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'ютеризована система контролю за умовами зберігання лікарських засобів*

Виконав(ла): студент(ка) *IV* курсу, групи *СІ-41*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

	(підпис)	*Невмержицький В. В.* (прізвище та ініціали)
Керівник	(підпис)	*Жаровський Р. О.* (прізвище та ініціали)
Нормоконтроль	(підпис)	*Тиш Є. В.* (прізвище та ініціали)
Завідувач кафедри	(підпис)	*Осухівська Г. М.* (прізвище та ініціали)
Рецензент	(підпис)	Марценюк В. П. (прізвище та ініціали)

Тернопіль
2024

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)
«25» 06 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

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

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

студенту *Невмержицькому Віталію Володимировичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система контролю за умовами зберігання лікарських засобів*

Керівник роботи *Жаровський Руслан Олегович, к.т.н., старший викладач*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*24*» *квітня* 20*24* року № 4/7-408

2. Термін подання студентом завершеної роботи *.06.2024 р.*

3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналіз технічного завдання
2. Проєктна частина
3. Практична частина
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Структурна схема
2. Схема електрична принципова
3. Блок-схема алгоритму
4. Програмний інтерфейс

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	*Пилипець М. І., проф. каф. МТ, д. т. н.*		

7. Дата видачі завдання *25.04.2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Розробка технічного завдання*	*01.02-09.02*	*Виконано*
2	*Аналіз технічного завдання*	*05.02 – 11.02*	*Виконано*
3	*Аналіз та обґрунтування можливих рішень*	*25.04 – 03.05*	*Виконано*
4	*Розробка структурної схеми*	*04.05-15.05*	*Виконано*
5	*Розробка електричної принципової схеми, вибір елементної бази*	*16.05-25.05*	*Виконано*
6	*Розробка програмного забезпечення для проєктованої системи*	*26.05-09.06*	*Виконано*
7	*Опрацювання питань розділу «Безпека життєдіяльності, основи охорони праці»*	*10.06-15.06*	*Виконано*
8	*Виготовлення тестової версії системи на макетній платі*	*16.06-20.06*	*Виконано*
9	*Оформлення пояснювальної записки кваліфікаційної роботи*	*16.06-20.06*	*Виконано*
10	*Оформлення графічної частини*	*16.06-20.06*	*Виконано*
11	*Попередній захист кваліфікаційної роботи*	*14.06*	*Виконано*
12	*Захист кваліфікаційної роботи*	*26.06*	*Виконано*

Студент _____ *Невмержицький В. В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Жаровський Р. О.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп’ютеризована система контролю за умовами зберігання лікарських засобів // Кваліфікаційна робота бакалавра // Невмержицький Віталій Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних систем та мереж, група СІ-41 // Тернопіль, 2024 // с. – 74, рис. – 62, табл. – 6, аркушів А1 – 4, бібліогр. – 23.

Ключові слова: КОНТРОЛЬ ТЕМПЕРАТУРИ, КОНТРОЛЬ ВОЛОГОСТІ, ОХОЛОДЖЕННЯ, ОСУШЕННЯ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, ЕНЕРГОЕФЕКТИВНІСТЬ.

Кваліфікаційна робота присвячена розробці та реалізації системи автоматичного контролю температури та вологості в шафі для зберігання ліків. Для досягнення цієї мети, використовується мікроконтролер STM32, сенсори та вентилятори. Ця система дозволяє ефективно регулювати параметри середовища шафи, забезпечуючи вимоги зберігання медикаментів, енергоефективність та комфорт для працівників та відвідувачів. У всіх режимах роботи системи збираються статистичні дані по вологості та температурі і зберігаються на SD-карту. Розроблено зручний інтерфейс користувача, який дозволяє контролювати всі покази, здійснювати налаштування системи, переглядати збережену статистику вибираючи необхідний файл у файловому менеджері. Система здатна автоматично зменшувати вологість чи температуру шляхом створення примусової вентиляції з зовнішнім середовищем за допомогою повітряних заслінок і вентиляторів та запобігти зниженню температури в шафі нижче критичної шляхом увімкнення обігрівача повітря. Розроблена система здатна забезпечувати параметри вологості та температури в шафі для зберігання лікарських засобів при кімнатній температурі.

ANNOTATION

A computerized system for monitoring of the medicines' storage conditions // Bachalor thesis // Nevmerzhytskyi Vitalii // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Systems and Networks, group CI-41 // Ternopil, 2024 // p. – 74, fig. – 62, table. – 6, sheets A1 – 4, ref. – 23.

Key words: TEMPERATURE CONTROL, HUMIDITY CONTROL, COOLING, DRYING, INFORMATION TECHNOLOGY, ENERGY EFFICIENCY.

The qualification work is devoted to the development and implementation of a system of automatic control of temperature and humidity in a medicine cabinet. To achieve this goal, an STM32 microcontroller, sensors and fans are used. This system allows you to effectively adjust the parameters of the cabinet environment, ensuring the requirements of medication storage, energy efficiency and comfort for employees and visitors. In all operating modes of the system, statistical data on humidity and temperature are collected and stored on the SD card. A convenient user interface has been developed that allows you to control all displays, perform system settings, view saved statistics by selecting the necessary file in the file manager. The system is able to automatically reduce humidity or temperature by creating forced ventilation with the outside environment with the help of air dampers and fans and prevent the temperature in the cabinet from falling below a critical temperature by turning on the air heater. The developed system is able to provide humidity and temperature parameters in the medicine storage cabinet at room temperature.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

GPIO	-	general-purpose input/output (інтерфейс введення/виведення загального призначення)
IWDG	-	independent watchdog (незалежний сторожовий таймер)
I^2C	-	inter-integrated circuit (послідовна шина даних для зв’язку інтегральних схем)
LCD	-	liquid crystal display (рідкокристалічний дисплей)
PWM	-	pulse-width modulation (широтно-імпульсна модуляція)
RTC	-	real time clock (годинник реального часу)
RCC	-	reset and clock control (керування скиданням та тактуванням)
SPI	-	serial peripheral interface (послідовний периферійний інтерфейс)
ЗІЗ	-	засоби індивідуального захисту
МК	-	мікроконтролер
ШІМ	-	широтно-імпульсна модуляція

ВСТУП

Для того щоб лікування та підтримка здоров'я людини стали успішними, важлива якість лікарських засобів. Однією із умов збереження якості ліків є їх правильне зберігання. Залежно від вимог виробника, лікарські засоби мають різні умови зберігання. Деякі ліки вимагається зберігати в темноті, інші – при знижених температурах: зберігання у холодильнику – це температура від +2° до +8 °С; зберігання у холодному чи прохолодному місці – від +8 °С до +15 °С. Навіть, до зберігання ліків при кімнатній температурі (від +15 °С до +25 °С) ставляться певні умови щодо вологості та чистоти повітря: відносна вологість повітря повинна бути не більше 60% (найбільше стосується лікарської рослинної сировини) [1].

Існує цілий ряд законодавчих документів, що окреслюють вимоги до зберігання ліків: Закон України «Про лікарські засоби», Накази «Про затвердження Правил зберігання та проведення контролю якості лікарських засобів у лікувально-профілактичних закладах», «Про організацію зберігання в аптечних закладах різних груп лікарських засобів та виробів медичного призначення» [2], настанова "Лікарські засоби. Належна практика зберігання" СТ-Н МОЗУ 42-5.1:2011, затверджена наказом МОЗ України «Про затвердження документів з питань забезпечення якості лікарських засобів» від 16 лютого 2009 року № 95.

Відповідно до цих документів визначено, що місця для зберігання лікарських засобів повинні бути обладнані приладами для здійснення контролю за температурою і відносною вологістю повітря. Холодильне обладнання для зберігання лікарських засобів має бути оснащено термометрами. Отримані дані щоденно заносять до журналу або картки обліку температури та відносної вологості. Засоби вимірювальної техніки повинні бути справними і підлягати регулярній метрологічній повірці.

Станом на початок 2024 року в Україні працює близько 18 000 аптек та 2474 лікарень різного типу (за відомостями Forbes CEO), й кожна з них повинна нести відповідальність за зберігання лікарських засобів, відповідно вимог. Тобто забезпечити всі умови згідно наказів та настанови.

Метою кваліфікаційної роботи є розробка та реалізація системи автоматичного контролю температури та вологості в обмеженому просторі, наприклад шафі. Для досягнення цієї мети використовується мікроконтролер (МК) STM32, сенсори та вентилятори. Ця система дозволить ефективно регулювати параметри середовища, забезпечуючи сталі умови та енергоефективність.

Даний проєкт дозволить ліцензіату, що зберігає, використовує чи продає лікарські засоби, не затрачаючи великих коштів, підготувати холодильну шафу, в якій можна зберігати медикаменти, дотримуючись температурного режиму, необхідної вологості та чистоти повітря, відповідно до вимог зберігання лікарських засобів.

Для реалізації мети проєкту слід реалізувати наступні задачі:

- виконати огляд і аналіз аналогічних систем, які вже впроваджені на ринку;
- розробити структурну схему системи контролю за умовами зберігання лікарських засобів;
- розробити електричну принципову схему проєктованої системи;
- розробити алгоритм щодо забезпечення запропонованої системи;
- створити програму для коректної роботи усіх складових системи.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз вимог до комп’ютерної системи

Для забезпечення умов зберігання лікарських засобів при знижених температурах (від +8 °С до +15 °С) в аптеках, лікарських установах використовується холодильна шафа. Однак для стабільної роботи обладнання потрібно відстежувати такі фактори, як конденсат, вологість, перепади температури, які істотно можуть скоротити термін експлуатації даної шафи й негативно вплинути на її вміст. Найважливішою умовою є оптимальна температура в шафі, що забезпечить тривалість та надійність її роботи, а також сприятиме зберіганню лікарських засобів.

Сучасні системи управління за контролем умов, розроблені на базі контролера, який має конфігурацією керуючої програми, налаштовану під певний тип повітрооброблюючих установок. Функції управління, контролю параметрів і захисту обладнання забезпечує оптимальний набір програмних і апаратних засобів автоматизації.

Основні функції системи автоматичного контролю [3]:

- робота системи за графіком;
- контроль засміченості фільтрів;
- регулювання температури повітря;
- підключення системи при пожежі і різних аваріях;
- дистанційне управління.

Загальні функції системи управління:

- регулювання температури припливного повітря або повітря в обмеженому просторі, згідно заданого значення;

- дистанційне включення / вимикання системи та контроль режимів «робота / аварія», за допомогою виносного пульта управління;
- автоматичне перемикання режимів «зима / літо» згідно заданого значення температури зовнішнього повітря;
- робота системи по добовому або тижневому графіку;
- контроль засмічення повітряних фільтрів;
- контроль параметрів мережі живлення;
- відключення системи при надходженні сигналу «пожежа»;
- відключення системи при переході в аварійний режим.

Для зберігання умов в середовищі шафи та відстеження змін використовую різні програмовані контролери для керування системами охолодження, вентиляції та кондиціонування повітря. Наприклад, контролер pCO5+ CAREL з дисплеєм, як на рис. 1.1 [4].

Рисунок 1.1 – контролер pCO5+ CAREL

Система керування холодильними установками працює в чотири етапи:

- розташовані на місцях датчики збирають інформацію про температуру;
- отримані дані передаються на центральний сервер;

- там вони обробляються програмним забезпеченням;
- у разі виявлення проблем автоматично вживаються необхідні заходи.

Головною метою автоматизації холодильних пристроїв є максимальна заміна ручної праці машинною.

1.2 Аналіз можливих рішень поставленого завдання

В приміщеннях аптек та лікарень широко використовують холодильні шафи. Вони дозволяють зберігати ліки відповідно до умов зазначених виробником. В даних шафах охолодження може бути пасивне і активне. Пасивне охолодження внутрішнього простору шафи відбувається за рахунок вільної циркуляції повітря, яке самовільно надходить в шафу та через отвори в перфорованих дверях й стінах виходить з неї.

Для інтенсивнішого охолодження пасивного варіанту часто не достатньо. Для організації активного охолодження використовують: примусову вентиляцію; кондиціонери; теплообмінники, як зображено на рис.1.2 [5].

Рисунок 1.2 – Принцип розміщення вентиляції в холодильній шафі

Примусова вентиляція охолодження найбільш ефективна, коли температура навколишнього середовища нижча, ніж всередині шафи. Щоб циркуляція повітря була ефективною блоки вентиляторів встановлюють внизу, і на даху шафи. Нижні вентилятори нагнітають холодні повітряні маси в корпус шафи, а вентилятори в даху шафи відводять нагріте повітря за її межі. Також можна встановлювати вентиляційні блоки на бічних стінках шафи. В такому випадку буде здійснюватися поздовжнє охолодження простору шафи, що також забезпечує ефективний захист від перегріву.

Автоматизовані системи вентиляції є найбільш ефективними, дозволяючи точно контролювати та регулювати мікроклімат цілий рік. Ряд повітроводів здійснює подачу повітря, інші служать для його витяжки. Система працює повністю в автоматичному режимі та характеризується [6]:

- надійністю;
- продуктивністю та високим ККД;
- функціональністю;
- точністю налаштування та підтримання заданих параметрів;
- енергоефективністю;
- низький рівень шуму;
- простотою технічного обслуговування.

Кондиціонери для охолодження порожнини шафи використовують у випадках:

- якщо всередині корпусу температура не вища за температуру навколишнього середовища;
- якщо потрібно знизити вологість;
- якщо електронна система генерує середню або велику кількість тепла.

Застосовувати теплообмінники варто для відведення тепла з корпусу в навколишню атмосферу в таких випадках:

- якщо електронне обладнання здатне функціонувати при температурі вище температури навколишнього середовища;

- якщо рівень вологості не має значення;
- якщо система генерує невелику або середню кількість тепла.

Сучасний ринок пропонує багато варіантів холодильних шаф. Наприклад, холодильна шафа CR 450 MEDICAL для аптек, поліклінік, лікарень, лабораторій. Виробник пропонує зберігати в ній ін'єкції для уколів, вакцини, сироватки, рідкі, спиртові настойки та екстракти, сиропи, розчини, що заміняють плазму, речовини, що використовуються у дезінтоксикаційній терапії, суспензії, гелі, мазі, що зберігаються при знижених температурах [7].

Підтримка певної температури та вологості повітря в шафі відбувається без участі фахівців, автоматично. За допомогою терморегулятора користувач встановлює нижній і верхній температурні пороги. Після цього кліматична система шафи буде відстежувати температуру та включать обігрів або охолодження. Вентилятори запускаються, якщо температура в шафі стає занадто високою. Нагрівальний елемент починає працювати при ризику переохолодження обладнання.

Шафа працює на основі компресора високої ефективності, електронного двигуна вентилятора EBD, цифрового термостата, сигналізації при зміні температури та дозволяє підтримувати температуру від +1 °C до +15 °C. Саме компресор дозволяє добиватися низької температури та підтримувати її в шафі.

Однак, якщо потрібно забезпечити зберігання ліків в умовах від +8 °C до +15 °C, то використання даної шафи є досить затратним, оскільки середня собівартість її становить 50000 гривень.

Підводячи підсумки аналізу, за основу було взято умову – зберігання медикаментів при температурі від +8 °C до +25 °C, та вологості повітря нижче 60%. Для реалізації завдання в проєктній роботі зі створення системи контролю за умовами зберігання лікарських засобів для охолодження шафи було взято мікроконтролер STM32, сенсори та вентилятори. Створена система вентиляції дозволяє ефективно регулювати параметри середовища в шафі, забезпечуючи вимоги зберігання медикаментів та енергоефективність.

РОЗДІЛ 2 ПРОЄКТНА ЧАСТИНА

2.1 Розробка узагальненої структури комп'ютерної системи

На рис.2.1 зображена структурна схема комп'ютеризованої системи контролю за умовами зберігання лікарських засобів.

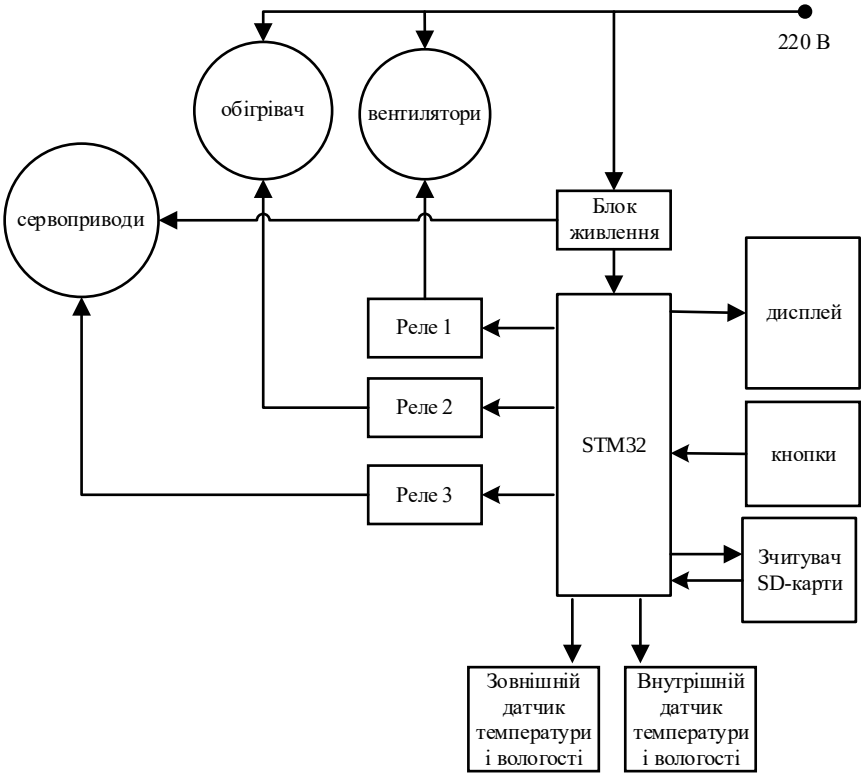

Рисунок 2.1 – Структурна схема комп'ютеризованої системи контролю за умовами зберігання лікарських засобів

Система складається з наступних складових:

- STM32 – це ядро системи. МК виконує зчитування, обробку даних та керує зовнішніми пристроями;

- блок живлення – подає напругу 5 В на STM32, сервоприводи, реле. Дисплей, датчики та зчитувач SD-карти живляться від напруги 3,3 В, що подається від стабілізатора напруги;
- дисплей – призначений для відображення інформації;
- кнопки – призначені для керування системою та її налаштування;
- зчитувач SD-карти – призначений для зчитування та зберігання інформації: налаштування, статистичні дані;
- вентилятори – призначені для створення примусової циркуляції повітря. Для ефективної роботи системи повітря необхідно нагнітати ззовні (з вулиці) фільтруючи його за допомогою фільтра перед подачею в шафу. Відводити повітря з шафи теж доречно за межі приміщення (на вулицю), щоб запах ліків не переходив із шафи в приміщення;
- сервоприводи – призначені для керування положенням заслінок;
- обігрівач – призначений для обігріву повітря для запобігання переохолодження лікарських засобів.

2.2 Обґрунтування вибору апаратного забезпечення проєктованого комп’ютерного засобу

2.2.1 Мікроконтролер

Для використання системи було обрано МК STM32F401CCU6. Плата для розробки та прототипування на цьому 32-бітному МК зображена на рисунку 2.2. Плата сумісна за формфактором із Blue Pill (STM32F103C8T6), але має більш сучасне ядро МК - ARM Cortex-M4.

На платі є всі необхідні елементи для роботи з цим МК:

- 36 портів GPIO, що розведені на контактні майданчики;
- має такий же форм-фактор як плата Blue Pill і роз’єм DIP-40;

- два кварци – 25 МГц для тактування ядра та 32768 Гц для тактування RTC;
- стабілізатор напруги 3,3 В для забезпечення можливості живлення плати від 5В;
- роз’єм для підключення SWD програматора;
- роз’єм USB Type-C (з’єднаний з апаратним USB інтерфейсом);
- два світло-діоди: червоний - наявність живлення, зелений – підключений до GPIO C13;
- три кнопки: NRST (для перезавантаження), BOOT0 (для переведення МК в режим завантаження прошивки), KEY (можливо використовувати в своїй програмі).

Завантажувати прошивку у МК можна через програматор SWD, наприклад ST-Link, або через USB-UART конвертер. Якщо ж прошити в чіп спеціальний завантажувач, то з’явиться можливість прошивки МК через вбудований роз’єм USB Type-C.

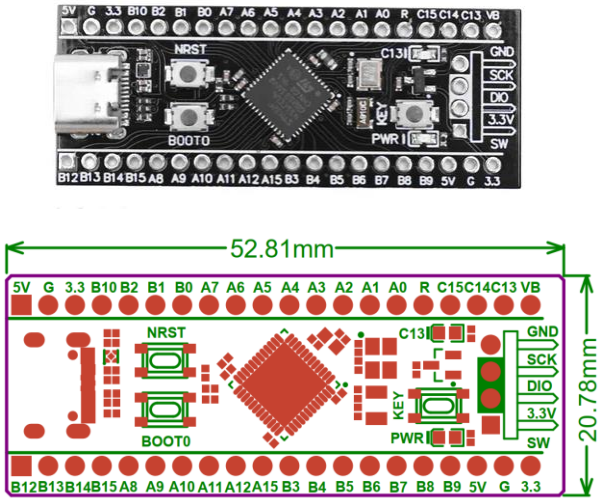

Рисунок 2.2 – Плата для розробки STM32F401CCU6

Характеристики МК STM32F401CCU6 представлені в таблиці 2.1.

Таблиця 2.1 – Характеристики МК STM32F401CCU6 [8]

Використовуваний МК	STM32F401CCU6
Ядро	ARM 32-bit Cortex-M4
Напруга живлення плати	2,7…5 В
Максимальна тактова частота	84 МГц
Об’єм флеш пам’яті	256 Кб
Об’єм оперативної пам’яті SRAM	64 Кб
Кількість портів GPIO	36
Кількість АЦП	1x 12bit (16 каналів)
Діапазон робочих температур	-40°C..+125°C
Комунікаційні інтерфейси:	до 3-х I^2C до 3-х USART до 4-х SPI до 2× I^2S 1× USB OTG

2.2.2 Сенсори для вимірювання температури та вологості

Для даного проєкту було обрано для використання датчики від компанії Bosch Sensortec - BME280.

Модуль датчика BME280 зображений на рис. 2.3 – це датчики, що дозволяють вимірювати не лише значення атмосферного тиску, а й температуру та вологість. Датчик характеризується високою точністю вимірювання, високою швидкодією інтерфейсу та малим споживанням. Для підключення використовується шина I^2C.

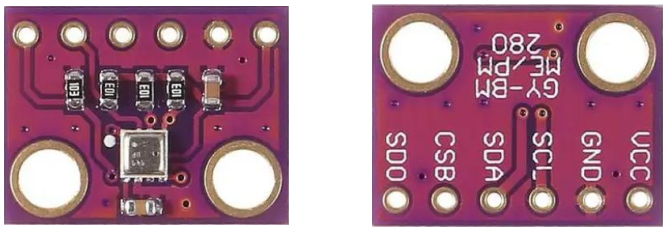

Рисунок 2.3 – Датчик температури, вологості та тиску BME280

Характеристики датчика температури, вологості та тиску BME280 представлені в табл. 2.2.

Таблиця 2.2 – Характеристики датчика BME280 [9]

Інтерфейс підключення	I^2C
Максимальна швидкодія інтерфейсу I^2C	до 3.4 МГц
Межі вимірювання температури	від -40º до 85º
Точність вимірювання температури	від 0.5º до 1º
Межі вимірювання вологості	від 0 до 100%
Точність ви вимірювання міру вологості	3%
Межі вимірювання тиску	від 300 до 1100 гПа
Точність вимірювання тиску	1 гПа
Напруга живлення	від 1,8 до 5 В
Споживаний струм у режимі вимірювання тиску	714 мкА
Споживаний струм у режимі вимірювання вологості	340 мкА
Споживаний струм в режимі вимірювання температури	350 мкА
Споживаний струм у режимі сну	від 0.1 мкА до 0.5 мкА
Розміри модуля	15 х 12 х 3 мм

Датчик може перебувати у трьох режимах: у нормальному (NORMAL MODE), у сплячому (SLEEP MODE) та у примусовому (FORCE MODE).

У нормальному режимі після зчитування значень із датчика останній залишається в цьому режимі і далі, а у примусовому, коли ми зчитуємо значення, датчик автоматично переходить у режим сну. І щоб нам знову зчитувати показники, нам треба заново ініціалізувати або нормальний або примусовий режими.

Для уникнення похибки у показах датчиків, при їх встановленні необхідно дотримуватись наступних правил:

- встановлювати датчики у тіні та уникати потрапляння прямих сонячних променів;

- датчики не повинні знаходитися поряд з вікнами, дверима, повітропроводами та іншими джерелами тепла.

Зовнішній датчик можна розмістити у водонепроникний корпус, приклад якого зображено на рис. 2.4.

Рисунок 2.4 – Водонепроникний корпус датчика температури та вологості

2.2.3 Дисплей

Було вибрано LCD дисплей ILI9341. Він має технічні характеристики що приведені в табл. 2.3.

Таблиця 2.3 – Характристики дисплея ILI9341

Можливі варіанти діагоналі	1.44, 1.8, 2.0, 2.2, 2.4, 2.8, 3.2, 3.5, 4.0 дюймів
Розширення	240x320
Тип матриці	TFT
Інтерфейс	SPI
Драйвер	ILI9341
Живлення	3.3 В
Особливості	Вбудований інтерфейс SD карти

Зовнішній вигляд дисплея зображено на рис. 2.5.

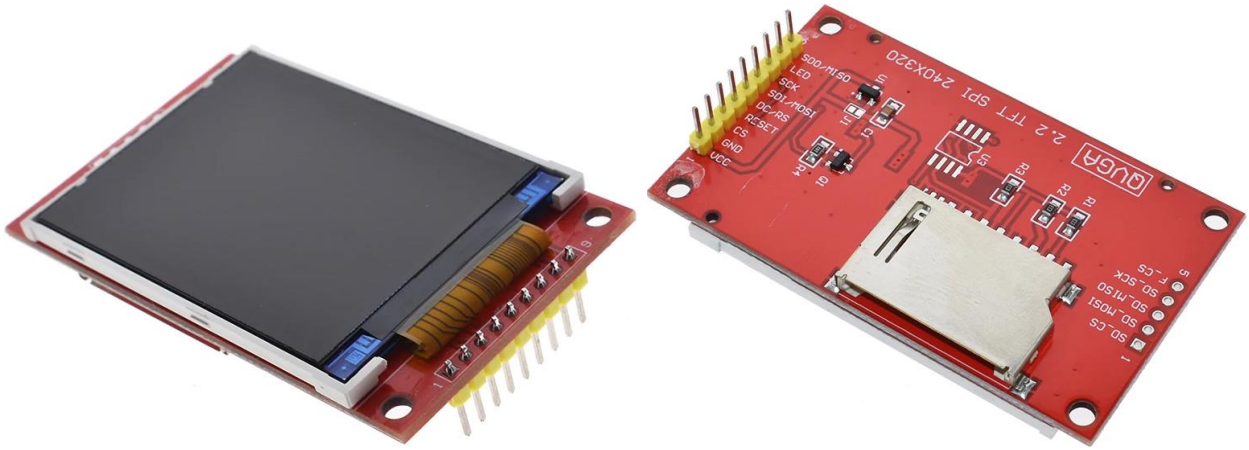

Рисунок 2.5 – Зовнішній вигляд дисплея ILI9341

2.2.4 Реле

Для комутації споживачів (вентилятори, обігрівач, сервопривід) бажано застосовувати реле, які спеціально розроблені для використання з мікроконтролерами (рисунок 2.6). Схема таких реле має гальванічну розв’язку з живленням. Що значно знижує навантаження на стабілізатор напруги 3,3 В, так як саме реле живиться від напруги 5 В чи 12 В (відповідно до специфікації реле).

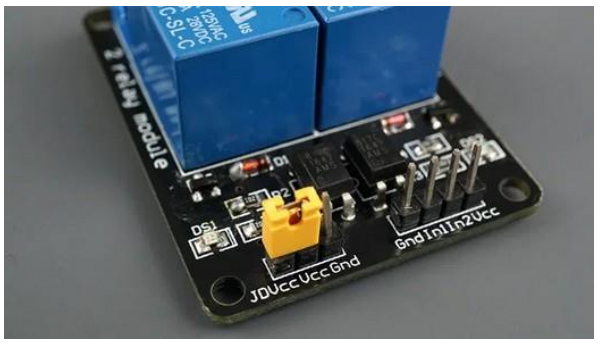

Рисунок 2.6 – Приклад зовнішнього вигляду реле для застосування з мікроконтролерами

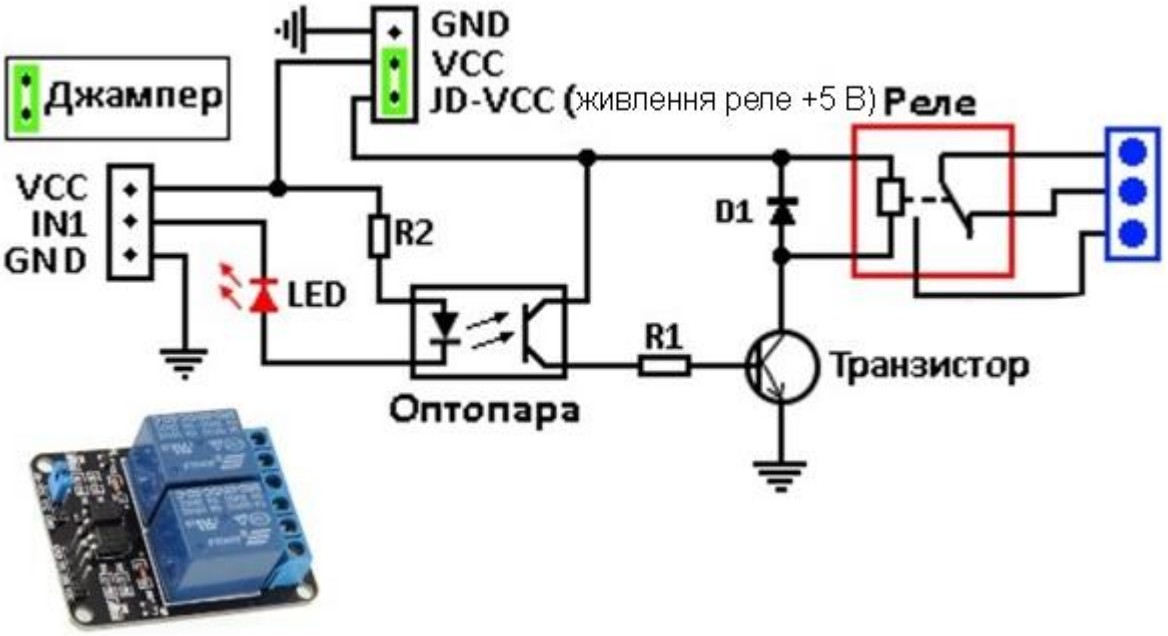

Рисунок 2.7 – Принципова схема для прикладу одного каналу реле

Для підключення до МК STM32 необхідно видалити джампер. На вивід GND необхідно подати мінус від джерела живлення для реле (-5 В або -12 В). На контакт JD-VCC необхідно подати плюс від джерела живлення для реле (+5 В). Вивід VCC необхідно підключити до +3,3 МК STM32. Вивід IN1(2) необхідно підключити до потрібного порту МК STM32. Таким чином весь модуль, у тому числі і реле, будуть заживлені від окремого джерела живлення (5 В), а з МК STM32 приходитиме лише 3,3 В: на резистор R2, керуючий світлодіод оптопари та сигнальний світлодіод (рис. 2.7).

Реле здатне комутувати споживачів з наступними характеристиками:

- Максимальна напруга: 250 В;
- Максимальний струм: 10 А;
- Максимальна потужність: 2500 Вт.

2.2.5 Сервоприводи та заслінки

Можливо використати сервопривід EMAX ES08A II (рис. 2.8) - мініатюрний та високочутливий сервопривід із нейлоновим редуктором. Підходить для невеликого моделювання з наступними характеристиками:

- Робоча напруга: 4.8В ~ 6.0В
- Зусилля утримання (при 4.8В): 1.5 кг / см
- Робоча швидкість (при 4.8В): 0.12 сек / 60° без навантаження
- Зусилля утримання (при 6.0В): 1.8 кг / см
- Робоча температура: 0 °C - 55 °C
- Матеріал шестерні: нейлон (пластик)
- Стандартний роз'єм: Futaba / JR універсальний
- Тип двигуна: Колекторний
- Розміри: 23 x 11.5 x 24 мм
- Вага: 8.5 грам

Рисунок 2.8 – Сервопривід EMAX ES08A II

Повітряна заслінка використовується в двопозиційному режимі керування «відкрито/закрито», а повітряний клапан – ще для пропорційного регулювання потоку повітря.

Вентиляційні заслінки та клапани можуть бути прямокутного перерізу та круглого перерізу, зображені на рис. 2.9.

Рисунок 2.9 – Повітряні клапани з круглим та поперечним перерізом

Повітряну заслінку можливо використати зі змонтованим сервоприводом як зображено на рис. 2.10 [10].

Рисунок 2.10 – Повітряна заслінка в пластиковому виконанні

Також можливо використовувати металеві заслінки з сервоприводами як на рис. 2.11 [11].

Рисунок 2.11 – Металева заслінка з сервоприводом

2.2.6 Вентилятори

Вентилятором називають механічний пристрій для переміщення під певним тиском повітря або газів. Приводом служать електричні двигуни, а потік повітря переміщують лопаті, які мають різну конфігурацію і габарити. Види вентиляторів можуть поділятись за конструкцією, габаритами та потужністю двигуна.

В даному випадку не потрібні вентилятори великої потужності. Можна обрати невеликий осьовий вентилятор, як зображено на рис. 2.12.

Рисунок 2.12 – Осьовий вентилятор

2.2.7 Фільтри

В системах вентиляції використовуються: панельні, компактні фільтри, фільтри з металевих сіток, фільтри тонкого очищення HEPA та вугільні фільтри. Фільтри необхідно підібрати враховуючи вимоги зберігання лікарських засобів, що зберігаються.

2.2.8 Електричні обігрівачі

Електричні обігрівачі призначені для нагрівання не запиленого повітря. На рис. 2.13 зображено приклад електричного обігрівача повітря круглого перерізу потужністю 0,6 кВт.

Рисунок 2.13 – Електричний канальний обігрівач повітря потужністю 0,6 кВт

2.3 Проєктування електричної принципової схеми

2.3.1 Середовище проєктування електричних схем

Для розробки електричної принципової схеми комп’ютеризованої системи контролю за умовами зберігання лікарських засобів було обрано середовище проєктування EasyEDA. Цей програмний засіб призначений для автоматизації процесу розробки електричних схем та створення монтажних плат.

Основним критерієм при виборі цього програмного засобу є те що він повністю безкоштовний для як для персонального використання, так і для комерційного за виключенням деяких платних опцій.

2.4 Проєктування комп’ютерного засобу

2.4.1 Вибір алгоритму «осушення»

Алгоритм процесу осушення полягає в тому, щоб виміряти температури (t, [°C]) та відносні вологості (h, [%]) у шафі та ззовні (на вулиці). На підставі

виміряних значень необхідно розрахувати абсолютні вологості (H_{abs}, [г/м^3]). Порівнюючи розраховані значення приймається рішення про включення вентиляторів і відкриття засувок чи навпаки: вимкнення вентиляторів і закриття засувок.

Якщо відомі температура (t, [°C]), атмосферний тиск (p, [гПа]), відносна вологість (h, [%]), то абсолютну вологість (H_{abs}, [г/м^3]) будемо розрахувати за формулою:

$$H_{abs}(p, t) = 10^5 \cdot P_{par} / (461{,}5 \cdot (t + 273{,}15)), \quad (2.1)$$

де:

$$P_{par} = h \cdot e_\omega'(p, t), \quad (2.2)$$

h - відносна вологість,

$e_\omega'(p, t)$ - насичений тиск водяної пари у вологому повітрі, розраховується за формулою:

$$e_\omega'(p, t) = f(p) \cdot e_\omega(t), \quad (2.3)$$

$e_\omega(t)$ - насичений тиск чистої фази водяної пари, розраховується за формулою:

$$e_w(t) = 6.112 \cdot e^{\frac{17.62t}{243.12+t}} \quad (2.4)$$

$f(p)$ - функція від значення атмосферного тиску розраховується за формулою:

$$f(p) = 1.0016 + 3.15 \cdot 10^{-6}p - 0.074 \cdot p^{-1} \quad (2.5)$$

Описаний алгоритм повинен виконуватися по перериванню таймера кожні 3 секунди.

2.4.2 Розробка схеми підключення сенсорів до мікроконтролера

Вибір портів GPIO для рішення задачі проєкту в табличному вигляді зображено в табл. 2.4.

Таблиця 2.4 – Розподілення портів GPIO

Змінна в програмі	Порт STM32	Порт пристрою
Датчики BME20 підключено до комунікаційного інтерфейсу I2C1		
I2C1_SCL	PB6	SCL датчиків BME280
I2C1_SDA	PB7	SDA датчиків BME280
TFT дисплей підключено до комунікаційного інтерфейсу SPI1		
LCD_DC	PA2	DC дисплея
LCD_CS	PA4	CS дисплея
LCD_SDA	PA7	SDI дисплея
LCD_RES	PA3	Reset дисплея
LCD_SCL	PA5	SCK дисплея
LCD_BLC	PA6	підсвідка дисплея
Зчитувач SD-карток підключено до комунікаційного інтерфейсу SPI2		
SD_CS	PB12	CS зчитувача SD-карток
SD_CLK	PB13	CLK зчитувача SD- карток
SD_MISO	PB14	MISO зчитувача SD- карток
SD_MOSI	PB15	MOSI зчитувача SD- карток
Кнопки керування пристроєм		
KEY_LEFT	PB4	кнопка вліво (Меню)
KEY_RIGHT	PB5	кнопка праворуч (OK)
KEY_UP	PB8	кнопка вгору
KEY_DOWN	PB9	кнопка вниз
Керування сервоприводами повітряних заслінок		
TIM2_CH1	PA0	повітряна заслінка №1
TIM2_CH2	PA1	повітряна заслінка №2
Керування реле		
PIN_RELAY	PB1	реле подачі живлення на вентилятори
PIN_SRV_PWR	PB2	реле подачі живлення на сервоприводи
PIN_HEAT	PB0	реле подачі живлення на обігрівач повітря
Вбудований світлодіод		
SD_LED	PC13	Світлодіод на МК світиться при операціях з SD-картою

2.4.3 Опис електричної принципової схеми

Електрична схема системи розроблена на базі структурної схеми системи (див. рис. 2.1). Центральним компонентом схеми є платформа STM32, яка позначена на схемі U1. Напруга +5 В поступає від стандартного блока живлення.

На рис. 2.14 зображено електричну принципову схему системи контролю за умовами зберігання лікарських засобів.

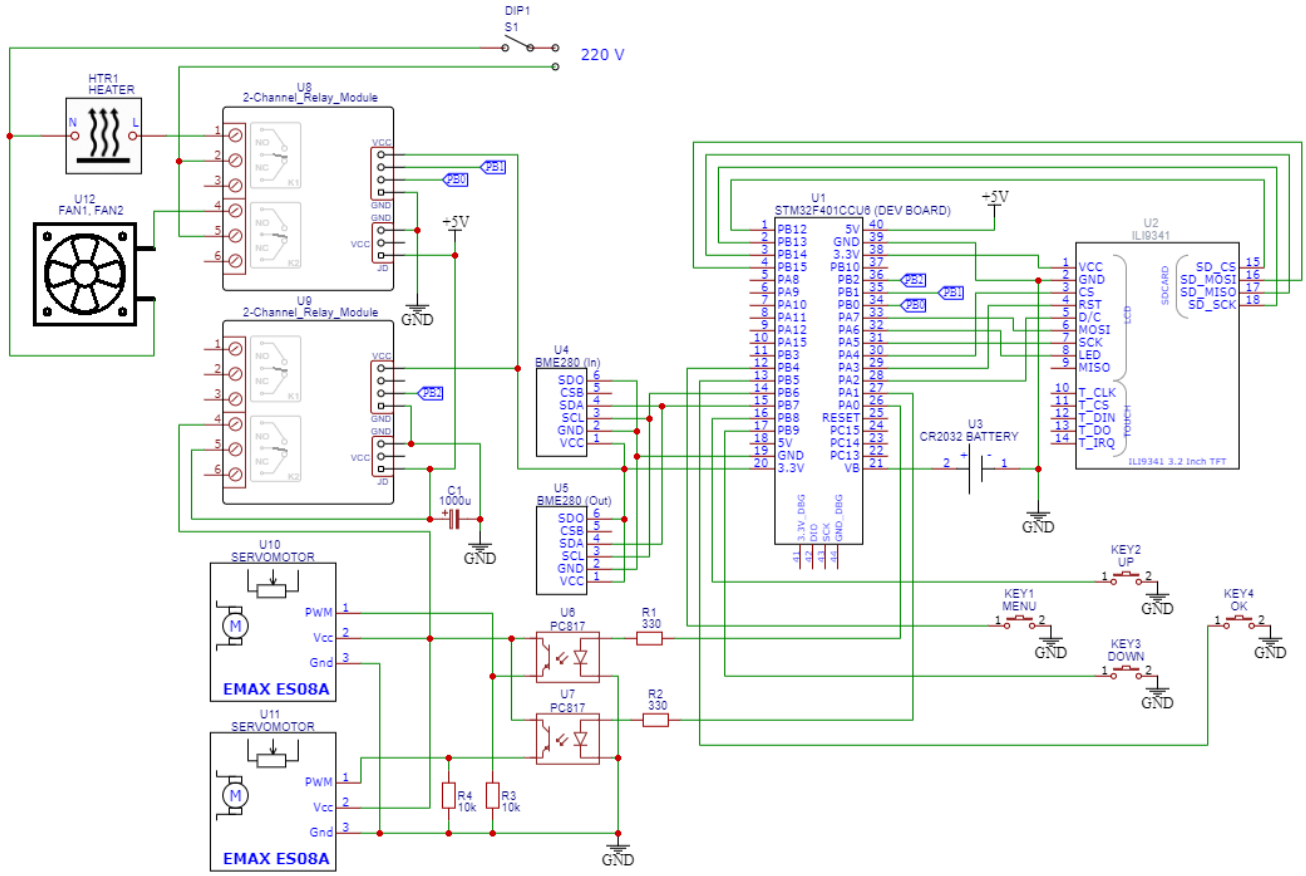

Рисунок 2.14 – Електрична принципова схема системи контролю за умовами зберігання лікарських засобів

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Реалізація і моделювання проєктних рішень

На рис. 3.1, 3.2 зображено блок-схему алгоритму роботи програми для МК проєктованої системи.

