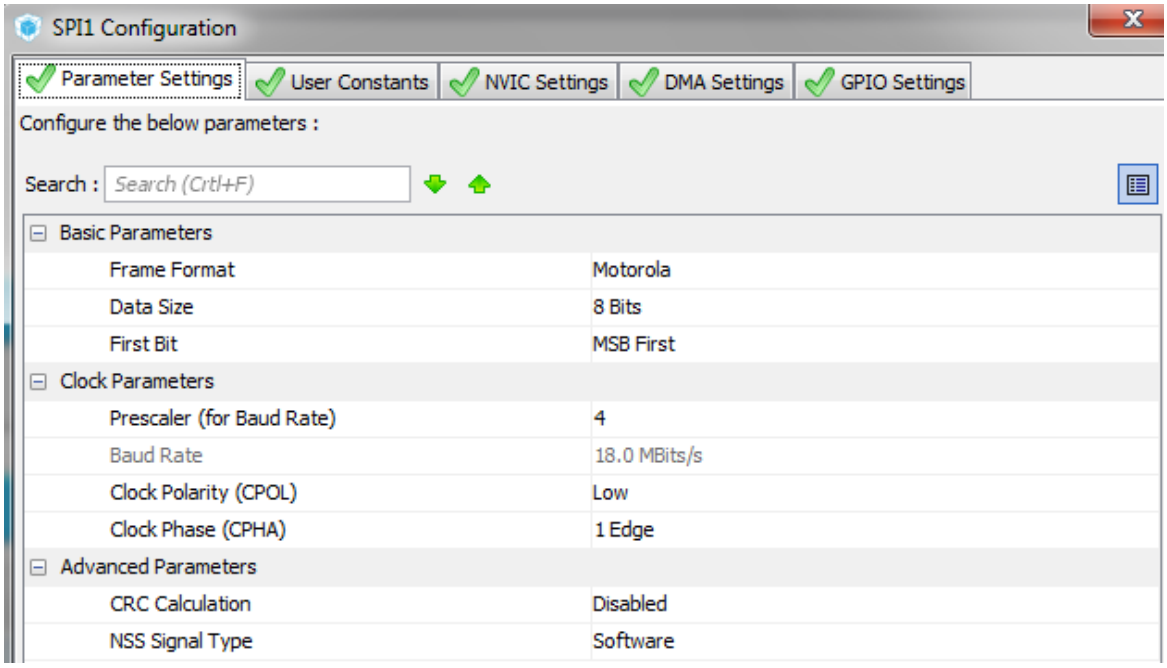

Рисунок 3.4 – Налаштування шини SPI

У середовищі STM32 CubeMX є можливість налаштувати генератор тактової частоти. В цьому проекті для максимальної швидкодії він налаштований на 72 МГц. На рис. 3.5 зображено налаштування тактового генератора.

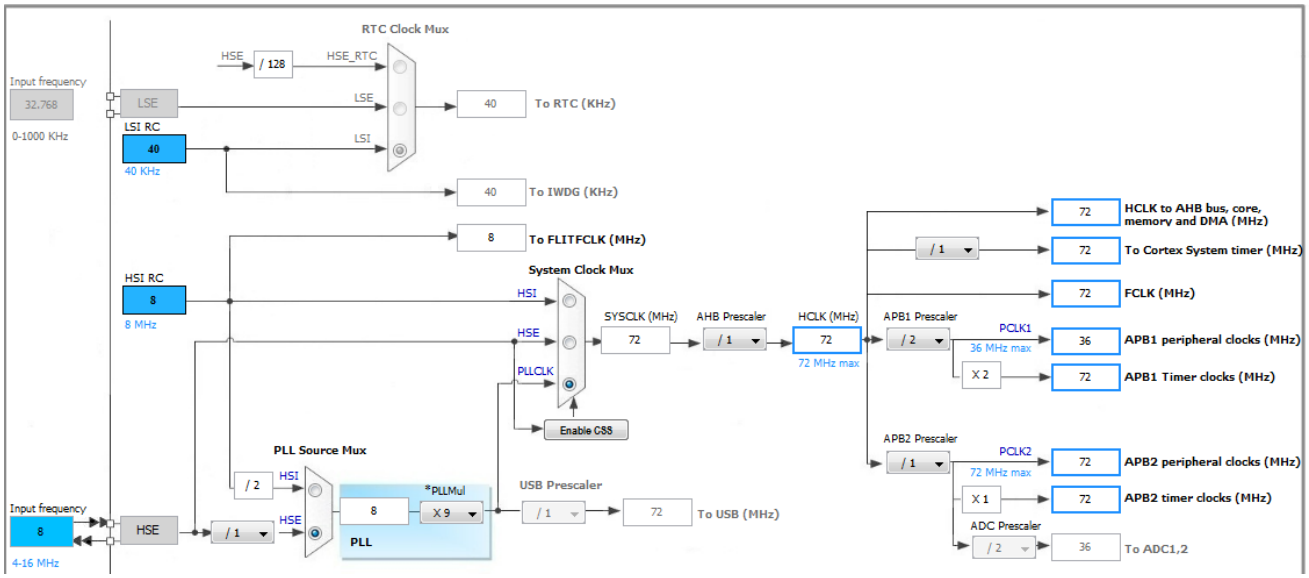

Рисунок 3.5 – Налаштування тактового генератора

Середовище STM32 CubeMX дає можливість генерувати початковий код з налаштуванням виводів мікроконтроллера. Для написання і редагування коду буде використовуватись середовище CooCox (CoIDE). На рис. 3.6 зображено згенерований код для роботи з шинами передачі даних.

```c
/* I2C1 init function */
static void MX_I2C1_Init(void)
{

  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
```

```c
/* SPI1 init function */
static void MX_SPI1_Init(void)
{

  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_1LINE;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_4;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
```

Рисунок 3.6 – Код для налаштування шини I2C і шини SPI

Для отримання значень температури і вологості використовуються датчики SHT30. Вони зв’язуються з мікроконтроллером по шині I2C. Для того щоб контролер знав з яким датчиком він має справу, датчики мають адреси. Датчики SHT30 мають дві стандартні адреси, які будуть використовуватись в залежності від того яке значення подається на вхід ADDR. На рис. 3.7 і рис. 3.8 зображено код для визначення температури і вологості [17].

```c

/*Глобальні змінні для роботи з датчиком температури і вологості*/
/*ПОЧАТОК*/
uint8_t Sensor1_Data[6]; // масив з 6 байт, 2-передане значення температури 1 датчика + 1-CRC + 2 - значення вологості 1 датчика + 1-CRC
uint8_t Sensor2_Data[6]; // масив з 6 байт, 2-передане значення температури 2 датчика + 1-CRC + 2 - значення вологості 2 датчика + 1-CRC
uint16_t Temperature1_RAW; // необроблене значення температури з датчика 1
uint16_t Temperature2_RAW; // необроблене значення температури з датчика 2
uint16_t Humidity1_RAW; // необроблене значення вологості з датчика 1
uint16_t Humidity2_RAW; // необроблене значення вологості з датчика 2
float Temperature1; // оброблене значення температури 1
float Temperature2; // оброблене значення температури 2
float Humidity1; // оброблене значення вологості 1
float Humidity2; // оброблене значення вологості 2
uint16_t Measure = 0x240B; // команда для зчитування інформації з датчика
#define Sensor1_Adress (0x44) //адреса першого датчика температури і вологості
#define Sensor2_Adress (0x45) //адреса другого датчика температури і вологості
/*КІНЕЦЬ*/

```

Рисунок 3.7 – Змінні необхідні для роботи з датчиками температури

```
/*************Отримування температури і вологості з датчика 1**********/
HAL_I2C_Mem_Read_IT(&hi2c1, Sensor1_Adress, Measure, I2C_MEMADD_SIZE_8BIT, (uint8_t*) Sensor1_Data[6];, 5); // Пере
Temperature1_RAW = ((uint16_t)(Sensor1_Data[0] << 8) | (Sensor1_Data[1])); // Об'єднання 2 елементів масиву, які мі
Temperature1 = (float)(Temperature1_RAW * 175 / 65535.00) - 45; // Перетворення значення отриманого з датчика по фо
Humidity1_RAW = ((uint16_t)(Sensor1_Data[3] << 8) | (Sensor1_Data[4])); // Об'єднання 2 елементів масиву, які містя
Humidity1 = (float)(Humidity1_RAW/65535)*100; // Перетворення значення отриманого з датчика по формулі в відсотки.
HAL_Delay(100);

/**********Отримування температури і вологості з датчика 2*************/
HAL_I2C_Mem_Read_IT(&hi2c1, Sensor2_Adress, Measure, I2C_MEMADD_SIZE_8BIT, (uint8_t*) Sensor2_Data[6];, 5); // Пере
Temperature2_RAW = ((uint16_t)(Sensor2_Data[0] << 8) | (Sensor2_Data[1])); // Об'єднання 2 елементів масиву, які мі
Temperature2 = (float)(Temperature2_RAW * 175 / 65535.00) - 45; // Перетворення значення отриманого з датчика по фо
Humidity2_RAW = ((uint16_t)(Sensor2_Data[3] << 8) | (Sensor2_Data[4])); // Об'єднання 2 елементів масиву, які містя
Humidity2 = (float)(Humidity2_RAW/65535)*100; // Перетворення значення отриманого з датчика по формулі в відсотки.
HAL_Delay(100);
```

Рисунок 3.8 – Код для роботи з датчиками температури

В змінних прописано масиви в яких будуть зберігатись байти інформації передані з датчиків. В змінній Temperature1_RAW і Temperature2_RAW зберігається не оброблене значення температури. В змінних Temperature1 і Temperature2 значення температури в градусах Цельсія. Humidity1_RAW і Humidity2_RAW – це змінні для сирих значень вологості на першому і другому датчиках. Оброблені значення вологості зберігаються в змінних Humidity1 і Humidity2. Змінні Sensor1_Adress і Sensor2_Adress містять відповідно адреси 1 і 2 датчиків. Measure містить код команди для зчитування даних.

Для знаходження значення атмосферного тиску в холодильній установці використовується датчик BMP 280, він також використовує для підключення шину I2C. Для роботи з датчиками задаються глобальні змінні, так само, як при роботі з SHT30. На рис. 3.9 і рис. 3.10 наведено код для роботи з датчиком тиску [17].

```
42 /*Глобальні змінні для роботи з датчиком тиску*/
43 /*ПОЧАТОК*/
44 uint16_t Pressure1_RAW; // необроблене значення тиску з датчика 1
45 uint16_t Pressure2_RAW; // необроблене значення тиску з датчика 2
46 float Pressure1; // оброблене значення тиску 1
47 float Pressure2; // оброблене значення тиску 2
48 #define ID_REG (0xD0)
49 #define PRESS_MSB_REG (0xF2)
50 #define Pressure1_Adress (0x76) //адреса першого датчика тиску
51 #define Pressure2_Adress (0x77) //адреса другого датчика тиску
52 /*КІНЕЦЬ*/
```

Рисунок 3.9 – Глобальні змінні для роботи з датчиком тиску

```
184     /*************Отримування 1 значення тиску**********/
185     HAL_I2C_Mem_Read( & hi2c1,  Pressure1_Adress,  ID_REG,  1 ,  & chipID,  1 ,  1000 );
186     HAL_I2C_Mem_Read(BME280_I2C,  Pressure1_Adress,  PRESS_MSB_REG,  1 ,  RawData,  8 ,  HAL_MAX_DELAY);
187     Pressure1_RAW  =  (RawData[ 0 ] << 12 ) | (RawData[ 1 ] << 4 ) | (RawData[ 2 ] >> 4 );
188     if  (Pressure1_RAW  ==  0x800000 )  Pressure1  =  0 ;  // значення у випадку, якщо вимірювання температури було вимкнено
189           else
190           {
191             if (SUPPORT_64BIT)
192             {
193             Pressure1  =  (BME280_compensate_P_int64  (Pressure1_RAW)) / 256.0 ;   // згідно з даними, тиск становить x256
194             }
195             if (SUPPORT_32BIT)
196             {
197             Pressure1  =  (BME280_compensate_P_int32  (Pressure1_RAW));   // згідно з даними, тиск дорівнює Pa
198             }
199           }
200
201     /*************Отримування 2 значення тиску**********/
202         HAL_I2C_Mem_Read( & hi2c1,  Pressure2_Adress,  ID_REG,  1 ,  & chipID,  1 ,  1000 );
203         HAL_I2C_Mem_Read(BME280_I2C,  Pressure2_Adress,  PRESS_MSB_REG,  1 ,  RawData,  8 ,  HAL_MAX_DELAY);
204         Pressure1_RAW  =  (RawData[ 0 ] << 12 ) | (RawData[ 1 ] << 4 ) | (RawData[ 2 ] >> 4 );
205         if  (Pressure1_RAW  ==  0x800000 )  Pressure2  =  0 ;  // значення у випадку, якщо вимірювання температури було вимкнено
206               else
207               {
208                 if (SUPPORT_64BIT)
209                 {
210                 Pressure2  =  (BME280_compensate_P_int64  (Pressure2_RAW)) / 256.0 ;   // згідно з даними, тиск становить x256
211                 }
212                 if (SUPPORT_32BIT)
213                 {
214                 Pressure2  =  (BME280_compensate_P_int32  (Pressure2_RAW));   // згідно з даними, тиск дорівнює Pa
215                 }
216               }
217
```

Рисунок 3.10 – Код для обчислення тиску

Змінні Pressure1_adress і Pressure2_adress містять адреси датчиків, для BMP 280 є дві стандартні адреси 0x76 і 0x77. Значення адреси на датчику залежить від способу його підключення. ID_REG і PRESS_MSB_REG зберігають в собі адреси службових регістрів, необхідних для зчитування даних. Змінні Pressure1_RAW і Pressure1містя відповідно сире і оброблене значення атмосферного тиску на датчику 1, аналогічні змінні використовуються для датчика 2.

Після того як з інформація з датчиків буде зчитана, вона порівнюється з мінімальними і максимальними параметрами, які користувач вказав в коді. Якщо виміряні значення знаходиться в заданому діапазоні, то RGB – світлодіод світиться зеленим, якщо нижче – синім, а якщо вище – червоним. На рис. 3.11 і рис. 3.12 зображено код для роботи з світлодіодами.

```
/*Значення параметрів встановлені користувачем*/
/*ПОЧАТОК*/
float User_Temperature1_Min = -10.00; // Мінімальне зачення температури на 1 датчику
float User_Temperature1_Max = -8.00; // Максимальне значення температури на 1 датчику
float User_Humidity1_Min = 12.00; // Мінімальне зачення вологості на 1 датчику
float User_Humidity1_Max = 25.00; // Максимальне значення вологості на 1 датчику

float User_Temperature2_Min = -12.00;// Мінімальне зачення температури на 2 датчику
float User_Temperature2_Max = 0.00;// Максимальне значення температури на 2 датчику
float User_Humidity2_Min = 12.00; // Мінімальне зачення вологості на 2 датчику
float User_Humidity2_Max = 25.00; // Максимальне значення вологості на 2 датчику

float User_Pressure1_Min = 735.00;// Мінімальне зачення температури на 2 датчику
float User_Pressure1_Max = 760.00;// Максимальне значення температури на 2 датчику
float User_Pressure2_Min = 750.00; // Мінімальне зачення вологості на 2 датчику
float User_Pressure2_Max = 770.00; // Максимальне значення вологості на 2 датчику
```

Рисунок 3.11 – Встановлення користувачем значень різних параметрів

```
/***********Індикація 1 групи датчиків***********/
if(Temperature1 < User_Temperature1_Min || Humidity1 < User_Humidity1_Min || Pressure1 < User_Pressure1_Min)
{
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET); // синій
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET); // зелений
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET); // червоний
}

if(Temperature1 > User_Temperature1_Max || Humidity1 > User_Humidity1_Max || Pressure1 > User_Pressure1_Max)
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET); // синій
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET); // зелений
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_SET); // червоний
    }
else // в іншому випадку (всі значення в межах норми) світиться зелений колір
{
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET); // синій
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET); // зелений
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET); // червоний
}
```

Рисунок 3.12 – Код для роботи RGB – світлодіода

Для забезпечення високої точності вимірювання система передбачає встановлення кількох однакових датчиків в одному приміщенні. В такому випадку передбачено обрахунок середнього арифметичного значення кожного параметру. На рис. 3.13 наведено код для виведення середнього арифметичного значення.

```
/********Знаходження середніх арифметичних значень параметрвів*********/
Temperature_Sered = (Temperature1 + Temperature2)/2;
Humidity_Sered = (Humidity1 + Humidity2)/2;
Pressure_Sered = (Pressure1 + Pressure2)/2;
```

Рисунок 3.13 – обчислення середній значень

Другою частиною системи без якої неможлива її робота є OLED – дисплей. Він працює на драйвері SSD1306, для зв'язку з мікроконтроллером використовується шина SPI [6, 18].