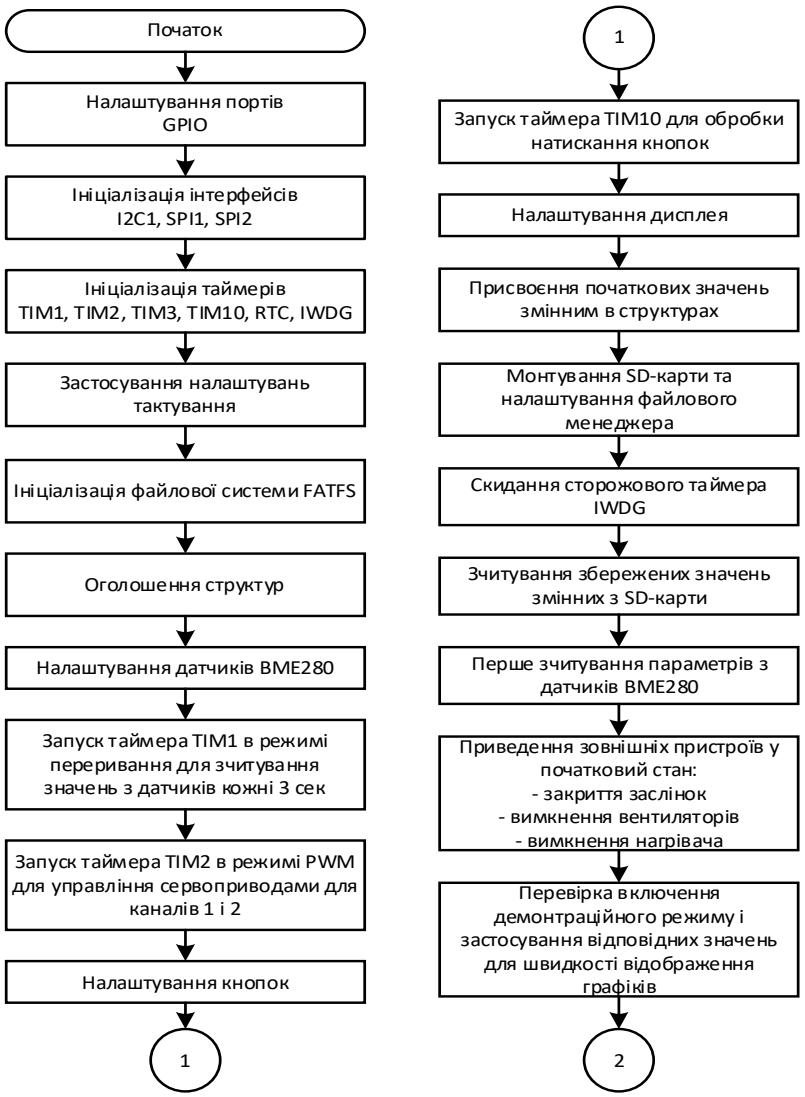

Рисунок 3.1 – Алгоритм роботи системи контролю за умовами зберігання лікарських засобів

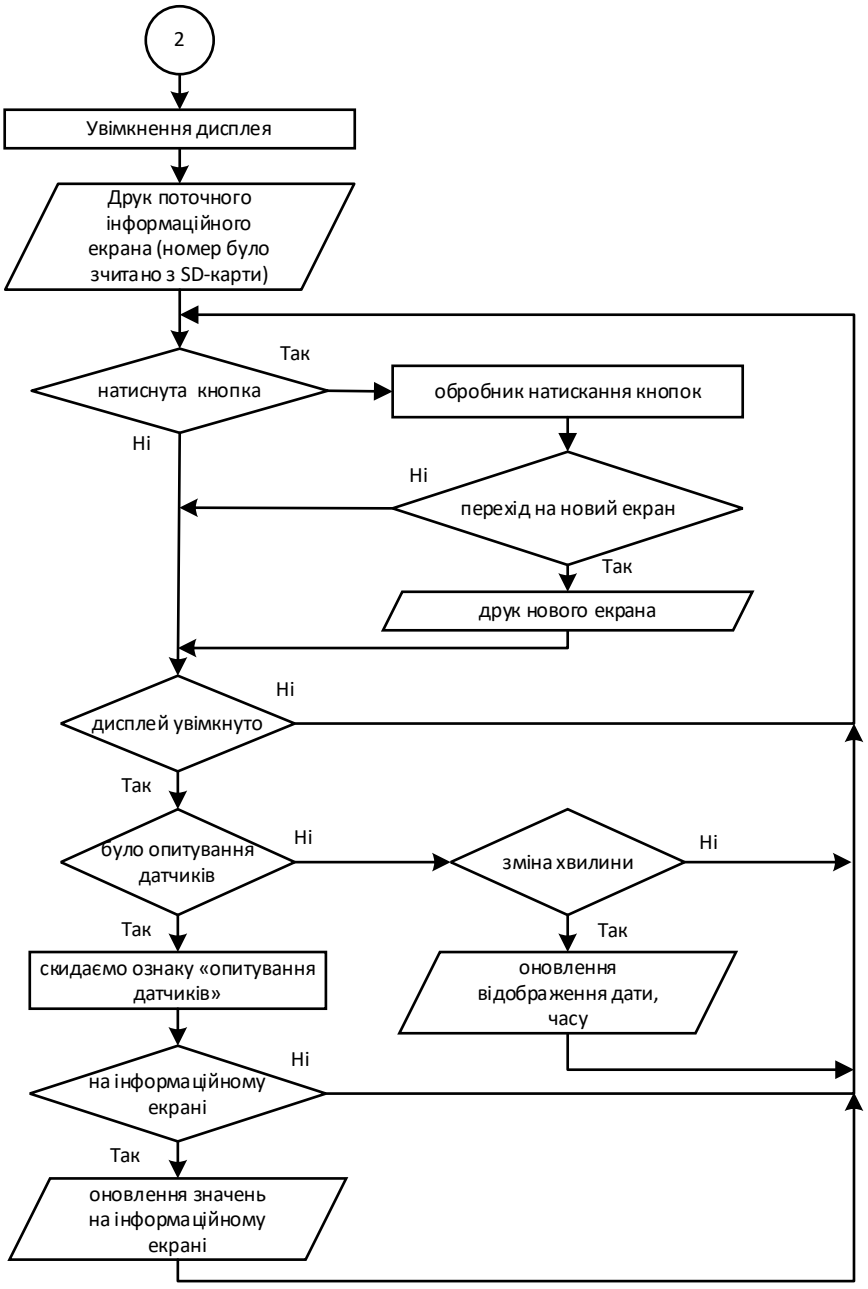

Рисунок 3.2 – Алгоритм роботи системи контролю за умовами зберігання лікарських засобів (продовження)

Опис роботи алгоритму приведено в п. 3.1.2 «Опис програми».

3.1.1 Налаштування середовища для розробки програмного забезпечення

Програма розроблялась в середовищі STM32CudeIDE. Для написання програми застосовувалась мова програмування С. Після аналізу доступних

джерел в Інтернет було вибрано бібліотеки та модулі для дисплея, кнопок, SD-зчитувача з джерел: [12, 14, 15, 16], які було додано до проєкту, створеного в середовищі STM32CudeIDE.

Перед створенням програми в середовищі STM32CudeIDE було увімкнено та налаштовано:

1. Необхідні комутаційні інтерфейси:
   - I2C1 для підключення датчиків BME280;
   - SPI1 для підключення дисплея;
   - SPI2 для підключення зчитувача SD-карт.
2. Порти GPIO у відповідності з табл. 2.4.
3. Необхідні таймери:
   - TIM1 для вимірювання датчиками BME280 параметрів оточуючого середовища кожні 3 секунди;
   - TIM2, канал 1, канал 2 в режимі генерації широтно-імпульсної модуляції (ШІМ – англ. pulse-width modulation, PWM) для керування положенням сервоприводів;
   - TIM3, канал 1 в режимі генерації ШІМ для керування яскравістю дисплея;
   - TIM10 для обробки натискання кнопок;
   - RTC, годинник реального часу.
4. Сторож IWDG для виявлення зависання МК і ініціалізації його перевантаження.

3.1.1.1 Налаштування тактування мікроконтролера

Система управління тактуванням в середовищі STM32CudeIDE називається RCC (reset and clock control). Для отримання максимально можливої частоти МК виконано налаштування RCC. В першу було вибрано кварци для генераторів тактових сигналів. Для двох генераторів вибрано

зовнішні кварци: High-speed external (HSE) та Low-speed external (LSE), так як вини мають кращу стабільність. Крім того, для коректної і точної роботи годинника реального часу (RTC) необхідно вибирати зовнішній кварц LSE.

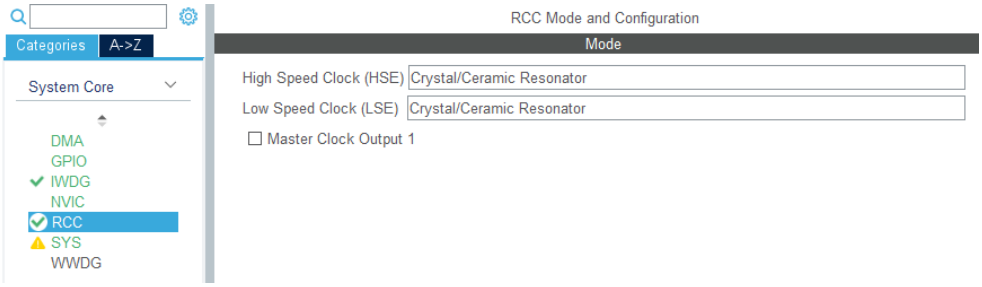

Рисунок 3.3 – Налаштування RCC

Налаштовано RTC (годинник та календар), як показано на рис. 3.4.

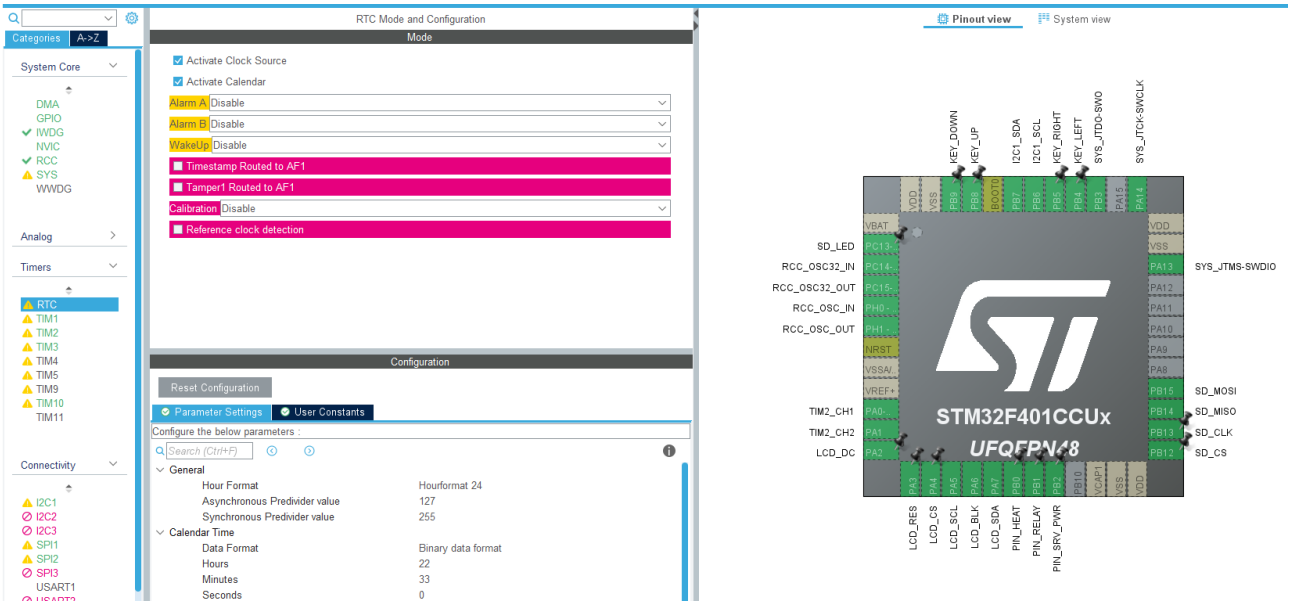

Рисунок 3.4 – Налаштування RTC

Результат налаштування тактування зображено на рис. 3.5.

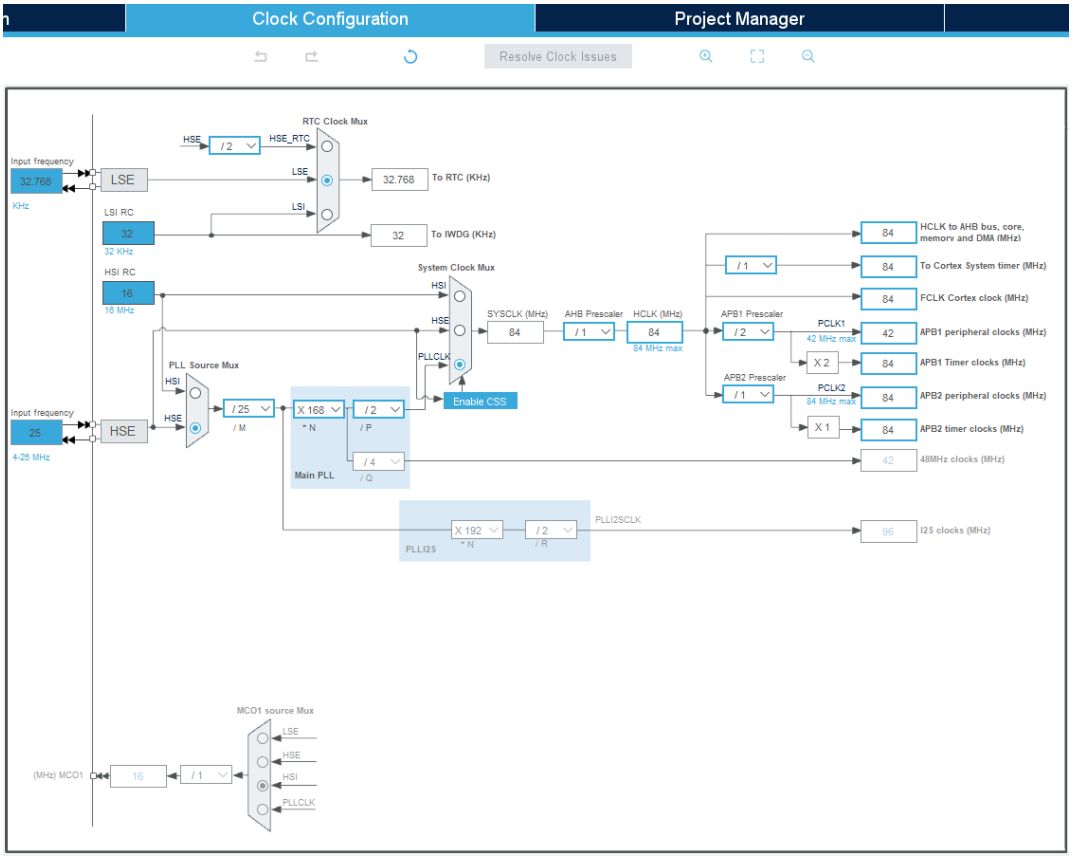

Рисунок 3.5 – Налаштування тактування

3.1.1.2 Налаштування комунікаційних інтерфейсів

Для підключення датчиків температури і вологості BME280 увімкнуто комунікаційний інтерфейс I^2C, як зображено на рис. 3.6.

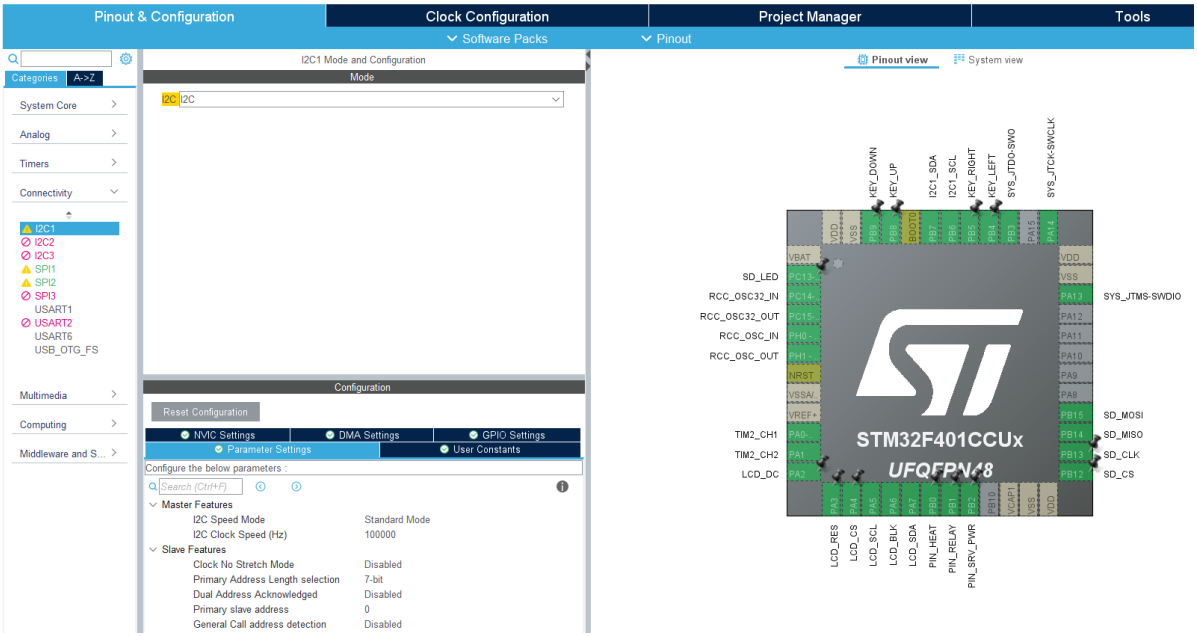

Рисунок 3.6 – Увімкнння та налаштування шини I²C

Для підключення дисплея налаштувано перший комунікаційний інтерфейс SPI, як зображено на рис. 3.7 - 3.9.

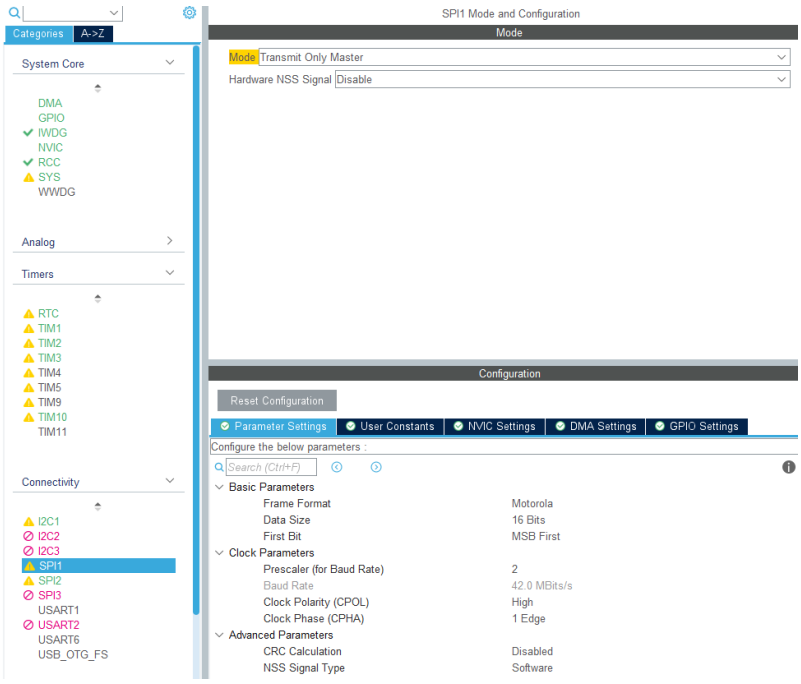

Рисунок 3.7 – Увімкнння шини SPI1

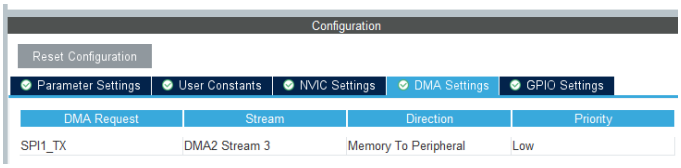

Рисунок 3.8 – Налаштування шини SPI1

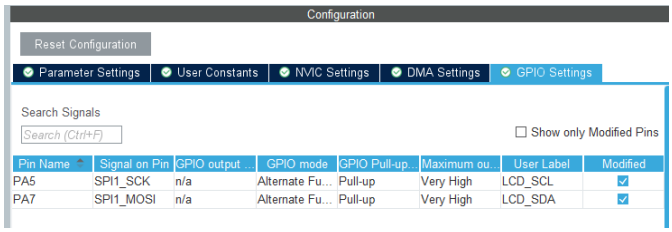

Рисунок 3.9 – Налаштування шини SPI1

Для підключення зчитувача SD-карток налаштувано другий комунікаційний інтерфейс SPI, як зображено на рис. 3.10 - 3.12.

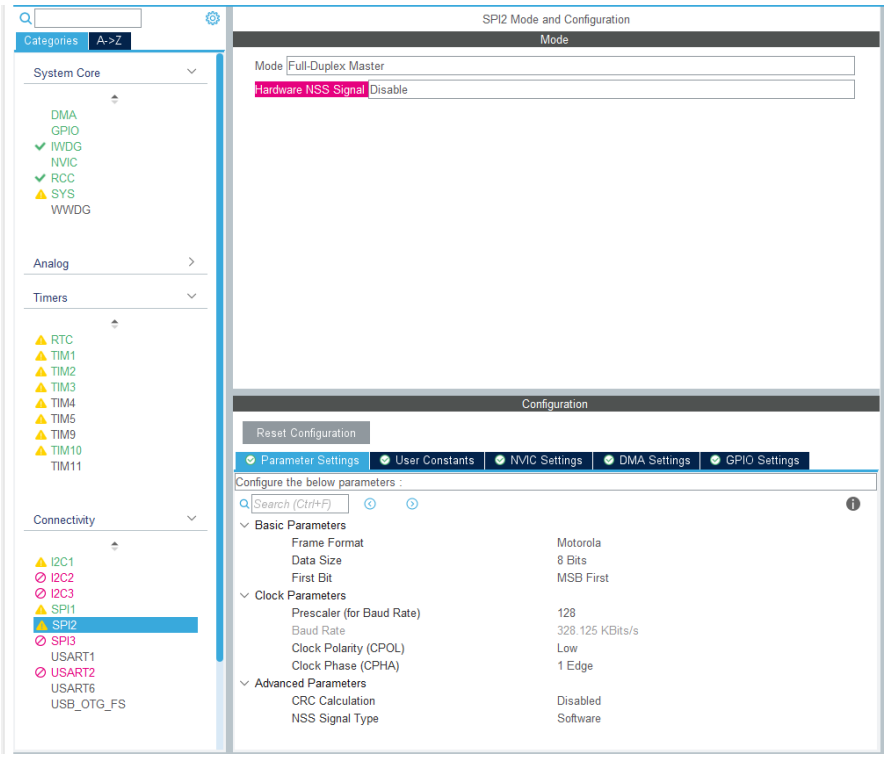

Рисунок 3.10 – Увімкнння шини SPI2

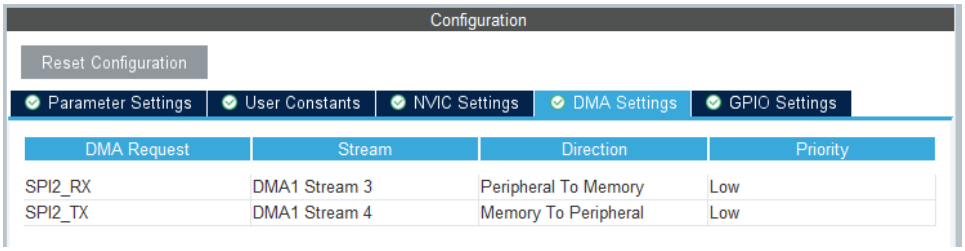

Рисунок 3.11 – Налаштування шини SPI2

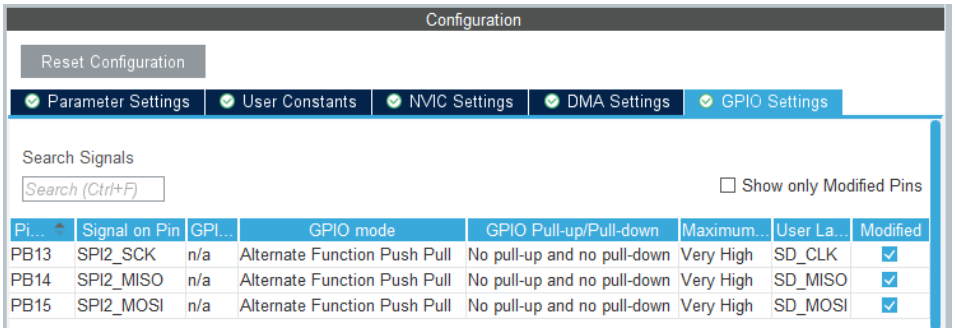

Рисунок 3.12 – Налаштування шини SPI2

3.1.1.3 Налаштування портів GPIO

Налаштування портів вводу-виводу GPIO виконано у відповідності з табл. 2.4. Результат налаштування відображено на рис. 3.13 - 3.16.

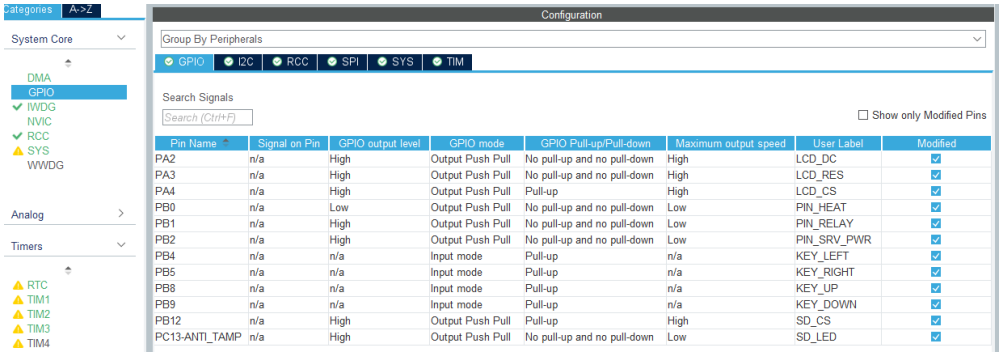

Рисунок 3.13 – Налаштування GPIO

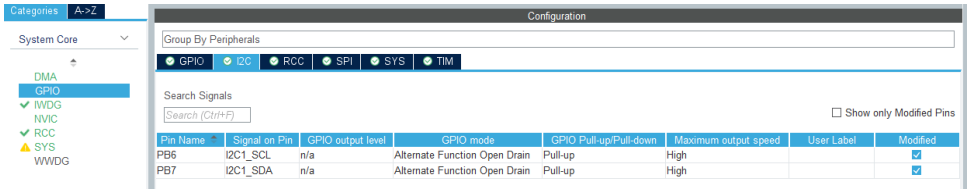

Рисунок 3.14 – Налаштування портів I^2C

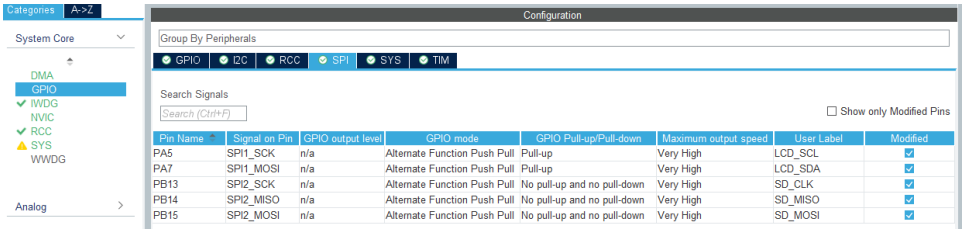

Рисунок 3.15 – Налаштування портів SPI

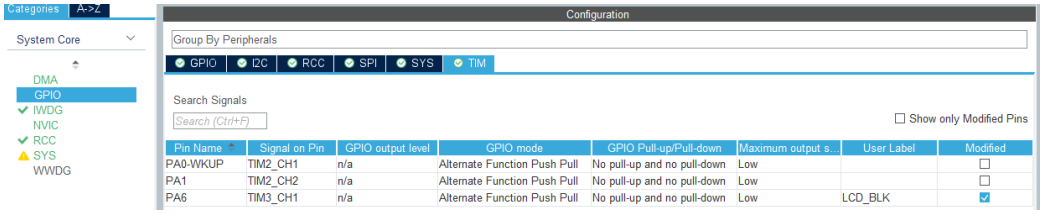

Рисунок 3.16 – Налаштування портів, що використовуються таймерами

3.1.1.4 Налаштування таймерів

3.1.1.4.1 TIM1

За допомогою таймера TIM1 реалізується періодичність вимірювання датчиками BME20 параметрів оточуючого середовища кожні 3 секунди. Виконані налаштування зображено на рис. 3.17, 3.18.

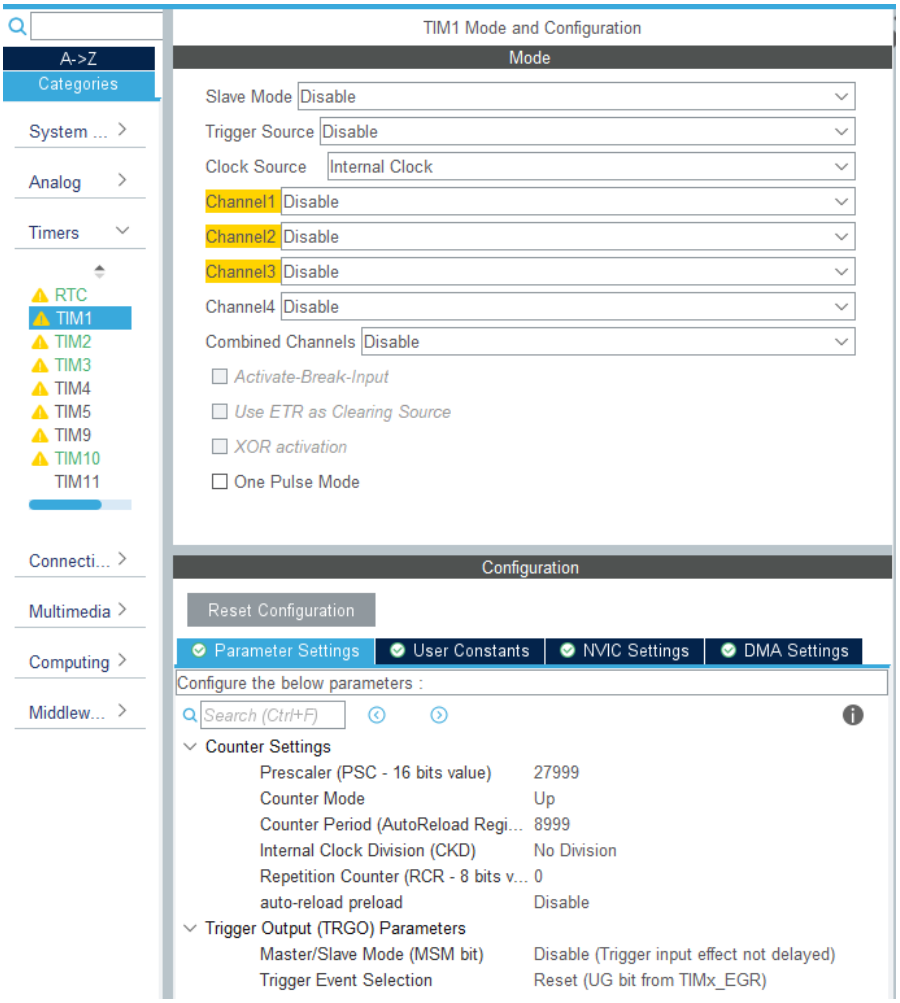

Рисунок 3.17 – Налаштування таймера TIM1

Розрахунок налаштувань для таймера TIM1:

- частота МК 84 МГц = 84 000 000 Гц;
- вибрано частоту лічильника: 3 кГц = 3 000 Гц;
- щоб її отримати: Prescaler = 84 000 000 / 3 000 = 28 000;
- але потрібно зменшити на 1, бо нумерація з 0. Тобто, Prescaler = 27999;
- Розраховано період для частоти 3 кГц : T = 1 / 3000 с.

Щоб отримати переривання кожні 3 с розраховано значення «Counter Period»:

- Counter Period = 3 / T = 9 000;

- але потрібно зменшити на 1, бо нумерація починається з 0. Тобто, Counter Period = 8999.

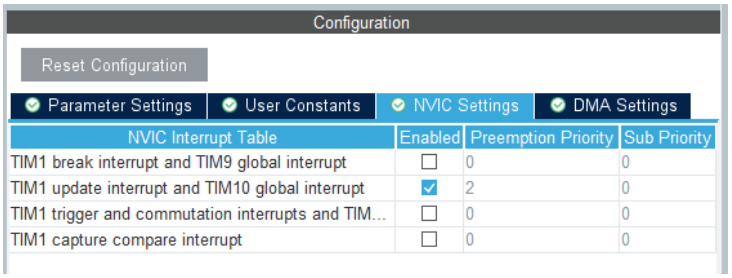

Рисунок 3.18 – Налаштування використання переривання таймером TIM1

Було відмічено ознаку, що будемо користуватись перериванням по таймеру TIM1, як показано на рис. 3.18.

В програмному коді додано необхідні строки, як показано в лістингу на рис. 3.19.

```
/* USER CODE BEGIN 2 */

// ТАЙМЕР htim1

  // очищаємо ознаку, щоб перший раз таймер не запустився відразу з запуском МК
    __HAL_TIM_CLEAR_FLAG(&htim1, TIM_SR_UIF);

//запускаємо таймер htim1 в режимі переривання для зчитування значень з датчиків
  HAL_TIM_Base_Start_IT(&htim1);
```

Рисунок 3.19 – Лістинг початкових налаштувань таймера TIM1

Функція обробника переривань таймера (яка запускається перериванням таймера): void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim).

3.1.1.4.2 TIM2

За допомогою каналів 1 та 2 таймера TIM2, які працюють в режимі генерації широтно-імпульсної модуляції (ШІМ – англ. pulse-width modulation,

PWM), реалізується керування положенням сервоприводів. Виконані налаштування зображено на рисунку 3.20.

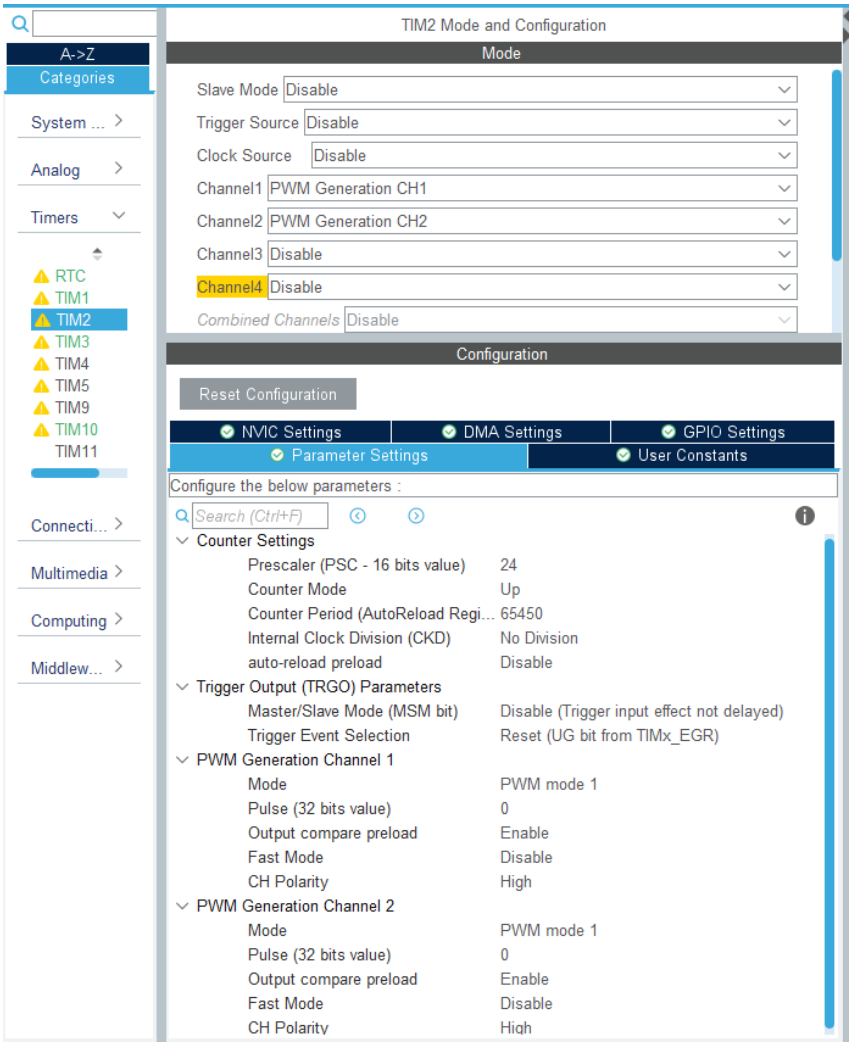

Рисунок 3.20 – Налаштування таймера TIM2

На керуючий вивід сервопривода необходно подавати ШІМ сигнал з частотою близько 50 Гц.

Для розрахунку необхідної частоти ШІМ застосовується наступна формула:

$$\text{Prescaler} = \frac{\text{Timer clock}}{\text{Frequency} * \text{Counter period}}, \quad (3.1)$$

де Timer clock = 84 000 000 Гц, Frequency = 50 Гц. Для більшої точності керування вибрано Counter period = 65451.

Отримано Prescaler = 25 (так як нумерація починається з 0, то в STM32CudeIDE встановлюємо 24).

В програмному коді додано необхідні строки, як вказано у лістингу на рис. 3.21.

```
// ТАЙМЕР htim2
//запускаємо таймер2 в режимі PWM (для управління сервоприводами) для каналів 1 і 2
HAL_TIM_PWM_Start_IT(&htim2, TIM_CHANNEL_1);
HAL_TIM_PWM_Start_IT(&htim2, TIM_CHANNEL_2);
```

Рисунок 3.21 – Лістинг запуску таймера htim2 в режимі PWM для каналів 1 і 2

3.1.1.4.3 TIM3

За допомогою каналу 1 таймера TIM3, який працює в режимі ШІМ, реалізується керування яскравістю дисплея. Необхідні налаштування зображено на рис. 3.22.

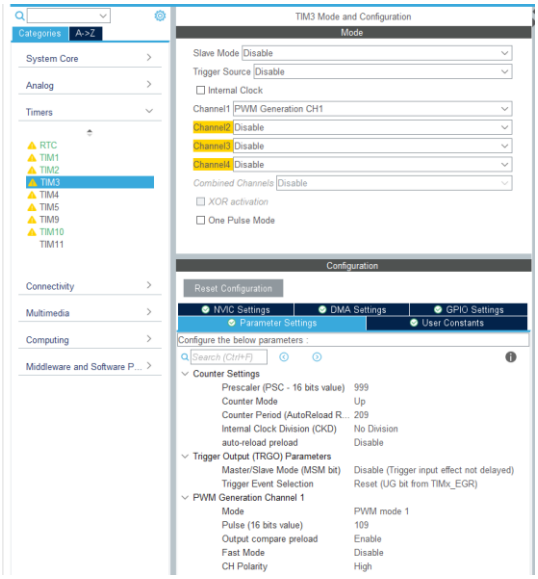

Рисунок 3.22 – Налаштування таймера TIM3

3.1.1.4.4 TIM10

Таймер TIM10 використовується для обробки натискання кнопок. Необхідні налаштування таймера для частоти 200 Гц зображено на рис. 3.23, 3.24. Використання такого значення частоти запобігає винекненню брязкоту контактів кнопок.

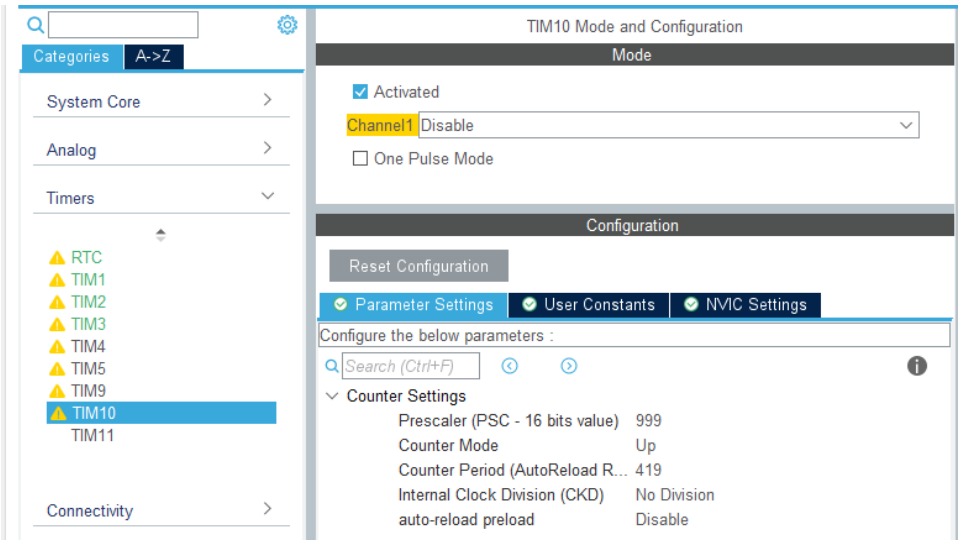

Рисунок 3.23 – Налаштування таймера TIM10

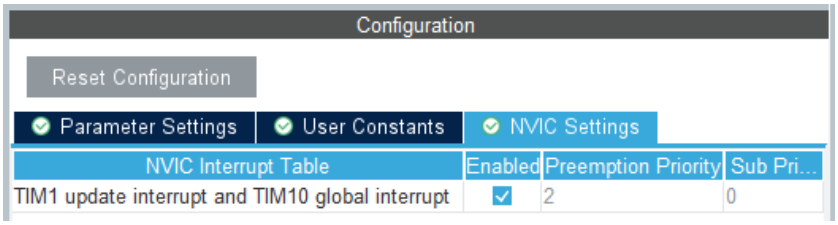

Рисунок 3.24 – Налаштування використання переривання таймером TIM10

3.1.1.4.5 IWDG

Так як в системі використовується комунікаційний інтерфейс I^2C, то при будь-якому збої в цій шині можливе зависання програми. В цьому випадку необхідне примусове перевантаження МК. Для реалізації цієї можливості

увімкнуто незалежний сторожовий таймер Independent watchdog (IWDG), як показано на рис. 3.25.

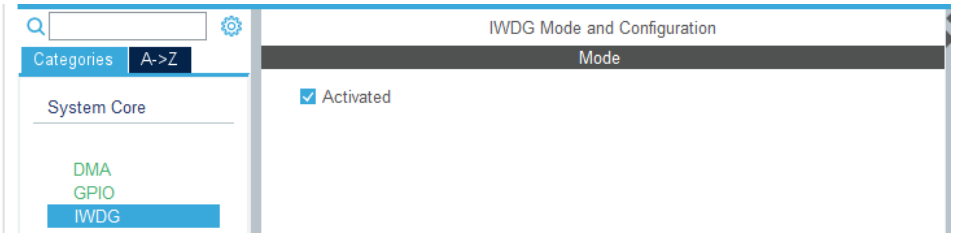

Рисунок 3.25 – Увімкнння IWDG

Сторожовий таймер IWDG необхідно налаштувати. Так як IWDG тактується від окремого джерела імпульсів LSI частотою 32 кілогерц, то встановлено у сторожовому таймері налаштування як показано на рис. 3.26.

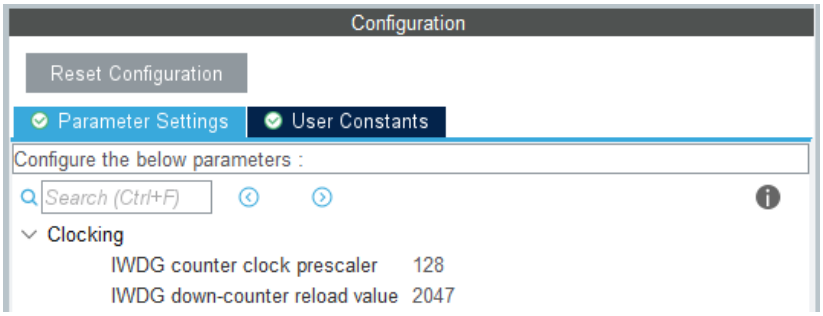

Рисунок 3.26 – Налаштування IWDG

Для перевірки розділимо на ці значення частоту тактування 32000, отримаємо приблизно 0,122. Це буде значення частоти спрацьовування таймера. Якщо ми знайдемо зворотне число від цього значення, отримаємо приблизно 8,2. Це буде значення інтервалу в секундах від пуску сторожового таймера IWDG до того, як він перезавантажить систему.

Щоб запобігти незапланованому перевантаженню МК в програму (у фунцію яка по таймеру TIM1 кожні 3 секунди зчитує значення з датчиків

BME280) дадано код, який буде періодично скидати сторожовий таймер: HAL_IWDG_Refresh(&hiwdg);

3.1.1.5 Підключення можливості читання/запису файлів

Для читання/запису файлів з/на SD-карту у середовищі STM32CudeIDE налаштовано цю можливість, як показано на рис. 3.27.