Для легкої роботи потрібно встановити змінні, тобто присвоїти виводам імена див. рис. 3.14.

```
/*Глобальні змінні для роботи з екраном*/
/*ПОЧАТОК*/
u8 OLED_GRAM[128][8];// масив для роботи з кешом
#define OLED_SCLK_Clr()  PAout(5)=0   //SCK
#define OLED_SCLK_Set()  PAout(5)=1   //SCK
#define OLED_SDIN_Clr()  PAout(7)=0   //SDA
#define OLED_SDIN_Set()  PAout(7)=1   //SDA
#define OLED_RES_Clr()   PAout(1)=0   //RST
#define OLED_RES_Set()   PAout(1)=1   //RST
#define OLED_DC_Clr()    PAout(3)=0    //DC
#define OLED_DC_Set()    PAout(3)=1    //DC
#define OLED_CMD  0 // Запис команди
#define OLED_DATA 1 // Запис даних
/*КІНЕЦЬ*/
```

Рисунок 3.14 – Змінні для роботи з екраном

Оскільки дисплей може виконувати багато простих дій, наприклад запуститись, виключитись, встановити курсор на початок і т.д., тому всі ці дій відображені в окремих функціях див. рис. 3.15.

```
360 void OLED_Display_On(void) // Включення дисплею
361 {
362     OLED_WR_Byte(0X8D,OLED_CMD);
363     OLED_WR_Byte(0X14,OLED_CMD);
364     OLED_WR_Byte(0XAF,OLED_CMD);
365 }
366
367 void OLED_Display_Off(void)// Виключення дисплею
368 {
369     OLED_WR_Byte(0X8D,OLED_CMD);
370     OLED_WR_Byte(0X10,OLED_CMD);
371     OLED_WR_Byte(0XAE,OLED_CMD);
372 }
373
374 void OLED_Clear(void)  //очистка дисплею
375 {
376     u8 i,n;
377     for(i=0;i<8;i++)for(n=0;n<128;n++)OLED_GRAM[n][i]=0X00;
378     OLED_Refresh_Gram();// Оновити відображення
379 }
380
381 void OLED_DrawPoint(u8 x,u8 y,u8 t) // відображення точок
382 {
383     u8 pos,bx,temp=0;
384     if(x>127||y>63)return;
385     pos=7-y/8;
386     bx=y%8;
387     temp=1<<(7-bx);
388     if(t)OLED_GRAM[x][pos]|=temp;
389     else OLED_GRAM[x][pos]&=~temp;
390 }
```

Рисунок 3.15. – Функції для роботи з дисплеєм

Основна робота з дисплеєм виконується у основній функції main() див. рис. 3.16.

```
if (Nomer_dat=1)
{
    OLED_Display_On();
    OLED_ShowString(50,0,"Temper_1");
    OLED_ShowNumber(0,15,Temperature1,4);
    OLED_ShowString(50,0,"Hum_1");
    OLED_ShowNumber(0,15,Humidity1,4);
    OLED_ShowString(50,0,"Press_1");
    OLED_ShowNumber(0,15,Pressure1,4);
    OLED_Refresh_Gram();
    delay_ms(50);
}
```

Рисунок 3.16. – Код для роботи дисплею в функції main

Аналогічно виводиться інформація для 2 комплекту датчиків і середніх значень.

3.2 Тестування

Компіляція коду здійснювалась у середовищі CooCox (CoIDE). Для компіляції використовувався компілятор gcc-arm-none-eabi-5_4. Це набір інструментів з відкритим вихідним кодом для програмування на C, C++ та асемблері. Компілятор розроблений спеціально для роботи з 32-розрядними сімействами процесорів Arm Cortex-A, Arm Cortex-M і Arm Cortex-R. Компілятор доступний для операційних систем Windows, Linux і Mac OS X і його можна безкоштовно завантажити з сайту розробника.

Для завантаження прошивки на плату можна використовувати ST-Link програматор і спеціальну програму ST-Link Utility. ST-Link Utility – повнофункціональний програмний інтерфейс, створений для програмування мікроконтролерів STM32. Середовище ST-Link Utility є ефективним і простим у використанні. Воно надає доступ для читання, запису та перевірки пам'яті пристрою. На рис. 3.17. зображено інтерфейс ST-Link Utility.

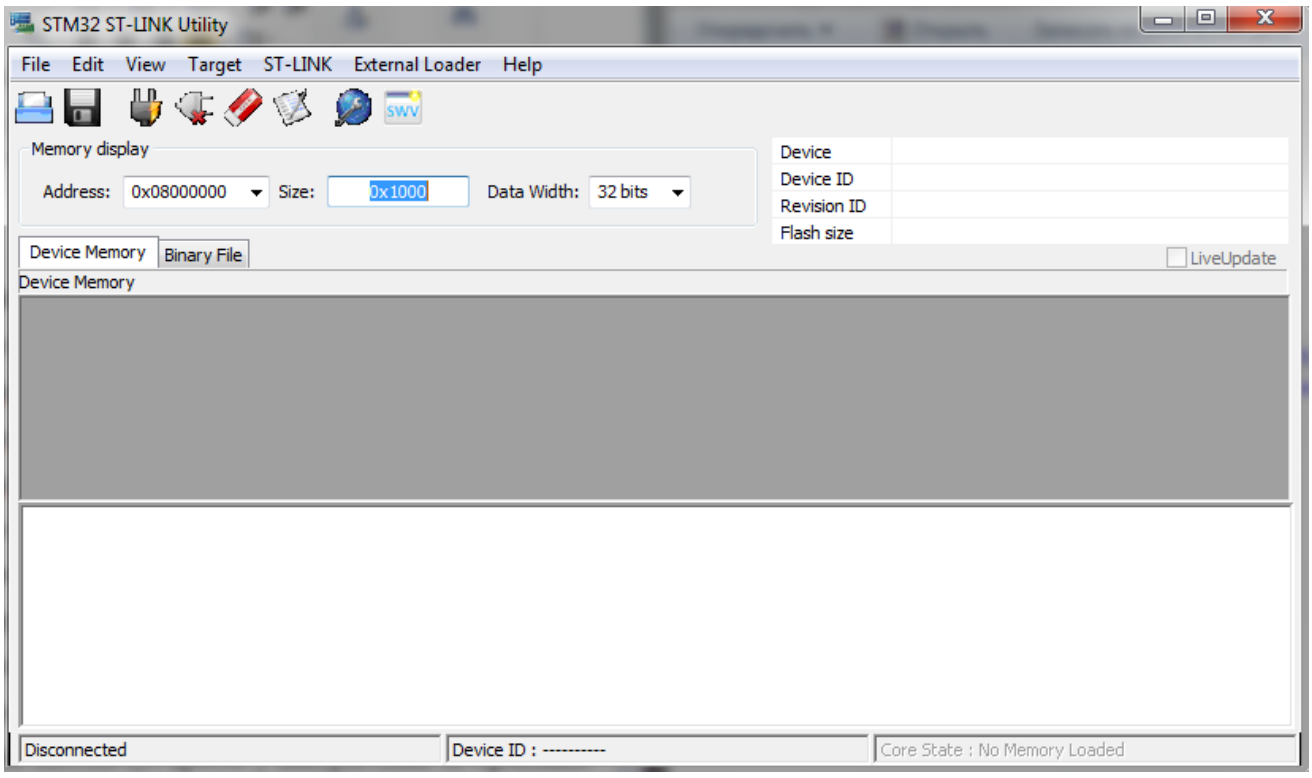

Рисунок 3.17 – Вікно програми ST-Link Utility

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при ураженні електричним струмом

Розроблювана комп'ютерна система живиться від мережі 220 Вольт, таким чином в разі поломки, чи при неправильному користуванні можливе ураження працівника електричним струмом. В разі виникнення такої ситуації потрібно виконати наступні дії [11]:

1) Розірвати контакт потерпілого з провідником струму.

– Підходити до потерпілого можна лише ізолювавши ноги від струму (вдягнути гумове взуття, покласти на підлогу матеріал, який не проводить струм).

– Розірвати зв'язок потерпілого з провідником струму. Для цього потрібно відключити живлення на приладі, або, в разі напруги до 1000 В, можна фізично відштовхнути потерпілого за допомогою предметів, що не проводять струм.

– Відтягнути потерпілого від джерела струму за допомогою предметів що не проводять струм на відстань не менше 10 м.

2) Перевірити стан дихальної та серцево судинної системи.

– Перевірити наявність дихання і пульсу у постраждалого.

– Звільнити прохідність дихальних шляхів. Для цього потрібно підняти постраждалому підборіддя, відтягнути нижню щелепу і закинути голову назад. Якщо виникає підозра на перелом хребта, то ці дії робити заборонено.

3) Якщо у постраждалого відсутні дихання і пульс, йому потрібно надати первинну допомогу.

– Непрямий масаж серця. Постраждалого потрібно положити на спину на рівну поверхню. Рятівник натискає по середині грудної клітки з частотою 60 натискань за 1 хвилину. При натисканнях грудна клітка має прогинатись на 5-6 сантиметрів див. рис.4.1.

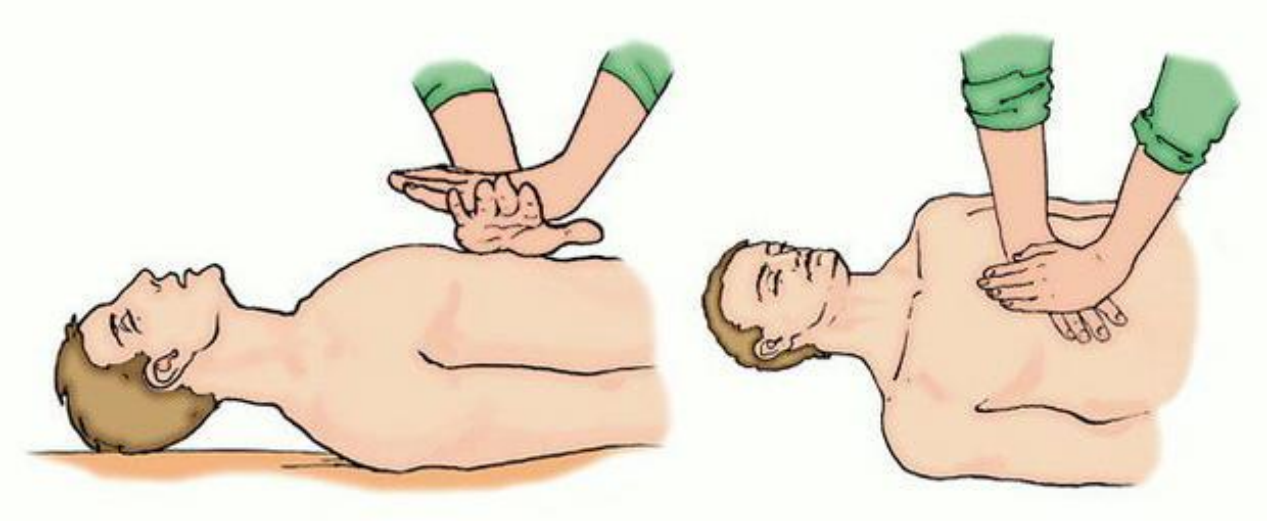

Рисунок 4.1 – Непрямий масаж серця

– Дихання рот в рот. Через кожні 12-15 натискань на грудну клітку здійснюється 2 повних видихи. При неможливості даного способу допустимо використовувати лише непрямий масаж серця див.рис.4.2.

Рисунок 4.2 – Проведення штучного дихання рот в рот

– Реанімаційні заходи проводять до приїзду швидкої допомоги або до появи ознак життя (шкіра набула рожевого кольору, з'явилися дихання і пульс). Постраждалого в якого з'явились ознаки життя потрібно перевернути на бік і в такому положенні чекати приїзду швидкої. Реанімаційні заходи потрібно проводити не більше 30 хв.

– Медикаментозне лікування (проводиться реанімаційною бригадою швидкої). У випадку, якщо наведені вище заходи не дали результату, то протягом 2-3 хвилин вводиться 1 мл адреналіну.

– Первинна обробка опіків. Якщо в наслідок ураження струму на тілі постраждалого з'являються опіки, то на них потрібно накласти суху марлеву пов'язку.

– Знеболюючі. Якщо постраждалий при свідомості то до приїзду швидкої йому можна дати знеболювальне і заспокійливе.

– Транспортування потерпілого потрібно здійснюватив лежачому положенні.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК, агрегату

Розроблювана система являє собою сукупність датчиків під'єднаних до пристроїв виведення (OLED – дисплей і світлодіодна індикація). Зміни в систему вносяться програмно за допомогою ПК. Для того щоб уникнути погіршення здоров'я при роботі з ПК потрібно дотримуватись таких правил техніки безпеки [9]:

1) Площа одного робочого місця має бути не менше 6$м^2$. Робоче місце потрібно розташовувати таким чином, щоб в поле зору працівника не потрапляли вікна чи освітлювальні прилади. Поверхня робочого столу не повинна бути полірованою і блискучою. Світло на монітор має падати зліва.

2) Монітор повинен знаходитись на відстані 500 – 700мм від користувача, під кутом 10 – 40 градусів.

3) ПК повинен стояти на відстані не менше 1м. до найближчого джерела тепла

4) Висота поверхні робочого столу має коливатись в межах 680-800 мм.

5) На робочому місці має бути достатній рівень освітленості не менше 400 лк у горизонтальній площині на висоті 0,8 м. від рівня підлоги і не більше 200 лк вертикальної освітленості . Освітлення має бути рівномірним див.рис.4.3.

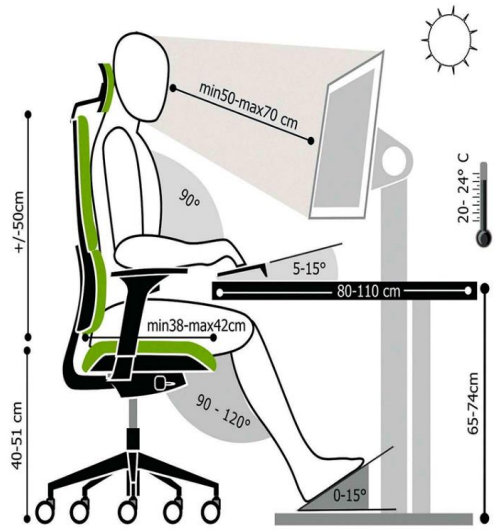

Рисунок 4.3 – Правильне розтушування людини за робочим столом при роботі з ПК

6) Температура навколишнього середовища повинна бути в межах 21 - 24 градуси С в холодну пору і 22 – 25 градусів С в теплу, відносна вологість повітря 40-60%.

7) Перед початком роботи працівник повинен оглянути ПК. Перевірити цілісність корпусу, монітора, кабелів живлення і їх підключень.

8) При виявленні ушкоджень працівник повинен попередити свого безпосереднього керівника.

9) Під час роботи потрібно забезпечити на робочому столі простір для переміщення клавіатури і миші.

10) Щоб зменшити негативний вплив на стан здоров’я працівника при роботі з ПК потрібно робити перерви тривалість 10 хв. через кожну годину роботи.

4.3 Методи боротьби з монотонністю праці на виробництві

У перекладі з грецької монотонність означає одноманітність. Монотонною вважається робота, яка відповідає таким ознакам: невелика кількість виконуваних дій, простота дій, часта повторюваність дій. Таким чином робота з розроблюваною комп’ютерною системою є монотонною, оскільки оператор повинен лише перемикатись між датчиками, слідкувати за їхніми показами і фіксувати їх, а також час від часу вносити поправки в налаштування [9, 16].

В залежності від виду роботи і навантаження на організм людини виділяють два типи монотонності:

– Рухова монотонність – характерна для робіт, де основне навантаження припадає на опорно – руховий апарат. Така робота характеризується одноманітними рухами і діями, а основне навантаження припадає на якусь обмежену групу м’язів. Прикладом таких робіт є прості верстатні роботи, робота на конвеєрі, ручні допоміжні роботи, тощо.

– Сенсорна монотонність – характерна для робіт пов’язаних з обробкою інформації. В них вимагається постійне напруження сенсорних органів, уваги, пам’яті. Прикладом таких робіт є тривале пасивне спостереження.

При тривалому виконанні монотонних робіт у працівників виникає втома, зменшується увага, погіршується якість виконаної роботи. Це все може призвести до помилкових дій і аварійних ситуацій.