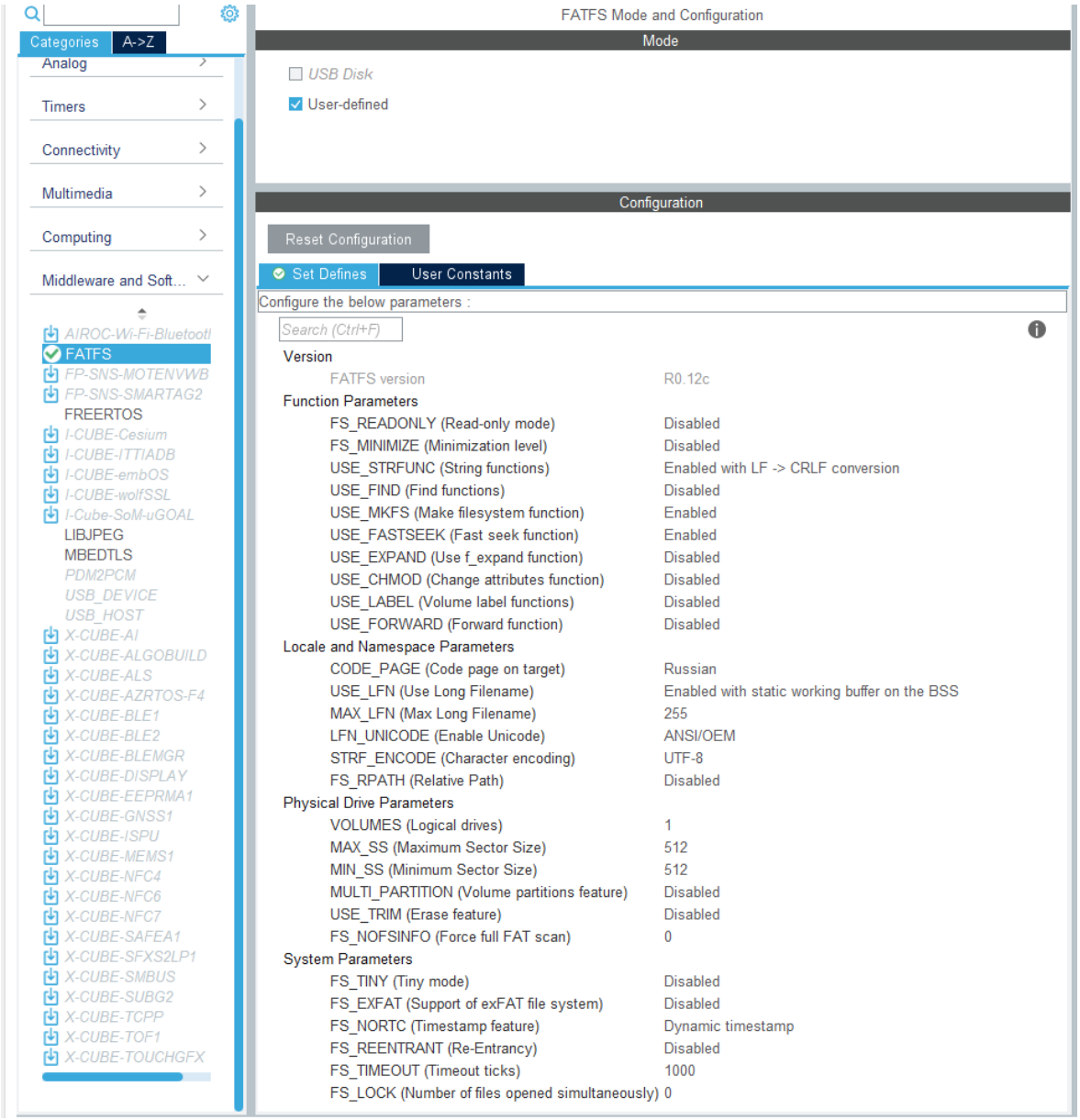

Рисунок 3.27 – Налаштування FATFS

3.1.2 Опис програми

Частина коду програми була згенерована середовищем STM32CudeIDE враховуючи вказані вище виконані налаштування.

Основною функцією в програмі є main(). В ній спочатку ініціалізуються налаштовані порти GPIO, необхідні комунікаційні інтерфейси, таймери, змінні та структури що були запрограмовані. Також виконуються необхідні початкові дії: зчитування файлу з SD-карти зі збереженими налаштуваннями, застосування цих налаштувань. Далі всі основні дії програми виконуються в безкінечному циклі скануючи стан кнопок і обробляючи їх натискання. Також кожні три секунди по перериванню таймера tim1 виконується функція HAL_TIM_PeriodElapsedCallback в якій реалізовано вимірювання параметрів зовнішнього середовища за допомогою двох датчиків BME280 (температура, вологість, тиск) в середині шафи і ззовні. В залежності від вибраного режиму роботи, виконується управління зовнішніми механізмами: вентиляторами, повітряними заслінками, обігрівачем повітря. Також постійно збирається статистична інформація з параметрами (температура, вологість): поточні значення зберігаються кожні 12 хвилин, і також визначаються максимальні й мінімальні значення. Вся зібрана інформація періодично зберігається на SD-картку: раз в 3 години для економії ресурсу SD-картки.

Статистичні дані зберігаються у файлах з іменем, що є датою доби, за яку зібрані дані.

В файловому менеджері (рис. 3.43) можливо вибрати необхідний файл й продивитись дані в графічному вигляді, рис. 3.44.

Сам код програми приведений в Додатку В, де надано коментарі щодо призначення створених структур, функцій, змінних.

3.1.2.1 Структури

Для роботи програми було створено структури, перелік і призначення яких приведено в табл. 3.1. Далі представлено опис створених структур.

Таблиця 3.1 - Перелік основних структур, створених в програмі

Структура	Призначення
DataForSave	для збереження даних на SD-карту
DATA_HISTORY	для графіків історії які зберігаються на SD-карту
Packet	для збереження поточних значень
KILCE	для навігації по меню
AVARAGE	для визначення середніх значень показів

3.1.2.1.1 Структура для збереження даних на SD-карту

Структура DataForSave призначена для збереження даних на SD-карту таких як: поточний режим роботи системи, загальний час роботи системи, час роботи вентиляторів, час роботи обігрівача, максимальні/мінімальні значення температур/вологості за період спостереження та час і дати їх фіксації, номер обраної колірної схеми та інших налаштувань, які завантажуються та застосовуються при старті програми. Більш детально інформація про структуру представлена в коментарях лістингу на рис. 3.28.

```
typedef struct DataForSave
{
        int8_t  mode;                   // режим роботи
        uint32_t hour_unit;             // загальний час роботи - вимірюється в інтервалах сканування датчиків = TIME_SCAN_SENSOR
        uint32_t hour_motor;            // час роботи вентиляторів - вимірюється в інтервалах сканування датчиків = TIME_SCAN_SENSOR
        uint32_t hour_heat;             // час роботи обігрівача - вимірюється в інтервалах сканування датчиків = TIME_SCAN_SENSOR
        int16_t  tOutMin, tInMin;       // Мінімальні температури за період спостереження - в сотих градуса
        int16_t  tOutMax,tInMax;        // Максимальні температури за період спостереження - в сотих градуса
        int16_t  hMax, hMin;            // Максимальні/Мінімальні вологість в шафі за період спостереження - в сотих долях %
        int16_t tick_SDflash;           // Змінна для збереження часу що пройшло після останнього збереження статистики,
                                        //      порівнюємо з TIME_HOUR і обнуляємо змінну та зберігаємо статистику

        uint8_t tOutMinTimeDate[5];     //minute,hour,day,month,year
        uint8_t tOutMaxTimeDate[5];     //minute,hour,day,month,year
        uint8_t tInMinTimeDate[5];      //minute,hour,day,month,year
        uint8_t tInMaxTimeDate[5];      //minute,hour,day,month,year
        uint8_t hMaxTimeDate[5];        //minute,hour,day,month,year
        uint8_t hMinTimeDate[5];        //minute,hour,day,month,year

        uint8_t SaveTimeDate[5];        //Час, Дата останнього збереження даних на SD-карту

        uint8_t         flags;  // байт ознак
                                // 0 біт - мотор увімкнено/вимкнено
                                // 1 біт - увімкнено/вимкнено
                                // 2 біт -[1 - dH_min задається в сотих грама на м*3] [0 - dH_min задається в процентах від absHIn]
                                // 3 біт - дисплей увімкнено
                                // 4 - MOTOR_IN_BIT - запуск вентиляторів
                                // 5 - MOTOR_OUT_BIT - вимкнення вентиляторів
                                // 6-7 -  ПУСТО
        int8_t  color_mode;     // Колірна схема: 0, 1, 2, 3
        int8_t  demo_mode;      // Вимкнено = 0 / Увімкнено = 1  демонстраційний режим
        int8_t  servo_mode;     // Тип повітряних заслінок: з серво-приводом = 0, з магнітним механізмом = 1
        uint8_t N_Screen;       // Номер поточного інфо екрана
//приведення показів датчиків до одного рівня (коригування)
        int8_t  T1;             //температура T1 в °C
        int16_t d_T1;           //різниця між датчиками в сотих °C при температурі T1

        int8_t  T2;             //температура T2 в °C
        int16_t d_T2;           //різниця між датчиками в сотих °C при температурі T2

        uint8_t H1;             //вологість H1 в %
        int16_t d_H1;           //різниця між датчиками в сотих % при вологості H1

        uint8_t H2;             //вологість H2 в %
        int16_t d_H2;           //різниця між датчиками в сотих % при вологості H1

        int16_t serv_90_deg;    //значення для повороту сервопривода на 90° - відкрито
        int16_t serv_0_deg;     //значення для повороту сервопривода на 0° - закрито

}       DataForSave;
```

Рисунок 3.28 – Лістинг структура для збереження даних на SD-карту

3.1.2.1.2 Структура для графіків історії, що зберігаються на SD-карту

Структура DATA_HISTORY призначена для збереження історичних даних параметрів температури та вологості з метою реалізації можливості відображення їх як показано на рис. 3.44.

Структура зберігає дані за одну добу (тобто, час від 00:00 до 24:00), для цього створено масиви по 120 елементів, що достатньо для збереження даних кожні 12 хвилин:

- 1 доба = 24 год. х 60 хв. = 1440 хв.
- 1400 хв. / 12 хв. = 120 - кількість точок на шкалі часу для 24 год.

Індекс в масиві визначає час отримання значень: 0 - вимірювання в діапазоні часу від 00:00 до 00:12, 1 - до 00:24 ... 119 - від 23:48 до 00:00.

Також ця структура використовується для зчитування даних за вказану дату з SD-карти з наступним виведенням на дисплей. Для цього оголошено дану структуру з іншим ім’ям: data_history.

Більш детально інформація про структуру представлена в коментарях лістингу на рис. 3.29.

```
typedef struct DATA_HISTORY
{
// 1 день = 24 год. х 60 хв. = 1440 хв.
// точку на графіку ставимо кожні 12 хв.
// 1400 хв. / 12 хв. = 120 - кількість точок на шкалі часу для 24 год.
// індекс в масиві визначає час отримання значень: 0 - вимірювання в діапазоні часу від 00:00 до 00:12,
//  1 - до 00:24 ... 119 - від 23:48 до 00:00
        uint8_t H_Date;         //день
        uint8_t H_Month;        //місяць
        uint8_t H_Year;         //рік
        int16_t tOut[120];      //температура ззовні - в сотих градуса
        int16_t tIn[120];       //температура в шафі - в сотих градуса
        int16_t absHOut[120];   //абсолютна вологість ззовні - в сотих грама на м*3
        int16_t absHIn[120];    //абсолютна вологість в шафі - в сотих грама на м*3
        int16_t relHOut[120];   //відносна вологість ззовні - в сотих відсотка
        int16_t relHIn[120];    //відносна вологість в шафі - в сотих відсотка
        int16_t dH[120];        //різниця ззовні та в шафі значень відносної вологості - в сотих грама на м*3
        int8_t Motor[120];      //ознака роботи вентилятора під час інтервалу графіка, якщо вентилятор працював
                                //(навіть якщо одне вимірювання), то на графіку фон міняється
        int8_t Heat[120];       //ознака роботи обігрівача під час інтервалу графіка, якщо обігрівач працював
                                //(навіть якщо одне вимірювання), то на графіку фон міняється
}       DATA_HISTORY;

//функція задання початкових значень структури DATA_HISTORY
DATA_HISTORY* DATA_HISTORY_New(void)
{
        DATA_HISTORY *chdr = (DATA_HISTORY*)calloc(1, sizeof(DATA_HISTORY));

        //початкові значення
        chdr->H_Date    = 0;
        chdr->H_Month   = 0;
        chdr->H_Year    = 0;
        for (uint8_t k = 0; k < 120; k++)
        {
            chdr->tOut[k]       = -1500;
            chdr->tIn[k]        = -1500;
            chdr->absHOut[k]    = 0;
            chdr->absHIn[k]     = 0;
            chdr->dH[k]         = 0;
            chdr->relHOut[k]    = 0;
            chdr->relHIn[k]     = 0;
            chdr->Motor[k]      = 0;
            chdr->Heat[k]       = 0;
        }
        return chdr;
}

//оголошення вказівників на структури DATA_HISTORY
DATA_HISTORY* data_today;//структура для збереження даних за поточну добу на SD-карту
DATA_HISTORY* data_history;//структура для зчитування даних за вказану дату з SD-карти з наступним виведенням на екран
```

Рисунок 3.29 – Лістинг структури для графіків історії, що зберігаються на SD-карту

3.1.2.1.3 Структура для збереження поточних значень

Структура Packet призначена для збереження поточних результатів в оперативну пам’ять МК. Для економії оперативної пам’яті значення температур та вологості зберігаються в сотих долях, а тиску – в десятих.

Більш детально інформація про структуру представлена в коментарях лістингу рис. 3.30.

```
typedef struct Packet
{
        int16_t         tOut;           // поточна температура ззовні - в сотих градуса
        int16_t         tIn;            // поточна температура в шафі - в сотих градуса
        int16_t         absHOut;        // абсолютна вологість ззовні - в сотих грама на м*3
        int16_t         absHIn;         // абсолютна вологість в шафі - в сотих грама на м*3
        int16_t         relHOut;        // відносна вологість ззовні - в сотих відсотка
        int16_t         relHIn;         // відносна вологість в шафі - в сотих відсотка
        int16_t         p;              // поточний тиск - в десятих гПа
        uint8_t         dH_min;         // поріг увімкнення вентилятора по різниці абсолютної вологості в сотих грама на м*3
                                        // або в % (залежить від flags:2), тільки позитивні значення
        uint8_t         T_min;          // поріг вимкнення вентилятора по температурі в долях градуса, тільки позитивні значення
        uint8_t         Step_Motor_In;  // номер кроку процесу запуску вентиляторів
        uint8_t         Step_Motor_Out; // номер кроку процесу вимкнення вентиляторів
}       Packet;

//функція задання початкових значень структури Packet
Packet* PacketNew(void)
{
        Packet *pct = (Packet*)calloc(1, sizeof(Packet));
        pct->tOut               = 500;  //+5 °C
        pct->tIn                = 500;  //+5 °C
        pct->absHOut    = 321;          //3.21 грама на м*3
        pct->absHIn             = 321;  //3.21 грама на м*3
        pct->relHOut    = 5000;         //50%
        pct->relHIn             = 5000; //50%
        pct->p                  = 10345;//1034.5 гПа
        pct->dH_min             = 255;
        pct->T_min              = 255;
        pct->Step_Motor_In      = 0;
        pct->Step_Motor_Out     = 0;
        return pct;
}

volatile Packet *packet; //оголошення вказівника на структуру Packet під ім’ям packet
```

Рисунок 3.30 – Лістинг структури для збереження даних на SD-карту

3.1.2.1.4 Структура для навігації по меню

Структура KILCE призначена для зручної реалізації навігації по всіх меню. Вона містить поточне, максимальне й мінімальне значення позиції на меню. Також реалізовано функції для циклічного збільшення/зменшення поточної позиції навігації. Оголошується масив структур: для кожного меню свій екземпляр структури з ім’ям Screens[]. Крім того, оголошується загальна

структура типу KILCE з ім’ям ScreenMode, яка зберігає номер поточного меню. Більш детально інформація про структуру представлена в коментарях лістингу рис. 3.31.

```
struct KILCE;   // Структура KILCE для навігації по меню

//оголошення функцій для роботи зі структурою KILCE
typedef void (*KilceSet_Poz)(struct KILCE *klc, uint8_t Poz);//встановлення поточного значення позиції

typedef uint8_t (*KilceGet_Poz)(struct KILCE *klc);     //отримання поточного значення позиції

typedef void (*KilceUp)(struct KILCE *klc);             //збільшення позиції

typedef void (*KilceDown)(struct KILCE *klc);           //зменшення позиції

typedef struct KILCE
{
        uint8_t _Min;           // мінімальне значення позиції
        uint8_t _Max;           // максимальне значення позиції
        volatile uint8_t _Poz;  // поточне значення позиції
        KilceSet_Poz Set_Poz;   // встановлення поточного значення позиції
        KilceGet_Poz Get_Poz;   // отримання поточного значення позиції
        KilceUp Up;             // збільшення позиції
        KilceDown Down;         // зменшення позиції
}       KILCE;

static inline void DoSetPoz(KILCE *klc, uint8_t Poz)    //встановлення поточного значення позиції
{       klc->_Poz = Poz;        }

static inline uint8_t DoGetPoz(KILCE *klc)              //отримання поточного значення позиції
{       return klc->_Poz;       }

static inline void DoUp(KILCE *klc)                     //збільшення позиції
{       klc->_Poz == klc->_Max ? klc->_Poz = klc->_Min : klc->_Poz++;   }

static inline void DoDown(KILCE *klc)                   //зменшення позиції
{    klc->_Poz == klc->_Min ? klc->_Poz = klc->_Max : klc->_Poz--;      }

KILCE* KILCE_New(uint8_t Min, uint8_t Max)      //функція задання початкових значень структури KILCE
{
        KILCE *klc      = (KILCE*)calloc(1, sizeof(KILCE));
        klc->_Min = Min;
        klc->_Max = Max;
        klc->_Poz = Min;
        klc->Set_Poz    = DoSetPoz;
        klc->Get_Poz    = DoGetPoz;
        klc->Up         = DoUp;
        klc->Down       = DoDown;
        return klc;
}
//оголошення вказівників на структури KILCE
KILCE *ScreenMode;              //для збереження номера поточного меню
KILCE *Screens[NUM_SCREEN];     //для збереження позиції в меню
```

Рисунок 3.31 – Лістинг структури для навігації по меню

3.1.2.1.5 Структура для визначення середніх значень показів

Для згладжування пульсацій вимірювань показів датчиків необхідно визначати середні значення за деякий останній період часу.

Структура AVARAGE призначена для визначення середніх значень показів. Для економії оперативної пам’яті використовується метод ковзного середнього [13]. Це дозволяє знаходити середнє значення, наприклад, для останніх десяти показів, при цьому не зберігати в оперативній пам’яті всі десять значень, а тільки їх суму. Більш детально інформація про структуру представлена в коментарях лістингу рис. 3.32.

```
struct AVARAGE;

//оголошення функцій для роботи зі структурою AVARAGE
typedef void (*AveragePush)(struct AVARAGE *avrg, int16_t X);//ф-я додає чергове значення і обчислюється ковзне середнє
typedef int16_t (*AverageMean)(struct AVARAGE *avrg);   //функція вертає значення ковзного середнього

typedef struct AVARAGE
{
        volatile int32_t _sum;          // сума всіх значень
        volatile uint8_t _MaxSize;      // максимальна кількість значень
        volatile uint8_t _count;        // поточна кількість значень
        volatile int16_t _mean;         // ковзне середнє значення
        AveragePush Push;               // функція додає чергове значення і обчислюється ковзне середнє
        AverageMean Mean;               // функція вертає значення ковзного середнього
}       AVARAGE;

static inline int16_t GetMean(AVARAGE *avrg)    //функція вертає значення ковзного середнього
{       return avrg->_mean;     }

void DoPush(AVARAGE *avrg, int16_t X)   //функція додає чергове значення і обчислюється ковзне середнє
{
        if(avrg->_count >= avrg->_MaxSize)
        {
                avrg->_sum =    (int32_t)(avrg->_sum - avrg->_mean + X);
                avrg->_mean =   (int16_t)(avrg->_sum / avrg->_MaxSize);
        }
        else
        {
                if(avrg->_count == 0)
                { avrg->_sum = X; avrg->_count = 1; avrg->_mean = X; }
                else
                {
                        avrg->_count++;
                        avrg->_sum = (int32_t)(avrg->_sum + X);
                        avrg->_mean = (int16_t)(avrg->_sum / avrg->_count);
                }
        }
}

AVARAGE* AVARAGE_New(uint8_t size)      //функція задання початкових значень структури AVARAGE
{
      AVARAGE *avrg             = (AVARAGE*)calloc(1, sizeof(AVARAGE));
      avrg->_MaxSize    = size;
      avrg->_count      = 0;
      avrg->_sum        = 0;
      avrg->_mean       = 0;
      avrg->Push        = DoPush;
      avrg->Mean        = GetMean;
      return avrg;
}
```

Рисунок 3.32 – Лістинг структури для визначення середніх значень показів

3.1.2.2 Функції

Для роботи програми було створено функції, перелік і призначення яких приведено в табл. 3.2.

Таблиця 3.2 - Перелік основних функцій, створних в програмі

Функція	Призначення
void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim)	обробник переривань таймера
void resetTimeDate(void);	скидання дат мін/максимальних статистичних значень
void writeSD();	запис на SD-карту
void write_setting_SD(void);	запис на SD структури settingRAM
void write_chart_SD(void);	запис на SD структури chart_dry
void readSD(void);	читання даних з SD-карти
void read_setting_SD(void);	читання з SD-карти структури settingRAM
void read_chart_SD(void);	читання з SD-карти структури chart_dry
void resetTemp(void);	скидання мінімальних/максимальних статистичних значень
void dry_static(int8_t IsP);	друк статичної картинки на інформаційних екранах №2 і №3
void printChart(int8_t IsP);	друк графіка на інформаційних екранах №2 і №3
void print_status0(void);	друк значень на інформаційних екранах №2 і №3
void dry_get_data(void);	заповнення масиву графіка
void CheckEkran(void);	перевірка: чи не потрібно вимкнути дисплей
void dry_update(int8_t IsP);	оновлення графіків на інформаційних екранах №2 і №3
void CheckON(void);	Перевірка статусу системи, чи не потрібно увімкнути/вимкнути зовнішні пристрої
void printSetTimeStatic(void);	друк екрану "Налаштування Часу/Дати"
void printSetTimeUpdate(void);	оновлення екрану "Налаштування Часу/Дати"
void printSettingStatic(void);	друк екрану налаштувань з вибором режиму роботи
void printInfoStatic0(void);	Друк статичної картинки інформаційного екрана №0
void printInfoStatic1(void);	Друк статичної картинки інформаційного екрана №1

Продовж.табл. 3.2

Функція	Призначення
void printInfoUpdate0(void);	Друк значень на інформаційному екрані №0
void printInfoUpdate1(void);	Друк значень на інформаційному екрані №1
void MotorON_SRV(void);	увімкнення вентиляторів, відкриття повітряних заслінок з серво-приводом
void MotorOFF_SRV(void);	вимкнення вентиляторів, закриття повітряних заслінок з серво-приводом
int16_t F_H_In_err(float H_X);	приведення показів вологості до одного рівня
int16_t F_T_In_err(float T_X);	приведення показів температури до одного рівня
int16_t calculationAbsH_P(float t, float h, float p);	функція для обчислення значення абсолютної вологості
void readDHT(void);	читання параметрів з датчиків
void OpenFileManager (void);	відкриття екрана файлового менеджера
int8_t button_click_handler(void);	обробник натискання кнопок
void Goto_New_Window(void);	функція відображає новий інформаційний екран
void save_history(void);	збереження параметрів на SD-карту
void read_history(char *file_name, int8_t show_error);	читання історичних даних з SD-карти в структуру data_history та виведення графіків
void read_today(char *file_name, int8_t show_error);	читання даних з SD-карти за поточну добу в структуру data_today
void printColorMode(void);	друк екрану налаштувань "Колірна схема"
void printDemoMode(void);	друк екрану налаштувань "Демонстраційний режим"
void printAirDampers(void);	друк екрану налаштувань "Налаштування повітряних заслінок"
void printSetServo(void);	друк екрану налаштувань "Налаштування серво-приводів"
void printSetServoUpdate(void);	оновлення екрану "Налаштування серво-приводів"
void Open_Servo(uint8_t Nom_serv);	відкриття повітряної заслінки з серво-приводом
void Close_Servo(uint8_t Nom_serv);	закриття повітряної заслінки з серво-приводом
void printKorDat(void);	друк екрану налаштувань "Коригування датчиків"
void KorDatUpdate(void);	оновлення екрану "Коригування датчиків"
void SetKorDat(void);	збереження заданих параметрів для приведення показів датчиків до одного рівня (коригування)

Далі дано детальний опис деяких основних функцій.

3.1.2.2.1 Функція int main(void)

Основною функцією в програмі є int main(void). З неї починається виконання програми. Спочатку в цій функції виконуються наступні дії:

- ініціалізуються таймери, порти вводу/виводу (GPIO), комунікаційні інтерфейси (I^2C, SPI);
- задаються початкові значення змінних та структур;
- виконуються необхідні налаштування;
- монтується файлова система SD-карти;
- зчитуються з SD-карти та застосовуються значення збережених налаштувань.

Потім виконуються безкінечний цикл в якому за допомогою обробника натискання кнопок реалізована навігація по меню.

Налаштування датчиків BME280 представлено в лістингу рис. 3.33. Два датчики використовують шину I^2C і мають різні адреси: 0x76, та 0x77. Адресу датчика визначає напруга подана на контакт SDO датчика. Якщо на SDO подано низький рівень (під'єднано до мінуса живлення), то адреса буде 0x76. Якщо подано високий рівень (під'єднано до плюса живлення) - 0x77. Працюють датчики в режимі роботи «нормальний». Тобто датчики постійно вимірюють покази. Значення зчитуються з датчиків по перериванню таймера TIM1 кожні 3 секунди.

```
/* ---------------------------- Налаштування датчиків BME280 ------------------------------ */
/* 1-й BME280 - dev1*/
dev1.dev_id = BME280_I2C_ADDR_PRIM;   // адреса 0x76
dev1.intf = BME280_I2C_INTF;
dev1.read = user_i2c_read;
dev1.write = user_i2c_write;
dev1.delay_ms = user_delay_ms;

rslt = bme280_init(&dev1);

dev1.settings.osr_h = BME280_OVERSAMPLING_1X;
dev1.settings.osr_p = BME280_OVERSAMPLING_1X;
dev1.settings.osr_t = BME280_OVERSAMPLING_1X;
dev1.settings.filter = BME280_FILTER_COEFF_OFF;
rslt = bme280_set_sensor_settings(BME280_OSR_PRESS_SEL | BME280_OSR_TEMP_SEL | BME280_OSR_HUM_SEL | BME280_FILTER_SEL, &dev1);

//налаштовуємо датчик 1 в режим роботи: нормальний. Він постійно буде вимірювати, а з необхідною нам періодичність будемо забирати значення
rslt = bme280_set_sensor_mode(BME280_NORMAL_MODE, &dev1);

/* 2-й BME280 - dev2*/
dev2.dev_id = BME280_I2C_ADDR_SEC;    // адреса 0x77
dev2.intf = BME280_I2C_INTF;
dev2.read = user_i2c_read;
dev2.write = user_i2c_write;
dev2.delay_ms = user_delay_ms;

rslt = bme280_init(&dev2);

dev2.settings.osr_h = BME280_OVERSAMPLING_1X;
dev2.settings.osr_p = BME280_OVERSAMPLING_1X;
dev2.settings.osr_t = BME280_OVERSAMPLING_1X;
dev2.settings.filter = BME280_FILTER_COEFF_OFF;
rslt = bme280_set_sensor_settings(BME280_OSR_PRESS_SEL | BME280_OSR_TEMP_SEL | BME280_OSR_HUM_SEL | BME280_FILTER_SEL, &dev2);

//налаштовуємо датчик 2 в режим роботи: нормальний. Він постійно буде вимірювати, а з необхідною нам періодичність будемо забирати значення
  rslt = bme280_set_sensor_mode(BME280_NORMAL_MODE, &dev2);
```

Рисунок 3.33 – Лістинг налаштування датчиків BME280

Безкінечний цикл приведено в лістингу рис. 3.34. В коментарях лістингу пояснені всі кроки коду.

```
while (1)
{       if (KEYB_kbhit() == 1)                          //якщо була натиснута будь-яка кнопка
        {       rslt = button_click_handler();          //то запуск обробника натискання кнопок
                if(rslt == 1)                           //якщо є перехід на новий екран,
                        Goto_New_Window();              //то відображаємо його
        }

        if (FLAG_EKRAN_CHECK)                           //якщо дисплей увімкнуто
        {       if (packet_ready == 1)                  //і якщо було опитування датчиків
                {       packet_ready = 0;               //скидаємо ознаку «опитування датчиків»
                        if(ScreenMode->Get_Poz(ScreenMode) == 0)// якщо знаходимся на якомусь інфо екрані
                        {       switch (Screens[0]->Get_Poz(Screens[0]))
                                { // оновлюємо зображення в залежності від номера екрана
                                        case 0: // якщо інформаційний екран 0
                                                if(packet_ready_E0 == 1)
                                                {       printInfoUpdate0();     // Друк значень на екрані 0
                                                        cubik();                // зафарбовування квадрата
                                                        packet_ready_E0 = 0;
                                                }
                                                break;
                                        case 1: // якщо інформаційний екран 1
                                                if(packet_ready_E1 == 1)
                                                {       printInfoUpdate1();     // Друк значень на екрані 1
                                                        cubik();
                                                        packet_ready_E1 = 0;
                                                }
                                                break;
                                        case 2: // якщо інформаційний екран 2
                                                print_status0();        // Друк значень
                                                dry_update(0);          // Друк графіків
                                                cubik();
                                                break;
                                        case 3: //якщо інформаційний екран 3
                                                print_status0();        // Друк значень
                                                dry_update(1);          // Друк графіків
                                                cubik();
                                }

                                if(packet_ready_Time == 1)      //якщо була зміна хвилини
                                {       show_time();    //оновлення відображення дати, часу
                                        packet_ready_Time = 0;
                                }
                        }
                        else
                        {
                           if(ScreenMode->Get_Poz(ScreenMode) == 11)//якщо на екрані "Коригування датчиків"
                           {  KorDatUpdate(); }         // Оновлення екрану "Коригування датчиків"
                        }
                }
        }
```

Рисунок 3.34 – Лістинг безкінечного циклу функції main()

Дані з датчиків зчитуються раз на три секунди за допомогою функції обробника переривань таймера void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim). Частина її приведена в лістингу рис. 3.35. В коментарях лістингу пояснені всі кроки коду. В лістингу пропущено знаходження максимальних/мінімальних значень температури та вологості. Повний лістинг функції приведено в Додатку Б.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
	if(htim->Instance == TIM1) //перевіряємо чи переривання від 1-го таймера TIM1
	{
		if(flag_measurement == 0)	//страхування, щоб другий раз не знімати дані, якщо ще не закінчився попередній раз
		{
			flag_measurement = 1;	//встановлюємо ознаку, що почали обробку показів датчиків

			HAL_IWDG_Refresh(&hiwdg); // Скидання сторожового таймера

			readDHT();		//читання параметрів з датчиків

			CheckEkran();		//перевірка чи не пора вимкнути дисплей

			get_time(0);		// отримання актуальних дати, часу і формування масиву TimeBuf для виводу

			if(FLAG_MOTOR_OUT_CHECK)//якщо іде процедура вимкнення вентиляторів
			{
				MotorOFF();	//вимкнення вентиляторів, закриття повітряних заслінок
			}
			else
			{
				if(FLAG_MOTOR_IN_CHECK) //якщо іде процедура запуску вентиляторів
				{
					MotorON();	//увімкнення вентиляторів, відкриття повітряних заслінок
				}
				else
				{
					CheckON();	// Перевірка стану системи
				}
			}

			if (FLAG_MOTOR_CHECK)	//якщо вентилятори працюють
			{
				chart_dry->ChartMotor = 1;//ознака того що потрібно показувати увімкнення вентиляторів на графіках
				settingRAM->hour_motor++;//облік часу роботи вентиляторів
			}

			if (FLAG_HEAT_CHECK)	//якщо обігрівач працює
			{
				chart_dry->ChartHeat = 1;//ознака того що потрібно показувати увімкнення обігрівача на графіках
				settingRAM->hour_heat++;//облік часу роботи обігрівача
			}

			settingRAM->hour_unit++;	//облік часу роботи системи

//		Оновлюємо максимум і мінімум температур на SD
//		Дані оновлюються раз в 3 секунды (TIME_SCAN_SENSOR),
//		а пишемо на SD раз в 3 години для економії ресурсу SD, кількість циклів перезапису обмежена

			settingRAM->tick_SDflash++;

			if (((int16_t)settingRAM->tick_SDflash) >= (int16_t)TIME_HOUR) // чи пора писати на SD
			{
				writeSD();// Запис на SD-карту
			}

			if (FLAG_EKRAN_CHECK)
			{
				hour_ekran++;	//облік часу увімкненого дисплея
			}

			dry_get_data(); // заповнення масиву для графіків

			flag_measurement = 0;//скидаємо ознаку, що почали обробку показів датчиків
			packet_ready = 1;//встановлюємо ознаку «опитування датчиків»
		}
	}
}
```

Рисунок 3.35 – Лістинг обробника переривань таймера

Читання параметрів з датчиків виконується за допомогою функції readDHT(). Після отримання з датчиків значень відносної вологості

вираховуються значення абсолютної вологості за допомогою функції calculationAbsH_P() за формулою (2.1). Функція приведена в лістингу рис. 3.36. Для економії оперативної пам'яті функція повертає значення абсолютної вологості в сотих грама на $м^3$.

```
int16_t calculationAbsH_P(float t, float h, float p)
{
	float P_nasish_T = 6.112 * pow(2.718281828, 17.62 * t / (243.12 + t));//гПа
	float P_nasish_P = 1.0016 + 0.00000315 * p - 0.074 / p;//гПа
	float P_nasish = P_nasish_P * P_nasish_T;//гПа

	float P_par = h * P_nasish;//гПа

	float AbsH_P = 100000.0 * P_par / (461.5 * (t + 273.15) );

	return (uint16_t)AbsH_P;
}
```

Рисунок 3.36 – Лістинг функції для обчислення значення абсолютної вологості

Після отримання значень з датчиків виконується функція CheckON() в якій реалізовано перевірка статусу системи і в залежності від вибраного режиму і від отриманих параметрів температури й вологості приймається рішення щодо управління зовнішніми пристроями (вентиляторами, засувками). Функція приведена в лістингу рис. 3.37. В коментарях лістингу пояснені всі кроки коду.

```
void CheckON(void)
{  int tmp = 0;
// 0.  Перевірка переохолодження приміщення, увімкнення обігрівача
        if (~FLAG_HEAT_CHECK)
        {       if (ave_tIn->Mean(ave_tIn) <= TEMP_LOW) // Контроль заморожування приміщення
                {       if (FLAG_MOTOR_CHECK)
                        {       MOTOR_OFF;      } HEAT_ON; return;      }       // увімкнення обігрівача
        }
        if (FLAG_HEAT_CHECK)//Вимкнути обігрівач коли температура підніметься на 0,5 градуса від критичної+значення гістерезису (1,1 °C)
          {     if (ave_tIn->Mean(ave_tIn) > TEMP_LOW+dT_OFF + 50)      HEAT_OFF;       }
// 1. Режими що не залежать від вологості і температури (високий пріоритет)
        if (settingRAM->mode == BLOCK_OFF)      //якщо режим Вимкнено
        {       if (FLAG_MOTOR_CHECK)   {       MOTOR_OFF;      }       return; }       // то вимикаємо вентилятори
        if (settingRAM->mode == HOOD_ON )       // якщо режим вентиляції
        {       if (~FLAG_MOTOR_CHECK)  {       MOTOR_ON;       }       return; }       // то вмикаємо вентилятори
// 2. Режим охолодження (другий пріоритет) температура всередині вище 8 радусів, на вологість не дивимся
        if (settingRAM->mode == COOLING)          // якщо режим охолодження
          {     if (FLAG_MOTOR_CHECK)                                           // вентилятори працюють
                {       if(ave_tIn->Mean(ave_tIn) <= ave_tOut->Mean(ave_tOut))// і температура всередині нижче ніж ззовні
                                MOTOR_OFF;                                                      // то вимикаємо вентилятори
                }
                else                                                            // якщо вентилятори вимкнено
                {       if(ave_tIn->Mean(ave_tIn) > (packet->T_min * 10.0))             //температура вище встановленої (8°C)
                        {       // dH_min використовується не штатно для температури для режимі COOLING
                           if((ave_tIn->Mean(ave_tIn) - ave_tOut->Mean(ave_tOut)) > packet->dH_min)
                                MOTOR_ON;                                                               //  то вмикаємо вентилятори
                        }
                }       return;                         // виходимо
        }
// 3. В режимі осушення - перевірка досягнення мінімальної температури в приміщенні - ТЕРМІНОВО ВиМКНУТи - третій пріоритет
        if (ave_tIn->Mean(ave_tIn) <= (packet->T_min * 10.0))
        {       if (FLAG_MOTOR_CHECK)   {       MOTOR_OFF;      } return;       }       // вимикаємо вентилятори
//ave_absHIn в сотих долях г/м3, ave_relHIn - в сотих долях %
// 4. Режими, які залежать від температури і вологості - низький приоритет (що залишилось)
        if (FLAG_MOTOR_CHECK)                                                   // якщо вентилятори працюють
        {       if((ave_tIn->Mean(ave_tIn) <= (packet->T_min * 10.0)) ||                //і температура нижче критичної
                (ave_absHIn->Mean(ave_absHIn) < (ave_absHOut->Mean(ave_absHOut) + dH_OFF)))//або абс вологість в приміщенні нижче ніж ззовні
                        MOTOR_OFF;                                                                      //то вимикаємо вентилятори
        }
        else                                                                    // якщо вентилятори вимкнено
        {  if(ave_tIn->Mean(ave_tIn)>(packet->T_min * 10.0))            //температура вище критичної
           {    // Розраховуємо ризницю спрацювання по вологості
              if (FLAG_ABS_H_CHECK)     { tmp = packet->dH_min; }//режими використовують абсолютну вологість - різниця в сотих грама на метр.куб
                else {tmp=(uint16_t)(ave_absHIn->Mean(ave_absHIn)*packet->dH_min/100.0);}//режими вик-ть дес.долі % від абс.різн.темп.в приміщ.
                  if((ave_absHIn->Mean(ave_absHIn) - tmp) > ave_absHOut->Mean(ave_absHOut))      //Якщо абс вологість ззовні менше
                    {   MOTOR_ON;       }       //то вмикаємо вентилятори
           }
        }
}
```

Рисунок 3.37 – Лістинг функції для перевірки статусу системи й управління зовнішніми пристороями

3.1.3 Інтеграція механічних елементів з мікроконтролером

3.1.3.1 Живлення вентиляторів, обігрівача повітря, сервоприводів

Живлення на вентилятори, обігрівач повітря, сервоприводи подається за допомогою реле, яке описано вище в розділі 2.2.4. «Вибір та обґрунтування реле». Керуючі сигнали подаються на реле у відповідності з табл. 2.4.

3.1.3.2 Керуючий сигнал сервопроводів

Для конвертації керуючого ШІМ сигналу сервоприводів з 3,3 В до 5 В доречно скористатись схемою на оптроні PC817 з гальванічною розв’язкою [17]:

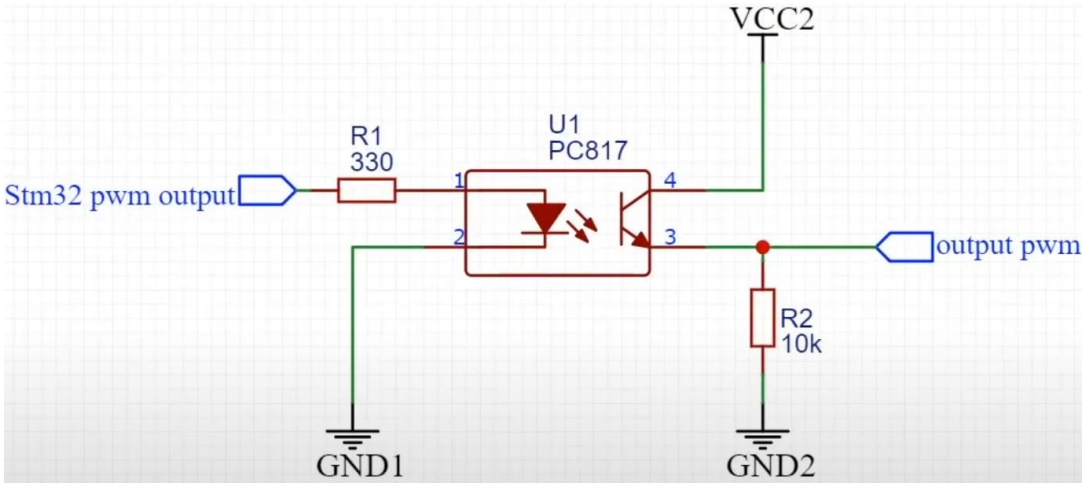

Рисунок 3.38 – Підключення керуючого виводу до сервоприводу

3.1.3.3 Кнопки керування пристроєм

Перший контакт кнопок керування пристроєм підключається до МК у відповідності з табл. 2.4. Другий контакт кнопок з’єднаний з мінусом живлення. Для перших контактів кнопок в STM32CubeIDE налаштована їх підтяжка до високого рівня сигналу, тобто до 3,3 В.

3.2 Тестування

Для тестування на макетній платі було виготовлено тестову версію системи: рис. 3.39.

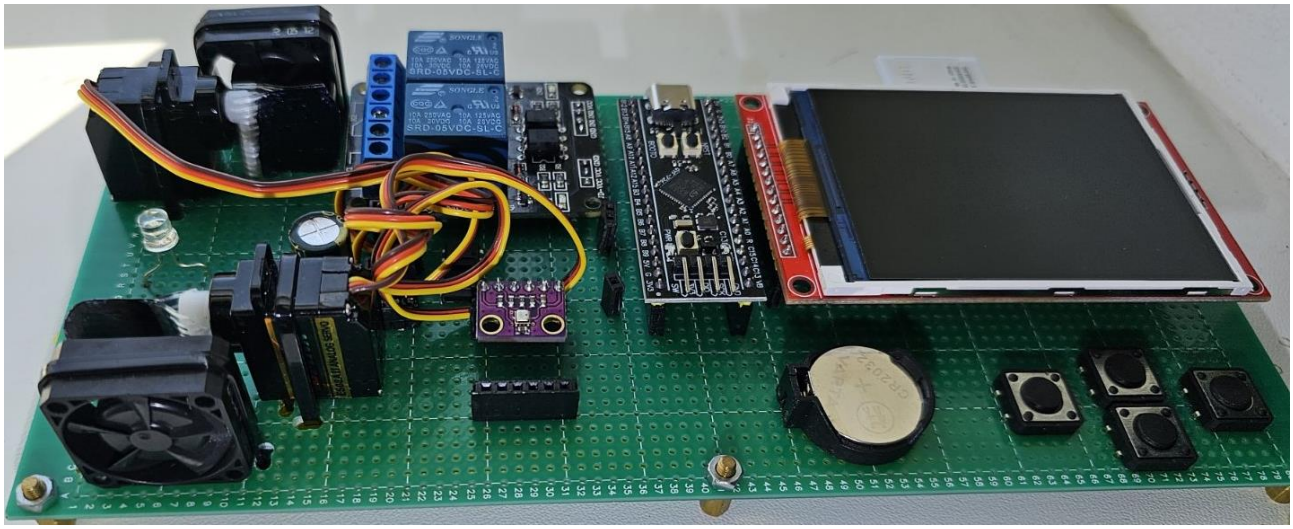

Рисунок 3.39 – Тестова версія системи на макетній платі

На макетній платі змонтовані всі елементи, крім реле комутації обігрівача: для емуляції обігрівача поставлено світлодіод.

Для зручності тестування датчик BME280 для зовнішніх параметрів виведено на провід довжиною 50 см.

3.2.1 Інтерфейс

Інтерфейс користувача складається з чотирьох інформаційних екранів та меню налаштувань.

На інформаційних екранах представлено:

- загальна інформація про роботу пристрою (час роботи пристрою, час роботи вентиляторів, час роботи обігрівача), рис. 3.40;
- максимальні/мінімальні значення параметрів та час і дата їх фіксування, рис. 3.41;
- поточні значення параметрів, рис. 3.42;
- графіки значень параметрів за останні 24 години, рис. 3.42.

На кожному інформаційному екрані зверху відображено вибраний режим роботи пристрою, стан вентиляторів, поточний час та дата.

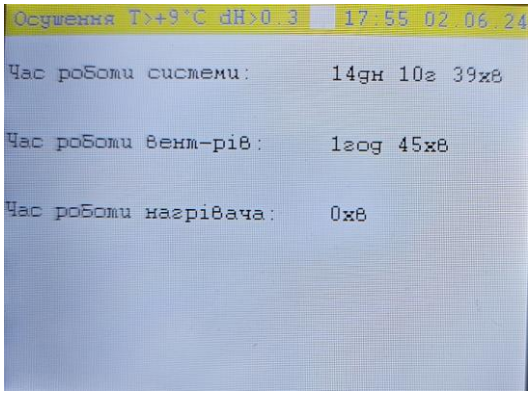

Рисунок 3.40 – Загальна інформація про роботу пристрою

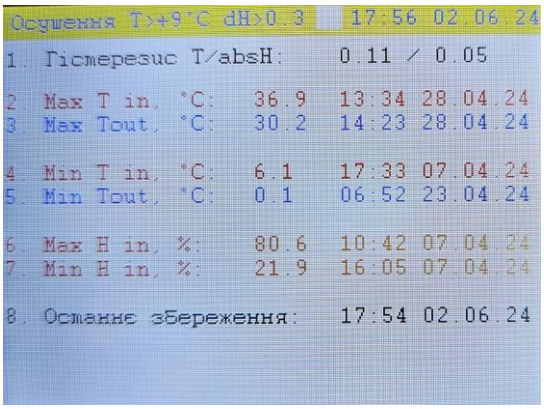

Рисунок 3.41 – Максимальні/мінімальні значення параметрів

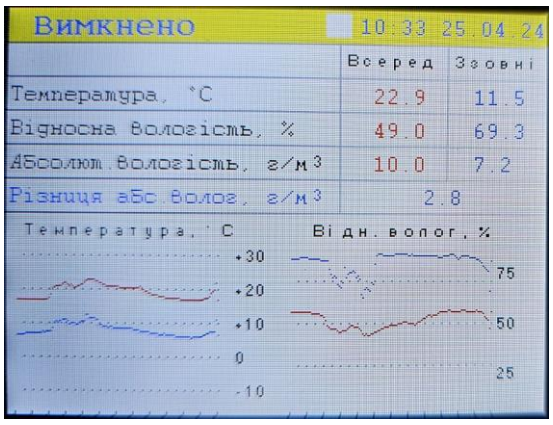

Рисунок 3.42 – Поточні значення параметрів та графіки значень параметрів за останні 24 години

В інтерфейсі реалізовано файловий менеджер (рис. 3.43), в якому можна вибрати необхідний файл для перегляду динаміки параметрів за обрану дату.

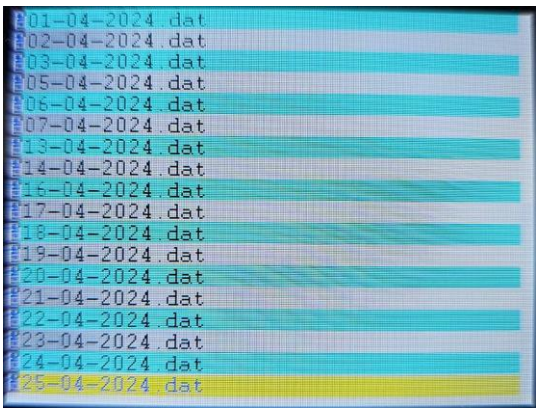

Рисунок 3.43 – Приклад відображення файлового менеджера

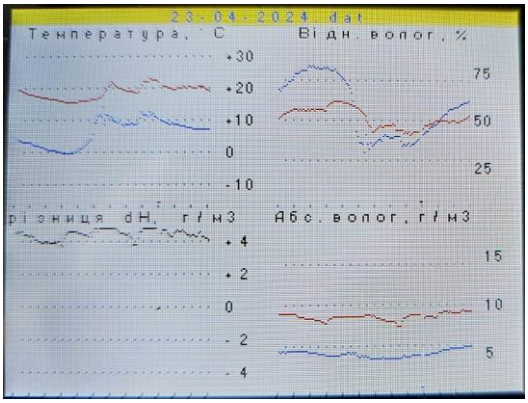

Рисунок 3.44 – Приклад відображення збережених значень параметрів

В меню налаштувань (рис. 3.45) користувач може вибрати необхідний режим роботи та змінити налаштування.

Реалізовано вісім режимів роботи системи:

1. Вимкнено;
2. Режим примусової вентиляції;
3. Охолодження при T > 9°C в шафі, різниця dT > 2°C;
4. Осушення при T > +9°C в шафі, різниця dH > 0,3 г/м^3;
5. Осушення при T > +9°C в шафі, різниця dH > 5%;

6. Осушення при T > +16°C в шафі, різниця dH > 0,6 г/м^3;
7. Осушення при T > +16°C в шафі, різниця dH > 10%;
8. Осушення при T > +16°C в шафі, різниця dH > 0,8 г/м^3.

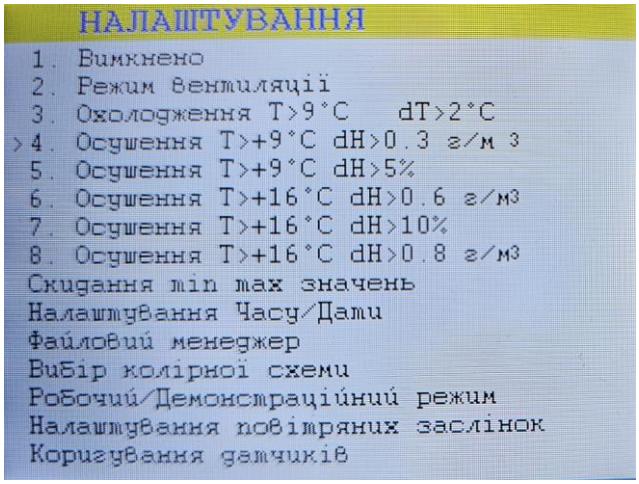

Рисунок 3.45 – Меню «Налаштування»

Температури 9 °C та 16 °C в режимах вибрані, враховуючи вимоги до мінімальних температур зберігання лікарських засобів додаючи 1 °C:

- у холодному чи прохолодному місці – від +8 °C до +15 °C;
- при кімнатній температурі (від +15 °C до +25 °C).

Також в меню налаштувань можна виконати:

- скидання мінімальних/максимальних значень параметрів;
- налаштування часу/дати;
- вибір колірної схеми з чотирьох (світла, темна, синя, зелена);
- увімкнення демонстраційного режиму роботи, в якому збір даних виконується швидко для демонстрації побудови графіків;
- налаштування повітряних заслінок та сервоприводів;
- коригування датчиків.

3.2.2 Коригування датчиків

Для правильного виконання режимів пристрою, які залежать від параметрів, необхідно виконати коригування датчиків. Для полегшення виконання цього в меню налаштувань створено відповідне меню (Коригування датчиків).

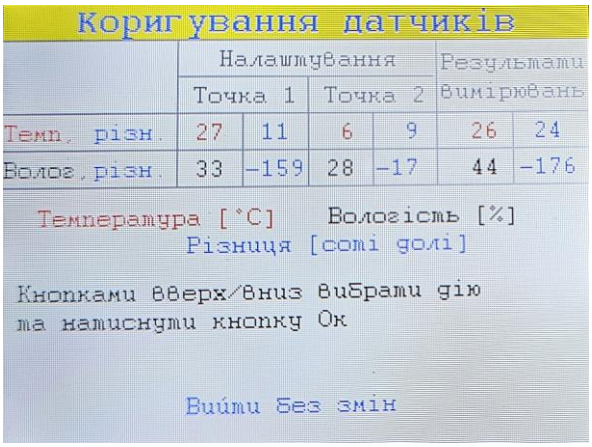

Рисунок 3.46 – Меню «Коригування датчиків»

Два датчики пристрою необхідно на тривалий час (не менше години) розмістити в спільне середовищі зі стабільними параметрами температури та вологості. Під час цього будуть виміряні різниці між показами датчиків. За допомогою меню налаштувань необхідно зберегти ці значення. Процедуру необхідно провести два рази для збереження двох точок при різних параметрах температури та вологості для більш коректного приведення показів датчиків один до одного.

3.2.3 Тестування режимів роботи

Після процедури коригування датчиків, систему було протестовано у всіх режимах роботи:

1. Вимкнено. Примусова вентиляція вимкнена (повітряні заслінки закриті, вентилятори вимкнено). Робота режиму не залежить від вологості та температури.
2. Режим вентиляції. Примусова вентиляція увімкнена (повітряні заслінки відкриті, вентилятори увімкнені). Робота режиму не залежить від вологості та температури.
3. Режим охолодження. Примусова вентиляція увімкнена якщо температура в шафі вища температури ззовні. Робота режиму не залежить від вологості.
4. Режим осушення. Примусова вентиляція увімкнена якщо вологість в шафі більша ніж вологість ззовні.

Крім того, для перевірки увімкнення обігрівача при критичній температурі 8 °C датчик BME280 було розміщено в холодне місце (в холодильник).

Результати тестування показали коректну роботу системи у всіх запланованих режимах, тобто коректне керування зовнішніми механізмами в залежності від зміни параметрів температури і вологості.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У кваліфікаційній роботі описується процес створення комп'ютеризованої системи для автоматичного контролю температури та вологості в шафі зберігання ліків, яка в свою чергу може розміщуватись в складському приміщенні чи торговій залі аптеки, тому у розділі розглядаються наступні питання: безпеки життєдіяльності, електробезпеки та мікроклімату. При підключенні розробленої системи існує небезпека ураження електричним струмом, тому передбачити заходи і рішення щодо усунення цієї небезпеки.

4.1 Заходи з безпеки життєдіяльності

Згідно Трудового кодексу робота з лікарськими засобами прирівнюється до роботи з шкідливими та небезпечними умовами праці. Тому дуже важливою умовою трудового процесу є охорона праці й безпека працівників. Особлива увага приділяється працівникам, що мають безпосередній контакт з лікарськими препаратами та їх складовими. Тому важливо створити такі умови праці, що відповідають державним нормативам з охорони праці. З цією метою створюється служба з охорони праці. Розробляються інструкції та положення з профілактики травматизму, професійних захворювань на виробництві, перевірка робочих місць, проводяться медичні огляди працівників.

Обов'язковим є забезпечення персоналу засобами індивідуального й колективного захисту. Це можуть бути спецодяг, спецвзуття, маски, знешкоджувальні косметичні засоби для захисту рук, після робіт з хімікатами при виготовленні лікарських медикаментів [18].

Враховуючи особливу специфіку роботи з ліками розроблений нормативний документ з питань охорони праці для внутрішнього користування на підприємстві чи закладі.

Кожен працівник повинен знати свої обов'язки з охорони праці та дотримуватися їх на своєму робочому місці. З цією метою з працівниками проводяться інструктажі та навчання.

Інструктаж з охорони праці включає в себе обов'язковими такі пункти:

– дотримання правил безпеки перед початком роботи (привести робоче місце відповідно до санітарних вимог);

– безпека під час роботи (дотримуватися правил роботи з хімічними речовинами, використовувати ЗІЗ, додержуватися правил особистої гігієни, при роботі з апаратурою виконувати заходи електробезпеки);

– правила безпеки після закінчення роботи (відключити апаратуру, провести санітарну обробку робочого місця та подбати про особисту гігієну);

– дії в аварійній ситуації (сповістити про аварію, надати допомогу постраждалим, вжити заходи по збереженню обстановки, при серйозній аварії допомагати в евакуації).

Під час роботи з лікарськими засобами працівник повинен зберігати стерильність, стежити за своєчасною утилізацією відходів, оскільки контакт з хімічними речовинами створює підвищену небезпеку для здоров'я людини.

«Інструкція про порядок зберігання та поводження в аптечних закладах з лікарськими засобами та виробами медичного призначення, які мають вогненебезпечні та вибухонебезпечні властивості», затверджена наказом Міністерства охорони здоров'я України від 16.03.1993 р. № 44, акцентує увагу на деяких аспектах роботи та збереження вогненебезпечних та вибухонебезпечних хімічних речовин.

Всі заклади, в яких знаходяться такі медичні препарати повинні мати та зберігати в належних місцях первинні засоби гасіння пожежі в кількостях, які узгоджені з місцевими органами Держпраці.

Приміщення для зберігання, згідно з нормами, забезпечується охоронними та протипожежними засобами.

Необхідну кількість вогненебезпечних речовин для поточних витрат допускається тримати в фасувальних кімнатах складів або аптек. Але при суворому дотриманні заходів пожежної безпеки. Залишкові кількості вогненебезпечних речовин після закінчення роботи в кінці зміни повертають на місце основного зберігання [19].

Відповідною інструкцією встановлюється протипожежний режим, який включає:

- визначення пожежних виходів та евакуаційних шляхів;
- проведення систематичних перевірок та технічного обслуговування протипожежного обладнання;
- встановлення попереджувальних систем пожежного сповіщення;
- забезпечення наявності пожежних вогнегасників та їх регулярна перевірка;
- гарантування безпеки електричних систем.

Основною складовою пожежної безпеки є навчання персоналу правилам пожежної безпеки та процедур поведінки в разі виникнення пожежі. Персонал повинен вміти виявляти пожежну небезпеку, користуватися пожежними вогнегасниками та ефективно діяти під час сигналу пожежі.

4.2 Заходи з дотримання електробезпеки

Електробезпека електронно-обчислюваної техніки відповідає вимогам НПАОП 40.1-1.01-97 [20]. Для кожної електронно-обчислювальної техніка, електропроводів та кабелів встановлюють апаратуру захисту від струму короткого замикання та інших аварійних режимів. Під час монтажу та подальшої експлуатації ліній електромережі необхідно перешкодити виникнення електричного джерела загоряння, що може виникнути внаслідок

короткого замикання або перевантаження проводів, лімітувати застосування проводів з легкозаймистою ізоляцією, а використовувати негорючу ізоляцію. Для живлення електронно-обчислюваних машин виконується окрема групова трипровідна мережа, в якій прокладається фазовий, нульовий робочий та нульовий, захисний провідники. Для заземлення електроприймачів використовується нульовий захисний провідник. Використовувати нульовий робочий провідник як нульовий захисний провідник не допускається. Нульовий захисний провідник прокладається від розподільного щита чи пункту до розеток електроживлення.

Не можна підключати на щиті до одного контактного затискача нульовий захисний та нульовий робочий провідники. Також не допускається підключення електронно-обчислювальних машин до звичайної двопровідної електромережі, навіть, якщо використовуються перехідні пристрої. Електромережу штепсельних розеток для живлення електронно-обчислюваної техніки прокладають у каналах в металевих трубах або гнучких металевих рукавах. Для цього не обирають провід і кабель в ізоляції з вулканізованої гуми або інших матеріалів, які містять сірку.

Коротке замикання – це електричне з'єднання двох точок електричного кола з різними значеннями потенціалів, яке порушує його нормальну роботу і не передбачене конструкцією пристрою. Коротке замикання може виникати при пошкодженні ізоляції струмоведучих елементів або внаслідок механічного контакту елементів, які працюють без ізоляції. Також коротким замиканням вважається стан, при якому опір навантаження менший за внутрішній опір джерела живлення [21]. Часто коротке замикання призводить до пожежі або повної зупинки функціонування мережі.

Причинами короткого замикання в електричній мережі є такі фактори:

- пошкодження ізоляції;
- порушення стандартів монтажу;
- механічні пошкодження;

– виробничі дефекти;

– несправність обладнання;

– несприятливі умови середовища;

Найчастішими причинами короткого замикання є пошкодження ізоляції та некваліфіковані дії обслуговуючого персоналу.

Наслідки короткого замикання можуть позначитися на всій системі або мати місцевий характер, це залежить від місця й тривалості виникнення ушкодження.

Існують спеціальні заходи, що знижують можливість появи короткого замикання [22]:

- використання пристроїв релейного захисту , що відключають пошкоджені ділянки кола;

- застосування обладнання автоматичного (автоматичні вимикачі чи плавкі запобіжники);

- використання розпаралелювання електричних кіл;

- застосовування знижувальних трансформаторів з розщепленою обмоткою низької напруги;

- встановлення струмообмежуючих електричних реакторів.

Працівники, що обслуговують електроустановки та контролюють комп’ютризовані системи повинні дотримуватись заходів електробезпеки.

4.3 Заходи до нормалізації мікроклімату

Приміщення, в яких виготовляються лікарські засоби повинні бути розміщені та розділені на технологічні зони (виробничі, складські та зону контролю якості).

Розподіл приміщень враховує послідовність виконання робіт та рівень її чистоти, щоб унеможливити перехрещення технологічних, матеріальних й людських потоків.

Внутрішні поверхні приміщень (стіни, підлога, стеля) мають легко очищуватися і дезінфікуватися.

Обов'язковою є діюча система вентиляції повітря, яка забезпечує чистоту навколишнього середовища для виконання певних робочих завдань.

Температура, освітлення, вологість та вентиляція відповідати технологічним вимогам і не впливати на якість роботи.

Встановлення і робота з обладнанням повинна буди ефективною, але виконуватися із дотриманням заходів електробезпеки.

Позитивно на мікроклімат впливає наявність кімнати відпочинку та їдальні, які повинні відокремлюватися від інших зон і укомплектовані зручними меблями.

Гардеробні, умивальні (душові) та туалети мають бути доступними і за кількістю відповідати вимогам нормативів.

Працівники, які працюють в чистих зонах виробництва лікарських препаратів, володіють знаннями і досвідом практичної роботи, які необхідні для їх виробництва та регулярно підвищувати кваліфікацію [23].

ВИСНОВКИ

Відповідно до поставленого завдання у кваліфікаційній роботі було вирішено завдання, що полягало в розробці комп'ютеризованої системи для контролю за умовами зберігання лікарських засобів в шафі для функціювання в двох температурних діапазонах: від 8 °С до 15 °С та від 15 °С до 25 °С з недопусканням перевищення вологості більше 60%. В результаті виконання цієї мети було отримано наступні результати:

1. Виконано огляд вимог до зберігання ліків в Україні та розгляд існуючих рішень.
2. Розроблено узагальнену структуру комп'ютерної системи та виконано обґрунтування вибору апаратного забезпечення.
3. Розроблено електричну принципову схему системи для контролю за умовами зберігання лікарських засобів.
4. Описано алгоритм роботи системи та написано програмний код для реалізації усіх задач, які описані в технічному завданні.

Розроблена комп'ютеризована система для контролю за умовами зберігання лікарських засобів була змонтована у вигляді робочого прототипу, створеного на основі платформи STM32.

Результати тестування показали, що усі режими працюють у повній відповідності до вимог технічного завдання.

Спроєктована система енергоефективна в порівняні з класичними рішеннями з осушувачами і охолоджувачами повітря, так як класичні системи мають великі енерговитрати і мають проблему відведення конденсату.

Спроєктована комп'ютеризована система можлива до застосування при виконанні задач щодо контролю за умовами зберігання лікарських засобів в шафі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. СТ-Н МОЗУ 42-5.1:2011. Настанова «Лікарські засоби. Належна практика зберігання».

2. Наказ «Про організацію зберігання в аптечних закладах різних груп лікарських засобів та виробів медичного призначення» № 44 від 16.03.1993р.

3. Автоматизація та диспетчеризація. URL: https://a-air.com.ua/ua-avtomatika-k-oborudovaniyu/.

4. Сімейство c.pco carel (карел). URL: https://eurocool.dp.ua/ua/p1026172108-semejstvo-cpco-carel.html.

5. Вибір правильного рішення для охолодження обладнання. URL: https://globalx-ua.com/vyibor-pravilnogo-resheniya-dlya-ohlajdeniya/.

6. Як зробити правильну вентиляцію. URL: https://aqua-life.ua/ua/kak-sdelat-pravilnuu-ventilyatsiu-v-ofise/.

7. CR 450 Medical Холодильна шафа аптечна/лабораторна Crystal S.A. URL: https://krist.com.ua/ua/p1556323893-450-medical-holodilnyj.html.

8. Datasheet STM32F401xB STM32F401xC.

9. Humidity sensor BME280. URL: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/.

10. Повітряна заслінка. URL: https://graviton.com.ua/ua/thermostats/add/air-flap/.

11. Дросель-клапани. URL: https://vents-shop.com.ua/aksesuary-uk/drossel-klapany-uk/.

12. Бібліотека управління дисплеями по SPI з DMA. Версія 1.4 (https://github.com/vadrov/stm32-display-spi-dma).

13. Метод ковзного середнього. URL: https://wiki.tntu.edu.ua/Метод_ковзного_середнього/.

14. Модуль для роботи з кнопками на мікроконтролері STM32F4 (https://github.com/vadrov/stm32-button-nobounce-autorepeat-buffer).

15. Підключення до мікроконтролера STM32 SD картки по SPI с DMA (https://www.youtube.com/watch?v=qrJ31JwTxZs).

16. Проєкт роботи файлового менеджера з використанням бібліотеки FATFS (https://github.com/vadrov/stm32-filemanager-sd-fatfs-display-buttons).

17. Сервопривід. Принцип роботи. Гальванічна розв'язка. Підключення до STM32. Практика #03. URL: https://github.com/Solderingironspb/Lessons-Stm32/tree/Servo/.

18. Техніка безпеки в аптеці. URL: https://www.mpi-dpr.com.ua/uk/blog/tekhnika-bezpeki-v-apteci-.

19. Інструкція по організації зберігання в аптечних закладах різних груп лікарських засобів і предметів медичного призначення. URL: https://zakon.rada.gov.ua/rada/show/v0044282-93#Text.

20. Основи охорони праці: Підручник 2-ге видання, доповнене та перероблене. К.Н.Ткачук, М.О. Халімовський, В.В. Зацарний, С.В. Зеркалов Р.В. Сабарно, О.І. Полукаров, В.С. Козяков, Л.О. Мітюк. За ред. К.Н. Ткачука і М.О. Халімовського – К.: Основа, 2006. 448с.

21. Коротке замикання: що це та які причини. URL: https://vse-e.com/ua/novosti/korotkoe-zamykanie-chto-eto-i-kakie-prichiny/.

22. Гурик О.Я., Король О.І., Сенчишин В.С. Методичні вказівки до лабораторної роботи з дисципліни “Основи охорони праці”. Тернопіль, 2006. 17 с.

23. Охорона праці під час виробництва лікарських засобів. URL: https://pro-op.com.ua/article/1265-ohorona-prats-pd-chas-virobnitstva-lkarskih-zasobv/.

ДОДАТОК А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра комп’ютерних систем та мереж

«ЗАТВЕРДЖУЮ»
Завідувач кафедри КС
_____ Осухівська Г. М.

«____» _____ 2024 р.

КОМП’ЮТЕРИЗОВАНА СИСТЕМА КОНТРОЛЮ ЗА УМОВАМИ ЗБЕРІГАННЯ ЛІКАРСЬКИХ ЗАСОБІВ

ТЕХНІЧНЕ ЗАВДАННЯ

на 9 листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп’ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІ-41
_____ к.т.н. Жаровський Р. О.	_____ Невмержицький В. В.
«____» _____2024 р.	«____» _____2024 р.

Тернопіль 2024

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Комп’ютеризована система контролю за умовами зберігання лікарських засобів».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.123.00.00.

1.2 Виконавець

Студент групи СІ-41, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Невмержицький Віталій Володимирович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету № 4/7-408 від 24.04.2024

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 29.02.2024 р.

Плановий термін завершення виконання кваліфікаційної роботи – 26.06.2024 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ІСО, ГОСТ, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи.

Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Система призначена для контролю за умовами зберігання лікарських засобів в шафі.

2.2 Мета створення системи

Результатом має бути функціональний та енергоефективний пристрій, здатний цілодобово забезпечувати параметри вологості менше 60% та температури нижче 25 °C і вище 8 °C в шафі для зберігання лікарських засобів чутливих до вологості таких як лікарська рослинна сировина. Необхідно

передбачити два температурних діапазони роботи системи: від 8 °С до 15 °С та від 15 °С до 25 °С.

2.3 Характеристика об'єкту

Система розробляється для контролю за умовами зберігання лікарських засобів і включає в себе:

- розробку структурної схеми;
- розробку схеми електричної принципової;
- розробку алгоритму роботи та програмного забезпечення для мікроконтролера.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

Необхідно розробити комп’ютеризовану систему контролю за умовами зберігання лікарських засобів, яка повинна бути економічною та енергоефективною. Система повинна зменшувати вологість в шафі зберігання лікарських засобів та знижувати температуру керуючи примусовою вентиляцією повітря зі зовнішнього середовища. А також повинна запобігати переохолодженню лікарських засобів, тобто не допускати зниження температури нижче 8 °С.

3.1.1 Вимоги до структури та функціонування системи

Структура системи контролю за умовами зберігання лікарських засобів включає в себе:

- мікроконтролер сімейства STM32, який забезпечує загальне керування функціонуванням системи;

- дисплей для відображення параметрів, стану системи;
- кнопки для налаштування системи та її керування;
- сенсори для вимірювання температури та вологості;
- вентилятори та фільтри;
- сервоприводи та заслінки;
- обігрівач повітря;
- реле для комутації вентиляторів, сервоприводів, обігрівача повітря.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Обмін даними між компонентами системи контролю за умовами зберігання лікарських засобів повинен здійснюватись з використанням провідних технологій.

3.1.3 Вимоги до режимів функціонування системи

Необхідно передбачити можливість вибору наступних режимів роботи системи:

1. Вимкнено. Примусова вентиляція вимкнена (повітряні заслінки закриті, вентилятори вимкнено). Робота режиму не залежить від вологості та температури.
2. Режим вентиляції. Примусова вентиляція увімкнена (повітряні заслінки відкриті, вентилятори увімкнені). Робота режиму не залежить від вологості та температури.
3. Режим охолодження. Примусова вентиляція увімкнена якщо температура в шафі вища температури ззовні. Робота режиму не залежить від вологості.

4. Режим осушення. Примусова вентиляція увімкнена якщо вологість в шафі більша ніж вологість ззовні.

У всіх режимах при досягненні в шафі дозволеної мінімальної температури примусова вентиляція повинна заборонятись. Також, у всіх режимах при зниженні температури до критичної (8 °C) повинен вмикатись підігрів та внутрішня вентиляція для запобігання переохолодження.

Для можливості вибору режиму необхідно розробити інтерфейс користувача.

З метою реалізації цілодобового контролю у всіх режимах повинні збиратись статистичні дані по вологості та температурі. Вимоги до періоду збереження даних: мінімальні та максимальні значення за весь час роботи системи з вказанням дати та часу, поточні значення за кожну добу зі збереженням в окремому файлі для кожної доби. Зібрані статистичні дані необхідно зберігати на SD-карту.

Необхідно реалізувати можливість вибору файлу визначеної дати та відображення графіків параметрів за вказану добу за допомогою файлового менеджера.

Система повинна автоматично реагувати на зміни температури та вологості:

- зменшувати вологість чи температуру шляхом створення примусової вентиляції з зовнішнім середовищем за допомогою повітряних заслінок і вентиляторів;
- запобігати зниженню температури в шафі нижче критичної (8 °C) шляхом увімкнення обігрівача повітря.

3.1.4 Перспективи розвитку та модернізації системи

Передбачаються перспективи розвитку системи, що включають можливість додавання безпровідного підключення датчиків вимірювання

параметрів повітря, автоматичного тестування працездатності зовнішніх механічних пристроїв шляхом додавання датчиків струму і контролювання сили струму в момент роботи пристрою, додавання можливості керування та моніторингу через Web-інтерфейс з підключенням до Інтернет.

3.1.5 Вимоги до надійності системи

Система повинна бути захищена від фізичних чи механічних пошкоджень на рівні апаратного та програмного забезпечення. Надійність системи повинна забезпечувати відновлюваність функціонування у випадку збою апаратного чи програмного забезпечення. Імовірність безвідмовної роботи системи повинна складати не менше 99,6 %.

3.1.6 Вимоги до функцій та задач, які виконує система

Функціональні вимоги та задачі, які повинна виконувати система, полягають в наступному:

- коректне відображення інформації на дисплеї;
- коректна робота у всіх запланованих режимах;
- можливість управління за допомогою кнопок користуючись розробленим інтерфейсом;
- зберігання на SD-диск параметрів температури та вологості за кожен день;
- наявність файлового менеджера для можливості вибору архівних файлів зі збереженими статистичними даними.

3.1.7 Вимоги до апаратного забезпечення

- режими роботи і умови експлуатації вибраних елементів повинні відповідати вказаним в ТЗ;

- вибрана елементна база має забезпечувати необхідні режими роботи системи;

- елементна база по можливості має бути широковживаною, доступною і дешевою. Необхідно також враховувати можливість заміни вибраних елементів на аналогічні (вітчизняні чи імпортного виробництва).

Мікроконтролер повинен бути із сімейства STM32, повинен містити необхідний набір вбудованих периферійних пристроїв (таймери, комунікаційні шини т.п.) та потрібну кількість керованих портів введення /виведення.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ.

Комплект документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

1. Структурна схема системи.
2. Схема електрична принципова.
3. Блок-схема алгоритму роботи програми.
4. Програмний інтерфейс.

*Примітка: У комплект документації можуть вноситися зміни та доповнення в процесі розробки.

5 Техніко-економічні показники.

Собівартість розробки системи повинна становити не більше 6000 грн.

Термін експлуатації системи повинен бути не менший 5 років.

*Примітка: собівартість системи може змінюватись під час розрахунку в процесі розробки.

6 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ етапу	Назва етапу виконання кваліфікаційної роботи	Терміни виконання
1	Розробка технічного завдання	01.02-09.02
2	Аналіз технічного завдання	05.02 – 11.02
3	Аналіз та обґрунтування можливих рішень	25.04 – 03.05
4	Розробка структурної схеми	04.05-15.05
5	Розробка електричної принципової схеми, вибір елементної бази	16.05-25.05
6	Розробка програмного забезпечення для проєктованої системи	26.05-09.06
7	Опрацювання питань розділу «Безпека життєдіяльності, основи охорони праці»	10.06-15.06
8	Виготовлення тестової версії системи на макетній платі	16.06-20.06
9	Оформлення пояснювальної записки кваліфікаційної роботи	16.06-20.06
10	Оформлення графічної частини	16.06-20.06
11	Попередній захист кваліфікаційної роботи	14.06
12	Захист кваліфікаційної роботи	

7 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

ДОДАТОК Б

Код програмного забезпечення

```
/* USER CODE BEGIN Header */
/*
  *
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------*/
#include "main.h"
#include "fatfs.h"

/* Private includes ----------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "string.h"
/*------- Підключення користувацьких модулів та драйверів ---------*/
#include "../../Keyboard/keyboard.h"       //модуль роботи з кнопками
#include "../../Display/display.h"         //драйвер роботи з
дисплеями по SPI
#include "../../Display/ili9341.h"         //драйвер для дисплея
ILI9341
#include "../../Filemanager/filemanager.h" //модуль файлового
менеджера
#include "../../MyString/mystring.h"       //бібліотека роботи з
рядками

#include "./BME280/bme280.h"                 //бібліотека роботи з
датчиком bme280
#include <stdio.h>
#include <math.h>

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------*/
```

```
I2C_HandleTypeDef hi2c1;

IWDG_HandleTypeDef hiwdg;

RTC_HandleTypeDef hrtc;

TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;

/* USER CODE BEGIN PV */

//В функції static void MX_RTC_Init(void) є дві структури, їх
оголошуємо як глобальні для виводу часу, дати
RTC_TimeTypeDef sTime;
RTC_DateTypeDef sDate;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_SPI1_Init(void);
static void MX_TIM10_Init(void);
static void MX_SPI2_Init(void);
static void MX_TIM3_Init(void);
static void MX_I2C1_Init(void);
static void MX_TIM1_Init(void);
static void MX_RTC_Init(void);
static void MX_TIM2_Init(void);
static void MX_IWDG_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */

/* Private user code ---------------------------------------------*/
/* USER CODE BEGIN 0 */

//функція для відладки по SWO. Використовуємо: printf("t=%d",t);
/*
int _write(int file, uint8_t *ptr, int len)
{
     for (int DataIdx = 0; DataIdx < len; DataIdx++)
     {
          ITM_SendChar(*ptr++);
     }
     return len;
}
*/
//Оголошення функцій
void show_time(void);                    // відображення поточного
часу, дати
```

```
void get_time(int8_t F_update);         // отримання актуальних дати,
часу і формування масиву TimeBuf для виводу
void SetTime(void);                     // встановлення заданих часу,
дати
void resetTimeDate(void);               // скидання дат
мін/максимальних статистичних значень
void writeSD();                              // Запис на SD-карту
void write_setting_SD(void);       // запис на SD структури
settingRAM
void write_chart_SD(void);              // запис на SD структури
chart_dry
void readSD();                          // Читання даних з SD-карти
void read_setting_SD(void);             // Читання з SD-карти
структури settingRAM
void read_chart_SD(void);               // Читання з SD-карти
структури chart_dry
void resetTemp(void);                   // скидання min max значень
void dry_static(int8_t IsP);            // Друк статичної картинки на
інфо екранах 2 і 3
void printChart(int8_t IsP);       // Друк графіка на інфо екранах 2
і 3
void print_status0(void);               // Друк значень на інфо
екранах 2 і 3
void dry_get_data(void);           // заповнення масиву графіка
void CheckEkran(void);                  // перевірка чи не пора
вимкнути дисплей
void dry_update(int8_t IsP);       // Оновлення графіків на інфо
екранах 2 і 3
void CheckON(void);                     // Перевірка статусу системи,
чи не потрібно увімкнути/вимкнути зовнішні пристрої
void Setting(uint8_t Flag_Change); //друк інформації про поточний
режим
void kvadrat(uint32_t color);      // відображення квадрату вказаним
кольором
void cubik(void);                       // зафарбовування квадрата
вказаним кольором
void printSetTimeStatic(void);          // Екран "Налаштування
Часу/Дати"
void Print_time_date(uint8_t *Dim_TimeDate, int x, int y, uint32_t
color);//форматування часу/дати у формат "12:34 12.12.17" і друк
вказаним кольором по вказаних координатах
void printSetTimeUpdate(void);          // Оновлення екрану
"Налаштування Часу/Дати"
void printSettingStatic(void);          //друк екрану налаштувань з
вибором режиму роботи
void printInfoStatic0(void);       // Друк статичної картинки екрана
0
void printInfoStatic1(void);       // Друк статичної картинки екрана
1
```

```
void Print_period(uint32_t period, int x, int y, uint32_t color);
//форматування тривалості часу у формат "123дн 13год 21хв" і друк
вказаним кольором по вказаних координатах
void printInfoUpdate0(void);       // Друк значень на екрані 0
void printInfoUpdate1(void);       // Друк значень на екрані 1
//void servo_deg(uint8_t Nom_serv, uint8_t deg); //функція управління
сервоприводами, Nom_serv - номер сервопривода, deg - кут повороту в
градусах
void MotorON(void);                     //увімкнення вентиляторів,
відкриття повітряних заслінок
void MotorOFF(void);                    //вимкнення вентиляторів,
закриття повітряних заслінок
void MotorON_SRV(void);                 //увімкнення вентиляторів,
відкриття повітряних заслінок з серво-приводом
void MotorON_EM(void);                  //увімкнення вентиляторів,
відкриття повітряних заслінок з електро-магнітним механізмом
void MotorOFF_SRV(void);           //вимкнення вентиляторів, закриття
повітряних заслінок з серво-приводом
void MotorOFF_EM(void);                 //вимкнення вентиляторів,
закриття повітряних заслінок з електро-магнітним механізмом
int16_t F_H_In_err(float H_X);          //приведення показів
вологості до одного рівня
int16_t F_T_In_err(float T_X);          //приведення показів
температури до одного рівня
int16_t calculationAbsH_P(float t, float h, float p); //функція для
обчислення значення абсолютної вологості
void readDHT(void);                     //читання параметрів з
датчиків
void OpenFileManager (void);       //відкриття екрана файлового
менеджера
int8_t button_click_handler(void); // обробник натискання кнопок
void Goto_New_Window(void);             //функція відображає новий
інфо екран

void save_history(void);           //збереження параметрів на SD-
карту
void read_history(char *file_name, int8_t show_error); // Читання
історичних даних з SD-карти в структуру data_history та виведення
графіків
void read_today(char *file_name, int8_t show_error);   // Читання
даних з SD-карти за поточну добу в структуру data_today

void printColorMode(void);              //Екран "Колірна схема"
void printDemoMode(void);               //Екран "Демонстраційний
режим"

void printAirDampers(void);             //Екран "Налаштування
повітряних заслінок"
void printServoMode(void);              //Екран "Тип повітряних
заслінок"
```

```
void printSetServo(void);               //Екран "Налаштування серво-
приводів"
void printSetServoUpdate(void);         //Оновлення екрану
"Налаштування серво-приводів"

void Open_Servo(uint8_t Nom_serv); //відкриття повітряної заслінки з
серво-приводом
void Close_Servo(uint8_t Nom_serv);     //закриття повітряної
заслінки з серво-приводом

void printKorDat(void);                 //Друк екрану "Коригування
датчиків"
void KorDatUpdate(void);           // Оновлення екрану "Коригування
датчиків"
void SetKorDat(void);                   // збереження заданих
параметрів для приведення показів датчиків до одного рівня
(коригування)

//   - текстові масиви   ---------------------------------------
// масив для форматування інформації перед виводом її на дисплей
(температура, вологість і тиск)
char TempBuf[60];

// масив для форматування інформації перед виводом її на дисплей
(дата, час)
char TimeBuf[15];

// масив використовується для перекодування тексту
char OutputBuf[255];

// - КОНСТАНТи -----------------------------------------
#define dH_OFF          5         // Гистерезис абсолютної вологості
в сотих грама на куб.метр
#define dT_OFF          11        // Гистерезис температури в сотих
градуса
#define TEMP_LOW        80        // Критична температура в шафі -
переохолодження (в сотих градуса) - вентилятори вимикаються і
вмикається обігрівач

// - ЧАСОВІ КОНСТАНТи ----------------------------------------
#define TIME_SCAN_SENSOR 3              // Період опитування
датчиків, сек.
#define TIME_HOUR            3600      // Кількість сек. в 1 год
#define TIME_MIN             20             // константа для
коректного вираховування часу роботи системи
                                              // = 60 сек. / 3
сек. (TIME_SCAN_SENSOR) = 20

int16_t TIME_PRINT_CHART = 720;             // Період виводу однієї
точки графіка, сек. 12 хв. = 720 сек.
```

```
volatile int8_t     flag_measurement    =    0;   // Ознака роботи функції використання датчиків по таймеру
//volatile - ключеве слово, яке каже компілятору, що змінна може змінитись поза межами основного циклу. В нашому випадку змінні будуть мінятись при перериванні по таймеру

#define TIME_EKRAN       180            //час очікування натискання кнопок, після якого підсвідка дисплея вимикається

volatile int16_t hour_ekran   =    0;        //час роботи дисплея після останнього натискання кнопки

//   - Налаштування -------------------------------------
#define NUM_SETTING    8                        // Кількість режимів роботи
#define BLOCK_OFF      0                        // Вимкнено (режим роботи)
#define HOOD_ON        1                        // Режим вентиляції (режим  роботи)
#define COOLING        2                        // Режим охолодження (режим  роботи)

#define NUM_INFO_SCREEN  4          // Кількість інформаційних екранів
#define NUM_POS               15        // Кількість позицій на екрані налаштувань
#define NUM_SCREEN       17         // Кількість екранів

#define dist   8+14           //для дистанціювання рядків при виводі

volatile int8_t packet_ready  =    0;             // 1 -  поступили нові дані з датчиків
int8_t packet_ready_E0        =    1;                  // 1 - поступили нові дані на екран 0
int8_t packet_ready_E1        =    1;                  // 1 - поступили нові дані на екран 1

int8_t packet_ready_Time =    1;                  // 1 -  поступили нові дані Time (змінилась дата чи час)

uint8_t last_min = 200;            // для виводу часу
uint8_t last_day = 200;            // для виводу дати

LCD_Handler *lcd;                  // оголошуємо вказівник на дисплей
File_Manager_Handler *fm;          // оголошуємо обробника файлового менеджера
File_Manager_Color_Scheme *color_scheme[4];// Оголошення масиву вказівників на колірні схеми

// Структура для збереження даних на SD-карту
typedef struct DataForSave
{
```

```
    int8_t    mode;               // режим роботи
    uint32_t hour_unit;      // загальний час роботи - вимірюється в
інтервалах сканування датчиків = TIME_SCAN_SENSOR
    uint32_t hour_motor;          // час роботи вентиляторів -
вимірюється в інтервалах сканування датчиків = TIME_SCAN_SENSOR
    uint32_t hour_heat;      // час роботи обігрівача - вимірюється
в інтервалах сканування датчиків = TIME_SCAN_SENSOR
    int16_t  tOutMin, tInMin;     // Мінімальні температури за
період спостереження - в сотих градуса
    int16_t  tOutMax,tInMax; // Максимальні температури за період
спостереження - в сотих градуса
    int16_t  hMax, hMin;          // Максимальні/Мінімальні
вологість в шафі за період спостереження - в сотих долях %
    int16_t   tick_SDflash;       // Змінна для збереження часу що
пройшло після останнього збереження статистики,
                          //   порівнюємо з TIME_HOUR і обнуляємо
змінну та зберігаємо статистику

    uint8_t tOutMinTimeDate[5];   //minute,hour,day,month,year
    uint8_t tOutMaxTimeDate[5];   //minute,hour,day,month,year
    uint8_t tInMinTimeDate[5];    //minute,hour,day,month,year
    uint8_t tInMaxTimeDate[5];    //minute,hour,day,month,year
    uint8_t hMaxTimeDate[5]; //minute,hour,day,month,year
    uint8_t hMinTimeDate[5]; //minute,hour,day,month,year

    uint8_t SaveTimeDate[5]; //Час, Дата останнього збереження даних
на SD-карту

    uint8_t        flags;    // байт ознак
                   // 0 біт - мотор увімкнено/вимкнено
                   // 1 біт - увімкнено/вимкнено
                   // 2 біт -[1 - dH_min задається в сотих грама на
м*3] [0 - dH_min задається в процентах від absHIn]
                   // 3 біт - дисплей увімкнено
                   // 4 - MOTOR_IN_BIT - запуск вентиляторів
                   // 5 - MOTOR_OUT_BIT - вимкнення вентиляторів
                   // 6-7 -  ПУСТО
    int8_t    color_mode;    // Колірна схема: 0, 1, 2, 3
    int8_t    demo_mode;     // Вимкнено = 0 / Увімкнено = 1
демонстраційний режим
    int8_t    servo_mode;    // Тип повітряних заслінок: з серво-
приводом = 0, з магнітним механізмом = 1
    uint8_t N_Screen;   // Номер поточного інфо екрана
//приведення показів датчиків до одного рівня (коригування)
    int8_t    T1;       //температура T1 в °C
    int16_t   d_T1;          //різниця між датчиками в сотих °C при
температурі T1

    int8_t    T2;       //температура T2 в °C
    int16_t   d_T2;          //різниця між датчиками в сотих °C при
температурі T2
```

```
    uint8_t   H1;       //вологість H1 в %
    int16_t   d_H1;          //різниця між датчиками в сотих % при
вологості H1

    uint8_t   H2;       //вологість H2 в %
    int16_t   d_H2;          //різниця між датчиками в сотих % при
вологості H1

    int16_t   serv_90_deg;   //значення для повороту сервопривода на
90° - відкрито
    int16_t   serv_0_deg;    //значення для повороту сервопривода на
0° - закрито

}   DataForSave;

DataForSave *settingRAM;

DataForSave* DataForSaveNew(void)
{
    DataForSave *stng = (DataForSave*)calloc(1,
sizeof(DataForSave));
    stng->mode              = 0;
    stng->hour_unit         = 0;
    stng->hour_motor    = 0;
    stng->hour_heat         = 0;
    stng->tOutMin       = 5555;
    stng->tInMin        = 5555;
    stng->tOutMax       = -5555;
    stng->tInMax        = -5555;
    stng->hMax          = 0;
    stng->hMin          = 10000;
    stng->tick_SDflash  = 0;
    stng->flags         = 0x00;
    stng->color_mode    = 3;
    stng->demo_mode         = 0;
    stng->servo_mode    = 0;
    stng->N_Screen      = 0;

    stng->T1       = 7;
    stng->d_T1          = 0;
    stng->T2       = 25;
    stng->d_T2          = 0;
    stng->H1       = 32;
    stng->d_H1          = 0;
    stng->H2       = 72;
    stng->d_H2          = 0;

//мінімум = 3900 при serv_90_deg = 0
//максимум = 9000 при serv_0_deg = 255
//   9000 - 3900 = 5100;      коеф = 5100 / 255 = 20
```

```
    stng->serv_90_deg   = 59;     //59*20 + 3900 = 5080;
    stng->serv_0_deg    = 205;    //205*20 + 3900 = 8000;

    return stng;
}

#define MOTOR_BIT                   0           // біт вентилятора в
settingRAM->flags
#define HEAT_BIT                    1           // біт обігрівача в
settingRAM->flags
#define ABS_H_BIT                   2           // біт кодування
вологості абс или % в settingRAM->flags
#define EKRAN_BIT                   3           // біт увімкнення
дисплея
#define MOTOR_IN_BIT                4           // біт, показує що
необхідно: видкрити заслінки, увімкнути вентилятори
#define MOTOR_OUT_BIT               5           // біт, показує що
необхідно: закрити заслінки, вимкнути вентилятори

#define FLAG_ABS_H_ON             settingRAM->flags |= (1<<ABS_H_BIT)
// біт ABS_H_BIT встановити в 1
#define FLAG_ABS_H_OFF            settingRAM->flags &=
~(1<<ABS_H_BIT)  // біт ABS_H_BIT встановити в 0
#define FLAG_ABS_H_CHECK          settingRAM->flags & (1<<ABS_H_BIT)
// біт ABS_H_BIT перевірити на 1