В довго строковій перспективі монотонна роботи може мати такі наслідки:

– швидкий розвиток втоми в зв’язку з локалізацією м’язових і нервових навантажень;

– гіподинамія;

– розвиток неврозів;

– незадоволення роботою і зниження творчої активності працівника;

– підвищена плинність кадрів.

Для боротьби з монотонністю існують два підходи:

– Зробити роботу менш одноманітною. Для цього робочий процес потрібно переробити так, щоб кількість окремих робіт зменшилась, але вони стали більш складними, наприклад об’єднати кілька простих процесів в один складний. Також, можна регулювати навантаження в залежності від стану робітників, наприклад, при роботі за конвеєром можна пускати його швидше чи повільніше.

– Додати сторонні подразники. Оскільки однією з причин монотонності є мала кількість подразників, то можна збільшити їх штучно, наприклад ввімкнувши на робочому місці музику. Звісно додавати додаткові подразники можна лише за умови, якщо вони не будуть відволікати працівників від основної роботи.

При роботі з розроблюваною системою перший метод не підходить, оскільки слідкувати за виміряними параметрами потрібно з певною частотою, яку не можна змінити.

Також потрібно чергувати монотонну працю з якоюсь іншою. При зміні діяльності потрібно враховувати наступне:

– операції, що підбираються для чергування, не повинні завантажувати ті самі органи й системи організму. Доцільно чергувати фізичну роботу з розумовою, навантаження на орган зору з роботою, де беруть участь інші аналізатори (слухові, дотикальні й ін.), роботу з керування механізмами — з ручною працею;

– при зміні форм діяльності необхідно враховувати вік працівників, тому що в молодих людей цей метод дає більший ефект, чим у людей похилого віку;

– систематичне чергування видів праці можна вводити лише тоді, коли працівники повністю опановують кожною з виконуваних операцій;

– робота, що сполучається, повинна бути помірною або легшою, порівняно з основною;

– при сполученні робіт найкращого результату можна досягти коли більш інтенсивна робота заміняється менш інтенсивною, важча й складніша — простішою;

– чергуючі роботи повинні відрізнятися за характером робочої пози, навантаженням на різні ланки рухового апарата, забезпечувати перемикання діяльності з одних м’язових груп на інші. Статична напруга м’язів у відомих межах є стимулятором динамічної роботи. Це необхідно враховувати при сполученні робіт;

– залежно від швидкості перебудови робочого динамічного стереотипу (це залежить від складності робіт) чергування виконуваних робіт у часі може здійснюватися протягом робочої зміни, тижня або більше тривалих відрізків часу;

– на ділянках з несприятливими умовами праці сполучення операцій застосовується з метою скорочення часу впливу несприятливих факторів на організм людини.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи бакалавра було розроблено комп'ютерну систему моніторингу стану повітря для холодильної установки, розроблено електричну принципову схему проектованого пристрою і написане відповідне програмне забезпечення.

Система розроблена на основі мікроконтроллера STM32F103C8T6, для зчитування температури і відносної вологості використовуються цифрові датчики SHT30, для вимірювання атмосферного тиску використовуються датчики BMP280, виміряна інформація виводиться на OLED – дисплей.

Для програмування системи використовувались графічне середовище CubeMX і CooCox (CoIDE). Для обміну інформацією між мікроконтроллером і датчиками використовується шина I2C, інформація на дисплей передається по шині SPI.

У першому розділі наведено аналіз технічного завдання, описано існуючі аналоги і наводяться можливі рішення.

В другому розділі описана структурна і електрична – принципова схеми розроблюваної системи, обґрунтовано вибір апаратного забезпечення і шин обміну інформації.

В третьому розділі описана програмна реалізація комп'ютерної системи.

Четвертий розділ містить опис питань по безпеці життєдіяльності і основах охорони праці.

Розроблюваний пристрій є завершеним і готовим до використання, проте в майбутньому його можна покращити. Для підвищення зручності використання можна додати клавіатуру, щоб користувач задавав для датчиків діапазон даних за допомогою пристрою, а не в коді програми. Також можна збільшити функціональність програми додавши більше датчиків інших типів, функціонально це можливо, оскільки шина I2C підтримує до 127 пристроїв.

ПЕРЕЛІК ПОСИЛАНЬ

1. BMP280 Digital Pressure Sensor: технічний опис, версія 1.14. 5 Травня 2015. 49 с.

2. OLED дисплей 0.96" I2C/SPI інтерфейси 128x64 (жовто-синій) від Waveshare. *Arduino UA*. URL: https://arduino.ua/prod2121-oled-displei-0-96-i2c-spi-interfeisi-128x64-jelto-sinii (дата звернення: 20.04.2022).

3. SDD1306-30pin: технічний опис, версія E. 40 с.

4. Sensirion_Humidity_Sensors_SHT3xA_Datasheet: технічний опис, 7 версія. Грудень 2019. 21 с.

5. STM32Cube initialization code generator. *STMicroelectronics. 2022.* URL: https://www.st.com/en/development-tools/stm32cubemx.html (дата звернення: 25.04.2022).

6. STM32F103C8T6 управляет 6-проводным OLED. URL: https://russianblogs.com/article/47791598064/ (дата звернення: 02.05.2022).

7. STM32F103x8 STM32F103xB: технічний опис, 12 версія. Червень 2010. 96 с.

8. Барометр BMP280 3.3В (датчик атмосферного тиску). *Arduino UA*. URL: https://arduino.ua/prod1758-barometr-datchik-atmosfernogo-davleniya-na-bmp280 (дата звернення: 16.04.2022).

9. Грибан В.Г., Негодченко О.В. Охорона праці. – К.: Центр учбової літератури, 2009. 209 с.

10. Датчик температури та вологості SHT30 I2C. *Arduino UA*. URL: https://arduino.ua/prod4497-datchik-temperatyri-i-vlajnosti-sht30-i2c (дата звернення: 14.04.2022).

11. Зеркалов Д.В. Безпека життєдіяльності. Навчальний посібник. - К.: Основа, 2011.

12. Интерфейсная шина IIC (I2C). 2009. URL: http://easyelectronics.ru/interface-bus-iic-i2c.html (дата звернення: 22.04.2022).

13. Логер температури двоканальний AZ-88394. *ThermoLab*.URL:

https://thermolab.net.ua/ua/p1457520075-logger-temperatury-dvuhkanalnyj.html?source=merchant_center&gclid=Cj0KCQjwwJuVBhCAARIsAOPwGAQltqvhf5MPGGZgJN3JFpa4R7T5qQ9GXumtimfL2111aDba7bWWDNoaAuHoEALw_wcB (дата звернення: 02.04.2022).

14. Моніторинг температури в холодильних камерах. URL: https://owen.ua/ua/projects/monitoryng-temperatury-v-holodylnyh-kamerah (дата звернення: 04.04.2022).

15. Обзор шины SPI. 2011. URL: https://habr.com/ru/post/123145/ (дата звернення: 24.04.2022).

16. Основи охорони праці. / Під ред. Ткачука К.Н., Халімовського Н.О. – К.: Основа, 2006. 448 с.

17. Подключение HTU21D к STM32. 2020. URL: https://cxemka.com/46-podklyuchenie-htu21d-k-stm32-c-hal-po-i2c-datchik-temperatury-i-vlazhnosti.html (дата звернення: 30.04.2022).

18. Подключение OLED дисплея ssd1306 к STM32. 2020. URL: https://habr.com/ru/post/514382/ (дата звернення: 04.05.2022).

Додаток А.

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

«Затверджую»

завідувач кафедри КС

_____ Осухівська Г.М.

"___ "_____ 2022 р.

Комп'ютерна система моніторингу стану повітря для холодильної установки

ТЕХНІЧНЕ ЗАВДАННЯ

на __5__ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»

Керівник курсового проєкту

_____ к.т.н., доц. Тиш Є.В.

«____»_____ 2022 р.

«ВИКОНАВЕЦЬ»

Студент групи СІс-43

_____ Купленний О.Д.

«____»_____ 2022 р

Тернопіль 2022

1. Назва та підстава для виконання проєкту.

1.1. Комп’ютерна система моніторингу стану повітря для холодильної установки.

1.2. Підставою для виконання кваліфікаційної роботи бакалавра є наказ по університету (№ 4/7 – 180 від 23.03.2022).

2. Виконавець.