#define FLAG_EKRAN_ON             settingRAM->flags |= (1<<EKRAN_BIT)
// біт дисплея встановити в 1
#define FLAG_EKRAN_OFF            settingRAM->flags &=
~(1<<EKRAN_BIT)  // біт дисплея встановити в 0
#define FLAG_EKRAN_CHECK          settingRAM->flags & (1<<EKRAN_BIT)
// біт дисплея перевірити на 1
#define EKRAN_ON                  { LCD_SleepOut(lcd); FLAG_EKRAN_ON;
hour_ekran     =    0;}  //виведення дисплея зі сплячого режиму
(увімкнення відображення дисплея та підсвічування)
#define EKRAN_OFF                 { LCD_SleepIn(lcd); FLAG_EKRAN_OFF;
}         //переведення дисплея в "сплячий режим" (вимкнення
відображення дисплея та підсвічування)

#define FLAG_MOTOR_ON                   settingRAM->flags |=
(1<<MOTOR_BIT)   // біт вентилятора встановити в 1
#define FLAG_MOTOR_OFF                  settingRAM->flags &=
~(1<<MOTOR_BIT)  // біт вентилятора встановити в 0
#define FLAG_MOTOR_CHECK           settingRAM->flags & (1<<MOTOR_BIT)
// біт вентилятора перевірити на 1

#define FLAG_HEAT_ON              settingRAM->flags |= (1<<HEAT_BIT)
// біт обігрівача встановити в 1
#define FLAG_HEAT_OFF             settingRAM->flags &= ~(1<<HEAT_BIT)
// біт обігрівача встановити в 0
```

```
#define FLAG_HEAT_CHECK           settingRAM->flags & (1<<HEAT_BIT)
// біт обігрівача перевірити на 1
#define HEAT_ON                   {
HAL_GPIO_WritePin(PIN_HEAT_GPIO_Port, PIN_HEAT_Pin, GPIO_PIN_SET);
FLAG_HEAT_ON;  }    // увімкнути обігрівач
#define HEAT_OFF                  {
HAL_GPIO_WritePin(PIN_HEAT_GPIO_Port, PIN_HEAT_Pin, GPIO_PIN_RESET);
FLAG_HEAT_OFF; }    // вимкнути обігрівач


#define FLAG_MOTOR_IN_ON           { settingRAM->flags |=
(1<<MOTOR_IN_BIT); settingRAM->flags &= ~(1<<MOTOR_OUT_BIT); }
#define FLAG_MOTOR_IN_OFF              settingRAM->flags &=
~(1<<MOTOR_IN_BIT)
#define FLAG_MOTOR_IN_CHECK            settingRAM->flags &
(1<<MOTOR_IN_BIT)

#define FLAG_MOTOR_OUT_ON              { settingRAM->flags |=
(1<<MOTOR_OUT_BIT); settingRAM->flags &= ~(1<<MOTOR_IN_BIT); }
#define FLAG_MOTOR_OUT_OFF             settingRAM->flags &=
~(1<<MOTOR_OUT_BIT)
#define FLAG_MOTOR_OUT_CHECK       settingRAM->flags &
(1<<MOTOR_OUT_BIT)

//#define MOTOR_ON       { HAL_GPIO_WritePin(PIN_RELAY_GPIO_Port,
PIN_RELAY_Pin, GPIO_PIN_SET); FLAG_MOTOR_ON;  }        // увімкнути
вентилятори
//#define MOTOR_OFF      { HAL_GPIO_WritePin(PIN_RELAY_GPIO_Port,
PIN_RELAY_Pin, GPIO_PIN_RESET); FLAG_MOTOR_OFF; } // вимкнути
вентилятори

#define MOTOR_ON                   {    if(~FLAG_MOTOR_IN_CHECK) {
FLAG_MOTOR_IN_ON; packet->Step_Motor_In = 0;}     }    // увімкнути
вентилятори
#define MOTOR_OFF                  {    if(~FLAG_MOTOR_OUT_CHECK) {
FLAG_MOTOR_OUT_ON; packet->Step_Motor_Out = 0;}   }    // вимкнути
вентилятори

// Структура для графіків історії, що зберігаються на SD-карту
typedef struct DATA_HISTORY
{
// 1 день = 24 год. х 60 хв. = 1440 хв.
// точку на графіку ставимо кожні 12 хв.
// 1400 хв. / 12 хв. = 120 - кількість точок на шкалі часу для 24
год.
// індекс в масиві визначає час отримання значень: 0 - вимірювання в
діапазоні часу від 00:00 до 00:12,
//  1 - до 00:24 ... 119 - від 23:48 до 00:00
     uint8_t H_Date;          //день
     uint8_t H_Month;    //місяць
     uint8_t H_Year;          //рік
```

```
    int16_t tOut[120];  //температура ззовні - в сотих градуса
    int16_t tIn[120];   //температура в шафі - в сотих градуса
    int16_t absHOut[120];    //абсолютна вологість ззовні - в сотих
грама на м*3
    int16_t absHIn[120];     //абсолютна вологість в шафі - в сотих
грама на м*3
    int16_t relHOut[120];    //відносна вологість ззовні - в сотих
відсотка
    int16_t relHIn[120];     //відносна вологість в шафі - в сотих
відсотка
    int16_t dH[120];    //різниця ззовні та в шафі значень відносної
вологості - в сотих грама на м*3
    int8_t Motor[120];  //ознака роботи вентилятора під час
інтервалу графіка, якщо вентилятор працював
                        //(навіть якщо одне вимірювання), то на графіку
фон міняється
    int8_t Heat[120];   //ознака роботи обігрівача під час інтервалу
графіка, якщо обігрівач працював
                        //(навіть якщо одне вимірювання), то на графіку
фон міняється
}    DATA_HISTORY;

//функція задання початкових значень структури DATA_HISTORY
DATA_HISTORY* DATA_HISTORY_New(void)
{
    DATA_HISTORY *chdr = (DATA_HISTORY*)calloc(1,
sizeof(DATA_HISTORY));

    //початкові значення
    chdr->H_Date   = 0;
    chdr->H_Month  = 0;
    chdr->H_Year   = 0;
    for (uint8_t k = 0; k < 120; k++)
    {
        chdr->tOut[k]   = -1500;
        chdr->tIn[k]    = -1500;
        chdr->absHOut[k]     = 0;
        chdr->absHIn[k] = 0;
        chdr->dH[k]          = 0;
        chdr->relHOut[k]     = 0;
        chdr->relHIn[k] = 0;
        chdr->Motor[k]  = 0;
        chdr->Heat[k]   = 0;
    }
    return chdr;
}

//оголошення вказівників на структури DATA_HISTORY
DATA_HISTORY* data_today;//структура для збереження даних за поточну
добу на SD-карту
```

```
DATA_HISTORY* data_history;//структура для зчитування даних за
вказану дату з SD-карти з наступним виведенням на екран

// структура для графіків
typedef struct CHART_DRY
{
// 1 день = 24 год. х 60 хв. = 1440 хв.
// точку на графіку ставимо кожні 12 хв
// 1400 хв. / 12 хв. = 120 - кількість точок на шкалі часу для 24
год.
// в масивах зберігаються вирахувані значення (координата Y) для
побудови графіків
     uint8_t tOutChart[120];       //температура ззовні
     uint8_t tInChart[120];        //температура в шафі
     uint8_t absHOutChart[120];    //абсолютна вологість ззовні
     uint8_t absHInChart[120];     //абсолютна вологість в шафі
     uint8_t relHOutChart[120];    //відносна вологість ззовні
     uint8_t relHInChart[120];     //відносна вологість в шафі
     uint8_t dHChart[120];         //різниця ззовні та в шафі значень
відносної вологості

     uint8_t posChart;        // позиція в масиві графиків - початок
виводу від 0 до 120-1
     uint8_t TimeChart;       // час до виводу чергової точки на
графік
     int8_t ChartMotor;       // ознака роботи вентилятора під час
інтервалу графіка, якщо
                           //вентилятор працював (навіть якщо одне
вимірювання), то на графіку фон міняється
     int8_t ChartHeat;        // ознака роботи обігрівача під час
інтервалу графіка, якщо обігрівач
                           //працював (навіть якщо одне вимірювання),
то на графіку фон міняється
     int8_t CoolData;         // ознака наявності нових даних, щоб
зайвий раз не виводити на екран одне й теж
}    CHART_DRY;

//функція задання початкових значень структури CHART_DRY
CHART_DRY* CHART_DRY_New(void)
{
     CHART_DRY *chdr = (CHART_DRY*)calloc(1, sizeof(CHART_DRY));
     chdr->posChart = 0;
     chdr->TimeChart     = 0;
     chdr->ChartMotor= 0;
     chdr->ChartHeat = 0;
     chdr->CoolData = 0;

     //початкові значення для графіків
     for (uint8_t k = 0; k < 120; k++)
     {
         chdr->tOutChart[k]   = 50;
```

```
          chdr->tInChart[k]    = 50;
          chdr->absHOutChart[k] = 0;
          chdr->absHInChart[k] = 0;
          chdr->dHChart[k]     = 0;
          chdr->relHOutChart[k] = 0;
          chdr->relHInChart[k] = 0;
     }
     return chdr;
}

//оголошення вказівників на структури CHART_DRY
CHART_DRY* chart_dry;    // структура  для графіків - інфо екрани 2
та 3. Значення за останні 24 години
CHART_DRY* chart_history;// структура  для графіків історії за
вибрану дату

// структура Packet для збереження поточних значень
typedef struct Packet
{
     int16_t        tOut;          // поточна температура ззовні - в
сотих градуса
     int16_t        tIn;      // поточна температура в шафі - в сотих
градуса
     int16_t        absHOut;  // абсолютна вологість ззовні - в сотих
грама на м*3
     int16_t        absHIn;        // абсолютна вологість в шафі - в
сотих грама на м*3
     int16_t        relHOut;  // відносна вологість ззовні - в сотих
відсотка
     int16_t        relHIn;        // відносна вологість в шафі - в
сотих відсотка
     int16_t        p;        // поточний тиск - в десятих гПа
     uint8_t        dH_min;        // поріг увімкнення вентилятора по
різниці абсолютної вологості в сотих грама на м*3
                         // або в % (залежить від flags:2), тільки
позитивні значення
     uint8_t        T_min;         // поріг вимкнення вентилятора по
температурі в долях градуса, тільки позитивні значення
     uint8_t        Step_Motor_In; // номер кроку процесу запуску
вентиляторів
     uint8_t        Step_Motor_Out;     // номер кроку процесу
вимкнення вентиляторів
}    Packet;

//функція задання початкових значень структури Packet
Packet* PacketNew(void)
{
     Packet *pct = (Packet*)calloc(1, sizeof(Packet));
     pct->tOut      = 500;    //+5 °C
     pct->tIn       = 500;    //+5 °C
     pct->absHOut   = 321;          //3.21 грама на м*3
```

```
    pct->absHIn         = 321;    //3.21 грама на м*3
    pct->relHOut   = 5000;        //50%
    pct->relHIn         = 5000;   //50%
    pct->p              = 10345;//1034.5 гПа
    pct->dH_min         = 255;
    pct->T_min          = 255;
    pct->Step_Motor_In  = 0;
    pct->Step_Motor_Out = 0;
    return pct;
}

volatile Packet *packet; //оголошення вказівника на структуру Packet
під ім'ям packet

struct KILCE;  // Структура KILCE для навігації по меню

//оголошення функцій для роботи зі структурою KILCE
typedef void (*KilceSet_Poz)(struct KILCE *klc, uint8_t
Poz);//встановлення поточного значення позиції

typedef uint8_t (*KilceGet_Poz)(struct KILCE *klc);    //отримання
поточного значення позиції

typedef void (*KilceUp)(struct KILCE *klc);       //збільшення
позиції

typedef void (*KilceDown)(struct KILCE *klc);          //зменшення
позиції

typedef struct KILCE
{
    uint8_t _Min;       // мінімальне значення позиції
    uint8_t _Max;       // максимальне значення позиції
    volatile uint8_t _Poz;   // поточне значення позиції
    KilceSet_Poz Set_Poz;    // встановлення поточного значення
позиції
    KilceGet_Poz Get_Poz;    // отримання поточного значення позиції
    KilceUp Up;         // збільшення позиції
    KilceDown Down;          // зменшення позиції
}    KILCE;

static inline void DoSetPoz(KILCE *klc, uint8_t Poz)   //встановлення
поточного значення позиції
{    klc->_Poz = Poz;    }

static inline uint8_t DoGetPoz(KILCE *klc)        //отримання
поточного значення позиції
{    return klc->_Poz;   }

static inline void DoUp(KILCE *klc)               //збільшення
позиції
```

```
{    klc->_Poz == klc->_Max ? klc->_Poz = klc->_Min : klc->_Poz++;
     }

static inline void DoDown(KILCE *klc)             //зменшення позиції
{    klc->_Poz == klc->_Min ? klc->_Poz = klc->_Max : klc->_Poz--;
     }

KILCE* KILCE_New(uint8_t Min, uint8_t Max)   //функція задання
початкових значень структури KILCE
{
     KILCE *klc     = (KILCE*)calloc(1, sizeof(KILCE));
     klc->_Min = Min;
     klc->_Max = Max;
     klc->_Poz = Min;
     klc->Set_Poz   = DoSetPoz;
     klc->Get_Poz   = DoGetPoz;
     klc->Up        = DoUp;
     klc->Down = DoDown;
     return klc;
}
//оголошення вказівників на структури KILCE
KILCE *ScreenMode;       //для збереження номера поточного меню
KILCE *Screens[NUM_SCREEN];   //для збереження позиції в меню

// Структура AVARAGE для визначення середніх значень (значення
ковзного середнього)
// середньє арифметичне не застосовується, т.я. для нього потрібно
виділяти багато пам'яті
// застосовувати середнє значення необхідно для зглажування пульсацій
вимірювань
struct AVARAGE;

//оголошення функцій для роботи зі структурою AVARAGE
typedef void (*AveragePush)(struct AVARAGE *avrg, int16_t X);//ф-я
додає чергове значення і обчислюється ковзне середнє
typedef int16_t (*AverageMean)(struct AVARAGE *avrg);  //функція
вертає значення ковзного середнього

typedef struct AVARAGE
{
     volatile int32_t _sum;         // сума всіх значень
     volatile uint8_t _MaxSize;     // максимальна кількість значень
     volatile uint8_t _count; // поточна кількість значень
     volatile int16_t _mean;        // ковзне середнє значення
     AveragePush Push;         // функція додає чергове значення і
обчислюється ковзне середнє
     AverageMean Mean;         // функція вертає значення ковзного
середнього
}    AVARAGE;
```

```
static inline int16_t GetMean(AVARAGE *avrg) //функція вертає
значення ковзного середнього
{    return avrg->_mean; }

void DoPush(AVARAGE *avrg, int16_t X)   //функція додає чергове
значення і обчислюється ковзне середнє
{
     if(avrg->_count >= avrg->_MaxSize)
     {
          avrg->_sum =   (int32_t)(avrg->_sum - avrg->_mean + X);
          avrg->_mean =  (int16_t)(avrg->_sum / avrg->_MaxSize);
     }
     else
     {
          if(avrg->_count == 0)
          { avrg->_sum = X; avrg->_count = 1; avrg->_mean = X; }
          else
          {
               avrg->_count++;
               avrg->_sum = (int32_t)(avrg->_sum + X);
               avrg->_mean = (int16_t)(avrg->_sum / avrg->_count);
          }
     }
}

AVARAGE* AVARAGE_New(uint8_t size) //функція задання початкових
значень структури AVARAGE
{
      AVARAGE *avrg      = (AVARAGE*)calloc(1, sizeof(AVARAGE));
      avrg->_MaxSize     = size;
      avrg->_count  = 0;
      avrg->_sum    = 0;
      avrg->_mean   = 0;
      avrg->Push    = DoPush;
      avrg->Mean    = GetMean;
      return avrg;
}

//оголошення вказівників на структури AVARAGE

//для обчислення та збереження середніх значень показів датчиків
AVARAGE *ave_tIn;
AVARAGE *ave_tOut;
AVARAGE *ave_relHIn;
AVARAGE *ave_relHOut;
AVARAGE *ave_absHIn;
AVARAGE *ave_absHOut;
AVARAGE *ave_p;

//для приведення показів датчиків до одного рівня
AVARAGE *ave_Tsrd;
```

```
AVARAGE *ave_Hsrd;
AVARAGE *ave_dT;
AVARAGE *ave_dH;
AVARAGE *ave_tInK;
AVARAGE *ave_relHInK;

void test(void)
{
     uint8_t poz;
     poz = ScreenMode->Get_Poz(ScreenMode);
     sprintf(TempBuf, "SM=%2d S=%2d", poz, Screens[poz]-
>Get_Poz(Screens[poz]));

     LCD_WriteString(lcd, 0, 200, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
}

//збереження параметрів на SD-карту
void save_history(void)
{
     uint8_t m;

     if(settingRAM->demo_mode == 0)
     {
          m = 119 - (sTime.Hours * 60 + sTime.Minutes) / 12;
     }
     else
     {
          m = 119 - chart_dry->posChart;
     }

     if(m == 120) m = 119;

     if(data_today->H_Date != sDate.Date)    //якщо змінилась дата
     {                                                  //то готуємо
дані для нового файла
          data_today->H_Date = sDate.Date;
          data_today->H_Month = sDate.Month;
          data_today->H_Year = sDate.Year;

          for (uint8_t k = 0; k < 120; k++)
          {
               data_today->tOut[k] = -1500;
               data_today->tIn[k]  = -1500;
               data_today->absHOut[k] = 0;
               data_today->absHIn[k] = 0;
               data_today->dH[k] = 0;
               data_today->relHOut[k] = 0;
               data_today->relHIn[k] = 0;
               data_today->Motor[k] = 0;
```

```
                data_today->Heat[k] = 0;
            }
        }

        // зберігаємо поточні значення в структуру для SD-карти
        data_today->tOut[m] = ave_tOut->Mean(ave_tOut);
        data_today->tIn[m] = ave_tIn->Mean(ave_tIn);
        data_today->absHOut[m] = ave_absHOut->Mean(ave_absHOut);
        data_today->absHIn[m] = ave_absHIn->Mean(ave_absHIn);
        data_today->dH[m] = ave_absHIn->Mean(ave_absHIn) - ave_absHOut-
>Mean(ave_absHOut);
        data_today->relHOut[m] = ave_relHOut->Mean(ave_relHOut);
        data_today->relHIn[m] = ave_relHIn->Mean(ave_relHIn);
        data_today->Motor[m] = chart_dry->ChartMotor;
        data_today->Heat[m] = chart_dry->ChartHeat;


        //пишем структуру data_today на SD-карту в файл з іменем дати
        FRESULT res;
        FIL logFile;
        char File_Name[20];

        sprintf(File_Name, "%02d-%02d-20%02d.dat", sDate.Date,
sDate.Month, sDate.Year);

        //FA_OPEN_ALWAYS - відкриваємо/створюємо файл з доступом для
запису. Якщо файл вже існує, то дані перезаписуються
        res = f_open(&logFile, File_Name, FA_OPEN_ALWAYS | FA_WRITE);
        //відкриваємо/створюємо файл
       if(res != FR_OK)     //якщо помилка, то вивід повідомлення
        {
            my_strcpy(TempBuf, "помилка відкриття/створення файлу ");
            my_strcat (TempBuf, File_Name);

            LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
            return;
       }

       unsigned int bytesToWrite = sizeof(DATA_HISTORY);  //визначаємо
необхідний розмір в байтах для запису

       unsigned int bytesWritten;
       res = f_write(&logFile, data_today, bytesToWrite, &bytesWritten);
        //пишемо дані у файл
       if(res != FR_OK)     //якщо помилка, то вивід повідомлення
        {
            my_strcpy(TempBuf, "помилка запису у файл ");
            my_strcat (TempBuf, File_Name);
```

```
          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
        return;
     }

    if(bytesWritten < bytesToWrite)     //перевіряємо чи вистачило
місця
     {
          my_strcpy(TempBuf, "помилка: недостатьо місця на SD");
          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
     }

    res = f_close(&logFile);  //закриваємо файл
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
     {
          my_strcpy(TempBuf, "помилка закриття файлу ");
          my_strcat (TempBuf, File_Name);
          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
     }
}

// Читання даних з SD-карти за поточну добу в структуру data_today
// show_error == 0 - не показувати помилки, show_error == 1 -
показувати помилки,
void read_today(char *file_name, int8_t show_error)
{
     FRESULT res;
     FIL msgFile;
    res = f_open(&msgFile, file_name, FA_READ);   //відкриваємо файл
з доступом для читання
    if(res != FR_OK && show_error == 1) //якщо помилка, то вивід
повідомлення
     {
          my_strcpy(TempBuf, "помилка відкриття/створення файлу ");
          my_strcat (TempBuf, file_name);
          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
          return;
     }

    unsigned int bytesRead;
    res = f_read(&msgFile, data_today, sizeof(DATA_HISTORY),
&bytesRead);   //читаємо дані з файлу в структуру data_history
    if(res != FR_OK && show_error == 1) //якщо помилка, то вивід
повідомлення
```

```
     {
          snprintf(TempBuf, 59, "помилка читання файлу %s",
file_name);
          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
          return;
     }

    res = f_close(&msgFile);  //закриваємо файл
    if(res != FR_OK && show_error == 1) //якщо помилка, то вивід
повідомлення
     {
          snprintf(TempBuf, 59, "помилка закриття файлу %s",
file_name);
          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
     }
}

// Читання історичних даних з SD-карти  та друк графіків
// show_error == 0 - не показувати помилки, show_error == 1 -
показувати помилки,
void read_history(char *file_name, int8_t show_error)
{
     FRESULT res;
     FIL msgFile;
    res = f_open(&msgFile, file_name, FA_READ);   //відкриваємо файл
з доступом для читання
    if(res != FR_OK && show_error == 1) //якщо помилка, то вивід
повідомлення
     {
          my_strcpy(TempBuf, "помилка відкриття/створення файлу ");
          my_strcat (TempBuf, file_name);

          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
          return;
     }

    unsigned int bytesRead;
    res = f_read(&msgFile, data_history, sizeof(DATA_HISTORY),
&bytesRead);   //читаємо дані з файлу в структуру data_history
    if(res != FR_OK && show_error == 1) //якщо помилка, то вивід
повідомлення
     {
          snprintf(TempBuf, 59, "помилка читання файлу %s",
file_name);
```

```
          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
          return;
     }

    res = f_close(&msgFile);  //закриваємо файл
    if(res != FR_OK && show_error == 1) //якщо помилка, то вивід
повідомлення
     {
          snprintf(TempBuf, 59, "помилка закриття файлу %s",
file_name);
          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
     }

     LCD_Fill(lcd, fm->color_scheme->bg_color);
     LCD_FillWindow(lcd, 0, 0, 319, 9, fm->color_scheme-
>cursor_color);
     LCD_WriteString(lcd, 100, 0, file_name, &Font_8x13, fm-
>color_scheme->text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);

//конвертуємо дані з структури data_history (в в сотих градуса, грама
на м*3 чи відсотка)
//в структуру chart_history (відносні значення координати Y на
графіках: в діапазон від 0 до 99)
     int i;

     for(i = 0; i < 120; i++)
     {
// Температура в шафі, діапазон -15 . . . +35 розтягуємо на 100 точок
          if (data_history->tIn[i] <= -1500)       chart_history-
>tInChart[i] = 0;       // Якщо температура нижче -15, то округляємо
до -15
               else  if (data_history->tIn[i] >= 3500)
chart_history->tInChart[i] = 99;   // Якщо температура вище 35, то
округляємо до 35
                    else chart_history->tInChart[i] = (data_history-
>tIn[i] + 1500) / 50;   // діапазон -15...+35 розтягуємо в два рази

          if (data_history->Motor[i] == 1) chart_history->tInChart[i]
|= 0x80; // ознака роботи вентиляторів - старший біт в 1 - колір фона
на графіку змінюється

// Температура ззовні, діапазон -15 . . . +35 розтягуємо на 100 точок
          if (data_history->tOut[i] <= -1500) chart_history-
>tOutChart[i] = 0;           // Якщо температура нижче -15, то
округляємо до -15
```

```
                else  if (data_history->tOut[i] >= 3500)
chart_history->tOutChart[i] = 99; // Якщо температура вище 35, то
округляємо до 35
                    else chart_history->tOutChart[i] = (data_history-
>tOut[i] + 1500) / 50; // діапазон -15...+35 розтягуємо в два рази

          if (data_history->Heat[i] == 1) chart_history->tOutChart[i]
|= 0x80; // ознака роботи обігрівача - старший біт в 1 - колір фона
на графіку змінюється

// Абсолютна вологість в шафі, діапазон від 0 до 20 грам на кубометр,
розтягуємо на 100 точок
          if (data_history->absHIn[i] >= 2000) chart_history-
>absHInChart[i] = 99;
                else chart_history->absHInChart[i] = data_history-
>absHIn[i] / 20;   // діапазон 0...20 розтягуємо в пять раз, в сотих
%, тому ділимо не на 100, а на 20

// Абсолютна вологість ззовні, діапазон від 0 до 20 грам на кубометр,
розтягуємо на 100 точок
          if (data_history->absHOut[i] >= 2000) chart_history-
>absHOutChart[i] = 99;
                else chart_history->absHOutChart[i] = data_history-
>absHOut[i] / 20;   // діапазон 0...20 розтягуємо в пять раз, в сотих
%, тому ділимо не на 100, а на 20

// Відносна вологість в шафі, діапазон від 0 до 100%, розтягуємо на
100 точок
          if (data_history->relHIn[i] <= 0) chart_history-
>relHInChart[i] = 0;
                else  if (data_history->relHIn[i] >= 10000)
chart_history->relHInChart[i] = 99;
                    else chart_history->relHInChart[i] =
     data_history->relHIn[i]/100;       // діапазон 0-100

// Відносна вологість ззовні, діапазон від 0 до 100%, розтягуємо на
100 точок
          if (data_history->relHOut[i] <= 0) chart_history-
>relHOutChart[i] = 0;
                else  if (data_history->relHOut[i] >= 10000)
chart_history->relHOutChart[i] = 99;
                    else chart_history->relHOutChart[i]     =
     data_history->relHOut[i]/100;      // діапазон 0-100

// Різниця вологості - діапазон від -5 до 5 грам на кубометр,
розтягуємо на 100 точок
          if (data_history->dH[i] <= -500) chart_history->dHChart[i]
= 0;           // Якщо різниця менше -5 то округляємо до -5
                else  if (data_history->dH[i] >= 500)  chart_history-
>dHChart[i] = 99; // Якщо різниця більше 5  то округляємо до 5
```

```
                    else chart_history->dHChart[i] = (data_history-
>dH[i] + 500) / 10; // діапазон -5...+5 розтягуємо в 10 раз
     }

// Друк графіків
     // Горизонтальна шкала по часу
     for(i = 0; i <= 120; i = i + 10)
     {
          LCD_DrawPixel(lcd, 4 + i, 124, COLOR_DARKGREY);
          LCD_DrawPixel(lcd, 4 + i, 123, COLOR_DARKGREY);
          LCD_DrawPixel(lcd, 167 + i, 124, COLOR_DARKGREY);
          LCD_DrawPixel(lcd, 167 + i, 123, COLOR_DARKGREY);

          LCD_DrawPixel(lcd, 4 + i, 239, COLOR_DARKGREY);
          LCD_DrawPixel(lcd, 4 + i, 238, COLOR_DARKGREY);
          LCD_DrawPixel(lcd, 167 + i, 239, COLOR_DARKGREY);
          LCD_DrawPixel(lcd, 167 + i, 238, COLOR_DARKGREY);
     }

     uint8_t tmp;

     for(i = 0; i < 120; i++)//i - індекс відповідає координаті x на
графіках
     {
// намалювати фон в залежності від статуса вентиляторів
          if  (chart_history->tInChart[i] >= 0x80)     // Вентилятори
працювали, то фон bg_line1_color - фон парних рядків
          {
               LCD_DrawLineV(lcd, 125 - i, 22, 100, fm->color_scheme-
>bg_line1_color);
               LCD_DrawLineV(lcd, 287 - i, 22, 100, fm->color_scheme-
>bg_line1_color);
               LCD_DrawLineV(lcd, 125 - i, 132, 100, fm-
>color_scheme->bg_line1_color);
               LCD_DrawLineV(lcd, 287 - i, 132, 100, fm-
>color_scheme->bg_line1_color);
          }
          else
               if (chart_history->tOutChart[i] >= 0x80)     //
обігрівач був включений, то фон bg_line2_color - фон парних рядків
               {
                    LCD_DrawLineV(lcd, 125 - i, 22, 100, fm-
>color_scheme->bg_line2_color);
                    LCD_DrawLineV(lcd, 287 - i, 22, 100, fm-
>color_scheme->bg_line2_color);
                    LCD_DrawLineV(lcd, 125 - i, 132, 100, fm-
>color_scheme->bg_line2_color);
                    LCD_DrawLineV(lcd, 287 - i, 132, 100, fm-
>color_scheme->bg_line2_color);
               }
                    else // все вимкнено
```

```
                    {
                         LCD_DrawLineV(lcd, 125 - i, 22, 100, fm-
>color_scheme->bg_color);
                         LCD_DrawLineV(lcd, 287 - i, 22, 100, fm-
>color_scheme->bg_color);
                         LCD_DrawLineV(lcd, 125 - i, 132, 100, fm-
>color_scheme->bg_color);
                         LCD_DrawLineV(lcd, 287 - i, 132, 100, fm-
>color_scheme->bg_color);
                    }

          if (i % 5 == 0) // Пунктирні лінії графіка
          {
//верхні графіки
               LCD_DrawPixel(lcd, 125 - i, 110, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 90, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 70, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 50, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 30, COLOR_DARKGREY);

               LCD_DrawPixel(lcd, 287 - i, 95, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 287 - i, 70, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 287 - i, 45, COLOR_DARKGREY);
//нижні графіки
               LCD_DrawPixel(lcd, 125 - i, 225, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 205, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 185, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 165, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 145, COLOR_DARKGREY);

               LCD_DrawPixel(lcd, 287 - i, 210, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 287 - i, 185, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 287 - i, 160, COLOR_DARKGREY);
          }

     // Вивести нову точку
//верхні графіки
          tmp = chart_history->tInChart[i] & 0x7f;   // Відкинути
старший розряд - ознака увімкнення вентиляторів
          if ((tmp == 0) || (tmp == 100))
               LCD_DrawPixel(lcd, 125 - i, 121 - tmp, fm-
>color_scheme->text_color_file);
          else
               LCD_DrawPixel(lcd, 125 - i, 121 - tmp, COLOR_RED);

          tmp = chart_history->tOutChart[i] & 0x7f;   // Відкинути
старший розряд - ознака увімкнення обігрівача
          if ((tmp ==0) || (tmp == 100))
               LCD_DrawPixel(lcd, 125 - i, 121 - tmp, fm-
>color_scheme->text_color_file);
          else
```

```
                LCD_DrawPixel(lcd, 125 - i, 121 - tmp, fm-
>color_scheme->text_color_cursor);

          if ((chart_history->relHInChart[i] == 0) || (chart_history-
>relHInChart[i] == 100))
                LCD_DrawPixel(lcd, 287 - i, 121 - chart_history-
>relHInChart[i], fm->color_scheme->text_color_file);
          else
                LCD_DrawPixel(lcd, 287 - i, 121 - chart_history-
>relHInChart[i], COLOR_RED);

          if ((chart_history->relHOutChart[i] == 0) ||
(chart_history->relHOutChart[i] == 100))
                LCD_DrawPixel(lcd, 287 - i, 121 - chart_history-
>relHOutChart[i], fm->color_scheme->text_color_file);
          else
                LCD_DrawPixel(lcd, 287 - i, 121 - chart_history-
>relHOutChart[i], fm->color_scheme->text_color_cursor);

//нижні графіки

          if ((chart_history->dHChart[i] == 0) || (chart_history-
>dHChart[i] == 100))
                LCD_DrawPixel(lcd, 125 - i, 236 - chart_history-
>dHChart[i], fm->color_scheme->text_color_file);
          else
                LCD_DrawPixel(lcd, 125 - i, 236 - chart_history-
>dHChart[i], fm->color_scheme->text_color_file);

          if (chart_history->absHInChart[i] == 100)
                LCD_DrawPixel(lcd, 287 - i, 236 - chart_history-
>absHInChart[i], fm->color_scheme->text_color_file);
          else
                LCD_DrawPixel(lcd, 287 - i, 236 - chart_history-
>absHInChart[i], COLOR_RED);

          if (chart_history->absHOutChart[i] == 100)
                LCD_DrawPixel(lcd, 287 - i, 236 - chart_history-
>absHOutChart[i], fm->color_scheme->text_color_file);
          else
                LCD_DrawPixel(lcd, 287 - i, 236 - chart_history-
>absHOutChart[i], fm->color_scheme->text_color_cursor);
     }

//написи на графіках

//верхні графіки
     LCD_WriteString(lcd, 10, 10, utf8_to_win1251("Температура,°C",
OutputBuf), &Font_8x13, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
```

```
     LCD_WriteString(lcd, 135, 25, "+", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 141, 25, "3", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 147, 25, "0", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

     LCD_WriteString(lcd, 135, 45, "+", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 141, 45, "2", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 147, 45, "0", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

     LCD_WriteString(lcd, 135, 65, "+", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 141, 65, "1", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 147, 65, "0", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

     LCD_WriteString(lcd, 135, 85, "0", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 135, 105, "-", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 141, 105, "1", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 147, 105, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);


     LCD_WriteString(lcd, 180, 10, utf8_to_win1251("Відн.волог,%",
OutputBuf), &Font_8x13, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 290, 35, "7", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
```

```
     LCD_WriteString(lcd, 296, 35, "5", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

     LCD_WriteString(lcd, 290, 65, "5", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 296, 65, "0", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

     LCD_WriteString(lcd, 290, 95, "2", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 296, 95, "5", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

//нижні графіки
     LCD_WriteString(lcd, 1, 125, utf8_to_win1251("різниця dH, г/м3",
OutputBuf), &Font_8x13, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 135, 140, "+4", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 135, 160, "+2", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 135, 180, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 135, 200, "-2", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 135, 220, "-4", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 165, 125, utf8_to_win1251("Абс.волог,г/м3",
OutputBuf), &Font_8x13, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 296, 150, "1", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 302, 150, "5", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
```

```
     LCD_WriteString(lcd, 296, 180, "1", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     LCD_WriteString(lcd, 302, 180, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

     LCD_WriteString(lcd, 296, 210, "5", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
}

void show_time(void)     //відображення поточного часу
{
     LCD_FillWindow(lcd, 207, 0, 319, 19, fm->color_scheme-
>cursor_color);
     LCD_WriteString(lcd, 207, 3, TimeBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
}

//отримання дати, часу і формування масиву TimeBuf для виводу
void get_time(int8_t F_update)//Якщо F_update=0, то оновлення TimeBuf
тільки при зміні хвилини
{
//sTime.Hours, sTime.Minutes
//sDate.Date, sDate.Month, sDate.Year

     HAL_RTC_GetTime(&hrtc, &sTime, RTC_FORMAT_BIN);        //
отримуємо час у форматі RTC_FORMAT_BIN
     HAL_RTC_GetDate(&hrtc, &sDate, RTC_FORMAT_BIN);        //
отримуємо дату у форматі RTC_FORMAT_BIN

     if (last_min != sTime.Minutes || F_update == 1)  // виводимо
тільки зміну - тобто, раз в хвилину. Або якщо встановлено ознаку
примусового оновлення F_update == 1
     {
          last_min = sTime.Minutes;
          // формуємо строку для виводу часу і дати
          last_min = sTime.Hours;
          if (last_min < 10) TimeBuf[0] = 48; else TimeBuf[0] =
last_min / 10 + 48;   // код "0" 48
          TimeBuf[1] = last_min % 10 + 48;
          TimeBuf[2] = 58; // ":"
          last_min = sTime.Minutes;
          if (last_min < 10) TimeBuf[3] = 48; else TimeBuf[3] =
last_min / 10 + 48;
          TimeBuf[4] = last_min % 10 + 48;
          if (last_day != sDate.Date || F_update == 1)
          {
               TimeBuf[5] = 32;// " "
```

```
                last_day = sDate.Date;
                if (last_day < 10) TimeBuf[6] = 48; else TimeBuf[6] =
last_day / 10 + 48;
                TimeBuf[7] = last_day % 10 + 48;
                TimeBuf[8] = 46;// "."

                uint8_t temp;
                temp = sDate.Month;
                if (temp < 10) TimeBuf[9] = 48; else TimeBuf[9] = temp
/ 10 + 48;
                TimeBuf[10] = temp % 10 + 48;
                TimeBuf[11] = 46;// "."

                temp = sDate.Year;
                if (temp < 10) TimeBuf[12] = 48; else TimeBuf[12] =
temp / 10 + 48;
                TimeBuf[13] = temp % 10 + 48;

                TimeBuf[14] = 0; // Кінець строки
          }

          packet_ready_Time = 1;   // поступили нові данні
          packet_ready_E0 = 1;
     }
}

void SetKorDat(void) // збереження заданих параметрів для приведення
показів датчиків до одного рівня (коригування)
{
     switch (Screens[11]->Get_Poz(Screens[11]))
     {
          case  0:  //Вийти без змін
                break;
          case  1:  //Внести результати вимірювань вологості до Точки
2
                settingRAM->H2 = (int8_t)((ave_Hsrd->Mean(ave_Hsrd) /
100));
                settingRAM->d_H2 = ave_dH->Mean(ave_dH);
                writeSD();     // Запис на SD-карту
                break;
          case  2:  //Внести результати вимірювань вологості до Точки
1
                settingRAM->H1 = (int8_t)((ave_Hsrd->Mean(ave_Hsrd) /
100));
                settingRAM->d_H1 = ave_dH->Mean(ave_dH);
                writeSD();     // Запис на SD-карту
                break;
          case  3:  //Внести результати вимірювань температури до
Точки 2
```

```
                settingRAM->T2 = (int8_t)((ave_Tsrd->Mean(ave_Tsrd) /
100));
                settingRAM->d_T2 = ave_dT->Mean(ave_dT);
                writeSD();     // Запис на SD-карту
                break;
          case  4:  //Внести результати вимірювань температури до
Точки 1
                settingRAM->T1 = (int8_t)((ave_Tsrd->Mean(ave_Tsrd) /
100));
                settingRAM->d_T1 = ave_dT->Mean(ave_dT);
                writeSD();     // Запис на SD-карту
                break;
     }

}

void SetTime(void) // встановлення заданих часу, дати
{
     LCD_Fill(lcd, fm->color_scheme->bg_color);
     sTime.Hours = Screens[2]->Get_Poz(Screens[2]);
     sTime.Minutes = Screens[3]->Get_Poz(Screens[3]);
     sTime.Seconds = 0;
     sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
     sTime.StoreOperation = RTC_STOREOPERATION_RESET;

     if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BIN) == HAL_OK)
          LCD_WriteString(lcd, 10, 30, utf8_to_win1251("Час успішно
змінено!", OutputBuf), &Font_15x25, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     else
          LCD_WriteString(lcd, 10, 30, utf8_to_win1251("Час не
вдалося змінити!", OutputBuf), &Font_15x25, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);

     sDate.WeekDay = Screens[7]->Get_Poz(Screens[7]);
     sDate.Month = Screens[5]->Get_Poz(Screens[5]);
     sDate.Date = Screens[4]->Get_Poz(Screens[4]);
     sDate.Year = Screens[6]->Get_Poz(Screens[6]);

     if (HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BIN) == HAL_OK)
     {
          LCD_WriteString(lcd, 10, 60, utf8_to_win1251("Дату успішно
змінено!", OutputBuf), &Font_15x25, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
          writeSD();     // Запис на SD-карту
     }
     else
          LCD_WriteString(lcd, 10, 60, utf8_to_win1251("Дату не
вдалося змінити!", OutputBuf), &Font_15x25, fm->color_scheme-
```

```
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);

     HAL_Delay(1000);

     get_time(1);   // отримання актуальних дати, часу і формування
масиву TimeBuf для виводу
}

void resetTimeDate(void) //скидання дат мін/максимальних статистичних
значень
{
     uint8_t   var;

     for(int i=0;i<5;i++)
     {
          switch (i)
          {
               case  0:                              //minute
               case  1:  var = 0;  break;    //hour
               case  2:                              //day
               case  3:  var = 1;  break;    //month
               case  4:  var = 24; break;    //year
          }

          settingRAM->tOutMinTimeDate[i] = var;
          settingRAM->tOutMaxTimeDate[i] = var;
          settingRAM->tInMinTimeDate[i] = var;
          settingRAM->tInMaxTimeDate[i] = var;
          settingRAM->hMaxTimeDate[i] = var;
          settingRAM->hMinTimeDate[i] = var;

          settingRAM->SaveTimeDate[i] = var;
     }
     packet_ready_E1 = 1;
}

// Запис на SD-карту ----------------------------------------------
--
void writeSD(void)
{
     // оновлення часу останнього збереження
     settingRAM->SaveTimeDate[0] = sTime.Minutes;
     settingRAM->SaveTimeDate[1] = sTime.Hours;
     settingRAM->SaveTimeDate[2] = sDate.Date;
     settingRAM->SaveTimeDate[3] = sDate.Month;
     settingRAM->SaveTimeDate[4] = sDate.Year;
     packet_ready_E1 = 1;

     settingRAM->tick_SDflash = 0;
```

```
     write_setting_SD(); // запис на SD структури settingRAM
     write_chart_SD();   // запис на SD структури chart_dry
}

// запис на SD структури settingRAM
void write_setting_SD(void)
{
     settingRAM->N_Screen = Screens[0]->Get_Poz(Screens[0]);     //
зберігаємо поточний інфо екран

     FRESULT res;
     FIL logFile;

//   FA_OPEN_ALWAYS - Створює новий файл. Якщо файл існує, він
скорочується та перезаписується
//   FA_WRITE - Визначає доступ для запису до об'єкта. Дані можна
записати у файл
    res = f_open(&logFile, "settingRAM.dat", FA_OPEN_ALWAYS |
FA_WRITE);     //відкриваємо/створюємо файл з доступом для запису
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
     {
//        snprintf(TempBuf, 39, "помилка відкриття файлу
settingRAM.dat");
          my_strcpy(TempBuf, "помилка відкриття файлу
settingRAM.dat");

          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
          return;
    }

    unsigned int bytesToWrite = sizeof(DataForSave);   //визначаємо
необхідний розмір в байтах для запису

    unsigned int bytesWritten;
    res = f_write(&logFile, settingRAM, bytesToWrite, &bytesWritten);
     //пишемо дані у файл
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
     {
          snprintf(TempBuf, 59, "помилка запису у файл
settingRAM.dat");
          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
        return;
    }

    if(bytesWritten < bytesToWrite)     //перевіряємо чи вистачило
місця
     {
```

```
          snprintf(TempBuf, 59, "помилка: недостатньо місця на SD");
          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
     }

     res = f_close(&logFile);  //закриваємо файл
     if(res != FR_OK)     //якщо помилка, то вивід повідомлення
      {
          snprintf(TempBuf, 59, "помилка закриття файлу
settingRAM.dat");
          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
     }
}

// запис на SD структури chart_dry
void write_chart_SD(void)
{
     FRESULT res;
     FIL logFile;

//   FA_OPEN_ALWAYS - Створює новий файл. Якщо файл існує, він
скорочується та перезаписується
//   FA_WRITE - Визначає доступ для запису до об'єкта. Дані можна
записати у файл
     res = f_open(&logFile, "chart_dry.dat", FA_OPEN_ALWAYS |
FA_WRITE);     //відкриваємо/створюємо файл
     if(res != FR_OK)     //якщо помилка, то вивід повідомлення
      {
          snprintf(TempBuf, 59, "помилка відкриття файлу
chart_dry.dat");
          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
          return;
     }

     unsigned int bytesToWrite = sizeof(CHART_DRY);     //визначаємо
розмір в байтах для запису

     unsigned int bytesWritten;
     res = f_write(&logFile, chart_dry, bytesToWrite, &bytesWritten);
      //пишемо дані у файл
     if(res != FR_OK)     //якщо помилка, то вивід повідомлення
      {
          snprintf(TempBuf, 59, "помилка запису у файл
chart_dry.dat");
```

```
          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
        return;
     }

    if(bytesWritten < bytesToWrite)     //перевіряємо чи вистачило
місця
     {
          snprintf(TempBuf, 59, "помилка: недостатньо місця на SD");
          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
     }

    res = f_close(&logFile);  //закриваємо файл
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
     {
          snprintf(TempBuf, 59, "помилка закриття файлу
chart_dry.dat");
          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
     }
}

// Читання даних з SD-карти ------------------------------------------
---------
void readSD(void)
{
     read_setting_SD();  // Читання з SD-карти структури settingRAM
     read_chart_SD();    // Читання з SD-карти структури chart_dry
}

// Читання з SD-карти структури settingRAM ---------------------------
-----------------------
void read_setting_SD(void)
{
    FRESULT res;

     FIL msgFile;
    res = f_open(&msgFile, "settingRAM.dat", FA_READ); //відкриваємо
файл з доступом для читання
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
     {
//        snprintf(TempBuf, 39, "помилка відкриття файлу
settingRAM.dat");
          my_strcpy(TempBuf, "помилка відкриття файлу
settingRAM.dat");
```

```
          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
        return;
      }

    unsigned int bytesRead;
    res = f_read(&msgFile, settingRAM, sizeof(DataForSave),
&bytesRead);   //читаємо дані з файлу
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
     {
//        snprintf(TempBuf, 39, "помилка читання з файлу
settingRAM.dat");
          my_strcpy(TempBuf, "помилка читання з файлу
settingRAM.dat");

          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
          return;
      }

    res = f_close(&msgFile);  //закриваємо файл
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
     {
//        snprintf(TempBuf, 39, "помилка закриття файлу
settingRAM.dat");
          my_strcpy(TempBuf, "помилка закриття файлу
settingRAM.dat");

          LCD_WriteString(lcd, 0, 200, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
      }

     if ((settingRAM->mode > (NUM_SETTING-1))||(settingRAM->mode <
0)) settingRAM->mode = 0; // страхування щоб не вийти з діапазону
допустимих режимів
}

// Читання з SD-карти структури chart_dry ---------------------------
----------------------
void read_chart_SD(void)
{
    FRESULT res;

     FIL msgFile;
    res = f_open(&msgFile, "chart_dry.dat", FA_READ);  //відкриваємо
файл з доступом для читання
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
     {
```

```
//        snprintf(TempBuf, 39, "помилка відкриття файлу
chart_dry.dat");
          my_strcpy(TempBuf, "помилка відкриття файлу
chart_dry.dat");

          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
          return;
     }

    unsigned int bytesRead;
    res = f_read(&msgFile, chart_dry, sizeof(CHART_DRY), &bytesRead);
     //читаємо дані з файлу
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
     {
//        snprintf(TempBuf, 39, "помилка читанн з файлу
chart_dry.dat");
          my_strcpy(TempBuf, "помилка читанн з файлу chart_dry.dat");

          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
          return;
     }

    res = f_close(&msgFile);  //закриваємо файл
    if(res != FR_OK)     //якщо помилка, то вивід повідомлення
      {
//        snprintf(TempBuf, 59, "помилка закриття файлу
chart_dry.dat");
          my_strcpy(TempBuf, "помилка закриття файлу chart_dry.dat");

          LCD_WriteString(lcd, 0, 220, utf8_to_win1251(TempBuf,
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);
     }
}

void resetTemp(void)  // скидання min max значень
{
     settingRAM->tOutMin =    5555;
     settingRAM->tInMin  =    5555;
     settingRAM->tOutMax =    -5555;
     settingRAM->tInMax  =    -5555;
     settingRAM->hMax    =    0;
     settingRAM->hMin    =    10000;

     resetTimeDate();

     writeSD();  //Скинули температури зберігаємо на SD-карту
```

```
}

// IsP - визначає які графіки друкувати
void dry_static(int8_t IsP)  // Друк статичної картинки на інфо
екранах 2 і 3
{
     int i;
// Таблиця
     LCD_DrawLineH(lcd, 0, 20, 319, fm->color_scheme->slider_color);
     LCD_DrawLineH(lcd, 0, 40, 319, fm->color_scheme->slider_color);
     LCD_DrawLineH(lcd, 0, 60, 319, fm->color_scheme->slider_color);
     LCD_DrawLineH(lcd, 0, 80, 319, fm->color_scheme->slider_color);
     LCD_DrawLineH(lcd, 0, 100, 319, fm->color_scheme->slider_color);
     LCD_DrawLineH(lcd, 0, 120, 319, fm->color_scheme->slider_color);
     LCD_DrawLineV(lcd, 196,20,100, fm->color_scheme->slider_color);
     LCD_DrawLineV(lcd, 260,20,80, fm->color_scheme->slider_color);

     if(settingRAM->demo_mode == 1)
     {
          LCD_WriteString(lcd, 10, 24,
utf8_to_win1251("демонстраційний режим", OutputBuf), &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
     }

  // Заголовки таблиць
     LCD_WriteString(lcd, 264, 24, utf8_to_win1251("Ззовні",
OutputBuf), &Font_8x13, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 200, 24, utf8_to_win1251("Всеред",
OutputBuf), &Font_8x13, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 1, 44, utf8_to_win1251("Температура, °C",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 1, 64, utf8_to_win1251("Відносна вологість,
%", OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 1, 84, utf8_to_win1251("Абсолют.вологість,
г/м", OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 180, 84, "3", &Font_8x13, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 1, 104, utf8_to_win1251("Різниця абс.волог,
г/м", OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 180, 104, "3", &Font_8x13, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
```

```
// Графіки
// написи на графіках
     if(IsP == 0)
     {
          LCD_WriteString(lcd, 1, 125, utf8_to_win1251("різниця dH,
г/м3", OutputBuf), &Font_8x13, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 135, 140, "+4", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 135, 160, "+2", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 135, 180, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 135, 200, "-2", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 135, 220, "-4", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

          LCD_WriteString(lcd, 165, 125,
utf8_to_win1251("Абс.волог,г/м3", OutputBuf), &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

//для того щоб щільніше були цифри друкуємо по одному символу
//LCD_SYMBOL_PRINT_PSETBYPSET - виводимо без затирання фону
//        LCD_WriteString(lcd, 296, 150, "15", &Font_8x13,
COLOR_WHITE, COLOR_BLACK, LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 296, 150, "1", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 302, 150, "5", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

//        LCD_WriteString(lcd, 296, 180, "10", &Font_8x13,
COLOR_WHITE, COLOR_BLACK, LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 296, 180, "1", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 302, 180, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

          LCD_WriteString(lcd, 296, 210, "5", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
```

```
     }
     else
     {
          LCD_WriteString(lcd, 10, 125,
utf8_to_win1251("Температура,°C", OutputBuf), &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
//        LCD_WriteString(lcd, 135, 140, "+30", &Font_8x13,
COLOR_WHITE, COLOR_BLACK, LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 135, 140, "+", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 141, 140, "3", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 147, 140, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

//        LCD_WriteString(lcd, 135, 160, "+20", &Font_8x13,
COLOR_WHITE, COLOR_BLACK, LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 135, 160, "+", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 141, 160, "2", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 147, 160, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

//        LCD_WriteString(lcd, 135, 180, "+10", &Font_8x13,
COLOR_WHITE, COLOR_BLACK, LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 135, 180, "+", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 141, 180, "1", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 147, 180, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

          LCD_WriteString(lcd, 135, 200, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

//        LCD_WriteString(lcd, 135, 220, "-10", &Font_8x13,
COLOR_WHITE, COLOR_BLACK, LCD_SYMBOL_PRINT_FAST);
```

```
          LCD_WriteString(lcd, 135, 220, "-", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 141, 220, "1", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 147, 220, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);


          LCD_WriteString(lcd, 180, 125,
utf8_to_win1251("Відн.волог,%", OutputBuf), &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
//        LCD_WriteString(lcd, 290, 150, "75", &Font_8x13,
COLOR_WHITE, COLOR_BLACK, LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 290, 150, "7", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 296, 150, "5", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

//        LCD_WriteString(lcd, 290, 180, "50", &Font_8x13,
COLOR_WHITE, COLOR_BLACK, LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 290, 180, "5", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 296, 180, "0", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);

//        LCD_WriteString(lcd, 290, 210, "25", &Font_8x13,
COLOR_WHITE, COLOR_BLACK, LCD_SYMBOL_PRINT_FAST);
          LCD_WriteString(lcd, 290, 210, "2", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
          LCD_WriteString(lcd, 296, 210, "5", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_PSETBYPSET);
     }

// Горизонтальна шкала по часу
     for(i = 0; i <= 120; i = i + 10)
     {
          LCD_DrawPixel(lcd, 4 + i, 239, COLOR_DARKGREY);
          LCD_DrawPixel(lcd, 4 + i, 238, COLOR_DARKGREY);
          LCD_DrawPixel(lcd, 167 + i, 239, COLOR_DARKGREY);
          LCD_DrawPixel(lcd, 167 + i, 238, COLOR_DARKGREY);
     }
```

```
}

void printChart(int8_t IsP) // Друк графіка на інфо екранах 2 і 3,
додається одна точка и графік зсувається
{
     int8_t i, x = 0;
     uint8_t tmp;
     for(i = 0; i < 120; i++)    // Графік зліва на право
     {
// Вирахувати координати поточної точки x в кільцевому буфері.
Змінюється від 0 до 120-1
     if (chart_dry->posChart < i) x = 120 + chart_dry->posChart - i;
else x = chart_dry->posChart - i;

// намалювати фон в залежності від статуса вентиляторів
          if  (chart_dry->tInChart[x] >= 0x80)    // Вентилятори
працювали, то фон bg_line1_color - фон парних рядків
          {
               LCD_DrawLineV(lcd, 125 - i, 137, 100, fm-
>color_scheme->bg_line1_color);
               LCD_DrawLineV(lcd, 287 - i, 137, 100, fm-
>color_scheme->bg_line1_color);
          }
          else
               if (chart_dry->tOutChart[x] >= 0x80)    // обігрівач
був включений, то фон оливковий
               {
                    LCD_DrawLineV(lcd, 125 - i, 137, 100, fm-
>color_scheme->bg_line2_color);
                    LCD_DrawLineV(lcd, 287 - i, 137, 100, fm-
>color_scheme->bg_line2_color);
               }
                    else // все вимкнено
                    {
                         LCD_DrawLineV(lcd, 125 - i, 137, 100, fm-
>color_scheme->bg_color);
                         LCD_DrawLineV(lcd, 287 - i, 137, 100, fm-
>color_scheme->bg_color);
                    }

          if (i % 5 == 0) // Пунктирні лінії графіка
          {
               LCD_DrawPixel(lcd, 125 - i, 225, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 205, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 185, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 165, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 125 - i, 145, COLOR_DARKGREY);

               LCD_DrawPixel(lcd, 287 - i, 210, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 287 - i, 185, COLOR_DARKGREY);
               LCD_DrawPixel(lcd, 287 - i, 160, COLOR_DARKGREY);
```

```
          }

     // Вивести нову точку
          if(IsP == 0)
          {
               if ((chart_dry->dHChart[x] == 0) || (chart_dry-
>dHChart[x] == 100))
                    LCD_DrawPixel(lcd, 125 - i, 236 - chart_dry-
>dHChart[x], fm->color_scheme->text_color_file);
               else
                    LCD_DrawPixel(lcd, 125 - i, 236 - chart_dry-
>dHChart[x], fm->color_scheme->text_color_file);

               if (chart_dry->absHInChart[x] == 100)
                    LCD_DrawPixel(lcd, 287 - i, 236 - chart_dry-
>absHInChart[x], fm->color_scheme->text_color_file);
               else
                    LCD_DrawPixel(lcd, 287 - i, 236 - chart_dry-
>absHInChart[x], COLOR_RED);

               if (chart_dry->absHOutChart[x] == 100)
                    LCD_DrawPixel(lcd, 287 - i, 236 - chart_dry-
>absHOutChart[x], fm->color_scheme->text_color_file);
               else
                    LCD_DrawPixel(lcd, 287 - i, 236 - chart_dry-
>absHOutChart[x], fm->color_scheme->text_color_cursor);
          }
          else
          {
               tmp = chart_dry->tInChart[x] & 0x7f;   // Відкинути
старший розряд - ознака увімкнення вентиляторів
               if ((tmp == 0) || (tmp == 100))
                    LCD_DrawPixel(lcd, 125 - i, 236 - tmp, fm-
>color_scheme->text_color_file);
               else
                    LCD_DrawPixel(lcd, 125 - i, 236 - tmp,
COLOR_RED);

               tmp = chart_dry->tOutChart[x] & 0x7f;   // Відкинути
старший розряд - ознака увімкнення обігрівача
               if ((tmp ==0) || (tmp == 100))
                    LCD_DrawPixel(lcd, 125 - i, 236 - tmp, fm-
>color_scheme->text_color_file);
               else
                    LCD_DrawPixel(lcd, 125 - i, 236 - tmp, fm-
>color_scheme->text_color_cursor);

               if ((chart_dry->relHInChart[x] == 0) || (chart_dry-
>relHInChart[x] == 100))
                    LCD_DrawPixel(lcd, 287 - i,236 - chart_dry-
>relHInChart[x], fm->color_scheme->text_color_file);
```

```
                else
                     LCD_DrawPixel(lcd, 287 - i,236 - chart_dry-
>relHInChart[x], COLOR_RED);

                if ((chart_dry->relHOutChart[x] == 0) || (chart_dry-
>relHOutChart[x] == 100))
                     LCD_DrawPixel(lcd, 287 - i,236 - chart_dry-
>relHOutChart[x], fm->color_scheme->text_color_file);
                else
                     LCD_DrawPixel(lcd, 287 - i,236 - chart_dry-
>relHOutChart[x], fm->color_scheme->text_color_cursor);
           }
     }
}

void print_status0(void) // Друк значень на інфо екранах 2 і 3
{
     sprintf(TempBuf, "%.1f", ave_tIn->Mean(ave_tIn) / 100.0);
     LCD_WriteString(lcd, 215, 44, TempBuf, &Font_12x20, COLOR_RED,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%.1f", ave_relHIn->Mean(ave_relHIn) / 100.0);
     LCD_WriteString(lcd, 215, 64, TempBuf, &Font_12x20, COLOR_RED,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%.1f", ave_absHIn->Mean(ave_absHIn) / 100.0);
     LCD_WriteString(lcd, 215, 84, TempBuf, &Font_12x20, COLOR_RED,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%.1f", ave_tOut->Mean(ave_tOut) / 100.0);
     LCD_WriteString(lcd, 275, 44, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%.1f", ave_relHOut->Mean(ave_relHOut) /
100.0);
     LCD_WriteString(lcd, 275, 64, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%.1f", ave_absHOut->Mean(ave_absHOut) /
100.0);
     LCD_WriteString(lcd, 275, 84, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%.1f\n", (ave_absHIn->Mean(ave_absHIn) -
ave_absHOut->Mean(ave_absHOut))/100.0 );
     LCD_WriteString(lcd, 245, 104, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
```

```
}

void dry_get_data(void)// заповнення масиву графіка
{
//  memcpy(&packet,OutputBuf,sizeof(type_packet_NRF24));  //
скопировать из буфера в структуру
   // Малюємо графіки в пам'яті, а коли потрібно то швидко виводимо
  chart_dry->TimeChart++;
     if ((int16_t)((int16_t)chart_dry-
>TimeChart*TIME_SCAN_SENSOR)>=(int16_t)TIME_PRINT_CHART) // перевірка
чи не пора добавляти точку на графік
     {
          save_history(); //додаємо інформацію в data_today і
зберігаємо на SD-карту

          chart_dry->CoolData =    1;  // Отримано нові дані
     // Працюємо через кольцевий буфер
     // Додати нову точку в кольцевий буфер
// Температура в шафі, діапазон -15 . . . +35 розтягуємо на 100 точок
          if (ave_tIn->Mean(ave_tIn) <= -1500)       chart_dry-
>tInChart[chart_dry->posChart] = 0;       // Якщо температура нижче -
15, то округляємо до -15
               else  if (ave_tIn->Mean(ave_tIn) >= 3500)  chart_dry-
>tInChart[chart_dry->posChart] = 99;   // Якщо температура вище 35,
то округляємо до 35
                    else chart_dry->tInChart[chart_dry->posChart] =
(ave_tIn->Mean(ave_tIn) + 1500) / 50;   // діапазон -15...+35
розтягуємо в два рази

          if (chart_dry->ChartMotor == 1) chart_dry-
>tInChart[chart_dry->posChart] |= 0x80; // ознака роботи вентиляторів
- старший біт в 1 - колір фона на графіку змінюється
          chart_dry->ChartMotor = 0;

// Температура ззовні, діапазон -15 . . . +35 розтягуємо на 100 точок
          if (ave_tOut->Mean(ave_tOut) <= -1500) chart_dry-
>tOutChart[chart_dry->posChart] = 0;           // Якщо температура
нижче -15, то округляємо до -15
               else  if (ave_tOut->Mean(ave_tOut) >= 3500)
chart_dry->tOutChart[chart_dry->posChart] = 99; // Якщо температура
вище 35, то округляємо до 35
                    else chart_dry->tOutChart[chart_dry->posChart] =
(ave_tOut->Mean(ave_tOut) + 1500) / 50; // діапазон -15...+35
розтягуємо в два рази

          if (chart_dry->ChartHeat == 1) chart_dry-
>tOutChart[chart_dry->posChart] |= 0x80; // ознака роботи обігрівача
- старший біт в 1 - колір фона на графіку змінюється
          chart_dry->ChartHeat = 0;
```

```
// Абсолютна вологість в шафі, діапазон від 0 до 20 грам на кубометр, розтягуємо на 100 точок
          if (ave_absHIn->Mean(ave_absHIn) >= 2000) chart_dry->absHInChart[chart_dry->posChart] = 99;
                else chart_dry->absHInChart[chart_dry->posChart] = ave_absHIn->Mean(ave_absHIn) / 20;   // діапазон 0...20 розтягуємо в пять раз, в сотих %, тому ділимо не на 100, а на 20

// Абсолютна вологість ззовні, діапазон від 0 до 20 грам на кубометр, розтягуємо на 100 точок
          if (ave_absHOut->Mean(ave_absHOut) >= 2000) chart_dry->absHOutChart[chart_dry->posChart] = 99;
                else chart_dry->absHOutChart[chart_dry->posChart] = ave_absHOut->Mean(ave_absHOut) / 20;   // діапазон 0...20 розтягуємо в пять раз, в сотих %, тому ділимо не на 100, а на 20

// Відносна вологість в шафі, діапазон від 0 до 100%, розтягуємо на 100 точок
          if (ave_relHIn->Mean(ave_relHIn) <= 0) chart_dry->relHInChart[chart_dry->posChart] = 0;
                else  if (ave_relHIn->Mean(ave_relHIn) >= 10000) chart_dry->relHInChart[chart_dry->posChart] = 99;
                     else chart_dry->relHInChart[chart_dry->posChart]
     =    ave_relHIn->Mean(ave_relHIn)/100;       // діапазон 0-100

// Відносна вологість ззовні, діапазон від 0 до 100%, розтягуємо на 100 точок
          if (ave_relHOut->Mean(ave_relHOut) <= 0) chart_dry->relHOutChart[chart_dry->posChart] = 0;
                else  if (ave_relHOut->Mean(ave_relHOut) >= 10000) chart_dry->relHOutChart[chart_dry->posChart] = 99;
                     else chart_dry->relHOutChart[chart_dry->posChart]
     =    ave_relHOut->Mean(ave_relHOut)/100;          // діапазон 0-100

// Різниця вологості - діапазон від -5 до 5 грам на кубометр, розтягуємо на 100 точок
     int16_t temp_dH;
     temp_dH = ave_absHIn->Mean(ave_absHIn) - ave_absHOut->Mean(ave_absHOut);

     if (temp_dH <= -500) chart_dry->dHChart[chart_dry->posChart] = 0;           // Якщо різниця менше -5 то округляємо до -5
          else  if (temp_dH >= 500)  chart_dry->dHChart[chart_dry->posChart] = 99; // Якщо різниця більше 5  то округляємо до 5
                else chart_dry->dHChart[chart_dry->posChart] = (temp_dH + 500) / 10; // діапазон -5...+5 розтягуємо в 10 раз
```

```
          if (chart_dry->posChart < 119) chart_dry->posChart++; else
chart_dry->posChart=0;  // Змінили положення в буфері і замкнули
буфер
          chart_dry->TimeChart = 0;  // Зсув графіка і друк нової
точки, скидання лічильника
     }
}

//перевірка чи не пора вимкнути дисплей
void CheckEkran(void)
{
     if (FLAG_EKRAN_CHECK)
     {
          if ((int16_t)(hour_ekran * TIME_SCAN_SENSOR) >=
(int16_t)TIME_EKRAN)
          {
               EKRAN_OFF;
          }
     }
}

void dry_update(int8_t IsP)
{
     if (chart_dry->CoolData == 1) // Якщо отримано нові дані для
графіків
     {
         if (chart_dry->TimeChart == 0) //і якщо прийшов час
відображення нової точки
          {
               printChart(IsP); // то оновлюємо графіки
          }
         chart_dry->CoolData = 0;
     }
}

// Перевірка статусу системи
void CheckON(void)
{  int tmp = 0;
// 0.  Перевірка переохолодження в шафі, увімкнення обігрівача
     if (~FLAG_HEAT_CHECK)
     {    if (ave_tIn->Mean(ave_tIn) <= TEMP_LOW) // Контроль
заморожування в шафі
          {    if (FLAG_MOTOR_CHECK)
               {    MOTOR_OFF;     } HEAT_ON; return;  }    //
увімкнення обігрівача
     }
     if (FLAG_HEAT_CHECK)//Вимкнути обігрівач коли температура
підніметься на 0,5 градуса від критичної+значення гістерезису (1,1
°C)
       {  if (ave_tIn->Mean(ave_tIn) > TEMP_LOW+dT_OFF + 50)
     HEAT_OFF; }
```

```
// 1. Режими що не залежать від вологості і температури (високий
пріоритет)
     if (settingRAM->mode == BLOCK_OFF) //якщо режим Вимкнено
     {    if (FLAG_MOTOR_CHECK)    {    MOTOR_OFF;     }    return;
     }    // то вимикаємо вентилятори
     if (settingRAM->mode == HOOD_ON )  // якщо режим вентиляції
     {    if (~FLAG_MOTOR_CHECK)   {    MOTOR_ON; }    return;   }
     // то вмикаємо вентилятори
// 2. Режим охолодження (другий пріоритет) температура всередині вище
8 радусів, на вологість не дивимся
     if (settingRAM->mode == COOLING)          // якщо режим
охолодження
       {  if (FLAG_MOTOR_CHECK)                                  //
вентилятори працюють
             { if(ave_tIn->Mean(ave_tIn) <= ave_tOut-
>Mean(ave_tOut))// і температура всередині нижче ніж ззовні
                  MOTOR_OFF;                                          //
то вимикаємо вентилятори
          }
          else                                                 // якщо
вентилятори вимкнено
          {    if(ave_tIn->Mean(ave_tIn) > (packet->T_min * 10.0))
          //температура вище встановленої (8°C)
               {    // dH_min використовується не штатно для
температури для режимі COOLING
                 if((ave_tIn->Mean(ave_tIn) - ave_tOut-
>Mean(ave_tOut)) > packet->dH_min)
                  MOTOR_ON;
     //  то вмикаємо вентилятори
               }
          }    return;                                 // виходимо
     }
// 3. В режимі осушення - перевірка досягнення мінімальної
температури в шафі - ТЕРМІНОВО ВиМКНУТи - третій пріоритет
     if (ave_tIn->Mean(ave_tIn) <= (packet->T_min * 10.0))
     {    if (FLAG_MOTOR_CHECK)    {    MOTOR_OFF;     } return; }
     // вимикаємо вентилятори
//ave_absHIn в сотих долях г/м3, ave_relHIn - в сотих долях %
// 4. Режими, які залежать від температури і вологості - низький
приоритет (що залишилось)
     if (FLAG_MOTOR_CHECK)
          // якщо вентилятори працюють
     {    if((ave_tIn->Mean(ave_tIn) <= (packet->T_min * 10.0)) ||
                  //і температура нижче критичної
          (ave_absHIn->Mean(ave_absHIn) < (ave_absHOut-
>Mean(ave_absHOut) + dH_OFF)))//або абс вологість в шафі нижче ніж
ззовні
               MOTOR_OFF;
          //то вимикаємо вентилятори
     }
```

```
     else
     // якщо вентилятори вимкнено
     {  if(ave_tIn->Mean(ave_tIn)>(packet->T_min * 10.0))
     //температура вище критичної
        { // Розраховуємо ризницю спрацювання по вологості
           if (FLAG_ABS_H_CHECK)   { tmp = packet->dH_min; }//режими
використовують абсолютну вологість - різниця в сотих грама на
метр.куб
          else {tmp=(uint16_t)(ave_absHIn->Mean(ave_absHIn)*packet-
>dH_min/100.0);}//режими вик-ть дес.долі % від абс.різн.темп.в шафі
            if((ave_absHIn->Mean(ave_absHIn) - tmp) > ave_absHOut-
>Mean(ave_absHOut)) //Якщо абс вологість ззовні менше
              { MOTOR_ON; }    //то вмикаємо вентилятори
        }
     }
}

// Вивід інформації про поточний режим і збереження налаштувань на
SD, якщо Flag_Change == 1 ---------------------------------
void Setting(uint8_t Flag_Change)
{ // Налаштування
     LCD_FillWindow(lcd, 0, 0, 180, 19, fm->color_scheme-
>cursor_color);
     switch (settingRAM->mode)
     {
          case  BLOCK_OFF:
               LCD_WriteString(lcd, 15, 0,
utf8_to_win1251("Вимкнено", OutputBuf), &Font_15x25, fm-
>color_scheme->text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
               packet->dH_min = 255;
               packet->T_min = 255;
               FLAG_ABS_H_ON;
               break;
          case  HOOD_ON:
               LCD_WriteString(lcd, 0, 0, utf8_to_win1251("Режим
вент-ції", OutputBuf), &Font_15x25, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
               packet->dH_min = 0;
               packet->T_min = 0;
               FLAG_ABS_H_ON;
               break;
          case  COOLING:
               LCD_WriteString(lcd, 5, 3, utf8_to_win1251("Охолод-я
T>9°C dT>2°C", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
               packet->dH_min = 200;
      // dH_min використовується не штатно для температури
               packet->T_min = 90;
```

```
                FLAG_ABS_H_ON;
                break;
          case  3:
                LCD_WriteString(lcd, 5, 3, utf8_to_win1251("Осушення
T>+9°C dH>0.3", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
                packet->dH_min = 30;
                packet->T_min = 90;
                FLAG_ABS_H_ON;
                break;
          case  4:
                LCD_WriteString(lcd, 5, 3, utf8_to_win1251("Осушення
T>+9°C dH>5%", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
                packet->dH_min = 5;
                packet->T_min = 90;
                FLAG_ABS_H_OFF;
                break;
          case  5:
                LCD_WriteString(lcd, 5, 3, utf8_to_win1251("Осушення
T>+16°C dH>0.6", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
                packet->dH_min = 60;
                packet->T_min = 160;
                FLAG_ABS_H_ON;
                break;
          case  6:
                LCD_WriteString(lcd, 5, 3, utf8_to_win1251("Осушення
T>+16°C dH>10%", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
                packet->dH_min = 10;
                packet->T_min = 160;
                FLAG_ABS_H_OFF;
                break;
          case  7:
                LCD_WriteString(lcd, 5, 3, utf8_to_win1251("Осушення
T>+16°C dH>0.8", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
                packet->dH_min = 80;
                packet->T_min = 160;
                FLAG_ABS_H_ON;
     }

     if(Flag_Change == 1)
     {
          FLAG_MOTOR_IN_OFF;
```

```
          FLAG_MOTOR_OUT_OFF;

          writeSD();       // Збереження на SD нових налаштувань

// Змінили налаштування - перевірка чи потрібно увімкнути вентилятори
          CheckON();           // Можливо необхідно увімкнути
вентилятори

          if(FLAG_MOTOR_CHECK)
          {
               if(FLAG_MOTOR_IN_CHECK)
               {
                    FLAG_MOTOR_IN_OFF;
               }
          }
          else //~FLAG_MOTOR_CHECK
          {
               if(FLAG_MOTOR_OUT_CHECK)
               {
                    FLAG_MOTOR_OUT_OFF;
               }
          }
     }
}

void kvadrat(uint32_t color)  // відображення квадрата вказаним
кольором
{
     LCD_DrawLineH(lcd, 188, 2, 15, color);
     LCD_DrawLineH(lcd, 188, 16, 15, color);
     LCD_DrawLineV(lcd, 188, 2, 15, color);
     LCD_DrawLineV(lcd, 202, 2, 15, color);

     LCD_DrawLineH(lcd, 188, 3, 15, color);
     LCD_DrawLineH(lcd, 188, 15, 15, color);
     LCD_DrawLineV(lcd, 189, 2, 15, color);
     LCD_DrawLineV(lcd, 201, 2, 15, color);
}

void cubik(void)    // зафарбовування квадрата вказаним кольором
{
     if (FLAG_MOTOR_OUT_CHECK || FLAG_MOTOR_IN_CHECK) return;

     LCD_FillWindow(lcd, 181, 0, 206, 19, fm->color_scheme-
>cursor_color);

     if (FLAG_MOTOR_CHECK)    //якщо вентилятори працюють
     {
          LCD_FillWindow(lcd, 188, 2, 202, 15, COLOR_GREEN);//то
квадрат зелений
     }
```

```
     else
     {
          if (FLAG_HEAT_CHECK) LCD_FillWindow(lcd, 188, 2, 202, 15,
COLOR_RED);//якщо обігрівач працює, то квадрат червоний
          else LCD_FillWindow(lcd, 188, 2, 202, 15, fm->color_scheme-
>bg_color);//якщо не працюють ні вентилятори ні обігрівач, то квадрат
має колір фону
     }
}

void printColorMode(void)     //Екран "Колірна схема"
{
     ScreenMode->Set_Poz(ScreenMode, 9);
     LCD_Fill(lcd, fm->color_scheme->bg_color);
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);

     Screens[9]->Set_Poz(Screens[9], settingRAM->color_mode);

     LCD_WriteString(lcd, 30, 0, utf8_to_win1251("Колірна схема",
OutputBuf), &Font_15x25, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 15, dist, utf8_to_win1251("Світла",
OutputBuf), &Font_12x20, COLOR_BLACK, COLOR_WHITE,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 15, dist*2, utf8_to_win1251("Синя",
OutputBuf), &Font_12x20, COLOR_CYAN, COLOR_BLUE,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 15, dist*3, utf8_to_win1251("Зелена",
OutputBuf), &Font_12x20, 0x19389E, COLOR_GREEN,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 15, dist*4, utf8_to_win1251("Темна",
OutputBuf), &Font_12x20, COLOR_GREEN, COLOR_BLACK,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 5, dist*(1+Screens[9]-
>Get_Poz(Screens[9])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
}

void printAirDampers(void)    //Екран "Налаштування повітряних
заслінок"
{
     ScreenMode->Set_Poz(ScreenMode, 12);
     LCD_Fill(lcd, fm->color_scheme->bg_color);
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);
```

```
     LCD_WriteString(lcd, 20, 5, utf8_to_win1251("Налаштування
повітряних заслінок", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 15, dist, utf8_to_win1251("Тип повітряних
заслінок", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 15, dist*2, utf8_to_win1251("Налаштування
серво-приводів", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 5, dist*(1+Screens[12]-
>Get_Poz(Screens[12])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
}

void printServoMode(void)     //Екран "Тип повітряних заслінок"
{
     ScreenMode->Set_Poz(ScreenMode, 13);
     LCD_Fill(lcd, fm->color_scheme->bg_color);
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);

     Screens[13]->Set_Poz(Screens[13], settingRAM->servo_mode);

     LCD_WriteString(lcd, 20, 0, utf8_to_win1251("Тип повітряних
заслінок", OutputBuf), &Font_15x25, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 15, dist, utf8_to_win1251("з серво-
приводом", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 15, dist*2, utf8_to_win1251("з магнітним
механізмом", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 5, dist*(1+Screens[13]-
>Get_Poz(Screens[13])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
}

void printSetServo(void) //Екран "Налаштування серво-приводів"
{
/*
     Screens[12] - значення для повороту сервопривода на 90° -
відкрито
     Screens[13] - значення для повороту сервопривода на 0° - закрито
     Screens[14] - Так/Ні
```

```
*/
     ScreenMode->Set_Poz(ScreenMode, 14);
     LCD_Fill(lcd, fm->color_scheme->bg_color);
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);

     LCD_WriteString(lcd, 45, 5, utf8_to_win1251("Налаштування серво-
приводів", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 50, 115, "^^^", &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 25, 80, utf8_to_win1251("відкрито
закрито", OutputBuf), &Font_12x20, fm->color_scheme->bg_line2_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     Screens[14]->Set_Poz(Screens[14], settingRAM->serv_90_deg);
     Screens[15]->Set_Poz(Screens[15], settingRAM->serv_0_deg);
     Screens[16]->Set_Poz(Screens[16], 0);

     HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_RESET);//увімкнення реле подачи живлення на сервоприводи
}

void printSetServoUpdate(void)     // Оновлення екрану "Налаштування
серво-приводів"
{
     sprintf(TempBuf, "%3d", Screens[14]->Get_Poz(Screens[14]));
     LCD_WriteString(lcd, 50, 100, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%3d", Screens[15]->Get_Poz(Screens[15]));
     LCD_WriteString(lcd, 125, 100, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     switch (Screens[16]->Get_Poz(Screens[16]))
     {
          case  0:
               LCD_WriteString(lcd, 200, 100, utf8_to_win1251("Так",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               break;
          case  1:
               LCD_WriteString(lcd, 200, 100, utf8_to_win1251("Hi ",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     }
```

```
     if(ScreenMode->Get_Poz(ScreenMode) == 14)
     {
          TIM2->CCR1 = Screens[14]->Get_Poz(Screens[14]) * 20 + 3900;
          TIM2->CCR2 = Screens[14]->Get_Poz(Screens[14]) * 20 + 3900;
     }
     else
          if(ScreenMode->Get_Poz(ScreenMode) == 15)
          {
               TIM2->CCR1 = Screens[15]->Get_Poz(Screens[15]) * 20 +
3900;
               TIM2->CCR2 = Screens[15]->Get_Poz(Screens[15]) * 20 +
3900;
          }
}

void printDemoMode(void) //Екран "Демонстраційний режим"
{
     ScreenMode->Set_Poz(ScreenMode, 10);
     LCD_Fill(lcd, fm->color_scheme->bg_color);
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);

     Screens[10]->Set_Poz(Screens[10], settingRAM->demo_mode);

     LCD_WriteString(lcd, 10, 0, utf8_to_win1251("Робочий /
Демонстраційний", OutputBuf), &Font_15x25, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 15, dist, utf8_to_win1251("Робочий режим",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 15, dist*2,
utf8_to_win1251("Демонстраційний режим", OutputBuf), &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 5, dist*(1+Screens[10]-
>Get_Poz(Screens[10])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
}

void printSetTimeStatic(void) //Екран "Налаштування Часу/Дати"
{
     ScreenMode->Set_Poz(ScreenMode, 2);
     LCD_Fill(lcd, fm->color_scheme->bg_color);
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);
```

```
     LCD_WriteString(lcd, 30, 0, utf8_to_win1251("Налаштування
Часу/Дати", OutputBuf), &Font_15x25, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 50, 115, "^^", &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 55, 80, utf8_to_win1251("Час     Дата",
OutputBuf), &Font_12x20, fm->color_scheme->bg_line2_color, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     Screens[3]->Set_Poz(Screens[3], sTime.Minutes);
     Screens[2]->Set_Poz(Screens[2], sTime.Hours);
     Screens[4]->Set_Poz(Screens[4], sDate.Date);
     Screens[5]->Set_Poz(Screens[5], sDate.Month);
     Screens[6]->Set_Poz(Screens[6], sDate.Year);
     Screens[7]->Set_Poz(Screens[7], sDate.WeekDay);
}

void printKorDat(void)   //Екран "Коригування датчиків"
{
     ScreenMode->Set_Poz(ScreenMode, 11);
     LCD_Fill(lcd, fm->color_scheme->bg_color);
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);
     LCD_WriteString(lcd, 40, 0, utf8_to_win1251("Коригування
датчиків", OutputBuf), &Font_15x25, fm->color_scheme-
>text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_DrawLineH(lcd, 0, 20, 319, fm->color_scheme->slider_color);
     LCD_DrawLineH(lcd, 95, 40, 139, fm->color_scheme->slider_color);
     LCD_DrawLineH(lcd, 0, 60, 319, fm->color_scheme->slider_color);
     LCD_DrawLineH(lcd, 0, 80, 319, fm->color_scheme->slider_color);
     LCD_DrawLineH(lcd, 0, 100, 319, fm->color_scheme->slider_color);

     LCD_DrawLineV(lcd, 95,20,80, fm->color_scheme->slider_color);
     LCD_DrawLineV(lcd, 130,60,40, fm->color_scheme->slider_color);
     LCD_DrawLineV(lcd, 165,40,60, fm->color_scheme->slider_color);
     LCD_DrawLineV(lcd, 200,60,40, fm->color_scheme->slider_color);

     LCD_DrawLineV(lcd, 235,20,80, fm->color_scheme->slider_color);

     LCD_DrawLineV(lcd, 277,60,40, fm->color_scheme->slider_color);


     LCD_WriteString(lcd, 1, 64, utf8_to_win1251("Темп,", OutputBuf),
&Font_12x20, COLOR_RED, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
```

```
    LCD_WriteString(lcd, 50, 64, utf8_to_win1251("різн.",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 1, 84, utf8_to_win1251("Волог,",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 50, 84, utf8_to_win1251("різн.",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 117, 24, utf8_to_win1251("Налаштування",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_dir, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 237, 26, utf8_to_win1251("Результати",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_dir, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 237, 42, utf8_to_win1251("вимірювань",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_dir, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 103, 44, utf8_to_win1251("Точка 1",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_dir, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 173, 44, utf8_to_win1251("Точка 2",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_dir, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    sprintf(TempBuf, "%3d", settingRAM->T1);
    LCD_WriteString(lcd, 96, 64, TempBuf, &Font_12x20, COLOR_RED,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    sprintf(TempBuf, "%3d", settingRAM->d_T1);
    LCD_WriteString(lcd, 131, 64, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

    sprintf(TempBuf, "%3d", settingRAM->T2);
    LCD_WriteString(lcd, 166, 64, TempBuf, &Font_12x20, COLOR_RED,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    sprintf(TempBuf, "%3d", settingRAM->d_T2);
    LCD_WriteString(lcd, 201, 64, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);


    sprintf(TempBuf, "%3d", settingRAM->H1);
    LCD_WriteString(lcd, 96, 84, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
```

```
     sprintf(TempBuf, "%3d", settingRAM->d_H1);
     LCD_WriteString(lcd, 131, 84, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%3d", settingRAM->H2);
     LCD_WriteString(lcd, 166, 84, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%3d", settingRAM->d_H2);
     LCD_WriteString(lcd, 201, 84, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);


     LCD_WriteString(lcd, 20, 110,  utf8_to_win1251("Температура
[°C]", OutputBuf), &Font_12x20, COLOR_RED, fm->color_scheme-
>bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 175, 110,  utf8_to_win1251("Вологість [%]",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 100, 125,  utf8_to_win1251("Різниця [соті
долі]", OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);


     LCD_WriteString(lcd, 10, 150,  utf8_to_win1251("Кнопками
вверх/вниз вибрати дію", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 10, 165,  utf8_to_win1251("та натиснути
кнопку Ок", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
}

void KorDatUpdate(void)  // Оновлення екрану "Коригування датчиків"
{
     sprintf(TempBuf, "%3d", ave_dT->Mean(ave_dT));
     LCD_WriteString(lcd, 280, 64, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%3d", ave_dH->Mean(ave_dH));
     LCD_WriteString(lcd, 280, 84, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     sprintf(TempBuf, "%3d", (int8_t)((ave_Tsrd->Mean(ave_Tsrd) /
100)));
     LCD_WriteString(lcd, 245, 64, TempBuf, &Font_12x20, COLOR_RED,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
```

```
     sprintf(TempBuf, "%3d", (int8_t)((ave_Hsrd->Mean(ave_Hsrd) /
100)));
     LCD_WriteString(lcd, 245, 84, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_FillWindow(lcd, 0, 200, 319, 239, fm->color_scheme-
>bg_color);

     switch (Screens[11]->Get_Poz(Screens[11]))
     {
          case  0:
               LCD_WriteString(lcd, 100, 210, utf8_to_win1251("Вийти
без змін", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
               break;
          case  1:
               LCD_WriteString(lcd, 50, 200, utf8_to_win1251("Внести
результати вимірювань", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
               LCD_WriteString(lcd, 80, 215,
utf8_to_win1251("вологості до Точки 2", OutputBuf), &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
               break;
          case  2:
               LCD_WriteString(lcd, 50, 200, utf8_to_win1251("Внести
результати вимірювань", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
               LCD_WriteString(lcd, 80, 215,
utf8_to_win1251("вологості до Точки 1", OutputBuf), &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
               break;
          case  3:
               LCD_WriteString(lcd, 50, 200, utf8_to_win1251("Внести
результати вимірювань", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
               LCD_WriteString(lcd, 80, 215,
utf8_to_win1251("температури до Точки 2", OutputBuf), &Font_12x20,
fm->color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
               break;
          case  4:
               LCD_WriteString(lcd, 50, 200, utf8_to_win1251("Внести
результати вимірювань", OutputBuf), &Font_12x20, fm->color_scheme-
```

```
>text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
                LCD_WriteString(lcd, 80, 215,
utf8_to_win1251("температури до Точки 1", OutputBuf), &Font_12x20,
fm->color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
                break;
     }
}

//форматування часу/дати у формат "12:34 12.12.17" і друк вказаним
кольором по вказаних координатах
void Print_time_date(uint8_t *Dim_TimeDate, int x, int y, uint32_t
color)
{
     uint8_t temp;
     char t[15];
     temp = Dim_TimeDate[1];//hour
     if (temp < 10) t[0]=48; else t[0]=temp/10+48;   // код "0" 48
     t[1] = temp%10+48;
     t[2] = 58; // ":"

     temp = Dim_TimeDate[0];//minute
     if (temp < 10) t[3] = 48; else t[3] = temp/10+48;
     t[4] = temp%10+48;
     t[5] = 32;// " "

     temp = Dim_TimeDate[2];//day
     if (temp < 10) t[6] = 48; else t[6] = temp/10+48;
     t[7] = temp%10+48;
     t[8] = 46;// "."

     temp = Dim_TimeDate[3];//month
     if (temp < 10) t[9] = 48; else t[9] = temp/10+48;
     t[10] = temp%10+48;
     t[11] = 46;// "."

     temp = Dim_TimeDate[4];//year
     if (temp < 10) t[12] = 48; else t[12] = temp/10+48;
     t[13] = temp%10+48;

     t[14] = 0; // Кінець строки

     LCD_WriteString(lcd, x, y, t, &Font_12x20, color, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
}

void printSetTimeUpdate(void) // Оновлення екрану "Налаштування
Часу/Дати"
{
```

```
     settingRAM->SaveTimeDate[0] = Screens[3]-
>Get_Poz(Screens[3]);//- minute
     settingRAM->SaveTimeDate[1] = Screens[2]->Get_Poz(Screens[2]);//
- hour
     settingRAM->SaveTimeDate[2] = Screens[4]->Get_Poz(Screens[4]);//
- dayOfMonth
     settingRAM->SaveTimeDate[3] = Screens[5]->Get_Poz(Screens[5]);//
- month
     settingRAM->SaveTimeDate[4] = Screens[6]->Get_Poz(Screens[6]);//
- year
     packet_ready_E1 = 1;

//   LCD_FillWindow(lcd, 50, 100, 319, 113, COLOR_BLACK);

     Print_time_date(settingRAM->SaveTimeDate, 50, 100, fm-
>color_scheme->text_color_file);

     switch (Screens[7]->Get_Poz(Screens[7]))
     {
          case  1:
               LCD_WriteString(lcd, 170, 100, utf8_to_win1251("Пн",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               break;
          case  2:
               LCD_WriteString(lcd, 170, 100, utf8_to_win1251("Вт",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               break;
          case  3:
               LCD_WriteString(lcd, 170, 100, utf8_to_win1251("Ср",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               break;
          case  4:
               LCD_WriteString(lcd, 170, 100, utf8_to_win1251("Чт",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               break;
          case  5:
               LCD_WriteString(lcd, 170, 100, utf8_to_win1251("Пт",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               break;
          case  6:
               LCD_WriteString(lcd, 170, 100, utf8_to_win1251("Сб",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               break;
          case  7:
```

```
                LCD_WriteString(lcd, 170, 100, utf8_to_win1251("Нд",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     }

     switch (Screens[8]->Get_Poz(Screens[8]))
     {
          case  0:
                LCD_WriteString(lcd, 217, 100, utf8_to_win1251("Так",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
                break;
          case  1:
                LCD_WriteString(lcd, 217, 100, utf8_to_win1251("Hi ",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     }
}