2.1. Студент групи СІс-43 кафедри КС
Тернопільського національного технічного університету ім. І. Пулюя
Купленний Олександр Дмитрович.

3. Мета роботи.

3.1. Метою роботи є розробити стрктуру, схему та програмне забезпечення комп’ютерної системи моніторингу стану повітря для холодильної установки.

4. Склад виробу.

4.1. До складу виробу повинні входити:

1) датчики температури і відносної вологості;
2) датчики атмосферного тиску;
3) кнопка і світлодіоди;
4) OLED – дисплей;
5) мікроконтролер або мікропроцесор;
6) комплект документації.

5. Технічні вимоги.

5.1. Вимоги по призначенню.

5.1.1. Вбудована система повинна мати наступні параметри:

1) Кількість датчиків температури і вологості шт. 2

2) Діапазон вимірювання температури, не гірше, °С -40… +100

3) Точність вимірювання температури, не гірше, °С +/- 0,3

4) Точність вимірювання вологості, не гірше, % +/- 3

5) Кількість датчиків атмосферного тиску шт 2

6) Діапазон вимірювання атмосферного тиску, hPa 300-1100

7) Кількість екранів, шт 1

8) Кількість кнопок управління, шт. 1

5.1.2. Система повинна живитись напругою постійного струму, В +5 ±0,2

5.2. Вимоги до умов експлуатації:

5.2.1. По умовам експлуатації виріб повинен відповідати вимогам ГОСТ 15150 для УХЛ4.1

5.2.2. Температура експлуатації від -35 до +95ºС

5.2.3. Відносна вологість до 100% при t=25ºС

5.3. Конструктивні вимоги.

5.3.1. Конструювання корпусу приладу в КРБ не передбачено.

5.3.2. Для побудови системи мають бути використані сучасні компоненти з можливістю поверхневого монтажу друкованого вузла.

5.3.3. При побудові системи необхідно передбачити розміщення роз’ємів живлення і обміну даними.

5.3.4. Габаритні розміри при макетуванні, мм, не більше:

довжина 800

ширина 600

висота 600

5.3.5. Маса макету, кг, не більше 3

5.3.6. Конструкція макету повинна забезпечувати доступ до всіх комплектуючих виробів при тестуванні.

5.4. Вимоги до надійності.

5.4.1. Система повинна відповідати вимогам ДСТУ 2862-94.

5.4.2. Наробка на відмову, не менше 5000 год.

5.5. Вимоги метрології.

5.5.1. Вимірювання параметрів системи при моделюванні повинно виконуватись на універсальних вимірювальних приладах.

6. Економічні показники.

6.1. Собівартість системи повинна бути не більше 8000 грн.

7. Вимоги до документації.

7.1. Конструкторська документація повинна відповідати вимогам ЄСКД, ДСТУ та ГОСТ.

7.2. До складу документації повинно входити:

1) ПЗ
2) Структурна схема Е1
3) Електрична схема Е2
4) Налаштування мікроконтроллера
5) Блок схема алгоритму роботи С1

8. Стадії та етапи розробки КРБ

8.1 Стадії та етапи виконання КРБ наведенні в таблиці 1.

Таблиця 1

№	Назва етапу	Строк виконання	
		початок	кінець
1	Технічне завдання	23.03.2022	31.03.2022
2	Розділ 1 Аналіз технічного завдання	01.04.2022	12.04.2022
3	Розділ 2 Проектна частина	13.04.2022	28.04.2022
4	Розділ 3 Практична частина	29.04.2022	15.05.2022
5	Розділ 4 Безпека життєдіяльності, основи охорони праці	16.05.2022	27.05.2022
6	Оформлення графічної документації	28.05.2022	10.06.2022
7	Нормоконтроль	11.06.2022	13.06.2022
8	Попередній захист	14.06.2022	21.06.2022
9	Захист	22.06.2022	23.06.2022

9. В дане ТЗ можуть вноситись зміни по узгодженню сторін.

Додаток Б

Перелік елементів

Позначення	*Найменування*	*К-сть.*	*Примітки*
	Мікросхеми		
STM32F103 C8T6	Мікроконтроллер STM32F103 C8T6	1	
CN1, CN2, CN3, CN4, CN5	Виводи мікроконтроллера STM32F103 C8T6	5	
L1, L2	RGB – світлодіоди	2	
SB1	Кнопка	1	
TS1, TS2	Датчики температури / вологості SHT30	2	
PS1, PS2	Датчики тиску BMP280	2	
UT-0206-P05	OLED - дисплей	1	
Q1, Q2	Транзистори	2	
	Резистори		
R1, R2, R3, R5, R6, R7	330 Ом	6	
R4, R16	10КОм	2	
R8, R9	47 КОм	2	
R10, R11	10 КОм	2	
R12, R13	4.7 КОм	2	
R17, R18	22 рОм	2	
R19, R14	100 КОм	2	
R15	910 КОм	1	
	Конденсатори		
C1, C6, C7, C8, C9, C10, C11, C18, C19	100 нФ	9	
C2, C3, C4,C5	1 мФ	4	
C14	4,7 мФ	1	
C15	2,2 мФ	1	
C12, C13	12 пФ	2	
C16, C17	100 пФ	2	
X1	8M0	1	
X2	32к768	1	

					КС КРБ 123.225.00.00 ПЗ
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	
Розробив		Купленний О.Д.			*Комп'ютерна система моніторингу стану повітря для холодильної установк.*
Перевірив		Тиш Є.В.			
Рецензент		Загородна Н.В.			
Н. Контр.		Луцик Н.С.			
Затвердив		Осухівська Г.М.			