//друк екрану налаштувань з вибором режиму роботи
void printSettingStatic(void)
{
     Screens[1]->Set_Poz(Screens[1], settingRAM->mode);//
встановлюємо в меню режим роботи

     ScreenMode->Set_Poz(ScreenMode, 1) ;
     LCD_Fill(lcd, fm->color_scheme->bg_color);

     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);
     LCD_WriteString(lcd, 40, 0, utf8_to_win1251("НАЛАШТУВАННЯ",
OutputBuf), &Font_15x25, fm->color_scheme->text_color_cursor, fm-
>color_scheme->cursor_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 15, dist, utf8_to_win1251("1. Вимкнено",
OutputBuf), &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 15, dist*2, utf8_to_win1251("2. Режим
вентиляції", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 15, dist*3, utf8_to_win1251("3. Охолодження
T>9°C   dT>2°C", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 15, dist*4, utf8_to_win1251("4. Осушення
T>+9°C dH>0.3 г/м", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 254, dist*4+1, "3", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
```

```
    LCD_WriteString(lcd, 15, dist*5, utf8_to_win1251("5. Осушення
T>+9°C dH>5%", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 15, dist*6, utf8_to_win1251("6. Осушення
T>+16°C dH>0.6 г/м", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 254, dist*6+1, "3", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 15, dist*7, utf8_to_win1251("7. Осушення
T>+16°C dH>10%", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 15, dist*8, utf8_to_win1251("8. Осушення
T>+16°C dH>0.8 г/м", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 254, dist*8+1, "3", &Font_8x13, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 15, dist*9, utf8_to_win1251("Скидання min
max значень", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
    LCD_WriteString(lcd, 15, dist*10, utf8_to_win1251("Налаштування
Часу/Дати", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 15, dist*11, utf8_to_win1251("Файловий
менеджер", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 15, dist*12, utf8_to_win1251("Вибір
колірної схеми", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 15, dist*13,
utf8_to_win1251("Робочий/Демонстраційний режим", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_file, fm->color_scheme-
>bg_color, LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 15, dist*14, utf8_to_win1251("Налаштування
повітряних заслінок", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 15, dist*15, utf8_to_win1251("Коригування
датчиків", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

    LCD_WriteString(lcd, 5, dist*(1+Screens[1]-
>Get_Poz(Screens[1])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
```

```
}

// Друк статичної картинки екрана 0
void printInfoStatic0(void)
{
     LCD_WriteString(lcd, 2, dist*2, utf8_to_win1251("Час роботи
системи:", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 2, dist*5, utf8_to_win1251("Час роботи
вент-рів:", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 2, dist*8, utf8_to_win1251("Час роботи
нагрівача:", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     if(settingRAM->demo_mode == 1)
     {
          LCD_WriteString(lcd, 50, 220,
utf8_to_win1251("демонстраційний режим", OutputBuf), &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
     }
}

// Друк статичної картинки екрана 1
void printInfoStatic1(void)
{
     LCD_WriteString(lcd, 2, 3+dist, utf8_to_win1251("1. Гістерезис
T/absH:", OutputBuf), &Font_12x20, fm->color_scheme->text_color_file,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 240, dist, "/", &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 2, dist*3, utf8_to_win1251("2. Max T in,
°C:", OutputBuf), &Font_12x20, COLOR_RED, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 2, dist*4, utf8_to_win1251("3. Max Tout,
°C:", OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 2, dist*6, utf8_to_win1251("4. Min T in,
°C:", OutputBuf), &Font_12x20, COLOR_RED, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 2, dist*7, utf8_to_win1251("5. Min Tout,
°C:", OutputBuf), &Font_12x20, fm->color_scheme->text_color_cursor,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 2, dist*9, utf8_to_win1251("6. Max H in,
%:", OutputBuf), &Font_12x20, COLOR_ORANGE, fm->color_scheme-
>bg_color, LCD_SYMBOL_PRINT_FAST);
```

```
     LCD_WriteString(lcd, 2, dist*10, utf8_to_win1251("7. Min H in,
%:", OutputBuf), &Font_12x20, COLOR_ORANGE, fm->color_scheme-
>bg_color, LCD_SYMBOL_PRINT_FAST);

     LCD_WriteString(lcd, 2, dist*12, utf8_to_win1251("8. Останнє
збереження:", OutputBuf), &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);

     if(settingRAM->demo_mode == 1)
     {
          LCD_WriteString(lcd, 50, dist*14,
utf8_to_win1251("демонстраційний режим", OutputBuf), &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
     }
}

//форматування тривалості часу у формат "123дн 13год 21хв" і друк
вказаним кольором по вказаних координатах
void Print_period(uint32_t period, int x, int y, uint32_t color)
{
     uint32_t all_hvilin = period / TIME_MIN;

     uint16_t k_dniv = (uint16_t)(all_hvilin / 1440);

     uint8_t k_hodin = (uint8_t)((all_hvilin % 1440) / 60);

     uint8_t k_hvilin = (uint8_t)(all_hvilin - k_dniv * 1440 -
k_hodin * 60);

     uint8_t i;
     i = 0;

     char t[17];

     uint16_t temp;

     if(k_dniv > 0)
     {
          temp = k_dniv;
          if (temp > 99)
          {
               t[i] = temp / 100 + 48;
               temp = temp % 100;
               i++;
          }

          if (temp > 9)
          {
               t[i] = temp / 10 + 48;
               temp = temp % 10;
```

```
            i++;
        }
        t[i] = temp + 48; i++;
        t[i] = 228; i++;    //"д"

        if (k_dniv < 99)
        {
            t[i] = 237; i++;    //"н"
        }
        t[i] = 32; i++;         // " "
    }

    if(k_hodin > 0)
    {
        temp = k_hodin;
        if (temp > 9)
        {
            t[i] = temp / 10 + 48;
            temp = temp % 10;
            i++;
        }

        t[i] = temp + 48; i++;
        t[i] = 227; i++;    //"г"

        if(k_dniv == 0)
        {
            t[i] = 238; i++;    //"о"
            t[i] = 228; i++;    //"д"
        }
        t[i] = 32; i++;         // " "
    }

    temp = k_hvilin;
    if (temp > 9)
    {
        t[i] = temp / 10 + 48;
        temp = temp % 10;
        i++;
    }

    t[i] = temp + 48; i++;
    t[i] = 245; i++;    //"х"
    t[i] = 226; i++;    //"в"
    t[i] = 0;           // Кінець строки

    LCD_WriteString(lcd, x, y, t, &Font_12x20, color, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
}

// Друк значень на екрані 0
```

```
void printInfoUpdate0(void)
{
//   Час роботи системи:
     Print_period(settingRAM->hour_unit, 200, dist*2, fm-
>color_scheme->text_color_file);

//   Час роботи вент-рів:
     Print_period(settingRAM->hour_motor, 200, dist*5, fm-
>color_scheme->text_color_file);

//   Час роботи нагрівача:
     Print_period(settingRAM->hour_heat, 200, dist*8, fm-
>color_scheme->text_color_file);
}

// Друк значень на екрані 1
void printInfoUpdate1(void)
{
//   LCD_FillWindow(lcd, 150, 36, 319, 122, COLOR_BLACK);
//   LCD_FillWindow(lcd, 215, 186, 319, 197, COLOR_BLACK);

//   1. Гістерезис T/absH:
     sprintf(TempBuf, "%.2f / %.2f", (float)(dT_OFF/100.0),
(float)(dH_OFF/100.0));
     LCD_WriteString(lcd, 200, 3+dist, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);

//   2. Max T in, °C:
     sprintf(TempBuf, "%.1f", (float)(settingRAM->tInMax/100.0));
     LCD_WriteString(lcd, 150, dist*3, TempBuf, &Font_12x20,
COLOR_RED, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     Print_time_date(settingRAM->tInMaxTimeDate, 200, dist*3,
COLOR_RED);

//3. Max T out, °C:
     sprintf(TempBuf, "%.1f", (float)(settingRAM->tOutMax/100.0));
     LCD_WriteString(lcd, 150, dist*4, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
     Print_time_date(settingRAM->tOutMaxTimeDate, 200, dist*4, fm-
>color_scheme->text_color_cursor);

//4. Min T in, °C:
     sprintf(TempBuf, "%.1f", (float)(settingRAM->tInMin/100.0));
     LCD_WriteString(lcd, 150, dist*6, TempBuf, &Font_12x20,
COLOR_RED, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     Print_time_date(settingRAM->tInMinTimeDate, 200, dist*6,
COLOR_RED);

//5. Min T out, °C:
```

```
     sprintf(TempBuf, "%.1f", (float)(settingRAM->tOutMin/100.0));
     LCD_WriteString(lcd, 150, dist*7, TempBuf, &Font_12x20, fm-
>color_scheme->text_color_cursor, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
     Print_time_date(settingRAM->tOutMinTimeDate, 200, dist*7, fm-
>color_scheme->text_color_cursor);

//6. Max H, %:
     sprintf(TempBuf, "%.1f\n", (float)(settingRAM->hMax/100.0));
     LCD_WriteString(lcd, 150, dist*9, TempBuf, &Font_12x20,
COLOR_ORANGE, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     Print_time_date(settingRAM->hMaxTimeDate, 200, dist*9,
COLOR_ORANGE);

//7. Min H, %:
     sprintf(TempBuf, "%.1f\n", (float)(settingRAM->hMin/100.0));
     LCD_WriteString(lcd, 150, dist*10, TempBuf, &Font_12x20,
COLOR_ORANGE, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
     Print_time_date(settingRAM->hMinTimeDate, 200, dist*10,
COLOR_ORANGE);

//8. Останнє збереження:
     Print_time_date(settingRAM->SaveTimeDate, 200, dist*12, fm-
>color_scheme->text_color_file);
}
/*
//функція управління сервоприводами
void servo_deg(uint8_t Nom_serv, uint8_t deg)     //Nom_serv - номер
сервопривода, deg - кут повороту в градусах
{
     deg >= 180 ? deg = 180 : deg;
     if(Nom_serv == 1 )
     {
         TIM2->CCR1 = 8000 - (deg * 32.44);
     }
     else
          {
               TIM2->CCR2 = 8000 - (deg * 32.44);
          }
}
*/

//відкриття повітряної заслінки
void Open_Servo(uint8_t Nom_serv)
{
//   servo_deg(Nom_serv, 90); //5080
     if(Nom_serv == 1 )  TIM2->CCR1 = settingRAM->serv_90_deg * 20 +
3900; //5080;
     else                TIM2->CCR2 = settingRAM->serv_90_deg * 20 +
3900; //5080;
}
```

```
//закриття повітряної заслінки
void Close_Servo(uint8_t Nom_serv) //Nom_serv - номер заслінки
{
//   servo_deg(Nom_serv, 0);       //8000
     if(Nom_serv == 1 )  TIM2->CCR1 = settingRAM->serv_0_deg * 20 +
3900; //8000;
     else                TIM2->CCR2 = settingRAM->serv_0_deg * 20 +
3900; //8000;
}

void MotorON(void)  //увімкнення вентиляторів, відкриття повітряних
заслінок
{
     if(settingRAM->servo_mode == 0)    // якщо заслінка з серво-
приводом
     {
          MotorON_SRV();
     }
     else                               // якщо заслінка з
електромагнітним механізмом
     {
          MotorON_EM();
     }
}

void MotorON_SRV(void)   //увімкнення вентиляторів, відкриття
повітряних заслінок з серво-приводом
{
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);

     switch (packet->Step_Motor_In)
     {
          case  0:  HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port,
PIN_SRV_PWR_Pin, GPIO_PIN_RESET);  //увімкнення реле подачи живлення
на сервоприводи
                    LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("увімкнено живлення на сервоприводи", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                    break;
          case  1:  Open_Servo(1);
                    //відкриття 1-ї повітряної заслінки
                    LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("відкрито 1-у повітряну заслінку", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                    break;
          case  2:  Open_Servo(2);
                    //відкриття 2-ї повітряної заслінки
```

```
                        LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("відкрито 2-у повітряну заслінку", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                        break;
          case  3:  HAL_GPIO_WritePin(PIN_RELAY_GPIO_Port,
PIN_RELAY_Pin, GPIO_PIN_RESET);    //увімкнення вентиляторів
                        LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("увімкнено живлення на вентилятори", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                        break;
          case  4:  HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port,
PIN_SRV_PWR_Pin, GPIO_PIN_SET);                  //вимкнення реле
подачи живлення на сервоприводи
                        LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("вимкнено живлення на сервоприводи", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);

                        FLAG_MOTOR_ON;
                        FLAG_MOTOR_IN_OFF;
                        packet->Step_Motor_In = 0;
                        writeSD(); // Запис на SD-карту

                        LCD_FillWindow(lcd, 0, 0, 319, 19, fm-
>color_scheme->cursor_color);
                        Setting(0);
                        cubik();
                        show_time();
                        break;
     }
     packet->Step_Motor_In++;
     packet_ready_E0 = 1;
     packet_ready_E1 = 1;
}

void MotorON_EM(void)    //увімкнення вентиляторів, відкриття
повітряних заслінок з електро-магнітним механізмом
{
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);

     switch (packet->Step_Motor_In)
     {
          case  0:  HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port,
PIN_SRV_PWR_Pin, GPIO_PIN_SET);    //вимкнення реле подачи живлення
на електромагніти
                        LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("вимкнено електромагніти", OutputBuf), &Font_12x20,
```

```
fm->color_scheme->text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
                         break;
          case  1:  HAL_GPIO_WritePin(PIN_RELAY_GPIO_Port,
PIN_RELAY_Pin, GPIO_PIN_RESET);    //увімкнення вентиляторів
                         LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("увімкнено живлення на вентилятори", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                         break;

                         FLAG_MOTOR_ON;
                         FLAG_MOTOR_IN_OFF;
                         packet->Step_Motor_In = 0;
                         writeSD();// Запис на SD-карту

                         LCD_FillWindow(lcd, 0, 0, 319, 19, fm-
>color_scheme->cursor_color);
                         Setting(0);
                         cubik();
                         show_time();
                         break;
     }
     packet->Step_Motor_In++;
     packet_ready_E0 = 1;
     packet_ready_E1 = 1;
}

void MotorOFF(void) //вимкнення вентиляторів, закриття повітряних
заслінок
{
     if(settingRAM->servo_mode == 0)    // якщо заслінка з серво-
приводом
     {
          MotorOFF_SRV();
     }
     else                               // якщо заслінка з
електромагнітним механізмом
     {
          MotorOFF_EM();
     }
}

void MotorOFF_SRV(void)  //вимкнення вентиляторів, закриття
повітряних заслінок з серво-приводом
{
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);

     switch (packet->Step_Motor_Out)
     {
```

```
          case  0:  HAL_GPIO_WritePin(PIN_RELAY_GPIO_Port,
PIN_RELAY_Pin, GPIO_PIN_SET);      //вимкнення вентиляторів
                          LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("вимкнено живлення на вентилятори", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                          break;
          case  1:  HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port,
PIN_SRV_PWR_Pin, GPIO_PIN_RESET);//увімкнення реле подачи живлення на
сервоприводи
                          LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("увімкнено живлення на сервоприводи", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                          break;
          case  2:  Close_Servo(1);
                    //закриття 1-ї повітряної заслінки
                          LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("закрито 1-у повітряну заслінку", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                          break;
          case  3:  Close_Servo(2);
                    //закриття 2-ї повітряної заслінки
                          LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("закрито 2-у повітряну заслінку", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                          break;
          case  4:  HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port,
PIN_SRV_PWR_Pin, GPIO_PIN_SET);             //вимкнення реле подачи
живлення на сервоприводи
                          LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("вимкнено живлення на сервоприводи", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);

                          FLAG_MOTOR_OFF;
                          FLAG_MOTOR_OUT_OFF;
                          packet->Step_Motor_Out = 0;
                          writeSD();// Запис на SD-карту

                          LCD_FillWindow(lcd, 0, 0, 319, 19, fm-
>color_scheme->cursor_color);

                          Setting(0);
                          cubik();  // зафарбовування квадрата
                          show_time();

                          break;
     }
```

```
     packet->Step_Motor_Out++;
     packet_ready_E0 = 1;
     packet_ready_E1 = 1;
}

void MotorOFF_EM(void)   //вимкнення вентиляторів, закриття
повітряних заслінок з електро-магнітним механізмом
{
     LCD_FillWindow(lcd, 0, 0, 319, 19, fm->color_scheme-
>cursor_color);

     switch (packet->Step_Motor_Out)
     {
          case  0:  HAL_GPIO_WritePin(PIN_RELAY_GPIO_Port,
PIN_RELAY_Pin, GPIO_PIN_SET);      //вимкнення вентиляторів
                         LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("вимкнено живлення на вентилятори", OutputBuf),
&Font_12x20, fm->color_scheme->text_color_cursor, fm->color_scheme-
>cursor_color, LCD_SYMBOL_PRINT_FAST);
                         break;
          case  1:  HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port,
PIN_SRV_PWR_Pin, GPIO_PIN_RESET);//увімкнення реле подачи живлення на
електромагніти
                         LCD_WriteString(lcd, 5, 3,
utf8_to_win1251("увімкнено електромагніти", OutputBuf), &Font_12x20,
fm->color_scheme->text_color_cursor, fm->color_scheme->cursor_color,
LCD_SYMBOL_PRINT_FAST);
                         break;

                         FLAG_MOTOR_OFF;
                         FLAG_MOTOR_OUT_OFF;
                         packet->Step_Motor_Out = 0;
                         writeSD();// Запис на SD-карту

                         LCD_FillWindow(lcd, 0, 0, 319, 19, fm-
>color_scheme->cursor_color);

                         Setting(0);
                         cubik();
                         show_time();

                         break;
     }
     packet->Step_Motor_Out++;
     packet_ready_E0 = 1;
     packet_ready_E1 = 1;
}

//приведення показів датчиків до одного рівня (коригування)
int16_t F_H_In_err(float H_X)
{
```

```
     float Y;
     Y = settingRAM->H2 * settingRAM->d_H1 - settingRAM->H1 *
settingRAM->d_H2;
     Y = Y - (float)(settingRAM->d_H1 - settingRAM->d_H2) * H_X;
     Y = Y / (settingRAM->H2 - settingRAM->H1);
     return (int16_t)Y;
}

int16_t F_T_In_err(float T_X)
{
     float Y;
     Y = settingRAM->T2 * settingRAM->d_T1 - settingRAM->T1 *
settingRAM->d_T2;
     Y = Y - (float)(settingRAM->d_T1 - settingRAM->d_T2) * T_X;
     Y = Y / (settingRAM->T2 - settingRAM->T1);
     return (int16_t)Y;
}

/*
#define H_X1   35.0
#define DH_X1  0.0       //218 різниця між датчиками в сотих г/м*3
при вологості H_X1
#define H_X2   72.0
#define DH_X2  0.0       //різниця між датчиками в сотих г/м*3 при
вологості H_X2

int16_t F_H_In_err(float H_X)
{
     float Y;
     Y = H_X2 * DH_X1 - H_X1 * DH_X2;
     Y = Y - (DH_X1 - DH_X2) * H_X;
     Y = Y / (H_X2 - H_X1);
     return (int16_t)Y;
}


#define T_X1   7.0
#define DT_X1  0.0       //різниця між датчиками в сотих °C при
температурі T_X1
#define T_X2   25.0
#define DT_X2  0.0       //-30 різниця між датчиками в сотих °C при
температурі T_X2

int16_t F_T_In_err(float T_X)
{
     float Y;
     Y = T_X2*DT_X1 - T_X1 * DT_X2;
     Y = Y - (DT_X1 - DT_X2) * T_X;
     Y = Y / (T_X2 - T_X1);
     return (int16_t)Y;
}
```

```
*/


int8_t user_i2c_read(uint8_t id, uint8_t reg_addr, uint8_t *data,
uint16_t len)
{
  if(HAL_I2C_Master_Transmit(&hi2c1, (id << 1), &reg_addr, 1, 10) !=
HAL_OK) return -1;
  if(HAL_I2C_Master_Receive(&hi2c1, (id << 1) | 0x01, data, len, 10)
!= HAL_OK) return -1;

  return 0;
}

void user_delay_ms(uint32_t period)
{
  HAL_Delay(period);
}

int8_t user_i2c_write(uint8_t id, uint8_t reg_addr, uint8_t *data,
uint16_t len)
{
  int8_t *buf;
  buf = malloc(len +1);
  buf[0] = reg_addr;
  memcpy(buf +1, data, len);

  if(HAL_I2C_Master_Transmit(&hi2c1, (id << 1), (uint8_t*)buf, len +
1, HAL_MAX_DELAY) != HAL_OK) return -1;

  free(buf);
  return 0;
}

  int8_t rslt;

 // 1-й датчик

  struct bme280_dev dev1;
  struct bme280_data comp_data1;

  // 2-й датчик
  struct bme280_dev dev2;
  struct bme280_data comp_data2;

//функція для обчислення значення абсолютної вологості
int16_t calculationAbsH_P(float t, float h, float p)
{
      float P_nasish_T = 6.112 * pow(2.718281828, 17.62 * t / (243.12
+ t));//гПа
      float P_nasish_P = 1.0016 + 0.00000315 * p - 0.074 / p;//гПа
```

```
      float P_nasish = P_nasish_P * P_nasish_T;//гПа

      float P_par = h * P_nasish;//гПа

      float AbsH_P = 100000.0 * P_par / (461.5 * (t + 273.15) );

      return (uint16_t)AbsH_P;
}

void readDHT(void) //читання параметрів з датчиків
{
     int8_t rslt1;
     int8_t rslt2;

     /* 1-й BME280 - dev1*/
     rslt1 = bme280_get_sensor_data(BME280_ALL, &comp_data1,
&dev1);// отримуємо дані з першого датчика (в шафі)
     if(rslt1 == BME280_OK)
     {
          int16_t Temp_In = comp_data1.temperature;         //
Отримуємо з датчика значення температури в сотих градуса °C
          packet->tIn = Temp_In + F_T_In_err(Temp_In/100.0);     //
враховуємо похибку (різницю між датчиками)

          int16_t Hum_In = comp_data1.humidity / 10.24;     //
Отримуємо значення відносної вологості у відсотках (якщо значеня з
датчика поділити на 1024)

     //і переводимо в соті долі (множимо на 100)
          packet->relHIn = Hum_In + F_H_In_err(Hum_In/100.0);    //
враховуємо похибку (різницю між датчиками)

          packet->p = comp_data1.pressure / 1000.0;    // Отримуємо
значення тиску в десятих hPa, якщо поділити на 1000

          //враховуємо значення абсолютної вологості в шафі в сотих
грама на м*3
          packet->absHIn = calculationAbsH_P((float)((packet-
>tIn)/100.0), (float)((packet->relHIn)/100.0), (float)((packet-
>p)/10.0));

          ave_tIn->Push(ave_tIn, packet->tIn);
          ave_relHIn->Push(ave_relHIn, packet->relHIn);
          ave_p->Push(ave_p, packet->p);
          ave_absHIn->Push(ave_absHIn, packet->absHIn);

          //для приведення показів датчиків до одного рівня
          ave_tInK->Push(ave_tInK, Temp_In);
          ave_relHInK->Push(ave_relHInK, Hum_In);
     }
```

```
     /* 2-й BME280 - dev2*/
     rslt2 = bme280_get_sensor_data(BME280_ALL, &comp_data2,
&dev2);// отримуємо дані з другого датчика (ззовні)
     if(rslt2 == BME280_OK)
     {
          packet->tOut = comp_data2.temperature;            //
Отримуємо з датчика значення температури в сотих градуса °C
          packet->relHOut = comp_data2.humidity / 10.24;    //
Отримуємо значення відносної вологості у відсотках (якщо значеня з
датчика поділити на 1024)

     //і переводимо в соті долі (множимо на 100)
          packet->p = comp_data2.pressure / 1000.0;         //
Отримуємо значення тиску в hPa, якщо поділити на 1000

          //вираховуємо значення абсолютної вологості ззовні в сотих
грама на м*3
          packet->absHOut     = calculationAbsH_P(((float)(packet-
>tOut)/100.0), (float)((packet->relHOut)/100.0), (float)((packet-
>p)/10.0));

          ave_tOut->Push(ave_tOut, packet->tOut);
          ave_relHOut->Push(ave_relHOut, packet->relHOut);
          ave_p->Push(ave_p, packet->p);
          ave_absHOut->Push(ave_absHOut, packet->absHOut);

          if(rslt1 == BME280_OK)
          {
               //для приведення показів датчиків до одного рівня
               ave_dT->Push(ave_dT, ave_tOut->Mean(ave_tOut) -
ave_tInK->Mean(ave_tInK));
               ave_dH->Push(ave_dH, ave_relHOut->Mean(ave_relHOut) -
ave_relHInK->Mean(ave_relHInK));

               ave_Tsrd->Push(ave_Tsrd, (ave_tOut->Mean(ave_tOut) +
ave_tInK->Mean(ave_tInK))/2);
               ave_Hsrd->Push(ave_Hsrd, (ave_relHOut-
>Mean(ave_relHOut) + ave_relHInK->Mean(ave_relHInK))/2);
          }
     }
}

  //обробник переривань таймера
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
     if(htim->Instance == TIM1) //перевіряємо чи переривання від 1-го
таймера TIM1
     {
          if(flag_measurement == 0)     //страхування, щоб другий раз
не знімати дані, якщо ще не закінчився попередній раз
          {
```

```
                flag_measurement = 1;    //встановлюємо ознаку, що
почали обробку показів датчиків

                HAL_IWDG_Refresh(&hiwdg); // Скидання сторожового
таймера

                readDHT();          //читання параметрів з датчиків

                CheckEkran();       //перевірка чи не пора вимкнути
дисплей

                get_time(0);        // отримання актуальних дати, часу
і формування масиву TimeBuf для виводу

                if(FLAG_MOTOR_OUT_CHECK)//якщо іде процедура вимкнення
вентиляторів
                {
                     MotorOFF();    //вимкнення вентиляторів, закриття
повітряних заслінок
                }
                else
                {
                     if(FLAG_MOTOR_IN_CHECK)  //якщо іде процедура
запуску вентиляторів
                     {
                          MotorON();     //увімкнення вентиляторів,
відкриття повітряних заслінок
                     }
                     else
                     {
                          CheckON();     // Перевірка стану системи
                     }
                }

                if (settingRAM->tInMax < ave_tIn->Mean(ave_tIn))//
знаходження максимальної температури в шафі
                {
                     settingRAM->tInMax = ave_tIn->Mean(ave_tIn);
                     packet_ready_E1 = 1;

                     settingRAM->tInMaxTimeDate[0] = sTime.Minutes;
                     settingRAM->tInMaxTimeDate[1] = sTime.Hours;
                     settingRAM->tInMaxTimeDate[2] = sDate.Date;
                     settingRAM->tInMaxTimeDate[3] = sDate.Month;
                     settingRAM->tInMaxTimeDate[4] = sDate.Year;
                }
                else
                {
                     if (settingRAM->tInMin > ave_tIn-
>Mean(ave_tIn))// знаходження мінімальної температури в шафі
                     {
```

```
                    settingRAM->tInMin = ave_tIn->Mean(ave_tIn);
                    packet_ready_E1 = 1;

                    settingRAM->tInMinTimeDate[0] =
sTime.Minutes;
                    settingRAM->tInMinTimeDate[1] = sTime.Hours;
                    settingRAM->tInMinTimeDate[2] = sDate.Date;
                    settingRAM->tInMinTimeDate[3] = sDate.Month;
                    settingRAM->tInMinTimeDate[4] = sDate.Year;
               }
          }

          if (settingRAM->tOutMax < ave_tOut->Mean(ave_tOut))
     // знаходження максимальної температури ззовні
          {
               settingRAM->tOutMax = ave_tOut->Mean(ave_tOut);
               packet_ready_E1 = 1;

               settingRAM->tOutMaxTimeDate[0] = sTime.Minutes;
               settingRAM->tOutMaxTimeDate[1] = sTime.Hours;
               settingRAM->tOutMaxTimeDate[2] = sDate.Date;
               settingRAM->tOutMaxTimeDate[3] = sDate.Month;
               settingRAM->tOutMaxTimeDate[4] = sDate.Year;
          }
          else
          {
               if (settingRAM->tOutMin > ave_tOut-
>Mean(ave_tOut)) // знаходження мінімальної температури ззовні
               {
                    settingRAM->tOutMin = ave_tOut-
>Mean(ave_tOut);
                    packet_ready_E1 = 1;

                    settingRAM->tOutMinTimeDate[0] =
sTime.Minutes;
                    settingRAM->tOutMinTimeDate[1] =
sTime.Hours;
                    settingRAM->tOutMinTimeDate[2] = sDate.Date;
                    settingRAM->tOutMinTimeDate[3] =
sDate.Month;
                    settingRAM->tOutMinTimeDate[4] = sDate.Year;
               }
          }

          if (settingRAM->hMax < ave_relHIn->Mean(ave_relHIn))
     // знаходження максимальної вологості
          {
               settingRAM->hMax = ave_relHIn->Mean(ave_relHIn);
               packet_ready_E1 = 1;

               settingRAM->hMaxTimeDate[0] = sTime.Minutes;
```

```
                    settingRAM->hMaxTimeDate[1] = sTime.Hours;
                    settingRAM->hMaxTimeDate[2] = sDate.Date;
                    settingRAM->hMaxTimeDate[3] = sDate.Month;
                    settingRAM->hMaxTimeDate[4] = sDate.Year;
               }
               else
               {
                    if (settingRAM->hMin > ave_relHIn-
>Mean(ave_relHIn))  // знаходження мінімальної вологості
                    {
                         settingRAM->hMin = ave_relHIn-
>Mean(ave_relHIn);
                         packet_ready_E1 = 1;

                         settingRAM->hMinTimeDate[0] = sTime.Minutes;
                         settingRAM->hMinTimeDate[1] = sTime.Hours;
                         settingRAM->hMinTimeDate[2] = sDate.Date;
                         settingRAM->hMinTimeDate[3] = sDate.Month;
                         settingRAM->hMinTimeDate[4] = sDate.Year;
                    }
               }

               if (FLAG_MOTOR_CHECK)    //якщо вентилятори працюють
               {
                    chart_dry->ChartMotor = 1;//ознака того що
потрібно показувати увімкнення вентиляторів на графіках
                    settingRAM->hour_motor++;//облік часу роботи
вентиляторів
               }

               if (FLAG_HEAT_CHECK)     //якщо обігрівач працює
               {
                    chart_dry->ChartHeat = 1;//ознака того що
потрібно показувати увімкнення обігрівача на графіках
                    settingRAM->hour_heat++;//облік часу роботи
обігрівача
               }

               settingRAM->hour_unit++; //облік часу роботи системи

//   Оновлюємо максимум і мінімум температур на SD
//   Дані оновлюються раз в 3 секунды (TIME_SCAN_SENSOR),
//   а пишемо на SD раз в 3 години для економії ресурсу SD, кількість
циклів перезапису обмежена

               settingRAM->tick_SDflash++;

               if (((int16_t)settingRAM->tick_SDflash) >=
(int16_t)TIME_HOUR) // чи пора писати на SD
               {
                    writeSD();// Запис на SD-карту
```

```
                }

                if (FLAG_EKRAN_CHECK)
                {
                     hour_ekran++;  //облік часу увімкненого дисплея
                }

                dry_get_data();     // заповнення масиву для графіків

                flag_measurement = 0;//скидаємо ознаку, що почали
обробку показів датчиків
                packet_ready = 1;//встановлюємо ознаку «опитування
датчиків»
           }
     }
}

void OpenFileManager (void)
{
     FileManagerPopupWin(fm, 0);   //"Спливаюче" вікно файлового
менеджера
     fm->Show(fm);                 //Запуск менеджера файлів

     if (fm->GetStatus(fm) == File_Manager_OK) // Завершення без
помилок із вибором файлу?
     {
          // "Склеюємо" шлях до файлу з ім'ям файлу
          char *tmp = my_strsplice(2, "/", fm->GetFilePath(fm), fm-
>GetFileName(fm));

          read_history(tmp, 1);
     }
     else
     {
          // Виводимо повідомлення, що роботу файлового менеджера
завершено з помилкою
          LCD_WriteString(lcd, 0, 200, "Error FileManager!", fm-
>font, fm->color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
     }

     // Очікуємо натискання будь-якої кнопки
     while(!KEYB_Inkeys());

     printSettingStatic();
}

// обробник натискання кнопок
// функція вертає 1, якщо змінився екран і потрібно надрукувати новий
int8_t button_click_handler(void)
{
```

```
     if(flag_measurement == 1) { return 0; } //якщо зараз виконується
отримання даних з датчиків, то вихід

     hour_ekran     =    0;   //скидання часу роботи дисплея після
останнього натискання кнопки

     uint32_t keys = 0;
     keys = KEYB_Inkeys();          //зчитуємо стан кнопок з буфера

     if (~FLAG_EKRAN_CHECK)         //якщо дисплей потушений
     {
          EKRAN_ON;       //то вмикаємо дисплей
          return 1;       //вертаємо 1 щоб оновити поточний екран, так
як при потушеному екрані дані та графіки не передруковувались
     }

// Обробка натискання кнопок
//вхід/вихід(без збереження змін) в/з меню налаштувань
     if ( keys & (1<<KEYB_LEFT)  ) //якщо нажали KEYB_LEFT
     {
          if(ScreenMode->Get_Poz(ScreenMode) == 0)     //і якщо
знаходимся на інфо екрані
          {
               printSettingStatic();    //то переходимо на екран
налаштувань з вибором режиму роботи
               return 0;
          }
          else //якщо знаходимся не на інфо екрані
          {
               ScreenMode->Set_Poz(ScreenMode, 0);     //то виходимо
з поточного екрана без збереження змін на інфо екран
               return 1;                               // змінився
екран - потрібно надрукувати новий
          }
     }

     uint8_t poz;
     if (keys & (1<<KEYB_DOWN))    //якщо нажали KEYB_DOWN
     {
          if (ScreenMode->Get_Poz(ScreenMode) == 1)    //якщо
знаходимся на екрані налаштувань
          {
               //стираємо символ ">" (друкуємо кольором фона на
старому місці)
               LCD_WriteString(lcd, 5, dist * (1 + Screens[1]-
>Get_Poz(Screens[1])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 9)    //якщо
знаходимся на екрані "Колірна схема"
```

```
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[9]-
>Get_Poz(Screens[9])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 10)   //якщо
знаходимся на екрані "Демонстраційний режим"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[10]-
>Get_Poz(Screens[10])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 12)   //якщо
знаходимся на екрані "Налаштування повітряних заслінок"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[12]-
>Get_Poz(Screens[12])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 13)   //якщо
знаходимся на екрані "Тип повітряних заслінок"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[13]-
>Get_Poz(Screens[13])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
          }

          poz = ScreenMode->Get_Poz(ScreenMode);
          Screens[poz]->Up(Screens[poz]);    //перехід до іншого
екрану вперед (чи до іншої позиції на екрані налаштувань)

          if (ScreenMode->Get_Poz(ScreenMode) == 1)    //якщо
знаходимся на екрані налаштувань
          {
               //друкуємо символ ">" в новому місці
               LCD_WriteString(lcd, 5, dist * (1 + Screens[1]-
>Get_Poz(Screens[1])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 9)    //якщо
знаходимся на екрані "Колірна схема"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[9]-
>Get_Poz(Screens[9])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
```

```
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 10)   //якщо
знаходимся на екрані "Демонстраційний режим"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[10]-
>Get_Poz(Screens[10])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 12)   //якщо
знаходимся на екрані "Налаштування повітряних заслінок"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[12]-
>Get_Poz(Screens[12])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 13)   //якщо
знаходимся на екрані "Тип повітряних заслінок"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[13]-
>Get_Poz(Screens[13])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 11)   //якщо
знаходимся на екрані "Коригування датчиків"
          {
               KorDatUpdate();     // Оновлення екрану "Коригування
датчиків"
               return 0;
          }

          if(ScreenMode->Get_Poz(ScreenMode) >= 2 && ScreenMode-
>Get_Poz(ScreenMode) <= 8)    // якщо на екрані "Налаштування
Часу/Дати"
          {
               printSetTimeUpdate();    // Оновлення екрану
"Налаштування Часу/Дати"
               return 0;
          }

          if(ScreenMode->Get_Poz(ScreenMode) >= 14)    // якщо на
екрані "Налаштування серво-приводів"
          {
               printSetServoUpdate();   // Оновлення екрану
"Налаштування серво-приводів"
```

```
                return 0;
            }

            return 1;                                   // змінився екран -
потрібно надрукувати новий
        }

        if (keys & (1<<KEYB_UP)) //якщо нажали KEYB_UP
        {
            if (ScreenMode->Get_Poz(ScreenMode) == 1)    //якщо
знаходимся на екрані налаштувань
            {
                //стираємо символ ">" (друкуємо кольором фона на
старому місці)
                LCD_WriteString(lcd, 5, dist * (1 + Screens[1]-
>Get_Poz(Screens[1])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
            }

            if (ScreenMode->Get_Poz(ScreenMode) == 9)    //якщо
знаходимся на екрані "Колірна схема"
            {
                LCD_WriteString(lcd, 5, dist * (1 + Screens[9]-
>Get_Poz(Screens[9])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
            }

            if (ScreenMode->Get_Poz(ScreenMode) == 10)   //якщо
знаходимся на екрані "Демонстраційний режим"
            {
                LCD_WriteString(lcd, 5, dist * (1 + Screens[10]-
>Get_Poz(Screens[10])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
            }

            if (ScreenMode->Get_Poz(ScreenMode) == 12)   //якщо
знаходимся на екрані "Налаштування повітряних заслінок"
            {
                LCD_WriteString(lcd, 5, dist * (1 + Screens[12]-
>Get_Poz(Screens[12])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
            }

            if (ScreenMode->Get_Poz(ScreenMode) == 13)   //якщо
знаходимся на екрані "Тип повітряних заслінок"
            {
                LCD_WriteString(lcd, 5, dist * (1 + Screens[13]-
>Get_Poz(Screens[13])), ">", &Font_12x20, fm->color_scheme->bg_color,
fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
            }
```

```
          poz = ScreenMode->Get_Poz(ScreenMode);
          Screens[poz]->Down(Screens[poz]); //перехід до іншого
екрану назад (чи до іншої позиції на екрані налаштувань)

          if (ScreenMode->Get_Poz(ScreenMode) == 1)    //якщо
знаходимся на екрані налаштувань
          {
               //друкуємо символ ">" в новому місці
               LCD_WriteString(lcd, 5, dist * (1 + Screens[1]-
>Get_Poz(Screens[1])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 9)    //якщо
знаходимся на екрані "Колірна схема"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[9]-
>Get_Poz(Screens[9])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 10)   //якщо
знаходимся на екрані "Демонстраційний режим"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[10]-
>Get_Poz(Screens[10])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 12)   //якщо
знаходимся на екрані "Налаштування повітряних заслінок"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[12]-
>Get_Poz(Screens[12])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 13)   //якщо
знаходимся на екрані "Тип повітряних заслінок"
          {
               LCD_WriteString(lcd, 5, dist * (1 + Screens[13]-
>Get_Poz(Screens[13])), ">", &Font_12x20, fm->color_scheme-
>text_color_file, fm->color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
               return 0;
          }
```

```
          if (ScreenMode->Get_Poz(ScreenMode) == 11)    //якщо
знаходимся на екрані "Коригування датчиків"
          {
               KorDatUpdate();     // Оновлення екрану "Коригування
датчиків"
               return 0;
          }

          if(ScreenMode->Get_Poz(ScreenMode) >= 2 && ScreenMode-
>Get_Poz(ScreenMode) <= 8)     // якщо на екрані "Налаштування
Часу/Дати"
          {
               printSetTimeUpdate();     // Оновлення екрану
"Налаштування Часу/Дати"
               return 0;
          }

          if(ScreenMode->Get_Poz(ScreenMode) >= 14)     // якщо на
екрані "Налаштування серво-приводів"
          {
               printSetServoUpdate();    // Оновлення екрану
"Налаштування серво-приводів"
               return 0;
          }

          return 1;                                   // змінився екран -
потрібно надрукувати новий
     }

     if ( (keys & (1<<KEYB_RIGHT)) )     //якщо нажали KEYB_RIGHT (OK)
     {
          if(ScreenMode->Get_Poz(ScreenMode) == 0)      // якщо
знаходимся на якомусь інфо екрані
          {
               if (FLAG_EKRAN_CHECK) EKRAN_OFF;    //тушим дисплей
               return 0;
          }

          if (ScreenMode->Get_Poz(ScreenMode) == 1)               //
якщо знаходимся на екрані налаштувань
          {
               // обробник нажимання в залежності від положення
курсора
               poz = Screens[1]->Get_Poz(Screens[1]);
               switch (poz)
               {
                    case 0:   // 1.     Вимкнено
                    case 1: // 2.  Режим вентиляції
                    case 2: // 3.  Режим охолодження T>+9
                    case 3: // 4.  Осушення T>+9 dH>0.3
                    case 4: // 5.  Осушення T>+9 dH>5%
```

```
                    case 5: // 6.  Осушення T>+16 dH>0.6
                    case 6: // 7.  Осушення T>+16 dH>10%
                    case 7: // 8.  Осушення T>+16 dH>0.8

                         if(settingRAM->mode != poz)
                         {
                              settingRAM->mode = poz;
                              Setting(1);    // Вивід інформації про
поточний режим і збереження налаштувань на SD
                         }
                         break;
                    case 8: // Скидання min max температур
                         resetTemp();
                         break;
                    case 9: // Друк екрану "Налаштування Часу/Дати
                         printSetTimeStatic();
                         printSetTimeUpdate();
                         break;
                    case 10: // Друк екрану "Файловий менеджер"
                         OpenFileManager();
                         break;
                    case 11: // Друк екрану "Колірна схема"
                         printColorMode();
                         break;
                    case 12: // Вимкнено / Увімкнено демонстраційний
режим"
                         printDemoMode();
                         break;
                    case 13: // Друк екрану "Налаштування повітряних
заслінок"
                         printAirDampers();
                         break;
                    case 14:
                         printKorDat(); //Друк статичних даних екрану
"Коригування датчиків"
                         KorDatUpdate();// Друк змінних даних екрану
"Коригування датчиків"
                         break;
               }

               if(poz < 9)    //якщо було вибрано режим роботи
системи
               {
                    ScreenMode->Set_Poz(ScreenMode, 0);          //то
встановлюємо поточним екраном - інфо екран
               }
               return 1;                                   // змінився
екран - потрібно надрукувати новий
          }
          else
          {
```

```
                if (ScreenMode->Get_Poz(ScreenMode) < 9)     //якщо на
екрані налаштування Часу/Дати
                {
                     if(ScreenMode->Get_Poz(ScreenMode) == 8)
                     {
                          if(Screens[8]->Get_Poz(Screens[8]) == 0)
     //якщо нажали Так
                          {
                               SetTime();
                          }
                          ScreenMode->Set_Poz(ScreenMode, 0);
                          return 1;                                   //
змінився екран - потрібно надрукувати новий
                     }
                     else
                     {
                          if(ScreenMode->Get_Poz(ScreenMode) >= 2)
                          {
                               if(ScreenMode->Get_Poz(ScreenMode) < 7)
                               {
     //12.15 23.12.17 Пн Так
     //0  3  6  9  12 15 18
                                    LCD_WriteString(lcd, 50 +
24*(ScreenMode->Get_Poz(ScreenMode) - 2), 115, "^^", &Font_12x20, fm-
>color_scheme->bg_color, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
                                    ScreenMode->Up(ScreenMode);
                                    LCD_WriteString(lcd, 50+
24*(ScreenMode->Get_Poz(ScreenMode) - 2), 115, "^^", &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
                                    return 0;
                               }
                               else
                               {
                                    if(ScreenMode->Get_Poz(ScreenMode)
== 7)
                                    {
     //12.15 23.12.17 Пн Так
     //0  3  6  9  12 15 18
                                         LCD_WriteString(lcd, 50 +
24*(ScreenMode->Get_Poz(ScreenMode) - 2), 115, "^^", &Font_12x20, fm-
>color_scheme->bg_color, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
                                         ScreenMode->Up(ScreenMode);
                                         LCD_WriteString(lcd, 50+
24*(ScreenMode->Get_Poz(ScreenMode) - 1), 115, "^^", &Font_12x20, fm-
>color_scheme->text_color_file, fm->color_scheme->bg_color,
LCD_SYMBOL_PRINT_FAST);
                                         return 0;
                                    }
```

```
                              }
                         }
                    }
               }
               else //if (ScreenMode->Get_Poz(ScreenMode) >= 9)
               {
                    if (ScreenMode->Get_Poz(ScreenMode) == 9)
          // якщо знаходимся на екрані "Колірна схема"
                    {
                         fm->SetColor(fm, color_scheme[Screens[9]-
>Get_Poz(Screens[9])]);
                         settingRAM->color_mode = Screens[9]-
>Get_Poz(Screens[9]);
                         ScreenMode->Set_Poz(ScreenMode, 0);
                         writeSD();// Запис на SD-карту
                         return 1;                              //
змінився екран - потрібно надрукувати новий
                    }
                    else
                         if (ScreenMode->Get_Poz(ScreenMode) == 10)
               // якщо знаходимся на екрані "Демонстраційний режим"
                         {
                              if(Screens[10]->Get_Poz(Screens[10]) ==
1)   //якщо нажали "Демонстраційний режим"
                              {
                                   settingRAM->demo_mode = 1;
                                   TIME_PRINT_CHART = 3;    // Період
виводу однієї точки графіка, в DEMO одна точка кожні 3 сек.
                              }
                              else
     //якщо нажали "Робочий режим"
                              {
                                   settingRAM->demo_mode = 0;
                                   TIME_PRINT_CHART = 720;
     // Період виводу однієї точки графіка, сек. 12 хв. = 720 сек.
                              }
                              ScreenMode->Set_Poz(ScreenMode, 0);
                              writeSD();// Запис на SD-карту
                              return 1;
     // змінився екран - потрібно надрукувати новий
                         }
                         else
                              if(ScreenMode->Get_Poz(ScreenMode) ==
13)           // якщо знаходимся на екрані "Тип повітряних заслінок"
                              {
                                   if(Screens[13]-
>Get_Poz(Screens[13]) == 0)        //якщо нажали "з серво-приводом"
                                   {
                                        settingRAM->servo_mode = 0;
```

```
     HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_RESET);//увімкнення реле подачи живлення на сервоприводи
                                        if(FLAG_MOTOR_CHECK)
                                        {
                                             Open_Servo(1);
                                             HAL_Delay(1000);
                                             Open_Servo(2);
                                        }
                                        else
                                        {
                                             Close_Servo(1);
                                             HAL_Delay(1000);
                                             Close_Servo(2);
                                        }
                                        HAL_Delay(1000);

     HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_SET);//вимкнення реле подачи живлення на сервоприводи
                                   }
                                   else
          //якщо нажали "з магнітним механізмом"
                                   {
                                        settingRAM->servo_mode = 1;

                                        if(FLAG_MOTOR_CHECK)
                                        {

     HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_SET); //вимкнення реле подачи живлення на електромагніти
                                        }
                                        else
                                        {

     HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_RESET);//увімкнення реле подачи живлення на електромагніти
                                        }
                                   }

                                   ScreenMode->Set_Poz(ScreenMode,
0);
                                   writeSD();     // Запис на SD-
карту
                                   return 1; // змінився екран -
потрібно надрукувати новий
                              }
                              else
                                   if (ScreenMode-
>Get_Poz(ScreenMode) == 11)    //якщо на екрані "Коригування датчиків"
                                   {
```

```
                                        SetKorDat();   // збереження
змінених параметрів для коригування (або вихід без збереження)

                                        ScreenMode-
>Set_Poz(ScreenMode, 0); // повертаємось на інфо екран
                                        return 1;
                // змінився екран - потрібно надрукувати новий
                                   }
                                   else
                                        if(ScreenMode-
>Get_Poz(ScreenMode) == 12)             // якщо знаходимся на екрані
"Налаштування повітряних заслінок"
                                        {
                                             if(Screens[12]-
>Get_Poz(Screens[12]) == 0)        //якщо нажали "Тип повітряних
заслінок"
                                             {
                                                  printServoMode();
                                                  return 0;
                                             }
                                             else
                        //якщо нажали "Налаштування серво-приводів"
                                             {
                                                  printSetServo();

     printSetServoUpdate();
                                                  return 0;
                                             }
                                        }
                                        else
                                             if(ScreenMode-
>Get_Poz(ScreenMode) >= 14)   //якщо на екрані "Налаштування серво-
приводів"
                                             {
                                                  if(ScreenMode-
>Get_Poz(ScreenMode) == 16)
                                                  {

     if(Screens[16]->Get_Poz(Screens[16]) == 0)    //якщо нажали Так
                                                       {

     settingRAM->serv_90_deg = Screens[14]->Get_Poz(Screens[14]);

     settingRAM->serv_0_deg = Screens[15]->Get_Poz(Screens[15]);

     writeSD();     // Запис на SD-карту
                                                       }
                                                       ScreenMode-
>Set_Poz(ScreenMode, 0);
```

```
     HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_RESET);//увімкнення реле подачи живлення на сервоприводи

     if(FLAG_MOTOR_CHECK)
                                                        {

     Open_Servo(1);

     HAL_Delay(1000);

     Open_Servo(2);
                                                        }
                                                        else
                                                        {

     Close_Servo(1);

     HAL_Delay(1000);

     Close_Servo(2);
                                                        }

     HAL_Delay(1000);

     HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_SET);//вимкнення реле подачи живлення на сервоприводи

                                                            return 1;
                                  // змінився екран - потрібно
надрукувати новий
                                                        }
                                                        else
                                                        {

     LCD_WriteString(lcd, 50 + 75*(ScreenMode->Get_Poz(ScreenMode) -
14), 115, "^^^", &Font_12x20, fm->color_scheme->bg_color, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
                                                            ScreenMode-
>Up(ScreenMode);

     LCD_WriteString(lcd, 50+ 75*(ScreenMode->Get_Poz(ScreenMode) -
14), 115, "^^^", &Font_12x20, fm->color_scheme->text_color_file, fm-
>color_scheme->bg_color, LCD_SYMBOL_PRINT_FAST);
                                                            return 0;
                                                        }
                                                    }
                }
           }
     }
     return 0;
```

```
}

//функція відображає новий інфо екран
void Goto_New_Window(void)
{
     if(ScreenMode->Get_Poz(ScreenMode) ==   0)   //якщо на одному з
інфо екранів
     {
          LCD_Fill(lcd, fm->color_scheme->bg_color);
          switch (Screens[0]->Get_Poz(Screens[0]))     // в
залежності від номера екрана
          {
               case 0:   // інфо екран 0
                         printInfoStatic0(); // Друк статичної
картинки екрана 0
                         printInfoUpdate0(); // Друк значень на
екрані 0
                         break;
               case 1:   // інфо екран 1
                         printInfoStatic1(); // Друк статичної
картинки екрана 1
                         printInfoUpdate1(); // Друк значень на
екрані 1
                         break;
               case 2:   // інфо екран 2
                         dry_static(0);      // Друк статичної
картинки екрана 2
                         print_status0();    // Друк значень
                         printChart(0);      // Друк графіка
                         break;
               case 3: // інфо екран 3
                         dry_static(1);      // Друк статичної
картинки екрана 3
                         print_status0();    // Друк значень
                         printChart(1);      // Друк графіка
          }

          Setting(0);    // Вивід інформації про поточний режим

          cubik();// зафарбовування квадрата

          show_time();        //відображення поточного часу
          packet_ready_Time = 0;

          packet_ready_E0     =    0;
          packet_ready_E1     =    0;
     }
/*   else // якщо не на інфо екранах
     {
```

```
          if(ScreenMode->Get_Poz(ScreenMode) >= 2 && ScreenMode-
>Get_Poz(ScreenMode) <= 8)    // якщо на екрані "Налаштування
Часу/Дати"
          {
               printSetTimeUpdate();    // Оновлення екрану
"Налаштування Часу/Дати"
          }
     }
*/
}

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */

  // вмикаємо кешування інструкцій
  #if (INSTRUCTION_CACHE_ENABLE != 0U)
     ((FLASH_TypeDef *) ((0x40000000UL + 0x00020000UL) + 0x3C00UL))-
>ACR |= (0x1UL << (9U));
  #endif
  // вмикаємо кешування даних
  #if (DATA_CACHE_ENABLE != 0U)
     ((FLASH_TypeDef *) ((0x40000000UL + 0x00020000UL) + 0x3C00UL))-
>ACR |= (0x1UL << (10U));
  #endif
  // вмикаємо систему попередньої вибірки інструкцій
  #if (PREFETCH_ENABLE != 0U)
     ((FLASH_TypeDef *) ((0x40000000UL + 0x00020000UL) + 0x3C00UL))-
>ACR |= (0x1UL << (8U));
  #endif

  /* 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_DMA_Init();
  MX_SPI1_Init();
  MX_TIM10_Init();
  MX_SPI2_Init();
  MX_TIM3_Init();
  MX_FATFS_Init();
  MX_I2C1_Init();
  MX_TIM1_Init();
  MX_RTC_Init();
  MX_TIM2_Init();
  MX_IWDG_Init();
  /* USER CODE BEGIN 2 */

  // ТАЙМЕР htim1

    // очищаємо ознаку, щоб перший раз таймер не запустився відразу з
запуском МК
     __HAL_TIM_CLEAR_FLAG(&htim1, TIM_SR_UIF);

  //запускаємо таймер htim1 в режимі переривання для зчитування
значень з датчиків
    HAL_TIM_Base_Start_IT(&htim1);

    // ТАЙМЕР htim2
    //запускаємо таймер2 в режимі PWM (для управління сервоприводами)
для каналів 1 і 2
    HAL_TIM_PWM_Start_IT(&htim2, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start_IT(&htim2, TIM_CHANNEL_2);

  /* ---------------------------- Налаштування датчиків BME280 ------
------------------------ */
  /* 1-й BME280 - dev1*/
  dev1.dev_id = BME280_I2C_ADDR_PRIM;   // адреса 0x76
  dev1.intf = BME280_I2C_INTF;
  dev1.read = user_i2c_read;
  dev1.write = user_i2c_write;
  dev1.delay_ms = user_delay_ms;

  rslt = bme280_init(&dev1);

  dev1.settings.osr_h = BME280_OVERSAMPLING_1X;
  dev1.settings.osr_p = BME280_OVERSAMPLING_1X;
```

```
  dev1.settings.osr_t = BME280_OVERSAMPLING_1X;
  dev1.settings.filter = BME280_FILTER_COEFF_OFF;
  rslt = bme280_set_sensor_settings(BME280_OSR_PRESS_SEL |
BME280_OSR_TEMP_SEL | BME280_OSR_HUM_SEL | BME280_FILTER_SEL, &dev1);

  //налаштовуємо датчик 1 в режим роботи: нормальний. Він постійно
буде вимірювати, а з необхідною нам періодичність будемо забирати
значення
  rslt = bme280_set_sensor_mode(BME280_NORMAL_MODE, &dev1);

  /* 2-й BME280 - dev2*/
  dev2.dev_id = BME280_I2C_ADDR_SEC;    // адреса 0x77
  dev2.intf = BME280_I2C_INTF;
  dev2.read = user_i2c_read;
  dev2.write = user_i2c_write;
  dev2.delay_ms = user_delay_ms;

  rslt = bme280_init(&dev2);

  dev2.settings.osr_h = BME280_OVERSAMPLING_1X;
  dev2.settings.osr_p = BME280_OVERSAMPLING_1X;
  dev2.settings.osr_t = BME280_OVERSAMPLING_1X;
  dev2.settings.filter = BME280_FILTER_COEFF_OFF;
  rslt = bme280_set_sensor_settings(BME280_OSR_PRESS_SEL |
BME280_OSR_TEMP_SEL | BME280_OSR_HUM_SEL | BME280_FILTER_SEL, &dev2);

  //налаштовуємо датчик 2 в режим роботи: нормальний. Він постійно
буде вимірювати, а з необхідною нам періодичність будемо забирати
значення
    rslt = bme280_set_sensor_mode(BME280_NORMAL_MODE, &dev2);

  /* ---------------------------- Налаштування кнопок ---------------
---------------- */
  KEYB_Add_Button(KEY_LEFT_GPIO_Port, KEY_LEFT_Pin, KEYB_LEFT,
KEYB_BUTTON_ACTIVE);
  KEYB_Add_Button(KEY_RIGHT_GPIO_Port, KEY_RIGHT_Pin, KEYB_RIGHT,
KEYB_BUTTON_ACTIVE);
  KEYB_Add_Button(KEY_UP_GPIO_Port, KEY_UP_Pin, KEYB_UP,
KEYB_BUTTON_ACTIVE);
  KEYB_Add_Button(KEY_DOWN_GPIO_Port, KEY_DOWN_Pin, KEYB_DOWN,
KEYB_BUTTON_ACTIVE);

  TIM10->DIER |= TIM_DIER_UIE;     //дозволяємо переривання по
оновленню таймера TIM10
  TIM10->CR1  |= TIM_CR1_CEN; //вмикаємо таймер TIM10 для обробки
натискання кнопок
  /* ---------------------------------------------------------------
--------------- */

  /* ---------------------------- Налаштування дисплея --------------
------------------ */
```

```
  LCD_DMA_TypeDef dma_tx = { DMA2,
                             LL_DMA_STREAM_3 };

  LCD_BackLight_data bkl_data = { TIM3,
                                  LL_TIM_CHANNEL_CH1,
                                       0,
                                       0,
                                       50  };

  LCD_SPI_Connected_data spi_con = { SPI1,
                                     dma_tx,
                                          LCD_RES_GPIO_Port,
                                          LCD_RES_Pin,
                                          LCD_DC_GPIO_Port,
                                          LCD_DC_Pin,
                                          LCD_CS_GPIO_Port,
                                          LCD_CS_Pin         };

  #ifndef LCD_DYNAMIC_MEM
  LCD_Handler lcd1;
#endif


  //для дисплея на контролері ili9341
  LCD = LCD_DisplayAdd( LCD,
#ifndef  LCD_DYNAMIC_MEM
                             &lcd1,
#endif
                             240,
                             320,
                             ILI9341_CONTROLLER_WIDTH,
                             ILI9341_CONTROLLER_HEIGHT,
                             PAGE_ORIENTATION_LANDSCAPE_MIRROR,
                             ILI9341_Init,
                             ILI9341_SetWindow,
                             ILI9341_SleepIn,
                             ILI9341_SleepOut,
                             &spi_con,
                             LCD_DATA_16BIT_BUS,
                             bkl_data                  );

     lcd = LCD; //вказівник на дисплей
     LCD_Init(lcd);
     LCD_Fill(lcd, COLOR_BLACK);

//напис на екрані:
//Комп’ютеризована система контролю за умовами зберігання лікарських
засобів
     LCD_WriteString(lcd, 60, 0, utf8_to_win1251("Комп’ютеризована",
OutputBuf), &Font_15x25, COLOR_YELLOW, COLOR_BLACK,
LCD_SYMBOL_PRINT_FAST);
```

```
     LCD_WriteString(lcd, 60, 30, utf8_to_win1251("система контролю",
OutputBuf), &Font_15x25, COLOR_YELLOW, COLOR_BLACK,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 30, 60, utf8_to_win1251("за умовами
зберігання", OutputBuf), &Font_15x25, COLOR_YELLOW, COLOR_BLACK,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 50, 90, utf8_to_win1251("лікарських
засобів", OutputBuf), &Font_15x25, COLOR_YELLOW, COLOR_BLACK,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 40, 150, utf8_to_win1251("Кваліфікаційна
робота", OutputBuf), &Font_15x25, COLOR_YELLOW, COLOR_BLACK,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 40, 180, utf8_to_win1251("на здобуття
бакалавра", OutputBuf), &Font_15x25, COLOR_YELLOW, COLOR_BLACK,
LCD_SYMBOL_PRINT_FAST);
     LCD_WriteString(lcd, 50, 210, utf8_to_win1251("Віталія
Невмержицького", OutputBuf), &Font_15x25, COLOR_YELLOW, COLOR_BLACK,
LCD_SYMBOL_PRINT_FAST);

//   присвоюємо початкові значення структурам
     settingRAM = DataForSaveNew();
     packet = PacketNew();

     ave_tIn = AVARAGE_New(10);
     ave_tOut = AVARAGE_New(10);
     ave_relHIn = AVARAGE_New(10);
     ave_relHOut = AVARAGE_New(10);
     ave_absHIn = AVARAGE_New(10);
     ave_absHOut = AVARAGE_New(10);

     ave_p = AVARAGE_New(20);

//для приведення показів датчиків до одного рівня
     ave_Tsrd = AVARAGE_New(20);
     ave_Hsrd = AVARAGE_New(20);
     ave_tInK = AVARAGE_New(20);
     ave_relHInK = AVARAGE_New(20);
     ave_dT = AVARAGE_New(20);
     ave_dH = AVARAGE_New(20);

//Навігація по меню
     ScreenMode = KILCE_New(0,NUM_SCREEN-1); //збереження позиції
поточного знаходження в навігації
                                                  //тобто який з
Screens[] використовується

     Screens[0] = KILCE_New(0,NUM_INFO_SCREEN-1); //0 - основні інфо
екрани - номер екрана
     Screens[1] = KILCE_New(0,NUM_POS-1);    //1 - екран налаштувань,
вибору режима - номер позиції
```

```
                                                    //налаштування
часу, дати:
     Screens[2] = KILCE_New(0, 23);          //2 - година
     Screens[3] = KILCE_New(0, 59);          //3 - хвилина
     Screens[4] = KILCE_New(0, 31);          //4 - число
     Screens[5] = KILCE_New(0, 12);          //5 - місяць
     Screens[6] = KILCE_New(24, 99);         //6 - рік
     Screens[7] = KILCE_New(1, 7);      //7 - день тижня
     Screens[8] = KILCE_New(0, 1);      //8 - Так/Ні

     Screens[9] = KILCE_New(0, 3);      //9 - Колірна схема
     Screens[10] = KILCE_New(0, 1);          //10 - Вимкнено /
Увімкнено демонстраційний режим

//Коригування датчиків один відносно одного
     Screens[11] = KILCE_New(0, 4);     //
                                    //0: Вийти без змін
                                    //1: Внести результати вимірювань
вологості до Точки 2
                                    //2: Внести результати вимірювань
вологості до Точки 1
                                    //3: Внести результати вимірювань
температури до Точки 2
                                    //4: Внести результати вимірювань
температури до Точки 1

     Screens[12] = KILCE_New(0, 1);          //12 - Екран
"Налаштування повітряних заслінок"
     Screens[13] = KILCE_New(0, 1);          //13 - "Тип повітряних
заслінок"

//значення для повороту сервоприводів
     Screens[14] = KILCE_New(0, 255);   //14 - значення для повороту
сервопривода на 90° - відкрито (5080)
     Screens[15] = KILCE_New(0, 255);   //15 - значення для повороту
сервопривода на 0° - закрито (8000)
     Screens[NUM_SCREEN-1] = KILCE_New(0, 1);//16 - Так/Ні


     chart_dry = CHART_DRY_New();       // Структура для графіків -
інфо екрани 2 та 3. Значення за останні 24 години
     chart_history = CHART_DRY_New();   // Структура для графіків
історії за вибрану дату

     data_today = DATA_HISTORY_New();   //структура для збереження
даних за поточну добу на SD-карту
     data_history = DATA_HISTORY_New(); //структура для зчитування
даних за вказану дату з SD-карти з наступним виведенням на екран

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

```
  // ------------------------------ Монтування диска -----------------
------------
  (void)f_mount(&USERFatFS, USERPath, 0);
  // -----------------------------------------------------------------
-------------
  // ------------------------ Налаштування файлового менеджера -------
-----------------
  // Визначення колірних схем
  // Варіант 1. Світла
  File_Manager_Color_Scheme color_scheme_tc = { COLOR_DARKGREY, //
text_color_dir - колір тексту найменування каталогу

     COLOR_BLACK, // text_color_file - колір тексту найменування
файлу

     COLOR_BLUE, // text_color_cursor - колір тексту вибраного
файлу/каталогу

     COLOR_WHITE, // bg_color - колір фону вікна менеджера (для
порожніх рядків)

     COLOR_CYAN, // bg_line1_color - колір фону парних рядків
                                                          0xF5F5F5,
// bg_line2_color - колір фону непарних рядків

     COLOR_YELLOW, // cursor_color - колір курсора поточного
файлу/каталогу
                                                          0xDBDBDB,
      // slider_color - колір повзунка вертикального прокручування
                                                          0xF5F5F5,
      // stripe_color - колір смуги вертикального прокручування
                                                          1
           // fl_cursor_fill - ознака зафарбовування курсору:

           //   0 - текст вибраного файлу/каталогу обводиться

           //   прямокутником кольору курсору (cursor_color).

           //   >0 - фон тексту вибраного файлу/каталогу дорівнює

           //   кольору курсору (cursor_color)
                                                           };
  // Варіант 2. Синя
  File_Manager_Color_Scheme color_scheme_far = { COLOR_WHITE,//
text_color_dir

COLOR_CYAN, // text_color_file

COLOR_GREENYELLOW,// text_color_cursor
```

```
COLOR_BLUE, // bg_color

0x66CCCC, // bg_line1_color

0x66CC66, // bg_line2_color

0x000095,   // cursor_color

COLOR_CYAN, // slider_color

COLOR_BLUE, // stripe_color
                                                                    1};
// fl_cursor_fill
  // Варіант 3. Зелена
  File_Manager_Color_Scheme color_scheme_frigate = { COLOR_BLUE,   //
text_color_dir

0x19389E,   // text_color_file

COLOR_BLACK,// text_color_cursor

COLOR_GREEN,// bg_color

COLOR_CYAN,// bg_line1_color

0xC4C4C4,// bg_line2_color

0x4CB24C,   // cursor_color

0xA6CBEE,   // slider_color

0xF8F8F5,   // stripe_color
                                                                    1};
// fl_cursor_fill
  // Варіант 4. Темна
  File_Manager_Color_Scheme color_scheme_dc = { COLOR_CYAN,//
text_color_dir

     COLOR_GREEN,// text_color_file

     COLOR_YELLOW,// text_color_cursor

     COLOR_BLACK,// bg_color

     COLOR_DARKGREEN,// bg_line1_color

     0x6B6B6B,// bg_line2_color
                                                                    0x0A246A,
// cursor_color
```

```
                                                                0xD4D0C8,
// slider_color
                                                                0xF0F0F0,
// stripe_color
                                                                1};
// fl_cursor_fill

  // Створення масиву вказівників на колірні схеми
 // File_Manager_Color_Scheme *color_scheme[] = {&color_scheme_tc,
&color_scheme_far, &color_scheme_frigate, &color_scheme_dc};
  color_scheme[0] = &color_scheme_tc;
  color_scheme[1] = &color_scheme_far;
  color_scheme[2] = &color_scheme_frigate;
  color_scheme[3] = &color_scheme_dc;


  fm = FileManagerNew();                        // Створення обробника
файлового менеджера

  fm->SetDisplay(fm, lcd);                           // Дисплей, на який
виводиться файловий менеджер (вказівник)
  fm->SetWin(fm, 0, 0, lcd->Width, lcd->Height);       // Параметры
окна файлового менеджера:

     //  - позиція лівого верхнього кута вікна по горизонталі;

     //  - позиція лівого верхнього кута вікна по вертикалі;

     //  - ширина вікна;

     //  - висота вікна.
  fm->SetFont(fm, &Font_12x20);                    // Шрифт
(вказівник)
//  fm->SetFont(fm, &Font_15x25);                        // Шрифт
(вказівник)
  fm->SetKeys(fm, KEYB_UP, KEYB_DOWN, KEYB_RIGHT);          // Кнопки
керування:
                                                                    //
- номер біта кнопки вгору;
                                                                    //
- номер біта кнопки вниз;
                                                                    //
- номер біта кнопки введення/вибір.
  //-----------------------------------------------------------------
---------------

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
```

```
     HAL_IWDG_Refresh(&hiwdg);     // Скидання сторожового таймера
перед читанням файлу
                                        // щоб не було перевантаження
МК у випадку проблем з доступом до файлу

     readSD(); // читаємо файли з SD-карти: settingRAM.dat -
структура з параметрами settingRAM, chart_dry.dat - структура з
даними графіків chart_dry

     readDHT();     //читання параметрів з датчиків

     if(settingRAM->servo_mode == 0)    // якщо заслінка з серво-
приводом
     {
          HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_RESET);//увімкнення реле подачи живлення на сервоприводи
          Close_Servo(1);
          HAL_Delay(3000);
          Close_Servo(2);
          HAL_Delay(3000);
          HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_SET);//вимкнення реле подачи живлення на сервоприводи
     }
     else
     {
          HAL_GPIO_WritePin(PIN_SRV_PWR_GPIO_Port, PIN_SRV_PWR_Pin,
GPIO_PIN_RESET);    //увімкнення реле подачи живлення на
електромагніти
          HAL_Delay(2000);
     }

     HAL_IWDG_Refresh(&hiwdg);     // Скидання сторожового таймера

     fm->SetColor(fm, color_scheme[settingRAM->color_mode]);//
встановлюємо "Колірну схему" зчитану з SD-карти
     LCD_Fill(lcd, fm->color_scheme->bg_color);

     HEAT_OFF;
     FLAG_MOTOR_OFF;
     FLAG_MOTOR_IN_OFF;
     FLAG_MOTOR_OUT_OFF;
     HAL_GPIO_WritePin(PIN_RELAY_GPIO_Port, PIN_RELAY_Pin,
GPIO_PIN_SET);//вимкнення реле подачи живлення на вентилятори

     if(settingRAM->demo_mode == 0)     //якщо demo-режим вимкнено
     {
          TIME_PRINT_CHART = 720;            // то період виводу
однієї точки графіка = 12 хв. = 720 сек.
     }
     else //в режимі DEMO швидше: нова точка на гріфіку кожні 3 сек.
```

```
     {
          TIME_PRINT_CHART = 3;
     }


     FLAG_EKRAN_ON;                    //ознака увімкнення дисплея

     get_time(1);   // отримання актуальних дати, часу і формування
масиву TimeBuf для виводу
     char File_Name[20];
     sprintf(File_Name, "%02d-%02d-20%02d.dat", sDate.Date,
sDate.Month, sDate.Year);

     HAL_IWDG_Refresh(&hiwdg);     // Скидання сторожового таймера
перед читанням файлу
                              // щоб не було перевантаження МУ у
випадку проблем з доступом до файлу

     read_today(File_Name, 1);     //читаємо дані з файлу в структуру
data_today

     Screens[0]->Set_Poz(Screens[0], settingRAM->N_Screen);//
встановлюємо поточним інфо екран, зчитаний з SD-карти
     Goto_New_Window();  // Друк поточного інфо екрана

     while (1)
     {    if (KEYB_kbhit() == 1)                        //якщо була
натиснута будь-яка кнопка
          {    rslt = button_click_handler();              //то
запуск обробника натискання кнопок
               if(rslt == 1)                         //якщо є перехід на
новий екран,
                    Goto_New_Window();            //то відображаємо
його
          }

          if (FLAG_EKRAN_CHECK)                         //якщо дисплей
увімкнуто
          {    if (packet_ready == 1)             //і якщо було
опитування датчиків
               {    packet_ready = 0;             //скидаємо ознаку
«опитування датчиків»
                    if(ScreenMode->Get_Poz(ScreenMode) == 0)// якщо
знаходимся на якомусь інфо екрані
                    {    switch (Screens[0]->Get_Poz(Screens[0]))
                         { // оновлюємо зображення в залежності від
номера екрана
                              case 0: // якщо інформаційний екран 0
                                   if(packet_ready_E0 == 1)
                                   {    printInfoUpdate0(); // Друк
значень на екрані 0
```

```
                                        cubik();       //
зафарбовування квадрата
                                        packet_ready_E0     = 0;
                                   }
                                   break;
                              case 1: // якщо інформаційний екран 1
                                   if(packet_ready_E1 == 1)
                                   {    printInfoUpdate1(); // Друк
значень на екрані 1
                                        cubik();
                                        packet_ready_E1     = 0;
                                   }
                                   break;
                              case 2: // якщо інформаційний екран 2
                                   print_status0();    // Друк
значень
                                   dry_update(0);      // Друк
графіків
                                   cubik();
                                   break;
                              case 3: //якщо інформаційний екран 3
                                   print_status0();    // Друк
значень
                                   dry_update(1);      // Друк
графіків
                                   cubik();
                         }

                         if(packet_ready_Time == 1)    //якщо була
зміна хвилини
                         {    show_time();   //оновлення відображення
дати, часу
                              packet_ready_Time = 0;
                         }
                    }
                    else
                    {
                       if(ScreenMode->Get_Poz(ScreenMode) ==
11)//якщо на екрані "Коригування датчиків"
                       {  KorDatUpdate(); }       // Оновлення екрану
"Коригування датчиків"
                    }
               }
          }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
     }
  /* USER CODE END 3 */
}
```

```
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  LL_FLASH_SetLatency(LL_FLASH_LATENCY_2);
  while(LL_FLASH_GetLatency()!= LL_FLASH_LATENCY_2)
  {
  }
  LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_SCALE2);
  LL_RCC_HSE_Enable();

   /* Wait till HSE is ready */
  while(LL_RCC_HSE_IsReady() != 1)
  {

  }
  LL_RCC_LSI_Enable();

   /* Wait till LSI is ready */
  while(LL_RCC_LSI_IsReady() != 1)
  {

  }
  LL_PWR_EnableBkUpAccess();
  LL_RCC_LSE_Enable();

   /* Wait till LSE is ready */
  while(LL_RCC_LSE_IsReady() != 1)
  {

  }
  LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_HSE,
LL_RCC_PLLM_DIV_25, 168, LL_RCC_PLLP_DIV_2);
  LL_RCC_PLL_Enable();

   /* Wait till PLL is ready */
  while(LL_RCC_PLL_IsReady() != 1)
  {

  }
  while (LL_PWR_IsActiveFlag_VOS() == 0)
  {
  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_2);
  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);
  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);
```

```
   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
  {

  }
  LL_SetSystemCoreClock(84000000);

   /* Update the time base */
  if (HAL_InitTick (TICK_INT_PRIORITY) != HAL_OK)
  {
    Error_Handler();
  }
  LL_RCC_SetTIMPrescaler(LL_RCC_TIM_PRESCALER_TWICE);
}

/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{

  /* USER CODE BEGIN I2C1_Init 0 */

  /* USER CODE END I2C1_Init 0 */

  /* USER CODE BEGIN I2C1_Init 1 */

  /* USER CODE END I2C1_Init 1 */
  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();
  }
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_Init 2 */

}

/**
  * @brief IWDG Initialization Function
```

```
  * @param None
  * @retval None
  */
static void MX_IWDG_Init(void)
{

  /* USER CODE BEGIN IWDG_Init 0 */

  /* USER CODE END IWDG_Init 0 */

  /* USER CODE BEGIN IWDG_Init 1 */

  /* USER CODE END IWDG_Init 1 */
  hiwdg.Instance = IWDG;
  hiwdg.Init.Prescaler = IWDG_PRESCALER_128;
  hiwdg.Init.Reload = 2047;
  if (HAL_IWDG_Init(&hiwdg) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN IWDG_Init 2 */

  /* USER CODE END IWDG_Init 2 */

}

/**
  * @brief RTC Initialization Function
  * @param None
  * @retval None
  */
static void MX_RTC_Init(void)
{

  /* USER CODE BEGIN RTC_Init 0 */

  /* USER CODE END RTC_Init 0 */

//  RTC_TimeTypeDef sTime = {0};
//  RTC_DateTypeDef sDate = {0};

  /* USER CODE BEGIN RTC_Init 1 */

  /* USER CODE END RTC_Init 1 */

  /** Initialize RTC Only
  */
  hrtc.Instance = RTC;
  hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
  hrtc.Init.AsynchPrediv = 127;
  hrtc.Init.SynchPrediv = 255;
```

```
  hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
  hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
  hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
  if (HAL_RTC_Init(&hrtc) != HAL_OK)
  {
    Error_Handler();
  }

  /* USER CODE BEGIN Check_RTC_BKUP */

  /* USER CODE END Check_RTC_BKUP */

  /** Initialize RTC and set the Time and Date  */
/*
  sTime.Hours = 22;
  sTime.Minutes = 33;
  sTime.Seconds = 0;
  sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
  sTime.StoreOperation = RTC_STOREOPERATION_RESET;
  if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BIN) != HAL_OK)
  {
    Error_Handler();
  }
  sDate.WeekDay = RTC_WEEKDAY_WEDNESDAY;
  sDate.Month = RTC_MONTH_MARCH;
  sDate.Date = 27;
  sDate.Year = 24;

  if (HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BIN) != HAL_OK)
  {
    Error_Handler();
  }
*/
  /* USER CODE BEGIN RTC_Init 2 */
//калібрування годинника
HAL_RTCEx_SetCoarseCalib(&hrtc, RTC_CALIBSIGN_POSITIVE, 4);
//Value = від 0 до 63 при RTC_CALIBSIGN_NEGATIVE
//Value = від 0 до 126 при RTC_CALIBSIGN_POSITIVE

  /* USER CODE END RTC_Init 2 */

}

/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{
```

```
  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */

  LL_SPI_InitTypeDef SPI_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_SPI1);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**SPI1 GPIO Configuration
  PA5   ------> SPI1_SCK
  PA7   ------> SPI1_MOSI
  */
  GPIO_InitStruct.Pin = LCD_SCL_Pin|LCD_SDA_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_5;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* SPI1 DMA Init */

  /* SPI1_TX Init */
  LL_DMA_SetChannelSelection(DMA2, LL_DMA_STREAM_3,
LL_DMA_CHANNEL_3);

  LL_DMA_SetDataTransferDirection(DMA2, LL_DMA_STREAM_3,
LL_DMA_DIRECTION_MEMORY_TO_PERIPH);

  LL_DMA_SetStreamPriorityLevel(DMA2, LL_DMA_STREAM_3,
LL_DMA_PRIORITY_LOW);

  LL_DMA_SetMode(DMA2, LL_DMA_STREAM_3, LL_DMA_MODE_NORMAL);

  LL_DMA_SetPeriphIncMode(DMA2, LL_DMA_STREAM_3,
LL_DMA_PERIPH_NOINCREMENT);

  LL_DMA_SetMemoryIncMode(DMA2, LL_DMA_STREAM_3,
LL_DMA_MEMORY_INCREMENT);

  LL_DMA_SetPeriphSize(DMA2, LL_DMA_STREAM_3,
LL_DMA_PDATAALIGN_HALFWORD);

  LL_DMA_SetMemorySize(DMA2, LL_DMA_STREAM_3,
LL_DMA_MDATAALIGN_HALFWORD);

  LL_DMA_DisableFifoMode(DMA2, LL_DMA_STREAM_3);
```

```
  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  SPI_InitStruct.TransferDirection = LL_SPI_FULL_DUPLEX;
  SPI_InitStruct.Mode = LL_SPI_MODE_MASTER;
  SPI_InitStruct.DataWidth = LL_SPI_DATAWIDTH_16BIT;
  SPI_InitStruct.ClockPolarity = LL_SPI_POLARITY_HIGH;
  SPI_InitStruct.ClockPhase = LL_SPI_PHASE_1EDGE;
  SPI_InitStruct.NSS = LL_SPI_NSS_SOFT;
  SPI_InitStruct.BaudRate = LL_SPI_BAUDRATEPRESCALER_DIV2;
  SPI_InitStruct.BitOrder = LL_SPI_MSB_FIRST;
  SPI_InitStruct.CRCCalculation = LL_SPI_CRCCALCULATION_DISABLE;
  SPI_InitStruct.CRCPoly = 10;
  LL_SPI_Init(SPI1, &SPI_InitStruct);
  LL_SPI_SetStandard(SPI1, LL_SPI_PROTOCOL_MOTOROLA);
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

}

/**
  * @brief SPI2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI2_Init(void)
{

  /* USER CODE BEGIN SPI2_Init 0 */

  /* USER CODE END SPI2_Init 0 */

  LL_SPI_InitTypeDef SPI_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_SPI2);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);
  /**SPI2 GPIO Configuration
  PB13   ------> SPI2_SCK
  PB14   ------> SPI2_MISO
  PB15   ------> SPI2_MOSI
  */
  GPIO_InitStruct.Pin = SD_CLK_Pin|SD_MISO_Pin|SD_MOSI_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_VERY_HIGH;
```

```
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_5;
  LL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /* SPI2 DMA Init */

  /* SPI2_RX Init */
  LL_DMA_SetChannelSelection(DMA1, LL_DMA_STREAM_3,
LL_DMA_CHANNEL_0);

  LL_DMA_SetDataTransferDirection(DMA1, LL_DMA_STREAM_3,
LL_DMA_DIRECTION_PERIPH_TO_MEMORY);

  LL_DMA_SetStreamPriorityLevel(DMA1, LL_DMA_STREAM_3,
LL_DMA_PRIORITY_LOW);

  LL_DMA_SetMode(DMA1, LL_DMA_STREAM_3, LL_DMA_MODE_NORMAL);

  LL_DMA_SetPeriphIncMode(DMA1, LL_DMA_STREAM_3,
LL_DMA_PERIPH_NOINCREMENT);

  LL_DMA_SetMemoryIncMode(DMA1, LL_DMA_STREAM_3,
LL_DMA_MEMORY_INCREMENT);

  LL_DMA_SetPeriphSize(DMA1, LL_DMA_STREAM_3,
LL_DMA_PDATAALIGN_BYTE);

  LL_DMA_SetMemorySize(DMA1, LL_DMA_STREAM_3,
LL_DMA_MDATAALIGN_BYTE);

  LL_DMA_DisableFifoMode(DMA1, LL_DMA_STREAM_3);

  /* SPI2_TX Init */
  LL_DMA_SetChannelSelection(DMA1, LL_DMA_STREAM_4,
LL_DMA_CHANNEL_0);

  LL_DMA_SetDataTransferDirection(DMA1, LL_DMA_STREAM_4,
LL_DMA_DIRECTION_MEMORY_TO_PERIPH);

  LL_DMA_SetStreamPriorityLevel(DMA1, LL_DMA_STREAM_4,
LL_DMA_PRIORITY_LOW);

  LL_DMA_SetMode(DMA1, LL_DMA_STREAM_4, LL_DMA_MODE_NORMAL);

  LL_DMA_SetPeriphIncMode(DMA1, LL_DMA_STREAM_4,
LL_DMA_PERIPH_NOINCREMENT);

  LL_DMA_SetMemoryIncMode(DMA1, LL_DMA_STREAM_4,
LL_DMA_MEMORY_INCREMENT);
```

```
  LL_DMA_SetPeriphSize(DMA1, LL_DMA_STREAM_4,
LL_DMA_PDATAALIGN_BYTE);

  LL_DMA_SetMemorySize(DMA1, LL_DMA_STREAM_4,
LL_DMA_MDATAALIGN_BYTE);

  LL_DMA_DisableFifoMode(DMA1, LL_DMA_STREAM_4);

  /* USER CODE BEGIN SPI2_Init 1 */

  /* USER CODE END SPI2_Init 1 */
  /* SPI2 parameter configuration*/
  SPI_InitStruct.TransferDirection = LL_SPI_FULL_DUPLEX;
  SPI_InitStruct.Mode = LL_SPI_MODE_MASTER;
  SPI_InitStruct.DataWidth = LL_SPI_DATAWIDTH_8BIT;
  SPI_InitStruct.ClockPolarity = LL_SPI_POLARITY_LOW;
  SPI_InitStruct.ClockPhase = LL_SPI_PHASE_1EDGE;
  SPI_InitStruct.NSS = LL_SPI_NSS_SOFT;
  SPI_InitStruct.BaudRate = LL_SPI_BAUDRATEPRESCALER_DIV128;
  SPI_InitStruct.BitOrder = LL_SPI_MSB_FIRST;
  SPI_InitStruct.CRCCalculation = LL_SPI_CRCCALCULATION_DISABLE;
  SPI_InitStruct.CRCPoly = 10;
  LL_SPI_Init(SPI2, &SPI_InitStruct);
  LL_SPI_SetStandard(SPI2, LL_SPI_PROTOCOL_MOTOROLA);
  /* USER CODE BEGIN SPI2_Init 2 */

  /* USER CODE END SPI2_Init 2 */

}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 27999;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
```

```
  htim1.Init.Period = 8999;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */

}

/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 24;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 65450;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
```

```
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_2) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */
  HAL_TIM_MspPostInit(&htim2);

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  LL_TIM_InitTypeDef TIM_InitStruct = {0};
  LL_TIM_OC_InitTypeDef TIM_OC_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);
```

```
  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  TIM_InitStruct.Prescaler = 999;
  TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
  TIM_InitStruct.Autoreload = 209;
  TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
  LL_TIM_Init(TIM3, &TIM_InitStruct);
  LL_TIM_DisableARRPreload(TIM3);
  LL_TIM_OC_EnablePreload(TIM3, LL_TIM_CHANNEL_CH1);
  TIM_OC_InitStruct.OCMode = LL_TIM_OCMODE_PWM1;
  TIM_OC_InitStruct.OCState = LL_TIM_OCSTATE_DISABLE;
  TIM_OC_InitStruct.OCNState = LL_TIM_OCSTATE_DISABLE;
  TIM_OC_InitStruct.CompareValue = 109;
  TIM_OC_InitStruct.OCPolarity = LL_TIM_OCPOLARITY_HIGH;
  LL_TIM_OC_Init(TIM3, LL_TIM_CHANNEL_CH1, &TIM_OC_InitStruct);
  LL_TIM_OC_DisableFast(TIM3, LL_TIM_CHANNEL_CH1);
  LL_TIM_SetTriggerOutput(TIM3, LL_TIM_TRGO_RESET);
  LL_TIM_DisableMasterSlaveMode(TIM3);
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**TIM3 GPIO Configuration
  PA6   ------> TIM3_CH1
  */
  GPIO_InitStruct.Pin = LCD_BLK_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_2;
  LL_GPIO_Init(LCD_BLK_GPIO_Port, &GPIO_InitStruct);

}

/**
  * @brief TIM10 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM10_Init(void)
{

  /* USER CODE BEGIN TIM10_Init 0 */

  /* USER CODE END TIM10_Init 0 */

  LL_TIM_InitTypeDef TIM_InitStruct = {0};
```

```
  /* Peripheral clock enable */
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_TIM10);

  /* TIM10 interrupt Init */
  NVIC_SetPriority(TIM1_UP_TIM10_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),2, 0));
  NVIC_EnableIRQ(TIM1_UP_TIM10_IRQn);

  /* USER CODE BEGIN TIM10_Init 1 */

  /* USER CODE END TIM10_Init 1 */
  TIM_InitStruct.Prescaler = 999;
  TIM_InitStruct.CounterMode = LL_TIM_COUNTERMODE_UP;
  TIM_InitStruct.Autoreload = 419;
  TIM_InitStruct.ClockDivision = LL_TIM_CLOCKDIVISION_DIV1;
  LL_TIM_Init(TIM10, &TIM_InitStruct);
  LL_TIM_DisableARRPreload(TIM10);
  /* USER CODE BEGIN TIM10_Init 2 */

  /* USER CODE END TIM10_Init 2 */

}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{

  /* Init with LL driver */
  /* DMA controller clock enable */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_DMA2);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_DMA1);

  /* DMA interrupt init */
  /* DMA1_Stream3_IRQn interrupt configuration */
  NVIC_SetPriority(DMA1_Stream3_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
  NVIC_EnableIRQ(DMA1_Stream3_IRQn);
  /* DMA1_Stream4_IRQn interrupt configuration */
  NVIC_SetPriority(DMA1_Stream4_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
  NVIC_EnableIRQ(DMA1_Stream4_IRQn);
  /* DMA2_Stream3_IRQn interrupt configuration */
  NVIC_SetPriority(DMA2_Stream3_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
  NVIC_EnableIRQ(DMA2_Stream3_IRQn);

}

/**
```

```
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOH);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

  /**/
  LL_GPIO_SetOutputPin(SD_LED_GPIO_Port, SD_LED_Pin);

  /**/
  LL_GPIO_SetOutputPin(GPIOA, LCD_DC_Pin|LCD_RES_Pin|LCD_CS_Pin);

  /**/
  LL_GPIO_SetOutputPin(GPIOB,
PIN_RELAY_Pin|PIN_SRV_PWR_Pin|SD_CS_Pin);

  /**/
  LL_GPIO_ResetOutputPin(PIN_HEAT_GPIO_Port, PIN_HEAT_Pin);

  /**/
  GPIO_InitStruct.Pin = SD_LED_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(SD_LED_GPIO_Port, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin = LCD_DC_Pin|LCD_RES_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_HIGH;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin = LCD_CS_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_HIGH;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
```

```
  LL_GPIO_Init(LCD_CS_GPIO_Port, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin = PIN_HEAT_Pin|PIN_RELAY_Pin|PIN_SRV_PWR_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin = SD_CS_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_HIGH;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
  LL_GPIO_Init(SD_CS_GPIO_Port, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin =
KEY_LEFT_Pin|KEY_RIGHT_Pin|KEY_UP_Pin|KEY_DOWN_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
  LL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error
return state */
  __disable_irq();
  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,
     ex: printf("Wrong parameters value: file %s on line %d\r\n",
file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```