Літ.	*Арк.*	*Акркушів*
	64	

ТНТУ, каф. КС, гр. СІс-43

Додаток В

Код програми

```
/**
ОСНОВНИЙ ФАЙЛ
  */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f1xx_hal.h"
#ifndef __OLED_H
#define __OLED_H
#include "sys.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;

SPI_HandleTypeDef hspi1;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/*Глобальні змінні для роботи з датчиком температури і вологості*/
/*ПОЧАТОК*/
uint8_t Sensor1_Data[6]; // масив з 6 байт, 2-передане значення температури 1 датчика + 1-CRC + 2 - значення вологості 1 датчика + 1-CRC
uint8_t Sensor2_Data[6]; // масив з 6 байт, 2-передане значення температури 2 датчика + 1-CRC + 2 - значення вологості 2 датчика + 1-CRC
uint16_t Temperature1_RAW; // необроблене значення температури з датчика 1
uint16_t Temperature2_RAW; // необроблене значення температури з датчика 2
uint16_t Humidity1_RAW; // необроблене значення вологості з датчика 1
uint16_t Humidity2_RAW; // необроблене значення вологості з датчика 2
float Temperature1; // оброблене значення температури 1
float Temperature2; // оброблене значення температури 2
float Humidity1; // оброблене значення вологості 1
float Humidity2; // оброблене значення вологості 2
uint16_t Measure = 0x240B; // команда для зчитування інформації з датчика
#define Sensor1_Adress (0x44) //адреса першого датчика температури і вологості
#define Sensor2_Adress (0x45) //адреса другого датчика температури і вологості
/*КІНЕЦЬ*/
```

```
/*Глобальні змінні для роботи з датчиком тиску*/
/*ПОЧАТОК*/
uint16_t Pressure1_RAW; // необроблене значення тиску з датчика 1
uint16_t Pressure2_RAW; // необроблене значення тиску з датчика 2
float Pressure1; // оброблене значення тиску 1
float Pressure2; // оброблене значення тиску 2
#define ID_REG (0xD0) //адреса першого датчика тиску
#define PRESS_MSB_REG (0xF2) //адреса першого датчика тиску
#define Pressure1_Adress (0x76) //адреса першого датчика тиску
#define Pressure2_Adress (0x77) //адреса другого датчика тиску
/*КІНЕЦЬ*/

/*Глобальні змінні для знаходження середніх значень*/
/*ПОЧАТОК*/
float Temperature_Sered; // середнє арифметичне значення температури з усіх датчиків у системі
float Humidity_Sered; // середнє арифметичне значення вологості з усіх датчиків у системі
float Pressure_Sered; // середнє арифметичне значення тиску з усіх датчиків у системі
/*КІНЕЦЬ*/

/*Глобальні змінні для роботи з екраном*/
/*ПОЧАТОК*/
u8 OLED_GRAM[128][8];// масив для роботи з кешом
#define OLED_SCLK_Clr()  PAout(5)=0   //SCK
#define OLED_SCLK_Set()  PAout(5)=1   //SCK
#define OLED_SDIN_Clr()  PAout(7)=0   //SDA
#define OLED_SDIN_Set()  PAout(7)=1   //SDA
#define OLED_RES_Clr()     PAout(1)=0   //RST
#define OLED_RES_Set()   PAout(1)=1   //RST
#define OLED_DC_Clr()    PAout(3)=0    //DC
#define OLED_DC_Set()      PAout(3)=1    //DC
#define OLED_CMD  0        // Запис команди
#define OLED_DATA 1        // Запис даних
/*КІНЕЦЬ*/

/*Значення параметрів встановлені користувачем*/
/*ПОЧАТОК*/
float User_Temperature1_Min = -10.00; // Мінімальне зачення температури на 1 датчику
float User_Temperature1_Max = -8.00; // Максимальне значення температури на 1 датчику
float User_Humidity1_Min = 12.00; // Мінімальне зачення вологості на 1 датчику
float User_Humidity1_Max = 25.00; // Максимальне значення вологості на 1 датчику

float User_Temperature2_Min = -12.00;// Мінімальне зачення температури на 2 датчику
float User_Temperature2_Max = 0.00;// Максимальне значення температури на 2 датчику
```

```
float User_Humidity2_Min = 12.00; // Мінімальне зачення вологості на 2 датчику

float User_Humidity2_Max = 25.00; // Максимальне значення вологості на 2 датчику

float User_Pressure1_Min = 735.00;// Мінімальне зачення температури на 2 датчику
float User_Pressure1_Max = 760.00;// Максимальне значення температури на 2 датчику
float User_Pressure2_Min = 750.00; // Мінімальне зачення вологості на 2 датчику
float User_Pressure2_Max = 770.00; // Максимальне значення вологості на 2 датчику

int Nomer_dat =1;
/*КІНЕЦЬ*

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_SPI1_Init(void);

/*Функції для роботи з дисплеєм*/
/*ПОЧАТОК*/
void OLED_WR_Byte(u8 dat,u8 cmd);        // Встановлює значення певного байта (адресу стовпця...)
void OLED_Display_On(void); // Функція для включення дисплею
void OLED_Display_Off(void); // Функція для виключення дисплею
void OLED_Refresh_Gram(void); // Функція оновлення кешу і відображення вмісту
void OLED_Init(void); // ініціалізація дисплею
void OLED_Clear(void); //очистка дисплею
void OLED_DrawPoint(u8 x,u8 y,u8 t);
void OLED_ShowChar(u8 x,u8 y,u8 chr,u8 size,u8 mode); // відображення символу
void OLED_ShowNumber(u8 x,u8 y,u32 num,u8 len,u8 size);// відображення номера
void OLED_ShowString(u8 x,u8 y,const u8 *p);// відображення числа
/*КІНЕЦЬ*/

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
```

```
  * @brief  The application entry point.
  *

  * @retval None
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_SPI1_Init();
  /* USER CODE BEGIN 2 */
  OLED_Init();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
          /**********Отримування температури і вологості з датчика 1**********/
 HAL_I2C_Mem_Read_IT(&hi2c1, Sensor1_Adress, Measure, I2C_MEMADD_SIZE_8BIT, (uint8_t*) Sensor1_Data[6];, 5); // Пересилання на датчик команди почати зчитування, зчитування і запис даних в масив
```

```
 Temperature1_RAW = ((uint16_t)(Sensor1_Data[0] << 8) | (Sensor1_Data[1])); // Об'єднання 2 елементів масиву, які містять значення температури в одне ціле
Temperature1 = (float)(Temperature1_RAW * 175 / 65535.00) - 45; // Перетворення значення отриманого з датчика по формулі в градуси цельсія.
Humidity1_RAW = ((uint16_t)(Sensor1_Data[3] << 8) | (Sensor1_Data[4])); // Об'єднання 2 елементів масиву, які містять значення вологості в одне ціле
Humidity1 = (float)(Humidity1_RAW/65535)*100; // Перетворення значення отриманого з датчика по формулі в відсотки.
 HAL_Delay(100);

        /**********Отримування температури і вологості з датчика 2**************/
HAL_I2C_Mem_Read_IT(&hi2c1, Sensor2_Adress, Measure, I2C_MEMADD_SIZE_8BIT, (uint8_t*) Sensor2_Data[6];, 5); // Пересилання на датчик команди почати зчитування, зчитування і запис даних в масив
Temperature2_RAW = ((uint16_t)(Sensor2_Data[0] << 8) | (Sensor2_Data[1])); // Об'єднання 2 елементів масиву, які містять значення температури в одне ціле
Temperature2 = (float)(Temperature2_RAW * 175 / 65535.00) - 45; // Перетворення значення отриманого з датчика по формулі в градуси цельсія.
Humidity2_RAW = ((uint16_t)(Sensor2_Data[3] << 8) | (Sensor2_Data[4])); // Об'єднання 2 елементів масиву, які містять значення вологості в одне ціле
 Humidity2 = (float)(Humidity2_RAW/65535)*100; // Перетворення значення отриманого з датчика по формулі в відсотки.
HAL_Delay(100);

        /*************Отримування 1 значення тиску**********/
HAL_I2C_Mem_Read( & hi2c1,  Pressure1_Adress,  ID_REG,  1 ,  & chipID,  1 ,  1000 );
HAL_I2C_Mem_Read(BME280_I2C,  Pressure1_Adress,  PRESS_MSB_REG,  1 ,  RawData,  8 ,  HAL_MAX_DELAY);
 Pressure1_RAW  =  (RawData[ 0 ] << 12 ) | (RawData[ 1 ] << 4 ) | (RawData[ 2 ] >> 4 );
if  (Pressure1_RAW  ==  0x800000 )  Pressure1  =  0 ;  // значення у випадку, якщо вимірювання температури було вимкнено
 else
{
        if (SUPPORT_64BIT)
         {
         Pressure1  =  (BME280_compensate_P_int64  (Pressure1_RAW)) / 256.0 ;   // згідно з даними, тиск становить x256
          }
          if (SUPPORT_32BIT)
          {
          Pressure1  =  (BME280_compensate_P_int32  (Pressure1_RAW));   // згідно з даними, тиск дорівнює Pa
          }
 }
```

```
        /**********Отримування 2 значення тиску**********/
 HAL_I2C_Mem_Read( & hi2c1,  Pressure2_Adress,  ID_REG,  1 ,  & chipID,  1 ,  1000 );
 HAL_I2C_Mem_Read(BME280_I2C,  Pressure2_Adress,  PRESS_MSB_REG,  1 ,  RawData,
8 ,  HAL_MAX_DELAY);

Pressure1_RAW  =  (RawData[ 0 ] << 12 ) | (RawData[ 1 ] << 4 ) | (RawData[ 2 ] >> 4 );
if  (Pressure1_RAW  ==  0x800000 )  Pressure2  =  0 ;  // значення у випадку, якщо
вимірювання температури було вимкнено
 else
{
        if (SUPPORT_64BIT)
        {
        Pressure2  =  (BME280_compensate_P_int64  (Pressure2_RAW)) / 256.0 ;   // згідно з
даними, тиск становить x256
         }
         if (SUPPORT_32BIT)
         {
         Pressure2  =  (BME280_compensate_P_int32  (Pressure2_RAW));   // згідно з даними,
тиск дорівнює Pa
         }
}
        /***********Індикація 1 групи датчиків***********/
if(Temperature1 < User_Temperature1_Min || Humidity1 < User_Humidity1_Min || Pressure1 <
User_Pressure1_Min) //якщо якесь значення  на  1 групі датчиків нижче за мінімальне
значення внесене користувачем, то засвічується синій колір світлодіода.
         {
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET); // синій
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET); // зелений
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET); // червоний
         }

 if(Temperature1 > User_Temperature1_Max || Humidity1 > User_Humidity1_Max || Pressure1
> User_Pressure1_Max) //якщо якесь значення на 1 групі датчиків вище за максимальне
значення внесене користувачем, то засвічується червоний колір світлодіода.
        {
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET); // синій
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET); // зелений
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_SET); // червоний
        }
 else // в іншому випадку (всі значення в межах норми) світиться зелений колір
         {
                 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET); // синій
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET); // зелений
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET); // червоний
         }
```

```
        /********Індикація 2 групи датчиків*********/

if(Temperature2 < User_Temperature2_Min || Humidity2 < User_Humidity2_Min || Pressure2 < User_Pressure2_Min) //якщо якесь значення  на  2 групі датчиків нижче за мінімальне значення внесене користувачем, то засвічується синій колір світлодіода.
        {
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_SET); // синій
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, GPIO_PIN_RESET); // зелений
          HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_RESET); // червоний
        }

if(Temperature2 > User_Temperature2_Max || Humidity2 > User_Humidity2_Max || Pressure2 > User_Pressure2_Max) //якщо якесь значення на 2 групі датчиків вище за максимальне значення внесене користувачем, то засвічується червоний колір світлодіода.
         {
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_RESET); // синій
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, GPIO_PIN_RESET); // зелений
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_SET); // червоний
        }
        else в іншому випадку (всі значення в межах норми) світиться зелений колір
        {
               HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_RESET); // синій
               HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, GPIO_PIN_SET); // зелений
               HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_RESET); // червоний
        }

        /********Знаходження середніх арифметичних значень параметрвів*********/
        Temperature_Sered = (Temperature1 + Temperature2)/2;
        Humidity_Sered = (Humidity1 + Humidity2)/2;
        Pressure_Sered = (Pressure1 + Pressure2)/2:

        /********Налаштування кнопки керування і екрану*********/
        if (! HAL_GPIO_ReafPin(GPIOB, GPIO_PIN_14))
        {
                Nomer_dat=Nomer_dat+1;
        }
        if(Nomer_dat>3)
        {
                Nomer_dat=1;
        }

if (Nomer_dat=1)
        {
                OLED_Display_On();
                OLED_ShowString(50,0,"Temper_1");
```

```
                OLED_ShowNumber(0,15,Temperature1,4);
                OLED_ShowString(50,0,"Hum_1");
                OLED_ShowNumber(0,15,Humidity1,4);
                OLED_ShowString(50,0,"Press_1");
                OLED_ShowNumber(0,15,Pressure1,4);
                OLED_Refresh_Gram();
                delay_ms(50);
        }

        if (Nomer_dat=2)
        {
                OLED_Display_On();
                OLED_ShowString(50,0,"Temper_2");
                OLED_ShowNumber(0,15,Temperature2,4);
                OLED_ShowString(50,0,"Hum_2");
                OLED_ShowNumber(0,15,Humidity1,4);
                OLED_ShowString(50,0,"Press_2");
                OLED_ShowNumber(0,15,Pressure2,4);
                OLED_Refresh_Gram();
                delay_ms(50);
        }

        if (Nomer_dat=3)
                {
                        OLED_Display_On();
                        OLED_ShowString(50,0,"Temp-ser");
                        OLED_ShowNumber(0,15,Temperature_Sered,4);
                        OLED_ShowString(50,0,"Hum-ser");
                        OLED_ShowNumber(0,15,Humidity_Sered,4);
                        OLED_ShowString(50,0,"Pres-ser");
                        OLED_ShowNumber(0,15,Pressure_Sered,4);
                        OLED_Refresh_Gram();
                        delay_ms(50);
                }
  /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */

}

/**

  * @brief System Clock Configuration
  * @retval None
  */
```

```
/*Функції дисплея*/
/*Початок*/
void OLED_Refresh_Gram(void)//оновлення кешу і відображення вмісту
{
	u8 i,n;
	for(i=0;i<8;i++)
	{
		OLED_WR_Byte (0xb0+i,OLED_CMD);    // Встановлює адрес сторінки
		OLED_WR_Byte (0x00,OLED_CMD);      // Встановлює нижній адрес
стовпця позиції відображення
		OLED_WR_Byte (0x10,OLED_CMD);      // Встановлює верхній адрес
стовпця позиції відображення
		for(n=0;n<128;n++)OLED_WR_Byte(OLED_GRAM[n][i],OLED_DATA);
	}
}

void OLED_WR_Byte(u8 dat,u8 cmd) // запис байту в дисплей
{
	u8 i;
	if(cmd)
	  OLED_RS_Set();
	else
	  OLED_RS_Clr();
	for(i=0;i<8;i++)
	{
		OLED_SCLK_Clr();
		if(dat&0x80)
		  OLED_SDIN_Set();
		else
		  OLED_SDIN_Clr();
		OLED_SCLK_Set();
		dat<<=1;
	}
	OLED_RS_Set();
}

void OLED_Display_On(void) // Включення дисплею
{
	OLED_WR_Byte(0X8D,OLED_CMD);
	OLED_WR_Byte(0X14,OLED_CMD);
	OLED_WR_Byte(0XAF,OLED_CMD);
}

void OLED_Display_Off(void)// Виключення дисплею
{
```

```
	OLED_WR_Byte(0X8D,OLED_CMD);
	OLED_WR_Byte(0X10,OLED_CMD);
	OLED_WR_Byte(0XAE,OLED_CMD);
}

void OLED_Clear(void)  //очистка дисплею
{
	u8 i,n;
	for(i=0;i<8;i++)for(n=0;n<128;n++)OLED_GRAM[n][i]=0X00;
	OLED_Refresh_Gram();// Оновити відображення
}

void OLED_DrawPoint(u8 x,u8 y,u8 t) // відображення точок
{
	u8 pos,bx,temp=0;
	if(x>127||y>63)return;
	pos=7-y/8;
	bx=y%8;
	temp=1<<(7-bx);
	if(t)OLED_GRAM[x][pos]|=temp;
	else OLED_GRAM[x][pos]&=~temp;
}
void OLED_ShowChar(u8 x,u8 y,u8 chr,u8 size,u8 mode) // відображення символів
{
	u8 temp,t,t1;
	u8 y0=y;
	chr=chr-' ';// отримуємо значеня зміщення
   for(t=0;t<size;t++)
   {
		if(size==12)temp=oled_asc2_1206[chr][t];
		else temp=oled_asc2_1608[chr][t];
      for(t1=0;t1<8;t1++)
		{
			if(temp&0x80)OLED_DrawPoint(x,y,mode);
			else OLED_DrawPoint(x,y,!mode);
			temp<<=1;
			y++;
			if((y-y0)==size)
			{
				y=y0;
				x++;
				break;
			}

		}
```

```
   }
}

u32 oled_pow(u8 m,u8 n)
{
	u32 result=1;
	while(n--)result*=m;
	return result;
}

void OLED_ShowNumber(u8 x,u8 y,u32 num,u8 len,u8 size)// Відображення чисел
{
	u8 t,temp;
	u8 enshow=0;
	for(t=0;t<len;t++)
	{
		temp=(num/oled_pow(10,len-t-1))%10;
		if(enshow==0&&t<(len-1))
		{
			if(temp==0)
			{
				OLED_ShowChar(x+(size/2)*t,y,' ',size,1);
				continue;
			}else enshow=1;

		}
		OLED_ShowChar(x+(size/2)*t,y,temp+'0',size,1);
	}
}

void OLED_ShowString(u8 x,u8 y,const u8 *p)//Відображення стрічки
{
#define MAX_CHAR_POSX 122
#define MAX_CHAR_POSY 58
   while(*p!='\0')
   {
      if(x>MAX_CHAR_POSX){x=0;y+=16;}
      if(y>MAX_CHAR_POSY){y=x=0;OLED_Clear();}
      OLED_ShowChar(x,y,*p,12,1);
      x+=8;
      p++;
   }
}
/*Кінець*/
```

```
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure the Systick interrupt time
    */
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

    /**Configure the Systick
    */
  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}
```

```
/* I2C1 init function */
static void MX_I2C1_Init(void)
{

  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/* SPI1 init function */
static void MX_SPI1_Init(void)
{

  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_1LINE;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_4;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
```

```
/** Configure pins as

        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
*/
static void MX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_8
                    |GPIO_PIN_9|GPIO_PIN_10, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_10|GPIO_PIN_11,
GPIO_PIN_RESET);

  /*Configure GPIO pins : PA1 PA2 PA3 PA8
                         PA9 PA10 */
  GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_8
                    |GPIO_PIN_9|GPIO_PIN_10;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PB1 PB10 PB11 */
  GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_10|GPIO_PIN_11;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : PB14 */
  GPIO_InitStruct.Pin = GPIO_PIN_14;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
```

```
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  file: The file name as string.
  * @param  line: The line in file as a number.
  * @retval None
  */
void _Error_Handler(char *file, int line)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  while(1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t* file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
/**
  * @}
  */
/**
  * @}
  */
/********************** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```